INTERNATIONAL STANDARD ISO/IEC 9075-5:1999
TECHNICAL CORRIGENDUM 2

Published 2003-06-01

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

Information technology — Database languages — SQL —

Part 5:
Host Language Bindings (SQL/Bindings)

TECHNICAL CORRIGENDUM 2

Technologies de l'information — Langages de base de données — SQL —

Partie 5: Liants de langage d'hôte (SQL/Liants)

RECTIFICATIF TECHNIQUE 2

Technical Corrigendum 2 to ISO/IEC 9075-5:1999 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*. ISO/IEC 9075-5:1999/Cor. 2:2003 cancels and replaces ISO/IEC 9075-5:1999/Cor. 1:2000.

Statement of purpose for rationale:

A statement indicating the rationale for each change to ISO/IEC 9075 is included. This is to inform the users of that standard as to the reason why it was judged necessary to change the original wording. In many cases the reason is editorial or to clarify the wording; in some cases it is to correct an error or an omission in the original wording.

Notes on numbering:

Where this Corrigendum introduces new Syntax, Access, General and Conformance Rules, the new rules have been numbered as follows:
Rules inserted between, for example, Rules 7) and 8) are numbered 7.1), 7.2), etc. [or 7) a.1), 7) a.2), etc.]. Those inserted before Rule 1) are numbered 0.1), 0.2), etc.
Where this Corrigendum introduces new Subclauses, the new subclauses have been numbered as follows:
Subclauses inserted between, for example, Subclause 4.3.2 and 4.3.3 are numbered 4.3.2a, 4.3.2b, etc.
Those inserted before, for example, 4.3.1 are numbered 4.3.0, 4.3.0a, etc.

ICS 35.060 **Ref. No. ISO/IEC 9075-5:1999/Cor.2:2003(E)**

© ISO/IEC 2003 – All rights reserved

Published in Switzerland

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Contents

Page

4.6.1 Classes of SQL-statements . . . 3
4.6.4 Embeddable SQL-statement . . . 3
4.6.5 Preparable and immediately executable SQL-statements . . . 4
4.6.6 Directly executable SQL-statements . . . 4
5.1 <token> and <separator> . . . 5
8.1 <routine invocation> . . . 5
10.5 <SQL-invoked routine> . . . 6
11.1 <SQL-client module definition> . . . 6
11.2 Calls to an <externally-invoked procedure> . . . 6
11.3 <SQL procedure statement> . . . 6
12.0 <fetch statement> . . . 7
12.1 <select statement: single row> . . . 8
12.2 <free locator statement> . . . 8
14.3 <set names statement> . . . 8
15.1 Description of SQL descriptor areas . . . 9
15.2 <allocate descriptor statement> . . . 9
15.3 <deallocate descriptor statement> . . . 9
15.4 <get descriptor statement> . . . 10
15.5 <set descriptor statement> . . . 10
15.6 <prepare statement> . . . 10
15.8 <describe statement> . . . 11
15.9 <input using clause> . . . 11
15.10 <output using clause> . . . 12
15.13 <dynamic declare cursor> . . . 12
15.14 <allocate cursor statement> . . . 12
16.1 <embedded SQL host program> . . . 13
16.2 <embedded exception declaration> . . . 16
16.3 <embedded SQL Ada program> . . . 16
16.4 <embedded SQL C program> . . . 17
16.5 <embedded SQL COBOL program> . . . 18
16.6 <embedded SQL FORTRAN program> . . . 19
16.7 <embedded SQL MUMPS program> . . . 19
16.8 <embedded SQL Pascal program> . . . 20
16.9 <embedded SQL PL/I program> . . . 21
17.1 <direct SQL statement> . . . 22
17.2 <direct select statement: multiple rows> . . . 22
18.1 <get diagnostics statement> . . . 23
20.1 SQLSTATE . . . 23
Annex A SQL Conformance Summary . . . 24
Annex B Implementation-defined elements . . . 26
Annex E Incompatibilities with ISO/IEC 9075:1992 . . . 27
Annex F SQL feature and package taxonomy . . . 27

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

© ISO/IEC 2003 – All rights reserved

Information technology — Database languages — SQL —

Part 5:
Host Language Bindings (SQL/Bindings)

TECHNICAL CORRIGENDUM 2

4.6.1 Classes of SQL-statements

1. Rationale: Clarify the semantics of SQL-data access indication.

Replace the 2nd paragraph with:

Insert this paragraph There are at least four additional ways of classifying SQL-statements:

— According to whether or not they may be embedded.

— According to whether they may be dynamically prepared and executed.

— According to whether or not they may be directly executed.

— According to whether they do not possibly contain SQL, possibly contain SQL, possibly read SQL-data, or possibly modify SQL-data.

4.6.4 Embeddable SQL-statement

1. Rationale: Correct the classification of SQL-statements.

Insert the following sub-bullet to the 7th bullet of the 1st paragraph:

- <hold locator statement>

2. Rationale: Correct the classification of SQL-statements.

Replace the 8th bullet of the 1st paragraph with:

— The following SQL-control statements:

- <call statement>
- <return statement>

--`,,`-`-`,,`,,`,`,,`---

© ISO/IEC 2003 - All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4.6.5 Preparable and immediately executable SQL-statements

1. Rationale: Correct the classification of SQL-statements.

Delete the following sub-bullet from the 4th bullet of the 1st paragraph:

- <free locator statement>

2. Rationale: Correct the classification of SQL-statements.

Insert the following bullet to the 2nd paragraph:

— <return statement>

4.6.6 Directly executable SQL-statements

1. Rationale: Correct the classification of SQL-statements.

Insert the following bullet to the 1st paragraph:

— The following SQL-control statements:

- <call statement>
- <return statement.

2. Rationale: Clarify the semantics of SQL-data access indication.

Insert the following Subclause after Subclause 4.6.6, "Directly executable SQL-statements":

4.6.6a SQL-statements and SQL-data access indication

Insert this paragraph The following are the other SQL-statements that possibly contain SQL:

— SQL embedded exception declaration

Insert this paragraph The following are the other SQL-statements that possibly read SQL-data:

— SQL-dynamic statements

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

© ISO/IEC 2003 – All rights reserved

Not for Resale

5.1 <token> and <separator>

1. Rationale: Editorial - Correct reserved and non-reserved word lists.

In the Format, in the production for <non-reserved word> add the alternatives:

```
| NESTING
| SCOPE_CATALOG
| SCOPE_NAME
| SCOPE_SCHEMA
| USER_DEFINED_TYPE_CATALOG
| USER_DEFINED_TYPE_NAME
| USER_DEFINED_TYPE_SCHEMA
```

2. Rationale: Editorial. Correct reserved word list.

In the Format, in the production for <reserved word>, delete the texts:

```
| DYNAMIC
| NESTING
```

8.1 <routine invocation>

1. Rationale: <Embedded variable specification> has also to be handled according to the Syntax Rules of Subclause 9.1, "Retrieval assignment" in Bindings.

Replace Syntax Rule 1 with:

1) Replace SR 8) c) i) 4) A) If A_i is a <host parameter specification> or an <embedded variable specification>, then P_i shall be assignable to A_i, according to the Syntax Rules of Subclause 9.1, ''Retrieval assignment'', with A_i and P_i as *TARGET* and *VALUE*, respectively.

2. Rationale: The current handling of output parameter in routine invocation is incomplete. It does not cover all alternatives of <target specification>.

Insert the following General Rule:

1) Replace GR 10) b) i) If TS_i is a <host parameter specification> or an <embedded variable specification>, then CPV_i is assigned to TS_i according to the rules of Subclause 9.1, ''Retrieval assignment''.

© ISO/IEC 2003 - All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.5 <SQL-invoked routine>

1. Rationale: Clarify the semantics of SQL-data access indication.

Replace Syntax Rule 1) with:

1) Insert before SR 18) c) It is implementation-defined whether the <SQL routine body> shall not contain an <SQL dynamic statement>.

11.1 <SQL-client module definition>

1. Rationale: Consistent use of terminology.

Replace General Rule 1) with:

1) Augments GR 5) After the last time that an SQL-agent performs a call of an <externally-invoked procedure>, following the effective execution of a <rollback statement> or a <commit statement>, a <deallocate descriptor statement> that specifies

```
DEALLOCATE DESCRIPTOR D
```

is effectively executed, where *D* is the <descriptor name> of any SQL descriptor area that is currently allocated within an SQL-session associated with the SQL-agent.

11.2 Calls to an <externally-invoked procedure>

1. Rationale: Editorial.

In Syntax Rule 1) replace the following constraints:

```
DYNAMIC_SQL_ERROR_UNDEFINED_DATA_TARGET:
     constant SQLSTATE_TYPE := "0700D";
DYNAMIC_SQL_ERROR_UNDEFINED_LEVEL_VALUE:
     constant SQLSTATE_TYPE := "0700E";
```

with:

```
DYNAMIC_SQL_ERROR_INVALID_DATA_TARGET:
     constant SQLSTATE_TYPE := "0700D";
DYNAMIC_SQL_ERROR_INVALID_LEVEL_VALUE:
     constant SQLSTATE_TYPE := "0700E";
```

11.3 <SQL procedure statement>

1. Rationale: Editorial.

In the Format, replace the production for <SQL procedure statement> with:

```
<SQL session statement> ::=
       !! All alternatives from ISO/IEC 9075-2
     | <set catalog statement>
     | <set schema statement>
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale
© ISO/IEC 2003 – All rights reserved

```
      ¦ <set names statement>
      ¦ <set path statement>
      ¦ <set transform group statement>
```

2. *Rationale: Consistent use of terminology.*

In the Format replace the production of <SQL dynamic statement> with:

```
<SQL dynamic statement> ::=
    <SQL descriptor statement>
  ¦ <prepare statement>
  ¦ <deallocate prepared statement>
  ¦ <describe statement>
  ¦ <execute statement>
  ¦ <execute immediate statement>
  ¦ <SQL dynamic data statement>
```

In the Format replace the production of <system descriptor statement> with:

```
<SQL descriptor statement> ::=
    <allocate descriptor statement>
  ¦ <deallocate descriptor statement>
  ¦ <set descriptor statement>
  ¦ <get descriptor statement>
```

12.0 <fetch statement>

1. *Rationale: Add missing Syntax and General Rules for <fetch statement>.*

Add a new Subclause as follows:

12.0 <fetch statement>

Function

Position a cursor on a specified row of a table and retrieve values from that row.

Format

No additional Format items.

Syntax Rules

1) Add after SR 6) b) iii) For each <target specification> *TS2* that is an <embedded variable name>, the Syntax Rules of Subclause 9.1, "Retrieval assignment", apply to each *TS2* and the corresponding column of table *T*, as *TARGET* and *VALUE*, respectively.

General Rules

1) Add after GR 7) b) ii) If *TV* is an <embedded variable name>, then the General Rules of Subclause 9.1, "Retrieval assignment" are applied to *TV* and *SV*, as *TARGET* and *VALUE*, respectively.

© ISO/IEC 2003 - All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12.1 <select statement: single row>

1. Rationale: Replace incorrect non-terminal.

Replace Syntax Rule 1).

1) Insert after SR4) For each <target specification> *TS* that is an <embedded variable specification>, then the Syntax Rules of Subclause 9.1, "Retrieval assignment", shall apply to *TS* and the corresponding element of the <select list>, as *TARGET* and *VALUE*, respectively.

2. Rationale: Remove redundant and incorrect rule.

Delete General Rule 1).

3. Rationale: Replace incorrect non-terminal.

Replace General Rule 2).

2) Insert after GR5) For each <target specification> TS that is an <embedded variable specification>, the corresponding value in the row of *Q* is assigned to *TS* according to the General Rules of Subclause 9.1, "Retrieval assignment", as *VALUE* and *TARGET*, respectively. The assignment of values to targets in the <select target list> is in an implementation-dependent order.

12.2 <free locator statement>

1. Rationale: Editorial - Typographical error.

In the Format, replace the production for <locator reference> with:

```
<locator reference> ::=
       !! All alternatives from ISO/IEC 9075-2
     | <embedded variable name>
```

14.3 <set names statement>

1. Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".

Replace Conformance Rule 1) with:

1) Without Feature F761, "Session management" and Feature F461, "Named character sets", conforming SQL language shall not contain any <set names statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale
© ISO/IEC 2003 – All rights reserved

15.1 Description of SQL descriptor areas

1. Rationale: Correct definition of the length of <reference type>s.

Replace Syntax Rule 6) n) with:

6) n) TYPE indicates REF, LENGTH is the length in octets for the REF type, and USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are a valid qualified user-defined type name, and SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME are a valid qualified table name.

15.2 <allocate descriptor statement>

1. Rationale: Consistent use of terminology.

Replace General Rule 2) with:

2) Case:

a) If an SQL descriptor area whose name is *V* and whose scope is specified by the <scope option> immediately contained in <descriptor name> is already currently allocated, then an exception condition is raised: *invalid SQL descriptor name*.

b) Otherwise, <allocate descriptor statement> allocates an SQL descriptor area whose name is *V* and whose scope is specified by the <scope option> immediately contained in <descriptor name>. The SQL descriptor area will have at least <occurrences> number of SQL item descriptor areas. The value of LEVEL in each of the item descriptor areas is set to 0 (zero). The values of all other fields in the SQL descriptor area are initially undefined.

15.3 <deallocate descriptor statement>

1. Rationale: Consistent use of terminology.

Replace General Rule 1) with:

1) Case:

a) If an SQL descriptor area is not currently allocated whose name is the value of the <simple value specification> immediately contained in <descriptor name> and whose scope is specified by the <scope option> immediately contained in <descriptor name>, then an exception condition is raised: *invalid SQL descriptor name*.

b) Otherwise, <deallocate descriptor statement> deallocates an SQL descriptor area whose name is the value of the <simple value specification> immediately contained in <descriptor name> and whose scope is specified by the <scope option> immediately contained in <descriptor name>.

© ISO/IEC 2003 - All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.4 <get descriptor statement>

1. Rationale: Consistent use of terminology.

Replace General Rule 1) with:

1) If a <descriptor name> identifies an SQL descriptor area that is not currently allocated whose name is the value of the <simple value specification> immediately contained in <descriptor name> and whose scope is specified by the <scope option> immediately contained in <descriptor name>, then an exception condition is raised: *invalid SQL descriptor name*.

15.5 <set descriptor statement>

1. Rationale: Consistent use of terminology.

Replace General Rule 1) with:

1) If a <descriptor name> identifies an SQL descriptor area that is not currently allocated whose name is the value of the <simple value specification> immediately contained in <descriptor name> and whose scope is specified by the <scope option> immediately contained in <descriptor name>, then an exception condition is raised: *invalid SQL descriptor name*.

15.6 <prepare statement>

1. Rationale: handle <dynamic parameter specification>s for <regular expression substring function>.

Insert the following General Rule:

6) a) vii.1) If *DP* is either *X1*, *X2* or *X3* in a <string value function> of the form "SUBSTRING (*X1* SIMILAR *X2* ESCAPE *X3*)" then

1) Case:

a) If the declared type of *X1* is CHARACTER, CHARACTER VARYING or CHARACTER LARGE OBJECT, then let *CS* be the character set of *X1*.

b) If the declared type of *X2* is CHARACTER, CHARACTER VARYING or CHARACTER LARGE OBJECT, then let *CS* be the character set of *X1*.

c) If the declared type of *X3* is CHARACTER, CHARACTER VARYING or CHARACTER LARGE OBJECT, then let *CS* be the character set of *X1*.

d) Otherwise, the character set *CS* is undefined

2) If *CS* is defined, then

a) If *DP* is *X1* or *X2*, then *DT* is CHARACTER VARYING (*ML*) with character set *CS*.

b) If *DP* is *X3*, then *DT* is CHARACTER (1) with character set *CS*.

--`,,`-`-`,,`,,`,`,,`---
© ISO/IEC 2003 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2. *Rationale: Use correct keywords for parameter modes.*

Replace General Rule 7) a) iii) with:

7) a) iii) For each <dynamic parameter specification> *D* contained in some <SQL argument> A_k, 1 (one) ≤ *k* ≤ *n*:

1) *D* is an input <dynamic parameter specification> if the <SQL parameter mode> of the *k*-th SQL parameter of *SR* is IN or INOUT.

2) *D* is an output <dynamic parameter specification> if the <SQL parameter mode> of the *k*-th SQL parameter of *SR* is OUT or INOUT.

15.8 <describe statement>

1. *Rationale: Consistent use of terminology.*

Replace General Rule 4) with:

4) If an SQL descriptor area is not currently allocated whose name is the value of the <simple value specification> immediately contained in <descriptor name> and whose scope is that specified by the <scope option> immediately contained in <descriptor name> , then an exception condition is raised: *invalid SQL descriptor name.*

2. *Rationale: Correct definition of the length of <reference type>s.*

Replace General Rule 8) d) ix) with:

8) d) ix) If TYPE indicates a <reference type>, then USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME are set to the <user-defined type name> of the referenced type and qualified name of the referenceable base table; LENGTH and OCTET_LENGTH are set to the length in octets of the <reference type>.

15.9 <input using clause>

1. *Rationale: Consistent use of terminology.*

Replace General Rule 1) with:

1) If <using input descriptor> is specified and an SQL descriptor area is not currently allocated whose name is the value of the <simple value specification> and whose scope is that specified by the <scope option> immediately contained in <descriptor name> immediately contained in <descriptor name>, then an exception condition is raised: *invalid SQL descriptor name.*

2. *Rationale: Editorial.*

Replace Conformance Rule 2) with:

2) Without Feature B031, ''Basic dynamic SQL'', conforming SQL language shall not contain any <input using clause>.

© ISO/IEC 2003 - All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.10 <output using clause>

1. Rationale: There is no syntax rule to contain either a <host parameter specification> or an <embedded variable specification>.

Insert the following Syntax Rule:

1) The <target specification> immediately contained in <into argument> shall be either a <host parameter specification> or an <embedded variable specification>.

2. Rationale: Consistent use of terminology.

Replace General Rule 1) with:

1) If <into descriptor> is specified and an SQL descriptor area is not currently allocated whose name is the value of the <simple value specification> immediately contained in <descriptor name> and whose scope is that specified by the <scope option> immediately contained in <descriptor name>, then an exception condition is raised: *invalid SQL descriptor name*.

15.13 <dynamic declare cursor>

1. Rationale: Editorial - typographical errors.

Replace Conformance Rule 6) with:

6) Without Feature B031, "Basic dynamic SQL", and Feature F431 "Read-only scrollable cursors", and Feature F831, "Full cursor update", if an <updatability clause> of FOR UPDATE with or without a <column name list> is specified, then <cursor scrollability> shall not be specified.

2. Rationale: Editorial - typographical errors.

Replace Conformance Rule 8) with:

8) Without Feature B031, "Basic dynamic SQL", and Feature F431 "Read-only scrollable cursors", a <dynamic declare cursor> shall not specify <cursor scrollability>.

15.14 <allocate cursor statement>

1. Rationale: Ensure that the cursor is positioned in same place in returned result set as last set in the called procedure.

Replace General Rule 4) g) with:

4) g) Cursor *CR* is placed in the open state

Case:

i) If *CR* is scrollable then, let *CRCN* be the <cursor name> of *CR* in *P*. The position of *CR* in *T* is before the row that would be retrieved if the following SQL-statement were executed in *P*:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale
© ISO/IEC 2003 – All rights reserved

```
FETCH NEXT FROM CRCN INTO ...
```

ii) Otherwise, the position of *CR* is before the first row of *T*.

16.1 <embedded SQL host program>

1. Rationale: Delete redundant syntax alternative.

In the Format, replace the production for <statement or declaration> with:

```
<statement or declaration> ::=
       <declare cursor>
     ¦ <dynamic declare cursor>
     ¦ <temporary table declaration>
     ¦ <embedded authorization declaration>
     ¦ <embedded path specification>
     ¦ <embedded transform group specification>
     ¦ <embedded exception declaration>
     ¦ <handler declaration>
     ¦ <SQL procedure statement>
```

2. Rationale: Correct use of undefined BNF term.

Replace Syntax Rule 9) with:

9) Case:

a) If <embedded transform group specification> is not specified, then an <embedded transform group specification> containing a <multiple group specification> with a <group specification> *GS* for each <host variable definition> that has an associated user-defined type *UDT*, but is not a user-defined locator variable is implicit. The <group name> of *GS* is implementation-defined and its <user-defined type> is *UDT*.

b) If <embedded transform group specification> contains a <single group specification> with a <group name> *GN*, then an <embedded transform group specification> containing a <multiple group specification> with a <group specification> *GS* for each <host variable definition> that has an associated user-defined type *UDT*, but is not a user-defined type locator variable is implicit. The <group name> of *GS* is *GN* and its <user-defined type> is *UDT*.

c) If <embedded transform group specification> contains a <multiple group specification> *MGS*, then an <embedded transform group specification> containing a <multiple group specification> that contains *MGS* extended with a <group specification> *GS* for each <host variable definition> that has an associated user-defined type *UDT*, but is not a user-defined locator variable and the <user-defined type name> of *UDT* is not contained in any <group specification> contained in *MGS* is implicit. The <group name> of *GS* is implementation-defined and its <user-defined type> is *UDT*.

3. Rationale: Correct use of undefined BNF term.

Replace Syntax Rule 10) with:

10) The implicit or explicit <embedded transform group specification> precedes in the text of the <embedded SQL host program> every <host variable definition>.

© ISO/IEC 2003 - All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4. *Rationale: Remove redundant rules that refer to a non-existent BNF term.*

Delete Syntax Rules 15) and 16).

5. *Rationale: Clarify assignable and comparable.*

Replace Syntax Rule 22) h) ii) 1) C) with:

22) h) ii) 1) C) Let the declared type of the single SQL parameter of *TSF* be *TPT*. *PT* shall be assignable to *TPT*.

Replace Syntax Rule 22) k) i) 6) H) with:

22) k) i) 6) H) For every j, 1 (one) $\leq j \leq a$, apply the Syntax Rules of Subclause 10.17, "Determination of a to-sql function", with TUI_j and GNI_j as TYPE and GROUP, respectively. There shall be an applicable to-sql function $TSFI_j$ identified by <routine name> $TSIN_j$. Let TTI_j be the data type of the single SQL parameter of $TSFI_j$. TSI_j shall be assignable to TTI_j.

Replace Syntax Rule 22) k) i) 6) I) with:

22) k) i) 6) I) For every l, 1 (one) $\leq l \leq c$, apply the Syntax Rules of Subclause 10.17, "Determination of a to-sql function", with $TUIO_l$ and $GNIO_l$ as TYPE and Subclause 7.9, "<group by clause>", respectively. There shall be an applicable to-sql function $TSFIO_l$ identified by <routine name> $TSION_l$. Let $TTIO_l$ be the data type of the single SQL parameter of $TSFIO_l$. $TSIO_l$ shall be assignable to $TTIO_l$.

Replace Syntax Rule 22) k) i) 6) J) with:

22) k) i) 6) J) For every k, 1 (one)$\leq k \leq b$, apply the Syntax Rules of Subclause 10.15, "Determination of a from-sql function", with TUO_k and GNO_k as TYPE and GROUP, respectively. There shall be an applicable from-sql function $FSFO_k$ identified by <routine name> $FSON_k$. Let TRO_k be the result data type of $FSFO_k$. TRO_k shall be assignable to TSO_k.

Replace Syntax Rule 22) k) i) 6) K) with:

22) k) i) 6) K) For every l, 1 (one) $\leq l \leq c$, apply the Syntax Rules of Subclause 10.15, "Determination of a from-sql function", with $TUIO_l$ and $GNIO_l$ as TYPE and GROUP, respectively. There shall be an applicable from-sql function $FSFIO_l$ identified by <routine name> $FSION_l$. Let $TRIO_l$ be the result data type of $FSFIO_l$. $TRIO_l$ shall be assignable to $TSIO_l$.

6. *Rationale: Editorial.*

Replace Syntax Rule 22) k) i) 6) G) with:

22) k) i) 6) G) Let GNI_j, 1 (one) $\leq j \leq a$, be the <group name>s corresponding to the <user-defined type name> of TUI_j contained in the <group specification> contained in <embedded transform group specification>. Let GNO_k, 1 (one) $\leq k \leq b$, be the <group name>s corresponding to the <user-defined type name> of TUO_k contained in the <group specification> contained in <embedded transform group specification>. Let $GNIO_l$, 1 (one)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
© ISO/IEC 2003 – All rights reserved
Not for Resale

≤ *l* ≤ *c*, be the <group name>s corresponding to the <user-defined type name> of $TUIO_l$ contained in the <group specification> contained in <embedded transform group specification>.

7. Rationale: Editorial.

Replace Syntax Rule 22) k) i) 8) with:

22) k) i) 8) The <SQL procedure statement> of *PS* is:

```
BEGIN ATOMIC
    DECLARE SVI1 TUI1 ;
    .
    .
    .
    DECLARE SVIa TUIa ;
    DECLARE SVO1 TUO1 ;
    .
    .
    .
    DECLARE SVOb TUOb ;
    DECLARE SVIO1 TUIO1 ;
    .
    .
    .
    DECLARE SVIOc TUIOc ;
    SET SVI1 = TSIN1 ( CAST ( PNI1 AS TTI1 ));
    .
    .
    .
    SET SVIa = TSINa ( CAST ( PNIa AS TTIa ));
    SET SVIO1 = TSION1 ( CAST ( PNIO1 AS TTIO1 ));
    .
    .
    .
    SET SVIOc = TSIONc (CAST ( PNIOc AS TTIOc ));
    NES;
    SET PNO1 = CAST ( FSON1 ( SVO1 )AS TSO1 );
    .
    .
    .
    SET PNOb = CAST ( FSONb ( SVOb )AS TSOb );
    SET PNIO1 = CAST ( FSION1 ( SVIO1 )AS TSIO1 );
    .
    .
    .
    SET PNIOc = CAST ( FSIONc ( SVIOc )AS TSIOc );
END;
```

8. Rationale: Remove rule referring to a non-existent feature.

Delete General Rule 2).

9. Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".

Replace Conformance Rule 2) with:

2) Without Feature F461, "Named character sets", an <embedded SQL declare section> shall not contain

© ISO/IEC 2003 - All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

an <embedded character set declaration>.

16.2 <embedded exception declaration>

1. Rationale: Remove an incorrect Conformance Rule.

Delete Conformance Rule 2).

16.3 <embedded SQL Ada program>

1. Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).

In the Format, replace the production for <Ada user-defined type locator variable> with:

```
<Ada user-defined type locator variable> ::=
     SQL TYPE IS <user-defined type> AS LOCATOR
```

2. Rationale: Correct definition of the length of <reference type>s.

Replace Syntax Rule 5) i) with:

5) i) The syntax

```
SQL TYPE IS <reference type>
```

for a given <Ada host identifier> RTV shall be replaced by

```
RTV : Interfaces.SQL.CHAR(1..<length>)
```

in any <Ada REF variable>, where <length> is the length in octets of the <reference type>.

3. Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".

Replace Conformance Rule 3) with:

3) Without Feature F461, "Named character sets", an <Ada type specification> shall not contain a <character set specification>.

4. Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).

Replace Conformance Rule 7) with:

7) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in an <Ada user-defined type locator variable> shall not identify a structured type.

© ISO/IEC 2003 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

16.4 <embedded SQL C program>

1. Rationale: Correct definition of the length of <reference type>s.

Replace Syntax Rule 5) n) with:

5) n) The syntax

```
SQL TYPE IS <reference type>
```

for a given <C host identifier> *hvn* shall be replaced by

```
unsigned char hvn[L]
```

in any <C REF variable>, where *L* is the length in octets of the <reference type>.

2. Rationale: Editorial.

Replace Syntax Rule 6) g) with:

6) g) The syntax

SQL TYPE IS NCLOB (*L*)

for a given <C host identifier> *hvn* shall be replaced by

```
struct {
  long          hvn_reserved;
  unsigned long hvn_length;
  char          hvn_data[L];
  } hvn
```

in any <C NCLOB variable>, where *L* is the numeric value of <large object length> as specified in Subclause 5.2, "<token> and <separator>", in ISO/IEC 9075-2.

3. Rationale: Standardise terminology.

Replace Syntax Rule 11) with:

11) In a <C character variable>, a <C VARCHAR variable>, or a <C CLOB variable>, if a <character set specification> is specified, then the equivalent SQL data type is CHARACTER, CHARACTER VARYING, or CHARACTER LARGE OBJECT whose character set is the same as the set specified by the <character set specification>. In a <C NCHAR variable>, a <C NCHAR VARYING variable>, or a <C NCLOB variable>, if a <character set specification> is specified, then the equivalent SQL data type is NATIONAL CHARACTER, NATIONAL CHARACTER VARYING, or NATIONAL CHARACTER LARGE OBJECT whose character set is the same as the set specified by the <character set specification>. If <character set specification> is not specified, then an implementation-defined <character set specification> is implicit.

--`,,`-`-`,,`,,`,`,,`---

© ISO/IEC 2003 - All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".*

Replace Conformance Rule 3) with:

3) Without Feature F461, "Named character sets", a <C variable definition> shall not contain a <character set specification>.

5. *Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).*

Replace Conformance Rule 7) with:

7) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in an <C user-defined type locator variable> shall not identify a structured type.

16.5 <embedded SQL COBOL program>

1. *Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).*

In the Format, replace the production for <COBOL user-defined type locator variable> with:

```
<COBOL user-defined type locator variable> ::=
     [ USAGE [ IS ] ] SQL TYPE IS <user-defined type> AS LOCATOR
```

2. *Rationale: Correct definition of the length of <reference type>s.*

Replace Syntax Rule 6) k) with:

6) k) The syntax

```
SQL TYPE IS <reference type>
```

for a given <COBOL host identifier> *HVN* shall be replaced by

```
01 HVN PICTURE X(L)
```

in any <COBOL REF variable>, where *L* is the length in octets of the <reference type>.

3. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".*

Replace Conformance Rule 3) with:

3) Without Feature F461, "Named character sets", a <COBOL type specification> shall not contain a <character set specification>.

4. *Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).*

Replace Conformance Rule 7) with:

7) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <COBOL user-defined type locator variable> shall not identify a structured type.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale
© ISO/IEC 2003 – All rights reserved

16.6 <embedded SQL FORTRAN program>

1. Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).

In the Format, replace the production for <Fortran user-defined type locator variable> with:

```
<Fortran user-defined type locator variable> ::=
     SQL TYPE IS <user-defined type> AS LOCATOR
```

2. Rationale: Correct definition of the length of <reference type>s and the BNF term <Fortran REF variable>.

Replace Syntax Rule 6) k) with:

6) k) The syntax

```
SQL TYPE IS <reference type>
```

for a given <Fortran host identifier> *HVN* shall be replaced by

```
CHARACTER HVN *<length>
```

in any <Fortran REF variable>, where <length> is the length in octets of the <reference type>.

3. Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".

Replace Conformance Rule 3) with:

3) 3) Without Feature F461, "Named character sets", a <Fortran type specification> shall not contain a <character set specification>.

4. Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).

Replace Conformance Rule 7) with:

7) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <Fortran user-defined type locator variable> shall not identify a structured type.

16.7 <embedded SQL MUMPS program>

1. Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).

In the Format, replace the production for <MUMPS user-defined type locator variable> with:

```
<MUMPS user-defined type locator variable> ::=
     SQL TYPE IS <user-defined type> AS LOCATOR
```

2. Rationale: Correct definition of the length of <reference type>s and the BNF term <MUMPS REF variable>.

Replace Syntax Rule 9) h) with:

© ISO/IEC 2003 - All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9) h) The syntax

```
SQL TYPE IS <reference type>
```

for a given <MUMPS host identifier> *HVN* shall be replaced by

```
VARCHAR HVN L
```

in any <MUMPS REF variable>, where *L* is the length in octets of the <reference type>.

3. *Rationale: Add missing conformance rule.*

Add Conformance Rule 7):

7) Without Feature F461, "Named character sets", a <MUMPS CLOB variable> shall not contain a <character set specification>.

4. *Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).*

Replace Conformance Rule 5) with:

7) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <MUMPS user-defined type locator variable> shall not identify a structured type.

16.8 <embedded SQL Pascal program>

1. *Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).*

In the Format, replace the production for <Pascal user-defined type locator variable> with:

```
<Pascal user-defined type locator variable> ::=
     SQL TYPE IS <user-defined type> AS LOCATOR
```

2. *Rationale: Correct definition of the length of <reference type>s and the BNF term <Pascal REF variable>.*

Replace Syntax Rule 5) l) with:

5) l) The syntax

```
SQL TYPE IS <reference type>
```

for a given <Pascal host identifier> *HVN* shall be replaced by

```
HVN : PACKED ARRAY [1..<length>] of CHAR
```

in any <Pascal REF variable>, where <length> is the length in octets of the <reference type>.

3. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".*

Replace Conformance Rule 3) with:

--`,,`-`-`,,`,,`,`,,`---
© ISO/IEC 2003 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3) Without Feature F461, "Named character sets", a <Pascal type specification> shall not contain a <character set specification>.

4. *Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).*

Replace Conformance Rule 7) with:

7) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <Pascal user-defined type locator variable> shall not identify a structured type.

16.9 <embedded SQL PL/I program>

1. *Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).*

In the Format, replace the production for <PL/I user-defined type locator variable> with:

```
<PL/I user-defined type locator variable> ::=
     SQL TYPE IS <user-defined type> AS LOCATOR
```

2. *Rationale: Correct definition of the length of <reference type>s and the BNF term <PL/I REF variable>.*

Replace Syntax Rule 5) i) with:

5) i) The syntax

```
SQL TYPE IS <reference type>
```

for a given <PL/I host identifier> *HVN* shall be replaced by

```
DCL HVN CHARACTER(<length>) VARYING
```

in any <PL/I REF variable>, where <length> is the length in octets of the <reference type>.

3. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".*

Replace Conformance Rule 3) with:

3) Without Feature F461, "Named character sets", a <PL/I type specification> shall not contain a <character set specification>.

4. *Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).*

Replace Conformance Rule 7) with:

7) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <PL/I user-defined type locator variable> shall not identify a structured type.

© ISO/IEC 2003 - All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

17.1 <direct SQL statement>

1. Rationale: Consistent use of terminology.

Replace General Rule 2) b) with:

2) b) Let *D* be the <descriptor name> of any SQL descriptor area that is currently allocated within the current SQL-session. A <deallocate descriptor statement> that specifies

```
DEALLOCATE DESCRIPTOR D
```

is effectively executed.

2. Rationale: Address requirement for multiple diagnostics areas

Replace General Rule 6) a) i) with:

6) a) i) The first diagnostics area is emptied.

Replace General Rule 6) b) v) with:

6) b) v) The first diagnostics area is emptied.

Replace General Rule 9) with:

9) Diagnostics information resulting from the execution of *S* is placed into the first diagnostics area, causing the first condition area in the first diagnostics area to become occupied.

17.2 <direct select statement: multiple rows>

1. Rationale: align the <order by clause> with <declare cursor>

Replace the Format with the following:

```
<direct select statement: multiple rows> ::=
  <cursor specification>
```

Delete Syntax Rule 1).

Replace Syntax Rule 2) with:

2) The <query expression> or <order by clause> of a <direct select statement: multiple rows> shall not contain any <value specification> other than a <literal>, CURRENT_USER, CURRENT_ROLE, SESSION_USER, SYSTEM_USER, CURRENT_PATH , CURRENT_DEFAULT_TRANSFORM_GROUP, or CURRENT_TRANSFORM_GROUP_FOR_TYPE.

2.1) The <cursor specification> shall not contain an <updatability clause>.

Delete Syntax Rules 3) and 4).

Delete General Rule 1).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
© ISO/IEC 2003 – All rights reserved
Not for Resale

Replace General Rule 2) with:

2) Let *Q* be the result of the <cursor specification>.

Delete General Rules 4) and 5)

18.1 <get diagnostics statement>

1. Rationale: Editorial. Non-reserved word - DYNAMIC_FUNCTION - not defined in Part 2.

Insert the following General Rule 3):

3) Replace GR 3) p) The values of CONNECTION_NAME and SERVER_NAME are respectively

Case:

i) If COMMAND_FUNCTION or DYNAMIC_FUNCTION identifies an <SQL connection statement>, then the <connection name> and the <SQL-server name> specified by or implied by the <SQL connection statement>.

ii) Otherwise, the <connection name> and <SQL-server name> of the SQL-session in which the condition was raised.

2. Rationale: Correct the SQL-statement codes entry for SET PATH

In Table 9, "SQL-statement codes", replace the row for <set path statement> with:

SQL-statement	Identifier	Code
<set path statement>	SET PATH	69

20.1 SQLSTATE

1. Rationale: Editorial.

In “Table 10—SQLSTATE class and subclass values”, replace the rows:

Category	Condition	Class	Subcondition	Subclass
			undefined DATA target	00D
			undefined LEVEL value	0

with:

Category	Condition	Class	Subcondition	Subclass
			invalid DATA target	00D
			invalid LEVEL value	0

© ISO/IEC 2003 - All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex A SQL Conformance Summary

1. Rationale: Editorial - typographical errors.

Replace Item 9) o) ii) with:

9) o) ii) Without Feature B031, "Basic dynamic SQL", and Feature F431 "Read-only scrollable cursors", a <dynamic declare cursor> shall not specify <cursor scrollability>.

2. Rationale: Editorial - typographical errors.

Replace Item 9) o) iv) with:

9) o) iv) Without Feature B031, "Basic dynamic SQL", and Feature F431 "Read-only scrollable cursors", and Feature F831, "Full cursor update", if an <updatability clause> of FOR UPDATE with or without a <column name list> is specified, then <cursor scrollability> shall not be specified.

3. Rationale: Editorial - typographical errors.

Replace Item 13) with:

13) Specifications for Feature F431 "Read-only scrollable cursors":

a) Subclause 15.13, "<dynamic declare cursor>":

i) Without Feature B031, "Basic dynamic SQL", and Feature F431 "Read-only scrollable cursors", and Feature F831, "Full cursor update", if an <updatability clause> of FOR UPDATE with or without a <column name list> is specified, then <cursor scrollability> shall not be specified.

4. Rationale: Editorial - typographical errors.

Replace Item 15) with:

15) Specifications for Feature F431 "Read-only scrollable cursors":

a) Subclause 15.13, "<dynamic declare cursor>":

i) Without Feature B031, "Basic dynamic SQL", and Feature F431 "Read-only scrollable cursors", a <dynamic declare cursor> shall not specify <cursor scrollability>.

5. Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".

Replace Item 16) with:

16) Specifications for Feature F451, "Character set definition":

a) Subclause 14.3, "<set names statement>":

--`,,`-`-`,,`,,`,`,,`---
© ISO/IEC 2003 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Without Feature F761, "Session management" and Feature F461, "Named character sets", conforming SQL language shall not contain any <set names statement>.

b) Subclause 16.1, "<embedded SQL host program>":

i) Without Feature F461, "Named character sets", an <embedded SQL declare section> shall not contain an <embedded character set declaration>.

c) Subclause 16.3, "<embedded SQL Ada program>":

i) Without Feature F461, "Named character sets", an <Ada type specification> shall not contain a <character set specification>.

d) Subclause 16.4, "<embedded SQL C program>":

i) Without Feature F461, "Named character sets", a <C variable definition> shall not contain a <character set specification>.

e) Subclause 16.5, "<embedded SQL COBOL program>":

i) Without Feature F461, "Named character sets", a <COBOL type specification> shall not contain a <character set specification>.

f) Subclause 16.6, "<embedded SQL Fortran program>":

i) Without Feature F461, "Named character sets", a <Fortran type specification> shall not contain a <character set specification>.

g) Subclause 16.7, "<embedded SQL MUMPS program>":

i) Without Feature F461, "Named character sets", a <MUMPS CLOB variable> shall not contain a <character set specification>.

h) Subclause 16.8, "<embedded SQL Pascal program>":

i) Without Feature F461, "Named character sets", a <Pascal type specification> shall not contain a <character set specification>.

i) Subclause 16.9, "<embedded SQL PL/I program>":

i) Without Feature F461, "Named character sets", a <PL/I type specification> shall not contain a <character set specification>.

6. *Rationale: Use the correct BNF (<user-defined type name> instead of <user-defined type>).*

Replace Item 26) with:

26) Specifications for Feature S231, "Structured type locators":

a) Subclause 16.3, "<embedded SQL Ada program>":

i) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in

© ISO/IEC 2003 - All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

an <Ada user-defined type locator variable> shall not identify a structured type.

b) Subclause 16.4, "<embedded SQL C program>":

i) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <C user-defined type locator variable> shall not identify a structured type.

c) Subclause 16.5, "<embedded SQL COBOL program>":

i) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <COBOL user-defined type locator variable> shall not identify a structured type.

d) Subclause 16.6, "<embedded SQL Fortran program>":

i) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <Fortran user-defined type locator variable> shall not identify a structured type.

e) Subclause 16.7, "<embedded SQL MUMPS program>":

i) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <MUMPS user-defined type locator variable> shall not identify a structured type.

f) Subclause 16.8, "<embedded SQL Pascal program>":

i) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <Pascal user-defined type locator variable> shall not identify a structured type.

g) Subclause 16.9, "<embedded SQL PL/I program>":

i) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <PL/I user-defined type locator variable> shall not identify a structured type.

Annex B Implementation-defined elements

1. Rationale: Clarify the distinction between character sets and character repertoires.

Insert the following Item:

17.1) Subclause 16.4, "<embedded SQL C program>":

a) The implicit character set in a <C character variable>, a <C VARCHAR variable>, or a <C CLOB variable> is implementation-defined.

© ISO/IEC 2003 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex E Incompatibilities with ISO/IEC 9075:1992

1. Rationale: Editorial.

Replace Point 2) with:

2) A number of additional <reserved word>s have been added to the language. These <reserved word>s are:

— CURRENT_DEFAULT_TRANSFORM_GROUP

— CURRENT_TRANSFORM_GROUP_FOR_TYPE

Annex F SQL feature and package taxonomy

1. Rationale: Correct a footnote.

Replace footnote 1 in Table 12, “Feature taxonomy and definition for Core SQL”, with:

1 A conforming SQL-implementation is required (by Clause 8, "Conformance", in ISO/IEC 9075-1) to support at least one embedded language, or the SQL-client module binding for at least one host language.

© ISO/IEC 2003 - All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

INTERNATIONAL STANDARD ISO/IEC 9075-5:1999/Amd.1:2001
TECHNICAL CORRIGENDUM 1

Published 2003-06-01

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

Information technology — Database languages — SQL —

Part 5:
Host Language Bindings (SQL/Bindings)

AMENDMENT 1: On-Line Analytical Processing (SQL/OLAP)

TECHNICAL CORRIGENDUM 1

Technologies de l'information — Langages de base de données — SQL —

Partie 5: Liants de langage d'hôte (SQL/Liants)

AMENDEMENT 1: Traitement analytique en ligne (SQL/OLAP)

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to ISO/IEC 9075-5:1999/Amd.1:2001 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

ICS 35.060 **Ref. No. ISO/IEC 9075-5:1999/Amd.1:2001/Cor.1:2003(E)**

© ISO/IEC 2003 – All rights reserved

Published in Switzerland

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Contents

Page

Introduction 3
3.3.2.1 Clause, Subclause, and Table relationships 3
4.1.1 Operations involving numbers 3
5.1 <token> and <separator> 4
6.1 <set function specification> 4
6.2 <numeric value function> 5
6.4 <value expression> 7
7.0 <table reference> 7
7.5 <window clause> 8
7.6 <query specification> 8
8.1 <aggregate function> 8
13 Information Schema 10
Annex B Implementation-defined elements 10
Annex F Incompatibilities with ISO/IEC 9075:1992 10

--`,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale
© ISO/IEC 2003 – All rights reserved

Information technology — Database languages — SQL —

Part 5:
Host Language Bindings (SQL/Bindings)

AMENDMENT 1

TECHNICAL CORRIGENDUM 1

Introduction

1. Rationale: Add description of new Annex.

Insert the following point:

21) Annex F, ''Incompatibilities with ISO/IEC 9075:1992'', is an informative Annex. It lists incompatibilities with the previous version of ISO/IEC 9075.

3.3.2.1 Clause, Subclause, and Table relationships

1. Rationale: Add relationship for new Annex.

Insert the following row into Table 1 "Clause, Subclause, and Table relationships":

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Annex F, ''Incompatibilities with ISO/IEC 9075:1992''	*(none)*	*(none)*

4.1.1 Operations involving numbers

1. Rationale: Editorial.

Replace the 7th bullet of the 1st paragraph with:

— <width bucket function> is a function of four arguments, returning an integer between 0 (zero) and the value of the final argument plus 1 (one), by assigning the first argument to an equi-width partitioning of the range of numbers between the second and third arguments. Values outside the range between the second and third arguments are assigned to either 0 (zero) or the value of the final argument plus 1 (one).
NOTE 2 – The semantics of <width bucket function> are illustrated in Figure 1, ''Illustration of WIDTH_BUCKET Semantics''.

© ISO/IEC 2003 - All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5.1 <token> and <separator>

1. Rationale: Correct the BNF of <non-reserved word> and <reserved word>.

Replace the productions for <nonreserved word> and <reserved word> with:

```
<non-reserved word> ::=
  !! All alternatives from ISO/IEC 9075-2
| !! All alternatives from ISO/IEC 9075-5
| EXCLUDE
| FOLLOWING
| NULLS
| OTHERS
| PRECEDING
| TIES
| UNBOUNDED

<reserved word> ::=
    !! All alternatives from ISO/IEC 9075-2
  | !! All alternatives from ISO/IEC 9075-5
  | CEIL | CEILING | CORR | COVAR_POP | COVAR_SAMP | CUME_DIST
  | DENSE_RANK
  | EXP
  | FILTER | FLOOR
  | LN
  | OVER
  | PARTITION | PERCENTILE_CONT | PERCENTILE_DISC | PERCENT_RANK | POWER
  | RANGE | RANK | REGR_AVGX | REGR_AVGY | REGR_COUNT | REGR_INTERCEPT
  | REGR_R2 | REGR_SLOPE | REGR_SXX | REGR_SXY | REGR_SYY | ROW_NUMBER
  | SQRT | STDDEV_POP | STDDEV_SAMP
  | VAR_POP | VAR_SAMP
  | WIDTH_BUCKET | WINDOW | WITHIN
```

6.1 <set function specification>

1. Rationale: The definition of aggregated argument must overlay the definition in ISO/IEC 9075-2:1999 introduced by a correction elsewhere in this TC.

Replace Syntax Rule 5) with:

5) Replace SR 3.1) A <value expression> *VE* simply contained in a <set function specification> *SFE* is an aggregated argument of *SFE* if either *SFE* is not an <ordered set function> or *VE* is simply contained in a <within group specification>; otherwise, *VE* is a non-aggregated argument of *SFE*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
© ISO/IEC 2003 – All rights reserved
Not for Resale

6.2 <numeric value function>

1) Rationale: Clarify the declared type of <numeric value function>

Replace Syntax Rules 1), 2) and 3) with:

1) Insert this SR The declared type of the result of <natural logarithm> is an implementation-defined approximate numeric type.

2) Insert this SR The declared type of the result of <exponential function> is an implementation-defined approximate numeric type.

3) Insert this SR The declared type of the result of <power function> is an implementation-defined approximate numeric type.

2) Rationale: Correct the result data type of <floor function> and <ceiling function>.

Replace Syntax Rule 5) with:

5) Insert this SR If <floor function> is specified, then

Case:

a) If the declared type of the simply contained <numeric value expression> *NVE* is exact numeric, then the declared type of the result is exact numeric with implementation-defined precision, with the radix of *NVE*, and with scale 0 (zero).

b) Otherwise, the declared type of the result is approximate numeric with implementation-defined precision.

3) Rationale: Correct the result data type of <floor function> and <ceiling function>.

Replace Syntax Rule 6) with:

6) Insert this SR If <ceiling function> is specified, then

Case:

a) If the declared type of the simply contained <numeric value expression> *NVE* is exact numeric, then the declared type of the result is exact numeric with implementation-defined precision, with the radix of *NVE*, and with scale 0 (zero).

b) Otherwise, the declared type of the result is approximate numeric with implementation-defined precision.

4) Rationale: Correct the result data type of <floor function> and <ceiling function>.

Replace General Rule 4) with:

4) Insert this GR If <floor function> is specified, then let *V* be the value of the simply contained <numeric value expression> *NVE*.

© ISO/IEC 2003 - All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Case:

a) If *V* is the null value, then the result is the null value.

b) Otherwise,

Case:

i) If the most specific type of *NVE* is exact numeric, then the result is the greatest exact numeric value with scale 0 (zero) that is less than or equal to *V*. If this result is not representable by the result data type, then an exception condition is raised: *data exception — numeric value out of range*.

ii) Otherwise, the result is the greatest whole number that is less than or equal to *V*. If this result is not representable by the result data type, then an exception condition is raised: *data exception — numeric value out of range*.

5. Rationale: Correct the result data type of <floor function> and <ceiling function>.

Replace General Rule 5) with:

5) Insert this GR If <ceiling function> is specified, then let *V* be the value of the simply contained <numeric value expression> *NVE*.

Case:

a) If *V* is the null value, then the result is the null value.

b) Otherwise,

Case:

i) If the most specific type of *NVE* is exact numeric, then the result is the least exact numeric value with scale 0 (zero) that is greater than or equal to *V*. If this result is not representable by the result data type, then an exception condition is raised: *data exception — numeric value out of range*.

ii) Otherwise, the result is the least whole number that is greater than or equal to *V*. If this result is not representable by the result data type, then an exception condition is raised: *data exception — numeric value out of range*.

6. Rationale: Editorial.

Replace General Rule 6) e) iii) with:

6) e) iii) Otherwise, the result is the greatest exact numeric value with scale 0 (zero) that is less than or equal to ((*WBC* * (*WBB1* - *WBO*) / (*WBB1* - *WBB2*)) + 1).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

© ISO/IEC 2003 – All rights reserved

6.4 <value expression>

1. Rationale: Define possibly non-deterministic <value expression>.

Insert the following Syntax Rules:

2) Replace SR 9) f) An <aggregate function> that specifies MIN or MAX and that simply contains a <value expression> whose declared type is character string, user-defined type, or datetime with time zone.

3) Insert after SR 9) g) A <windowed table function> that specifies ROW_NUMBER or whose associated <window specification> specifies ROWS.

7.0 <table reference>

1. Rationale: The scope of a <table reference> must include <window clause>

Insert the following Subclause

7.0 <table reference>

Function

Reference a table

Format

No additional format items

Syntax Rules

1) Replaces SR 3)a) If a <table reference> *TR* is contained in a <from clause> *FC* with no intervening <query expression>, then the *scope clause SC* of *TR* is the <select statement: single row> or innermost <query specification> that contains *FC*. The scope of the exposed <correlation name> or exposed <table or query name> of *TR* is the <select list>, <where clause>, <group by clause>, <having clause>, and <window clause> of *SC*, together with every <lateral derived table> that is simply contained in *FC* and is preceded by *TR*, and every <collection derived table> that is simply contained in *FC* and is preceded by *TR*, and the <join condition> of all <joined table>s contained in *SC* that contain *TR*. If *SC* is the <query specification> that is the <query expression body> of a simple table query *STQ*, then the scope of the exposed <correlation name> or exposed <table or query name> also includes the <order by clause> of *STQ*.

General Rules

No additional General Rules

Conformance Rules

No additional Conformance Rules

© ISO/IEC 2003 - All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.5 <window clause>

1. Rationale: Define missing symbol.

Replace Syntax Rule 4) with:

4) Each <column reference> contained in the <window partition clause> or <window order clause> of *WDEF* shall unambiguously reference a column of the derived table *T* that is the result of *TE*. A column referenced in a <window partition clause> is a *partitioning column*.
NOTE 7 — If *T* is a grouped table, then the <column reference>s contained in <window partition clause> or <window order clause> must reference columns of the grouped table obtained by performing the syntactic transformation in Subclause 7.6, ''<query specification>''.

7.6 <query specification>

1. Rationale: Correct symbol.

Replace Syntax Rule 3) m) with:

4) m) If *N2* = 0 (zero), then let *CRL* be a zero-length string; otherwise, let *CRL* be:

$$CR_1 \text{ AS } CRI_2, CR_2 \text{ AS } CRI_2, \ldots, CR_{N2} \text{ AS } CRI_{N2}$$

2. Rationale: Correct the definition of possibly non-deterministic.

Replace Syntax Rule 4 with:

4) Insert after SR 11) d) The <window clause> contains a reference to a column *C* of *T* that has a data type of character string, user-defined type, TIME WITH TIME ZONE, or TIMESTAMP WITH TIME ZONE, and the functional dependency $G \twoheadrightarrow C$, where *G* is the set consisting of the grouping columns of *T*, holds in *T*.

8.1 <aggregate function>

1. Rationale: Clarify the declared type of numeric set functions

Replace Syntax Rule 6) with:

6) If COUNT is specified, then the declared type of the result is an implementation-defined exact numeric type with scale of 0 (zero).

Replace Syntax Rule 7) g) with:

7) g) If SUM or AVG is specified, then:

i) *DT* shall be a numeric type or an interval type.

ii) If SUM is specified and *DT* is exact numeric with scale *S*, then the declared type of the result is an implementation-defined exact numeric type with scale *S*.

--`,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale
© ISO/IEC 2003 – All rights reserved

iii) If AVG is specified and *DT* is exact numeric, then the declared type of the result is an implementation-defined exact numeric type with precision not less than the precision of *DT* and scale not less than the scale of *DT*.

iv) If *DT* is approximate numeric, then the declared type of the result is an implementation-defined approximate numeric type with precision not less than the precision of *DT*.

v) If *DT* is interval, then the declared type of the result is interval with the same precision as *DT*.

Replace Syntax Rule 7) h) with:

7) h) If VAR_POP or VAR_SAMP is specified, then the declared type of the result is an implementation-defined approximate numeric type. If *DT* is an approximate numeric type, then the precision of the result is not less than the precision of *DT* .

Replace Syntax Rule 9) c) with:

9) c) Case:

i) The declared type of REGR_COUNT is an implementation-defined exact numeric type with scale of 0 (zero).

ii) Otherwise, the declared type of the result is an implementation-defined approximate numeric type. If the declared type *DTDVE* of *DVE* is an approximate numeric type, then the precision of the result is not less than the precision of *DTDVE*. If the declared type *DTIVE* of *I'VE* is an approximate numeric type, then the precision of the result is not less than the precision of *DTIVE*.

2. *Rationale: SQL-implementations might need to raise exception conditions during the computation of <aggregate function>s.*

Insert the following General Rule:

0) If during the computation of the result of *AF* an intermediate result is not representable in the declared type of the intermediate result, then an exception condition is raised: *data exception — numeric value out of range.*

3. *Rationale: Correct set operator.*

Replace General Rule 5) g) with:

5) g) The result is the result of the <scalar subquery>

```
( SELECT WIFTVAL
    FROM ( SELECT MARKER, WIFT() OVER ( ORDER BY WSP1, ... , WSPK )
             FROM ( SELECT 0, SK1, ... , SKK
                      FROM TNAME
                    UNION ALL
                    VALUES ( 1, VE1, ... , VEK )
                  ) AS TXNAME ( MARKER, CN1, ... , CNK )
         ) AS TEMPTABLE ( MARKER, WIFTVAL )
  WHERE MARKER = 1 )
```

© ISO/IEC 2003 - All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

13 Information Schema

1. Rationale: Clarify the semantics of built-in functions.

Delete the entire Clause.

Annex B Implementation-defined elements

1. Rationale: Correct the result data type of <floor function> and <ceiling function>. Clarify the declared type of numeric expressions

Replace item 2) with:

2) Insert this list element Subclause 6.2, ''<numeric value function>'':

a) The declared type of the result of <natural logarithm> is an implementation-defined approximate numeric type.

b) The declared type of the result of <exponential function> is an implementation-defined approximate numeric type.

c) The declared type of the result of <power function> is an implementation-defined approximate numeric type.

d) The declared type of the result of <floor function> is exact numeric with implementation-defined precision and scale 0 (zero) if the declared type of the simply contained <numeric value expression> is exact numeric; otherwise the declared type of the result is approximate numeric with implementation-defined precision.

e) The declared type of the result of <ceiling function> is exact numeric with implementation-defined precision and scale 0 (zero) if the declared type of the simply contained <numeric value expression> is exact numeric; otherwise the declared type of the result is approximate numeric with implementation-defined precision.

Annex F Incompatibilities with ISO/IEC 9075:1992

1. Rationale: Correct the BNF of <non-reserved word> and <reserved word>.

Insert the following Annex:

Annex F Incompatibilities with ISO/IEC 9075:1992

This part of ISO/IEC 9075 introduces some incompatibilities with the earlier version of Database Language SQL as specified in ISO/IEC 9075:1992. Unless specified in this Annex, features and capabilities of Database Language SQL are compatible with the earlier version of ISO/IEC 9075:1992.

1) A number of additional <reserved word>s have been added to the language. These <reserved word>s are:

– CEIL

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale
© ISO/IEC 2003 – All rights reserved

— CEILING

— CORR

— COVAR_POP

— COVAR_SAMP

— CUME_DIST

— DENSE_RANK

— EXP

— FILTER

— FLOOR

— LN

— OVER

— PARTITION

— PERCENT_RANK

— PERCENTILE_CONT

— PERCENTILE_DISC

— POWER

— RANGE

— RANK

— REGR_AVGX

— REGR_AVGY

— REGR_COUNT

— REGR_INTERCEPT

— REGR_R2

— REGR_SLOPE

— REGR_SXX

— REGR_SXY

— REGR_SYY

© ISO/IEC 2003 - All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

— ROW_NUMBER

— SQRT

— STDDEV_POP

— STDDEV_SAMP

— VAR_POP

— VAR_SAMP

— WIDTH_BUCKET

— WINDOW

— WITHIN

© ISO/IEC 2003 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

INTERNATIONAL STANDARD

ISO/IEC 9075-1
9075-2
9075-5

First edition
1999-12-01

AMENDMENT 1
2001-03-15

Information technology — Database languages — SQL —

Part 1:
Framework (SQL/Framework)
Part 2:
Foundation (SQL/Foundation)
Part 5:
Host Language Bindings (SQL/Bindings)

AMENDMENT 1: On-Line Analytical Processing (SQL/OLAP)

Technologies de l'information — Langages de base de données — SQL —

Partie 1: Charpente (SQL/Charpente)

Partie 2: Fondations (SQL/Fondations)

Partie 5: Liants de langage d'hôte (SQL/Liants)

AMENDEMENT 1: Traitement analytique en ligne (SQL/OLAP)

Reference number
ISO/IEC 9075 (parts 1, 2 and 5):1999/Amd1:2001(E)

© ISO/IEC 2001

--`,,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO/IEC 2001

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

© ISO/IEC 2001 – All rights reserved

Not for Resale

Contents

Page

Foreword ... vii

Introduction ... viii

1 Scope ... 1

2 Normative references ... 3

3 Definitions, notations, and conventions ... 5
3.1 Definitions ... 5
3.2 Notation ... 5
3.3 Conventions ... 5
3.3.1 Use of terms ... 5
3.3.1.1 Syntactic containment ... 5
3.3.2 Relationships to other parts of ISO/IEC 9075 ... 5
3.3.2.1 Clause, Subclause, and Table relationships ... 5

4 Concepts ... 9
4.1 Numbers ... 9
4.1.1 Operations involving numbers ... 9
4.2 Tables ... 10
4.2.1 Windowed tables ... 10
4.3 Data analysis operations (involving tables) ... 11
4.3.1 Group functions ... 11
4.3.2 Window functions ... 12
4.3.3 Aggregate functions ... 13

5 Lexical elements ... 17
5.1 <token> and <separator> ... 17
5.2 Names and identifiers ... 19

6 Scalar expressions ... 21
6.1 <set function specification> ... 21
6.2 <numeric value function> ... 23
6.3 <window function> ... 27
6.4 <value expression> ... 31

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7 Query expressions . . . 33
7.1 <table expression> . . . 33
7.2 <joined table> . . . 34
7.3 <where clause> . . . 35
7.4 <having clause> . . . 36
7.5 <window clause> . . . 37
7.6 <query specification> . . . 47

8 Additional common elements . . . 51
8.1 <aggregate function> . . . 51
8.2 <sort specification list> . . . 61

9 Schema definition and manipulation . . . 63
9.1 <drop table constraint definition> . . . 63
9.2 <drop user-defined ordering statement> . . . 64

10 SQL-client modules . . . 65
10.1 Calls to an <externally-invoked procedure> . . . 65

11 Data manipulation . . . 67
11.1 <declare cursor> . . . 67
11.2 <select statement: single row> . . . 68

12 Dynamic SQL . . . 69
12.1 <prepare statement> . . . 69

13 Information Schema . . . 71
13.1 Definition of SQL built-in functions . . . 71

14 Status codes . . . 73
14.1 SQLSTATE . . . 73

15 Conformance . . . 75
15.1 General conformance requirements . . . 75

Annex A SQL conformance summary . . . 77

Annex B Implementation-defined elements . . . 81

Annex C Implementation-dependent elements . . . 85

Annex D SQL feature and package taxonomy . . . 87

Annex E SQL Packages . . . 89
E.1 OLAP . . . 89

Index . . . 91

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

FIGURES

Figure		Page
1	Illustration of WIDTH_BUCKET Semantics	9

© ISO/IEC 2001 – All rights reserved

--`,,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

TABLES

Tables **Page**

1 Clause, Subclause, and Table relationships 5
2 SQLSTATE class and subclass values 73
3 Implied feature relationships 75
4 SQL/OLAP feature taxonomy for features outside Core SQL 87
5 SQL Packages 89

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this Amendment may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 1 to parts 1, 2 and 5 of ISO/IEC 9075:1999 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Introduction

The organization of this Amendment is as follows:

1) Clause 1, "Scope", specifies the scope of this Amendment.

2) Clause 2, "Normative references", identifies additional standards that, through reference in this Amendment, constitute provisions of this Amendment.

3) Clause 3, "Definitions, notations, and conventions", defines the notations and conventions used in this Amendment.

4) Clause 4, "Concepts", presents concepts used in the definition of On-Line Analytical Processing facilities.

5) Clause 5, "Lexical elements", defines a number of lexical elements used in the definition of On-Line Analytical Processing facilities.

6) Clause 6, "Scalar expressions", defines a number of scalar expressions used in the definition of On-Line Analytical Processing facilities.

7) Clause 7, "Query expressions", defines the elements of the language that produce rows and tables of data as used in On-Line Analytical Processing facilities.

8) Clause 8, "Additional common elements", defines additional common elements used in the definition of On-Line Analytical Processing facilities.

9) Clause 9, "Schema definition and manipulation", defines the schema definition and manipulation statements associated with the definition of On-Line Analytical Processing facilities.

10) Clause 10, "SQL-client modules", defines SQL-client modules and externally-invoked procedures.

11) Clause 11, "Data manipulation", defines data manipulation operations associated with On-Line Analytical Processing facilities.

12) Clause 12, "Dynamic SQL", defines the SQL dynamic statements.

13) Clause 13, "Information Schema", defines viewed tables that contain schema information related to On-Line Analytical Processing facilities.

14) Clause 14, "Status codes", defines SQLSTATE values related to On-Line Analytical Processing facilities.

15) Clause 15, "Conformance", defines the criteria for conformance to this Amendment.

16) Annex A, "SQL conformance summary", is an informative Annex. It summarizes the conformance requirements of the SQL language.

17) Annex B, "Implementation-defined elements", is an informative Annex. It lists those features for which the body of this Amendment states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or any other behavior is partly or wholly implementation-defined.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

18) Annex C, "Implementation-dependent elements", is an informative Annex. It lists those features for which the body of this Amendment states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or any other behavior is partly or wholly implementation-dependent.

19) Annex D, "SQL feature and package taxonomy", is an informative Annex. It identifies features of the SQL language specified in this Amendment by a numeric identifier and a short descriptive name. This taxonomy is used to specify conformance to Core SQL and may be used to develop other profiles involving the SQL language.

20) Annex E, "SQL Packages", is an informative Annex. It specifies a package of SQL language features.

In the text of this Amendment, Clauses begin a new odd-numbered page, and in Clause 5, "Lexical elements", through Clause 15, "Conformance", Subclauses begin a new page. Any resulting blank space is not significant.

© ISO/IEC 2001 – All rights reserved

--`,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Information technology — Database languages — SQL —

Part 1: Framework (SQL/Framework)
Part 2: Foundation (SQL/Foundation)
Part 5: Host Language Bindings (SQL/Bindings)

AMENDMENT 1:
On-Line Analytical Processing (SQL/OLAP)

1 Scope

This Amendment specifies the syntax and semantics of database language facilities that support on-line analytical processing.

The database language facilities that support on-line analytical processing include:

— Rank functions.

— Distribution functions.

— Inverse distribution functions (percentiles).

— Hypothetical set functions.

— Cumulative and other forms of moving aggregates.

— Variance, standard deviation, covariance, correlation, and linear regression functions.

This amendment also incidentally defines several new numeric functions.

NOTE 1 – The context for this Amendment is described by the Reference Model of Data Management (ISO/IEC 10032:1993).

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this Amendment. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this Amendment are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 9075-1:1999, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework).*

ISO/IEC 9075-2:1999, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation).*

ISO/IEC 9075-5:1999, *Information technology — Database languages — SQL — Part 5: Host Language Bindings (SQL/Bindings).*

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3 Definitions, notations, and conventions

3.1 Definitions

Insert this paragraph For the purposes of this Amendment, the definitions given in ISO/IEC 9075-1 and ISO/IEC 9075-2 apply.

3.2 Notation

Insert this paragraph The syntax notation used in this Amendment is an extended version of BNF ("Backus Normal Form" or "Backus Naur Form"). This version of BNF is fully described in Subclause 3.2, "Notation", of ISO/IEC 9075-1.

3.3 Conventions

Insert this paragraph Except as otherwise specified in this Amendment the conventions used in this Amendment are identical to those described in ISO/IEC 9075-1 and ISO/IEC 9075-2.

3.3.1 Use of terms

3.3.1.1 Syntactic containment

Replace 2nd paragraph *A1 directly contains B1* if *A1* contains *B1* without an intervening <subquery>, <within group specification>, or <set function specification> that is not an <ordered set function>.

3.3.2 Relationships to other parts of ISO/IEC 9075

3.3.2.1 Clause, Subclause, and Table relationships

Table 1—Clause, Subclause, and Table relationships

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Clause 1, “Scope”	Clause 1, "Scope"	ISO/IEC 9075-2
Clause 2, “Normative references”	Clause 2, "Normative references"	ISO/IEC 9075-2
Clause 3, “Definitions, notations, and conventions”	Clause 3, "Definitions, notations, and conventions"	ISO/IEC 9075-2
Subclause 3.1, “Definitions”	Subclause 3.1, "Definitions"	ISO/IEC 9075-2
Subclause 3.2, “Notation”	Subclause 3.2, "Notation"	ISO/IEC 9075-2

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3.3 Conventions

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 3.3, “Conventions”	Subclause 3.3, "Conventions"	ISO/IEC 9075-2
Subclause 3.3.1, “Use of terms”	Subclause 3.3.1, "Use of terms"	ISO/IEC 9075-2
Subclause 3.3.1.1, “Syntactic containment”	Subclause 3.3.1.1, "Syntactic containment"	ISO/IEC 9075-2
Subclause 3.3.2, “Relationships to other parts of ISO/IEC 9075”	*(none)*	*(none)*
Subclause 3.3.2.1, “Clause, Subclause, and Table relationships”	*(none)*	*(none)*
Clause 4, “Concepts”	Clause 4, "Concepts"	ISO/IEC 9075-2
Subclause 4.1, “Numbers”	Subclause 4.5, "Numbers"	ISO/IEC 9075-2
Subclause 4.1.1, “Operations involving numbers”	Subclause 4.5.2, "Operations involving numbers"	ISO/IEC 9075-2
Subclause 4.2, “Tables”	Subclause 4.16, "Tables"	ISO/IEC 9075-2
Subclause 4.2.1, “Windowed tables”	*(none)*	*(none)*
Subclause 4.3, “Data analysis operations (involving tables)”	*(none)*	*(none)*
Subclause 4.3.1, “Group functions”	*(none)*	*(none)*
Subclause 4.3.2, “Window functions”	*(none)*	*(none)*
Subclause 4.3.3, “Aggregate functions”	*(none)*	*(none)*
Clause 5, “Lexical elements”	Clause 5, "Lexical elements"	ISO/IEC 9075-2
Subclause 5.1, “<token> and <separator>”	Subclause 5.2, "<token> and <separator>"	ISO/IEC 9075-2
Subclause 5.2, “Names and identifiers”	Subclause 5.4, "Names and identifiers"	ISO/IEC 9075-2
Clause 6, “Scalar expressions”	Clause 6, "Scalar expressions"	ISO/IEC 9075-2
Subclause 6.1, “<set function specification>”	Subclause 6.16, "<set function specification>"	ISO/IEC 9075-2
Subclause 6.2, “<numeric value function>”	Subclause 6.17, "<numeric value function>"	ISO/IEC 9075-2
Subclause 6.3, “<window function>”	*(none)*	*(none)*
Subclause 6.4, “<value expression>”	Subclause 6.23, "<value expression>"	ISO/IEC 9075-2
Clause 7, “Query expressions”	Clause 7, "Query expressions"	ISO/IEC 9075-2
Subclause 7.1, “<table expression>”	Subclause 7.4, "<table expression>"	ISO/IEC 9075-2
Subclause 7.2, “<joined table>”	Subclause 7.7, "<joined table>"	ISO/IEC 9075-2

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 7.3, “<where clause>”	Subclause 7.8, "<where clause>"	ISO/IEC 9075-2
Subclause 7.4, “<having clause>”	Subclause 7.10, "<having clause>"	ISO/IEC 9075-2
Subclause 7.5, “<window clause>”	*(none)*	*(none)*
Subclause 7.6, “<query specification>”	Subclause 7.11, "<query specification>"	ISO/IEC 9075-2
Clause 8, “Additional common elements”	Clause 10, "Additional common elements"	ISO/IEC 9075-2
Subclause 8.1, “<aggregate function>”	*(none)*	*(none)*
Subclause 8.2, “<sort specification list>”	*(none)*	*(none)*
Clause 9, “Schema definition and manipulation”	Clause 11, "Schema definition and manipulation"	ISO/IEC 9075-2
Subclause 9.1, “<drop table constraint definition>”	Subclause 11.19, "<drop table constraint definition>"	ISO/IEC 9075-2
Subclause 9.2, “<drop user-defined ordering statement>”	Subclause 11.55, "<drop user-defined ordering statement>"	ISO/IEC 9075-2
Clause 10, “SQL-client modules”	Clause 13, "SQL-client modules"	ISO/IEC 9075-2
Subclause 10.1, “Calls to an <externally-invoked procedure>”	Subclause 13.4, "Calls to an <externally-invoked procedure>"	ISO/IEC 9075-2
Clause 11, “Data manipulation”	Clause 14, "Data manipulation"	ISO/IEC 9075-2
Subclause 11.1, “<declare cursor>”	Subclause 14.1, "<declare cursor>"	ISO/IEC 9075-2
Subclause 11.2, “<select statement: single row>”	Subclause 14.5, "<select statement: single row>"	ISO/IEC 9075-2
Clause 12, “Dynamic SQL”	Clause 15, "Dynamic SQL"	ISO/IEC 9075-5
Subclause 12.1, “<prepare statement>”	Subclause 15.6, "<prepare statement>"	ISO/IEC 9075-5
Clause 13, “Information Schema”	Clause 20, "Information Schema"	ISO/IEC 9075-2
Subclause 13.1, “Definition of SQL built-in functions”	Subclause 20.70, "Definition of SQL built-in functions"	ISO/IEC 9075-2
Clause 14, “Status codes”	Clause 22, "Status codes"	ISO/IEC 9075-2
Subclause 14.1, “SQLSTATE”	Subclause 22.1, "SQLSTATE"	ISO/IEC 9075-2
Clause 15, “Conformance”	Clause 23, "Conformance"	ISO/IEC 9075-2
Subclause 15.1, “General conformance requirements”	Subclause 23.1, "General conformance requirements"	ISO/IEC 9075-2
Annex A, “SQL conformance summary”	Annex A, "SQL Conformance Summary"	ISO/IEC 9075-2

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3.3 Conventions

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Annex B, “Implementation-defined elements”	Annex B, "Implementation-defined elements"	ISO/IEC 9075-2
Annex C, “Implementation-dependent elements”	Annex C, "Implementation-dependent elements"	ISO/IEC 9075-2
Annex D, “SQL feature and package taxonomy”	Annex F, "SQL feature and package taxonomy"	ISO/IEC 9075-2
Annex E, “SQL Packages”	Annex B, "SQL Packages"	ISO/IEC 9075-1
Subclause E.1, “OLAP”	*(none)*	*(none)*
Figure 1, “Illustration of WIDTH_BUCKET Semantics”	*(none)*	*(none)*
Table 1, “Clause, Subclause, and Table relationships”	*(none)*	*(none)*
Table 2, “SQLSTATE class and subclass values”	Table 27, "SQLSTATE class and subclass values"	ISO/IEC 9075-2
Table 3, “Implied feature relationships”	Table 30, "Implied feature relationships"	ISO/IEC 9075-2
Table 4, “SQL/OLAP feature taxonomy for features outside Core SQL”	Table 32, "SQL/Foundation feature taxonomy for features outside Core SQL"	ISO/IEC 9075-2
Table 5, “SQL Packages”	Table 2, "SQL Packages"	ISO/IEC 9075-1

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4 Concepts

4.1 Numbers

4.1.1 Operations involving numbers

Insert this paragraph The following are also functions that return numbers:

— <natural logarithm> computes the natural logarithm of its argument.

— <exponential function> computes the exponential function, that is, *e*, (the base of natural logarithms) raised to the power equal to its argument.

— <power function> raises its first argument to the power of its second argument.

— <square root> computes the square root of its argument.

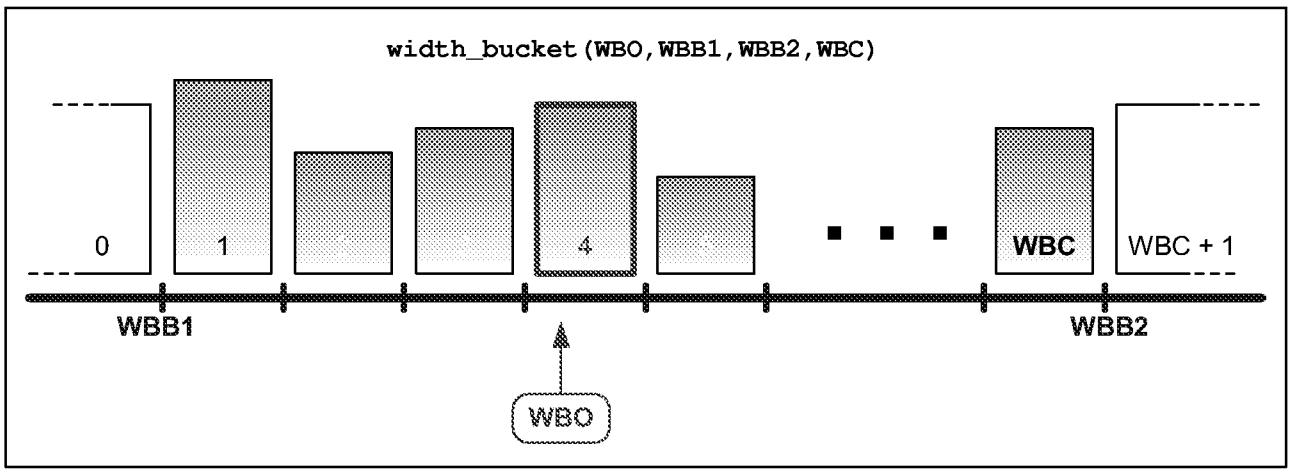
— <floor function> computes the greatest integer less than or equal to its argument.

— <ceiling function> computes the least integer greater than or equal to its argument.

— <width bucket> is a function of four arguments, returning an integer between 0 (zero) and the value of the final argument plus 1 (one), by assigning the first argument to an equi-width partitioning of the range of numbers between the second and third arguments. Values outside the range between the second and third arguments are assigned to either 0 (zero) or the value of the final argument plus 1 (one).

NOTE 2 – The semantics of <width bucket> are illustrated in Figure 1, "Illustration of WIDTH_BUCKET Semantics".

Figure 1—Illustration of WIDTH_BUCKET Semantics



Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4.2 Tables

4.2.1 Windowed tables

A *windowed table* is a table together with one or more windows. A *window* is a transient data structure associated with a <table expression>. A window is defined explicitly by a <window definition> or implicitly by an <inline window specification>. Implicitly defined windows have an implementation-dependent window name. A window is used to specify window partitions and window frames, which are multisets of rows used in the definition of <window function>s.

Every window defines a *window partitioning* of the rows of the <table expression>. The window partitioning is specified by a list of columns. Window partitioning is similar to forming groups of a grouped table. However, unlike grouped tables, each row is retained in the result of the <table expression>. The *window partition* of a row R is the multiset of rows $R2$ that are not distinct from R, for all columns enumerated in the window partitioning clause. The window partitioning clause is optional; if omitted, there is a single window partition consisting of all the rows in the result.

If a <table expression> is grouped and also has a window, then there is a syntactic transformation that segregates the grouping into a <derived table>, so that the window partitions consist of rows of the <derived table> rather than groups of rows.

A window may define a *window ordering* of rows within each window partition defined by the window. The window ordering of rows within window partitions is specified by a list of <value expression>s, followed by ASC (for ascending order) or DESC (for descending order). In addition, NULLS FIRST or NULLS LAST may be specified, to indicate whether a null value should appear before or after all non-null values in the ordered sequence of each <value expression>.

Optionally, a window may define a *window frame* for each row R. A window frame is always defined relative to the current row. A window frame is specified by up to four syntactic elements:

— The choice of RANGE, to indicate a logical definition of the window frame by offsetting forward or backward from the current row by an increment or decrement to the sort key; or ROWS, to indicate a physical definition of the window frame, by counting rows forward or backward from the current row.

— A starting row, which may be the first row of the window partition of R, the current row, or some row determined by a logical or physical offset from the current row.

— An ending row, which may be the last row of the window partition of R, the current row, or some row determined by a logical or physical offset from the current row.

— A <window frame exclusion>, indicating whether to exclude the current row and/or its peers (if not already excluded by being prior to the starting row or after the ending row).

A window is described by a *window structure descriptor*, including:

— The window name.

— Optionally, the ordering window name—that is, the name of another window, called the *ordering window*, that is used to define the partitioning and ordering of the present window.

— The window partitioning clause—that is, a <window partition clause>, if any is specified in either the present <window specification> or in the window descriptor of the ordering window.

— The window ordering clause—that is, a <window order clause>, if any is specified in either the present <window specification> or in the window descriptor of the ordering window.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

— The window framing clause—that is, a <window frame clause>, if any.

In general, two <window function>s are computed independently, each one performing its own sort of its data, even if they use the same data and the same <sort specification list>. Since sorts may specify partial orderings, the computation of <window function>s is inevitably non-deterministic to the extent that the ordering is not total. Nevertheless, the user may desire that two <window function>s be computed using the same ordering, so that, for example, two moving aggregates move through the rows of a partition in precisely the same order. Two <window function>s are computed using the same (possibly non-deterministic) window ordering of the rows if any of the following are true:

— The <window function>s identify the same window structure descriptor.

— The <window function>s' window structure descriptors have window partitioning clauses that enumerate the same number of column references, and those column references are pairwise equivalent in their order of occurrence; and their window structure descriptors have window ordering clauses with the same number of <sort key>s, and those <sort key>s are all column references, and those column references are pairwise equivalent in their order of occurrence, and the <sort specification>s pairwise specify or imply <collate clause>s that specify equivalent <collation name>s, the same <ordering specification> (ASC or DESC), and the same <null ordering> (NULLS FIRST or NULLS LAST).

— The window structure descriptor of one <window function> is the ordering window of the other <window function>, or both window structure descriptors identify the same ordering window.

4.3 Data analysis operations (involving tables)

A data analysis function is a function that returns a value derived from a number of rows in the result of a <table expression>. A data analysis function may only be invoked as part of a <query specification> or <select statement: single row>, and then only in certain contexts, identified below. A data analysis function is one of:

— A group function, which is invoked on a grouped table and computes a grouping operation or an aggregate function from a group of the grouped table.

— A window function, which is invoked on a windowed table and computes a rank, row number or window aggregate function.

4.3.1 Group functions

A group function may only appear in the <select list>, <having clause> or <window clause> of a <query specification> or <select statement: single row>, or in the <order by clause> of a cursor that is a simple table query.

A group function is one of:

— The *grouping operation*.

— A *group aggregate function*.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The grouping operation is of the form `GROUPING(<column reference>)`. The result of such an invocation is 1 (one) in the case of a row whose values are the results of aggregation over that <column reference> during the execution of a grouped query containing CUBE, ROLLUP, or GROUPING SET, and 0 (zero) otherwise.

4.3.2 Window functions

A window function is a function whose result for a given row is derived from the window frame of that row as defined by a window structure descriptor of a windowed table. Window functions may only appear in the <select list> of a <query specification> or <select statement: single row>, or the <order by clause> of a simply table query.

A window function is one of:

— A rank function.

— A distribution function.

— The row number function.

— A window aggregate function.

The rank functions compute the ordinal rank of a row R within the window partition of R as defined by a window structure descriptor, according to the window ordering of those rows, also specified by the same window structure descriptor. Rows that are not distinct with respect to the window ordering within their window partition are assigned the same rank. There are two variants, indicated by the keywords RANK and DENSE_RANK.

— If RANK is specified, then the rank of row R is defined as 1 (one) plus the number of rows that precede R and are not peers of R.

NOTE 3 – This implies that if two or more rows are not distinct with respect to the window ordering, then there will be one or more gaps in the sequential rank numbering.

— If DENSE_RANK is specified, then the rank of row R is defined as the number of rows preceding and including R that are distinct with respect to the window ordering.

NOTE 4 – This implies that there are no gaps in the sequential rank numbering of rows in each window partition.

The distribution functions compute a relative rank of a row R within the window partition of R defined by a window structure descriptor, expressed as an approximate numeric ratio between 0.0 and 1.0. There are two variants, indicated by the keywords PERCENT_RANK and CUME_DIST.

— If PERCENT_RANK is specified, then the relative rank of a row R is defined as $(RK-1)/(NR-1)$, where RK is defined to be the RANK of R and NR is defined to be the number of rows in the window partition of R.

— If CUME_DIST is specified, then the relative rank of a row R is defined as NP/NR, where NP is defined to be the number of rows preceding or peer with R in the window ordering of the window partition of R and NR is defined to be the number of rows in the window partition of R.

The ROW_NUMBER function computes the sequential row number, starting with 1 (one) for the first row, of the row within its window partition according to the window ordering of the window.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The window aggregate functions compute an aggregate value (COUNT, SUM, AVG, *etc.*), the same as a group aggregate function, except that the computation aggregates over the window frame of a row rather than over a group of a grouped table. The hypothetical set functions are not permitted as window aggregate functions.

4.3.3 Aggregate functions

An aggregate function is a function whose result is derived from an aggregation of rows defined by one of:

— The grouping of a grouped table, in which case the aggregate function is a group aggregate function, or set function, and for each group there is one aggregation, which includes every row in the group.

— The window frame of a row *R* of a windowed table relative to a particular window structure descriptor, in which case the aggregate function is a window aggregate function, and the aggregation consists of every row in the window frame of *R*, as defined by the window structure descriptor.

Optionally, the multiset of rows in an aggregation may be filtered, retaining only those rows that satisfy a <search condition> that is specified by a <filter clause>.

The result of the aggregate function COUNT (*) is the number of rows in the aggregation.

Every other aggregate function may be classified as a *unary aggregate function*, a *binary aggregate function*, an *inverse distribution*, or a *hypothetical set function*.

Every unary aggregate function takes an arbitrary <value expression> as the argument; most unary aggregate functions can optionally be qualified with either DISTINCT or ALL. Of the rows in the aggregation, the following do not qualify:

— If DISTINCT is specified, then redundant duplicates.

— Every row in which the <value expression> evaluates to the null value.

If no row qualifies, then the result of COUNT is 0 (zero), and the result of any other aggregate function is the null value.

Otherwise (*i.e.*, at least one row qualifies), the result of the aggregate function is:

— If COUNT <value expression> is specified, then the number of rows that qualify.

— If SUM is specified, then the sum of <value expression> evaluated for each row that qualifies.

— If AVG is specified, then the average of <value expression> evaluated for each row that qualifies.

— If MAX is specified, then the maximum value of <value expression> evaluated for each row that qualifies.

— If MIN is specified, then the minimum value of <value expression> evaluated for each row that qualifies.

— If EVERY is specified, then true if the <value expression> evaluates to *true* for every row that qualifies, otherwise, *false* .

— If ANY or SOME is specified, then *true* if the <value expression> evaluates to *true* for at least one row remaining in the group; otherwise, *false* .

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4.3 Data analysis operations (involving tables)

— If VAR_POP is specified, then the population variance of <value expression> evaluated for each row remaining in the group, defined as the sum of squares of the difference of <value expression> from the mean of <value expression>, divided by the number of rows remaining.

— If VAR_SAMP is specified, then the sample variance of <value expression> evaluated for each row remaining in the group, defined as the sum of squares of the difference of <value expression> from the mean of <value expression>, divided by the number of rows remaining minus 1 (one).

— If STDDEV_POP is specified, then the population standard deviation of <value expression> evaluated for each row remaining in the group, defined as the square root of the population variance.

— If STDDEV_SAMP is specified, then the sample standard deviation of <value expression> evaluated for each row remaining in the group, defined as the square root of the sample variance.

Neither DISTINCT nor ALL are allowed to be specified for VAR_POP, VAR_SAMP, STDDEV_POP or STDDEV_SAMP; redundant duplicates are not removed when computing these functions.

The binary aggregate functions take a pair of arguments, the <dependent variable expression> and the <independent variable expression>, which are both <numeric value expression>s. Any row in which either argument evaluates to the null value is removed from the group. If there are no rows remaining in the group, then the result of REGR_COUNT is 0 (zero), and the other binary aggregate functions result in the null value. Otherwise, the computation concludes and the result is:

— If REGR_COUNT is specified, then the number of rows remaining in the group.

— If COVAR_POP is specified, then the population covariance, defined as the sum of products of the difference of <independent variable expression> from its mean times the difference of <dependent variable expression> from its mean, divided by the number of rows remaining.

— If COVAR_SAMP is specified, then the sample covariance, defined as the sum of products of the difference of <independent variable expression> from its mean times the difference of <dependent variable expression> from its mean, divided by the number of rows remaining minus 1 (one).

— If CORR is specified, then the correlation coefficient, defined as the ratio of the population covariance divided by the product of the population standard deviation of <independent variable expression> and the population standard deviation of <dependent variable expression>.

— If REGR_R2 is specified, then the square of the correlation coefficient.

— If REGR_SLOPE is specified, then the slope of the least-squares-fit linear equation determined by the (<independent variable expression>, <dependent variable expression>) pairs.

— If REGR_INTERCEPT is specified, then the y-intercept of the least-squares-fit linear equation determined by the (<independent variable expression>, <dependent variable expression>) pairs.

— If REGR_SXX is specified, then the sum of squares of <independent variable expression>.

— If REGR_SYY is specified, then the sum of squares of <dependent variable expression>.

— If REGR_SXY is specified, then the sum of products of <independent variable expression> times <dependent variable expression>.

— If REGR_AVGX is specified, then the average of <independent variable expression>.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

— If REGR_AVGY is specified, then the average of <dependent variable expression>.

There are two inverse distribution functions, PERCENTILE_CONT and PERCENTILE_DISC. Both inverse distribution functions specify an argument and an ordering of a value expression. The value of the argument should be between 0 (zero) and 1 (one) inclusive. The value expression is evaluated for each row of the group, nulls are discarded, and the remaining rows are ordered. The computation concludes:

— If PERCENTILE_CONT is specified, by considering the pair of consecutive rows that are indicated by the argument, treated as a fraction of the total number of rows in the group, and interpolating the value of the value expression evaluated for these rows.

— If PERCENTILE_DISC is specified, by treating the group as a window partition of the CUME_DIST window function, using the specified ordering of the value expression as the window ordering, and returning the first value expression whose cumulative distribution value is greater than or equal to the argument.

The hypothetical set functions are related to the window functions RANK, DENSE_RANK, PERCENT_RANK and CUME_DIST, and use the same names, though with a different syntax. These functions take an argument *A* and an ordering of a value expression *VE*. *VE* is evaluated for all rows of the group. This multiset of values is augmented with *A*; the resulting collection is treated as a window partition of the corresponding window function whose window ordering is the ordering of the value expression. The result of the hypothetical set function is the value of the eponymous window function for the hypothetical "row" that contributes *A* to the collection.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5 Lexical elements

5.1 <token> and <separator>

Function

Specify lexical units (tokens and separators) that participate in SQL language.

Format

```
<non-reserved word> ::=
        !! All alternatives from ISO/IEC 9075-2
      | !! All alternatives from ISO/IEC 9075-5

      | CEIL | CEILING | CORR | COVAR_POP | COVAR_SAMP | CUME_DIST

      | DENSE_RANK

      | EXCLUDE | EXP

      | FILTER | FLOOR | FOLLOWING

      | LN

      | NULLS

      | OTHERS | OVER

      | PARTITION | PERCENTILE_CONT | PERCENTILE_DISC | PERCENT_RANK | POWER | PRECEDING


      | RANGE | RANK | REGR_AVGX | REGR_AVGY | REGR_COUNT | REGR_INTERCEPT
      | REGR_R2 | REGR_SLOPE | REGR_SXX | REGR_SXY | REGR_SYY | ROW_NUMBER

      | SQRT | STDDEV_POP | STDDEV_SAMP

      | TIES

      | UNBOUNDED

      | VAR_POP | VAR_SAMP

      | WIDTH_BUCKET

<reserved word> ::=
        !! All alternatives from ISO/IEC 9075-2
      | !! All alternatives from ISO/IEC 9075-5

      | WINDOW
```

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5.1 <token> and <separator>

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2001 – All rights reserved

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5.2 Names and identifiers

Function

Specify names.

Format

```
<window name> ::= <identifier>
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Insert this GR A <window name> identifies a window.

Conformance Rules

1) Insert this CR Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify <window name>.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

(Blank page)

© ISO/IEC 2001 – All rights reserved

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6 Scalar expressions

6.1 <set function specification>

Function

Specify a value derived by the application of a function to an argument.

Format

```
<set function specification> ::=
        <aggregate function>
      | <grouping operation>
```

Delete the definition of <general set function>

Delete the definition of <set function type>

Delete the definition of <computational operation>

Delete the definition of <set quantifier>

NOTE 5 – The deleted definitions are now placed in Subclause 8.1, "<aggregate function>".

Syntax Rules

1) Delete SR 1) through 3)

2) Insert this SR If <aggregate function> specifies a <filter clause>, then the <filter condition> shall not contain a <set function specification>.

3) Replace SR 4) If <aggregate function> specifies a <general set function>, then the <value expression> simply contained in the <general set function> shall not contain a <set function specification> or a <subquery>. If the <value expression> contains a column reference that is an outer reference, then that outer reference shall be the only column reference contained in the <value expression>.

4) Insert this SR If <aggregate function> specifies <binary set function>, then neither the <dependent variable expression> nor the <independent variable expression> simply contained in the <binary set function> shall contain a <set function specification> or a <subquery>. If the <dependent variable expression> or <independent variable expression> contains a column reference that is an outer reference, then that outer reference shall be the only column reference contained in the <dependent variable expression> or <independent variable expression>.

5) Insert this SR A <value expression> *VE* simply contained in a <set function specification> *SFE* is an *aggregated argument* of *SFE* if either *SFE* is not an <ordered set function> or *VE* is simply contained in a <within group specification>; otherwise, *VE* is a *non-aggregated argument* of *SFE*.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.1 <set function specification>

6) Replace SR 5) If an aggregated argument of a <set function specification> contains a column reference that is an outer reference, then the <set function specification> shall be contained in either:

   a) A <select list>.

   b) A <window clause>.

   c) A <subquery> of a <having clause>, in which case the qualifying table of the <column reference> shall be the table referenced by a <table reference> that is directly contained in the <table expression> that directly contains the <having clause>.

7) Insert this SR If <aggregate function> is specified, then the declared type of the result is the declared type of the <aggregate function>. If the declared type is character string, then the collating sequence and coercibility characteristic are the collating sequence and coercibility characteristic of the <aggregate function>.

8) Delete SRs 6) through 14)

9) Delete SR 16)

Access Rules

No additional Access Rules.

General Rules

1) Delete GR 1) and GR 2)

2) Delete GRs 3)a) through 3)g)

3) Insert this GR If <aggregate function> is specified, then the result is the value of the <aggregate function>.

Conformance Rules

1) Delete CRs 1) through 6)

2) Delete CRs 8) and 9)

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.2 <numeric value function>

Function

Specify a function yielding a value of type numeric.

Format

```
<numeric value function> ::=
        !! All alternatives from ISO/IEC 9075-2
      | <natural logarithm>
      | <exponential function>
      | <power function>
      | <square root>
      | <floor function>
      | <ceiling function>
      | <width bucket function>

<natural logarithm> ::=
      LN <left paren> <numeric value expression> <right paren>

<exponential function> ::=
      EXP <left paren> <numeric value expression> <right paren>

<power function> ::=
      POWER <left paren> <numeric value expression base>
          <comma> <numeric value expression exponent> <right paren>

<numeric value expression base> ::=
      <numeric value expression>

<numeric value expression exponent> ::=
      <numeric value expression>

<square root> ::=
      SQRT <left paren> <numeric value expression> <right paren>

<floor function> ::=
      FLOOR <left paren> <numeric value expression> <right paren>

<ceiling function> ::=
      { CEIL | CEILING } <left paren> <numeric value expression> <right paren>

<width bucket function> ::=
      WIDTH_BUCKET <left paren> <width bucket operand> <comma>
          <width bucket bound 1> <comma> <width bucket bound 2>
          <comma> <width bucket count> <right paren>

<width bucket operand> ::=
      <numeric value expression>

<width bucket bound 1> ::=
      <numeric value expression>

<width bucket bound 2> ::=
      <numeric value expression>

<width bucket count> ::=
      <numeric value expression>
```

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.2 <numeric value function>

Syntax Rules

1) Insert this SR The declared type of the result of <natural logarithm> is approximate numeric with implementation-defined precision.

2) Insert this SR The declared type of the result of <exponential function> is approximate numeric with implementation-defined precision.

3) Insert this SR The declared type of the result of <power function> is approximate numeric with implementation-defined precision.

4) Insert this SR If <square root> is specified, then let *NVE* be the simply contained <numeric value expression>. The <square root> is equivalent to

```
POWER (NVE, 0.5)
```

5) Insert this SR The declared type of the result of <floor function> is exact numeric with implementation-defined precision, with the radix of the simply contained <numeric value expression>, and with scale 0 (zero).

6) Insert this SR The declared type of the result of <ceiling function> is exact numeric with implementation-defined precision, with the radix of the simply contained <numeric value expression>, and with scale 0 (zero).

7) Insert this SR If <width bucket function> is specified, then the declared type of <width bucket count> shall be exact numeric with scale 0 (zero). The declared type of the result of <width bucket function> is the declared type of <width bucket count>.

Access Rules

No additional Access Rules.

General Rules

1) Insert this GR If <natural logarithm> is specified, then let *V* be the value of the simply contained <numeric value expression>.

   a) If *V* is the null value, then the result is the null value.

   b) If *V* is 0 (zero) or negative, then an exception condition is raised: *data exception — invalid argument for natural logarithm*.

   c) Otherwise, the result is the natural logarithm of *V*.

2) Insert this GR If <exponential function> is specified, then let *V* be the value of the simply contained <numeric value expression>.

   a) If *V* is the null value, then the result is the null value.

   b) Otherwise, the result is *e* (the base of natural logarithms) raised to the power *V*. If the result is not representable in the the declared type of the result, then an exception condition is raised: *data exception — numeric value out of range*.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3) Insert this GR If <power function> is specified, then let *NVEB* be the <numeric value expression base>, then let *VB* be the value of *NVEB*, let *NVEE* be the <numeric value expression exponent>, and let *VE* be the value of *NVEE*.

   a) If either *VB* or *VE* is the null value, then the result is the null value.

   b) If *VB* is 0 (zero) and *VE* is negative, then an exception condition is raised: *data exception — invalid argument for power function*.

   c) If *VB* is 0 (zero) and *VE* is 0 (zero), then the result is 1 (one).

   d) If *VB* is 0 (zero) and *VE* is positive, then the result is 0 (zero).

   e) If *VB* is negative and *VE* is not equal to an exact numeric value with scale 0 (zero), then an exception condition is raised: *data exception — invalid argument for power function*.

   f) If *VB* is negative and *VE* is equal to an exact numeric value with scale 0 (zero) that is an even number, then the result is the result of

      EXP(*NVEE**LN(-*NVEB*))

   g) If *VB* is negative and *VE* is equal to an exact numeric value with scale 0 (zero) that is an odd number, then the result is the result of

      -EXP(*NVEE**LN(-*NVEB*))

   h) Otherwise, the result is the result of

      EXP(*NVEE**LN(*NVEB*))

4) Insert this GR If <floor function> is specified, then let *V* be the value of the simply contained <numeric value expression>.

   Case:

   a) If *V* is the null value, then the result is the null value.

   b) Otherwise, the result is the greatest exact numeric value with scale 0 (zero) that is less than or equal to *V*. If this result is not representable by the result data type, then an exception condition is raised: *data exception — numeric value out of range*.

5) Insert this GR If <ceiling function> is specified, then let *V* be the value of the simply contained <numeric value expression>.

   Case:

   a) If *V* is the null value, then the result is the null value.

   b) Otherwise, the result is the least exact numeric value with scale 0 (zero) that is greater than or equal to *V*. If this result is not representable by the result data type, then an exception condition is raised: *data exception — numeric value out of range*.

6) Insert this GR If <width bucket function> is specified, then let *WBO* be the value of <width bucket operand>, let *WBB1* be the value of <width bucket bound 1>, let *WBB2* be the value of <width bucket bound 2>, and let *WBC* be the value of <width bucket count>.

   Case:

   a) If any of *WBO*, *WBB1*, *WBB2*, or *WBC* is the null value, then the result is the null value.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.2 <numeric value function>

b) If *WBC* is less than or equal to 0 (zero), then an exception condition is raised: *data exception — invalid argument for width bucket function.*

c) If *WBB1* equals *WBB2*, then an exception condition is raised: *data exception — invalid argument for width bucket function.*

d) If *WBB1* is less than *WBB2*, then

Case:

i) If *WBO* is less than *WBB1*, then the result is 0 (zero).

ii) If *WBO* is greater than or equal to *WBB2*, then the result is *WBC*+1. If the result is not representable in the declared type of the result, then an exception condition is raised: *data exception — numeric value out of range.*

iii) Otherwise, the result is the greatest exact numeric value with scale 0 (zero) that is less than or equal to `((WBC * (WBO - WBB1) / (WBB2 - WBB1)) + 1)`.

e) If *WBB1* is greater than *WBB2*, then

Case:

i) If *WBO* is greater than *WBB1*, then the result is 0 (zero).

ii) If *WBO* is less than or equal to *WBB2*, then the result is *WBC*+1. If the result is not representable in the declared type of the result, then an exception condition is raised: *data exception — numeric value out of range.*

iii) Otherwise, the result is the least exact numeric value with scale 0 (zero) that is less than or equal to `((WBC * (WBB1 - WBO) / (WBB1 - WBB2)) + 1)`.

Conformance Rules

The following restrictions apply for Core SQL:

1) Insert this CR Without Feature T621, "Enhanced numeric functions", conforming SQL language shall not specify <natural logarithm>, <exponential function>, <power function>, <square root>, <floor function> or <ceiling function>.

2) Insert this CR Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify <width bucket function>.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.3 <window function>

Function

Specify a window function.

Format

```
<window function> ::=
      <window function type> OVER <window name or specification>

<window function type> ::=
        <rank function type> <left paren> <right paren>
      | ROW_NUMBER <left paren> <right paren>
      | <aggregate function>

<rank function type> ::=
        RANK
      | DENSE_RANK
      | PERCENT_RANK
      | CUME_DIST

<window name or specification> ::=
        <window name>
      | <in-line window specification>

<in-line window specification> ::=
      <window specification>
```

Syntax Rules

1) An <aggregate function> simply contained in a <window function> shall not simply contain a <hypothetical set function>.

2) Let *OF* be the <window function>.

3) Case:

   a) If *OF* is contained in an <order by clause>, then the <order by clause> shall be contained in a <cursor specification> that is a simple table query. Let *ST* be the sort table that is obtained by applying the syntactic transformation of a simple table query, as specified in Subclause 11.1, "<declare cursor>". Let *TE* be the <table expression> contained in the result of that syntactic transformation.

   b) Otherwise, *OF* shall be contained in a <select list> that is immediately contained in a <query specification> *QS* or a <select statement: single row> *SSSR*. Let *QSS* be the innermost <query specification> contained in *QS* that contains *OF*. Let *TE* be the <table expression> immediately contained in *QSS* or *SSSR*.

4) *OF* shall not contain an outer reference or a <subquery>.

5) If the window ordering clause or the window framing clause of the window structure descriptor that describes the <window name or specification> is present, then no <aggregate function> simply contained in <window function> shall specify DISTINCT or <ordered set function>.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.3 <window function>

6) Let *WNS* be the <window name or specification>. Let *WDX* be a window structure descriptor that describes the window defined by *WNS*.

7) If <rank function type> or ROW_NUMBER is specified, then:

   a) If RANK or DENSE_RANK is specified, then the window ordering clause *WOC* of *WDX* shall be present.

   b) The window framing clause of *WDX* shall not be present.

   c) Case:

      i) If *WNS* is a <window name>, then let *WNS1* be *WNS*.

      ii) Otherwise, let *WNS1* be the <window specification details> contained in *WNS*.

   d) `RANK ( ) OVER` *WNS* is equivalent to:

```
( COUNT (*) OVER ( WNS1 RANGE UNBOUNDED PRECEDING )
  - COUNT (*) OVER ( WNS1 RANGE CURRENT ROW ) + 1 )
```

   e) If DENSE_RANK is specified, then:

      i) Let VE_1, . . . , VE_N be an enumeration of the <value expression>s that are <sort key>s simply contained in *WOC*.

      ii) `DENSE_RANK ( ) OVER` *WNS* is equivalent to the <window function>:

```
COUNT (DISTINCT ROW ( VE1, . . . , VEN ) )
OVER (WNS1 RANGE UNBOUNDED PRECEDING)
```

   f) `ROW_NUMBER ( ) OVER` *WNS* is equivalent to the <window function>:

```
COUNT (*) OVER (WNS1 ROWS UNBOUNDED PRECEDING)
```

   g) Let *ANT1* be an approximate numeric type with implementation-defined precision. `PERCENT_RANK ( ) OVER` *WNS* is equivalent to:

```
CASE
  WHEN COUNT(*) OVER (WNS1 RANGE BETWEEN UNBOUNDED PRECEDING
                                     AND UNBOUNDED FOLLOWING) = 1
  THEN CAST (0 AS ANT1)
  ELSE
       (CAST (RANK () OVER (WNS1) AS ANT1) - 1) /
       (COUNT (*) OVER (WNS1 RANGE BETWEEN UNBOUNDED PRECEDING
                           AND UNBOUNDED FOLLOWING) - 1)
END
```

   h) Let *ANT2* be an approximate numeric type with implementation-defined precision. `CUME_DIST ( ) OVER` *WNS* is equivalent to:

© ISO/IEC 2001 – All rights reserved

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
( CAST ( COUNT (*) OVER
      ( WNS1 RANGE UNBOUNDED PRECEDING ) AS ANT2 ) /
    COUNT(*) OVER ( WNS1 RANGE BETWEEN UNBOUNDED PRECEDING
                                   AND UNBOUNDED FOLLOWING ) )
```

8) Let *SL* be the <select list> that simply contains *OF*.

NOTE 6 – If *OF* is originally contained in an <order by clause> of a cursor that is a simple table query, the syntactic transformation of Subclause 11.1, "<declare cursor>", must be applied prior to this rule.

9) Let *SQ* be the <set quantifier> of the <query specification> or <select statement: single row> that simply contains *SL*. If there is no <set quantifier>, then let *SQ* be a zero-length string.

10) If <in-line window specification> is specified, then:

a) Let *WS* be the <window specification>.

b) Let *WSN* be an implementation-dependent <window name> that is not equivalent to any other <window name> in the <table expression> or <select statement: single row> that simply contains *WS*.

c) Let *OFT* be the <window function type>.

d) Let *SLNEW* be the <select list> that is obtained from *SL* by replacing *OF* by:

```
OFT OVER WSN
```

e) Let *FC*, *WC*, *GBC*, and *HC* be <from clause>, <where clause>, <group by clause>, and <having clause>, respectively, of *TE*. If any of <where clause>, <group by clause>, or <having clause> is missing, then let *WC*, *GBC*, or *HC*, respectively, be a zero-length string.

f) Case:

i) If there is no <window clause> simply contained in *TE*, then let *WICNEW* be:

```
WINDOW WSN AS WS
```

ii) Otherwise, let *WIC* be the <window clause> simply contained in *TE* and let *WICNEW* be:

```
WIC, WSN AS WS
```

g) Let *TENEW* be:

```
FC WC GBC HC WICNEW
```

h) Case:

i) If *OF* is simply contained in a <query specification>, then that <query specification> is equivalent to:

```
SELECT SQ SLNEW TENEW
```

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.3 <window function>

ii) Otherwise, *OF* is simply contained in a <select statement: single row>. Let *STL* be the <select target list> of that <select statement: single row>. The <select statement: single row> is equivalent to:

```
SELECT SQ SLNEW INTO STL TENEW
```

Access Rules

None.

General Rules

1) The value of <window function> is the value of the <aggregate function>.

Conformance Rules

1) Without Feature T611, "Elementary OLAP operators", conforming SQL language shall not specify a <window function>.

2) Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify <window name>.

3) Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify PERCENT_RANK or CUME_DIST.

4) Without Feature T612, "Advanced OLAP operators", if ROW_NUMBER is specified, then the window ordering clause of *WDX* shall be present.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

6.4 <value expression>

Function

Specify a value.

Format

```
<nonparenthesized value expression primary> ::=
         !! All alternatives from ISO/IEC 9075-2
       | <window function>
```

Syntax Rules

1) Replace SR 2) The declared type of a <value expression primary> is the declared type of the simply contained <unsigned value specification>, <column reference>, <set function specification>, <scalar subquery>, <case expression>, <value expression>, <cast specification>, <subtype treatment>, <attribute or method reference>, <reference resolution>, <collection value constructor>, <field reference>, <element reference>, <method invocation>, or <static method invocation>, or <window function>, or the effective returns type of the immediately contained <routine invocation>, respectively.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 5) The value of a <value expression primary> is the value of the simply contained <unsigned value specification>, <column reference>, <set function specification>, <rank function specification>, <scalar subquery>, <case expression>, <value expression>, <cast specification>, <subtype treatment>, <collection value constructor>, <field reference>, <element reference>, <method invocation>, <static method invocation>, <routine invocation>, <attribute or method reference>, or <window function>.

Conformance Rules

1) Insert this CR Without Feature T611, "Elementary OLAP operators", a <value expression> shall not be a <window function>.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2001 – All rights reserved

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7 Query expressions

7.1 <table expression>

Format

```
<table expression> ::=
       <from clause>
       [ <where clause> ]
       [ <group by clause> ]
       [ <having clause> ]
       [ <window clause> ]
```

Syntax Rules

1) Replace SR 1) The result of a <table expression> is a derived table, whose row type *RT* is the row type of the result of the application of the last of the immediately contained <from clause>, <where clause>, <group by clause> or <having clause> specified in the <table expression>, together with the window structure descriptors defined by the <window clause>, if specified.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify a <window clause>.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.2 <joined table>

Function

Specify a table derived from a Cartesian product, inner or outer join, or union join.

Format

```
!! No additional Format items
```

Syntax Rules

1) Replace SR 5)b) If a <value expression> directly contained in the <search condition> is a <set function specification>, then the <joined table> shall be contained in a <having clause> or <select list>, the <set function specification> shall contain an aggregated argument that is a column reference that is an outer reference, and every column reference contained in an aggregated argument of the <set function specification> shall be an outer reference.

2) Insert after SR 5) The <search condition> shall not contain a <window function> without an intervening <subquery>.

Access Rules

No additional Access Rules

General Rules

No additional General Rules

Conformance Rules

No additional Conformance Rules.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3 <where clause>

Function

Specify a table derived by the application of a <search condition> to the result of the preceding <from clause>.

Format

```
!! No additional Format items
```

Syntax Rules

1) Replace SR 2) If a <value expression> directly contained in the <search condition> is a <set function specification>, then the <where clause> shall be contained in a <having clause> or <select list>, the <set function specification> shall contain a column reference, and every column reference contained in an aggregated argument of the <set function specification> shall be an outer reference.

2) Insert after SR 2) The <search condition> shall not contain a <window function> without an intervening <subquery>.

3) Replace SR 4) No column reference contained in a <subquery> in the <search condition> that references a column of T shall be specified in an aggregated argument of a <set function specification>.

Access Rules

No additional Access Rules

General Rules

No additional General Rules

Conformance Rules

No additional Conformance Rules

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.4 <having clause>

Function

Specify a grouped table derived by the elimination of groups that do not satisfy a <search condition>.

Format

```
!! No additional Format items
```

Syntax Rules

1) Replace SR 4) Each column reference contained in a <subquery> in the <search condition> that references a column of T shall reference a column that is functionally dependent on G or shall be specified within an aggregated argument of a <set function specification>.

2) Insert after SR 4) The <search condition> shall not contain a <window function> without an intervening <subquery>.

Access Rules

No additional Access Rules

General Rules

1) Replace GR 2) When the <search condition> is applied to a given group of R, that group is the argument source of each <set function specification> directly contained in the <search condition>, unless the <column reference> in the <set function specification> is an outer reference.

Conformance Rules

1) Replace CR 2) Without Feature T301, "Functional dependencies", each column reference contained in a <subquery> in the <search condition> that references a column of T shall be one of the following:

   a) An unambiguous reference to a grouping column of T.

   b) Contained in an aggregated argument of a <set function specification>.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.5 <window clause>

Function

Specify one or more window definitions.

Format

```
<window clause> ::=
       WINDOW <window definition list>

<window definition list> ::=
       <window definition> [ { <comma> <window definition> }... ]

<window definition> ::=
       <new window name> AS <window specification>

<new window name> ::= <window name>

<window specification> ::=
       <left paren> <window specification details> <right paren>

<window specification details> ::=
       [ <existing window name> ]
       [ <window partition clause> ]
       [ <window order clause> ]
       [ <window frame clause> ]

<existing window name> ::= <window name>

<window partition clause> ::=
       PARTITION BY <window partition column reference list>

<window partition column reference list> ::=
       <window partition column reference>
           [ { <comma> <window partition column reference> }... ]

<window partition column reference> ::=
       <column reference> [ <collate clause> ]

<window order clause> ::=
       ORDER BY <sort specification list>

<window frame clause> ::=
       <window frame units>
       <window frame extent>
       [ <window frame exclusion> ]

<window frame units> ::=
         ROWS
       | RANGE

<window frame extent> ::=
         <window frame start>
       | <window frame between>

<window frame start> ::=
         UNBOUNDED PRECEDING
       | <window frame preceding>
```

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.5 <window clause>

```
       | CURRENT ROW

<window frame preceding> ::=
       <unsigned value specification> PRECEDING

<window frame between> ::=
       BETWEEN <window frame bound 1>
           AND <window frame bound 2>

<window frame bound 1> ::=
       <window frame bound>

<window frame bound 2> ::=
       <window frame bound>

<window frame bound> ::=
         <window frame start>
       | UNBOUNDED FOLLOWING
       | <window frame following>

<window frame following> ::=
       <unsigned value specification> FOLLOWING

<window frame exclusion> ::=
         EXCLUDE CURRENT ROW
       | EXCLUDE GROUP
       | EXCLUDE TIES
       | EXCLUDE NO OTHERS
```

Syntax Rules

1) Let *TE* be the <table expression> that immediately contains the <window clause>.

2) <new window name> *NWN1* shall not be contained in the scope of another <new window name> *NWN2* such that *NWN1* and *NWN2* are equivalent.

3) Let *WDEF* be a <window definition>.

4) Each <column reference> contained in the <window partition clause> or <window order clause> of *WDEF* shall unambiguously reference a column of the derived table that is the result of *TE*. A column referenced in a <window partition clause> is a *partitioning column*.

   NOTE 7 – If *TE* is a grouped table, then the <column reference>s contained in <window partition clause> or <window order clause> must reference columns of the grouped table obtained by performing the syntactic transformation in Subclause 7.6, "<query specification>".

5) The declared type of a partitioning column shall not be LOB-ordered, array-ordered, UDT-EC-ordered, or UDT-NC-ordered.

6) If *T* is a grouped table, then let *G* be the set of grouping columns of *T*. Each column reference contained in <window clause> that references a column of *T* shall reference a column that is functionally dependent on *G* or be contained in an aggregated argument of a <set function specification>.

7) A <window clause> shall not contain a <window function> without an intervening <subquery>.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8) If *WDEF* specifies <window frame between>, then:

   a) <window frame bound 1> shall not specify UNBOUNDED FOLLOWING.

   b) <window frame bound 2> shall not specify UNBOUNDED PRECEDING.

   c) If <window frame bound 1> specifies CURRENT ROW, then <window frame bound 2> shall not specify <window frame preceding>.

   d) If <window frame bound 1> specifies <window frame following>, then <window frame bound 2> shall not specify <window frame preceding> or CURRENT ROW.

9) If *WDEF* specifies <window frame extent>, and does not specify <window frame between>, then let *WAGS* be the <window frame start>. The <window frame extent> is equivalent to

```
BETWEEN WAGS AND CURRENT ROW
```

10) If *WDEF* specifies an <existing window name> *EWN*, then:

   a) *WDEF* shall be within the scope of a <window name> that is equivalent to <existing window name>.

   b) Let *WDX* be the window structure descriptor identified by *EWN*.

   c) *WDEF* shall not specify <window partition clause>.

   d) If *WDX* has a window ordering clause, then *WDEF* shall not specify <window order clause>.

   e) *WDX* shall not have a window framing clause.

11) If *WDEF*'s <window frame clause> specifies <window frame preceding> or <window frame following>, then let *UVS* be the <unsigned value specification> simply contained in the <window frame preceding> or <window frame following>.

   Case:

   a) If RANGE is specified, then *WDEF*'s <window order clause> shall contain a single <sort key> *SK*. The declared type of *SK* shall be numeric, datetime, or interval. The declared type of *UVS* shall be numeric if the declared type of *SK* is numeric; otherwise, it shall be an interval type that may be added to or subtracted from the declared type of *SK* according to the Syntax Rules of Subclause 6.28, "<datetime value expression>", and Subclause 6.29, "<interval value expression>", in ISO/IEC 9075-2.

   b) If ROWS is specified, then the declared type of *UVS* shall be exact numeric with scale 0 (zero).

12) The scope of the <new window name> simply contained in *WDEF* consists of any <window definition>s that follow *WDEF* in the <window clause>, together with the <select list> of the <query specification> or <select statement: single row> that simply contains the <window clause>. If the <window clause> is simply contained in a <query specification> that is the <query expression body> of a <declare cursor> that is a simple table query, then the scope of <new window name> also includes the <order by clause> of the <declare cursor>.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.5 <window clause>

13) Two window structure descriptors *WD1* and *WD2* are *order-equivalent* if all of the following conditions are met:

a) Let $WPCR1_i$, 1 (one) $\leq i \leq N1$, and $WPCR2_i$, 1 (one) $\leq i \leq N2$, be enumerations of the <window partition column reference>s contained in the window partitioning clauses of *WD1* and *WD2*, respectively, in order from left to right. $N1 = N2$, and, for all i, $WPCR1_i$ and $WPCR2_i$ are equivalent column references.

b) Let $SS1_i$, 1 (one) $\leq i \leq M1$, and $SS2_i$, 1 (one) $\leq i \leq M2$, be enumerations of the <sort specification>s contained in the window ordering clauses of *WD1* and *WD2*, respectively, in order from left to right. $M1 = M2$, and, for all i, $SS1_i$ and $SS2_i$ contain <sort key>s that are equivalent column references, specify or imply the same <ordering specification>, specify or imply the same <collate clause>, if any, and specify or imply the same <null ordering>.

Access Rules

None.

General Rules

1) Let *TE* be the <table expression> that simply contains the <window clause>. Let *SL* be the <select list> of the <query specification> or <select statement: single row> that immediately contains *TE*.

Case:

a) If *SL* does not simply contain a <window function>, then the <window clause> is disregarded, and the result of *TE* is the result of the last <from clause>, <where clause>, <group by clause> or <having clause> of *TE*.

b) Otherwise, let *RTE* be the result of the last <from clause> or <where clause> simply contained in *TE*.

NOTE 8 – Although it is permissible to have a <group by clause> or a <having clause> with a <window clause>, if there are any <window function>s, then the <group by clause> and <having clause> are removed by a syntactic transformation in Subclause 7.6, "<query specification>", and so are not considered here.

i) A window structure descriptor *WDESC* is created for each <window definition> *WDEF*, as follows:

1) *WDESC*'s window name is the <new window name> simply contained in *WDEF*.

2) If <existing window name> is specified, then let *EWN* be the <existing window name> simply contained in *WDEF* and let *WDX* be the window structure descriptor identified by *EWN*.

3) If <existing window name> is specified and the window ordering clause of *WDX* is present, then the ordering window name of *WDESC* is *EWN*; otherwise, there is no ordering window name.

4) Case:

A) If *WDEF* simply contains <window partition clause> *WDEFWPC*, then *WDESC*'s window partitioning clause is *WDEFWPC*.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B) If <existing window name> is specified, then *WDESC*'s window partitioning clause is the window partitioning clause of *WDX*.

C) Otherwise, *WDESC* has no window partitioning clause.

5) Case:

A) If *WDEF* simply contains <window order clause> *WDEFWOC*, then *WDESC*'s window ordering clause is *WDEFWOC*.

B) If <existing window name> is specified, then *WDESC*'s window ordering clause is the window ordering clause of *WDX*.

C) Otherwise, *WDESC* has no window ordering clause.

6) If *WDEF* simply contains <window frame clause> *WDEFWFC*, then *WDESC*'s window framing clause is *WDEFWFC*; otherwise, *WDESC* has no windows framing.

ii) The result of <window clause> is *RTE*, together with the window structure descriptors defined by the <window clause>.

2) Let *WD* be a window structure descriptor.

3) *WD* defines, for each row *R* of *RTE*, the *window partition* of *R* under *WD*, consisting of the multiset of rows of *RTE* that are not distinct from *R* in the window partitioning columns of *WD*. If *WD* has no window partitioning clause, then the window partition of *R* is the entire result *RTE*.

4) *WD* also defines the window ordering of the rows of each window partition defined by *WD*, according to the General Rules of Subclause 8.2, "<sort specification list>", using the <sort specification list> simply contained in *WD*'s window ordering clause. If *WD* has no window ordering clause, then the window ordering is entirely implementation-dependent, and all rows are peers. Although the window ordering of peer rows within a window partition is implementation-dependent, the window ordering shall be the same for all window structure descriptors that are order-equivalent. It shall also be the same for any pair of windows *W1* and *W2* such that *W1* is the ordering window for *W2*.

5) *WD* also defines for each row *R* of *RTE* the window frame *WF* of *R*, consisting of a multiset of rows. *WF* is defined as follows:

a) If *WD* has no window framing clause, then

Case:

i) If the window ordering clause of *WD* is not present, then *WF* is the window partition of *R*.

ii) Otherwise, *WF* consists of all rows of the window partition of *R* that precede *R* or are peers of *R* in the window ordering of the window partition defined by the window ordering clause.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.5 <window clause>

b) Otherwise, let *WF* initially be the window partition of *R* defined by *WD*. Let *WFC* be the window framing clause of *WD*. Let *WFB1* be the <window frame bound 1> and let *WFB2* be the <window frame bound 2> contained in *WFC*.

i) If RANGE is specified, then:

1) In the following subrules, when performing addition or subtraction to combine a datetime and a year-month interval, if the result would raise the exception condition *data exception — datetime field overflow* because the <primary datetime field> DAY is not valid for the computed value of the <primary datetime field>s YEAR and MONTH, then the <primary datetime field> DAY is set to the last day that is valid for the <primary datetime field>s YEAR and MONTH, and no exception condition is raised.

2) Case:

NOTE 9 – In the following subrules, if *WFB1* specifies UNBOUNDED PRECEDING, then no rows are removed from *WF* by this step. *WFB1* may not be UNBOUNDED FOLLOWING.

A) If *WFB1* specifies <window frame preceding>, then let *V1P* be the value of the <unsigned value specification>.

I) If *V1P* is negative or the null value, then an exception condition is raised: *data exception — invalid preceding or following size in window function*.

II) Otherwise, let *SK* be the only <sort key> contained in the window ordering clause of *WD*. Let *VSK* be the value of *SK* for the current row.

Case:

1) If *VSK* is the null value and if NULLS LAST is specified or implied, then remove from *WF* all rows *R2* such that the value of *SK* in row *R2* is not the null value.

2) If *VSK* is not the null value, then:

a) If NULLS FIRST is specified or implied, then remove from *WF* all rows *R2* such that the value of *SK* in row *R2* is the null value.

b) Case:

i) If the <ordering specification> contained in the window ordering clause specifies DESC, then let *BOUND* be the value *VSK+V1P*. Remove from *WF* all rows *R2* such that the value of *SK* in row *R2* is greater than *BOUND*.

ii) Otherwise, let *BOUND* be the value *VSK–V1P*. Remove from *WF* all rows *R2* such that the value of *SK* in row *R2* is less than *BOUND*.

B) If *WFB1* specifies CURRENT ROW, then remove from *WF* all rows that are not peers of the current row and that precede the current row in the window ordering defined by *WD*.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

C) If *WFB1* specifies <window frame following>, then let *V1F* be the value of the <unsigned value specification>.

I) If *V1F* is negative or the null value, then an exception condition is raised: *data exception — invalid preceding or following size in window function*.

II) Otherwise, let *SK* be the only <sort key> contained in the window ordering clause of *WD*. Let *VSK* be the value of *SK* for the current row.

Case:

1) If *VSK* is the null value and if NULLS LAST is specified or implied, then remove from *WF* all rows *R2* such that the value of *SK* in row *R2* is not the null value.

2) If *VSK* is not the null value, then:

a) If NULLS FIRST is specified or implied, then remove from *WF* all rows *R2* such that the value of *SK* in row *R2* is the null value.

b) Case:

i) If the <ordering specification> contained in the window ordering clause specifies DESC, then let *BOUND* be the value *VSK–V1F*. Remove from *WF* all rows *R2* such that the value of *SK* in row *R2* is greater than *BOUND*.

ii) Otherwise, let *BOUND* be the value *VSK+V1F*. Remove from *WF* all rows *R2* such that the value of *SK* in row *R2* is less than *BOUND*.

3) Case:

NOTE 10 – In the following subrules, if *WFB2* specifies UNBOUNDED FOLLOWING, then no rows are removed from *WF* by this step. *WFB2* may not be UNBOUNDED PRECEDING.

A) If *WFB2* specifies <window frame preceding>, then let *V2P* be the value of the <unsigned value specification>.

I) If *V2P* is negative or the null value, then an exception condition is raised: *data exception — invalid preceding or following size in window function*.

II) Otherwise, let *SK* be the only <sort key> contained in the window ordering clause of *WD*. Let *VSK* be the value of *SK* for the current row.

Case:

1) If *VSK* is the null value and if NULLS FIRST is specified or implied, then remove from *WF* all rows *R2* such that the value of *SK* in row *R2* is not the null value.

2) If *VSK* is not the null value, then:

a) If NULLS LAST is specified or implied, then remove from *WF* all rows *R2* such that the value of *SK* in row *R2* is the null value.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.5 <window clause>

b) Case:

i) If the <ordering specification> contained in the window ordering clause specifies DESC, then let *BOUND* be the value *VSK+V2P*. Remove from *WF* all rows *R2* such that the value of *SK* in row *R2* is less than *BOUND*.

ii) Otherwise, let *BOUND* be the value *VSK–V2P*. Remove from *WF* all rows *R2* such that the value of *SK* in row *R2* is greater than *BOUND*.

B) If *WFB2* specifies CURRENT ROW, then remove from *WF* all rows following the current row in the ordering defined by *WD* that are not peers of the current row.

C) If *WFB2* specifies <window frame following>, then let *V2F* be the value of the <unsigned value specification>.

I) If *V2F* is negative or the null value, then an exception condition is raised: *data exception — invalid preceding or following size in window function*.

II) Otherwise, let *SK* be the only <sort key> contained in the window ordering clause of *WD*. Let *VSK* be the value of *SK* for the current row.

Case:

1) If *VSK* is the null value and if NULLS FIRST is specified or implied, then remove from *WF* all rows *R2* such that the value of *SK* in row *R2* is not the null value.

2) If *VSK* is not the null value, then:

a) If NULLS LAST is specified or implied, then remove from *WF* all rows *R2* such that the value of *SK* in row *R2* is the null value.

b) Case:

i) If the <ordering specification> contained in the <window order clause> specifies DESC, then let *BOUND* be the value *VSK–V2F*. Remove from *WF* all rows *R2* such that the value of *SK* in row *R2* is less than *BOUND*.

ii) Otherwise, let *BOUND* be the value *VSK+V2F*. Remove from *WF* all rows *R2* such that the value of *SK* in row *R2* is greater than *BOUND*.

ii) If ROWS is specified, then:

1) Case:

NOTE 11 – In the following subrules, if *WFB1* specifies UNBOUNDED PRECEDING, then no rows are removed from *WF* by this step. *WFB1* may not be UNBOUNDED FOLLOWING.

A) If *WFB1* specifies <window frame preceding>, then let *V1P* be the value of the <unsigned value specification>.

I) If *V1P* is negative or the null value, then an exception condition is raised:

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

data exception — invalid preceding or following size in window function.

II) Otherwise, remove from *WF* all rows that are more than *V1P* rows preceding the current row in the window ordering defined by *WD*.

B) If *WFB1* specifies CURRENT ROW, then remove from *WF* all rows that precede the current row in the window ordering defined by *WD*.

NOTE 12 – This step removes any peers of the current row that precede it in the implementation-dependent window ordering.

C) If *WFB1* specifies <window frame following>, then let *V1F* be the value of the <unsigned value specification>.

I) If *V1F* is negative or the null value, then an exception condition is raised: *data exception — invalid preceding or following size in window function.*

II) Otherwise, remove from *WF* all rows that precede the current row and all rows that are less than *V1F* rows following the current row in the window ordering defined by *WD*.

NOTE 13 – If *V1F* is zero, then the current row is not removed from *WF* by this step; otherwise, the current row is removed from *WF*.

2) Case:

NOTE 14 – In the following subrules, if *WFB2* specifies UNBOUNDED FOLLOWING, then no rows are removed from *WF* by this step. *WFB2* may not be UNBOUNDED PRECEDING.

A) If *WFB2* specifies <window frame preceding>, then let *V2P* be the value of the <unsigned value specification>.

I) If *V2P* is negative or the null value, then an exception condition is raised: *data exception — invalid preceding or following size in window function.*

II) Otherwise, remove from *WF* all rows that follow the current row and all rows that are less than *V2P* rows preceding the current row in the window ordering defined by *WD*.

NOTE 15 – If *V2P* is zero, then the current row is not removed from *WF* by this step; otherwise, the current row is removed from *WF*.

B) If *WFB2* specifies CURRENT ROW, then remove from *WF* all rows that follow the current row in the window ordering defined by *WD*.

NOTE 16 – This step removes any peers of the current row that follow it in the implementation-dependent window ordering.

C) If *WFB2* specifies <window frame following>, then let *V2F* be the value of the <unsigned value specification>.

I) If *V2F* is negative or the null value, then an exception condition is raised: *data exception — invalid preceding or following size in window function.*

II) Otherwise, remove from *WF* all rows that are more than *V2F* rows following the current row in the window ordering defined by *WD*.

iii) If <window framing exclusion> *WFE* is specified, then

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.5 <window clause>

Case:

1) If EXCLUDE CURRENT ROW is specified and the current row is still a member of *WF*, then remove the current row from *WF*.

2) If EXCLUDE GROUP is specified, then remove the current row and any peers of the current row from *WF*.

3) If EXCLUDE TIES is specified, then remove any rows other than the current row that are peers of the current row from *WF*.

   NOTE 17 – If the current row is already removed from *WF*, then it remains removed from *WF*.

NOTE 18 – If EXCLUDE NO OTHERS is specified, then no additional rows are removed from *WF* by this Rule.

Conformance Rules

1) Without Feature T611, "Elementary OLAP operators", conforming SQL language shall not specify <window specification>.

2) Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify a <window clause>.

3) Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify <existing window name>.

4) Without Feature T301, "Functional dependencies", if *T* is a grouped table, then each column reference contained in <window clause> that references a column of *T* shall be a reference to a grouping column of *T* or be contained in an aggregated argument of a <set function specification>.

5) Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify <window framing exclusion>.

6) Without Feature S024, "Enhanced structured types", the declared type of a partitioning column shall not be ST-ordered.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.6 <query specification>

Function

Specify a table derived from the result of a <table expression>.

Format

```
!! No additional Format items.
```

Syntax Rules

1) Insert after SR 10) Each column reference contained in a <window function> shall unambiguously reference a column of *T*.

2) Insert this SR If both of the following two conditions are satisfied, then *QS* is a *grouped, windowed query*:

   a) *T* is a grouped table.

   b) Some <derived column> simply contained in *QS* simply contains a <window function>.

3) Insert this SR A grouped, windowed query *GWQ* is transformed to an equivalent <query specification> as follows:

   a) If *GWQ* contains an <in-line window specification>, then apply the syntactic transformation specified in Subclause 6.3, "<window function>".

   b) If the <select list> of *GWQ* contains <asterisk> or <qualified asterisk>, then apply the syntactic transformations specified in Subclause 7.11, "<query specification>", in ISO/IEC 9075-2.

   c) Let *GWQ2* be the result of the preceding transformations, if any.

   d) Let *SL*, *FC*, *WC*, *GBC*, *HC*, and *WIC* be the <select list>, <from clause>, <where clause>, <group by clause>, <having clause>, and <window clause>, respectively, of *GWQ2*. If any of <where clause>, <group by clause>, or <having clause>, are missing, then let *WC*, *GBC*, and *HC*, respectively, be a zero-length string. Let *SQ* be the <set quantifier> immediately contained in the <query specification> of *GWQ2*, if any; otherwise, let *SQ* be a zero-length string.

      NOTE 19 – *GWQ2* can not lack a <window clause>, since the syntactic transformation of Subclause 6.3, "<window function>", will create one if there is not one in *GWQ* already.

   e) Let *N1* be the number of <set function specification>s simply contained in *GWQ2*.

   f) Let SFS_i, 1 (one) $\leq i \leq N1$, be an enumeration of the <set function specification>s simply contained in *GWQ2*.

   g) Let $SFSI_i$, 1 (one) $\leq i \leq N1$, be a list of <identifier>s that are distinct from each other and distinct from all <identifier>s contained in *GWQ2*.

   h) If *N1* = 0 (zero), then let *SFSL* be a zero-length string; otherwise, let *SFSL* be:

      SFS_1 AS $SFSI_1$, SFS_2 AS $SFSI_2$, . . . , SFS_{N1} AS $SFSI_{N1}$

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.6 <query specification>

i) Let *HCNEW* be obtained from *HC* by replacing each <set function specification> SFS_i by the corresponding <identifier> $SFSI_i$.

j) Let *N2* be the number of <column reference>s that are contained in *SL* or *WIC* without an intervening <subquery> or <set function specification>.

k) Let CR_j, 1 (one) $\leq j \leq$ *N2*, be an enumeration of the <column reference>s that are contained in *SL* or *WIC* without an intervening <subquery> or <set function specification>.

l) Let CRI_j, 1 (one) $\leq j \leq$ *N2*, be a list of <identifier>s that are distinct from each other, distinct from all identifiers in *GWQ2*, and distinct from all $SFSI_i$.

m) If *N2* = 0 (zero), then let *SFSL* be a zero-length string; otherwise, let *CRL* be:

CR_1 AS CRI_2, CR_2 AS CRI_2, . . . , CR_{N2} AS CRI_{N2}

n) Let *N3* be the number of <derived column>s simply contained in *SL* that do not specify <as clause>.

o) Let $DCOL_k$, 1 (one) $\leq k \leq$ *N3*, be the <derived column>s simply contained in *SL* that do not specify an <as clause>. For each k, let $COLN_k$ be the <column name> determined as follows:

i) If $DCOL_k$ is a single column reference, then let $COLN_k$ be the <column name> of the column designated by the column reference.

ii) Otherwise, let $COLN_k$ be an implementation-dependent <column name> that is not equivalent to the <column name> of any column, other than itself, of a table referenced by any <table reference> contained in the SQL-statement.

p) Let *SL2* be obtained from *SL* by replacing each <derived column> $DCOL_k$ by

$DCOL_k$ AS $COLN_k$

q) Let *GWQN* be an arbitrary <identifier>.

r) Let *SLNEW* be the <select list> obtained from *SL2* by replacing each simply contained <set function specification> SFS_i by $GWQN.SFSI_i$ and replacing each <column reference> CR_j that is contained without an intervening <subquery> or <set function specification> by $GWQN.CRI_j$.

s) Let *WICNEW* be the <window clause> obtained from *WIC* by replacing each <set function specification> SFS_i by $GWQN.SFSI_i$ and by replacing each <column reference> CR_j by $GWQN.CRI_j$.

t) If either *SFSL* or *CRL* is a zero-length string, then let *COMMA* be a zero-length string; otherwise, let *COMMA* be "," (a <comma>).

u) *GWQ* is equivalent to the following <query specification>:

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
SELECT SLNEW
FROM ( SELECT SQ SFSL COMMA CRL
       FC
       WC
       GBC
       HC ) AS GWQN
WICNEW
```

4) Insert after SR 11)c) The <query specification> contains a <window function> that specifies ROW_NUMBER or whose associated <window specification> specifies ROWS.

5) Replace SR 12) If <table expression> does not immediately contain a <group by clause> and <select list> contains either a <value expression> that contains a <set function specification> that has an aggregated argument that contains a reference to a column of *T* or a <value expression> that directly contains a <set function specification> that does not have an aggregated argument that contains an outer reference, then `GROUP BY ()` is implicit.

6) Replace SR 13) If *T* is a grouped table, then let *G* be the set of grouping columns of *T*. In each <value expression> contained in <select list> , each column reference that references a column of *T* shall reference some column *C* that is functionally dependent on *G* or shall be contained in an aggregated argument of a <set function specification>.

7) Insert after SR 16)a)viii) A <window function> whose <window function type> does not contain <rank function type>, ROW_NUMBER, or an <aggregate function> that simply contains COUNT.

Access Rules

No additional Access Rules.

General Rules

1) Replace introduction of GR 1)b)ii) If *T* has one or more groups, then each <value expression> is applied to each group of *T* yielding a table *TEMP* of *M* rows, where *M* is the number of groups in *T*. The *i*-th column of *TEMP* contains the values derived by the evaluation of the *i*-th <value expression>. When a <value expression> is applied to a given group of *T*, that group is the argument source of each <set function specification> in the <value expression>.

Conformance Rules

1) Replace CR 3) Without Feature T301, "Functional dependencies", if *T* is a grouped table, then in each <value expression>, each column reference that references a column of *T* shall reference a grouping column or be specified in an aggregated argument of a <set function specification>.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2001 – All rights reserved

--`,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8 Additional common elements

8.1 <aggregate function>

Function

Specify a value computed from a multiset of rows.

Format

```
<aggregate function> ::=
        COUNT <left paren> <asterisk> <right paren> [ <filter clause> ]
      | <general set function> [ <filter clause> ]
      | <binary set function> [ <filter clause> ]
      | <ordered set function> [ <filter clause> ]

<general set function> ::=
      <set function type> <left paren> [ <set quantifier> ]
          <value expression> <right paren>

<set function type> ::=
      <computational operation>

<computational operation> ::=
        AVG | MAX | MIN | SUM
      | EVERY | ANY | SOME
      | COUNT
      | STDDEV_POP | STDDEV_SAMP | VAR_SAMP | VAR_POP

<set quantifier> ::=
        DISTINCT
      | ALL

<filter clause> ::=
      FILTER <left paren> WHERE <search condition> <right paren>

<binary set function> ::=
      <binary set function type> <left paren>
        <dependent variable expression> <comma>
        <independent variable expression> <right paren>

<binary set function type> ::=
        COVAR_POP | COVAR_SAMP | CORR | REGR_SLOPE
      | REGR_INTERCEPT | REGR_COUNT | REGR_R2 | REGR_AVGX | REGR_AVGY
      | REGR_SXX | REGR_SYY | REGR_SXY

<dependent variable expression> ::=
      <numeric value expression>

<independent variable expression> ::=
      <numeric value expression>

<ordered set function> ::=
        <hypothetical set function>
```

© ISO/IEC 2001 – All rights reserved

--`,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.1 <aggregate function>

```
    | <inverse distribution function>

<hypothetical set function> ::=
    <rank function type> <left paren>
      <hypothetical set function value expression list> <right paren>
    <within group specification>

<within group specification> ::=
    WITHIN GROUP <left paren> ORDER BY
      <sort specification list> <right paren>

<hypothetical set function value expression list> ::=
    <value expression> [ { <comma> <value expression> }... ]

<inverse distribution function> ::=
    <inverse distribution function type> <left paren>
      <inverse distribution function argument> <right paren>
    <within group specification>

<inverse distribution function argument> ::=
    <numeric value expression>

<inverse distribution function type> ::=
      PERCENTILE_CONT
    | PERCENTILE_DISC
```

Syntax Rules

1) Let *AF* be the <aggregate function>.

2) If STDDEV_POP, STDDEV_SAMP, VAR_POP or VAR_SAMP is specified, then <set quantifier> shall not be specified.

3) In a <general set function>, if <set quantifier> is not specified, then ALL is implicit.

4) The argument source of an <aggregate function> is

   Case:

   a) If *AF* is immediately contained in a <set function specification>, then a table or group of a grouped table as specified in Subclause 7.4, "<having clause>", and Subclause 7.6, "<query specification>".

   b) Otherwise, the multiset of rows in the current row's window frame defined by the window structure descriptor identified by the <window aggregate> that simply contains *AF*, as defined in Subclause 7.5, "<window clause>".

5) Let *T* be the argument source of *AF*.

6) If COUNT is specified, then the declared type of the result is exact numeric with implementation-defined precision and scale of 0 (zero).

7) If <general set function> is specified, then:

   a) The <value expression> shall not contain a <window function>.

   b) Let *DT* be the declared type of the <value expression>.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

c) If *AF* specifies a <general set function> whose <set quantifier> is DISTINCT, then *DT* shall not be LOB-ordered, array-ordered, UDT-EC-ordered, or UDT-NC-ordered.

d) If *AF* specifies a <set function type> that is MAX or MIN, then *DT* shall not be LOB-ordered, array-ordered, UDT-EC-ordered, or UDT-NC-ordered.

e) If EVERY, ANY, or SOME is specified, then *DT* shall be boolean and the declared type of the result is boolean.

f) If MAX or MIN is specified, then the declared type of the result is *DT*.

g) If SUM or AVG is specified, then:

i) *DT* shall be a numeric type or an interval type.

ii) If SUM is specified and *DT* is exact numeric with scale *S*, then the declared type of the result is exact numeric with implementation-defined precision and scale *S*.

iii) If AVG is specified and *DT* is exact numeric, then the declared type of the result is exact numeric with implementation-defined precision not less than the precision of *DT* and implementation-defined scale not less than the scale of *DT*.

iv) If *DT* is approximate numeric, then the declared type of the result is approximate numeric with implementation-defined precision not less than the precision of *DT*.

v) If *DT* is interval, then the declared type of the result is interval with the same precision as *DT*.

h) If VAR_POP or VAR_SAMP is specified, then the declared type of the result is approximate numeric with implementation-defined precision not less than the precision of *DT*.

i) `STDDEV_POP(`*X*`)` is equivalent to `SQRT(VAR_POP(`*X*`))`.

j) `STDDEV_SAMP(`*X*`)` is equivalent to `SQRT(VAR_SAMP(`*X*`))`.

k) If the declared type of the result is character string, then the collating sequence and the coercibility characteristic are determined as in Subclause 4.2.3, "Rules determining collating sequence usage", in ISO/IEC 9075-2:1999.

8) A <filter clause> shall not contain a <subquery>, a <window function>, or an outer reference.

9) If <binary set function> is specified, then:

a) The <dependent variable expression> *DVE* and the <independent variable expression> *IVE* shall not contain a <window function>.

b) Let *DTDVE* be the declared type of *DVE* and let *DTIVE* be the declared type of *IVE*.

c) Case:

i) The declared type of REGR_COUNT is exact numeric with implementation-defined precision and scale of 0 (zero).

ii) Otherwise, the declared type of the result is approximate numeric with implementation-defined precision not less than the precision of the declared type of *DVE* and not less than the precision of the declared type of *IVE*.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.1 <aggregate function>

10) If <hypothetical set function> is specified, then:

a) The <hypothetical set function> shall not contain a <window function>, a <set function specification>, or a <subquery>.

b) If a <value expression> simply contained in the <hypothetical set function value expression list> or the <sort specification list> contains a column reference that is an outer reference, then that outer reference shall be the only column reference contained in the <value expression>.

c) The number of <value expression>s simply contained in <hypothetical set function value expression list> shall be the same as the number of <sort key>s simply contained in the <sort specification list>.

d) The declared type of each <value expression> simply contained in the <hypothetical set function value expression list> shall be the same as the declared type of the corresponding <sort key> simply contained in the <sort specification list>, with the same precision and scale, if applicable. If this is a character type, then the character repertoires shall be the same, and the coercibility of the <value expression> shall not be Explicit. The collation is determined by Table 1, "Collating coercibility rules for monadic operators", in ISO/IEC 9075-2, using the coercibility and collation of the <sort key>.

e) Case:

i) If RANK or DENSE_RANK is specified, then the declared type of the result is exact numeric with implementation-defined precision and with scale 0 (zero).

ii) Otherwise, the declared type of the result is approximate numeric with implementation-defined precision.

11) If <inverse distribution function> is specified, then:

a) The <within group specification> shall contain a single <sort specification>.

b) The <inverse distribution function> shall not contain a <window function>, a <set function specification>, or a <subquery>.

c) If the <inverse distribution function argument> or the <sort specification> contains a column reference that is an outer reference, then that outer reference shall be the only column reference contained in the <inverse distribution function argument> or <sort specification>.

d) Let *DT* be the declared type of the <value expression> simply contained in the <sort specification>.

e) If PERCENTILE_CONT is specified, then *DT* shall be numeric or interval.

f) The declared type of the result is

Case:

i) If *DT* is numeric, then approximate numeric with implementation-defined precision.

ii) If *DT* is interval, then *DT*.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Access Rules

None.

General Rules

1) Case:

   a) If <filter clause> is specified, then the <search condition> is applied to each row of *T*. Let *T1* be the multiset of rows of *T* for which the result of the <search condition> is true.

   b) Otherwise, let *T1* be *T*.

2) If COUNT(*) is specified, then the result is the cardinality of *T1*.

3) If <general set function> is specified, then:

   a) Let *TX* be the single-column table that is the result of applying the <value expression> to each row of *T1* and eliminating null values. If one or more null values are eliminated, then a completion condition is raised: *warning — null value eliminated in set function*.

   b) Case:

      i) If DISTINCT is specified, then let *TXA* be the result of eliminating redundant duplicate values from *TX*, using the comparison rules specified in Subclause 8.2, "<comparison predicate>", in ISO/IEC 9075-2, to identify the redundant duplicate values.

      ii) Otherwise, let *TXA* be *TX*.

   c) Let *N* be the cardinality of *TXA*.

   d) Case:

      i) If the <general set function> COUNT is specified, then the result is *N*.

      ii) If *TXA* is empty, then the result is the null value.

      iii) If AVG is specified, then the result is the average of the values in *TXA*.

      iv) If MAX or MIN is specified, then the result is respectively the maximum or minimum value in *TXA*. These results are determined using the comparison rules specified in Subclause 8.2, "<comparison predicate>", in ISO/IEC 9075-2. If *DT* is a user-defined type and the comparison of two values in *TXA* results in *unknown*, then the maximum or minimum of *TXA* is implementation-dependent.

      v) If SUM is specified, then the result is the sum of the values in *TXA*. If the sum is not within the range of the declared type of the result, then an exception condition is raised: *data exception — numeric value out of range*.

      vi) If EVERY is specified, then

      Case:

         1) If the value of some element of TXA is *false*, then the result is *false*.

         2) Otherwise, the result is *true*.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.1 <aggregate function>

vii) If ANY or SOME is specified, then

Case:

1) If the value of some element of *TXA* is *true* , then the result is *true* .

2) Otherwise, the result is *false* .

viii) If VAR_POP or VAR_SAMP is specified, then let *S1* be the sum of values in the column of *TXA*, and *S2* be the sum of the squares of the values in the column of *TXA*.

1) If VAR_POP is specified, then the result is `(S2-S1*S1/N)/N`.

2) If VAR_SAMP is specified, then:

A) If *N* is 1 (one), then the result is the null value.

B) Otherwise, the result is `(S2-S1*S1/N)/(N-1)`

4) If <binary set function type> is specified, then:

a) Let *TXA* be the two-column table that is the result of applying the <dependent variable expression> and the <independent variable expression> to each row of *T1* and eliminating each row in which either <dependent variable expression> or <independent variable expression> is the null value. If one or more null values are eliminated, then a completion condition is raised: *warning — null value eliminated in set function.*

b) Let *N* be the cardinality of *TXA*, let *SUMX* be the sum of the column of values of <independent variable expression>, let *SUMY* be the sum of the column of values of <dependent variable expression>, let *SUMX2* be the sum of the squares of values in the <independent variable expression> column, let *SUMY2* be the sum of the squares of values in the <dependent variable expression> column, and let *SUMXY* be the sum of the row-wise products of the value in the <independent variable expression> column times the value in the <dependent variable expression> column.

c) Case:

i) If REGR_COUNT is specified, then the result is *N*.

ii) If *N* is 0 (zero), then the result is the null value.

iii) If REGR_SXX is specified, then the result is `(SUMX2-SUMX*SUMX/N)`.

iv) If REGR_SYY is specified, then the result is `(SUMY2-SUMY*SUMY/N)`.

v) If REGR_SXY is specified, then the result is `(SUMXY-SUMX*SUMY/N)`.

vi) If REGR_AVGX is specified, then the result is `SUMX/N`.

vii) If REGR_AVGY is specified, then the result is `SUMY/N`.

viii) If COVAR_POP is specified, then the result is `(SUMXY-SUMX*SUMY/N)/N`.

ix) If COVAR_SAMP is specified, then

Case:

1) If *N* is 1 (one), then the result is the null value.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) Otherwise, the result is `(SUMXY-SUMX*SUMY/N)/(N-1)`

x) If CORR is specified, then

Case:

1) If *N*SUMX2* equals *SUMX*SUMX*, then the result is the null value.

NOTE 20 – In this case, all remaining values of <independent variable expression> are equal, and consequently the <independent variable expression> does not correlate with the <dependent variable expression>.

2) If *N*SUMY2* equals *SUMY*SUMY*, then the result is the null value.

NOTE 21 – In this case, all remaining values of <dependent variable expression> are equal, and consequently the <dependent variable expression> does not correlate with the <independent variable expression>.

3) Otherwise, the result is `SQRT(POWER(N*SUMXY-SUMX*SUMY,2)/((N*SUMX2-SUMX*SUMX)*(N*SUMY2-SUMY*SUMY)))`. If the exponent of the approximate mathematical result of the operation is not within the implementation-defined exponent range for the result data type, then the result is the null value.

xi) If REGR_R2 is specified, then

Case:

1) If *N*SUMX2* equals *SUMX*SUMX*, then the result is the null value.

NOTE 22 – In this case, all remaining values of <independent variable expression> are equal, and consequently the least-squares fit line would be vertical, or, if $N = 1$ (one), there is no uniquely determined least-squares-fit line.

2) If *N*SUMY2* equals *SUMY*SUMY*, then the result is 1 (one).

NOTE 23 – In this case, all remaining values of <dependent variable expression> are equal, and consequently the least-squares fit line is horizontal.

3) Otherwise, the result is `POWER(N*SUMXY-SUMX*SUMY,2)/((N*SUMX2-SUMX*SUMX)*(N*SUMY2-SUMY*SUMY))`. If the exponent of the approximate mathematical result of the operation is not within the implementation-defined exponent range for the result data type, then the result is the null value.

xii) If REGR_SLOPE(*Y*, *X*) is specified, then

Case:

1) If *N*SUMX2* equals *SUMX*SUMX*, then the result is the null value.

NOTE 24 – In this case, all remaining values of <independent variable expression> are equal, and consequently the least-squares fit line would be vertical, or, if $N = 1$ (one), then there is no uniquely determined least-squares-fit line.

2) Otherwise, the result is `(N*SUMXY-SUMX*SUMY)/(N*SUMX2-SUMX*SUMX)`. If the exponent of the approximate mathematical result of the operation is not within the implementation-defined exponent range for the result data type, then the result is the null value.

xiii) If REGR_INTERCEPT is specified, then

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

8.1 <aggregate function>

Case:

1) If *N*SUMX2* equals *SUMX*SUMX*, then the result is the null value.
NOTE 25 – In this caes, all remaining values of <independent variable expression> are equal, and consequently the least-squares fit line would be vertical, or, if N = 1 (one), then there is no uniquely determined least-squares-fit line.

2) Otherwise, the result is `(SUMY*SUMX2-SUMX*SUMXY) / (N*SUMX2-SUMX*SUMX)`. If the exponent of the approximate mathematical result of the operation is not within the implementation-defined exponent range for the result data type, then the result is the null value.

5) If <hypothetical set function> is specified, then

a) Let *WIFT* be the <rank function type>.

b) Let *TNAME* be an implementation-dependent name for *T1*.

c) Let K be the number of <value expression>s simply contained in <hypothetical set function value expression list>.

d) Let VE_1, . . . , VE_K be the <value expression>s simply contained in the <hypothetical set function value expression list>.

e) Let *WIFTVAL*, *MARKER* and CN_1, . . . , CN_K be distinct implementation-dependent column names.

f) Let SP_1, . . . , SP_K be the <sort specification>s simply contained in the <sort specification list>. For each i, let WSP_i be the <sort specification> obtained from SP_i by replacing the <sort key> with CN_i.

g) The result is the result of the <scalar subquery>

```
( SELECT WIFTVAL
   FROM ( SELECT MARKER, WIFT() OVER
              ( ORDER BY WSP1,  . . .  , WSPK )
          FROM ( SELECT 0, SK1,  . . .  , SKK
                 FROM TNAME
                   UNION
                      VALUES ( 1, VE1,  . . .  , VEK ) )
                   AS TXNAME ( MARKER, CN1,  . . .  , CNK  )
        ) AS TEMPTABLE ( MARKER, WIFTVAL )
  WHERE MARKER = 1 )
```

6) If <inverse distribution function> is specified, then

a) Let *NVE* be the value of the <inverse distribution function argument>.

b) If *NVE* is the null value, then the result is the null value.

c) If *NVE* is less than 0 (zero) or greater than 1 (one), then an exception condition is raised: *data exception — numeric value out of range.*

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

d) Let *TXA* be the single-column table that is the result of applying the <value expression> simply contained in the <sort specification> to each row of *T1* and eliminating null values. If one or more null values are eliminated, then a completion condition is raised: *warning — null value eliminated in set function*. *TXA* is ordered by the <sort specification> as specified in the General Rules of Subclause 8.2, "<sort specification list>".

e) Let *TXANAME* be an implementation-dependent name for *TXA*.

f) Let *TXCOLNAME* be an implementation-dependent column name for the column of *TXA*.

g) Let *WSP* be obtained from the <sort specification> by replacing the <sort key> with *TXCOLNAME*.

h) Case:

i) If PERCENTILE_CONT is specified, then:

1) Let *ROW0* be the greatest exact numeric value with scale 0 (zero) that is less than or equal to *NVE**(*N*−1). Let *ROWLIT0* be a <literal> representing *ROW0*.

2) Let *ROW1* be the least exact numeric value with scale 0 (zero) that is greater than or equal to *NVE**(*N*−1). Let *ROWLIT1* be a <literal> representing *ROW1*.

3) Let *FACTOR* be a <approximate numeric literal> representing *NVE**(*N*−1)−*ROW0*.

4) The result is the result of the <scalar subquery>

```
( WITH TEMPTABLE(X, Y) AS
       ( SELECT ROW_NUMBER()
                  OVER (ORDER BY WSP) - 1,
                TXCOLNAME
         FROM TXANAME )
SELECT CAST ( T0.Y + FACTOR * (T1.Y - T0.Y) AS DT )
FROM TEMPTABLE T0, TEMPTABLE T1
WHERE T0.ROWNUMBER = ROWLIT0
  AND T1.ROWNUMBER = ROWLIT1 )
```

NOTE 26 – Although ROW_NUMBER is nondeterministic, the values of T0.Y and T1.Y are determined by this expression. Note that the only column of *TXA* is completely ordered by *WSP*. If *NVE**(*N*−1) is a whole number, then the rows selected from T0 and T1 are the same and the result is just T0.Y. Otherwise, the subquery performs a linear interpolation between the two consecutive values whose row numbers in the ordered set, seen as proportions of the whole, bound the argument of the PERCENTILE_CONT operator.

ii) If PERCENTILE_DISC is specified, then

1) If the <ordering specification> simply contained in *WSP* is DESC, then let *MAXORMIN* be MAX; otherwise let *MAXORMIN* be MIN.

2) Let *NVELIT* be a <literal> representing the value of *NVE*.

3) The result is the result of the <scalar subquery>

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.1 <aggregate function>

```
( SELECT MAXORMIN (TXCOLNAME)
  FROM ( SELECT TXCOLNAME,
                CUME_DIST() OVER (ORDER BY WSP)
  FROM TXANAME ) AS TEMPTABLE (TXCOLNAME, CUMEDIST)
  WHERE CUMEDIST >= NVELIT )
```

Conformance Rules

1) Without Feature T031, "BOOLEAN data type", conforming SQL language shall not contain a <set function type> that specifies EVERY, ANY, or SOME.

2) Without Feature F561, "Full value expressions", or Feature F801, "Full set function", if a <general set function> specifies DISTINCT, then the <value expression> shall be a column reference.

3) Without Feature F441, "Extended set function support", if a <general set function> specifies or implies ALL, then COUNT shall not be specified.

4) Without Feature F441, "Extended set function support", if a <general set function> specifies or implies ALL, then the <value expression> shall contain a column reference that references a column of *T*.

5) Without Feature F441, "Extended set function support", if a <binary set function> is specified, then either the <dependent variable expression> or the <independent variable expression> shall contain a column reference that references a column of *T*.

6) Without Feature F441, "Extended set function support", if the <value expression> simply contained in a <general set function> contains a column reference that is an outer reference, then the <value expression> shall be a column reference.

7) Without Feature F441, "Extended set function support", if the <numeric value expression> simply contained in <dependent variable expression> or <independent variable expression> contains a column reference that is an outer reference, then the <numeric value expression> shall be a column reference.

8) Without Feature F441, "Extended set function support", no column reference contained in an <aggregate function> shall reference a column derived from a <value expression> that generally contains an <aggregate function> *SFS2* without an intervening <routine invocation>.

9) Without Feature S024, "Enhanced structured types", in a <general set function>, if MAX or MIN is specified, then the <value expression> shall not be an ST-ordered declared type.

10) Without Feature S024, "Enhanced structured types", the declared type of a <general set function> shall not be structured type.

11) Without Feature T621, "Enhanced numeric functions", conforming SQL language shall not specify STDDEV_POP, STDDEV_SAMP, VAR_POP, VAR_SAMP, or <binary set function type>.

12) Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify <hypothetical set function> or <inverse distribution function>.

13) Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify <filter clause>.

14) Without Feature S024, "Enhanced structured types", if a <set quantifier> of DISTINCT is specified, then the <value expression> shall not be of an ST-ordered declared type.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

8.2 <sort specification list>

Function

Specify a sort order.

Format

```
<sort specification list> ::=
       <sort specification> [ { <comma> <sort specification> }... ]

<sort specification> ::=
       <sort key>
           [ <ordering specification> ] [ <null ordering> ]

<sort key> ::=
       <value expression>

<ordering specification> ::=
         ASC
       | DESC

<null ordering> ::=
         NULLS FIRST
       | NULLS LAST
```

Syntax Rules

1) Let *DT* be the declared type of the <value expression> immediately contained in the <sort key> contained in a <sort specification>. *DT* shall not be LOB-ordered, array-ordered, UDT-EC-ordered, or UDT-NC-ordered.

2) If <null ordering> is not specified, then an implementation-defined <null ordering> is implicit. The implementation-defined default for <null ordering> shall not depend on the context outside of <sort specification list>.

Access Rules

None.

General Rules

1) A <sort specification list> defines an ordering of rows, as follows:

   a) Let N be the number of <sort specification>s.

   b) Let K_i, 1 (one) $\leq i \leq N$, be the <sort key> contained in the i-th <sort specification>.

   c) Each <sort specification> specifies the *sort direction* for the corresponding sort key K_i. If DESC is not specified in the i-th <sort specification>, then the sort direction for K_i is ascending and the applicable <comp op> is the <less than operator>. Otherwise, the sort direction for K_i is descending and the applicable <comp op> is the <greater than operator>.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.2 <sort specification list>

d) Let P be any row of the multiset of rows to be ordered, and let Q be any other row of the same multiset of rows.

e) Let PV_i and QV_i be the values of K_i in P and Q, respectively. The relative position of rows P and Q in the result is determined by comparing PV_i and QV_i according to the rules of Subclause 8.2, "<comparison predicate>", in ISO/IEC 9075-2, where the <comp op> is the applicable <comp op> for K_i, with the following special treatment of null values.

Case:

i) If PV_i and QV_i are both null, then they are considered equal to each other.

ii) If PV_i is null and QV_i is not null, then

Case:

1) If NULLS FIRST is specified or implied, then PV_i <comp op> QV_i is considered to be *true* .

2) If NULLS LAST is specified or implied, then PV_i <comp op> QV_i is considered to be *false* .

iii) If PV_i is not null and QV_i is null, then

Case:

1) If NULLS FIRST is specified or implied, then PV_i <comp op> QV_i is considered to be *false* .

2) If NULLS LAST is specified or implied, then PV_i <comp op> QV_i is considered to be *true* .

f) PV_i is said to *precede* QV_i if the value of the <comparison predicate> "PV_i <comp op> QV_i" is *true* for the applicable <comp op>.

g) If PV_i and QV_i are not null and the result of "PV_i <comp op> QV_i" is *unknown* , then the relative ordering of PV_i and QV_i is implementation-dependent.

h) The relative position of row P is before row Q if PV_n precedes QV_n for some n, 1 (one) $\leq n \leq N$, and PV_i is not distinct from QV_i for all $i < n$.

i) Two rows that are not distinct with respect to the <sort specification>s are said to be *peers* of each other. The relative ordering of peers is implementation-dependent.

Conformance Rules

1) Without Feature T611, "Elementary OLAP operators", conforming SQL language shall not specify <null ordering>.

2) Without Feature S024, "Enhanced structured types", the declared type of the <value expression> immediately contained in the <sort key> shall not be ST-ordered.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9 Schema definition and manipulation

9.1 <drop table constraint definition>

Function

Destroy a constraint on a table.

Format

```
!! No additional Format items.
```

Syntax Rules

1) Replace SR 4) If V is a view that contains a <query specification> QS that contains a column reference to a column C in its <select list> that is not contained in an aggregated argument of a <set function specification>, and if G is the set of columns defined by the <grouping column reference list> of QS, and if the table constraint TC is needed to conclude that $G \rightarrow C$ is a known functional dependency in QS, then V is said to be *dependent on TC*.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

9.2 <drop user-defined ordering statement>

Function

Destroy a user-defined ordering method.

Format

```
!! No additional Format items
```

Syntax Rules

1) Insert this subrule after SR 4)jj) A <sort key> that simply contains a <value expression> whose declared type is *S*-ordered.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10 SQL-client modules

10.1 Calls to an <externally-invoked procedure>

Function

Define the call to an <externally-invoked procedure> by an SQL-agent.

Syntax Rules

1) Insert into SR 2)e)

```
DATA_EXCEPTION_INVALID_ARGUMENT_FOR_NATURAL_LOGARITHM:
        constant SQLSTATE_TYPE :="2201E";
DATA_EXCEPTION_INVALID_ARGUMENT_FOR_POWER_FUNCTION:
        constant SQLSTATE_TYPE :="2201F";
DATA_EXCEPTION_INVALID_ARGUMENT_FOR_WIDTH_BUCKET_FUNCTION:
        constant SQLSTATE_TYPE :="2201G";
DATA_EXCEPTION_INVALID_PRECEDING_OR_FOLLOWING_SIZE_IN_WINDOW_FUNCTION:
        constant SQLSTATE_TYPE :="22013";
```

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11 Data manipulation

11.1 <declare cursor>

Function

Define a cursor

Format

```
!! No additional Format items
```

Delete the definition of <sort specification list>

Delete the definition of <sort specification>

Delete the definition of <sort key>

Delete the definition of <ordering specification>

NOTE 27 – The deleted definitions are now placed in Subclause 8.2, "<sort specification list>".

Syntax Rules

1) Replace SR 18)f)i)2)A)II) *QS* shall not immediately contain the <set quantifier> DISTINCT or directly contain one or more <set function specification>s.

2) Delete SR 19)

3) Delete SR 22)

NOTE 28 – The deleted Syntax Rules are now found in Subclause 8.2, "<sort specification list>".

Access Rules

No additional Access Rules

General Rules

1) Replace GR 2) If an <order by clause> is specified, then the ordering of rows of the result is determined by the <sort specification list>. The result table specified by the <cursor specification> is *TS* with all extended sort key columns (if any) removed.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2001 1 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.2 <select statement: single row>

Function

Retrieve values from a specified row of a table.

Format

```
!! No additional Format items
```

Syntax Rules

1) Delete SR 5)

2) Insert after SR 6) The <select statement: single row> is possibly nondeterministic if *S* is possibly nondeterministic.

Access Rules

No additional Access Rules

General Rules

No additional General Rules

Conformance Rules

No additional Conformance Rules.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12 Dynamic SQL

12.1 <prepare statement>

Function

Retrieve values from a specified row of a table.

Format

```
!! No additional Format items
```

Syntax Rules

No additional Syntax Rules

Access Rules

No additional Access Rules

General Rules

1) Insert after GR 6)a)xxvii) If *DP* is contained in a <window frame preceding> or a <window frame following> contained in a <window specification> *WS*, then

   Case:

   1) If *WS* specifies ROWS, then *DT* is NUMERIC (*MP*, 0).

   2) Otherwise, let *SDT* be the data type of the single <sort key> contained in *WS*.

      Case:

      A) If *SDT* is a numeric type, then *DT* is *SDT*.

      B) If *SDT* is DATE, then *DT* is INTERVAL DAY.

      C) If *SDT* is TIME(*P*) WITHOUT TIME ZONE or TIME(*P*) WITH TIME ZONE, then *DT* is INTERVAL HOUR TO SECOND(*P*).

      D) If *SDT* is TIMESTAMP(*P*) WITHOUT TIME ZONE or TIMESTAMP(*P*) WITH TIME ZONE, then DT is INTERVAL DAY TO SECOND(*P*).

      E) If *SDT* is an interval type, then *DT* is *SDT*.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

13 Information Schema

13.1 Definition of SQL built-in functions

Function

Define the SQL built-in functions.

Definition

Add the following Definitions

```
CREATE FUNCTION "CEIL" (
    N       numeric_type )
    RETURNS NUMERIC ( MP , 0 )
    SPECIFIC CEIL
    RETURN CEIL ( N ) ;

CREATE FUNCTION "CEILING" (
    N       numeric_type )
    RETURNS NUMERIC ( MP, 0 )
    SPECIFIC CEILING
    RETURN CEILING ( N ) ;

CREATE FUNCTION "EXP" (
    N       numeric_type )
    RETURNS DOUBLE PRECISION
    SPECIFIC EXP
    RETURN EXP ( N ) ;

CREATE FUNCTION "FLOOR" (
    N       numeric_type )
    RETURNS NUMERIC ( MP, 0)
    SPECIFIC FLOOR
    RETURN FLOOR ( N ) ;

CREATE FUNCTION "LN" (
    N       numeric_type )
    RETURNS DOUBLE PRECISION
    SPECIFIC LN
    RETURN LN ( N ) ;

CREATE FUNCTION "POWER" (
    M       numeric_type,
    N       numeric_type)
    RETURNS DOUBLE PRECISION
    SPECIFIC POWER
    RETURN POWER ( M, N ) ;

CREATE FUNCTION "SQRT" (
    N       numeric_type )
    RETURNS DOUBLE PRECISION
    SPECIFIC SQRT
    RETURN SQRT ( N ) ;
```

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

13.1 Definition of SQL built-in functions

```
CREATE FUNCTION "WIDTH_BUCKET" (
    A       numeric_type,
    B       numeric_type,
    C       numeric_type,
    D       numeric_type )
    RETURNS NUMERIC (MP, 0)
    SPECIFIC WIDTH_BUCKET
    RETURN WIDTH_BUCKET ( A, B, C, D) ;

GRANT EXECUTE ON SPECIFIC FUNCTION INFORMATION_SCHEMA."CEIL"
TO PUBLIC;

GRANT EXECUTE ON SPECIFIC FUNCTION INFORMATION_SCHEMA."CEILING"
TO PUBLIC;

GRANT EXECUTE ON SPECIFIC FUNCTION INFORMATION_SCHEMA."EXP"
TO PUBLIC;

GRANT EXECUTE ON SPECIFIC FUNCTION INFORMATION_SCHEMA."FLOOR"
TO PUBLIC;

GRANT EXECUTE ON SPECIFIC FUNCTION INFORMATION_SCHEMA."LN"
TO PUBLIC;

GRANT EXECUTE ON SPECIFIC FUNCTION INFORMATION_SCHEMA."POWER"
TO PUBLIC;

GRANT EXECUTE ON SPECIFIC FUNCTION INFORMATION_SCHEMA."SQRT"
TO PUBLIC;

GRANT EXECUTE ON SPECIFIC FUNCTION INFORMATION_SCHEMA."WIDTH_BUCKET"
TO PUBLIC;
```

Description

1) Insert this Description *numeric_type* is the implementation-defined numeric type that has the highest type precedence among all numeric types supported by that implementation.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

14 Status codes

14.1 SQLSTATE

Table 2—SQLSTATE class and subclass values

Category	Condition	Class	Subcondition	Subclass
	All alternatives from ISO/IEC 9075-2			
	All alternatives from ISO/IEC 9075-5			
X	data exception	22	invalid argument for natural logarithm	01E
			invalid argument for power function	01F
			invalid argument for width bucket function	01G
			invalid preceding or following size in window function	013

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15 Conformance

15.1 General conformance requirements

Insert this paragraph The list of features that are implied by other features is shown in Table 3, "Implied feature relationships".

Table 3—Implied feature relationships

Feature ID	Feature Description	Implied Feature ID	Implied Feature Description
T612	Advanced OLAP operators	T611	Elementary OLAP operators

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex A
(informative)

SQL conformance summary

Replicated paragraph The contents of this Annex summarizes all Conformance Rules, ordered by Feature ID and by Subclause.

1) Specifications for Feature F441, "Extended set function support":

   a) Subclause 8.1, "<aggregate function>":

      i) Without Feature F441, "Extended set function support", if a <binary set function> is specified, then either the <dependent variable expression> or the <independent variable expression> shall contain a column reference that references a column of *T*.

      ii) Without Feature F441, "Extended set function support", if a <general set function> specifies or implies ALL, then the <value expression> shall contain a column reference that references a column of *T*.

      iii) Without Feature F441, "Extended set function support", no column reference contained in an <aggregate function> shall reference a column derived from a <value expression> that generally contains an <aggregate function> *SFS2* without an intervening <routine invocation>.

      iv) Without Feature F441, "Extended set function support", if the <numeric value expression> simply contained in <dependent variable expression> or <independent variable expression> contains a column reference that is an outer reference, then the <numeric value expression> shall be a column reference.

      v) Without Feature F441, "Extended set function support", if the <value expression> simply contained in a <general set function> contains a column reference that is an outer reference, then the <value expression> shall be a column reference.

      vi) Without Feature F441, "Extended set function support", if a <general set function> specifies or implies ALL, then COUNT shall not be specified.

2) Specifications for Feature F561, "Full value expressions":

   a) Subclause 8.1, "<aggregate function>":

      i) Without Feature F561, "Full value expressions", or Feature F801, "Full set function", if a <general set function> specifies DISTINCT, then the <value expression> shall be a column reference.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3) Specifications for Feature S024, "Enhanced structured types":

   a) Subclause 7.5, "<window clause>":

      i) Without Feature S024, "Enhanced structured types", the declared type of a partitioning column shall not be ST-ordered.

   b) Subclause 8.1, "<aggregate function>":

      i) Without Feature S024, "Enhanced structured types", the declared type of a <general set function> shall not be structured type.

      ii) Without Feature S024, "Enhanced structured types", in a <general set function>, if MAX or MIN is specified, then the <value expression> shall not be an ST-ordered declared type.

      iii) Without Feature S024, "Enhanced structured types", if a <set qualifier> of DISTINCT is specified, then the <value expression> shall not be of an ST-ordered declared type.

   c) Subclause 8.2, "<sort specification list>":

      i) Without Feature S024, "Enhanced structured types", the declared type of the <value expression> immediately contained in the <sort key> shall not be ST-ordered.

4) Specifications for Feature T031, "BOOLEAN data type":

   a) Subclause 8.1, "<aggregate function>":

      i) Without Feature T031, "BOOLEAN data type", conforming SQL language shall not contain a <set function type> that specifies EVERY, ANY, or SOME.

5) Specifications for Feature T301, "Functional dependencies":

   a) Subclause 7.4, "<having clause>":

      i) Replace CR 2) Without Feature T301, "Functional dependencies", each column reference contained in a <subquery> in the <search condition> that references a column of *T* shall be one of the following:

         a) An unambiguous reference to a grouping column of *T*.

         b) Contained in an aggregated argument of a <set function specification>.

   b) Subclause 7.5, "<window clause>":

      i) Without Feature T301, "Functional dependencies", if *T* is a grouped table, then each column reference contained in <window clause> that references a column of *T* shall be a reference to a grouping column of *T* or be contained in an aggregated argument of a <set function specification>.

   c) Subclause 7.6, "<query specification>":

      i) Replace CR 3) Without Feature T301, "Functional dependencies", if *T* is a grouped table, then in each <value expression>, each column reference that references a column of *T* shall reference a grouping column or be specified in an aggregated argument of a <set function specification>.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6) Specifications for Feature T611, "Elementary OLAP operators":

   a) Subclause 6.3, "<window function>":

      i) Without Feature T611, "Elementary OLAP operators", conforming SQL language shall not specify a <window function>.

   b) Subclause 6.4, "<value expression>":

      i) Insert this CR Without Feature T611, "Elementary OLAP operators", a <value expression> shall not be a <window function>.

   c) Subclause 7.5, "<window clause>":

      i) Without Feature T611, "Elementary OLAP operators", conforming SQL language shall not specify <window specification>.

   d) Subclause 8.2, "<sort specification list>":

      i) Without Feature T611, "Elementary OLAP operators", conforming SQL language shall not specify <null ordering>.

7) Specifications for Feature T612, "Advanced OLAP operators":

   a) Subclause 5.2, "Names and identifiers":

      i) Insert this CR Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify <window name>.

   b) Subclause 6.2, "<numeric value function>":

      i) Insert this CR Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify <width bucket function>.

   c) Subclause 6.3, "<window function>":

      i) Without Feature T612, "Advanced OLAP operators", if ROW_NUMBER is specified, then the window ordering clause of *WDX* shall be present.

      ii) Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify PERCENT_RANK or CUME_DIST.

      iii) Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify <window name>.

   d) Subclause 7.1, "<table expression>":

      i) Insert this CR Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify a <window clause>.

   e) Subclause 7.5, "<window clause>":

      i) Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify a <window clause>.

      ii) Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify <existing window name>.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

iii) Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify <window framing exclusion>.

f) Subclause 8.1, "<aggregate function>":

i) Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify <hypothetical set function> or <inverse distribution function>.

ii) Without Feature T612, "Advanced OLAP operators", conforming SQL language shall not specify <filter clause>.

8) Specifications for Feature T621, "Enhanced numeric functions":

a) Subclause 6.2, "<numeric value function>":

i) Insert this CR Without Feature T621, "Enhanced numeric functions", conforming SQL language shall not specify <natural logarithm>, <exponential function>, <power function>, <square root>, <floor function> or <ceiling function>.

b) Subclause 8.1, "<aggregate function>":

i) Without Feature T621, "Enhanced numeric functions", conforming SQL language shall not specify STDDEV_POP, STDDEV_SAMP, VAR_POP, VAR_SAMP, or <binary set function type>.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex B
(informative)

Implementation-defined elements

Insert this paragraph This Annex references those features that are identified in the body of this Amendment as implementation-defined.

Insert this paragraph The term *implementation-defined* is used to identify characteristics that may differ between SQL-implementations, but shall be defined for each particular SQL-implementation.

1) Insert this list element Subclause 4.3.2, "Window functions":

   a) If PERCENT_RANK is specified, then the declared type of the result is approximate numeric with implementation-defined precision.

   b) If CUME_DIST is specified, then the declared type of the result is approximate numeric with implementation-defined precision.

2) Insert this list element Subclause 6.2, "<numeric value function>":

   a) The declared type of the result of <natural logarithm> is approximate numeric with implementation-defined precision.

   b) The declared type of the result of <exponential function> is approximate numeric with implementation-defined precision.

   c) The declared type of the result of <power function> is approximate numeric with implementation-defined precision.

   d) The declared type of the result of <floor function> is exact numeric with implementation-defined precision and scale 0 (zero).

   e) The declared type of the result of <ceiling function> is exact numeric with implementation-defined precision and scale 0 (zero).

3) Insert this list element Subclause 8.1, "<aggregate function>":

   a) If <hypothetical set function> is specified, then:

      i) If RANK or DENSE_RANK is specified, then the declared type of the result is exact numeric with implementation-defined precision and with scale 0 (zero).

      ii) Otherwise, the declared type of the result is approximate numeric with implemenation-defined precision.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) If the declared type of the <value expression> simply contained in the <sort specification> of an <inverse distribution function> that specifies PERCENTILE_CONT is numeric, then the result type is approximate numeric with implementation-defined precision.

c) If COUNT is specified, then the declared type of the result is exact numeric with implementation-defined precision and scale of 0 (zero).

d) If SUM or AVG is specified, then:

   i) If SUM is specified and the declared type of the argument is exact numeric with scale *S*, then the declared type of the result is exact numeric with implementation-defined precision and scale *S*.

   ii) If AVG is specified and the declared type of the argument is exact numeric, then the declared type of the result is exact numeric with implementation-defined precision not less than the precision of *DT* and implementation-defined scale not less than the scale of *DT*.

   iii) If the declared type of the argument is approximate numeric, then the declared type of the result is approximate numeric with implementation-defined precision not less than the precision of *DT*.

   iv) If the declared type of the argument is interval, then the declared type of the result is interval with the same precision as *DT*.

e) If VAR_POP or VAR_SAMP is specified, then the declared type of the result is approximate numeric with implementation-defined precision not less than the precision of *DT*.

f) If <binary set function type> is specified, then:

   i) If REGR_COUNT is specified, then the declared type of the result is exact numeric with implementation-defined precision and scale of 0 (zero).

   ii) If REGR_AVGX is specified, then:

      1) If the declared type *DTIVE* of the second parameter is exact numeric, then the declared type of the result is exact numeric with implementation-defined precision not less than the precision of *DTIVE* and implementation-defined scale not less than the scale of *DTIVE*.

      2) Otherwise, the declared type of the result is approximate numeric with precision not less than the precision of *DTIVE*.

   iii) If REGR_AVGY is specified, then:

      1) If the declared type *DTDVE* of the first parameter is exact numeric, then the declared type of the result is exact numeric with implementation-defined precision not less than the precision of *DTDVE* and implementation-defined scale not less than the scale of *DTDVE*.

      2) Otherwise, the declared type of the result is approximate numeric with precision not less than the precision of *DTDVE*.

   iv) Otherwise, the declared type of the result is approximate numeric with implementation-defined precision.

 © ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4) Insert this list element Subclause 8.2, "<sort specification list>":

a) If <null ordering> is not specified, then an implementation-defined <null ordering> is implicit. The implementation-defined default for <null ordering> shall not depend on the context outside of <sort specification list>.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex C
(informative)

Implementation-dependent elements

Insert this paragraph This Annex references those places where this Amendment states explicitly that the actions of a conforming SQL-implementation are implementation-dependent.

Insert this paragraph The term *implementation-dependent* is used to identify characteristics that may differ between SQL-implementations, but that are not necessarily specified for any particular SQL-implementation.

1) Insert this list element Subclause 4.2.1, "Windowed tables":

   a) The window name of a window defined implicitly by an <in-line window specification> is implementation-dependent.

2) Insert this list element Subclause 7.5, "<window clause>":

   a) If the window ordering clause of a window structure descriptor is absent, then the window ordering is entirely implementation-dependent.

   b) The window ordering of peer rows within a window partition is implementation-dependent, but the window ordering shall be the same for all window structure descriptors that are order-equivalent. It shall also be the same for windows *W1* and *W2* if *W1* is the ordering window for *W2*.

3) Insert this list element Subclause 7.6, "<query specification>":

   a) The <column name> of a <derived column> that is not a <column reference> and that has no <as clause> is implementation-dependent, but shall not be equivalent to the <column name> of any column, other than itself, of a table referenced by any <table reference> contained in the SQL-statement.

4) Insert this list element Subclause 8.1, "<aggregate function>":

   a) If the declared type of the argument of MAX or MIN is a user-defined type and the comparison of two values results in *unknown* , then the maximum or minimum is implementation-dependent.

5) Insert this list element Subclause 8.2, "<sort specification list>":

   a) If PV_i and QV_i are not null and the result of "PV_i <comp op> QV_i" is *unknown* , then the relative ordering of $PV_i$and QV_i is implementation-dependent.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) The relative ordering of two rows that are not distinct with respect to the <sort specification> is implementation-dependent.

© ISO/IEC 2001 – All rights reserved

--`,,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex D
(informative)

SQL feature and package taxonomy

This Annex describes a taxonomy of features of the SQL language.

Table 4, "SQL/OLAP feature taxonomy for features outside Core SQL", contains a taxonomy of the features of the SQL language that are introduced in this Amendment. In this table, the first column contains a counter that may be used to quickly locate rows of the table; these values otherwise have no use and are not stable — that is, they are subject to change in future editions of, or even in Technical Corrigenda to, ISO/IEC 9075 without notice.

The second column, "Feature ID", specifies the formal identification of each feature and each subfeature contained in the table. The Feature ID is stable and can be depended on to remain constant. A Feature ID value comprises either a letter and three digits or a letter, three digits, a hyphen, and one or two additional digits. Feature ID values containing a hyphen and additional digits indicate "subfeatures" that help to define complete features, which are in turn indicated by Feature ID values without a hyphen. Only entire features are used to specify the contents of Core SQL and various packages.

The "Feature Name" column contains a brief description of the feature or subfeature associated with the Feature ID value.

Table 4—SQL/OLAP feature taxonomy for features outside Core SQL

	Feature ID	Feature Name
1	**T611**	**Elementary OLAP operators**
2	**T612**	**Advanced OLAP operators**
3	**T621**	**Enhanced numeric functions**

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex E
(informative)

SQL Packages

Append this paragraph This Annex describes an additional package of features of the SQL language. Table 5, "SQL Packages", contains a row for the additional package, showing the Package ID and Package Description of this package. the SQL language.

Table 5—SQL Packages

	Package ID	Package Description
1	PKG010	OLAP

E.1 OLAP

The package called "OLAP" comprises the following features of the SQL language as specified in the SQL Feature Taxonomy Annex of the various parts of ISO/IEC 9075.

— Feature T431, "Extended grouping capabilities"

— Feature T611, "Elementary OLAP operators"

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Index

Index entries appearing in **boldface** indicate the page where the word, phrase, or BNF nonterminal was defined; index entries appearing in *italics* indicate a page where the BNF nonterminal was used in a Format; and index entries appearing in roman type indicate a page where the word, phrase, or BNF nonterminal was used in a heading, Function, Syntax Rule, Access Rule, General Rule, Leveling Rule, Table, or other descriptive text.

— A —

aggregated argument • 21, 22, 34, 35, 36, 38, 46, 49, 63, 78
<aggregate function> • *21*, *22*, *27*, 30, 49, ***51***, *52*, 60, 77
ALL • 13, 14, *51*, 52, 60, 77
amendment • 1, 3
AND • 28, 29, *38*, 39
ANY • 13, *51*, 53
approximate numeric • 12, 24, 28, 53, 54, 59, 81, 82
<approximate numeric literal> • 59
array • 38, 53, 61
AS • 28, 29, *37*, *47*, *48*, 49, 58, 59, 60
ASC • 10, 11, *61*
<as clause> • 48, 85
associated with • 10, 87
<asterisk> • 47, *51*
attribute • 31
ATTRIBUTE • 5, 73, 75, 87, 89
<attribute or method reference> • 31
ATTRIBUTES • 5, 73, 75, 87, 89
AVG • 13, 14, 15, *17*, *51*, 53, 55, 56, 82

— B —

based on • 3
BETWEEN • 28, 29, *38*, 39
binary aggregate functions • 14
<binary set function> • 21, ***51***, 53, 60, 77
<binary set function type> • ***51***, 56, 60, 80, 82
boolean • 53
BOOLEAN • 60, 78
built-in function • 7, 71
built-in functions • 7, 71
BY • *37*, *49*, *52*, 58, 59, 60

— C —

cardinality • 55, 56
<case expression> • 31
CAST • 28, 29, 59
<cast specification> • 31
CEIL • *17*, *23*, 72
CEILING • *17*, *23*, 72
<ceiling function> • 9, ***23***, 24, 25, 26, 80, 81
character • 22, 53, 54, 81, 85
character repertoire • 54
character type • 54
coercibility • 22, 53, 54
<collate clause> • 11, *37*, 40
collating sequence • 22, 53
collation • 11, 54
<collation name> • 11
collection • 15, 31
<collection value constructor> • 31
column • 10, 11, 12, 21, 22, 31, 34, 35, 36, 37, 38, 40, 41, 46, 47, 48, 49, 54, 55, 56, 58, 59, 60, 63, 67, 77, 78, 85, 87
<column name> • 48, 85
<column reference> • 12, 22, 31, 36, *37*, 38, 48, 85
<comma> • *23*, *37*, 48, *51*, *52*, *61*
<comparison predicate> • 55, 62
completion condition • 55, 56
<comp op> • 61, 62, 85
<computational operation> • 21, ***51***
condition • 13, 21, 24, 25, 26, 34, 35, 36, 40, 42, 43, 44, 45, 47, 51, 55, 56, 58, 59, 73, 78
conforming SQL-implementation • 85
containing • 5, 12, 87
CORR • *17*, *51*, 57
COUNT • 13, 14, *17*, 28, 29, 49, *51*, 52, 53, 55, 56, 60, 77, 82
COVAR_POP • 14, *17*, *51*, 56
COVAR_SAMP • 14, *17*, *51*, 56
CREATE • 72
CUBE • 12
CUME_DIST • 12, 15, *17*, *27*, *28*, 30, 60, 79, 81
current • 3, 10, 42, 43, 44, 45, 46, 52
CURRENT • 28, *38*, 39, 42, 44, 45
<cursor specification> • 27, 67

— D —

data exception • 24, 25, 26, 42, 43, 44, 45, 55, 58, 73
data type • 25, 57, 58, 60, 69, 78
date • 3, 39, 42
DATE • 69
datetime field overflow • 42
<datetime value expression> • 39

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

DAY • 42, 69
<declare cursor> • 7, 39, 67
declared type • 22, 24, 26, 31, 38, 39, 52, 53, 54, 55, 60, 61, 64, 78, 81, 82, 85
DENSE_RANK • 12, 15, *17*, *27*, *28*, 54, 81
dependencies • 36, 46, 49, 78
dependent • 8, 10, 11, 14, 15, 21, 29, 36, 38, 41, 45, 48, 49, 51, 53, 55, 56, 57, 58, 59, 60, 62, 63, 77, 85, 86
dependent on • 36, 38, 49, 63
<dependent variable expression> • 14, 15, 21, ***51***, 53, 56, 57, 60, 77
<derived column> • 47, 48, 85
derived table • 10, 33
<derived table> • 10
DESC • 10, 11, 42, 43, 44, *61*
descriptor • 10, 11, 12, 13, 27, 28, 33, 39, 40, 41, 52, 85
deterministic • 11, 59, 68
directly contains • 5, 22, 49
distinct • 10, 12, 41, 47, 48, 58, 62, 86
DISTINCT • 13, 14, 27, 28, *51*, 53, 55, 60, 67, 77, 78
DOUBLE • 72
<drop table constraint definition> • 7, 63
<drop user-defined ordering statement> • 7, 64
duplicate • 13, 14, 55
duplicates • 13, 14

— E —

effective • 31
<element reference> • 31
elements • 6, 7, 8, 10, 17, 51, 81, 85
empty • 55
EVERY • *51*, 53
exact numeric • 24, 25, 26, 39, 52, 53, 54, 59, 81, 82
exception • 24, 25, 26, 42, 43, 44, 45, 55, 58, 73
EXCEPTION • *65*
exception condition • 24, 25, 26, 42, 55, 58
EXCLUDE • *17*, *38*, 46
EXECUTE • 72
<existing window name> • ***37***, 39, 40, 41, 46, 79
EXP • *17*, *23*, *25*, 72
Explicit • 54
<exponential function> • 9, ***23***, 24, 26, 80, 81
extended sort key columns • 67
externally-invoked procedure • 7, 65
<externally-invoked procedure> • 7, 65

— F —

Feature F441 • 60, 77
Feature F561 • 60, 77
Feature S024 • 46, 60, 62, 78
Feature T031 • 60, 78
Feature T301 • 36, 46, 49, 78
Feature T431 • 89
Feature T611 • 30, 31, 46, 62, 79, 89
Feature T612 • 19, 26, 30, 33, 46, 60, 79, 80
Feature T621 • 26, 60, 80
field reference • 31
<field reference> • 31
FILTER • *17*, *51*
<filter clause> • 13, 21, ***51***, 53, 55, 60, 80
final • 9
FIRST • 10, 11, 42, 43, 44, *61*, 62
FLOOR • *17*, *23*, 72
<floor function> • 9, ***23***, 24, 25, 26, 80, 81
FOLLOWING • *17*, 28, 29, *38*, 39, 42, 43, 44, 45, *65*
FROM • 49, 58, 59, 60
<from clause> • 29, *33*, 35, 40, 47
FUNCTION • *65*, *72*
functional dependency • 63

— G —

generally contain • 60, 77
<general set function> • 21, ***51***, 52, 53, 55, 60, 77, 78
GRANT • 72
<greater than operator> • 61
GROUP • *49*, *52*
group aggregate function • 11, 13
<group by clause> • 29, *33*, 40, 47, 49
grouped, windowed query • 47
group function • 11
GROUPING • *12*
grouping column • 36, 38, 46, 49, 63, 78
<grouping column reference list> • 63
grouping operation • 11, 12, 21
<grouping operation> • 21

— H —

<having clause> • 7, 11, 22, 29, *33*, 34, 35, 36, 40, 47
HOUR • 69
hypothetical set function • 13, 15, 27, 51, 52, 54, 58, 60, 80, 81
<hypothetical set function> • 27, *51*, ***52***, 54, 58, 60, 80, 81
<hypothetical set function value expression list> • ***52***, 54, 58

— I —

identified • 11, 39, 40, 52, 81
identifier • 6, 19, 47, 48
<identifier> • *19*, 47, 48
identify • 11, 55, 81, 85
immediate • 27, 31, 33, 38, 40, 47, 49, 52, 61, 62, 67, 78
immediately contain • 27, 31, 33, 40, 47, 49, 52, 61, 62, 67, 78
implementation-defined • 24, 28, 52, 53, 54, 57, 58, 61, 72, 81, 82, 83
implementation-dependent • 10, 29, 41, 45, 48, 55, 58, 59, 62, 85, 86
Implicit • 10
include • 1, 13, 39
independent • 11, 14, 21, 51, 53, 56, 57, 58, 60, 77
<independent variable expression> • 14, 21, ***51***, 53, 56, 57, 60, 77
Information Schema • 7, 71
INFORMATION_SCHEMA • 72
<in-line window specification> • ***27***, 29, 47, 85

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

innermost • 27
INTERVAL • 69
interval type • 39, 53, 69
<interval value expression> • 39
INTO • 30
invalid • 24, 25, 26, 42, 43, 44, 45, 73
invalid argument for natural logarithm • 24, 73
invalid argument for power function • 25, 73
invalid argument for width bucket function • 26, 73
invalid preceding or following size in window function • 42, 43, 44, 45, 73
inverse distribution • 13, 15, 52, 54, 58, 60, 80, 82
<inverse distribution function> • ***52***, 54, 58, 60, 80, 82
<inverse distribution function argument> • ***52***, 54, 58
<inverse distribution function type> • ***52***

— J —

<joined table> • 6, 34

— K —

K • 12, *17*, *27*, *28*, 54, 81
known functional dependency • 63

— L —

LAST • 10, 11, 42, 43, 44, *61*, 62
<left paren> • *23*, *27*, *37*, *51*, *52*
<less than operator> • 61
<literal> • 59
LN • *17*, *23*, *25*, 72

— M —

MAX • 13, *51*, 53, 55, 59, 60, 78, 85
method • 31, 64
<method invocation> • 31
MIN • 13, *51*, 53, 55, 59, 60, 78, 85
monadic • 54
MONTH • 42

— N —

NATURAL • *65*
<natural logarithm> • 9, ***23***, 24, 26, 80, 81
<new window name> • ***37***, 38, 39, 40
NO • *38*, 46
non-aggregated argument • 21
<nonparenthesized value expression primary> • ***31***
<non-reserved word> • ***17***
not distinct • 10, 12, 41, 62, 86
not permitted • 13
<null ordering> • 11, 40, ***61***, 62, 79, 83
NULLS • 10, 11, *17*, 42, 43, 44, *61*, 62
null value • 10, 13, 14, 24, 25, 42, 43, 44, 45, 55, 56, 57, 58, 59, 62
null value eliminated in set function • 55, 56, 59
numbers • 6, 9, 59
NUMERIC • 69, 72
numeric types • 72
<numeric value expression> • 14, *23*, 24, 25, *51*, *52*, 60, 77
<numeric value expression base> • ***23***, 25
<numeric value expression exponent> • ***23***, 25
<numeric value function> • 6, ***23***
numeric value out of range • 24, 25, 26, 55, 58

— O —

OLAP • 19, 26, 30, 31, 33, 46, 60, 62, 75, 79, 80, 87, 89
ON • 72
ORDER • *37*
<order by clause> • 11, 12, 27, 29, 39, 67
<ordered set function> • 5, 21, 27, ***51***
order-equivalent • 40, 41, 85
<ordering specification> • 11, 40, 42, 43, 44, 59, ***61***, 67
ordering window • 10, 11, 40, 41, 85
OTHERS • *17*, *38*, 46
outer reference • 21, 22, 27, 34, 35, 36, 49, 53, 54, 60, 77
OVER • *17*, *27*, *28*, 29, 58, 59, 60

— P —

package • 8, 87, 89
Part 1 • 3
Part 2 • 3
Part 5 • 3
PARTITION • *17*, *37*
partitioning column • 38, 41, 46, 78
part of • 5, 6, 7, 8, 11
peers • 10, 12, 41, 42, 44, 45, 46, 62
PERCENTILE_CONT • 15, *17*, *52*, 54, 59, 82
PERCENTILE_DISC • 15, *17*, *52*, 59
PERCENT_RANK • 12, 15, *17*, *27*, *28*, 30, 79, 81
permitted • 13
possibly non-deterministic • 11
POWER • *17*, *23*, *24*, *57*, *65*, 72
<power function> • 9, ***23***, 24, 25, 26, 80, 81
precede • 12, 41, 42, 45, 62, 72
PRECEDING • *17*, 28, *29*, *37*, *38*, 39, *42*, 43, 44, 45, *65*
precision • 24, 28, 52, 53, 54, 81, 82
PRECISION • 72
<prepare statement> • 7, 69
<primary datetime field> • 42
PUBLIC • 72

— Q —

<qualified asterisk> • 47
query • 5, 7, 11, 12, 21, 22, 27, 29, 31, 34, 35, 36, 38, 39, 40, 47, 48, 49, 53, 54, 58, 59, 63, 78
<query expression body> • 39
<query specification> • 7, 11, 12, 27, 29, 39, 40, 47, 48, 49, 63

— R —

RANGE • 10, *17*, 28, 29, *37*, 39, 42
RANK • 12, *17*, *27*, *28*, 54, 81
rank functions • 12
<rank function type> • ***27***, 28, 49, *52*, 58
redundant duplicates • 13, 14
<reference resolution> • 31
REGR_AVGX • 14, *17*, *51*, 56, 82

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

REGR_AVGY • 15, *17*, *51*, 56, 82
REGR_COUNT • 14, *17*, *51*, 53, 56, 82
REGR_INTERCEPT • 14, *17*, *51*, 57
REGR_R2 • 14, *17*, *51*, 57
REGR_SLOPE • 14, *17*, *51*, 57
REGR_SXX • 14, *17*, *51*, 56
REGR_SXY • 14, *17*, *51*, 56
REGR_SYY • 14, *17*, *51*, 56
repertoire • 54
<reserved word> • ***17***
result data type • 25
RETURN • 72
RETURNS • 72
<right paren> • *23*, *27*, *37*, *51*, *52*
ROLLUP • 12
<routine invocation> • 31, 60, 77
ROW • 28, *38*, 39, 42, 44, 45, 46
row number function • 12
ROWS • 10, 28, *37*, 39, 44, 49, 69
row type • 33
ROW_NUMBER • 12, *17*, *27*, *28*, 30, 49, 59, 79

— S —

<scalar subquery> • 31, 58, 59
SCHEMA • 72
scope • 1, 5, 38, 39
search condition • 13, 34, 35, 36, 55, 78
<search condition> • 13, 34, 35, 36, *51*, 55, 78
SECOND • 69
SELECT • 29, 30, 49, 58, 59, 60
<select list> • 11, 12, 22, 27, 29, 34, 35, 40, 47, 48, 49, 63
<select statement: single row> • 7, 11, 12, 27, 29, 30, 39, 40, 68
<select target list> • 30
<separator> • 6, 17
sequence • 10, 22, 53
SET • 12
<set function specification> • 5, 6, ***21***, 22, 31, 34, 35, 36, 38, 46, 47, 48, 49, 52, 54, 63, 67, 78
<set function type> • 21, ***51***, 53, 60, 78
<set quantifier> • 21, 29, 47, ***51***, 52, 53, 60, 67
SIMPLE • 3, 24, 25, 28, 29, 30, 39, 47, 48, 58, 59
simple table query • 11, 27, 29
simply contain • 21, 24, 25, 27, 28, 29, 30, 31, 39, 40, 41, 47, 48, 49, 52, 54, 58, 59, 60, 64, 77, 82
simply contained in • 21, 27, 28, 29, 30, 39, 40, 41, 47, 48, 54, 58, 59, 60, 77, 82
SIZE • *65*
SOME • 13, *51*, 53, 56, 60, 78
sort direction • 61
<sort key> • 11, 28, 39, 40, 42, 43, 44, 54, 58, 59, ***61***, 64, 67, 69
<sort specification> • 11, 40, 54, 58, 59, ***61***, 62, 67, 82, 86
<sort specification list> • 11, *37*, 41, *52*, 54, 58, ***61***, 67, 83
sort table • 27
source • 36, 49, 52
SPECIFIC • 72
specified by • 10, 13, 67
specify • 10, 11, 15, 19, 26, 27, 30, 33, 39, 40, 46, 48, 60, 62, 79, 80, 87
SQL-agent • 65
SQL-client • 7, 65
SQL-client module • 7, 65
SQL-implementation • 81, 85
SQLSTATE • 7, 8, *65*, 73
SQL-statement • 48, 85
SQRT • *17*, *23*, *53*, *57*, 72
<square root> • 9, ***23***, 24, 26, 80
static method • 31
<static method invocation> • 31
STDDEV_POP • 14, *17*, *51*, 52, *53*, 60, 80
STDDEV_SAMP • 14, *17*, *51*, 52, *53*, 60, 80
structured type • 46, 60, 62, 78
<subquery> • 5, 21, 22, 27, 34, 35, 36, 38, 48, 53, 54, 78
subtype • 31
<subtype treatment> • 31
SUM • 13, *51*, 53, 55, *56*, *57*, *58*, 82

— T —

table constraint • 7, 63
<table expression> • 6, 10, 11, 22, 27, 29, ***33***, 38, 40, 47, 49
<table reference> • 22, 48, 85
target • 30
THEN • *28*
TIES • *17*, *38*, 46
TIME • 69
TIMESTAMP • 69
TO • 69, 72
<token> • 6, 17
transient • 10
types • 46, 60, 62, 72, 78

— U —

unary aggregate function • 13
UNBOUNDED • *17*, 28, 29, *37*, *38*, 39, 42, 43, 44, 45
UNION • 58
<unsigned value specification> • 31, *38*, 39, 42, 43, 44, 45
user-defined • 7, 55, 64, 85
user-defined type • 55, 85

— V —

valid • 3, 24, 25, 26, 42, 43, 44, 45, 73
<value expression> • 6, 10, 13, 14, 21, 28, 31, 34, 35, 49, *51*, *52*, 54, 55, 58, 59, 60, *61*, 62, 64, 77, 78, 79, 82
<value expression primary> • 31
VALUES • 58
VAR_POP • 14, *17*, *51*, 52, *53*, 56, 60, 80, 82
VAR_SAMP • 14, *17*, *51*, 52, *53*, 56, 60, 80, 82
view • 63

© ISO/IEC 2001 – All rights reserved

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

— W —

warning • 55, 56, 59
WHERE • *51*, 58, 59, 60
<where clause> • 7, 29, *33*, 35, 40, 47
<width bucket bound 1> • ***23***, 25
<width bucket bound 2> • ***23***, 25
<width bucket count> • ***23***, 24, 25
<width bucket function> • ***23***, 24, 25, 26, 79
<width bucket operand> • ***23***, 25
WIDTH_BUCKET • 9, *17*, *23*, *65*, *72*
window • 6, 7, 10, 11, 12, 13, 15, 19, 22, 27, 28, 29, 30, 31, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 52, 53, 54, 69, 73, 78, 79, 80, 81, 85
WINDOW • *17*, *29*, *37*, *65*
window aggregate functions • 13
<window clause> • 11, 22, 29, *33*, ***37***, 38, 39, 40, 41, 46, 47, 48, 78, 79
<window definition> • 10, ***37***, 38, 39, 40
<window definition list> • ***37***
windowed table • 10, 11, 12, 13
window frame • 10, 11, 12, 13, 37, 38, 39, 41, 42, 43, 44, 45, 52, 69
<window frame between> • *37*, ***38***, 39
<window frame bound 1> • ***38***, 39, 42
<window frame bound 2> • ***38***, 39, 42
<window frame bound> • ***38***
<window frame clause> • 11, ***37***, 39, 41
<window frame exclusion> • 10, *37*, ***38***
<window frame extent> • ***37***, 39
<window frame following> • ***38***, 39, 43, 44, 45, 69
<window frame preceding> • *37*, ***38***, 39, 42, 43, 44, 45, 69
<window frame start> • ***37***, *38*, 39
<window frame units> • ***37***
<window function> • 10, 11, ***27***, 28, 30, *31*, 34, 35, 36, 38, 40, 47, 49, 52, 53, 54, 79
<window function type> • ***27***, 29, 49
<window name> • ***19***, *27*, 28, 29, 30, *37*, 39, 79
<window name or specification> • ***27***, 28
<window order clause> • 10, ***37***, 38, 39, 41, 44
window ordering • 10, 11, 12, 15, 27, 28, 30, 39, 40, 41, 42, 43, 44, 45, 79, 85
window partition • 10, 11, 12, 37, 38, 39, 40, 41, 42, 85
<window partition clause> • 10, ***37***, 38, 39, 40
<window partition column reference> • ***37***, 40
<window partition column reference list> • ***37***
window partitioning • 10, 11, 40, 41
<window specification> • 10, *27*, 29, ***37***, 46, 49, 69, 79
<window specification details> • 28, ***37***
window structure descriptor • 10, 11, 12, 13, 27, 28, 33, 39, 40, 41, 52, 85
WITH • 69
<within group specification> • 5, 21, ***52***, 54
WITHOUT • 69
without an intervening • 5, 34, 35, 36, 38, 48, 60, 77

— Y —

YEAR • 42
year-month • 42
year-month interval • 42

— Z —

ZONE • 69

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ICS 35.060

Price based on 82 pages

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4851903 0859926 817

INTERNATIONAL STANDARD ISO/IEC 9075-1:1999
ISO/IEC 9075-2:1999
ISO/IEC 9075-3:1999
ISO/IEC 9075-4:1999
ISO/IEC 9075-5:1999

TECHNICAL CORRIGENDUM 1

Published 2000-12-15

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

Information technology — Database languages — SQL —

Part 1: Framework (SQL/Framework)
Part 2: Foundation (SQL/Foundation)
Part 3: Call-Level Interface (SQL/CLI)
Part 4: Persistent Stored Modules (SQL/PSM)
Part 5: Host Language Bindings (SQL/Bindings)

TECHNICAL CORRIGENDUM 1

Technologies de l'information — Langages de base de données — SQL —

Partie 1: Charpente (SQL/Charpente)
Partie 2: Fondations (SQL/Fondations)
Partie 3: Interface de niveau d'appel (SQL/CLI)
Partie 4: Modules stockés persistants (SQL/PSM)
Partie 5: Liants de langage d'hôte (SQL/Liants)

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to parts 1 to 5 of International Standard ISO/IEC 9075:1999 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange.*

Statement of purpose for rationale:

A statement indicating the rationale for each change to ISO/IEC 9075 is included. This is to inform the users of that standard as to the reason why it was judged necessary to change the original wording. In many cases the reason is editorial or to clarify the wording; in some cases it is to correct an error or an omission in the original wording.

Notes on numbering:

Where this Corrigendum introduces new Syntax, Access, General and Conformance Rules, the new rules have been numbered as follows:
Rules inserted between, for example, Rules 7) and 8) are numbered 7.1), 7.2), etc. [or 7) a.1), 7) a.2), etc.].
Those inserted before Rule 1) are numbered 0.1), 0.2), etc.
Where this Corrigendum introduces new subclauses, the new subclauses have been numbered as follows:
Subclauses inserted between, for example, subclause 4.3.2 and 4.3.3 are numbered 4.3.2a, 4.3.2b, etc.
Those inserted before, for example, 4.3.1 are numbered 4.3.0, 4.3.0a, etc.

ICS 35.060 **Ref. No. ISO/IEC 9075 (parts 1 to 5):1999/Cor.1:2000(E)**

© ISO/IEC 2000 – All rights reserved

Printed in Switzerland

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4851903 0859927 753

Page

ISO/IEC 9075-1:1999
Database Languages - SQL-Part 1:Framework ... 9
4.8.2.3 Locators ... 9
4.11.2 SQL-statements classified by function ... 9
5.3.3 SQL-statements specified in ISO/IEC 9075-2 ... 9
5.5.1 SQL-statements specified in ISO/IEC 9075-4 ... 9
5.6.5.1 Additional functional classes of SQL-statements ... 10
6.2.5 Relationships of incremental parts to ISO/IEC 9075-2, Foundation ... 10
Annex B SQL Packages ... 13
Annex B.9 SQL/MM Support ... 13

ISO/IEC 9075-2:1999
Database Languages - SQL-Part 2:Foundation ... 15
3.1.1 Definitions taken from ISO/IEC 10646 ... 15
3.1.2 Definitions taken from Unicode ... 15
3.1.5 Definitions provided in Part 2 ... 15
4.1 Data types ... 17
4.2.1 Character strings and collating sequences ... 19
4.2.4 Named character sets ... 19
4.3.1 Binary string comparison ... 20
4.4.1 Bit string comparison and assignment ... 20
4.7.1 Datetimes ... 20
4.7.3 "Operations involving datetimes and intervals" ... 21
4.8 User-defined types ... 21
4.8.1 Observers and mutators ... 22
4.8.2 Constructors ... 22
4.8.4 User-defined type comparison and assignment ... 23
4.10 Reference types ... 23
4.13 Data conversions ... 24
4.16 Tables ... 24
4.16.3 Operations involving tables ... 25
4.18.1 General rules and definitions ... 25
4.18.9 Known functional dependencies in the result of <having clause> ... 26
4.20 SQL-schemas ... 26
4.21 SQL-client modules ... 26
4.23 SQL-invoked routines ... 26
4.24 Built-in functions ... 29
4.25 SQL-paths ... 29
4.26.4 Locators ... 30
4.30.1 Classes of SQL-statements ... 30
4.30.2 SQL-statements classified by functions ... 30
4.30.3 SQL-statements and transaction states ... 31
4.30.4 SQL-statement atomicity ... 32
4.31.1 Authorization identifiers ... 32
4.34.1 Execution contexts ... 32
4.35.2 Execution of triggers ... 33
5.2 <token> and <separator> ... 33
5.3 <literal> ... 36
5.4 Names and identifiers ... 36
6.1 <data type> ... 37
6.3 <value specification> and <target specification> ... 37
6.5 <identifier chain> ... 38
6.11 <method invocation> ... 39

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.16 <set function specification> ... 40
6.18 <string value function> ... 41
6.20 <interval value function> ... 45
6.22 <cast specification> ... 45
6.23 <value expression> ... 47
6.24 <new specification> ... 48
6.25 <subtype treatment> ... 49
6.26 <numeric value expression> ... 49
6.27 <string value expression> ... 50
6.30 <boolean value expression> ... 51
7.1 <row value constructor> ... 52
7.3 <table value constructor> ... 52
7.6 <table reference> ... 53
7.7 <joined table> ... 57
7.8 <where clause> ... 57
7.9 <group by clause> ... 57
7.10 <having clause> ... 64
7.11 <query specification> ... 65
7.12 <query expression> ... 66
8.2 <comparison predicate> ... 66
8.3 <between predicate> ... 67
8.4 <in predicate> ... 68
8.6 <similar predicate> ... 68
8.8 <quantified comparison predicate> ... 69
8.10 <unique predicate> ... 69
8.11 <match predicate> ... 70
8.12 <overlaps predicate> ... 70
8.13 <distinct predicate> ... 70
9.0 Determination of identical values ... 71
9.1 Retrieval assignment ... 72
9.3 Data types of results of aggregations ... 72
10.4 <routine invocation> ... 73
10.5 <privileges> ... 77
10.6 <character set specification> ... 78
10.7 <specific routine designator> ... 78
10.12 Execution of triggers ... 80
10.13 Execution of array-returning functions ... 81
11.1 <schema definition> ... 82
11.2 <drop schema statement> ... 83
11.3 <table definition> ... 83
11.4 <column definition> ... 85
11.7 <unique constraint definition> ... 85
11.8 <referential constraint definition> ... 86
11.17 <drop column definition> ... 86
11.21 <view definition> ... 86
11.30 <character set definition> ... 87
11.31 <drop character set statement> ... 88
11.38 <trigger definition> ... 88
11.40 <user-defined type definition> ... 89
11.41 <attribute definition> ... 97
11.43 <add attribute definition> ... 98
11.44 <drop attribute definition> ... 98
11.45 <add original method specification> ... 98
11.46 <add overriding method specification> ... 102
11.47 <drop method specification> ... 104

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0859929 526

11.48 <drop data type statement> 108
11.49 <SQL-invoked routine> 108
11.50 <alter routine statement> 114
11.51 <drop routine statement> 114
11.52 <user-defined cast definition> 115
11.54 <user-defined ordering definition> 116
11.55 <drop user-defined ordering statement> 117
11.56 <transform definition> 119
11.57 <drop transform statement> 119
12.2 <grant privilege statement> 119
12.4 <select statement: single row> 120
12.6 <revoke statement> 120
13.1 <SQL-client module definition> 121
13.2 <module name clause> 121
13.3 <externally-invoked procedure> 121
13.5 <SQL procedure statement> 121
14.1 <declare cursor> 122
14.3 <fetch statement> 124
14.5 <select statement: single row> 124
14.6 <delete statement: positioned> 125
14.8 <insert statement> 126
14.9 <update statement: positioned> 127
14.10 <update statement: searched> 127
14.11<temporary table declaration> 128
14.18 Effect of inserting a table into a derived table 128
15.2 <return statement> 128
16.4 <savepoint statement> 128
16.5 <release savepoint statement> 128
16.7 <rollback statement> 129
19.1 <get diagnostics statement> 129
20.11 ATTRIBUTES view 130
20.18 COLUMNS view 131
20.19 CONSTRAINT_COLUMN_USAGE view 132
20.21 DATA_TYPE_PRIVILEGES view 133
20.26 DOMAINS view 133
20.27 ELEMENT_TYPES view 134
20.29 FIELDS view 134
20.31 METHOD_SPECIFICATION_PARAMETERS view 135
20.32 METHOD_SPECIFICATIONS view 135
20.33 PARAMETERS view 136
20.34 REFERENCED_TYPES view 136
20.38 ROLE_TABLE_GRANTS view 137
20.56 TABLES view 137
20.62 TRIGGERS view 138
20.68 VIEWS view 140
20.69 Short name views 141
21.3 EQUAL_KEY_DEGREES assertion 143
21.6 ASSERTIONS base table 144
21.7 ATTRIBUTES base table 144
21.8 CHARACTER_SETS base table 144
21.12 COLLATIONS base table 145
21.14 COLUMNS base table 145
21.15 DATA_TYPE_DESCRIPTOR base table 146
21.16 DIRECT_SUPERTABLES 150
21.23 METHOD_SPECIFICATION_PARAMETERS 150

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4851903 0859930 248

21.24 METHOD_SPECIFICATIONS base table 151
21.25 PARAMETERS base table 152
21.33 ROUTINES base table 152
21.34 SCHEMATA base table 152
21.35 SQL_FEATURES base table 153
21.36 SQL_IMPLEMENTATION_INFO base table 153
21.38 SQL_SIZING base table 153
21.39 SQL_SIZING_PROFILES base table 154
21.43 TABLES base table 154
21.44 TRANSFORMS base table 155
21.45 TRANSLATIONS base table 155
21.47 TRIGGER_COLUMN_USAGE base table 155
21.49 TRIGGERS base table 156
21.52 USER_DEFINED_TYPES base table 157
21.56 VIEWS base table 158
22.1 SQLSTATE 158
22.3 SQL Multimedia and Application Package SQLSTATE Subclasses 158
23.1 General conformance requirements 158
Annex A SQL conformance summary 158
Annex B Implementation-defined elements 164
Annex C Implementation-dependent elements 165
Annex E Incompatibilities with ISO/IEC 9075:1992 and ISO/IEC 9075-4:1996 166
Annex F SQL feature and package taxonomy 167

ISO/IEC 9075-3:1999
Database Languages - SQL-Part 3:Call Level Interface 169
3.1.1.Definitions provided in Part 3 169
3.3.2.1 Clause, subclause and table relationships 169
4.4.5 Connection attributes 170
5.1 <CLI routine> 170
5.3 Description of CLI item descriptor areas 170
5.5 Implicit DESCRIBE USING clause 171
5.11 Deferred parameter check 171
5.13 Description of CLI item descriptor areas 171
5.14 Other tables associated with CLI 172
5.15 Data type correspondences 173
6.5 BindCol 174
6.6 BindParameter 174
6.9 ColAttribute 174
6.10 ColumnPrivileges 175
6.11 Columns 175
6.12 Connect 175
6.14 DataSources 176
6.15 DescribeCol 176
6.17 EndTran 177
6.18 Error 178
6.19 ExecDirect 178
6.21 Fetch 178
6.22 FetchScroll 179
6.23 ForeignKeys 179
6.28 GetConnectAttr 180
6.29 GetCursorName 180
6.30 GetData 180
6.32 GetDescRec 180
6.34 GetDiagRec 181

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0859931 184

6.36 GetFeatureInfo 181
6.40 GetParamData 181
6.50 Prepare 182
6.51 PrimaryKeys 182
6.54 SetConnectAttr 182
6.55 SetCursorName 182
6.56 SetDescField 183
6.60 SpecialColumns 183
6.62 TablePrivileges 183
6.63 Tables 184
A.1 C header file SQLCLI.H 184
A.2 COBOL library item SQLCLI 185

ISO/IEC 9075-4:1999
Database Languages - SQL-Part 4:Persistent Stored Modules 187
3.3.2.1 Clause, Subclause, and Table relationships 187
4.8 Cursors 187
4.10 SQL-statements 187
4.10.1 SQL-statements classified by function 188
5.1 <token> and <separator> 188
6.2 <identifier chain> 188
9.18 <SQL-server module definition> 189
11.2 <SQL procedure statement> 189
12.2 <fetch statement> 189
13.1 <compound statement> 190
13.6 <resignal statement> 190
13.7 <if statement> 190
15.1 <embedded SQL host program> 191
16.1 <get diagnostics statement> 191
16.3 <resignal statement> 191
17.2 MODULE_PRIVILEGES view 191
17.4 MODULES view 192
Annex A SQL Conformance Summary 192
Annex E Incompatibilities with ISO/IEC 9075:1992 193
Annex F SQL Feature Taxonomy 193

ISO/IEC 9075-5:1999
Database Languages - SQL-Part 5:Bindings 195
4.6.1 Classes of SQL-statements 195
4.6.4 Embeddable SQL-statement 195
4.6.5 Preparable and immediately executable SQL-statements 195
4.6.6 Directly executable SQL-statements 196
5.1 <token> and <separator> 196
10.5 <SQL-invoked routine> 197
11.2 Calls to an <externally-invoked procedure> 197
11.3 <SQL procedure statement> 197
12.0 <fetch statement> 197
12.1 <select statement: single row> 198
12.2 <free locator statement> 198
14.3 <set names statement> 199
15.1 Description of SQL descriptor areas 199
15.6 <prepare statement> 199
15.8 <describe statement> 200
15.13 <dynamic declare cursor> 200
15.14 <allocate cursor statement> 200

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4851903 0859932 010

16.1 <embedded SQL host program> 201
16.3 <embedded SQL Ada program> 203
16.4 <embedded SQL C program> 204
16.5 <embedded SQL COBOL program> 204
16.6 <embedded SQL FORTRAN program> 205
16.7 <embedded SQL MUMPS program> 206
16.8 <embedded SQL Pascal program> 206
16.9 <embedded SQL PL/I program> 207
19.1 <get diagnostics statement> 208
20.1 SQLSTATE 208
Annex A SQL Conformance Summary 209
Annex E Incompatibilities with ISO/IEC 9075:1992 211

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ISO/IEC 9075-1:1999
Database Languages - SQL-Part 1:Framework

4.8.2.3 Locators

1. Rationale: Correct the specification of which locators are marked invalid when an SQL-transaction ends.

Replace the 8th paragraph with:

> A non-holdable locator remains valid until the end of the SQL-transaction in which it was generated, unless it is explicitly made invalid by the execution of a <free locator statement> or a <rollback statement> that specifies a <savepoint clause> is executed before the end of that SQL-transaction if the locator was generated subsequent to the establishment of the savepoint identified by the <savepoint clause>.

Replace 9th paragraph with:

> A holdable locator may remain valid beyond the end of the SQL-transaction in which it is generated. A holdable locator becomes invalid whenever a <free locator statement> identifying that locator is executed or the SQL-transaction in which it is generated or any subsequent SQL-transaction is rolled back. All locators remaining valid at the end of an SQL-session are marked invalid when that SQL-session terminates.

4.11.2 SQL-statements classified by function

1. Rationale: Correct the classification of SQL-statements.

Add a list element to the 1st paragraph:

— SQL-dynamic statements, which support the preparation and execution of dynamically generated SQL-statements, and obtaining information about them.

5.3.3 SQL-statements specified in ISO/IEC 9075-2

1. Rationale: Correct the classification of SQL-statements.

Replace the 4th bullet of the 1st paragraph with:

— Two SQL-control statements (CALL and RETURN), which can be used to invoke a procedure and specify a value to be returned by a function.

5.5.1 SQL-statements specified in ISO/IEC 9075-4

1. Rationale: Correct the classification of SQL-statements.

Replace the 1st bullet of the 1st paragraph with:

— Additional SQL-control statements which may be used to control the execution of an SQL routine.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Previous page is blank.

Not for Resale

2. *Rationale: Correct the classification of SQL-statements.*

Delete the 2nd bullet from the 1st paragraph

3. *Rationale: Correct the classification of SQL-statements.*

Replace the 3rd bullet of the 1st paragraph with:

— Additional SQL-diagnostics statements, which may be used to signal exceptions.

4. *Rationale: Correct the classification of SQL-statements.*

Add the following bullets to the 1st paragraph:

— SQL-control declaration statements which may be used to declare variables and exception handlers.

— Additional SQL-schema statements, which may be used to create and drop modules.

5.6.5.1 Additional functional classes of SQL-statements

1. *Rationale: Correct the classification of SQL-statements.*

Replace the bullet of the 1st paragraph with:

— SQL-dynamic statements, which support the preparation and execution of dynamically generated SQL-statements, and obtaining information about them.

2. *Rationale: Correct the classification of SQL-statements.*

Replace the 2nd paragraph with:

A number of SQL-data statements are also added, most of which contain the word "dynamic" in their names. They are not to be confused with SQL-dynamic statements.

6.2.5 Relationships of incremental parts to ISO/IEC 9075-2, Foundation

1. *Rationale: To permit the modification of Parts 1, 3 and 10 as well as Parts 2 and 5 in other Parts, since Part 1 needs to be updated by the Conformance clauses of other Parts and Parts 3 and 10 are analogous in functionality to Part 5.*

Replace the entire Subclause with:

6.2.5 Relationships of incremental parts within ISO/IEC 9075

Parts of ISO/IEC 9075 other than this part of ISO/IEC 9075 and ISO/IEC 9075-2 depend on ISO/IEC 9075-1, ISO/IEC 9075-2 and its Technical Corrigenda and are referenced as incremental parts. Each incremental part is to be used as though it were merged with the text of ISO/IEC 9075. This Subclause describes the conventions used to specify the merger.

The merger described also accounts for the Technical Corrigenda that have been published to correct ISO/IEC 9075. This accommodation is typically indicated by the presence of a phrase like "in the Technical Corrigenda" or "in the TC".

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.2.5.1 New and modified Clauses, Subclauses, and Annexes

Where a Clause (other than Clause 1, "Scope", and Clause 2, "Normative references"), Subclause, or Annex in any incremental part of ISO/IEC 9075 has a name identical to a Clause, Subclause, or Annex in ISO/IEC 9075-1, ISO/IEC 9075-2, ISO/IEC 9075-3, ISO/IEC 9075-5 or ISO/IEC 9075-10 (unless the incremental part is itself ISO/IEC 9075-3, ISO/IEC 9075-5 or ISO/IEC 9075-10), it supplements the Clause, Subclause, or Annex, respectively, in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10, regardless of whether or not the number or letter of the Clause, Subclause, or Annex corresponds. It typically does so by adding or replacing paragraphs, Format items, or Rules.

In each incremental part, Table 1, "Clause, Subclause, and Table relationships", identifies the relationships between each Clause, Subclause, and Annex in that incremental part and the corresponding Clause, Subclause, or Annex in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10.

Where a Clause, Subclause, or Annex in an incremental part has a name that is not identical to the name of some Clause, Subclause, or Annex in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10, it provides language specification particular to that part. A Subclause that is part of a Clause or Subclause identified as new is inherently new and is not marked.

The Clauses, Subclauses, and Annexes in each incremental part appear in the order in which they are intended to appear in the merged document. In the absence of other explicit instructions regarding its placement, any new Clause, Subclause, or Annex is to be positioned as follows: Locate the prior Clause, Subclause, or Annex in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10 whose name is identical to the name of a corresponding Clause, Subclause, or Annex that appears in the incremental part of ISO/IEC 9075. The new Clause, Subclause, or Annex shall immediately follow that Clause, Subclause, or Annex. If there are multiple new Clauses, Subclauses, or Annexes with no intervening Clause, Subclause, or Annex that modifies an existing Clause, Subclause, or Annex, then those new Clauses, Subclauses, or Annexes appear in order, following the prior Clause, Subclause, or Annex whose name was matched.

When an incremental part performs a modification to the Clause, Subclause, or Annex in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10, then the modifications are applied in the following sequence:

1) All modifications to ISO/IEC 9075-1 from the incremental part.
2) All modifications to ISO/IEC 9075-3 from the incremental part.
3) All modifications to ISO/IEC 9075-5 from the incremental part.
4) All modifications to ISO/IEC 9075-10 from the incremental part.
5) All modifications to ISO/IEC 9075-2 from ISO/IEC 9075-3, including all modifications that were added, augmented, or replaced as a result of step 2.
6) All modifications to ISO/IEC 9075-2 from ISO/IEC 9075-5, including all modifications that were added, augmented, or replaced as a result of step 2.
7) All modifications to ISO/IEC 9075-2 from ISO/IEC 9075-10, including all modifications that were added, augmented, or replaced as a result of step 2.
8) All modifications to ISO/IEC 9075-2 from the incremental part. Note that modifications in this third step may augment or replace modifications applied as a result of steps 2, 3 and 4.

Modifications to ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10 by more than one incremental part do not interact. The modifications made by an incremental part only have influence on the language specification of that part and those specifications are not influenced by modifications made by any other incremental part.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.2.5.2 New and modified Format items

In a modified Subclause, a Format item that defines a BNF non-terminal symbol (that is, the BNF non-terminal symbol appears on the left-hand side of the ::= mark) either modifies a Format item whose definition appears in ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10, or replaces a Format item whose definition appears in ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10, or defines a new Format item that does not have a definition at all in ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10. Those Format items in the incremental part that modify a Format item whose definition appears in ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10 are identified by the existence of a "Format comment" such as:

```
<modified item> ::=
       !! All alternatives from ISO/IEC 9075-2
     | <new alternative>
```

By contrast, Format items that completely replace Format items in ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10 have BNF non-terminal symbols identical to BNF non-terminal symbols of Format items in ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10, but do not state that they include any alternatives from ISO/IEC 9075-2 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10.

New Format items that have no correspondence to any Format item in ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10 are not distinguished in the incremental part.

Format items in new Subclauses are unmarked.

6.2.5.3 New and modified paragraphs and rules

In modified Subclauses, each paragraph or Rule is marked to indicate whether it is a modification of a paragraph or Rule in ISO/IEC 9075-1, ISO/IEC 9075-2, ISO/IEC 9075-3, ISO/IEC 9075-5 or ISO/IEC 9075-10 or is a new paragraph or Rule added by this incremental part.

Modifications of paragraphs or Rules in ISO/IEC 9075-2 are identified by the inclusion of an indicative phrase enclosed in a box.

Replace the 5th paragraph means that the following text is to replace the fifth paragraph of the corresponding Subclause in ISO/IEC 9075-2.

Replace SR6) b) ii) means that the following text is to replace Syntax Rule 6)b)ii) of the corresponding Subclause in ISO/IEC 9075-2.

Augments SR3) means that the following text is to extend or enhance Syntax Rule 3). In most instances, the augmentation is the addition of a new alternative meant to support new syntax. New paragraphs or Rules in an incremental part is marked to indicate where it is to be inserted.

Insert before 2nd paragraph means that the following text is to be read as though it were inserted immediately before the second paragraph of the corresponding Subclause in ISO/IEC 9075-2.

Insert before GR4) means that the following text is to be read as though it were inserted immediately before General Rule 4) of the corresponding Subclause in ISO/IEC 9075-2.

If no specific insertion point is indicated, as in Insert this paragraph or Insert this GR, then the following text is to be read as though it were appended at the end of the appropriate section (the General Rules, for example) of the corresponding Subclause in ISO/IEC 9075-2.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Modifications of paragraphs or Rules in ISO/IEC 9075-1 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10 are identified in the same way as for modifications of ISO/IEC 9075-2, except that "in Part 1", "in Part 3 ", "in Part 5 " or "in Part 10" is appended to the indicative phrase, as appropriate.

In such indications, "SR" is used to mean "Syntax Rule", "AR" is used to mean "Access Rule", "GR" is used to mean "General Rule", and "CR" is used to mean "Conformance Rule". "Desc." is used to mean "Description" and "Func." is used to mean "Function".

All paragraphs, Format items, and Rules in new Clauses or Subclauses are also new and are therefore unmarked.

6.2.5.4 New and modified tables

If the name of a table in an incremental part is identical to that of a table in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10, then the table supplements the table in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10, typically by adding or replacing one or more table entries; otherwise, it is a new table.

In each incremental part, there is a table, Table 1, "Clause, Subclause, and Table relationships", that identifies the relationships between tables in that incremental part and the corresponding tables in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10.

The rows in modified tables are generally new rows to be effectively inserted into the corresponding table in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10, though in rare cases a row already in a table in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10 is effectively replaced by a row in the table in the incremental part. Such replacement is required wherever the value in the first column of the corresponding table is the same.

Annex B SQL Packages

1. *Rationale: Withdraw the package called "OLAP facilities", which is being superseded by the package called "OLAP" as defined in Amendment 1 of ISO/IEC 9075:1999.*

In Table 2 — SQL Packages, delete line 3:

Delete the entire Annex B.3, "OLAP facilities".

Annex B.9 SQL/MM Support

1. *Rationale: The package "SQL/MM support" is no longer needed since it is now defined explicitly in ISO/IEC 13249-1.*

Delete the entire Annex B.9, "SQL/MM support".

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ISO/IEC 9075-2:1999
Database Languages - SQL-Part 2:Foundation

3.1.1 Definitions taken from ISO/IEC 10646

1. Rationale: Remove unused definitions.

Replace the Subclause with:

This part of ISO/IEC 9075 makes use of the following terms defined in ISO/IEC 10646:

a) character

3.1.2 Definitions taken from Unicode

1. Rationale: Remove unused definitions.

Replace the Subclause with:

This part of ISO/IEC 9075 makes use of the following terms defined in The Unicode Standard:

a) control character

3.1.5 Definitions provided in Part 2

1. Rationale: Clarify the definition of "assignment"

Replace item b) with:

b) **assignment**: The operation whose effect is to ensure that the value at a site *T* (known as the target) is identical to a given value *S* (known as the source). Assignment is frequently indicated by the use of the phrase "*T* is set to *S*" or "the value of *T* is set to *S*".

2. Rationale: Clarify the definition of "comparable"

Replace item i) with:

i) **comparable** (of a pair of values): Capable of being compared, according to the rules of Subclause 8.2 "<comparison predicate>". In most, but not all, cases the values of a data type can be compared one with another. For the specification of comparability of individual data types, see Subclauses 4.2 to 4.11. Further, if a value of one data type can be compared with a value of another data type, then the two data types are said to be (mutually) comparable, see Subclause 4.12, "Type conversions and mixing of data types".

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Previous page is blank.

Not for Resale

3. Rationale: Correct and clarify the notion of "distinct".

Replace item l) with:

l) **distinct**: (of a pair of comparable values): Informally: not equal or not both null or having a pair of corresponding components that are distinct. Formally:

Two null values of comparable type are not distinct.

A null value and a nonnull value of comparable type are distinct.

For two nonnull values, the following rules apply:

— Two values of predefined type or reference type are distinct if and only if they are not equal.

— If two values *V1* and *V2* are of a user-defined type whose comparison form is RELATIVE or MAP and the result of comparing them for equality according to Subclause 8.2, "<comparison predicate>" is ***unknown***, then it is implementation-dependent whether they are distinct or not; otherwise, they are distinct if and only if they are not equal.

— If two values *V1* and *V2* are of a structured type whose comparison form is STATE, then they are distinct if their most specific types are different, or if there is an attribute *A* of their common most specific type such that the value of *A* in *V1* and the value of *A* in *V2* are distinct.

— Two rows (or partial rows) are distinct if and only if at least one of their pairs of respective fields is distinct.

— Two arrays are distinct if the arrays do not have the same cardinality or if, for arrays of the same cardinality, elements in the same ordinal position in the two arrays are distinct.

NOTE 0.1 — The result of evaluating whether or not two comparable values are distinct is never ***unknown***. The result of evaluating whether or not two values that are not comparable, for example, values of a user-defined type that has no comparison type, is not defined.

4. Rationale: Define "equals".

Insert the following definition:

o.1) **equal** (of a pair of comparable values): Yielding ***true*** if passed as arguments in a <comparison predicate>. See Subclause 8.2, "<comparison predicate>".

5. Rationale: Define "identical"

Insert the following definition:

s.1) **identical** (of a pair of values): Indistinguishable, in the sense that it is impossible, by any means available in SQL, to detect any difference between them. For the full definition, see Subclause 9.0 "Determination of identical values".

6. Rationale: Remove <identifier ignorable character>s from the definition of <white space>.

In item vv) delete the following bullet items:

— U+200C, Zero Width Non-Joiner

— U+200D, Zero Width Joiner

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

— U+200E, Left-To-Right Mark

— U+200F, Right-To-Left Mark

7. *Rationale: Add missing character to the definition of <white space>.*

In item vv) add the following bullet item:

— U+202F, Narrow No-Break Space

4.1 Data types

1. *Rationale: Change "user-defined data type" to "user-defined type".*

Replace the 3rd paragraph with:

SQL supports three sorts of data types: *predefined data types*, *constructed types*, and *user-defined types*. Predefined data types are sometimes called the "built-in data types", though not in this International Standard. User-defined types can be defined by a standard, by an implementation, or by an application.

2. *Rationale: Correct oversight in definition of directly based on.*

Replace the 11th paragraph with:

A structured type *ST* is *directly based on* a data type *DT* if any of the following are true:

a) *DT* is the declared type of some attribute of *ST*.

b) *DT* is the direct supertype of *ST*.

c) *DT* is a direct subtype of *ST*.

d) *DT* is compatible with *ST*.

3. *Rationale: Add definition of non-referentially based on.*

Add the following paragraph immediately after the 15th paragraph:

A data type *DT1* is non-referentially based on a data type *DT2* if *DT1* is not a reference type and is either directly based on *DT2* or is directly based on some data type that is non-referentially based on *DT2*.

4. *Rationale: Provide consistent rules for comparison operations.*

Add the following paragraphs after the last paragraph:

Ordering and comparison of values of the predefined data types requires knowledge only about those predefined data types. However, to be able to compare and order values of constructed types or of user-defined types, additional information is required. We say that some type *T* is *S*-ordered, for some set of types *S*, if, in order to compare and order values of type *T*, information about ordering at least one of the types in *S* is first required. A definition of *S*-ordered is required for several *S* (that is, for several sets of types), but not for all possible such sets.

The general definition of *S*-ordered is this:

Let *T* be a type and let *S* be a set of types. Then *T* is *S*-ordered if one of the following is true:

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

1. *T* is a member of *S*.

2. *T* is a row type and the declared type of some field of *T* is *S*-ordered.

3. *T* is an array type and the element type of *T* is *S*-ordered.

4. *T* is a structured type whose comparison form is STATE and the declared type of some attribute of *T* is *S*-ordered.

5. *T* is a user-defined type whose comparison form is MAP and the return type of the SQL-invoked function that is identified by the <map function specification> is *S*-ordered.

6. *T* is a reference type with a derived representation and the declared type of some attribute enumerated by the <derived representation> is *S*-ordered.

The notion of *S*-ordered is applied in the following definitions:

A type *T* is *LOB-ordered* if *T* is *L*-ordered, where *L* is the set of large object types.

A type *T* is *array-ordered* if *T* is *ARR*-ordered, where *ARR* is the set of array types.

A type *T* is *reference-ordered* if *T* is *REF*-ordered, where *REF* is the set of reference types.

A type *T* is *DT-EC-ordered* if *T* is *DTE*-ordered, where *DTE* is the set of distinct types with EQUALS ONLY comparison form (DT-EC stands for "distinct type - equality comparison")

A type *T* is *DT-FC-ordered* if *T* is *DTF*-ordered, where *DTF* is the set of distinct types with FULL comparison form.

A type *T* is *DT-NC-ordered* if *T* is *DTN*-ordered, where *DTN* is the set of distinct types with no comparison form.

A type *T* is *ST-EC-ordered* if *T* is *STE*-ordered, where *STE* is the set of structured types with EQUALS ONLY comparison form.

A type *T* is *ST-FC-ordered* if *T* is *STF*-ordered, where *STF* is the set of structured types with FULL comparison form.

A type *T* is *ST-NC-ordered* if *T* is *STN*-ordered, where *STN* is the set of structured types with no comparison form.

A type *T* is *ST-ordered* if *T* is either ST-EC-ordered, ST-FC-ordered or ST-NC-ordered.

A type *T* is *UDT-EC-ordered* if *T* is either DT-EC-ordered or ST-EC-ordered (UDT stands for "user-defined type").

A type *T* is *UDT-FC-ordered* if *T* is either DT-FC-ordered or ST-FC-ordered

A type *T* is *UDT-NC-ordered* if T is either DT-NC-ordered or ST-NC-ordered

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4.2.1 Character strings and collating sequences

1. Rationale: Remove redundant definition.

Replace the 4th paragraph with:

> The collating sequence used for a particular comparison is determined as in Subclause 4.2.3, "Rules determining collating sequence usage".

4.2.4 Named character sets

1. Rationale: Correct grammatical error.

Replace the 1st bullet of the 1st paragraph with:

— SQL_CHARACTER specifies the name of a character repertoire that consists of the 88 <SQL language character>s as specified in Subclause 5.1, "<SQL terminal character>". It consists of the 52 uppercase and lowercase simple latin characters, 10 digits, and 26 <SQL special character>s, including: <space>, <double quote>, <percent>, <ampersand>, <quote>, <left paren>, <right paren>, <asterisk>, <plus sign>, <comma>, <minus sign>, <period>, <solidus>, <colon>, <semicolon>, <less than operator>, <equals operator>, <greater than operator>, <question mark>, <underscore>, <vertical bar>, <left bracket>, <right bracket>, <circumflex>, <left brace>, and <right brace>. The 88 characters specified as <SQL language character>s are all included in the ISO International Reference Version (IRV) characters specified in ISO 646:1991.
The characters in IRV are included in many other international character set definitions. In addition, 82 of these characters (all except <vertical bar>, <left bracket>, <right bracket>, <circumflex>, <left brace>, and <right brace>) are in the most stable subset of IRV that, by ISO convention, is included in every latin-based ISO standard set of characters. As far as can be determined, <vertical bar>, <left bracket>, <right bracket>, <circumflex>, <left brace>, and <right brace> are included in most character sets that enjoy world-wide use. Thus, the SQL_CHARACTER repertoire is the most universal of the character sets named herein. The collation and form-of-use of SQL_CHARACTER are implementation-defined.

2. Rationale: Editorial.

Replace the 6th bullet of the 1st paragraph with:

— UTF16 and ISO10646 specify the name of a character repertoire that consists of every character represented by The Unicode Standard Version 2.0 and by ISO/IEC 10646 UTF-16, where each character is encoded using the UTF-16 encoding, occupying either 2 or 4 octets.

3. Rationale: Editorial.

Replace the 11th bullet of the 1st paragraph with:

> NOTE 4.1 — The character sets SQL_ CHARACTER, GRAPHIC_ IRV (or ASCII_ GRAPHIC), LATIN1, ISO8BIT (or ASCII_ FULL), and UNICODE (or ISO10646) have both a " floor" and " ceiling" requirement to consist of exactly the characters specified. Any character data type associated with one of these character sets has an implied integrity constraint limiting a value of the data type to be a character string consisting only of characters from the specified character set. The SQL_ TEXT and SQL_ IDENTIFIER character sets have a similar " floor" requirement in that they must contain all characters that are in other character sets supported by the implementation (for SQL- data and for <identifier> s, respectively); however, SQL_ TEXT and SQL_ IDENTIFIER do not have a " ceiling" requirement.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4.3.1 Binary string comparison

1. *Rationale: Remove redundant definition.*

Replace the paragraph with:

All binary strings are mutually comparable.

4.4.1 Bit string comparison and assignment

1. *Rationale: Remove redundant definition.*

Replace the paragraph with:

All bit strings are mutually comparable.

4.7.1 Datetimes

1. *Rationale: Correct and clarify Table 5.*

Add the following legend to Table 5, "Datetime data type conversions":

SV	Source value
TV	Target value
.UTC	UTC component of *SV* or *TV* (if source or target has time zone)
.TZ	Time zone component of *SV* or *TV* (if source or target has time zone)
STZD	SQL-session default time zone displacement
=>	Cast from the type before the arrow to the type after the arrow
TIME w/ TZ	TIME WITH TIME ZONE
TIME w/o TZ	TIME WITHOUT TIME ZONE
TS w/ TZ	TIMESTAMP WITH TIME ZONE
TS w/o TZ	TIMESTAMP WITHOUT TIME ZONE

In the table cell for the conversion from TIME WITHOUT TIME ZONE to TIMESTAMP WITHOUT TIME ZONE, change "*TZ*" to "*SV*".

In the table cell for the conversion from TIME WITH TIME ZONE to TIMESTAMP WITHOUT TIME ZONE, replace the cell contents with: *SV* => TS w/ TZ => TS w/o TZ.

In the table cell for the conversion from TIMESTAMP WITHOUT TIME ZONE to TIME WITHOUT TIME ZONE, change "*TZ*" to "*SV*"

In the table cell for the conversion from TIMESTAMP WITHOUT TIME ZONE to TIME WITH TIME ZONE, change "TS w/o TZ" to "TIME w/ TZ".

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4.7.3 "Operations involving datetimes and intervals"

1. *Rationale: Clarify the meaning of the <overlaps predicate>.*

Replace the 5th paragraph with:

<overlaps predicate> uses the operator OVERLAPS to determine whether or not two chronological periods overlap in time. A chronological period is specified either as a pair of datetimes (starting and ending) or as a starting datetime and an interval. If the length of the period is greater than 0, then the period consists of all points of time greater than or equal to the lower endpoint, and less than the upper endpoint. If the length of the period is 0, the period consists of a single point in time, the lower endpoint. Two periods overlap if they have at least one point in common.

2. *Rationale: Editorial - typographical error.*

Replace the 7th paragraph with:

<interval absolute value function> operates on an interval argument and returns its absolute value in the same most specific type.

4.8 User-defined types

1. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values and correct incorrect terminology and <method characteristic> should not include <transform group specification>. Correct use of <specific name> for method specifications by replacing with <specific method name>.*

Replace the 3rd paragraph with:

The definition of a user-defined type may include a <method specification list> consisting of one or more <method specification>s. A <method specification> is either an <original method specification> or an <overriding method specification>. Each <original method specification> specifies the <method name>, the <specific method name>, the <SQL parameter declaration list>, the <returns data type>, the <result cast from type> (if any), whether the method is type-preserving, the <language clause>, the <parameter style> if the language is not SQL, whether STATIC or CONSTRUCTOR is specified, whether the method is deterministic, whether the method possibly modifies SQL-data, possibly reads SQL-data, possibly contains SQL, or does not possibly contain SQL, and whether the method should be evaluated as the null value whenever any argument is the null value, without actually invoking the method.

2. *Rationale: Correct use of <specific name> for method specifications by replacing with <specific method name>.*

Replace the 4th paragraph with:

Each <overriding method specification> specifies the <method name>, the <specific method name>, the <SQL parameter declaration list> and the <returns data type>. For each <overriding method specification>, there must be an <original method specification> with the same <method name> and <SQL parameter declaration list> in some proper supertype of the user-defined type. Every SQL-invoked method in a schema must correspond to exactly one <original method specification> or <overriding method specification> associated with some user-defined type existing in that schema.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0859945 779

3. *Rationale: Correct the specification of implicit generation of cast functions when reference values are generated via <user-defined representation>.*

In the 8th paragraph, replace the 12th bullet with:

— If the user-defined type is a structured type, then whether the referencing type of the structured type has a user-defined representation, a derived representation, or a system-defined representation, and the type descriptor of the representation type of the referencing type of the structured type if user-defined representation is specified or the list of attributes if derived representation is specified.
NOTE 5 — "user-defined representation", "derived representation", and "system-defined representation" of a reference type are defined in Subclause 4.10, "Reference types".

4. *Rationale: Correct use of <specific name> for method specifications by replacing with <specific method name>.*

Replace the 1st sub-bullet in the 13th bullet of the 8th paragraph with:

- The <method name>.
- The <specific method name>.

5. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace the 8th sub-bullet in the 13th bullet of the 8th paragraph with:

- If the <method specification> is an <original method specification>, then an indication of whether STATIC or CONSTRUCTOR is specified.

4.8.1 Observers and mutators

1. *Rationale: Attribute values might be null. Clarify the effect of observers and mutuators.*

Replace the 3rd paragraph with:

Let *V* be a value in data type *T* and let *AV* be a value in data type *AT*. The invocation *A*(*V*, *AV*) returns *MV* such that *A*(*MV*) is identical to from *AV* and for every attribute *A'*(*A'* ≠ *A*)of *T*, *A'*(*MV*) is identical to *A'*(*V*). The most specific type of *MV* is the most specific type of *V*.

4.8.2 Constructors

1. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace the 1st and 2nd paragraphs with:

Associated with each structured type *ST* is one *constructor function*, implicitly defined when *ST* is defined. The constructor function is defined if and only if *ST* is instantiable.

Let *TN* be the name of a structured type *T*. The signature of the constructor function for *T* is *TN*( ) and its result data type is *T*. The invocation *TN*( ) returns a value *V* such that *V* is not null and, for every attribute *A* of *T*, *A*(*V*) returns the default value of *A*. The most specific type of *V* is *T*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

For every structured type *ST* that is instantiable, zero or more SQL-invoked constructor methods can be specified. The name of those methods must be equivalent to the name of the type for which they are specified.

NOTE 6.1 — SQL-invoked constructor methods are original methods that cannot be overloaded. An SQL-invoked constructor method and a regular function may exist such that both have equivalent function names, the types of the first parameters (if any) of the method's augmented parameter list and the function's parameter list are the same, and the types of the corresponding remaining parameters (if any) are identical according to the Syntax Rules of Subclause 10.14, "Data type identity".

4.8.4 User-defined type comparison and assignment

1. *Rationale: Values do not have declared types. Every user-defined type has a comparison type. Comparison forms and categories do not have to be the same throughout a subtype family.*

Replace the 6th paragraph with:

Two values *V1* and *V2* whose most specific types are user-defined types *T1* and *T2* are comparable if and only if *T1* and *T2* are in the same subtype family and each have some comparison type, *CT1* and *CT2* respectively. *CT1* and *CT2* constrain the comparison forms and comparison categories of *T1* and *T2* to be the same. Their respective comparison types *CT1* and *CT2* constrain the comparison forms and comparison categories of *T1* and *T2* to be the same, and the same as those of all their supertypes. If the comparison category is either STATE or RELATIVE, then the comparison functions of *T1* and *T2* are constrained to be equivalent; if the comparison category is MAP, they are not constrained to be equivalent.

4.10 Reference types

1. *Rationale: Correct the specification of implicit generation of cast functions when reference values are generated via <user-defined representation>.*

Replace the 2nd paragraph with:

Given a structured type *T*, the REF values that can reference rows in typed tables defined on *T* collectively form a certain data type *RT* known as a *reference type*. *RT* is the *referencing type* of *T*. *T* is the referenced type of *RT*.

2. *Rationale: Correct definition of the length of <reference type>s.*

Replace the 7th paragraph and its associated bullet items with:

A reference type is described by a reference type descriptor. A reference type descriptor includes:

— The name of the referenceable table, if any, that is the scope of the reference type.

— The name of the referenced type.

— The length of the referenced type.

3. *Rationale: Correct definition of the length of <reference type>s.*

Replace the 8th paragraph with:

In a host variable, a REF value is materialized as an *N*-octet value, where *N* is the length of the <reference type>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4.13 Data conversions

1. *Rationale: To clarify the concept of user-defined casts.*

Replace the 1st paragraph with:

Explicit data conversions can be specified by a *CAST operator*. A CAST operator defines how values of a source data type are converted into a value of a target data type according to the Syntax Rules and General Rules of Subclause 6.22, "<cast specification>". Data conversions between predefined data types and between constructed types are defined by the rules of this part of ISO/IEC 9075. Data conversions between a user-defined type and another data type are defined by a user-defined cast.

4.16 Tables

1. *Rationale: Clean up typed table insertability property for non-instantiable types.*

Replace the 7th paragraph with:

All base tables are *updatable*. Derived tables are either updatable or not updatable. The operations of update and delete are permitted for updatable tables, subject to constraining Access Rules. Some updatable tables, including all base tables whose row type is not derived from a user-defined type that is not instantiable, are also *insertable-into*, in which case the operation of insert is also permitted, again subject to Access Rules.

2. *Rationale: Clean up typed table insertability property for non-instantiable types.*

Replace the 11th paragraph with:

Every table descriptor includes:

— The column descriptor of each column in the table.

— The name, if any, of the structured type, if any, associated with the table.

— An indication of whether the table is insertable-into or not.

— An indication of whether the base table is a referenceable table or not, and an indication of whether the self-referencing column is a system-generated, a user-generated, or a derived self-referencing column.

— A list, possibly empty, of the names of its direct supertables.

— A list, possibly empty, of the names of its direct subtables.

3. *Rationale: Clean up typed table insertability property for non-instantiable types.*

Replace the 13th paragraph with:

A derived table descriptor describes a derived table. In addition to the components of every table descriptor, a derived table descriptor includes:

— The <query expression> that defines how the table is to be derived.

— An indication of whether the derived table is updatable or not.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4.16.3 Operations involving tables

1. Rationale: Correct the description of the effect of deleting, updating and inserting a row in a table.

Replace the 4th, 5th and 6th paragraphs with:

Let *T* be a table whose value is *TV1*. For every row *RR*, let n_{RR} be the number of rows in *TV1* that are identical to *RR*.

The effect of deleting a row *R* from *T* is to replace the value *TV1* of *T* with the value *TV2* such that, for *RR* identical to *R*, the number of rows in *TV2* that are identical to *RR* is n_{RR} - 1 (one) and for *RR* not identical to *R*, the number of rows in *TV2* that are not identical to *RR* is n_{RR}. The primary effect of a <delete statement: positioned> on *T* is to delete exactly one specified row from *T*. The primary effect of a <delete statement: searched> on *T* is to delete zero or more rows from *T*.

The effect of replacing a row *R1* with the row *R2* in *T* is to replace the value *TV1* of *T* with the value *TV2* such that, if *R1* is not identical to *R2*, then, for *RR* identical to *R1*, the number of rows in *TV2* that are identical to *RR* is n_{RR} - 1 (one), for *RR* identical to *R2*, the number of rows in *TV2* that are identical to *RR* is n_{RR} + 1 (one), and for *RR* identical to neither *R1* nor *R2*, the number of rows in *TV2* that are identical to *RR* is n_{RR}. Otherwise *R1* is identical to *R2* and the number of rows in *TV2* that are identical to *RR* is n_{RR} for all *RR*.

The primary effect of an <update statement: positioned> on *T* is to replace exactly one specified row in *T* with some specified row. The primary effect of an <update statement: searched> on *T* is to replace zero or more rows in *T*.

If *T*, as well as being updatable, is insertable-into, then rows can be inserted into it. The effect of inserting a row *R* into *T* is to replace the value *TV1* of *T* with the value *TV2* such that, for *RR* identical to *R*, the number of rows in *TV2* that are identical to *RR* is n_{RR} + 1 (one) and, for *RR* not identical to *R*, the number of rows in *TV2* that are identical to *RR* is n_{RR}. The primary effect of an <insert statement> on *T* is to insert into *T* each of the zero or more rows contained in a specified table.

4.18.1 General rules and definitions

1. Rationale: Use the notion of distinct.

Replace the 2nd paragraph with:

Let "*T*: *A* ↠ *B*" (read "in *T*, *A* determines *B*" or "*B* is functionally dependent on *A* in *T*") denote the functional dependency of *B* on *A* in *T*, which is true if, for any possible value of *T*, any two rows that are not distinct for every column in *A* also are not distinct for every column in *B*. When the table *T* is understood from context, the abbreviation "*A* ↠ *B*" may also be used.

2. Rationale: Correct erroneous symbol.

Replace the 8th paragraph with:

In the following Subclauses, let a column *C1* be a *counterpart* of a column *C2* under qualifying table *QT* if *C1* is specified by a column reference (or by a <value expression> that is a column reference) that references *C2* and *QT* is the qualifying table of *C2*. If *C1* is a counterpart of *C2* under qualifying table *QT1* and *C2* is a counterpart of *C3* under qualifying table *QT2*, then *C1* is a counterpart of *C3* under *QT2*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4.18.9 Known functional dependencies in the result of <having clause>

1. Rationale: More than one <search condition> may be directly contained in a <having clause>.

Replace the 1st paragraph with:

Let *T1* be the table that is the operand of the <having clause>, let *SC* be the <search condition> simply contained in the <having clause>, and let *R* be the result of the <having clause>.

4.20 SQL-schemas

1. Rationale: Cleanup of imprecise wording.

Replace the 5th paragraph with:

Base tables and views are identified by <table name>s. A <table name> consists of a <schema name> and an <identifier>. The <schema name> identifies the schema in which a persistent base table or view identified by the <table name> is defined. Base tables and views defined in different schemas can have <identifier>s that are equal according to the General Rules of Subclause 8.2, "comparison predicate>".

4.21 SQL-client modules

1. Rationale: Editorial.

Replace NOTE 29 with:

NOTE 29 — The <module character set specification> has no effect on the SQL language contained in the SQL-client module and exists only for compatibility with ISO/IEC 9075:1992. It may be used to document the character set of the SQL-client module.

4.23 SQL-invoked routines

1. Rationale: Editorial.

Replace the 4th, 5th and 6th paragraphs with:

A <predicate> *P* is said to be dependent on an SQL-invoked routine *SR* if and only if *SR* is the ordering function included in the descriptor of a user-defined type *UDT* and one of the following conditions is true:

— *P* is a <comparison predicate> that immediately contains a <row value expression> whose declared type is some user-defined type *T1* whose comparison type is *UDT*.

— *P* is a <quantified comparison predicate> that immediately contains a <row value expression> that has some field whose declared type is some user-defined type *T1* whose comparison type is *UDT*.

— *P* is a <unique predicate> that immediately contains a <table subquery> that has a column whose declared type is some user-defined type *T1* whose comparison type is *UDT*.

— *P* is a <match predicate> that immediately contains a <row value expression> that has some field whose declared type is some user-defined type *T1* whose comparison type is *UDT*.

— *P* is a <comparison predicate> with some corresponding value whose declared type is some array type whose element type is a user-defined type *T1* whose comparison type is *UDT*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

— *P* is a <quantified comparison predicate> that immediately contains a <row value expression> that has some field whose declared type is some array type whose element type is a user-defined type *T1* whose comparison type is *UDT*.

— *P* is a <unique predicate> that immediately contains a <table subquery> that has a column whose declared type is some array type whose element type is a user-defined type *T1* whose comparison type is *UDT*.

— *P* is a <match predicate> that immediately contains a <row value expression> that has some field whose declared type is some array type whose element type is a user-defined type *T1* whose comparison type is *UDT*.

NOTE 30 — "Comparison type" is defined in Subclause 4.8.4, "User-defined type comparison and assignment".

A <set function specification> *SFS* is said to be dependent on an SQL-invoked routine *SR* if and only if all the following are true:

— *SR* is the ordering function included in the descriptor of a user-defined type *UDT*.

— *SFS* is a <general set function> whose <set function type> *SFS* is MAX or MIN or *SFS* is a <general set function> whose <set qualifier> is DISTINCT.

— The declared type of the <value expression> of *SFS* is *UDT*.

A <group by clause> *GBC* is said to be dependent on an SQL-invoked routine *SR* if and only if all the following are true:

— *SR* is the ordering function included in the descriptor of a user-defined type *UDT*.

— The declared type of a grouping column of *GBC* is *UDT*.

2. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace the 9th paragraph that begins with "An SQL-invoked method is an SQL-invoked function ..." with:

An *SQL-invoked method* is an SQL-invoked function that is specified by <method specification designator> (see Subclause 11.49, "<SQL-invoked routine>"). There are three kinds of SQL-invoked methods: *instance SQL-invoked methods*, *SQL-invoked constructor methods*, and *static SQL-invoked methods*. All SQL-invoked methods are associated with a user-defined type, also known as the *type of the method*. The <method characteristic>s of an SQL-invoked method are specified by a <method specification> contained in the <user-defined type definition> of the type of the method. An instance SQL-invoked method and an SQL-invoked constructor method both satisfy the following conditions:

— Its first parameter, called the *subject parameter*, has a declared type that is a user-defined type. The type of the subject parameter is the type of the method. A parameter other than the subject parameter is called an *additional parameter*.

— Its descriptor is in the same schema as the descriptor of the data type of its subject parameter.

An SQL-invoked constructor method satisfies the following additional condition:

— Its <method name> is equivalent to the <qualified identifier> simply contained in the <user-defined type name> included in the user-defined type descriptor of the type of the method.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0859951 T72

3. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace the 16th paragraph which begins with "SQL-invoked routines are invoked differently..." with:

SQL-invoked routines are invoked differently depending on their form. SQL-invoked procedures are invoked by <call statement>s. SQL-invoked regular functions are invoked by <routine invocation>s. Instance SQL-invoked methods are invoked by <method invocation>s, while SQL-invoked constructor methods are invoked by <new invocation>s and static SQL-invoked methods are invoked by <static method invocation>s. An invocation of an SQL-invoked routine specifies the <routine name> of the SQL-invoked routine and supplies a sequence of argument values corresponding to the <SQL parameter declaration>s of the SQL-invoked routine. A *subject routine* of an invocation is an SQL-invoked routine that may be invoked by a <routine invocation>. After the selection of the subject routine of a <routine invocation>, the SQL arguments are evaluated and the SQL-invoked routine that will be executed is selected. If the subject routine is an instance SQL-invoked method, then the SQL-invoked routine that is executed is selected from the set of overriding methods of the subject routine. (The term "set of overriding methods" is defined in the General Rules of Subclause 10.4, "<routine invocation>".) The overriding method that is selected is the overriding method with a subject parameter the type designator of whose declared type precedes that of the declared type of the subject parameter of every other overriding method in the type precedence list of the most specific type of the value of the SQL argument that corresponds to the subject parameter. See the General Rules of Subclause 10.4, "<routine invocation>". If the subject routine is not an SQL-invoked method, then the SQL-invoked routine executed is that subject routine. After the selection of the SQL-invoked routine for execution, the values of the SQL arguments are assigned to the corresponding SQL parameters of the SQL-invoked routine and its <routine body> is executed. If the SQL-invoked routine is an SQL routine, then the <routine body> is an <SQL procedure statement> that is executed according to the General Rules of <SQL procedure statement>. If the SQL-invoked routine is an external routine, then the <routine body> identifies a program written in some standard programming language other than SQL that is executed according to the rules of that standard programming language.

4. *Rationale: Editorial.*

Replace the 20th paragraph, which begins "If a <routine invocation> is contained in a <query expression> of a view, a check constraint, or an assertion, ...", with:

If a <routine invocation> is contained in a <query expression> of a view, a check constraint, or an assertion, the <triggered action> of a trigger, or in an <SQL-invoked routine>, then the subject routine for that invocation is determined at the time the view is created, the check constraint is defined, the assertion is created, the trigger is created, or the SQL-invoked routine is created. If the subject routine is an SQL-invoked procedure, an SQL-invoked regular function, or a static SQL-invoked method, then the same SQL-invoked routine is executed whenever the view is used, the check constraint or assertion is evaluated, the trigger is executed, or the SQL-invoked routine is invoked. If the subject routine is an instance SQL-invoked method, then the SQL-invoked routine that is executed is determined whenever the view is used, the check constraint or assertion is evaluated, the trigger is executed, or the SQL-invoked routine is invoked, based on the most specific type of the value resulting from the evaluation of the SQL argument that correspond to the subject parameter. See the General Rules of Subclause 10.4, "<routine invocation>".

5. *Rationale: Clarify the semantics of SQL-data access indication.*

Replace the 23rd paragraph, which begins "An external routine either *does not possibly contain SQL* or *possibly contains SQL*...", with:

An external routine *does not possibly contain SQL*, *possibly contains SQL*, *possibly reads SQL-data*, or *possibly modifies SQL-data*. Only an external routine that possibly contains SQL, or possibly reads SQL-data or possibly modifies SQL-data may execute SQL-statements during its invocation.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Replace the 24th paragraph, which begins "An SQL-invoked routine may or may not *possibly read SQL-data*.", with:

An SQL routine *possibly contains SQL*, *possibly reads SQL-data*, or *possibly modifies SQL-data*. Only an SQL-invoked routine that possibly reads SQL-data or possibly modifies SQL-data may read SQL-data during its invocation.

Replace the 25th paragraph, which begins "An SQL-invoked routine may or may not *possibly modify SQL-data*.", with:

Only an SQL-invoked routine that *possibly modifies SQL-data* may modify SQL-data during its invocation.

6. *Rationale: Missing word.*

Replace the 28th paragraph, which begins "The identifiers in this new entry of the authorization stack are then modified", with:

The identifiers in this new entry of the authorization stack are then modified depending on whether the SQL-invoked routine is an SQL routine or an external routine. If the SQL-invoked routine is an SQL routine, then, if the routine authorization identifier is a user identifier, the user identifier is set to the routine authorization identifier and the role name is set to null; otherwise, the role name is set to the routine authorization identifier and the user identifier is set to null.

4.24 Built-in functions

1. *Rationale: Remove undefined term.*

Replace the 1st paragraph with:

Certain operators whose invocation employs syntax similar to that of <routine invocation> are designated built-in functions. Those that are so designated are those that appear in the result of the following <query expression>:

```
SELECT DISTINCT ROUTINE_NAME
  FROM INFORMATION_SCHEMA.ROUTINES
 WHERE SPECIFIC_SCHEMA = 'INFORMATION_SCHEMA'
```

4.25 SQL-paths

1. *Rationale: Use the correct BNF non-terminal (<user-defined type> rather than <user-defined type name>).*

Replace the 3rd paragraph with:

— The user-defined type when the <user-defined type> does not contain a <schema name>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4.26.4 Locators

1. Rationale: Correct the specification of which locators are marked invalid when an SQL-transaction ends.

Replace 5th paragraph with:

A non-holdable locator remains valid until the end of the SQL-transaction in which it was generated, unless it is explicitly made invalid by the execution of a <free locator statement> or a <rollback statement> that specifies a <savepoint clause> is executed before the end of that SQL-transaction if the locator was generated subsequent to the establishment of the savepoint identified by the <savepoint clause>.

Replace 6th paragraph with:

A holdable locator may remain valid beyond the end of the SQL-transaction in which it is generated. A holdable locator becomes invalid whenever a <free locator statement> identifying that locator is executed or the SQL-transaction in which it is generated or any subsequent SQL-transaction is rolled back. All locators remaining valid at the end of an SQL-session are marked invalid when that SQL-session terminates.

4.30.1 Classes of SQL-statements

1. Rationale: Clarify the semantics of SQL-data access indication.

Replace the 2nd paragraph with:

In this part of ISO/IEC 9075, there are at least three ways of classifying SQL-statements:

— According to their effect on SQL objects, whether persistent objects, i.e., SQL-data, SQL-client modules, and schemas, or transient objects, such as SQL-sessions and other SQL-statements.

— According to whether or not they start an SQL-transaction, or can, or must, be executed when no SQL-transaction is active.

— According to whether they do not possibly contain SQL, possibly contain SQL, possibly read SQL-data, or possibly modify SQL-data.

4.30.2 SQL-statements classified by functions

1. Rationale: Correct the classification of SQL-statements.

Replace the 13th bullet of the 2nd paragraph with:

— <grant privilege statement>

2. Rationale: Correct the classification of SQL-statements.

Add the following bullets to the 2nd paragraph:

— <user-defined cast definition>

— <drop user-defined cast statement>

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

3. Rationale: Clarify the semantics of SQL-data access indication.

Add the following Subclause after Subclause 4.30.2, "SQL-statements classified by functions":

4.30.2a SQL-statements and SQL-data access indication

The following SQL-statements possibly contain SQL:

— SQL-control statements

— SQL-control declaration statements

— SQL-session statements

— SQL-diagnostics statements

— <free locator statement>

— <hold locator statement>

The following SQL-statements possibly read SQL-data:

— SQL-data statements other than SQL-data change statements, <free locator statement>, or and <hold locator statement>

— SQL-statements that simply contain a <subquery>

The following SQL-statements possibly modify SQL-data:

— SQL-schema statements

— SQL-data change statements

NOTE 31.1 — SQL-transaction statements, SQL-connection statements and SQL-dynamic statements are not included in the above classification since they are not permitted to occur in SQL-invoked routines.

4.30.3 SQL-statements and transaction states

1. Rationale: Correct the classification of SQL-statements.

Replace the 3rd paragraph with:

The following SQL-statements are possibly transaction-initiating SQL-statements:

— <return statement>

— <call statement>

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4.30.4 SQL-statement atomicity

1. Rationale: Add missing concept.

Add the following paragraphs after the 1st paragraph:

Depending on the type of SQL-statement being executed , a statement execution context may be said to be either atomic or non-atomic.

A *non-atomic execution context* is said to be active during the execution of a non-atomic SQL-statement.

4.31.1 Authorization identifiers

1. Rationale: Editorial.

Replace the 1st paragraph that begins with "An <authorization identifier> identifies a set of privileges. ..." with:

An <authorization identifier> identifies a set of privileges. An <authorization identifier> can be either a <user identifier> or a <role name>. A <user identifier> represents a user of the database system. The mapping of <user identifier>s to operating system users is implementation-dependent.
A <role name> represents a role.

4.34.1 Execution contexts

1. Rationale: Correct cross reference.

Replace the 1st paragraph with:

Execution contexts augment an SQL-session context to cater for certain special circumstances that might pertain from time to time during invocations of SQL-statements. An execution context is either a trigger execution context or a routine execution context. There is always a statement execution context, a routine execution context, and zero or more trigger execution contexts. For certain SQL-statements, the execution context is always atomic. A routine execution context is either atomic or non-atomic. Every trigger execution context is atomic. Statement execution contexts are described in Subclause 4.30.4, "SQL-statement atomicity". Trigger execution contexts are described in Subclause 4.35, "Triggers".

2. Rationale: Clarify the semantics of SQL-data access indication.

Replace the 2nd paragraph with:

A routine execution context consists of:

— An indication as to whether or not an SQL-invoked routine is active.

— An SQL-data access indication, which indicates what SQL-statements, if any, are allowed during the execution of an SQL-invoked routine. The SQL-data access indication is one of the following: does not possibly contain SQL, possibly contains SQL, possibly reads SQL-data, or possibly modifies SQL-data.

— An identification of the SQL-invoked routine that is active.

— The routine SQL-path derived from the routine SQL-path if the SQL-invoked routine that is active is an SQL routine and from the external routine SQL-path if the SQL-invoked routine that is active is an external routine.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4.35.2 Execution of triggers

1. Rationale: Editorial.

Replace the 2nd paragraph that begins with "A trigger execution context consists of a set of ..." with:

A trigger execution context consists of a set of *state changes*. Within a trigger execution context, each state change is uniquely identified by a *trigger event*, a *subject table*, and a *column list*. The trigger event can be DELETE, INSERT, or UPDATE. A state change *SC* contains a *set of transitions*, a set of statement-level triggers *considered as executed* for *SC*, and a set of row-level triggers, each paired with the set of rows in *SC* for which it is considered as executed.

2. Rationale: Clarify the notion of state changes.

Replace the 9th paragraph with:

When execution of an SQL-data change statement S_i causes a trigger execution context *TEC* to come into existence, the set of state changes SSC_i is empty. Let *TE* be one of DELETE, INSERT, or UPDATE contained in S_i. Let *ST* be the subject table of S_i. If *TE* is INSERT or DELETE, then let *PSC* be a set whose only element is the empty set. If *TE* is UPDATE, then let *CL* be the list of columns being updated by S_i, and let *OC* be the set of column names identifying the columns in *CL*. Let *PSC* be the set consisting of the empty set and every subset of the set of column names of *ST* that has at least one column name that is in *OC*. Let *PSCN* be the number of elements in *PSC*. A state change SC_{ij} for *j* varying from 1 (one) to *PSCN*, identified by *TE*, *ST*, and the *j*-th element in *PSC*, is added to SSC_i, provided SSC_i does not already contain a state change corresponding to SC_{ij}. A transition T_{ijk} is added to SC_{ij} when a row is inserted into, updated in, or deleted from *ST* during the execution of S_i or the checking of referential constraints according to the General Rules of Subclause 11.8, "<referential constraint definition>", Subclause 14.6, "<delete statement: positioned>", Subclause 14.7, "<delete statement: searched>", Subclause 14.8, "<insert statement>", Subclause 14.9, "<update statement: positioned>", and Subclause 14.10, "<update statement: searched>".

3. Rationale: Correct the use of unidentified identifiers.

Replace the 12th paragraph with:

When a state change SC_{ij} arises in SSC_i, one or more triggers are activated by SC_{ij}. A trigger *TR* is activated by SC_{ij} if and only if the subject table of *TR* is the subject table of SC_{ij}, the trigger event of *TR* is the trigger event of SC_{ij}, and the set of column names listed in the trigger column list of *TR* is the equivalent to the set of column names of SC_{ij}.

NOTE 33.1 — The trigger column list is included in the descriptor of *TR*; it is empty if the trigger event is DELETE or INSERT. The trigger column list is also empty if the trigger event is UPDATE, but the <trigger event> of the <trigger definition> that defined *TR* does not specify a <trigger column list>.

5.2 <token> and <separator>

1. Rationale: <underscore> is already defined in <identifier part>.

In the Format, replace the production for <identifier body> with:

```
<identifier body> ::= <identifier start> [ { <identifier part> }... ]
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2. *Rationale: Editorial. Misplaced and incorrect reserved words.*

In the Format, in the production for <non-reserved word>, delete the texts:

```
BETWEEN
BITVAR
DYNAMIC_FUNCTION
DYNAMIC_FUNCTION_CODE
EXISTING
EXISTS
HOLD
INFIX
METHOD
OVERLAPS
SIMILAR
SUBLIST
USER_DEFINED_TYPE_CATALOG
USER_DEFINED_TYPE_NAME
USER_DEFINED_TYPE_SCHEMA
```

3. *Rationale: Editorial. Misplaced and incorrect reserved words.*

In the Format, in the production for <non-reserved word>, add the texts:

```
ADMIN
ATTRIBUTE
CHARACTERISTICS
EVERY
DERIVED
ORDERING
SCOPE
STATEMENT
```

in the correct alphabetical order.

4. *Rationale: Editorial. Misplaced and incorrect reserved words.*

In the Format, in the production for <reserved word>, delete the texts:

```
ADMIN
AGGREGATE
ALIAS
CLASS
COMPLETION
DESTROY
DESTRUCTOR
DICTIONARY
EVERY
HOST
IGNORE
INITIALIZE
ITERATE
LESS
LIMIT
MODIFY
OFF
OPERATION
PARAMETERS
POSTFIX
PREFIX
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

```
PREORDER
SCOPE
SEQUENCE
STATEMENT
STRUCTURE
TERMINATE
THAN
VARIABLE
```

5. *Rationale: Editorial. Misplaced and incorrect reserved words.*

In the Format, in the production for <reserved word>, add the texts:

```
BETWEEN
EXISTS
HOLD
METHOD
OVERLAPS
RELEASE
SIMILAR
```

in the correct alphabetical order.

6. *Rationale: Use case-normal forms consistently to define equivalent identifiers.*

Delete NOTE 24.

Delete Syntax Rule 23)

Replace Syntax Rule 24) with:

24) The case- normal form of a <regular identifier> shall not be equal, according to the comparison rules in Subclause 8.2, "< comparison predicate>", to any <reserved word> (with every letter that is a lower- case letter replaced by the corresponding upper- case letter or letters), treated as the repetition of a <character string literal> that specifies a <character set specification> of SQL_ IDENTIFIER.

Replace Syntax Rules 25) and 26) with:

25) Two <regular identifier>s are equivalent if the case-normal forms of their <identifier body>s, considered as the repetition of a <character string literal> that specifies a <character set specification> of SQL_IDENTIFIER and an implementation-defined collation IDC that is sensitive to case, compare equally according to the comparison rules in Subclause 8.2, "<comparison predicate>".

26) A <regular identifier> and a <delimited identifier> are equivalent if the case-normal form of the <identifier body> of the <regular identifier> and the <delimited identifier body> of the <delimited identifier> (with all occurrences of <quote> replaced by <quote symbol> and all occurrences of <doublequote symbol> replaced by <double quote>), considered as the repetition of a <character string literal> that specifies a <character set specification> of SQL_IDENTIFIER and collation IDC, compare equally according to the comparison rules in Subclause 8.2, "<comparison predicate>".

7. *Rationale: <underscore> is already defined in <identifier part>.*

Replace Conformance Rule 1) with:

1) Without Feature F391, "Long identifiers", in a <regular identifier>, the number of <identifier part>s shall be less than 18.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5.3 <literal>

1. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".*

Replace Conformance Rule 7) with:

7) Without Feature F461, "Named character sets", a <character string literal> shall not specify a <character set specification>.

2. *Rationale: Editorial.*

Replace Conformance Rules 8) and 9) with:

8) Without Feature F411, "Time zone specification", conforming SQL shall not specify a <time zone interval>.

9) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain any <binary string literal>.

5.4 Names and identifiers

1. *Rationale: Use "equivalent" instead of "same" when comparing identifiers.*

Replace Syntax Rules 13), 14) and 15) with:

13) Two <schema qualified name>s are equivalent if and only if their <qualified identifier>s are equivalent and their <schema name>s are equivalent, regardless of whether the <schema name>s are implicit or explicit.

13.1) Two <local or schema qualified name>s are equivalent if and only if their <qualified identifier>s are equivalent and either they both specify MODULE or they both specify or imply <schema name>s that are equivalent.

14) Two <character set name>s are equivalent if and only if their <SQL language identifier>s are equivalent and their <schema name>s are equivalent, regardless of whether the <schema name>s are implicit or explicit.

15) Two <schema name>s are equivalent if and only if their <unqualified schema name>s are equivalent and their <catalog name>s are equivalent, regardless of whether the <catalog name>s are implicit or explicit.

2. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".*

Replace Conformance Rule 12) with:

12) Without Feature F461, "Named character sets", conforming SQL language shall not contain any <character set name>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

6.1 <data type>

1. Rationale: Delete a redundant rule.

Delete General Rule 11)

2. Rationale: Use the correct BNF non-terminal (<user-defined type> rather than <user-defined type name>).

Replace Conformance Rule 1) with:

1) Without Feature S023, "Basic structured types", <user-defined type>, if specified, shall not identify a structured type.

3. Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".

Replace Conformance Rule 8) with:

8) Without Feature F461, "Named character sets", a <data type> shall not specify CHARACTER SET.

6.3 <value specification> and <target specification>

1. Rationale: Editorial.

Replace Syntax Rule 7) with:

7) A <target specification> or <simple target specification> that is a <column reference> shall be a new transition variable column reference.
NOTE 65 — "New transition variable column reference" is defined in Subclause 6.5, "<identifier chain>".

8) Let *X* denote either a column *C* or the <key word> VALUE. Given a <boolean value expression> *BVE* and *X*, the notion "*BVE* is a known-not-null condition for *X*" is defined recursively as follows:

a) If *BVE* is a <predicate>, then

Case:

i) If *BVE* is a <predicate> of the form "*RVE* IS NOT NULL", where *RVE* is a <row value expression> that simply contains a <row value constructor element> that is a <column reference> that references *C*, then *BVE* is a *known-not-null condition* for *C*.

ii) If *BVE* is the <predicate> "VALUE IS NOT NULL", then *BVE* is a *known-not-null condition* for VALUE.

iii) Otherwise, *BVE* is not a known-not-null condition for *X*.

b) If *BVE* is a <value expression primary>, then

Case:

i) If *BVE* is of the form "<left paren> <value expression> <right paren>" and the <value expression> is a known-not-null condition for *X*, then *BVE* is a *known-not-null condition* for *X*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ii) Otherwise, *BVE* is not a known-not-null condition for *X*.

c) If *BVE* is a <boolean test>, then let *BP* be the <boolean primary> immediately contained in *BVE*. If *BP* is a known-not-null condition for *X*, and <truth value> is not specified, then *BVE* is a *known-not-null condition* for *X*. Otherwise, *BVE* is not a known-not-null condition for *X*.

d) If *BVE* is of the form "NOT *BT*", where *BT* is a <boolean test>, then

Case:

i) If *BT* is "*CR* IS NULL", where *CR* is a column reference that references column *C*, then *BVE* is a *known-not-null condition* for *C*.

ii) If *BT* is "VALUE IS NULL", then *BVE* is a *known-not-null condition* for VALUE.

iii) Otherwise, *BVE* is not a known-not-null condition for *X*.

NOTE 66 — For simplicity, the rules do not attempt to analyze conditions such as "NOT NOT A IS NULL", or "NOT (A IS NULL OR NOT (B = 2))"

e) If *BVE* is of the form "*BVE1* AND *BVE2*", then

Case:

i) If either *BVE1* or *BVE2* is a known-not-null condition for *X*, then *BVE* is a *known-not-null condition* for *X*.

ii) Otherwise, *BVE* is not a known-not-null condition for *X*.

f) If *BVE* is of the form "*BVE1* OR *BVE2*", then *BVE* is not a known-not-null condition for *X*.

NOTE 67 — For simplicity, this rule does not detect cases such as "A IS NOT NULL OR A IS NOT NULL", which might be classified as a known-not-null condition.

6.5 <identifier chain>

1. Rationale: Take query names into account.

Replace Syntax Rule 7) a) with:

7) a) If $N = 1$ (one), then *IC* shall be contained within the scope of one or more exposed <table or query name>s or <correlation name>s whose associated tables include a column whose <identifier> is equivalent to I_1 or within the scope of a <routine name> whose associated <SQL parameter declaration list> includes an SQL parameter whose <identifier> is equivalent to I_1. Let the phrase *possible scope tags* denote those exposed <table or query name>s, <correlation name>s, and <routine name>s.

Case:

i) If the number of possible scope tags in the innermost scope containing a possible scope tag is 1 (one), then

Case:

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

1) If the innermost possible scope tag is a <table or query name> or <correlation name>, then let *T* be the table associated with the possible scope tag, and let *C* be the column of *T* whose <identifier> is equivalent to I_1. *PIC1* is the basis of *IC*, the basis length is 1 (one), the basis scope is the scope of *T*, and the basis referent is *C*.

2) If the innermost possible scope tag is a <routine name>, then let *SP* be the SQL parameter whose <identifier> is equivalent to I_1. *PIC1* is the basis of *IC*, the basis length is 1 (one), the basis scope is the scope of *SP*, and the basis referent is *SP*.

ii) Otherwise, each possible scope tag shall be a <table or query name> or a <correlation name> of a <table reference> that is directly contained in a <joined table> *JT*. I_1 shall be a common column name in *JT*. Let *C* be the column of *JT* that is identified by I_1. *PIC1* is the basis of *IC*, the basis length is 1 (one), and the basis referent is *C*.

NOTE 72 — "Common column name" is defined in Subclause 7.7, "<joined table>".

2. *Rationale: Editorial. Add missing text.*

Replace Syntax Rule 11) with:

11) A <basic identifier chain> whose basis referent is a column is a *column reference*. If the basis length is 2, and the basis scope is a <trigger definition> whose <trigger action time> is BEFORE, and I_1 is equivalent to the <new values correlation name> of the <trigger definition>, then the column reference is a *new transition variable column reference*.

6.11 <method invocation>

1. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

In the Format add the following production:

<constructor method selection> ::= <routine invocation>

2. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Delete Syntax Rule 3).

3. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace Syntax Rule 5) with:

5) Case:

a) If <method invocation> is immediately contained in <new invocation>, then let *TP* be an SQL-path containing the <schema name> of schema that includes the descriptor of *UDT*.

b) Otherwise, let *TP* be an SQL-path, arbitrarily defined, containing the <schema name> of every schema that includes a descriptor of a supertype or subtype of *UDT*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4. *Rationale: Provide correct logical structure.*

Replace Syntax Rule 6) with

6) Case:
   a) If <generalized invocation> is specified, then let *DT* be the <data type> simply contained in the <generalized invocation>. Let *RI* be the following <method selection>:

      ```
      MN ( VEP AS DT AL )
      ```

   b) Otherwise,

      Case:

      i) If <method invocation> is immediately contained in <new invocation>, then let *RI* be the following <constructor method selection>:

         ```
         MN ( VEP AL )
         ```

      ii) Otherwise, let *RI* be the following <method selection>:

         ```
         MN ( VEP AL )
         ```

6.16 <set function specification>

1. *Rationale: Disallow DISTINCT on columns with types based on LOB and array types. Provide consistent Syntax Rules for comparison operations.*

Add the following Syntax Rule:

6.1) If the <set function specification> specifies a <general set function> whose <set quantifier> is DISTINCT, then *DT* shall not be LOB-ordered, array-ordered, UDT-EC-ordered, or UDT-NC-ordered.

Delete Syntax Rule 7).

2. *Rationale: Provide consistent Syntax Rules for comparison operations.*

Replace Syntax Rule 8) with:

8) If the <set function specification> specifies a <set function type> that is MAX or MIN, then DT shall not be LOB-ordered, array-ordered, reference-ordered, UDT-EC-ordered, UDT-NC-ordered, or a row type.

Delete Syntax Rule 10).

3. *Rationale: Changes to <group by clause> have changed the treatment of <grouping operation>. Editorial - Erroneous text not deleted.*

Replace General Rules 3) h) and 3) i) with:

NOTE 78.1 — the value of <grouping operation> is computed by means of syntactic transformations in 7.9 "<group by clause>".

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4. *Rationale: Correcting the Feature Name of Feature T431 in the Conformance Rules of <set function specification>.*

Replace Conformance Rule 7) with:

7) Without Feature T431, "Extended grouping capabilities", conforming SQL language shall not contain a <set function specification> that is a <grouping operation>.

5. *Rationale: Provide consistent Conformance Rules for comparison operations.*

Replace Conformance Rule 8) with:

8) Without Feature S024, "Enhanced structured types", in a <general set function>, if MAX or MIN is specified, then the <value expression> shall not be of an ST-ordered declared type.

Add the following Conformance Rule:

10) Without Feature S024, "Enhanced structured types", if a <set qualifier> of DISTINCT is specified, then the <value expression> shall not be of ST-ordered declared type.

6.18 <string value function>

1. *Rationale: Provide more meaningful keywords for the <regular expression substring function>.*

In the Format, replace the production for <regular expression substring function> with:

```
<regular expression substring function> ::=
     SUBSTRING <left paren> <character value expression>
     SIMILAR <character value expression>
     ESCAPE <escape character> <right paren>
```

2. *Rationale: A <character value function> that operates on a <character value expression> of character large object should return character large object.*

Replace Syntax Rule 4) with:

4) If <character substring function> is specified, then:

a) Case:

i) If the declared type of <character value expression> is fixed-length character string or variable-length character string, then the declared type of the <character substring function> is variable-length character string with maximum length equal to the fixed length or maximum length of the <character value expression>.

ii) Otherwise, the declared type of the <character substring function> is large object character string with a maximum length equal to the maximum length of the <character value expression>.

a.1) The character repertoire and form-of-use of the <character substring function> are the same as the character repertoire and form-of-use of the <character value expression>.

b) The collating sequence and the coercibility characteristic are determined as specified for monadic operators in Subclause 4.2.3, "Rules determining collating sequence usage", where the first operand of <character substring function> plays the role of the monadic operand.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0859965 567

3. *Rationale: A <character value function> that operates on a <character value expression> of character large object should return character large object.*

Replace Syntax Rule 5) with:

5) If <regular expression substring function> is specified, then:

a) The declared types of the <escape character> and the <character value expression>s of the <regular expression substring function> shall be character string with the same character repertoire.

b) Case:

i) If the declared type of <character value expression> is fixed-length character string or variable-length character string, then the declared type of the <regular expression substring function> is variable-length character string with maximum length equal to the maximum length of the first <character value expression>.

ii) Otherwise, the declared type of the <regular expression substring function> is large object character string with a maximum length equal to the maximum length of the first <character value expression>.

b.1) The character repertoire and form-of-use of <regular expression substring function> are the same as the character repertoire and form-of-use of the first <character value expression>.

c) The value of the <escape character> shall have length 1 (one).

d) The collating sequence and the coercibility characteristic are determined as specified for monadic operators in Subclause 4.2.3, "Rules determining collating sequence usage", where the first operand of <regular expression substring function> plays the role of the monadic operand.

4. *Rationale: A <character value function> that operates on a <character value expression> of character large object should return character large object.*

Replace Syntax Rule 7) with:

7) If <form-of-use conversion> is specified, then:

a) <form-of-use conversion> shall be simply contained in a <value expression> that is immediately contained in a <derived column> that is immediately contained in a <select sublist> or shall immediately contain either a <simply value specification> that is a <host parameter name> or a <value specification> that is a <host parameter specification>.

b) A <form-of-use conversion name> shall identify a form-of-use conversion.

c) Case:

i) If the declared type of <character value expression> is fixed-length character string or variable-length character string, then the declared type of the result is variable-length character string with implementation-defined maximum length.

ii) Otherwise, the declared type of the result is large object character string with implementation-defined maximum length.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c.1) The character set of the result is the same as the character repertoire of the <character value expression> and form-of-use determined by the form-of-use conversion identified by the <form-of-use conversion name>. Let *CR* be that character repertoire. The result has the *Implicit* coercibility characteristic and its collating sequence is the default collating sequence of *CR*.

5. *Rationale: A <character value function> that operates on a <character value expression> of character large object should return character large object.*

Replace Syntax Rule 8) with:

8) If <character translation> is specified, then:

a) A <translation name> shall identify a character translation.

b) Case:

i) If the declared type of <character value expression> is fixed-length character string or variable-length character string, then the declared type of the <character translation> is variable-length character string with implementation-defined maximum length.

ii) Otherwise, the declared type of the <character translation> is large object character string with implementation-defined maximum length.

b.1) The declared type of the <character translation> has a character repertoire equal to the character repertoire of the target character set of the translation. Let *CR* be that character repertoire. The result has the *Implicit* coercibility characteristic and its collating sequence is the default collating sequence of *CR*.

6. *Rationale: A <character value function> that operates on a <character value expression> of character large object should return character large object.*

Replace Syntax Rule 9) with:

9) If <trim function> is specified, then;

a) If FROM is specified, then:

i) Either <trim specification> or <trim character> or both shall be specified.

ii) If <trim specification> is not specified, then BOTH is implicit.

iii) If <trim character> is not specified, then ' ' is implicit.

b) Otherwise, let *SRC* be <trim source>.

```
TRIM ( SRC )
```

is equivalent to

```
TRIM ( BOTH ' ' FROM SRC )
```

c) Case:

i) If the declared type of <character value expression> is fixed-length character string or variable-length character string, then the declared type of the <trim function> is variable-

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

length character string with maximum length equal to the fixed length or maximum length of the <trim source>.

ii) Otherwise, the declared type of the <trim function> is large object character string with maximum length equal to the maximum length of the <trim source>.

d) If a <trim character> is specified, then <trim character> and <trim source> shall be comparable.

e) The character repertoire and form-of-use of the <trim function> are the same as those of the <trim source>.

f) The collating sequence and the coercibility characteristic are determined as specified for monadic operators in Subclause 4.2.3, "Rules determining collating sequence usage", where the <trim source> plays the role of the monadic operand.

7. *Rationale: Clarify the algorithm for <regular expression substring function>.*

Replace General Rules 4)c), 4)d) and 4)e) with:

4) c) If the length in characters of *E* is not equal to 1 (one), then an exception condition is raised: *data exception — invalid escape character.*

d) If *R* does not contain exactly two occurrences of the two-character sequence consisting of *E* followed by the <double quote> character, then an exception condition is raised: *data exception — invalid use of escape character.*

e) Let *R1*, *R2*, and *R3* be the substrings of *R*, such that

```
'R' = 'R1' || 'E' || '"' || 'R2' || 'E' || '"' || 'R3'
```

is true.

f) If any one of *R1*, *R2* or *R3* is not a zero-length string and does not have the format of a <regular expression>, then an exception condition is raised: *data exception — invalid regular expression.*

g) If the predicate

```
'C' SIMILAR TO '(R1)' || '(R2)' || '(R3)' ESCAPE 'E'
```

is not true, then the result of the <regular expression substring function> is the null value.

h) Otherwise, the result of the <regular expression substring function> is computed as follows:

i) Let *S1* be the shortest initial substring of *C* such that there is a sub-string *S23* of *C* such that the following <search condition> is true:

```
'C' = 'S1' || 'S23' AND
'S1' SIMILAR TO 'R1' ESCAPE 'E' AND
'S23' SIMILAR TO '(R2)(R3)' ESCAPE 'E'
```

ii) Let *S3* be the shortest final substring of *S23* such that there is a sub-string *S2* of *S23* such that the following <search condition> is true:

```
'S23' = 'S2' || 'S3' AND
'S2' SIMILAR TO 'R2' ESCAPE 'E' AND
'S3' SIMILAR TO 'R3' ESCAPE 'E'
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

iii) The result of the <regular expression substring function> is *S2*.

8. Rationale: Editorial.

Replace Conformance Rule 3) with:

3) Without Feature T042, "Extended LOB data type support", conforming SQL language shall not contain any <blob value function>.

6.20 <interval value function>

1. Rationale: Editorial - typographical error.

Replace Syntax Rule 1) with:

1) If <interval absolute value function> is specified, then the declared type of the result is the declared type of the <interval value expression>.

Replace General Rule 1) with:

1) If <interval absolute value function> is specified, then let *N* be the value of the <interval value expression>.

Case:

a) If *N* is the null value, then the result is the null value.

b) If $N \geq 0$ (zero), then the result is *N*.

c) Otherwise, the result is -1***N*.

2. Rationale: Editorial.

Replace Conformance Rule 1) with:

1) Without Feature F052, "Intervals and datetime arithmetic", conforming SQL shall contain no <interval value function>.

6.22 <cast specification>

1. Rationale: Editorial.

Replace Syntax Rule 4) with:

4) Let *C* be some column and let *CO* be the <cast operand> of a <cast specification> *CS*. *C* is a leaf column of *CS* if *CO* consists of a single column reference that identifies *C* or of a single <cast specification> *CS1* of which *C* is a leaf column.

2. Rationale: Editorial.

Replace the introductory paragraph of Syntax Rule 6) with:

6) If the <cast operand> is a <value expression>, then the valid combinations of *TD* and *SD* in a <cast specification> are given by the following table. "Y" indicates that the combination is syntactically

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

valid without restriction; "M" indicates that the combination is valid subject to other Syntax Rules in this Subclause being satisfied; and "N" indicates that the combination is not valid:

3. *Rationale: Editorial.*

Replace Syntax Rule 13) c) with:

13) c) CAST (*VALUE* AS *ETD*)
where *VALUE* is a <value expression> of declared type *ESD*, shall be a valid <cast specification>.

4. *Rationale: Improve wording.*

Replace Syntax Rule 8) d) with:

8) d) If *SD* is a fixed-length bit string or variable-length bit string, then let *LSV* be the value of BIT_LENGTH(*SV*) and let *B* be the BIT_LENGTH of the character with the smallest BIT_LENGTH in the form-of-use of *TD*. Let *PAD* be the value of the remainder of the division *LSV* / *B*. Let *NC* be a character whose bits all have the value 0 (zero).

If *PAD* is not 0 (zero), then append (*B* - *PAD*) 0-valued bits to the least significant end of *SV*; a completion condition is raised: *warning — implicit zero-bit padding*.

Let *SVC* be the possibly padded value of *SV* regarded as a character string without regard to valid character encodings and let *LTDS* be a character string of *LTD* characters of value *NC* characters in the form-of-use of *TD*.

TV is the result of

```
SUBSTRING ( SVC || LTDS FROM 1 FOR LTD )
```

Case:

i) If the length of *TV* is less than the length of *SVC*, then a completion condition is raised: *warning — string data, right truncation*.

ii) If the length of *TV* is greater than the length of *SVC*, then a completion condition is raised: *warning — implicit zero-bit padding*.

Replace Syntax Rule 8) d) with:

9) d) If *SD* is a fixed-length bit string or variable-length bit string, then let *LSV* be the value of BIT_LENGTH(*SV*) and let *B* be the BIT_LENGTH of the character with the smallest BIT_LENGTH in the form-of-use of *TD*.

Let *PAD* be the value of the remainder of the division *LSV* / *B*.

If *PAD* is not 0 (zero), then append (*B* - *PAD*) 0-valued bits to the least significant end of *SV*; a completion condition is raised: *warning — implicit zero-bit padding*.

Let *SVC* be the possible padded value of *SV* regarded as a character string without regard to valid character encodings.

Case:

i) If CHARACTER_LENGTH(*SVC*) is not greater than *MLTD*, then *TV* is *SVC*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ii) Otherwise, *TV* is the result of

```
SUBSTRING ( SVC FROM 1 FOR MLTD )
```

If the length of *TV* is less than the length of *SVC*, then a completion condition is raised: *warning — string data, right truncation*.

Replace Syntax Rule 11) with:

11) If *TD* is fixed-length bit string, then let *LTD* be the length in bits of *TD*. Let *BLSV* be the result of BIT_LENGTH(*SV*).

Case:

a) If *BLSV* is equal to *LTD*, then *TV* is *SV* regarded as a bit string with a length in bits of *BLSV*.

b) If *BLSV* is larger than *LTD*, then *TV* is the first *LTD* bits of *SV* regarded as a bit string with a length in bits of *LTD*, and a completion condition is raised: *warning — string data, right truncation*.

c) If *BLSV* is smaller than *LTD*, then *TV* is *SV* regarded as a bit string extended on the right with *LTD* - *BLSV* bits whose values are all 0 (zero) and a completion condition is raised: *warning — implicit zero-bit padding*.

Replace Syntax Rule 12) with:

12) If *TD* is variable-length bit string, then let *MLTD* be the maximum length in bits of *TD*. Let *BLSV* be the result of BIT_LENGTH(*SV*).

Case:

a) If *BLSV* is less than or equal to *MLTD*, then *TV* is *SV* regarded as a bit string with a length in bits of *BLSV*.

b) If *BLSV* is larger than *MLTD*, then *TV* is the first *MLTD* bits of *SV* regarded as a bit string with a length in bits of *MLTD* and a completion condition is raised: *warning — string data, right truncation*.

5. Rationale: Editorial.

Replace General Rule 16) d) with:

16) d) If *SD* is TIMESTAMP WITH TIME ZONE, then the UTC component of *TV* is the hour, minute, and second <primary datetime field>s of *SV*, with implementation-defined rounding or truncation if necessary, and the time zone component of *TV* is the time zone displacement of *SV*.

6.23 <value expression>

1. Rationale: Correct the declared data type and value of <value expression primary>.

Replace Syntax Rule 2) with:

2) The declared type of a <value expression primary> is the declared type of the simply contained <unsigned value specification>, <column reference>, <set function specification>, <scalar subquery>, <case expression>, <value expression>, <cast specification>, <subtype treatment>, <attribute or

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

method reference>, <reference resolution>, <collection value constructor>, <field reference>, <element reference>, <method invocation>, or <static method invocation>, or the effective returns type of the immediately contained <routine invocation>, respectively.

2. *Rationale: Correct the declared data type and value of <value expression primary>.*

Replace General Rule 5) with:

5) The value of a <value expression primary> is the value of the simply contained <unsigned value specification>, <column reference>, <set function specification>, <scalar subquery>, <case expression>, <value expression>, <cast specification>, <subtype treatment>, <collection value constructor>, <field reference>, <element reference>, <method invocation>, <static method invocation>, <routine invocation>, or <attribute or method reference>.

6.24 <new specification>

1. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

In the Format, replace the production for <new invocation> with:

```
<new invocation> ::=
      <method invocation>
    | <routine invocation>
```

2. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace Syntax Rule 3) with:

3) Case:

a) If the <new specification> is of the form

```
NEW RN( )
```

then:

Case:

i) If *S* does not include the descriptor of a constructor method whose method name is equivalent to *MN* and whose unaugmented parameter list is empty, then the <new specification> is equivalent to the <new invocation>

```
RN( )
```

ii) Otherwise, the <new specification> is equivalent to the <new invocation>

```
RN( ).MN( )
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

b) Otherwise, the <new specification>

```
NEW RN( a1, a2, ..., an )
```

is equivalent to the <new invocation>

```
RN().MN( a1, a2, ..., an )
```

3. Rationale: Editorial.

Replace Conformance Rule 1) with:

1) Without Feature S023, "Basic structured types", conforming SQL language shall not contain any <new specification>.

6.25 <subtype treatment>

1. Rationale: Resolve incorrect duplication of BNF non-terminal symbol.

Replace the Format with:

```
<subtype treatment> ::=
     TREAT <left paren> <subtype operand>
       AS <target subtype> <right paren>

<subtype operand> ::= <value expression>

<target subtype> ::=
     <user-defined type>
```

2. Rationale: Editorial.

Replace Conformance Rule 1) with:

1) Without Feature S161, "Subtype treatment", conforming SQL Language shall contain no <subtype treatment>.

6.26 <numeric value expression>

1. Rationale: Editorial.

Replace General Rule 5) with:

5) If the most specific type of the result of an arithmetic operation is exact numeric, then

Case:

a) If the operator is not division and the mathematical result of the operation is not exactly representable with the precision and scale of the result data type, then an exception condition is raised: *data exception — numeric value out of range.*

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) If the operator is division and the approximate mathematical result of the operation represented with the precision and scale of the result data type loses one or more leading significant digits after rounding or truncating if necessary, then an exception condition is raised: *data exception — numeric value out of range*. The choice of whether to round or truncate is implementation-defined.

6.27 <string value expression>

1. Rationale: Include character large object in the description of the effects of <concatenation>.

Replace Syntax Rule 3) with:

3) Case:

a) If <concatenation> is specified, then:
Let *D1* be the declared type of the <character value expression> and let *D2* be the declared type of the <character factor>. Let *M* be the length in characters of *D1* plus the length in characters of *D2*. Let *VL* be the implementation-defined maximum length of a variable-length character string, let *LOL* be the implementation-defined maximum length of a large object character string, and let *FL* be the implementation-defined maximum length of a fixed-length character string.

Case:

i) If the declared type of the <character value expression> or <character value expression> or <character factor> is large object character string, then the declared type of the <concatenation> is large object character string with maximum length equal to the lesser of *M* and *LOL*.

ii) If the declared type of the <character value expression> or <character factor> is variable-length character string, then the declared type of the <concatenation> is variable-length character string with maximum length equal to the lesser of *M* and *VL*.

iii) If the declared type of the <character value expression> and <character factor> is fixed-length character string, then *M* shall not be greater than *FL* and the declared type of the <concatenation> is fixed-length character string with length *M*.

b) Otherwise, the declared type of the <character value expression> is the declared type of the <character factor>.

2. Rationale: Include character large object in the description of the effects of <concatenation>.

Replace General Rule 2) with:

2) If <concatenation> is specified, then let *S1* and *S2* be the result of the <character value expression> and <character factor>, respectively.

Case:

a) If either *S1* or *S2* is the null value, then the result of the <concatenation> is the null value.

b) Otherwise, let *S* be the string consisting of *S1* followed by *S2* and let *M* be the length of *S*.

Case:

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

i) If the most specific type of either *S1* or *S2* is large object character string, then let *LOL* be the implementation-defined maximum length of a large object character string.

Case:

1) If *M* is less than or equal to *LOL*, then the result of the <concatenation> is *S* with length *M*.

2) If *M* is greater than *LOL* and the right-most *M* characters of *S* are all the <space> character, then the result of the <concatenation> is the first *LOL* characters of *S* with length *LOL*.

3) Otherwise, an exception condition is raised: *data exception — string data, right truncation*.

ii) If the most specific type of either *S1* or *S2* is variable-length character string, then let *VL* be the implementation-defined maximum length of a variable-length character string.

Case:

1) If *M* is less than or equal to *VL*, then the result of the <concatenation> is *S* with length *M*.

2) If *M* is greater than *VL* and the right-most *M* characters of *S* are all the <space> character, then the result of the <concatenation> is the first *VL* characters of *S* with length *VL*.

3) Otherwise, an exception condition is raised: *data exception — string data, right truncation*.

iii) If the most specific types of both *S1* and *S2* are fixed-length character string, then the result of the <concatenation> is *S*.

6.30 <boolean value expression>

1. Rationale: Correct use of undefined non-terminal.

Replace Syntax Rule 1) and 2) with:

1) The declared type of a <nonparenthesized value expression primary> shall be boolean.

2) If NOT is specified in a <boolean test>, then let *BP* be the contained <boolean primary> and let *TV* be the contained <truth value>. The <boolean test> is equivalent to:

```
( NOT ( BP IS TV ) )
```

Replace Syntax Rule 3) b) with:

b) If *BVE* is a <parenthesized boolean value expression> and the immediately contained <boolean value expression> is a known-not-null condition for *X*, then *BVE* is a *known-not-null condition* for *X*.

b.1) If *BVE* is a <nonparenthesized value expression primary>, then *BVE* is not a known-not-null condition for *X*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.1 <row value constructor>

1. Rationale: Specify Syntax Rules for "contextually typed" terms.

Add the following Syntax Rules:

4.1) A <contextually typed row value constructor element> immediately contained in a <contextually typed row value constructor> shall not be a <value expression> of the form "<left paren> <value expression> <right paren>"
NOTE 93.1 — This Rule removes a syntactic ambiguity. A <contextually typed row value constructor> of this form is permitted, but is parsed in the form "<left paren> <contextually typed row value constructor list> <right paren>".

4.2) A <contextually typed row value constructor element> immediately contained in a <contextually typed row value constructor> shall not be a <value expression> that is a <row value expression>.
NOTE 93.2 — This Rule removes a syntactic ambiguity, since otherwise a <contextually typed row value constructor> could be a <row value expression>, and a <row value expression> could be a <contextually typed row value constructor>.

4.3) Let *CTRVC* be the <contextually typed row value constructor>. The declared type of *CTRVC* is a row type described by a sequence of (<field name>, <data type>) pairs, corresponding in order to each <contextually typed row value constructor element> *X* simply contained in *CTRVC*. The data type is the declared type of *X* and the <field name> is implementation-dependent and not equivalent to the <column name> name of any column or field, other than itself, of a table referenced by any <table reference> contained in the SQL-statement.

4.4) The degree of a <contextually typed row value constructor> is the degree of its declared type.

2. Rationale: Supply missing Conformance Rules for <contextually typed row value constructor>

Add the following Conformance Rules:

5) Without Feature F641, "Row and table constructors", a <contextually typed row value constructor> that is not simply contained in a <contextually typed table value constructor> shall not contain more than one <row value constructor element>.

6) Without Feature F641, "Row and table constructors", a <contextually typed row value constructor> shall not be a <row subquery>.

7.3 <table value constructor>

1. Rationale: Use the correct non-terminal symbols.

Replace Conformance Rule 1) with:

1) Without Feature F641, "Row and table constructors", the <contextually typed row value expression list> of a <contextually typed table value constructor> shall contain exactly one <contextually typed row value constructor> *RVE*. *RVE* shall be of the form "(<contextually typed row value constructor element list>)".

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

2. *Rationale: Supply missing Conformance Rules for <contextually typed table value constructor>*

Add the following Conformance Rules:

3) Without Feature F641, "Row and table constructors", the <contextually typed row value expression list> of a <contextually typed table value constructor> shall contain exactly one <contextually typed row value constructor> *RVE*. *RVE* shall be of the form "(<contextually typed row value constructor element list>)".

4) Without Feature F641, "Row and table constructors", conforming SQL language shall not contain any <contextually typed table value constructor>.

7.6 <table reference>

1. Rationale: Use correct containment.

Replace Syntax Rules 1) and 2) with:

1) If a <table reference> *TR* simply contains a <collection derived table> *CDT*, then let *C* be the <collection value expression> simply contained in *CDT*, let *CN* be the <correlation name> simply contained in *TR*, and let *TEMP* be an <identifier> that is not equivalent to *CN* nor to any other <identifier> contained in *TR*.

a) Case:

i) If *TR* specifies a <derived column list> *DCL*, then

Case:

1) If *CDT* specifies WITH ORDINALITY, then *DCL* shall contain 2 <column name>s. Let *N1* and *N2* be respectively the first and second of those <column name>s.

2) Otherwise, *DCL* shall contain 1 (one) <column name>; let *N1* be that <column name>. Let *N2* be a <column name> that is not equivalent to *N1*, *CN*, *TEMP*, or any other <identifier> contained in *TR*.

ii) Otherwise, let *N1* and *N2* be two <column name>s that are not equivalent to one another nor to *CN*, *TEMP*, or any other <identifier> contained in *TR*.

b) Let *RECQP* be:

```
WITH RECURSIVE TEMP ( N1, N2 ) AS
  ( SELECT C[1] AS N1, 1 AS N2
      FROM ( VALUES( 1 ) ) AS CN
     WHERE 0 < CARDINALITY( C )
    UNION
    SELECT C[N2+1] AS N1, N2+1 AS N2
      FROM TEMP
     WHERE N2 < CARDINALITY( C )
  )
```

c) Case:

i) If *TR* specifies a <derived column list> *DCL*, then let *PDCLP* be:

```
( DCL )
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ii) Otherwise, let *PDCLP* be a zero-length string.

d) Case:

i) If *CDT* specifies WITH ORDINALITY, then let *ELDT* be:

```
LATERAL ( RECQP SELECT * FROM TEMP AS CN PDCLP )
```

ii) Otherwise, let *ELDT* be:

```
LATERAL ( RECQP SELECT N1 FROM TEMP AS CN PDCLP )
```

e) *CDT* is equivalent to the <lateral derived table> *ELDT*.

2) A <correlation name> simply contained in a <table reference> *TR* is *exposed* by *TR*. A <table or query name> simply contained in a <table reference> *TR* is *exposed* by *TR* if and only if *TR* does not specify a <correlation name>.

2. *Rationale: The scope of an exposed table name or correlation name must include the <order by clause> of a simple table query. Use correct containment.*

Replace Syntax Rule 3) with:

3) Case:

a) If a <table reference> *TR* is contained in a <from clause> *FC* with no intervening <query expression>, then the *scope clause SC* of *TR* is the <select statement: single row> or innermost <query specification> that contains *FC*. The scope of the exposed <correlation name> or exposed <table or query name> of *TR* is the <select list>, <where clause>, <group by clause>, and <having clause> of *SC*, together with every <lateral derived table> that is simply contained in *FC* and is preceded by *TR*, and every <collection derived table> that is simply contained in *FC* and is preceded by *TR*, and the <join condition> of all <joined table>s contained in *SC* that contain *TR*. If *SC* is the <query specification> that is the <query expression body> of a simple table query *STQ*, then the scope of the exposed <correlation name> or exposed <table or query name> also includes the <order by clause> of *STQ*.

b) Otherwise, the *scope clause SC* of *TR* is the outermost <joined table> that contains *TR* with no intervening <query expression>. The scope of the exposed <correlation name> or exposed <table or query name> of *TR* is the <join condition> of *SC* and of all <joined table>s contained in *SC* that contain *TR*.

3. *Rationale: supply correct rules for equivalency of <table name>s, <query name>s and <correlation name>s.*

Replace Syntax Rules 4) and 5) with:

4) A <table or query name> that is a <table name> that is exposed by a <table reference> *TR* shall not be equivalent to any other <table or query name> that is a <table name> that is exposed by a <table reference> with the same scope clause as *TR*.

4.1) A <table or query name> that is a <query name> that is exposed by a <table reference> *TR* shall not be equivalent to the <qualified identifier> of any <table or query name> that is a <table name> that is exposed by a <table reference> with the same scope clause as *TR* , and shall not be equivalent to any other <table or query name> that is a <query name> that is exposed by a <table reference> with the same scope clause as *TR*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

5) A <correlation name> that is exposed by a <table reference> *TR* shall not be equivalent to any other <correlation name> that is exposed by a <table reference> with the same scope clause as *TR* and shall not be equivalent to the <qualified identifier> of any <table or query name> that is a <table name> that is exposed by a <table reference> with the same scope clause as *TR*, and shall not be equivalent to any <table or query name> that is a <query name> that is exposed by a <table reference> with the same scope clause as *TR*.

4. Rationale: Use correct containment.

Replace Syntax Rules 6) and 7) with:

6) A <table or query name> simply contained in a <table reference> *TR* has a scope clause and scope defined by that <table reference> if and only if the <table or query name> is exposed by *TR*.

7) If *TR* simply contains <only spec> *OS* and the table identified by the <table or query name> *TN* is not a typed table, then *OS* is equivalent to *TN*.

Replace Syntax Rule 11) with:

11) Case:

a) If no <derived column list> is specified, then the row type *RT* of the <table reference> is the row type of its simply contained <table or query name>, <derived table>, <lateral derived table>, or <joined table>.

b) Otherwise, the row type *RT* of the <table reference> is described by a sequence of (<field name>, <data type>) pairs, where the <field name> in the *i*-th pair is the *i*-th <column name> in the <derived column list> and the <data type> in the *i*-th pair is the declared type of the *i*-th column of the <derived table>, <joined table>, or <lateral derived table>, of the table identified by the <table or query name> simply contained in the <table reference>.

Replace Syntax Rule 12) and 13) with:

12) A <derived table> or <lateral derived table> is an *updatable derived table* if and only if the <query expression> simply contained in *TR* is updatable.

13) A <derived table> or <lateral derived table> is an *insertable-into derived table* if and only if the <query expression> simply contained in *TR* is insertable-into.

Replace Syntax Rule 15) c) with:

15) c) If *TR* immediately contains a <derived table> or <lateral derived table>, then every updatable column of the table identified by the <query expression> simply contained in <derived table> or <lateral derived table> is called an *updatable column* of *TR*.

Replace Syntax Rule 16) with:

16) If the <table or query name> simply contained in <table reference> is not a query name in scope, then let *T* be the table identified by the <table name> immediately contained in <table or query name>. If the <table reference> is not contained in a <schema definition>, then the schema identified by the explicit or implicit qualifier of the <table name> shall include the descriptor of *T*. If the <table reference> is contained in a <schema definition> *S*, then the schema identified by the explicit or implicit qualifier of the <table name> shall include the descriptor of *T*,or *S* shall contain a <schema element> that creates the descriptor of *T*.
NOTE 94 — "*query name in scope*" is defined in Subclause 7.12, "<query expression>".

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5. *Rationale: Use correct containment and correct the <only spec> and untyped tables.*

Replace Access Rule 1) with:

1) If <table reference> simply contains a <table or query name> that is a <table name>, then:

a) Let *T* be the table identified by the <table name> immediately contained in the <table or query name> simply contained in <table reference>.

b) If *T* is a base table or a viewed table and the <table reference> is contained in any of:

— A <query expression> simply contained in a <cursor specification>, a <view definition>, or an <insert statement>.

— A <table expression> or <select list> immediately contained in a <select statement: single row>.

— A <search condition> immediately contained in a <delete statement: searched> or an <update statement: searched>.

— A <value expression> simply contained in a <row value expression> immediately contained in a <set clause>.

then

Case:

i) If <table reference> is contained in an <SQL schema statement> then, the applicable privileges of the <authorization identifier> that owns the containing schema shall include SELECT on at least one column of *T*.

ii) Otherwise, the current privileges shall include SELECT on at least one column of *T*.

NOTE 95 — "applicable privileges" and "current privileges" are defined in Subclause 10.5, "<privileges>".

c) If the <table reference> is contained in a <query expression> simply contained in a <view definition> then the applicable privileges of the <authorization identifier> that owns the view shall include SELECT for at least one column of *T*.

d) If *TR* simply contains <only spec> and *TR* identifies a typed table, then

Case:

i) If <table reference> is contained in a <schema definition>, then the applicable privileges of the <authorization identifier> that owns the containing schema shall include SELECT WITH HIERARCHY OPTION on at least one supertable of *T*.

ii) Otherwise, the current privileges shall include SELECT WITH HIERARCHY OPTION on at least one supertable of *T*.

NOTE 96 — "applicable privileges" and *current privileges* are defined in Subclause 10.5, "<privileges>".

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

6. *Rationale: Use correct containment.*

Replace General Rule 1) with:

1) A <correlation name> or exposed <table or query name> simply contained in a <table reference> defines that <correlation name> or <table or query name> to be an identifier of the table identified by the <table or query name> or <derived table> or <lateral derived table> of that <table reference>.

7.7 <joined table>

1. *Rationale: NATURAL joins should be subject to the same syntactic restrictions as <named column join>s. Provide consistent Syntax Rules for comparison operations.*

Replace Syntax Rule 7) c) with:

7) c) Let C_1 and C_2 be a pair of corresponding join columns contained in T_1 and T_2, respectively. C_1 and C_2 shall be comparable, and the declared type of C_1 or C_2 shall not be LOB-ordered or array_ordered.

2. *Rationale: Named column joins implicitly perform <comparison predicate>s, and should be subject to the same Conformance Rules. Provide consistent Conformance Rules for comparison operations.*

Add the following Conformance Rule:

4.1) Without Feature S024, "Enhanced structured types", if NATURAL or <named column join> is specified, and if C is a corresponding join column, then the declared type of C shall not be an ST-ordered type.

7.8 <where clause>

1. *Rationale: Clarification.*

Replace Syntax Rule 2) with:

2) If a <value expression> directly contained in the <search condition> is a <set function specification>, then the <where clause> shall be contained in a <having clause> or <select list>, the <set function specification> shall contain a column reference, and every column reference contained in the <set function specification> shall be an outer reference .

7.9 <group by clause>

1. *Rationale: Disambiguate BNF non-terminal names; provide a unified treatment of CUBE and ROLLUP.*

In the Format, replace the production for <group by clause> with:

```
<group by clause> ::= GROUP BY <grouping element list>
```

Insert the following production:

```
<grouping element list> ::=
     <grouping element> [ {<comma> <grouping element>}...]
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

In the Format, replace the production for <grouping specification> with:

```
<grouping element> ::=
      <ordinary grouping set>
    | <rollup list>
    | <cube list>
    | <grouping sets specification>
    | <grand total>
```

In the Format, replace the production for <grouping sets list> with:

```
<grouping sets specification> ::=
    GROUPING SETS <left paren> <grouping set list> <right paren>
```

Replace the production for <grouping set> with:

```
<grouping set> ::=
      <ordinary grouping set>
    | <rollup list>
    | <cube list>
    | <grouping sets specification>
    | <grand total>
```

2. *Rationale: Extend the restrictions on the use of columns with types based on LOB and array types. Provide consistent Syntax Rules for comparison operations.*

Delete Syntax Rule 2).

3. *Rationale: Provide a correct, unified treatment of CUBE and ROLLUP.*

Replace Syntax Rule 3) with:

3) Let *QS* be the <query specification> that simply contains the <group by clause>, and let *SL*, *FC*, *WC*, *GBC* and *HC* be the <select list>, the <from clause>, the <where clause> if any, the <group by clause> and the <having clause> if any, respectively, that are simply contained in *QS*.

Delete Syntax Rule 4).

4. *Rationale: Extend the restrictions on the use of columns with types based on LOB and array types. Provide consistent Syntax Rules for comparison operations.*

Replace Syntax Rule 5) with:

5) The declared type of a grouping column shall not be LOB-ordered, array-ordered, UDT-EC-ordered, or UDT-NC-ordered.

5. *Rationale: Provide a correct, unified treatment of CUBE and ROLLUP*

Insert the following Syntax Rules:

6.1) A <grouping set list> shall not contain a <grouping sets specification>.

6.2) If a <group by clause> simply contains a <grouping sets specification> *GSS*, then *GSS* shall be the only <grouping element> simply contained in the <group by clause>.

6.3) Let *SL1* be obtained from *SL* by replacing every <asterisk> and <asterisked identifier chain> using the syntactic transformations in the Syntax Rules of Subclause 7.12 "<query specification>".

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

6.4) A <group by clause> is *primitive* if it does not contain a <rollup list>, <cube list>, <grouping sets specification>, or <grouping column reference list>, and does not contain both a <grouping column reference> and an <empty grouping set>.

6.5) A <group by clause> is *simple* if it does not contain a <rollup list>, <cube list> or <grouping sets specification>.

6.6) If *GBC* is a simple <group by clause> that is not primitive, then *GBC* is transformed into a primitive <group by clause> as follows:

a) Let *NSGB* be the number of <grouping column reference>s contained in *GBC*.

b) Case:

i) If *NSGB* is 0 (zero), then *GBC* is replaced by

```
GROUP BY ()
```

ii) Otherwise:

1) Let $SGCR_1$, ... $SGCR_{NSGB}$ be an enumeration of the <grouping column reference>s contained in *GBC*.

2) *GBC* is replaced by

GROUP BY $SGCR_1$, ... , $SGCR_{NSGB}$

NOTE 101.1 — that is, a simple <group by clause> that is not primitive may be transformed into a primitive <group by clause> by deleting all parentheses, and deleting extra <comma>s as necessary for correct syntax. If there are no grouping columns at all (for example, GROUP BY (), ()) this is transformed to the canonical form GROUP BY ().

6.7) If *GBC* is a primitive <group by clause>, then let *SLNEW* and *HCNEW* be obtained from *SL1* and *HC*, respectively, by replacing every <grouping operation> by the exact numeric literal 0 (zero). *QS* is equivalent to

```
SELECT SLNEW FC WC GBC HCNEW
```

Replace Syntax Rules 7), 8), 9), 10) and 11) with:

7) If *RL* is a <rollup list>, then let GCR_i range over the *n* <grouping column reference>s contained in *RL*. *RL* is equivalent to

```
GROUPING SETS (
( GCR1, GCR2 , ..., GCRn ),
( GCR1 , GCR2 , ..., GCRn-1 ),
( GCR1 , GCR2 , ..., GCRn-2 ),
...
( GCR1 ),
()
)
```

NOTE 102 — The result of the transform is to replace *RL* with a <grouping sets specification> that contains a <grouping set> for every initial sublist of the <grouping column reference list> of the <rollup list> , obtained by dropping elements from the right, one by one , and regarding <empty grouping set> as the shortest such initial sublist.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

8) If *CL* is a <cube list>, then let GCR_i range over the n <grouping column reference>s contained in *CL*. *CL* is transformed as follows:

a) Let $M = 2^n - 1$.

b) For each i between 0 (zero) and M:

i) Let BSL_i be the <bit string literal> containing n <bit>s whose binary value is i.

ii) For each j between 1 (one) and n, let $B_{i,j}$ be the j-th <bit>, counting from left to right, in BSL_i.

iii) For each j between 1 (one) and n, let $GSLCR_{i,j}$ be

Case:

1) If $B_{i,j}$ is 0 (zero), then the zero-length string.

2) If $B_{i,j}$ is 1 (one), and $B_{i,k}$ is 0 (zero) for all $k < j$, then GCR_j.

3) Otherwise, <comma> GCR_j.

iv) Let GSL_i be the <ordinary grouping set>

($GSLCR_{i,1}$ $GSLCR_{i,2}$ ... $GSLCR_{i,n}$)

c) *CL* is equivalent to:

GROUPING SETS (GSL_M , GSL_{M-1} , . . . GSL_0)

NOTE 103 — The result of the transform is to replace *CL* with a <grouping sets specification> that contains a <grouping set> for all possible subsets of the set of grouping columns in the <grouping column reference list> of the <cube list>, including <empty grouping set> as the empty subset with no grouping columns. For example, CUBE (A, B, C) is equivalent to:

```
GROUPING SETS (/* BSL_i */
( A, B, C),     /* 111 */
( A, B   ),         /* 110 */
( A,    C),     /* 101 */
( A      ),     /* 100 */
(    B, C),     /* 011 */
(    B   ),     /* 010 */
(       C),     /* 001 */
(        )      /* 000 */
)
```

9) If <grouping sets specification> *GSSA* simply contains another <grouping sets specification> *GSSB*, then *GSSA* is transformed as follows:

a) Let *NA* be the number of <grouping set>s simply contained in *GSSA*, and let *NB* be the number of <grouping set>s simply contained in *GSSB*.

b) Let GSA_i be an enumeration of the <grouping set>s simply contained in *GSSA*, for i between 1 (one) and *NA*.

c) Let GSB_i be an enumeration of the <grouping set>s simply contained in *GSSB*, for i between 1 (one) and *NB*.

d) Let k be the index such that $GSSB = GSA_k$.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

e) *GSSA* is equivalent to:

```
GROUPING SETS
  ( GSA1 , GSA2 , ... GSAk-1 ,
    GSB1 , ... , GSBNB,
    GSAk-1 , ... , GSANA
  )
```

NOTE 103.1 — Thus the nested <grouping sets specification> is removed by simply "promoting" each of its <grouping set>s to be a <grouping set> of the encompassing <grouping sets specification>.

10) If *CGB* is a <group by clause> that is not simple, then *CGB* is transformed as follows:

a) Previous Syntax Rules are applied repeatedly to eliminate any <grouping set specification> that is nested in another <grouping set specification>, as well as any <rollup list> and any <cube list>.

NOTE 103.2 — As a result, *CGB* is a list of two or more <grouping set>s, each of which is an <ordinary grouping set>, an <empty grouping set> or a <grouping sets specification> that contains only <ordinary grouping set>s and <empty grouping set>s. There are no remaining <rollup list>s, <cube list>s, or nested <grouping sets specification>s.

b) Any <grouping element> *GS* that is an <ordinary grouping set> or an <empty grouping set> is replaced by the <grouping sets specification>:

```
GROUPING SETS ( GS )
```

NOTE 103.3 — As a result, *CGB* s a list of two or more <grouping sets specification>s.

c) Let *GSSX* and *GSSY* be the first two <grouping sets specification>s in *CGB*. *CGB* is transformed by replacing *GSSX* <comma> *GSSY* as follows:

i) Let *NX* be the number of <grouping set>s in *GSSX* and let *NY* be the number of <grouping set>s in *GSSY*.

ii) Let GSX_i for *i* between 1 (one) and *NX* be the <grouping set>s contained in *GSSX*, and let GSY_i for *i* between 1 (one) and *NY* be the <grouping set>s contained in *GSSY*.

iii) Let *MX(i)* be the number of <grouping column reference>s in GSX_i, and let *MY(i)* be the number of <grouping column reference>s in GSY_i.

NOTE 103.4 — If GSX_i is <empty grouping set>, then *MX(i)* is 0 (zero); and similarly for *GSY i*.

iv) Let $GCRX_{i,j}$ for *j* between 1 (one) and *MX(i)* be the <grouping column reference>s contained in GSX_i, and let $GCRY_{i,j}$ for *j* between 1(one) and *MY(i)* be the <grouping column reference>s contained in GSY_i.

NOTE 103.5 — If GSX_i is <empty grouping set>, then there are no $GCRX_{i,j}$; and similarly for GSY_i.

v) For each *a* between 1 (one) and *NX* and each *b* between 1 (one) and *NY*, let $GST_{a,b}$ be

($GCRX_{a,1}$, ... , $GCRX_{a,MX(a)}$, $GCRY_{b,1}$, ... , $GCRY_{b,MY(b)}$)

that is, an <ordinary grouping set> consisting of $GCRA_{a,j}$ for all *j* between 1 (one) and *MX(a)* followed by $GCRY_{b,j}$ for all *j* between 1 (one) and *MY(b)*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

vi) *CGB* is transformed by replacing *GSSX* <comma> *GSSY* with:

```
GROUPING SETS
  ( GST1,1 , ... GST1,NY ,
    GST2,1 , ... GST2,NY ,
    . . .
    GSTNX,1 , ... GSTNX,NY
  )
```

NOTE 103.6 — Thus each <ordinary grouping set> in *GSSA* is "concatenated" with each <ordinary grouping set> in *GSSB*. For example,

```
GROUP BY GROUPING SETS ( ( A, B ), ( C ) ),
GROUPING SETS ( ( X, Y ), ( ) )
```

is transformed to.

```
GROUP BY GROUPING SETS (
  ( A, B, X, Y ),
  ( A, B ),
  ( C, X, Y ), ( C ) )
```

d) The previous subrule of this Syntax Rule is applied repeatedly until *CGB* consists of a single <grouping sets specification>.

11) If <grouping specification> consists of a single <grouping sets specification> *GSS* that contains only <ordinary grouping set>s or <grand total>, then:

a) Let *m* be the number of <grouping set>s contained in *GSS*.

b) Let G_{si}, $1 <= i <= m$, range over the <grouping set>s contained in *GSS*.

c) Let *p* be the number of distinct <column reference>s that are contained in *GSS*.

d) Let *PC* be an ordered list of these <column reference>s ordered according to their left-to-right occurrence in the list.

e) Let PC_k, $1 <= k <= p$, be the *k*-th <column reference> in *PC*.

f) Let $DTPC_k$ be the declared type of the column identified by PC_k.

g) Let $CNPC_k$ be the column name of the column identified by $_{Pck}$.

h) For each GS_i:

i) If GS_i is a <grand total>, then let *n(i)* be 0 (zero). If $GS_i$is a <grouping column reference>, then let *n(i)* be 1 (one). Otherwise, let *n(i)* be the number of <grouping column reference>s contained in the <grouping column reference list>.

ii) Let $GCR_{i,j}$, $1 <= j <= n(i)$, range over the <grouping column reference>s contained in GS_i.

iii) Case:

1) If GS_i is an <ordinary grouping set>, then

A) Transform *SL1* to obtain *SL2*, and transform *HC* to obtain *HC2*, as follows:

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

For every PC_k:

If there is no j such that $PC_k = GCR_{i,j}$, then make the following replacements in *SL1* and *HC*:

I) Replace each <derived column> in *SL1* that is a <column reference> that references PC_k by:

```
CAST ( NULL AS DTPC_k ) AS CNPC_k
```

II) Replace each <column reference> in *SL1* and *HC* that references PC_k and that is not an entire <derived column> by:

```
CAST ( NULL AS DTPC_k )
```

III) Replace each <grouping operation> in *SL1* and *HC* that contains a <column reference> that references PC_k by the <literal> 1 (one).

B) Transform *SL2* to obtain *SLNEW*, and transform *HC2* to obtain *HCNEW* by replacing each <grouping operation> that remains in *SL2* and *HC2* by the <literal> 0 (zero).

NOTE 103.7 — Thus the value of a <grouping operation> is 0 (zero) if the grouping column referenced by the <grouping operation> is among the $GCR_{i,j}$ and 1(one) if it is not.

C) Let $GSSQL_i$ be:

```
SELECT SLNEW FC WC
GROUP BY GCR_i,1 , ... , GCR_i,n(i)
HCNEW
```

2) If G_{si} is a <grand total>, then

A) Transform *SL1* to obtain *SLNEW*, and transform *HC* to obtain *HCNEW*, as follows:

For every k, $1 <= k <= p$:

I) Replace each <derived column> in *SL1* that is a <column reference> that references PC_k by:

```
CAST( NULL AS DTPC_k ) AS CNPC_k
```

II) Replace each <column reference> in *SL1* and *HC* that references PC_k and that is not an entire <derived column> by:

```
CAST ( NULL AS DTPC_k ) AS CNPC_k
```

III) Replace each <grouping operation> that contains a <column reference> that references PC_k by the <literal> 1 (one).

B) Let $GSSQL_i$ be:

```
SELECT SLNEW FC WC
GROUP BY ( )
HCNEW
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) *QS* is equivalent to:

GSSQL$_1$
UNION ALL
GSSQL$_2$
UNION ALL
.
.
.
UNION ALL
GSSQL$_m$

Delete Syntax Rule 12).

6. *Rationale: Adapting the Conformance Rules of <group by clause> to the Feature name and broader scope of Feature T431, "Extended grouping capabilities".*

Replace Conformance Rule 1) with:

1) Without Feature T431, "Extended grouping capabilities", conforming SQL language shall not specify ROLLUP, CUBE, GROUPING SETS, or <grand total>.

7. *Rationale: Provide consistent Conformance Rules for comparison operations.*

Replace Conformance Rule 3) with:

3) Without Feature S024, "Enhanced structured types", a grouping column shall not be of an ST-ordered declared type.

8. *Rationale: Adapting the Conformance Rules of <group by clause> to the Feature name and broader scope of Feature T431, "Extended grouping capabilities".*

Add the following Conformance Rule:

3.1) Without Feature T431, "Extended grouping capabilities", an <ordinary grouping set> shall be a <grouping column reference>.

7.10 <having clause>

1. *Rationale: Supply missing application of functional dependencies to <having clause>.*

Add the following Conformance Rules:

1) Without Feature T301, "Functional dependencies", each column reference directly contained in the <search condition> shall be one of the following:

   a) an unambiguous reference to a grouping column of *T*, or

   b) an outer reference.

2) Without Feature T301, "Functional dependencies", each column reference contained in a <subquery> in the <search condition> that references a column of *T* shall be one of the following:

   a) an unambiguous reference to a grouping column of *T*, or

   b) contained in a <set function specification>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.11 <query specification>

1. Rationale: Disallow DISTINCT on columns with types based on LOB and array types. Provide consistent Syntax Rules for comparison operations.

Replace Syntax Rule 5) with:

5) If a <set quantifier> DISTINCT is specified, then no column of *T* shall have a declared type that is LOB-ordered, array-ordered, UDT-EC-ordered, or UDT-NC-ordered.

2. Rationale: Clarify which <value expression>s are meant.

Replace Syntax Rule 10) with:

10) Each column reference directly contained in each <value expression> simply contained in the <select list> and each column reference contained in a <set function specification> directly contained in each <value expression> simply contained in the <select list> shall unambiguously reference a column of *T*.

3. Rationale: Additional circumstance for indeterminacy.

Replace Syntax Rule 11) d) ii) with:

11) d) ii) The functional dependency $G \twoheadrightarrow C$, where *G* is the set consisting of the grouping columns of *T*, holds in *T*.

4. Rationale: Clarify which <value expression>s are meant.

Replace Syntax Rule 13) with:

13) If *T* is a grouped table, then let *G* be the set consisting of every column referenced by a <column reference> contained in the <group by clause> immediately contained in <table expression>. In each <value expression> contained in the <select list>, each <column reference> that references a column of *T* shall reference some column *C* that is functionally dependent on *G* or shall be contained in a <set function specification>.

5. Rationale: Clarify which <value expression>s are meant.

Replace Conformance Rule 3) with:

3) Without Feature T301, "Functional dependencies", if *T* is a grouped table, then in each <value expression> contained in the <select list>, each <column reference> that references a column of *T* shall reference a grouping column or be specified in a <set function specification>.

6. Rationale: Provide consistent Conformance Rules for comparison operations.

Replace Conformance Rule 4) with:

4) Without Feature S024, "Enhanced structured types", if any column in the result of a <query specification> is of an ST-ordered declared type, then DISTINCT shall not be specified or implied.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.12 <query expression>

1. Rationale: Provide consistent Syntax Rules for comparison operations.

Replace Syntax Rule 11) c) with:

11) c) If the set operator is UNION DISTINCT, EXCEPT DISTINCT, EXCEPT ALL, INTERSECT DISTINCT or INTERSECT ALL, then the declared type of each column of *T1* and *T2* shall not be LOB-ordered, array-ordered, UDT-EC-ordered, or UDT-NC-ordered.

Delete Syntax Rules 21) and 22).

2. Rationale: Provide consistent Conformance Rules for comparison operations.

Replace Conformance Rule 8) with:

8) Without Feature S024, "Enhanced structured types", if any column in the result of a <query expression> is of ST-ordered declared type, then DISTINCT shall not be specified or implied, and neither INTERSECT nor EXCEPT shall be specified.

8.2 <comparison predicate>

1. Rationale: Provide consistent Syntax Rules for comparison operations.

Replace Syntax Rule 5) a) with:

5) a) If the declared type of X_i or Y_i is LOB-ordered, array-ordered, reference-ordered, or UDT-EC-ordered, then <comp op> shall be either <equals operator> or <not equals operator>.

2. Rationale: Comparison forms and categories do not have to be the same throughout a subtype family.

Replace NOTE 126 with:

NOTE 126 — The comparison form and comparison categories included in the user-defined type descriptors of both *UDT1* and *UDT2* are constrained to be the same, and the same as those of all their supertypes. If the comparison category is either STATE or RELATIVE, then the comparison functions of *UDT1* and *UDT2* are constrained to be equivalent; if the comparison category is MAP, they are not constrained to be equivalent.

3. Rationale: SR 5) b) i) 2) was incorrectly nested.

Delete Syntax Rule 5) b) i) 2).

Delete Syntax Rule 5) b) i) 3) and the accompanying NOTE.

Add Syntax Rule 5) b) i.1):

5) b) i.1) If the declared types of X_i and Y_i are reference types, then the referenced type of the declared type of X_i and the referenced type of the declared type of Y_i shall have a common supertype.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4. *Rationale: clarify that the choice of an ordering function is not determined by subject routine determination rules.*

Replace General Rule 1) b) iii) 1) with:

1) b) iii) 1) If the comparison category of UDT_x is MAP, then let *HF1* be the <routine name> with explicit <schema name> of the comparison function of UDT_x and let *HF2* be the <routine name> with explicit <schema name> of the comparison function of UDT_y. If *HF1* is an SQL-invoked function that is a method, then let *HFX* be *X.HF1*; otherwise, let *HFX* be *HF1* (*X*). If *HF2* is an SQL-invoked function that is a method, then let *HFY* be *Y.HF2*; otherwise, let *HFY* be *HF2* (*Y*).

X <comp op> *Y*

has the same result as

HFX <comp op> *HFY*

Replace General Rule 1) b) iii) 2) A) with:

1) b) iii) 2) A) Let *RF* be the <routine name> with explicit <schema name> of the comparison function of UDT_x.

5. *Rationale: Editorial - typographical error.*

Replace General Rule 4) with:

4) The comparison of two binary string values, *X* and *Y*, is determined by comparison of their octets with the same ordinal position. If X_i and Y_i are the values of the *i*-th octets of *X* and *Y*, respectively, and if L_x is the length in octets of *X* and L_y is the length in octets of *Y*, then *X* is equal to *Y* if and only if $L_x = L_y$ and if $X_i = Y_i$ for all *i*.

6. *Rationale: Provide consistent Conformance Rules for comparison operations.*

Replace Conformance Rule 1) with:

1) Without Feature T042, "Extended LOB data type support", no field of the declared row type of a <row value expression> that is simply contained in a <comparison predicate> shall be of a LOB-ordered declared type.

Replace Conformance Rule 2) with:

2) Without Feature S024, "Enhanced structured types", no field of the declared type of a <row value expression> that is simply contained in a <comparison predicate> shall be of a declared type that is ST-ordered.

8.3 <between predicate>

1. *Rationale: Provide consistent Conformance Rules for comparison operations.*

Replace Conformance Rule 2) with:

2) Without Feature S024, "Enhanced structured types", no field of the declared type of a <row value expression> that is simply contained in a <between predicate> shall be of a declared type that is ST-ordered.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.4 <in predicate>

1. Rationale: Resolve an ambiguous case, in which <in predicate value> might be interpreted either as a <table subquery> or as a <scalar subquery>, by choosing the former interpretation.

Add the following Syntax Rule:

0.1) If <in value list> consists of a single <row value expression>, it shall not be a <scalar subquery>.

2. Rationale: Provide consistent Conformance Rules for comparison operations.

Replace Conformance Rules 2), 3) and 4) with:

2) Without Feature T042, "Extended LOB data type support", no field of the declared row type of a <row value expression> or a <table subquery> contained in an <in predicate> shall be of a LOB-ordered declared type.

3) Without Feature S024, "Enhanced structured types", no field of the declared row type of a <row value expression> or a <table subquery> that is simply contained in an <in predicate> shall be of an ST-ordered declared type.

8.6 <similar predicate>

1. Rational: "P" is referenced in General Rules 5) a), 6) e), 6) g), and 6) i) but is not defined.

Replace General Rule 5) a) with:

5) a) If the enumeration is specified in the form "<character specifier> <minus sign> <character specifier>", then the set of all characters that collate greater than or equal to the character represented by the left <character specifier> and less than or equal to the character represented by the right <character specifier>, according to the collating sequence of the pattern *PCV*.

Replace General Rule 6) e) with:

6) e) L (<percent>)

is the set of all strings of any length (zero or more) from the character set of the pattern *PCV*.

Replace General Rule 6) g) with:

6) g) L (<underscore>)

is the set of all strings of length 1 (one) from the character set of the pattern *PCV*.

Replace General Rule 6) i) with:

6) i) L (<left bracket> <circumflex> <character enumeration> <right bracket>)

is the set of all strings of length 1 (one) with characters from the character set of the pattern *PCV* that are not contained in the set of characters in the <character enumeration>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

2. *Rationale: Provide consistent Conformance Rules for comparison operations.*

Add the following Conformance Rule:

2) Without Feature T042, "Extended LOB data type support", a <character value expression> contained in a <similar predicate> shall not be of declared type CHARACTER LARGE OBJECT.

8.8 <quantified comparison predicate>

1. *Rationale: Provide consistent Conformance Rules for comparison operations.*

Replace the Conformance Rules 1) and 2) with:

1) Without Feature T042, "Extended LOB data type support", no field of the declared row type of a <row value expression> or a <table subquery> contained in a <quantified comparison predicate> shall be of a LOB-ordered declared type.

2) Without Feature S024, "Enhanced structured types", no field of the declared row type of a <row value expression> shall be of an ST-ordered declared type.

8.10 <unique predicate>

1. *Rationale: Clarify that unordered types do not support <unique predicate>.*

In Syntax Rules, replace "None" with:

1) Each column of user-defined type in the result of the <table subquery> shall have a comparison type.

2. *Rationale: Provide consistent Syntax Rules for comparison operations.*

Add the following Syntax Rule:

2) The declared type of a column identified by a <column name> in the <unique column list> shall not be LOB-ordered, array-ordered, UDT-EC-ordered, or UDT-NC-ordered.

3. *Rationale: Use the notion of distinct rather than equality*

Replace General Rule 2) with:

2) If there are no two rows in *T* such that the value of each column in one row is non-null and is not distinct from the value of the corresponding column in the other row, then the result of the <unique predicate> is *true*; otherwise, the result of the <unique predicate> is *false*.

4. *Rationale: Provide consistent Conformance Rules for comparison operations.*

Replace Conformance Rule 2) with:

2) Without Feature S024, "Enhanced structured types", no column of the result of the <table subquery> shall be of ST-ordered declared type.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.11 <match predicate>

1. Rationale: Editorial - typographical error.

Replace Syntax Rule 1) with:

1) The row type of the <row value expression> and the row type of the <table subquery> shall be comparable.

2. Rationale: Provide consistent Conformance Rules for comparison operations.

Replace Conformance Rule 2) with:

2) Without Feature S024, "Enhanced structured types", no field of the declared row type of the <row value expression> shall be of an ST-ordered declared type and no column of the result of the <table subquery> shall be of an ST-ordered declared type.

3) Without Feature T042, "Extended LOB data type support", no field of the declared row type of the <row value expression> shall be of a LOB-ordered declared type.

8.12 <overlaps predicate>

1. Rationale: Create a separate Feature for <overlaps predicate>.

Replace Conformance Rule 1) with:

1) Without Feature F053, "OVERLAPS predicate", conforming SQL language shall not contain any <overlaps predicate>.

8.13 <distinct predicate>

1. Rationale: Use the notion of distinct.

Replace General Rules 1), 2) and 3) with:

1) The result of <distinct predicate> is *true* if the value of <row value expression 3> is distinct from the value of <row value expression 4>; otherwise the result is *false*.
NOTE 136.1 — "distinct" is defined in Subclause 3.1.5, "Definitions provided in Part 2".

2. Rationale: Provide consistent Conformance Rules for comparison operations.

Replace Conformance Rule 2) with:

2) Without Feature S024, "Enhanced structured types", no field of the declared row type of either <row value expression> shall be of an ST-ordered declared type.

3) Without Feature T042, "Extended LOB data type support", no field of the declared row type of either <row value expression> shall be of a LOB-ordered declared type.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.0 Determination of identical values

1. Rationale: Specify the meaning of "identical"
Source: BHX-049R3

Insert the following subclause:

9.0 Determination of identical values

Function

Determine whether two instances of values are identical, that is to say are occurrences of the same value.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *V1* and *V2* be two values specified in an application of this subclause.

NOTE 137.1 — This subclause is invoked implicitly wherever the word *identical* is used of two values.

2) Case:

a) If *V1* and *V2* are both null, then *V1* is identical to *V2*.

b) If *V1* is null and *V2* is not null or *V1* is not null and *V2* is null, then *V1* is not identical to *V2*.

c) If *V1* and *V2* are of comparable predefined types, then

Case:

i) If *V1* and *V2* are character strings then

Case:

1) If *V1* is not distinct from *V2* and CAST (*V1* AS BIT (BIT_LENGTH (*V1*)) is not distinct from CAST (*V2* AS BIT (BIT_LENGTH (*V2*)), then *V1* is identical to *V2*.

2) Otherwise, *V1* is not identical to *V2*.

ii) If *V1* and *V2* are time with time zone or timestamp with time zone and are not distinct and their time zone fields are not distinct, then *V1* is identical to *V2*.

iii) Otherwise, *V1* is identical to *V2* if and only if *V1* is not distinct from *V2*.

d) If *V1* and *V2* are of constructed types, then

Case:

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) If *V1* and *V2* are rows and their respective fields are identical, then *V1* is identical to *V2*.

ii) If *V1* and *V2* are arrays and have the same cardinality and elements in the same ordinal position in the two arrays are identical, then *V1* is identical to *V2*.

iii) If *V1* and *V2* are references and *V1* is not distinct from *V2*, then *V1* is identical to *V2*.

iv) Otherwise, *V1* is not identical to *V2*.

e) If *V1* and *V2* are of the same most specific type, *MST*, and *MST* is a user-defined type, then:

Case:

i) If *MST* is a distinct type, whose source type is *SDT* and the results of *SDT* (*V1*) and *SDT* (*V2*) are identical, then *V1* is identical to *V2*.

ii) If *MST* is a structured type and, for every observer function *O* defined for *MST*, the results of the invocations *O* (*V1*) and *O* (*V2*) are identical, then *V1* is identical to *V2*.

iii) Otherwise, *V1* is not identical to *V2*.

f) Otherwise, *V1* is not identical to *V2*.

Conformance Rules

None.

9.1 Retrieval assignment

1. Rationale: Restore the facility of assignments involving character strings if there is a user-defined cast function.

Replace Syntax Rule 3) b) with:

3) b) Otherwise, the declared type of *V* shall be a mutually assignable character string type or an appropriate user-defined cast function *UDCF* shall be available to assign *V* to *T*.

9.3 Data types of results of aggregations

1. Rationale: Editorial.

Replace Syntax Rule 3) i) with:

i) If any data type in *DTS* is a row type, then each data type in *DTS* shall be a row type with the same degree and the data type of each field in the same ordinal position of every row type shall be comparable. The result data type is a row defined by an ordered sequence of (<field name>, data type) pairs FD_i, where data type is the data type resulting from the application of this Subclause to the set of data types of fields in the same ordinal position as FD_i in every row type in *DTS* and <field name> is determined as follows:

Case:

i) If the names of fields in the same ordinal position as FD_i in every row type in *DTS* is *F*, then the <field name> in FD_i is *F*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ii) Otherwise, the <field name> in FD_i is implementation-dependent.

10.4 <routine invocation>

1. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

Replace Syntax Rule 4) with:

4) Case:

a) If *RI* is immediately contained in a <call statement>, then an SQL-invoked routine *R* is a *possibly candidate routine* for *RI* (henceforth, simply "possibly candidate routine") if *R* is an SQL-invoked procedure and the <qualified identifier> of the <routine name> of *R* is equivalent to the <qualified identifier> of *RN*.

b) If *RI* is immediately contained in a <method selection>, then an SQL-invoked routine *R* is a *possibly candidate routine* for *RI* if *R* is an instance SQL-invoked method and the <qualified identifier> of the <routine name> of *R* is equivalent to the <qualified identifier> of *RN*.

c) If *RI* is immediately contained in a <constructor method selection>, then an SQL-invoked routine *R* is a *possibly candidate routine* for *RI* if *R* is an SQL-invoked constructor method and the <qualified identifier> of the <routine name> of *R* is equivalent to the <qualified identifier> of *RN*.

d) If *RI* is immediately contained in a <static method selection>, then an SQL-invoked routine *R* is a possibly candidate routine for *RI* if *R* is a static SQL-invoked method and the <qualified identifier> of the <routine name> of *R* is equivalent to the <qualified identifier> of *RN* and *MSD* is included in a user-defined type descriptor for *UDTSM* or for some supertype of *UDTSM*.

e) Otherwise, an SQL-invoked routine *R* is a *possibly candidate routine* for *RI* if *R* is an SQL-invoked function that is not an SQL-invoked method and the <qualified identifier> of the <routine name> of *R* is equivalent to the <qualified identifier> of *RN*.

2. Rationale: Change "user-defined data type" to "user-defined type".

Replace Syntax Rule 6) b) iii) 2) A) with:

6) b) iii) 2) A) If the declared type of P_i is a user-defined type, then:

I) Let ST_i be the set of subtypes of the declared type of A_i.

II) The type designator of the declared type of P_i shall be in the type precedence list of the data type of some type in ST_i.

NOTE 149 — "type precedence list" is defined in Subclause 9.5, "Type precedence list determination".

3. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

Replace Syntax Rule 7) b) i) 1) A) with:

7) b) i) 1) A) If *RI* is immediately contained in a <method selection>, <static method selection> or a <constructor method selection>, then let *DP* be *TP*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4. *Rationale: Editorial.*

Replace Syntax Rule 7) b) i) 2) C) II) with:

7) b) i) 2) C) II) If the routine descriptor of *R1* includes a STATIC indication, then there is no other invocable routine *R2* for which the user-defined type described by the user-defined type descriptor that includes the routine descriptor of *R2* is a subtype of the user-defined type described by the user-defined type descriptor that includes the routine descriptor of *R1*.

5. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace Syntax Rule 8) b) i) 1) A) with:

8) b) i) 1) A) If *RI* is immediately contained in a <method selection>, <static method selection> or a <constructor method selection>, then let *DP* be *TP*.

6. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace Syntax Rule 9) with:

9) If *SR* is a constructor function, then *RI* shall be simply contained in a <new invocation>.

7. *Rationale: Change "user-defined data type" to "user-defined type".*

Replace General Rule 2) b) iii) 1) A) IV) with:

2) b) iii) 1) A) IV) The declared type of PM_j ,$2 \le j \le N$, is compatible with the declared type of PSR_j.
NOTE 152 — *SR* is an element of the set *SM*.

8. *Rationale: Clarify the semantics of SQL-data access indication.*

Replace General Rule 6) e) with:

6) e) Case:
i) If the SQL-data access indication of *CSC* specifies possibly contains SQL and *R* possibly reads SQL-data, or *R* possibly modifies SQL-data, then

1) If *R* is an external routine, then an exception condition is raised: *external routine exception — reading SQL-data not permitted.*

2) Otherwise, an exception condition is raised: *SQL routine exception — reading SQL-data not permitted.*

ii) If the SQL-data access indication of *CSC* specifies possibly reads SQL-data and *R* possibly modifies SQL-data, then:

1) If *R* is an external routine, then an exception condition is raised: *external routine exception — modifying SQL-data not permitted.*

2) Otherwise, an exception condition is raised: *SQL routine exception — modifying SQL-data not permitted.*

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Add the following General Rule:

6) e.1) Case:

i) If *R* does not possibly contain SQL, then set the SQL-data access indication in the routine execution context of *RSC* to does not possibly contain SQL.

ii) If *R* possibly contains SQL, then set the SQL-data access indication in the routine execution context of *RSC* to possibly contains SQL.

iii) If *R* possibly reads SQL-data, then set the SQL-data access indication in the routine execution context of *RSC* to possibly reads SQL-data.

iv) If *R* possibly modifies SQL-data, then set the SQL-data access indication in the routine execution context of *RSC* to possibly modifies SQL-data.

Delete General Rule 7).

9. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

Replace General Rule 8) a) with:

8) a) If *R* is an SQL-invoked method whose routine descriptor does not include a STATIC indication and if CPV_1 is the null value, then:

i) Let *RV* be the null value.

ii) If *R* is a mutator, then an exception condition is raised: *data exception — null instance used in mutator function.*

10. Rationale: Clarify the semantics of SQL-data access indication.

Replace General Rule 8) c) iv) with:

8) c) iv) If the SQL-data access indication of *RSC* specifies possibly contains SQL, and, before the completion of the execution of the <SQL routine body> of *R*, an attempt is made to execute an SQL-statement that possibly reads SQL-data, or an attempt is made to execute an SQL-statement that possibly modifies SQL-data, then an exception condition is raised: *SQL routine exception — reading SQL-data not permitted.*

11. Rationale: Clarify the semantics of SQL-data access indication.

Replace General Rule 8) c) v) with:

8) c) v) If the SQL-data access indication of *RSC* specifies possibly reads SQL-data, and, before the completion of the execution of the <SQL routine body> of *R*, an attempt is made to execute an SQL statement that possibly modifies SQL-data, then an exception condition is raised: *SQL routine exception — modifying SQL-data not permitted.*

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12. Rationale: Fix the subscripts in the references to CPV in General Rule 9) b) i) 1) C) and General Rule 9) b) i) 2) B).

Replace General Rule 9) b) i) 1) C) with:

9) b) i) 1) C) For *i* ranging from (*PN*+1)+1 to (*PN*+1)+*N*, the *i*-th entry in *ESPL* is the SQL indicator argument corresponding to $CPV_{i-(PN+1)}$.

13. Rationale: Fix the subscripts in the references to CPV in General Rule 9) b) i) 1) C) and General Rule 9) b) i) 2) B).

Replace General Rule 9) b) i) 2) B) with:

9) b) i) 2) B) For *i* ranging from *PN*+1 to *PN*+*N*, the *i*-th entry in *ESPL* is the SQL indicator argument corresponding to CPV_{i-PN}.

14. Rationale: Clarify the semantics of SQL-data access indication.

Add the following General Rule:

9) f) iii) 2.1) If the SQL-data access indication of *RSC* specifies does not possibly contain SQL, and before the completion of any execution of *P*, an attempt is made to execute an SQL-statement, then an exception condition is raised: *external routine exception — containing SQL not permitted.*

15. Rationale: Clarify the semantics of SQL-data access indication.

Delete General Rule 9) f) iii) 4).

16. Rationale: Clarify the semantics of SQL-data access indication.

Replace General Rule 9) f) iii) 5) with:

9) f) iii) 5) If the SQL-data access indication of *RSC* specifies possibly contains SQL, and, before the completion of any execution of *P*, an attempt is made to execute an SQL-statement that possibly reads SQL-data, or an attempt is made to execute an SQL-statement that possibly modifies SQL-data, then an exception condition is raised: *external routine exception — reading SQL-data not permitted.*

17. Rationale: Clarify the semantics of SQL-data access indication.

Replace General Rule 9) f) iii) 6) with:

9) f) iii) 6) If the SQL-data access indication of *RSC* specifies possibly reads SQL-data, and, before the completion of any execution of *P*, an attempt is made to execute an SQL-statement that possibly modifies SQL-data, then an exception condition is raised: *external routine exception — modifying SQL-data not permitted.*

18. Rationale: Check for the most specific type of the returned value from type-preserving functions.

Replace General Rule 10) with:

10) Case:

a) If *R* is an SQL-invoked function, then:

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

i) If *R* is type preserving, then:

1) Let *MAT* be the most specific type of the value of the argument substituted for the result SQL parameter of *R*.

2) If the most specific type of *RV* is not compatible with *MAT*, then an exception condition is raised: *data exception — most specific type mismatch.*

ii) Let *ERDT* be the effective returns data type of the <routine invocation>.

iii) Let the result of the <routine invocation> be the result of assigning *RV* to a target of declared type *ERDT* according to the rules of Subclause 9.2, "Store assignment".

b) Otherwise, for each SQL parameter P_i of *R* that is an output SQL parameter or both an input SQL parameter and an output SQL parameter, let TS_i be the <target specification> of the corresponding <SQL argument> A_i.

Case:

i) If TS_i is a <host parameter specification>, then CPV_i is assigned to TS_i according to the rules of Subclause 9.1, "Retrieval assignment".

ii) If TS_i is the <SQL parameter name> of an SQL parameter of an SQL-invoked routine, then CPV_i is assigned to TS_i according to the rules of Subclause 9.2, "Store assignment".

19. Rationale: Ensure that the complete result set is available for scrollable cursors

Replace General Rule 11) e) with:

11) e) Case:

i) If $FRCN_i$ is a scrollable cursor, then let NXT_i, $1 \le i \le RTN$, be 1 (one).

ii) Otherwise, let NXT_i, $1 \le i \le RTN$, be the ordinal number of the row of RS_i that would be retrieved if the following SQL-statement were executed:

```
FETCH NEXT FROM FRCN_i INTO . . .
```

10.5 <privileges>

1. Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".

Replace Conformance Rule 9) with:

9) Without Feature F461, "Named character sets", in conforming SQL language, an <object name> shall not specify CHARACTER SET.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.6 <character set specification>

1. *Rationale: Align Syntax Rules with the Concepts.*

   3) The <standard character set name>s shall include: SQL_CHARACTER and all those specified in Subclause 4.2.4, "Named character sets"as defined by other standards.

2. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".*

Replace Conformance Rule 1) with:

   1) Without Feature F461, "Named character sets", conforming SQL language shall not contain a <character set specification>.

10.7 <specific routine designator>

1. *Rationale: Use the correct BNF (<user-defined type name> instead of <user-defined type>).*

In the Format, replace the production for <specific routine designator> with:

```
<specific routine designator> ::=
      SPECIFIC <routine type> <specific name>
    | <routine type> <member name> [ FOR <user-defined type name> ]
```

2. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

In the Format, replace the production for <routine type> with:

```
<routine type> ::=
      ROUTINE
    | FUNCTION
    | PROCEDURE
    | [ INSTANCE | STATIC | CONSTRUCTOR ] METHOD
```

3. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace Syntax Rule 2) with:

   2) If <routine type> specifies METHOD and neither INSTANCE nor STATIC nor CONSTRUCTOR is specified, then INSTANCE is implicit.

4. *Rationale: use the correct BNF (<user-defined type name> instead of <user-defined type>) and correct namespace problems associated with methods used to initialize newly-constructed structured type values. Replace incorrectly used concept of data type identity of routine parameters with the concept of data type compatibility.*

Replace Syntax Rule 3) with:

   3) If a <member name> *MN* is specified, then:

      a) If <user-defined type name> is specified, then <routine type> shall specify METHOD. If METHOD is specified, then <user-defined type name> shall be specified.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) Let *RN* be the <schema qualified routine name> of *MN* and let *SCN* be the <schema name> of *MN*.

c) Case:

i) If *MN* contains a <data type list>, then:

1) If <routine type> specifies FUNCTION, then there shall be exactly one SQL-invoked function that is not an SQL-invoked method in the schema identified by *SN* whose <schema qualified routine name> is *RN* such that for all *i* the declared type of its *i*-th SQL parameter is compatible with the *i*-th <data type> in the <data type list> of *MN*. The <specific routine designator> identifies that SQL-invoked function.

2) If <routine type> specifies PROCEDURE, then there shall be exactly one SQL-invoked procedure in the schema identified by *SN* whose <schema qualified routine name> is *RN* such that for all *i* the declared type of its *i*-th SQL parameter is compatible with the *i*-th <data type> in the <data type list> of *MN*. The <specific routine designator> identifies that SQL-invoked function.

3) If <routine type> specifies METHOD, then

Case:

A) If STATIC is specified, then there shall be exactly one static SQL-invoked method of the type identified by <user-defined type name> such that for all *i*, the declared data type of its *i*-th SQL parameter is compatible with the *i*-th <data type> in the <data type list> of *MN*. The <specific routine designator> identifies that static SQL-invoked method.

B) If CONSTRUCTOR is specified, then there shall be exactly one SQL-invoked constructor method of the type identified by <user-defined type name> such that for all *i*, the declared data type of its *i*-th SQL parameter in the unaugmented <SQL parameter declaration list> is compatible with the *i*-th <data type> in the <data type list> of *MN*. The <specific routine designator> identifies that SQL-invoked constructor method.

C) Otherwise, there shall be exactly one instance SQL-invoked method of the type identified by <user-defined type name> such that for all *i*, the declared data type of its *i*-th SQL parameter in the unaugmented <SQL parameter declaration list> is compatible with the *i*-th <data type> in the <data type list> of *MN*. The <specific routine designator> identifies that SQL-invoked method.

4) If <routine type> specifies ROUTINE, then there shall be exactly one SQL-invoked routine in the schema identified by *SN* whose <schema qualified routine name> is *RN* such that for all *i* the declared type of its *i*-th SQL parameter is compatible with the *i*-th <data type> in the <data type list> of *MN*. The <specific routine designator> identifies that SQL-invoked routine.

ii) Otherwise:

1) If <routine type> specifies FUNCTION, then there shall be exactly one SQL-invoked function in the schema identified by *SN* whose <schema qualified routine name> is *RN*. The <specific routine designator> identifies that SQL-invoked function.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) If <routine type> specifies PROCEDURE, then there shall be exactly one SQL-invoked procedure in the schema identified by *SN* whose <schema qualified routine name> is *RN*. The <specific routine designator> identifies that SQL-invoked procedure.

3) If <routine type> specifies METHOD, then

Case:

A) If STATIC is specified, then there shall be exactly one static SQL-invoked method of the user-defined type identified by <user-defined type name>. The <specific routine designator> identifies that static SQL-invoked method.

B) If CONSTRUCTOR is specified, then there shall be exactly one SQL-invoked constructor method of the user-defined type identified by <user-defined type name>. The <specific routine designator> identifies that SQL-invoked constructor method.

C) Otherwise, there shall be exactly one instance SQL-invoked method of the user-defined type identified by <user-defined type name> that is not a static SQL-invoked method. The <specific routine designator> identifies that SQL-invoked method.

4) If <routine type> specifies ROUTINE, then there shall be exactly one SQL-invoked routine in the schema identified by *SN* whose <schema qualified routine name> is *RN*. The <specific routine designator> identifies that SQL-invoked routine.

5. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace Syntax Rule 4) with:

4) If FUNCTION is specified, then the SQL-invoked routine that is identified shall be an SQL-invoked function that is not an SQL-invoked method. If PROCEDURE is specified, then the SQL-invoked routine that is identified shall be an SQL-invoked procedure. If STATIC METHOD is specified, then the SQL-invoked routine that is identified shall be a static SQL-invoked method. If CONSTRUCTOR METHOD is specified, then the SQL-invoked routine shall be an SQL-invoked constructor method. If INSTANCE METHOD is specified or implicit, then the SQL-invoked routine shall be an instance SQL-invoked method. If ROUTINE is specified, then the SQL-invoked routine that is identified is either an SQL-invoked function or an SQL-invoked procedure.

10.12 Execution of triggers

1. Rationale: Editorial.

Replace General Rule 4) b) with:

4) b) If the execution of *TSS* is not successful, then an exception condition is raised: *triggered action exception*. The exception information associated with *TSS* is entered into the diagnostics area in a location other than the location corresponding to condition number 1 (one).

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.13 Execution of array-returning functions

1. Rationale: Editorial.

Replace General Rule 6) with:

6) If the call type data item has a value of -1 (indicating "*open call*"), then *P* is executed with a list of *EN* parameters PD_i whose parameter names are PN_i and whose values are set as follows:

a) Depending on whether the language of *R* specifies ADA, C, COBOL, FORTRAN, MUMPS, PASCAL, or PLI, let the *operative data type correspondences* table be Table 18, "Data type correspondences for Ada", Table 19, "Data type correspondences for C", Table 20, "Data type correspondences for COBOL", Table 21, "Data type correspondences for Fortran", Table 22, "Data type correspondences for MUMPS", Table 23, "Data type correspondences for Pascal", or Table 24, "Data type correspondences for PL/I", respectively. Refer to the two columns of the operative data type correspondences table as the "SQL data type" column and the "host data type" column.

b) For *i* varying from 1 (one) to *EN*, the <data type> DT_i of PD_i is the data type listed in the host data type column of the row in the data type correspondences table whose value in the SQL data type column corresponds to the data type of ESP_i.

c) The value of PD_i is set to the value of ESP_i.

Replace General Rule 8) a) with:

8) a) *P* is executed with a list of *EN* parameters PD_i whose parameter names are PN_i and whose values are set as follows:

i) Depending on whether the language of *R* specifies ADA, C, COBOL, FORTRAN, MUMPS, PASCAL, or PLI, let the *operative data type correspondences* table be Table 18, "Data type correspondences for Ada", Table 19, "Data type correspondences for C", Table 20, "Data type correspondences for COBOL", Table 21, "Data type correspondences for Fortran", Table 22, "Data type correspondences for MUMPS", Table 23, "Data type correspondences for Pascal", or Table 24, "Data type correspondences for PL/I", respectively. Refer to the two columns of the operative data type correspondences table as the "SQL data type" column and the "host data type" column.

ii) For *i* varying from 1 (one) to *EN*, the <data type> DT_i of PD_i is the data type listed in the host data type column of the row in the data type correspondences table whose value in the SQL data type column corresponds to the data type of ESP_i.

c) The value of PD_i is set to the value of ESP_i.

Replace General Rule 8) b) iii) with:

8) b) iii) Otherwise, set the value of the call type data item to 1 (one) (indicating *close call*).

Replace General Rule 9) with:

9) If the call type data item has a value of 1 (indicating "*close call*"), then *P* is executed with a list of *EN* parameters PD_i whose parameter names are PN_i and whose values are set as follows:

a) Depending on whether the language of *R* specifies ADA, C, COBOL, FORTRAN, MUMPS, PASCAL, or PLI, let the *operative data type correspondences* table be Table 18, "Data type

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

correspondences for Ada", Table 19, "Data type correspondences for C", Table 20, "Data type correspondences for COBOL", Table 21, "Data type correspondences for Fortran", Table 22, "Data type correspondences for MUMPS", Table 23, "Data type correspondences for Pascal", or Table 24, "Data type correspondences for PL/I", respectively. Refer to the two columns of the operative data type correspondences table as the "SQL data type" column and the "host data type" column.

b) For *i* varying from 1 (one) to *EN*, the <data type> DT_i of PD_i is the data type listed in the host data type column of the row in the data type correspondences table whose value in the SQL data type column corresponds to the data type of ESP_i.

c) The value of PD_i is set to the value of ESP_i.

11.1 <schema definition>

1. Rationale: Remove redundant item. <grant role statement> is defined in <grant statement>.

In the Format, replace the production for <schema element> with:

```
<schema element> ::=
      <table definition>
    | <view definition>
    | <domain definition>
    | <character set definition>
    | <collation definition>
    | <translation definition>
    | <assertion definition>
    | <trigger definition>
    | <user-defined type definition>
    | <schema routine>
    | <grant statement>
    | <role definition>
    | <user-defined cast definition>
    | <user-defined ordering definition>
    | <transform definition>
```

2. Rationale: Correct the references to schema which do not treat it as a descriptor.

Replace General Rules 1) with:

1) A <schema definition> creates an SQL-schema *S* in a catalog. *S* includes:

   a) A schema name that is equivalent to the explicit or implicit <schema name>.

   b) A schema authorization identifier that is equivalent to the explicit or implicit <authorization identifier>.

   c) A schema character set name that is equivalent to the explicit or implicit <schema character set specification>.

   d) A schema SQL-path that is equivalent to the explicit or implicit <schema path specification>.

   e) The descriptor created by every <schema element> of the <schema definition>.

Delete General Rule 4).

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3. *Rationale: Remove redundant item. <grant role statement> is defined in <grant statement>.*

Delete Conformance Rule 2).

4. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".*

Replace Conformance Rules 8) and 9) with:

8) Without Feature F461, "Named character sets", a <schema character set specification> shall not be specified.

9) Without Feature F451, "Character set definition", conforming SQL language shall not contain any <character set definition>.

11.2 <drop schema statement>

1. *Rationale: Correct the references to schema which do not treat it as a descriptor.*

Replace General Rule 13) with:

13) *S* is destroyed.

11.3 <table definition>

1. *Rationale: Incorrect symbol in syntax substitute for <like clause>.*

Replace Syntax Rule 6) b) with:

6) b) Let *nt* be the number of columns in *T1*. Let C_i, 1 (one) $\leq i \leq nt$, be the columns of *T1*, in the order in which they appear in *T1*, let CN_i be the column name included in the column descriptor of C_i, and let DT_i be the data type included in the column descriptor of C_i. Let CD_1 be:

CN_1 DT_1

If *nt* is greater than 1 (one), then let CD_i, $2 \leq i \leq nt$, be:

, CN_i DT_i

The <like clause> is effectively replaced by CD_i, 1 (one) $\leq i \leq nt$.
NOTE 169 – <column constraint>s are not included in columns; <column constraint>s are effectively transformed to <table constraints>s and are thereby excluded.

2. *Rationale: Incomplete rule—should prohibit <column definition>s as well as <like clause>s.*

Replace Syntax Rule 10) b) with:

10) b) *TEL1* shall not contain a <like clause> or a <column definition>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3. *Rationale: Align the rules with the BNF in the Format,*

Replace Syntax Rule 16) with:

16) If *TEL1* contains a <column options>, then *TD* shall specify OF <user-defined type> and any <column definition> shall be contained before the first <column options>.

4. *Rationale: Align the rules with the BNF in the Format.*

Replace Access Rule 4) with:

4) If "OF <user-defined type>" is specified, then the applicable privileges of *A* shall include USAGE on *ST*.

5. *Rationale: Editorial - typographical error.*

Replace General Rule 3) with:

3) For each <column options> CO, if CO contains a <scope clause> SC, then let CD be the column descriptor identified by the <column name> specified in CO. The <table name> specified in SC is included in the reference type descriptor that is included in CD.

6. *Rationale: Clean up typed table insertability property for non-instantiable types.*

Add the following General Rule:

5) 1.1) Case:

i) If OF <user-defined type> is not specified, then an indication that *T* is insertable-into.

ii) Otherwise,

Case:

1) If the data type descriptor of *R* indicates that *R* is instantiable, then an indication that *T* is insertable-into.

2) Otherwise, an indication that *T* is not insertable-into.

7. *Rationale: Correct privileges for <temporary tables>s.*

Replace General Rule 8) with:

8) A set of privilege descriptors is created that define the privileges INSERT, SELECT, UPDATE, DELETE, TRIGGER, and REFERENCES on this table and SELECT, INSERT, UPDATE, and REFERENCES for every <column definition> in the table definition. If OF <user-defined type> is specified, then a table/method privilege descriptor is created on this table for every method of the structured type identified by the <user-defined type> and the table SELECT privilege has the WITH HIERARCHY OPTION. These privileges are grantable. The grantor for each of these privilege descriptors is set to the special grantor value "_SYSTEM". The grantee is <authorization identifier> *A*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

8. *Rationale: Editorial.*

Replace General Rule 10) with:

10) The row type *RT* of the table *T* defined by the <table definition> is the set of pairs (<field name>, <data type>) where <field name> is the name of a column *C* of *T* and <data type> is the declared type of *C*. This set of pairs contains one pair for each column of *T*, in the order of their ordinal position in *T*.

9. *Rationale: Permit tables of structured type without requiring Feature S043, "Enhanced reference types".*

Replace Conformance Rule 5) with:

5) Without Feature S043, "Enhanced reference types", a <self-referencing column specification> shall specify SYSTEM GENERATED.

11.4 <column definition>

1. *Rationale: Provide consistent Syntax Rules for comparison operations.*

Add the following Syntax Rule:

9.1) If <data type> is a <reference type> that identifies a reference type that is LOB-ordered, array-ordered, UDT-EC-ordered or UDT-NC-ordered, then REFERENCES ARE CHECKED shall not be specified.

2. *Rationale: Provide consistent Conformance Rules for comparison operations.*

Add the following Conformance Rule:

7) Without Feature S024, "Enhanced structured types", if <data type> is a <reference type> that identifies a reference type that is ST-ordered, then REFERENCES ARE CHECKED shall not be specified.

11.7 <unique constraint definition>

1. *Rationale: Provide consistent Syntax Rules for comparison operations.*

Replace Syntax Rule 1) with:

1) The declared type of no column identified by any <column name> in the <unique column list> shall be LOB-ordered, array-ordered, UDT-EC-ordered, or UDT-NC-ordered.

2. *Rationale: Provide consistent Conformance Rules for comparison operations.*

Add the following Conformance Rule:

3) Without Feature S024, "Enhanced structured types", the declared type of no column identified by any <column name> in the <unique column list> shall be of ST-ordered declared type.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.8 <referential constraint definition>

1. Rationale: Add a missing prefix.

Replace Syntax Rule 3) b) ii) with:

3) b) ii) For a given row in the referenced table, every matching row for that given row that is a matching row only to the given row in the referenced table for the referential constraint is a *unique matching row*. For a given row in the referenced table, a matching row for that given row that is not a unique matching row for that given row for the referential constraint is a *non-unique matching row*.

2. Rationale: Provide consistent Syntax Rules for comparison operations.

Add the following Syntax Rule:

8.1) No referencing column shall have a declared type that is LOB-ordered, array-ordered, UDT-EC-ordered, or UDT-NC-ordered.

3. Rationale: Provide consistent Conformance Rules for comparison operations.

Add the following Conformance Rule:

6) Without Feature S024, "Enhanced structured types", no referencing column shall be of ST-ordered declared type.

11.17 <drop column definition>

1. Rationale: Remove unneeded misleading text.

Delete NOTE 193

Delete General Rule 2)

11.21 <view definition>

1. Rationale: Replace incorrect non-terminal.

Replace Syntax Rule 21) e) with:

21) e) If <subview clause> is not specified, then <self-referencing column specification> shall be specified.

Replace Syntax Rule 21) i) with:

21) i) If <self-referencing column specification> is specified, then:

i) <subview clause> shall not be specified.

ii) SYSTEM GENERATED shall not be specified.

iii) Let *RST* be the reference type REF(*ST*).

Case:

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

1) If USER GENERATED is specified, then:

A) *RST* shall have a user-defined representation.

B) Let *m* be 1 (one).

2) If DERIVED is specified, then:

A) *RST* shall have a derived representation.

B) Let *m* be 0 (zero).

Replace Syntax Rule 21) s) with:

21) s) If <self-referencing column specification> is specified, then

Case:

i) If *RST* has a user-defined representation, then:

1) *TQN* shall have a candidate key consisting of a single column *RC*.

2) Let *SS* be the first <select sublist> in the <select list> of *QS*.

3) *SS* shall consist of a single <cast specification> *CS* whose leaf column is *RC*.

NOTE 202 — "Leaf column of a <cast specification>" is defined in Subclause 6.22, "<cast specification>".

4) The declared type of F_1 shall be REF(*ST*).

ii) Otherwise, *RST* has a derived representation.

1) Let C_i, 1 (one) $\leq i \leq n$, be the columns of *V* that correspond to the attributes of the derived representation of *RST*.

2) *TQN* shall have a candidate key consisting of some subset of the underlying columns of C_i, 1 (one) $\leq i \leq n$.

11.30 <character set definition>

1. Rationale: Replace undefined non-terminals .

Replace Syntax Rule 3) with:

3) The character set identified by the <character set specification> contained in <character set source> shall have associated with it a privilege descriptor that was effectively defined by the <grant statement>

```
GRANT USAGE ON CHARACTER SET CS TO PUBLIC
```

where *CS* is the <character set name>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2. *Rationale: Delete unnecessary and incorrect rule .*

Delete General Rule 4).

3. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets" .*

Replace Conformance Rules 1) and 2) with:

1) Without Feature F451, "Character set definition", conforming SQL language shall not specify any <character set definition>.

2) Without Feature F451, "Character set definition", and Feature F691, "Collation and translation", <collation source> shall specify DEFAULT.

11.31 <drop character set statement>

1. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets" .*

Replace Conformance Rule 1) with:

1) Without Feature F451, "Character set definition", conforming SQL language shall contain no <drop character set statement>.

11.38 <trigger definition>

1. *Rationale: Clarify that the subject table of a <trigger definition> cannot be a declared local temporary table.*

Replace Syntax Rule 5) with:

5) *T* shall be a base table that is not a declared local temporary table.

2. *Rationale: Provide missing Access Rule.*

Add the following Access Rule:

3) If an <old or new values alias> has been specified, then the applicable privileges for *A* shall include SELECT on *T*.

3. *Rationale: Editorial.*

Replace Conformance Rule 1) with:

1) Without Feature T211, "Basic trigger capability", conforming SQL language shall not contain a <trigger definition>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.40 <user-defined type definition>

1. *Rationale: Correct the specification of implicit generation of cast functions when reference values are generated via <user-defined representation>.*

In the Format, replace the production of <user-defined type body> with:

```
<user-defined type body> ::=
     <user-defined type name>
     [ <subtype clause> ]
     [ AS <representation> ]
     [ <instantiable clause> ]
     <finality>
     [ <reference type specification> ]
     [ <ref cast option> ]
     [ <cast option> ]
     [ <method specification list> ]
```

In the Format, replace the production of <user-defined representation> with:

```
<user-defined representation> ::= REF USING <predefined type>
```

2. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values. Correct use of <specific name> for method specifications by replacing with <specific method name>.*

In the Format, replace the production for <partial method specification> with:

```
<partial method specification> ::=
     [ INSTANCE | STATIC | CONSTRUCTOR ] METHOD <method name>
     <SQL parameter declaration list>
     <returns clause>
     [ SPECIFIC <specific method name> ]
```

3. *Rationale: <method characteristic> should not include <transform group specification>.*

In the Format, replace the production for <method characteristic> with:

```
<method characteristic> ::=
       <language clause>
     | <parameter style clause>
     | <deterministic characteristic>
     | <SQL-data access indication>
     | <null-call clause>
```

4. *Rationale: Correct use of <specific name> for method specifications by replacing with <specific method name>.*

In the Format, insert the following production:

```
<specific method name> ::=
     [ <schema name> <period> ] <qualified identifier>
```

5. *Rationale: Remove unneeded rule during process of correcting the oversight in recursive type definition. Move check to <attribute definition>.*

Delete Syntax Rule 6) g) and associated NOTE 225.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6. *Rationale: Correct the specification of implicit generation of cast functions when reference values are generated via <user-defined representation>.*

Replace Syntax Rule 6) i) i) with:

6) i) i) Let *BT* be <predefined type>. *BT* is the *representation type of the referencing type* of *UDT*.

Add the following Syntax Rule:

6) k) iii.1) If the user-defined type descriptor of *SST* indicates that the referencing type of *SST* has a user-defined representation, then let *BT* be the data type described by the data type descriptor of the representation type of the referencing type of *SST* included in the user-defined type descriptor of *SST*.

1) If <cast to ref> is specified, then let *FNREF* be <cast to ref identifier>; otherwise, let *FNREF* be the <qualified identifier> of *UDTN*.

2) Case:

A) If <cast to type> is specified, then let *FNTYP* be <cast to type identifier>.

B) Otherwise, the Syntax Rules of Subclause 9.7, "Type name determination", are applied to *BT*, yielding an <identifier> *FNTYP*.

Add the following Syntax Rule:

6) k.1) If <ref cast option> is specified, then exactly one of the following shall be true:

i) <user-defined representation> is specified.

ii) <subtype clause> is specified and the user-defined type descriptor of the direct supertype of *UDT* indicates that the referencing type of the direct supertype of *UDT* has a user-defined representation.

7. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace Syntax Rule 7) a) with:

7) a) Let *M* be the number of <method specification>s MS_i, 1 (one) $\leq i \leq M$, contained in <method specification list>. Let MN_i be the <method name> of MS_i.

8. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Delete Syntax Rule 7) b) i)

9. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Delete Syntax Rule 7) b) ii)

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

10. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

Replace Syntax Rule 7) b) iii) with:

7) b) iii) If MS_i does not specify INSTANCE, CONSTRUCTOR or STATIC, then INSTANCE is implicit.

11. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

Add the following Syntax Rule:

7) b) iv.1) If MS_i specifies CONSTRUCTOR, then:

1) SELF AS RESULT shall be specified.

2) OVERRIDING shall not be specified.

3) MN_i shall be equivalent to the <qualified identifier> of *UDTN*.

4) The <returns data type> shall specify *UDTN*.

5) MS_i specifies an SQL-invoked constructor method.

12. Rationale: Correct use of <specific name> for method specifications by replacing with <specific method name>.

Replace Syntax Rules 7) b) vi), 7) b) vii), 7) b) viii) with:

7) b) vi) If <specific method name> is not specified, then an implementation-dependent <specific method name> whose <schema name> is equivalent to *SN* is implicit.

7) b) vii) If <specific method name> contains a <schema name>, then that <schema name> shall be equivalent to *SN*. If <specific method name> does not contain a <schema name>, then the <schema name> of *SN* is implicit

7) b) viii) The schema identified by the explicit or implicit <schema name> of the <specific method name> shall not include a routine descriptor whose specific name is equivalent to <specific method name> or a user-defined type descriptor that includes a method specification descriptor whose specific method name is equivalent to <specific method name>.

13. Rationale: Remove the anomaly in <SQL parameter namer>s.

Replace Syntax Rule 7) b) ix) with:

7) b) ix) Let PDL_i be the <SQL parameter declaration list> contained in MS_i.

1) No two <SQL parameter name>s contained in PDL_i shall be equivalent.

2) No <SQL parameter name> contained in PDL_i shall be equivalent to SELF.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

14. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

Add the following Syntax Rule:

7) b) xiii) 1.1) Both MS_i and MS_k either specify CONSTRUCTOR, or both do not specify CONSTRUCTOR.

15. Rationale: <method characteristic> should not include <transform group specification>.

Replace Syntax Rule 7) b) xv) 1) with:

7) b) xv) 1) The <method characteristics> of MS_i shall contain at most one <language clause>, at most one <parameter style clause>, at most one <deterministic characteristic>, at most one <SQL-data access indication>, and at most one <null-call clause>.

16. Rationale: Clarify the semantics of SQL-data access indication.

Replace Syntax Rule 7) b) xv) 4) with:

7) b) xv) 4) <SQL-data access indication> shall be specified.

17. Rationale: <method characteristic> should not include <transform group specification>.

Replace Syntax Rule 7) b) xv) 6) A) III) with:

7) b) xv) 6) A) III) <parameter style clause> shall not be specified.

18. Rationale: <method characteristic> should not include <transform group specification>.

Delete Syntax Rule 7) b) xv) 6) B) II)

19. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

Add the following Syntax Rule:

7) b) xv) 7) D.1) MS_i does not specify CONSTRUCTOR.

20. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

Replace Syntax Rule 7) b) xv) 10):

7) b) xv) 10) If MS_i does not specify STATIC or CONSTRUCTOR, then there shall be no SQL-invoked function F that satisfies all the following conditions:

A) The routine name of F and RN_i have equivalent <qualified identifier>s.

B) If F is not a static method, then F has AN_i SQL parameters; otherwise, F has (AN_i - 1) SQL parameters.

C) The data type being defined is a proper subtype of
Case:

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

I) If *F* is not a static method, then the declared type of the first SQL parameter of *F*.

II) Otherwise, the user-defined type whose user-defined type descriptor includes the routine descriptor of *F*.

D) The declared type of the *i*-th SQL parameter in *NPLi* ,$2 \le i \le AN_i$ is compatible with

Case:

I) If *F* is not a static method, then the declared type of *i*-th SQL parameter of *F*.

II) Otherwise, the declared type of the (*i*-1)-th SQL parameter of *F*.

21. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

Replace Syntax Rule 7) b) xvi) 1) with:

7) b) xvi) 1) MS_i shall not specify STATIC or CONSTRUCTOR.

22. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

Add the following Syntax Rule:

7) b) xvi) 3) D.1) The descriptor of COMS shall not include an indication that STATIC or CONSTRUCTOR has been specified.

23. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

Replace Syntax Rule 7) b) xvi) 5):

7) b) xvi) 5) COMS shall not be the corresponding method specification of a mutator or observer function.

24. Rationale: For overriding method specifications, provide missing syntax rules to enforce proper <locator indication> in <SQL parameter declaration list>s. Also provide missing syntax rules to enforce that the parameter names are properly aligned with the parameter names of the corresponding parameters of the corresponding original method specification.

Replace Syntax Rule 7) b) xvi) 6) with:

7) b) xvi) 6) For *j* ranging from 1 (one) to N_i, all of the following shall be true:

A) If the $POVMS_j$ contains an <SQL parameter name> *PNM1*, then $PCOMS_j$ contains an <SQL parameter name> that is equivalent to *PNM1*.

B) If $PCOMS_j$ contains an <SQL parameter name> *PNM2*, then $POVMS_j$ contains an <SQL parameter name> that is equivalent to *PNM2*.

C) If the $POVMS_j$ contains a <locator indication>, then $PCOMS_j$ contains a <locator indication>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

D) If $PCOMS_j$ contains a <locator indication>, then $POVMS_j$ contains a <locator indication>.

25. Rationale: Correct the first <user-defined cast definition> and the <transform definition>.

Replace General Rule 1) b) with:

1) b) The following SQL-statements are executed without further Access Rule checking:

```
CREATE FUNCTION SN.FNUDT ( SDTP SDT )
RETURNS UDTN
LANGUAGE SQL
DETERMINISTIC
RETURN RV1

CREATE METHOD SN.FNSDT ( )
RETURNS SDT
FOR UDTN
LANGUAGE SQL
DETERMINISTIC
RETURN RV2

CREATE CAST ( UDTN AS SDT )
WITH METHOD SN.FNSDT ( ) FOR UDTN
AS ASSIGNMENT

CREATE CAST ( SDT AS UDTN )
WITH FUNCTION SN.FNUDT ( SDT )
AS ASSIGNMENT

CREATE TRANSFORM FOR UDTN
FNUDT ( FROM SQL WITH METHOD SN.FNSDT ( ) FOR UDTN,
TO SQL WITH FUNCTION SN.FNUDT ( SDT ) )
```

where: *SN* is the explicit or implicit <schema name> of *UDTN*; *RV1* is an implementation-dependent <value expression> such that for every invocation of *SN.FNUDT* with argument value *AV1*, *RV1* evaluates to the representation of *AV1* in the data type identified by *UDTN*; *RV2* is an implementation-dependent <value expression> such that for every invocation of *SN.FNSDT* with argument value *AV2*, *RV2* evaluates to the representation of *AV2* in the data type *SDT*, and *SDTP* is an <SQL parameter name> arbitrarily chosen.

26. Rationale: Provide appropriate declarations for the system-generated ordering for distinct type whose source type is LOB

Add the following General Rule:

1) c) Case:

i) If *SDT* is not a large object type, then the following SQL-statement is executed without further Access Rule checking:

```
CREATE ORDERING FOR UDTN
ORDER FULL BY MAP
WITH METHOD     FNSDT ( ) FOR UDTN
```

ii) If *SDT* is a large object type, and the SQL-implementation supports Feature T042, "Extended LOB data type support", then the following SQL-statement is executed without further Access Rule checking:

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

```
CREATE ORDERING FOR UDTN
ORDER EQUALS ONLY BY MAP
WITH METHOD FNSDT ( ) FOR UDTN
```

NOTE 228 — If *SDT* is a large object type, and the SQL-implementation does not support Feature T042, "Extended LOB data type support", then no ordering for *UDTN* is created.

27. *Rationale: Editorial.*

Replace General Rule 2) b) with:

2) b) If INSTANTIABLE is specified, then let *V* be a value of the most specific type *UDT* such that, for every attribute *ATT* of *UDT*, invocation of the corresponding observer function on *V* yields the default value for *ATT*. The following <SQL-invoked routine> is effectively executed:

```
CREATE FUNCTION UDTN ( ) RETURNS UDTN
RETURN V
```

This SQL-invoked function is the *constructor function* for *UDT*.

28. *Rationale: Correct the <SQL-invoked routine> statement that specifies SN.FNTYP and correct the last <user-defined cast definition>. Correct the specification of implicit generation of cast functions when reference values are generated via <user-defined representation>.*

Replace General Rule 2) c) with:

2) c) If <user-defined representation> is specified or if <subtype clause> is specified and the user-defined type descriptor of the direct supertype of *UDT* indicates that the referencing type of the direct supertype of *UDT* has a user-defined representation, then the following SQL-statements are executed without further Access Rule checking:

```
CREATE FUNCTION SN.FNREF ( BTP BT )
  RETURNS REF ( UDTN )
  LANGUAGE SQL
  DETERMINISTIC
  STATIC DISPATCH
  RETURN RV1

CREATE FUNCTION SN.FNTYP ( UDTNP REF( UDTN ))
  RETURNS BT
  LANGUAGE SQL
  DETERMINISTIC
  STATIC DISPATCH
  RETURN RV2

CREATE CAST ( BT AS REF ( UDTN ) )
  WITH FUNCTION SN.FNREF ( BT )

CREATE CAST ( REF( UDTN ) AS BT )
  WITH FUNCTION SN.FNTYP ( REF( UDTN ) )
```

where: *SN* is the explicit or implicit <schema name> of *UDTN*; *RV1* is an implementation-dependent <value expression> such that for every invocation of *SN.FNREF* with argument value *AV1*, *RV1* evaluates to the representation of *AV1* in the data type identified by REF (*UDTN*); *RV2* is an implementation-dependent <value expression> such that for every invocation of *SN.FNTYP* with argument value *AV2*, *RV2* evaluates to the representation of *AV2* in the data type *BT*; and *UDTNP* is an <SQL parameter name> arbitrarily chosen.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

29. Rationale: Redundant subrule.

Replace General Rule 5) e) with:

5) e) If *UDT* is a distinct type, then the data type descriptor of *SDT*.

30. Rationale: Correct the specification of implicit generation of cast functions when reference values are generated via <user-defined representation>.

Replace General Rule 5) f) v) with:

5) f) v) Case:

1) If <user-defined representation> is specified, then an indication that the referencing type of *UDT* has a user-defined representation and the data type descriptor of the representation type of the referencing type of *UDT*.

2) If <derived representation> is specified, then an indication that the referencing type of *UDT* has a derived representation, and the attributes specified by <list of attributes>.

3) Otherwise, an indication that the referencing type of *UDT* has a system-defined representation.

31. Rationale: Ordering form and category for a structured type.

Add the following General Rules:

5) f) v.1) The ordering form NONE.

v.2) The ordering category STATE.

32. Rationale: Correct the specification of implicit generation of cast functions when reference values are generated via <user-defined representation>.

Add the following General Rule:

5) f) v.3) If <subtype clause> is specified, then let *SUDT* be the direct supertype of *UDT* and let *DSUDT* be the user-defined type descriptor of *SUDT*. Let *RUDT* be the referencing type of *UDT* and let *RSUDT* be the referencing type of *SUDT*.

Case:

1) If *DSUDT* indicates that *RSUDT* has a user-defined representation, then an indication that *RUDT* has a user-defined representation and the data type descriptor of the representation type of *RSUDT* included in *DSUDT*.

2) If *DSUDT* indicates that *RSUDT* has a derived representation, then an indication that *RUDT* has a derived representation and the list of attributes included in *DSUDT*.

3) If *DSUDT* indicates that *RSUDT* has a system-defined representation, then an indication that *RUDT* has a system-defined representation.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

33. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

Replace General Rule 5) g) ii) with:

5) g) ii) An indication of whether STATIC or CONSTRUCTOR is specified.

34. Rationale: Correct use of <specific name> for method specifications by replacing with <specific method name>.

Replace General Rule 5) g) iv) with:

5) g) iv) The <specific method name> of *ORMS*.

35. Rationale: <method characteristic> should not include <transform group specification>.

Delete General Rule 5) g) x)

36. Rationale: Clarify the semantics of SQL-data access indication.

Replace General Rule 5) g) xii) with:

5) g) xii) An indication of whether the method possibly modifies SQL data, possibly reads SQL-data, possibly contains SQL, or does not possibly contain SQL.

37. Rationale: Correct use of <specific name> for method specifications by replacing with <specific method name>.

Replace General Rule 5) h) iii) with:

5) g) iii) The <specific method name> of *OVMS*.

38. Rationale: <method characteristic> should not include <transform group specification>.

Delete General Rule 5) h) ix)

39. Rationale: Clarify that Feature S023 also comprises <method specification list>.

Add the following Conformance Rule:

8.1) Without Feature S023, "Basic structured types", conforming SQL language shall not specify <method specification list>.

11.41 <attribute definition>

1. Rationale: Correct oversight in rules for recursive type definition.

Add the following Syntax Rule:

6.1) *DT* shall not be based on *UDT*.

NOTE 229.1 — The notion of one data type being based on another data type is defined in Subclause 4.1, "Data types".

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

11.43 <add attribute definition>

1. *Rationale: Correct oversight in rules for recursive type definition.*

Replace Syntax Rule 3) with:

3) The declared type of a column of a base table shall not be SPRD, SBRD, SPAD, or SBAD.

2. *Rationale: Correct oversight in rules for recursive type definition.*

Replace Syntax Rule 4) with:

4) The declared type of a column of a base table shall not be based on D.

11.44 <drop attribute definition>

1. *Rationale: Editorial.*

Replace Syntax Rule 4) with:

4) Let *SPD* be any supertype of *D*. Let *SBD* be any subtype of *D*. Let *RD* be the reference type whose referenced type is *D*. Let *SPRD* be any supertype of *RD*. Let *SBRD* be any subtype of *RD*. Let *AD* be the array type whose element type is *D*. Let *SPAD* be any array type whose element type is *SPD* or *SPRD*. Let *SBAD* be any array type whose element type is *SBD* or *SBRD*.

2. *Rationale: Correct oversight in rules for recursive type definition.*

Replace Syntax Rule 5) with:

5) The declared type of any column of any base table shall not be *SPRD*, *SBRD*, *SPAD*, or *SBAD*.

3. *Rationale: Correct oversight in rules for recursive type definition.*

Replace Syntax Rule 6) with:

6) The declared type of any column of any base table shall not be based on *D*.

4. *Rationale: Dependency on a user-defined ordering requires only the existence of the ordering..*

Replace Syntax Rule 8) c) with:

8) c) *R1* or *R2* shall not be the ordering function in the descriptor of any user-defined type..

11.45 <add original method specification>

1. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Delete Syntax Rule 4)

2. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Delete Syntax Rule 5)

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

3. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace Syntax Rule 6) with:

6) If *PORMS* does not specify INSTANCE, CONSTRUCTOR or STATIC, then INSTANCE is implicit.

6.1) If *PORMS* specifies CONSTRUCTOR, then:

a) SELF AS RESULT shall be specified.

b) *MN* shall be equivalent to the <qualified identifier> of *DN*.

c) The <returns data type> shall specify *DN*.

d) *PORMS* specifies an SQL-invoked constructor method.

4. *Rationale: Correct use of <specific name> for method specifications by replacing with <specific method name>.*

Replace Syntax Rule 9) with:

9) Case:

a) If *PORMS* does not specify <specific method name>, then an implementation-dependent <specific method name> is implicit whose <schema name> is equivalent to *SN*.

b) Otherwise:

Case:

i) If <specific method name> contains a <schema name>, then that <schema name> shall be equivalent to *SN*.

ii) Otherwise, the <schema name> *SN* is implicit.

The schema identified by the explicit or implicit <schema name> of the <specific method name> shall not include a routine descriptor whose specific name is equivalent to <specific method name> or a user-defined type descriptor that includes a method specification descriptor whose specific method name is equivalent to <specific method name>.

5. *Rationale: <method characteristic> should not include <transform group specification>.*

Replace Syntax Rule 10) with:

10) *MCH* shall contain at most one <language clause>, at most one <parameter style clause>, at most one <deterministic characteristic>, at most one <SQL-data access indication>, and at most one <null-call clause>.

a) If <language clause> is not specified in *MCH*, then LANGUAGE SQL is implicit.

b) Case:

i) If LANGUAGE SQL is specified or implied, then:

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

1) <parameter style clause> shall not be specified.

2) <SQL-data access indication> shall not specify NO SQL.

3) Every <SQL parameter declaration> contained in <SQL parameter declaration list> shall contain an <SQL parameter name>.

4) The <returns clause> shall not specify a <result cast>.

ii) Otherwise:

1) If <parameter style clause> is not specified, then PARAMETER STYLE SQL is implicit.

2) If a <result cast> is specified, then let *V* be some value of the <data type> specified in the <result cast> and let *RT* be the <returns data type>. The following shall be valid according to the Syntax Rules of Subclause 6.22, "<cast specification>":
CAST (*V* AS *RT*)

3) If <result cast from type> *RCT* simply contains <locator indication>, then *RCT* shall be either binary large object type, character large object type, array type, or user-defined type.

c) If <deterministic characteristic> is not specified in *MCH*, then NOT DETERMINISTIC is implicit.

d) If <SQL-data access indication> is not specified, then CONTAINS SQL is implicit.

e) If <null call clause> is not specified in *MCH*, then CALLED ON NULL INPUT is implicit.

6. *Rationale: Remove the anomaly in <SQL parameter name>s.*

Replace Syntax Rule 11) with:

11) No two <SQL parameter name>s contained in *MPDL* shall be equivalent.

11.1) No <SQL parameter name> contained in *MPDL* shall be equivalent to SELF.

7. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace Syntax Rule 15) with:

15) Case:

a) If *ORMS* specifies CONSTRUCTOR, then let a *conflicting method specification CMS* be a method specification whose descriptor is included in the descriptor of *D*, such that the following are all true:

i) *MPDL* and the unaugmented SQL parameter list of *CMS* have the same number *N* of SQL parameters.

ii) Let $PCMS_j$, 1 (one) $\leq j \leq N$, be the *j*-th SQL parameter in the unaugmented SQL parameter declaration list of *CMS*. Let PMS_j, 1 (one) $\leq j \leq N$, be the *j*-th SQL parameter in the unaugmented SQL parameter declaration list *MPDL*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

iii) For *j* varying from 1 (one) to *N*, the Syntax Rules of Subclause 10.14, "Data type identity", are applied with the declared type of $PCMS_j$ and the declared type of PMS_j.

iv) *CMS* is an SQL-invoked constructor method.

b) Otherwise, let a *conflicting method specification CMS* be a method specification whose descriptor is included in the descriptor of some *SPD* or *SBD*, such that the following are all true:

i) *MN* and the method name included in the descriptor of *CMS* are equivalent.

ii) *MPDL* and the unaugmented SQL parameter list of *CMS* have the same number *N* of SQL parameters.

iii) Let $PCMS_j$, 1 (one) $\leq j \leq N$, be the *j*-th SQL parameter in the unaugmented SQL parameter declaration list of *CMS*. Let PMS_j, 1 (one) $\leq j \leq N$, be the *j*-th SQL parameter in the unaugmented SQL parameter declaration list *MPDL*.

iv) For *j* varying from 1 (one) to *N*, the Syntax Rules of Subclause 10.14, "Data type identity", are applied with the declared type of $PCMS_j$ and the declared type of PMS_j.

v) *CMS* and *ORMS* either both are instance methods or one of *CMS* and *ORMS* is a static method and the other is an instance method.

8. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

Replace Syntax Rule 18) with:

18) If *PORMS* does not specify STATIC or CONSTRUCTOR, then there shall be no SQL-invoked function *F* that satisfies all the following conditions:

a) *F* is not an SQL-invoked method.

b) The <routine name> of *F* and *RN* have equivalent <qualified identifier>s.

c) *F* has *AN* SQL parameters.

d) *D* is a subtype or supertype of the declared type of the first SQL parameter of *F*.

e) The declared type of the *i*-th SQL parameter in *NPL*, $2 \leq i \leq AN$ is compatible with the declared type of *i*-th SQL parameter of *F*.

9. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

Replace General Rule 1 b) with:

1) b) An indication of whether STATIC or CONSTRUCTOR is specified.

10. Rationale: Correct use of <specific name> for method specifications by replacing with <specific method name>.

Replace General Rule 1) d) with:

1) d) The <specific method name> of *PORMS*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11. Rationale: <method characteristic> should not include <transform group specification>.

Delete General Rule 1) j)

12. Rationale: Clarify the semantics of SQL-data access indication.

Replace General Rule 1) n) with:

1) n) An indication of whether the method possibly modifies SQL data, possibly reads SQL-data, possibly contains SQL, or does not possibly contain SQL.

11.46 <add overriding method specification>

1. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

Replace Syntax Rule 2) with:

2) Let *POVMS* be the <partial method specification> immediately contained in *OVMS*. *POVMS* shall not specify STATIC or CONSTRUCTOR.

2. Rationale: Remove syntax rule that conflicts with format rules.

Delete Syntax Rule 5)

3. Rationale: Correct use of <specific name> for method specifications by replacing with <specific method name>.

Replace Syntax Rule 7) with:

7) Case:

a) If *POVMS* does not specify <specific method name>, then an implementation-dependent <specific method name> is implicit whose <schema name> is equivalent to *SN*.

b) Otherwise:

Case:

i) If <specific method name> contains a <schema name>, then that <schema name> shall be equivalent to *SN*.

ii) Otherwise, the <schema name> *SN* is implicit.

The schema identified by the explicit or implicit <schema name> of the <specific method name> shall not include a routine descriptor whose specific name is equivalent to <specific method name> or a user-defined type descriptor that includes a method specification descriptor whose specific method name is equivalent to <specific method name>.

4. Rationale: Correct improper use of terms in Syntax Rule 9) a).

Replace Syntax Rule 9) a) with:

9) a) *MN* and the <method name> of *COMS* are equivalent.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

5. *Rationale: Remove improper use of "shall".*

Replace Syntax Rule 9) b) with:

9) b) Let *N* be the number of elements of the augmented SQL parameter declaration list *UPCOMS* generally included in the descriptor of *COMS*. *MPDL* contains (*N*-1) SQL parameter declarations.

6. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Add the following Syntax Rule:

9) c.1) The descriptor of *COMS* shall not include an indication that STATIC or CONSTRUCTOR has been specified.

7. *Rationale: Rectify misplacement of Syntax Rule 9) d); correct the subrules i) and iv) to enforce correct match of parameter names and locator indications.*

Delete Syntax Rule 9) d)

8. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Delete Syntax Rule 11)

9. *Rationale: Rectify misplacement of Syntax Rule 9) d); correct the subrules i) and iv) to enforce correct match of parameter names and locator indications.*

Add the following Syntax Rule:

12.1) For *i* varying from 2 to *N*:

a) If $POVMS_{i-1}$ contains an <SQL parameter name> *PNM1*, then the descriptor of the *i*-th parameter of the augmented <SQL parameter declaration list> of *UPCOMS* shall include a parameter name that is equivalent to *PNM1*.

b) If the descriptor of the *i*-th parameter of the augmented <SQL parameter declaration list> of *UPCOMS* includes a parameter name *PNM2*, then $POVMS_{i-1}$ shall contain an <SQL parameter name> that is equivalent to *PNM2*.

c) $POVMS_{i-1}$ shall not contain <parameter mode>. A <parameter mode> IN is implicit.

d) $POVMS_{i-1}$ shall not specify RESULT.

e) If the <parameter type> PT_{i-1} immediately contained in $POVMS_{i-1}$ contains a <locator indication>, then the descriptor of the *i*-th parameter of the augmented <SQL parameter declaration list> of *UPCOMS* shall include a <locator indication>.

f) If the descriptor of the *i*-th parameter of the augmented <SQL parameter declaration list> of *UPCOMS* includes a <locator indication>, then the <parameter type> PT_{i-1} immediately contained in $POVMS_{i-1}$ shall contain a <locator indication>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10. Rationale: Add missing syntax rule that prevents the addition of a conflicting overriding method specification to some structured type.

Add the following Syntax Rule:

13.1) Let a *conflicting overriding method specification COVMS* be an overriding method specification that is included in the descriptor of *D*, such that the following are all true:

a) *MN* and the method name of *COVMS* are equivalent.

b) The augmented SQL parameter declaration list of *COVMS* contains *N* elements.

c) For *i* varying from 2 to *N*, the data types of the SQL parameter *POVMSi-1* and the SQL parameter *PCOVMSi* of *COVMS* are compatible.

There shall be no conflicting overriding method specification *COVMS*.

11. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

Add the following Syntax Rule:

16) b) ii) 3.1) MSD_i does not include an indication that CONSTRUCTOR has been specified.

12. Rationale: Correct use of <specific name> for method specifications by replacing with <specific method name>.

Replace General Rule 1) c) with:

1) c) The <specific method name> of *POVMS*.

13. Rationale: <method characteristic> should not include <transform group specification>.

Delete General Rule 1) j)

11.47 <drop method specification>

1. Rationale: Correct use of <specific name> for method specifications by replacing with <specific method name>.

Replace the Format with:

```
<drop method specification> ::=
    DROP <specific method specification designator> RESTRICT

<specific method specification designator> ::=
      SPECIFIC METHOD <specific method name>
    | [ INSTANCE | STATIC | CONSTRUCTOR] METHOD <method name>
        [ <data type list> ]
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

2. *Rationale: Correct use of <specific name> for method specifications by replacing with <specific method name>.*

Replace the Syntax Rules with:

1) Let *D* be the user-defined type identified by the <user-defined type name> *DN* immediately contained in the <alter type statement> containing the <drop method specification> *DORMS*. Let *DSN* be the explicit or implicit <schema name> of *DN*. Let *SMSD* be the <specific method specification designator> immediately contained in *DORMS*.

2) If *SMSD* immediately contains a <specific method name> *SMN*, then

   a) If <specific method name> contains a <schema name>, then that <schema name> shall be equivalent to *DSN*. If <specific method name> does not contain a <schema name>, then the <schema name> *DSN* is implicit.

   b) The descriptor of *D* shall include a method specification descriptor *DOOMS* whose specific method name is equivalent to *SMN*.

   c) Let *PDL* be the augmented parameter list included in *DOOMS*.

   d) Let *MN* be the <method name> included in *DOOMS*.

3) If *SMSD* immediately contains a <method name> *ME*, then:

   a) If none of INSTANCE, STATIC, or CONSTRUCTOR is immediately contained in *SMSD*, then INSTANCE is implicit.

   b) The descriptor of *D* shall include a method specification descriptor *DOOMS* whose method name *MN* is equivalent to *ME*.

   c) If *SMSD* immediately contains a <data type list> *DTL*, then

      Case:

      i) If STATIC is specified, then the descriptor of *D* shall include exactly one method specification descriptor *DOOMS* which includes:

         A) An indication that the method specification is STATIC.

         B) An indication that the method specification is original.

         C) An augmented parameter list *PDL* such that for all *i*, the declared type of its *i*-th parameter is identical to the *i*-th declared type in *DTL*.

      ii) If CONSTRUCTOR is specified, then the descriptor of *D* shall include exactly one method specification descriptor *DOOMS* which includes:

         A) An indication that the method specification is CONSTRUCTOR.

         B) An indication that the method specification is original.

         C) An augmented parameter list *PDL* such that for all $i > 1$, the declared type of its *i*-th parameter is identical to the *(i -1)*-th declared type in *DTL* and the declared type of the first parameter of *PDL* is identical to *DN*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

iii) Otherwise, the descriptor of *D* shall include exactly one method specification descriptor *DOOMS* for which:

A) If *DOOMS* includes an indication that the method specification is original, then *DOOMS* shall not include an indication that the method specification is either STATIC or CONSTRUCTOR.

B) *DOOMS* includes an augmented parameter list *PDL* such that for all $i > 1$, the declared type of its *i*-th parameter is identical to the *(i -1)*-th declared type in *DTL* and the declared type of the first parameter of *PDL* is identical to *DN*.

d) If *SMSD* does not immediately contain a <data type list>, then:

Case:

i) If STATIC is specified, the descriptor of *D* shall include exactly one method specification descriptor *DOOMS* which includes an indications that the method specification is both original and STATIC.

ii) If CONSTRUCTOR is specified, the descriptor of *D* shall include exactly one method specification descriptor *DOOMS* which includes an indications that the method specification is both original and CONSTRUCTOR.

iii) Otherwise, the descriptor of *D* shall include exactly one method specification descriptor *DOOMS* for which: if *DOOMS* includes an indication that the method specification is original, then *DOOMS* shall not include an indication that the method specification is either STATIC or CONSTRUCTOR.

4) Case:

a) If *DOOMS* includes an indication that the method specification is original, then:

Case:

i) If *DOOMS* includes an indication that the method specification specified STATIC, then there shall be no SQL-invoked function *F* that satisfies all of the following conditions:

1) The <routine name> of *F* and *MN* have equivalent <qualified identifier>s.

2) If *N* is the number of elements in *PDL*, then *F* has *N* SQL parameters.

3) The declared type of the first SQL parameter of *F* is *D*.

4) The declared type of the *i*-th element of *PDL*, $1 <= i <= N$, is compatible with the declared type of SQL parameter P_i of *F*.

5) *F* is an SQL-invoked method.

6) *F* includes an indication that STATIC is specified.

ii) If *DOOMS* includes an indication that the method specification specified CONSTRUCTOR, then there shall be no SQL-invoked function *F* that satisfies all of the following conditions:

1) The <routine name> of *F* and *MN* have equivalent <qualified identifier>s.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

2) If *N* is the number of elements in *PDL*, then *F* has *N* SQL parameters.

3) The declared type of the first SQL parameter of *F* is *D*.

4) The declared type of the *i*-th element of *PDL*, 2 <= *i* <= *N*, is compatible with the declared type of SQL parameter P_i of *F*.

5) *F* is an SQL-invoked method.

6) *F* includes an indication that CONSTRUCTOR is specified.

iii) Otherwise:

1) There shall be no proper subtype *PSBD* of *D* whose descriptor includes the descriptor *DOVMS* of an overriding method specification such that all of the following is true:

A) *MN* and the < method name> included in *DOVMS* have equivalent <qualified identifier>s.

B) If *N* is the number of elements in *PDL*, then the augmented SQL parameter declaration list *APDL* included in *DOVMS* has *N* SQL parameters.

C) *PSBD* is the declared type the first SQL parameter of *APDL*.

D) The declared type of the *i*-th element of *PDL*, 2 <= *i* <= *N*, is compatible with the declared type of SQL-parameter P_i of *APDL*.

2) There shall be no SQL-invoked function *F* that satisfies all of the following conditions:

A) The <routine name> of *F* and *MN* have equivalent <qualified identifier>s.

B) If *N* is the number of elements in *PDL*, then *F* has *N* SQL parameters.

C) The declared type of the first SQL parameter of *F* is *D*.

D) The declared type of the *i*-th element of *PDL*, 2 <= *i* <= *N*, is compatible with the declared type of SQL parameter P_i of *F*.

E) *F* is an SQL-invoked method.

F) *F* does not include an indication that either STATIC or CONSTRUCTOR is specified.

b) Otherwise, there shall be no SQL-invoked function *F* that satisfies all of the following conditions:

i) The <routine name> of *F* and *MN* have equivalent <qualified identifier>s.

ii) If *N* is the number of elements in *PDL*, then *F* has *N* SQL parameters.

iii) The declared type of the first SQL parameter of *F* is *D*.

iv) The declared type of the *i*-th element of *PDL*, 2 <= *i* <= *N*, is compatible with the declared type of SQL parameter P_i of *F*.

v) *F* is an SQL-invoked method.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

vi) *F* does not include an indication that either STATIC or CONSTRUCTOR is specified.

Replace the General Rules with:

1) Let *STDS* be the descriptor of *D*.

2) *DOOMS* is removed from *STDS*.

3) *DOOMS* is destroyed.

11.48 <drop data type statement>

1. Rationale: Correct oversight in rules for recursive type definition.

Replace Syntax Rule 4) a) with:

4) a) The declared type of no column, field, or attribute whose descriptor is not included in the descriptor of *D* shall be based on *SRD*, or *SAD*.

2. Rationale: Correct oversight in rules for recursive type definition.

Replace Syntax Rule 4) b) with.

4) b) The declared type of no column, field, or attribute shall be based on *D*.

3. Rationale: Dependency on a user-defined ordering requires only the existence of the ordering.

Replace Syntax Rule 4) h) iii) with:

4) h) iii) *R* shall not be the ordering function included in the descriptor of any user-defined type.

Delete Syntax Rule 4) h) iv).

11.49 <SQL-invoked routine>

1. Rationale: Clarify that implicit schema of user-defined type is schema name of either the <schema definition> or the <SQL-client module definition> in which the corresponding SQL statement appears. Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

In the Format, replace the production for <method specification designator> with:

```
<method specification designator> ::=
     [ INSTANCE | STATIC | CONSTRUCTOR ] METHOD <method name>
     <SQL parameter declaration list>
     [ <returns clause> ]
     FOR <user-defined type name>
```

2. Rationale: <method characteristic> should not include <transform group specification>.

In the Format, replace the production for <routine characteristic> with:

```
<routine characteristic> ::=
       <language clause>
     | <parameter style clause>
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

```
| SPECIFIC <specific name>
| <deterministic characteristic>
| <SQL-data access indication>
| <null-call clause>
| <dynamic result sets characteristic>
```

3. *Rationale: <method characteristic> should not include <transform group specification>.*

In the Format, replace the production for <external body reference> with:

```
<external body reference> ::=
     EXTERNAL [ NAME <external routine name> ]
     [ <parameter style clause> ]
     [ <transform group specification> ]
     [ <external security clause> ]
```

4. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace Syntax Rule 3) with:

3) An <SQL-invoked routine> specified as an <SQL-invoked procedure> is called an *SQL-invoked procedure*; an <SQL-invoked routine> specified as an <SQL-invoked function> is called *an SQL-invoked function*. An <SQL-invoked function> that specifies a <method specification designator> is further called an *SQL-invoked method*. An SQL-invoked method that specifies STATIC is called a *static SQL-invoked method*. An SQL-invoked method that specifies CONSTRUCTOR is called an *SQL -invoked constructor method*.

5. *Rationale: Clarify that implicit schema of user-defined type is schema name of either the <schema definition> or the <SQL-client module definition> in which the corresponding SQL statement appears.*

Replace Syntax Rule 4) a) with:

4) a) Let *UDTN* be the <user-defined type name> immediately contained in <method specification designator>. Let *UDT* be the user-defined type identified by *UDTN*.

6. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace Syntax Rule 4) b) with:

4) b) There shall exist a method specification descriptor *DMS* in the descriptor of *UDT* such that the <method name> of *DMS* is equivalent to the <method name>, *DMS* indicates STATIC if and only if the <method specification designator> specifies STATIC, *DMS* indicates CONSTRUCTOR if and only if the <method specification designator> specifies CONSTRUCTOR and the declared type of every SQL parameter in the unaugmented SQL parameter declaration list in *DMS* is compatible with the declared type of the corresponding SQL parameter in the <SQL parameter declaration list> contained in the <method specification designator>. *DMS* identifies the corresponding method specification of the <method specification designator>.

7. *Rationale: Remove the anomaly in <SQL parameter name>s.*

Replace Syntax Rule 4) e) with:

4) e) For i varying from 1 (one) to MN, the <SQL parameter name>s contained in $PCOMS_i$ and $POVMS_i$ shall be equivalent.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8. Rationale: <method characteristic> should not include <transform group specification>.

Delete Syntax Rule 4) j).

9. Rationale: Correct use of <specific name> for method specifications by replacing with <specific method name>.

Replace Syntax Rule 4) k) with:

4) k) Let *SPN* be the <specific method name> in *DMS*. *SPN* is the <specific name> of *R*.

10. Rationale: <method characteristic> should not include <transform group specification>.

Replace Syntax Rule 5) a) with:

5) a) <routine characteristics> shall contain at most one <language clause>, at most one <parameter style clause>, at most one <specific name>, at most one <deterministic characteristic>, at most one <SQL-data access indication>, at most one <null-call clause>, and at most one <dynamic result sets characteristic>.

11. Rationale: Clarify the semantics of SQL-data access indication.

Replace Syntax Rule 5) g) with:

5) g) <SQL-data access indication> shall be specified.

12. Rationale: <method characteristic> should not include <transform group specification>.

Replace Syntax Rule 5) j) iii) with:

5) j) iii) <parameter style clause> shall not be specified.

13. Rationale: Remove the anomaly in <SQL parameter name>s.

Replace Syntax Rule 10) with:

10) No two <SQL parameter name>s contained in *NPL* shall be equivalent.

14. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

Add the following Syntax Rule:

15) b) i) 1.1) If *R* is an SQL-invoked constructor method, then let *SCR* be the set containing every SQL-invoked constructor method of type *UDT*, including *R*, whose <schema qualified routine name> is equivalent to *RN* and whose number of SQL parameters is *PN*.

15. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

Replace Syntax Rule 15) b) i) 2) with:

15) b) i) 2) Otherwise, let *SCR* be the set containing every SQL-invoked function in *S* that is not a static SQL-invoked method or an SQL-invoked constructor method, including *R*, whose

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

<schema qualified routine name> is equivalent to *RN* and whose number of SQL parameters is *PN*.

16. Rationale: Clarify the semantics of SQL-data access indication.

Replace Syntax Rule 18) c) with:

18) c) If READS SQL DATA is specified, then it is implementation-defined whether the <SQL routine body> shall not contain an <SQL procedure statement> *S* that satisfies at least one of the following:

i) *S* is an <SQL data change statement>.

ii) *S* contains a <routine invocation> whose subject routine is an SQL-invoked routine that possibly modifies SQL-data.

iii) *S* contains an <SQL procedure statement> that is an <SQL data change statement>.

Replace Syntax Rule 18) d) with:

18) d) If CONTAINS SQL is specified, then it is implementation-defined whether the <SQL routine body> shall not contain an <SQL procedure statement> *S* that satisfies at least one of the following:

i) *S* is an <SQL data statement> other than <free locator statement> and <hold locator statement>.

ii) *S* contains a <routine invocation> whose subject routine is an SQL-invoked routine that possibly modifies SQL-data or possibly reads SQL-data.

iii) *S* contains an <SQL procedure statement> that is an <SQL data statement> other than <free locator statement> and <hold locator statement>.

17. Rationale: Clarify that the implicit schema of user-defined type used as a return type or parameter type is resolved using the SQL-path.

Replace Syntax Rule 19) d) iii) 2) B) II) with:

19) d) iii) 2) B) II) If *RT* specifies a <user-defined type> without a <locator indication>

1) Case:

a) If *R* is an SQL-invoked method that is an overriding method, then the Syntax Rules of Subclause 10.18, "Determination of a to-sql function for an overriding method", are applied with *R* as *ROUTINE*. There shall be an applicable to-sql function *TSF*.

b) Otherwise, the Syntax Rules of Subclause 10.17, "Determination of a to-sql function", are applied with the data type identified by *RT* and the <group name> contained in the <group specification> that contains *RT* as *TYPE* and *GROUP*, respectively. There shall be an applicable to-sql function *TSF*.

2) *TSF* is called the to-sql function associated with the result.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3) Case:

a) If *TSF* is an SQL-invoked method, then *PT* is the <parameter type> of the second SQL parameter of *TSF*.

b) Otherwise, *PT* is the <parameter type> of the first SQL parameter of *TSF*.

18. Rationale: Clarify that the implicit schema of user-defined type used as a return type or parameter type is resolved using the SQL-path.

Replace Syntax Rule 19) d) iv) 1) B) with:

19) d) iv) 1) B) If the <parameter type> T_i simply contained in the *i*-th <SQL parameter declaration> is a <user-defined type> without a <locator indication>, then:

I) Case:

1) If the <parameter mode> immediately contained in the *i*-th <SQL parameter declaration> is IN, then:

a) The Syntax Rules of Subclause 10.15, "Determination of a from-sql function", are applied with the data type identified by T_i and the <group name> contained in the <group specification> that contains T_i as *TYPE* and *GROUP*, respectively. There shall be an applicable from-sql function FSF_i. FSF_i is called the from-sql function associated with the *i*-th SQL parameter.

b) The *i*-th effective SQL parameter list entry is the *i*-th <SQL parameter declaration> with the <parameter type> replaced by the <returns data type> of FSF_i.

2) If the <parameter mode> immediately contained in the *i*-th <SQL parameter declaration> is OUT, then:

a) The Syntax Rules of Subclause 10.17, "Determination of a to-sql function", are applied with the data type identified by T_i and the <group name> contained in the <group specification> that contains T_i as *TYPE* and *GROUP*, respectively. There shall be an applicable to-sql function TSF_i. TSF_i is called the to-sql function associated with *i*-th SQL parameter.

b) The *i*-th effective SQL parameter list entry is the *i*-th <SQL parameter declaration> with the <parameter type> replaced by

Case:

i) If TSF_i is an SQL-invoked method, then the <parameter type> of the second SQL parameter of TSF_i.

ii) Otherwise, the <parameter type> of the first SQL parameter of *TSFi*

.

3) Otherwise:

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

a) The Syntax Rules of Subclause 10.15, "Determination of a from-sql function", are applied with the data type identified by T_i and the <group name> contained in the <group specification> that contains T_i as *TYPE* and *GROUP*, respectively. There shall be an applicable from-sql function FSF_i. FSF_i is called the from-sql function associated with the *i*-th SQL parameter.

b) The Syntax Rules of Subclause 10.17, "Determination of a to-sql function", are applied with the data type identified by T_i and the <group name> contained in the <group specification> that contains T_i as *TYPE* and *GROUP*, respectively. There shall be an applicable to-sql function TSF_i. TSF_i is called the to-sql function associated with the *i*-th SQL parameter.

c) The *i*-th effective SQL parameter list entry is the *i*-th <SQL parameter declaration> with the <parameter type> replaced by the <returns data type> of FSF_i.

19. Rationale: Clarify the semantics of SQL-data access indication.

Replace Syntax Rule 20) with:

20) Case:

a) If <method specification designator> is specified, then:

i) If *DMS* indicates that the method is deterministic, then *R* is deterministic; otherwise, *R* is possibly non-deterministic.

ii) If the SQL-data access indication of *DMS* indicates that the method possibly modifies SQL-data, then *R* possibly modifies SQL-data. If the SQL-data access indication of *DMS* indicates that the method possibly reads SQL-data, then *R* possibly reads SQL-data. If the SQL-data access indication of *DMS* indicates that the method possibly contains SQL, then *R* possibly contains SQL. Otherwise, *R* does not possibly contain SQL.

b) Otherwise:

i) If DETERMINISTIC is specified, then *R* is deterministic; otherwise, it is possibly non-deterministic.

ii) An <SQL-invoked routine> possibly modifies SQL-data if and only if <SQL-data access indication> specifies MODIFIES SQL DATA.

iii) An <SQL-invoked routine> possibly reads SQL-data if and only if <SQL-data access indication> specifies READS SQL DATA.

iv) An <SQL-invoked routine> possibly contains SQL if and only if <SQL-data access indication> specifies CONTAINS SQL.

Replace General Rule 3) o) with:

3) o) The SQL-invoked routine descriptor includes an indication whether the SQL-invoked routine does not possibly contain SQL, possibly contains SQL, possibly reads SQL-data, or possibly modifies SQL-data.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

20. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace General Rule 4) d) with:

4) d) The routine descriptor includes an indication that the SQL-invoked routine is an SQL-invoked function that is an SQL-invoked method, an indication of the user-defined type *UDT*, and an indication of whether STATIC or CONSTRUCTOR was specified.

21. *Rationale: Editorial.*

Replace Conformance Rules 1) and 4) with:

1) Without Feature T471, "Result sets return value", conforming SQL language shall not specify <dynamic result sets characteristic>.

4) Without Feature S241, "Transform functions", conforming SQL language shall not specify <transform group specification>.

22. *Rationale: Remove useless item from Core.*

Add the following Conformance Rule:

10.1 Without at least one of the features:
S231 "Structured Type locators",
S232 "Array Locators",
T041 "Basic LOB data type"
conforming SQL language shall not specify <locator indication>.

11.50 <alter routine statement>

1. *Rationale: Dependency on a user-defined ordering requires only the existence of the ordering..*

Replace Syntax Rule 4) a) with:

4) a) *SR* shall not be the ordering function included in the descriptor of any user-defined type *UDT*.

11.51 <drop routine statement>

1. *Rationale: Dependency on a user-defined ordering requires only the existence of the ordering.*

Replace Syntax Rule 4) with:

4) If RESTRICT is specified, then *SR* shall not be the ordering function included in the descriptor of any user-defined type *UDT*.

2. *Rationale: When dropping a routine that is an order function, it is also necessary to perform a <drop user-defined ordering statement>.*

Add the following General Rule:

3.1) If *SR* is the ordering function included in the descriptor of any user-defined type *UDT*, then let *UDTN* be a <user-defined type name> that identifies *UDT*. The following <drop user-defined ordering statement> is effectively executed without further Access Rule checking:

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

DROP ORDERING FOR *UDTN* CASCADE

11.52 <user-defined cast definition>

1. Rationale: Clarify that implicit schema of user-defined type is schema name of either the <schema definition> or the <SQL-client module definition> in which the corresponding SQL statement appears.

Replace Syntax Rules 4) and 5) with:

4) At least one of *SDT* or *TDT* shall contain a <user-defined type name> or a <reference type>.

5) If *SDT* contains a <user-defined type name>, then let *SSDT* be the schema that includes the descriptor of the user-defined type identified by *SDT*.

2. Rationale: Provide consistent Syntax Rules for comparison operations.

Add the following Syntax Rules:

6) a) i.1) EQUALS ONLY shall be specified.

i.2) The declared type of each attribute of *UDT* shall not be UDT-NC-ordered.

3. Rationale: Clarify that implicit schema of user-defined type is schema name of either the <schema definition> or the <SQL-client module definition> in which the corresponding SQL statement appears.

Replace Syntax Rule 7) with:

7) If *TDT* contains a <user-defined type name>, then let *STDT* be the schema that includes the descriptor of the user-defined type identified by *TDT*.

4. Rationale: Provide consistent Syntax Rules for comparison operations.

Add the following Syntax Rule:

8) c) If the result data type of *F* is a large object type, then ORDER FULL shall not be specified.

5. Rationale: Clarify that implicit schema of user-defined type is schema name of either the <schema definition> or the <SQL-client module definition> in which the corresponding SQL statement appears.

Replace Syntax Rule 9) with:

9) If both *SDT* and *TDT* contain a <user-defined type name> or a <reference type>, then the <authorization identifier> that owns *SSDT* and the <authorization identifier> that owns *STDT* shall be equivalent.

Replace Syntax Rule 10) c) with:

10) c) The <authorization identifier> that owns *SSDT* or *STDT* (both, if both *SDT* and *TDT* are <user-defined type name>s) shall own the schema that includes the SQL-invoked routine descriptor of *F*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6. *Rationale: Provide consistent Conformance Rules for comparison operations.*

Add the following Conformance Rules:

2) Without Feature T042, "Extended LOB data type support", if <state category> is specified, then the declared type of each attribute of *UDT* shall not be LOB-ordered.

3) Without Feature T042, "Extended LOB data type support", if <map category> is specified, the result type of *F* shall not be a large object type.

11.54 <user-defined ordering definition>

1. *Rationale: Clarify that implicit schema of user-defined type is schema name of either the <schema definition> or the <SQL-client module definition> in which the corresponding SQL statement appears.*

In the Format, replace the production for <user-defined ordering definition> with:

```
<user-defined ordering definition> ::=
    CREATE ORDERING FOR <user-defined type name> <ordering form>
```

Replace Syntax Rule 1) with:

1) Let *UDTN* be the <user-defined type name>. Let *UDT* be the user-defined type identified by *UDTN*.

2. *Rationale: Editorial.*

Replace Syntax Rule 2) with:

2) The descriptor of *UDT* shall include an ordering form that specifies NONE.

3. *Rationale: Provide consistent Syntax Rules for comparison operations.*

Add the following Syntax Rules:

6) i.1) EQUALS ONLY shall be specified.

i.2) The declared type of each attribute of *UDT* shall not be UDT-NC-ordered.

Append the following Syntax Rules:

8) c) If the result data type of *F* is a large object type, then ORDER FULL shall not be specified.

4. *Rationale: Editorial.*

Replace Access Rule 1) with:

1) The enabled authorization identifiers shall include the <authorization identifier> that owns the schema that includes the descriptor of *UDT* and the schema that includes the routine descriptor of *F*.

5. *Rationale: Incorrect informative note.*

Delete NOTE 264

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

6. *Rationale: Editorial.*

Replace General Rule 4) with:

4) The <specific routine designator> identifying the ordering function, depending on the ordering category, in the descriptor of *UDT* is set to *SRD*.

7. *Rationale: Editorial.*

Replace Conformance Rule 1) with:

1) Without Feature S251, "User-defined orderings", conforming SQL shall contain no <user-defined ordering definition>.

8. *Rationale: Provide consistent Conformance Rules for comparison operations.*

Add the following Conformance Rules:

2) Without Feature T042, "Extended LOB data type support", if <state category> is specified, then the declared type of each attribute of *UDT* shall not be LOB-ordered.

3) Without Feature T042, "Extended LOB data type support", if <map category> is specified, the result type of *F* shall not be a large object type.

11.55 <drop user-defined ordering statement>

1. *Rationale: Clarify that implicit schema of user-defined type is schema name of either the <schema definition> or the <SQL-client module definition> in which the corresponding SQL statement appears.*

In the Format, replace the production for <drop user-defined ordering statement> with:

```
<drop user-defined ordering statement> ::=
    DROP ORDERING FOR <user-defined type name> <drop behavior>
```

Replace Syntax Rule 1) with:

1) Let *UDTN* be the <user-defined type name>. Let *UDT* be the user-defined type identified by *UDTN*.

2. *Rationale: Editorial.*

Replace Syntax Rule 2) with:

2) The descriptor of *UDT* shall include an ordering form that specifies EQUALS or FULL.

3. *Rationale: Recognize that a <distinct predicate> may depend on a user-defined ordering. Provide the correct set of dependencies for a user-defined ordering.*

Replace Syntax Rule 4) with:

4) Let *S* be the set of types whose comparison type is *UDT*. Let *DEP* designate any of the following:

a) A <comparison predicate>, <between predicate>, <in predicate>, <quantified comparison predicate>, <match predicate>, or <distinct predicate> that simply contains a <row value expression> whose declared type is S-ordered.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) A <table subquery> simply contained in a <quantified comparison predicate>, <in predicate>, <unique predicate>, or <match predicate> that has a column whose declared type is S-ordered.

c) A <set function specification> that specifies DISTINCT, MAX, or MIN and that simply contains a <value expression> whose declared type is S-ordered.

d) A <group by clause> that contains a <column reference> to a column whose declared type is S-ordered.

e) A <query specification> that specifies the <set quantifier> DISTINCT and that has a column whose declared type is S-ordered.

f) A <query expression> that specifies or implies UNION DISTINCT and that has a column whose declared type is S-ordered.

g) A <query expression> that specifies INTERSECT or EXCEPT and that has a column whose declared type is S-ordered.

j) A <joined table> that specifies NATURAL or USING and that has a corresponding join column whose declared type is S-ordered.

Replace Syntax Rule 5) with:

5) If RESTRICT is specified, then *DEP* shall not be specified in any of the following:

a) The <SQL routine body> of any routine descriptor.

b) The <query expression> of any view descriptor.

c) The <search condition> of any constraint descriptor or assertion descriptor.

d) The triggered action of any trigger descriptor.

NOTE 267 — If CASCADE is specified, then such referencing objects will be dropped by the execution of the <revoke statement> specified in the General Rules of this Subclause.

4. Rationale: Provide the correct set of dependencies for a user-defined ordering.

Replace General Rules 1), 2), 3), 4), 5) and 6) with:

1) Let *R* be any SQL-invoked routine that contains *DEP* in its <SQL routine body>. Let *SN* be the specific name of *R*. The following <drop routine statement> is effectively executed without further Access Rule checking:
DROP SPECIFIC ROUTINE *SN* CASCADE

2) Let *V* be any view that contains *DEP* in its <query expression>. Let *VN* be the <table name> of *V*. The following <drop view statement> is effectively executed without further Access Rule checking:
DROP VIEW *VN* CASCADE

3) Let *T* be any table that contains *DEP* in the <search condition> of any constraint *C* whose constraint descriptor is included in the table descriptor of *T*. Let *TN* be the <table name> of *T*. Let *TCN* be the <constraint name> of *C*. The following <alter table statement> is effectively executed without further Access Rule checking:
ALTER TABLE *TN* DROP CONSTRAINT *TCN* CASCADE

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4) Let *A* be any assertion that contains *DEP* in its <search condition>. Let *AN* be the <constraint name> of *A*. The following <drop assertion statement> is effectively executed without further Access Rule checking:
DROP ASSERTION *AN* CASCADE

5) Let *D* be any domain that contains *DEP* in the <search condition> of any constraint descriptor or in the <default option> included in the domain descriptor of *D*. Let *DN* be the <domain name> of *D*. The following <drop domain statement> is effectively executed without further Access Rule checking:
DROP DOMAIN *DN* CASCADE

6) Let *T* be any trigger that contains *DEP* in its triggered action. Let *TN* be the <trigger name> of *T*. The following <drop trigger statement> is effectively executed without further Access Rule checking:
DROP TRIGGER *TN* CASCADE

5. *Rationale: Editorial.*

Replace General Rule 7) with:

7) In the user-defined data type descriptor of *UDT*, the ordering form is set to NONE and the ordering category is set to STATE. No ordering function is included in the user-defined data type descriptor of *UDT*.

11.56 <transform definition>

1. *Rationale: Clarify that implicit schema of user-defined type is schema name of either the <schema definition> or the <SQL-client module definition> in which the corresponding SQL statement appears.*

In the Format, replace the production for <transform definition> with:

```
<transform definition> ::=
     CREATE { TRANSFORM | TRANSFORMS } FOR
        <user-defined type name> <transform group>...
```

11.57 <drop transform statement>

1. *Rationale: Clarify that implicit schema of user-defined type is schema name of either the <schema definition> or the <SQL-client module definition> in which the corresponding SQL statement appears.*

In the Format, replace the production for <drop transform statement> with:

```
<drop transform statement> ::=
     DROP { TRANSFORM | TRANSFORMS } <transforms to be dropped>
        FOR <user-defined type name> <drop behavior>
```

12.2 <grant privilege statement>

1. *Rationale: Correct the references to <grant statement>.*

Replace Syntax Rule 5) with:

5) If the <grant privilege statement> is not contained in a <schema definition>, then the schema identified by the explicit or implicit qualifier of the <object name> shall include the descriptor of *O*. If <grant privilege statement> is contained in a <schema definition> S, then the schema identified by the

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

explicit or implicit qualifier of the <object name> shall include the descriptor of *O* or *S* shall include a <schema element> that creates the descriptor of *O*.

2. *Rationale: Editorial.*

Replace General Rule 7) with:

7) Let *SWH* be the set of privilege descriptors in *CPD* whose action is SELECT WITH HIERARCHY OPTION, and let *ST* be the set of subtables of *O*, then for every grantee *G* in *SWH* and for every table *T* in *ST*, the following <grant statement> is effectively executed without further Access Rule checking:

GRANT SELECT ON *T* TO *G* GRANTED BY *A*

3. *Rationale: Correct the references to <grant statement>.*

Replace Conformance Rule 1) with:

1) Without Feature S024, " Enhanced structured types", a <specific routine designator> contained in a <grant privilege statement> shall not identify a method.

12.4 <select statement: single row>

1. *Rationale: Replace incorrect non-terminal.*

Replace Syntax Rule 1) with:

1) Insert after SR4) For each <target specification> *TS* that is an <SQL variable reference>, the Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, apply to *TS* and the corresponding element of the <select list>, as *TARGET* and *VALUE*, respectively.

Replace General Rule 1) with:

1) Insert after GR5) For each <target specification> *TS* that is an <SQL variable reference>, the corresponding value in the row of *Q* is assigned to *TS* according to the General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, as *VALUE* and *TARGET*, respectively. The assignment of values to targets in the <select target list> is in an implementation-dependent order.

12.6 <revoke statement>

1. *Rationale: Correct the references to schema which do not treat it as a descriptor.*

Replace Syntax Rule 30) with:

30) *S1* is said to be *lost* if the revoke destruction action would result in *A1* no longer having in its applicable privileges USAGE privilege on the default character set included in *S1*.

Replace General Rule 10) with:

10) For every lost schema *S1*, the default character set of that schema is modified to include the name of the implementation-defined <character set specification> that would have been this schema's default character set had the <schema definition> not specified a <schema character set specification>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

13.1 <SQL-client module definition>

1. Rationale: Editorial - supply missing ellipsis.

In the Format, replace the production <SQL-client module definition> with:

```
<SQL-client module definition> ::=
     <module name clause>
     <language clause>
     <module authorization clause>
     [ <module path specification> ]
     [ <module transform group specification> ]
     [ <temporary table declaration>... ]
     <module contents>...
```

13.2 <module name clause>

1. Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".

Replace Conformance Rule 1) with:

1) Without Feature F461, "Named character sets", <module character set specification> shall not be specified.

13.3 <externally-invoked procedure>

1. Rationale: Complete removal of deprecated feature.

In the Format, replace the production for <host parameter declaration setup> with:

```
<host parameter declaration setup> ::= <host parameter declaration list>
```

Delete NOTE 274.

13.5 <SQL procedure statement>

1. Rationale: Remove redundant item. <grant role statement> is defined in <grant statement>.

In the Format, replace the production for <SQL schema definition statement> with:

```
<SQL schema definition statement> ::=
       <schema definition>
     | <table definition>
     | <view definition>
     | <SQL-invoked routine>
     | <grant statement>
     | <role definition>
     | <domain definition>
     | <character set definition>
     | <collation definition>
     | <translation definition>
     | <assertion definition>
     | <trigger definition>
     | <user-defined type definition>
     | <user-defined cast definition>
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
| <user-defined ordering definition>
| <transform definition>
```

2. *Rationale: Clarify the semantics of SQL-data access indication.*

Delete Syntax Rule 4).

Delete Syntax Rule 5).

Delete Syntax Rule 6).

3. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".*

Replace Conformance Rule 3) and 10) with:

3) Without Feature F451, "Character set definition", an <SQL schema definition statement> shall not be a <character set definition>.

10) Without Feature F451, "Character set definition", an <SQL schema definition statement> shall not be a <drop character set statement>.

14.1 <declare cursor>

1. *Rationale: Eliminate redundant <collate clause>.*

In the Format, replace the production for <sort specification> with:

```
<sort specification> ::= <sort key> [ <ordering specification> ]
```

2. *Rationale: Correct non-terminals.*

Replace Syntax Rules 7), 8) and 9) with:

7) If <cursor scrollability> is not specified, then NO SCROLL is implicit.

8) If <cursor holdability> is not specified, then WITHOUT HOLD is implicit.

9) If <cursor returnability> is not specified, then WITHOUT RETURN is implicit.

3. *Rationale: Prohibit ordering on data types that do not support ordering. Provide consistent Conformance Rules for comparison operations.*

Replace Syntax Rule 18) c) with:

18) c) *DT* shall not be LOB-ordered, array-ordered, UDT-EC-ordered, UDT-NC-ordered, or reference-ordered.

4. *Rationale: A <sort key> of character type need not be a column reference.*

Replace Syntax Rule 19) with:

19) If a <sort specification> contains a <collate clause>, then the declared type of the <value expression> contained in the <sort specification> shall be character string. The column descriptor of the

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

corresponding column in the result has the collating sequence specified in <collate clause> and the coercibility characteristic Explicit.

5. *Rationale: Editorial.*

Replace Syntax Rule 20) with:

20) If an <updatability clause> of FOR UPDATE without a <column name list> is specified or implicit, then a <column name list> that consists of the <column name> of every column of *LUT* is implicit.

6. *Rationale: Syntax Rule 22) states the wrong restriction on <sort key>s of user-defined type. The correct restriction is already present in Syntax Rule 18)c).*

Delete Syntax Rule 22).

7. *Rationale: Clarify that nulls sort as if they were equal.*

Replace General Rule 2) c) with:

2) c) Let *P* be any row of *TS* and let *Q* be any other row of *TS*, and let PV_i and QV_i be the values of K_i in these rows, respectively. The relative position of rows *P* and *Q* in the result is determined by comparing PV_i and QV_i according to the rules of Subclause 8.2, "<comparison predicate>", where the <comp op> is the applicable <comp op> for K_i, with the following special treatment of null values. A sort key value that is null is considered equal to another sort key value that is null. Whether a sort key value that is null is considered greater or less than a non-null value is implementation-defined, but all sort key values that are null shall either be considered greater than all non-null values or be considered less than all non-null values. PV_i is said to precede QV_i if the value of the <comparison predicate> "PV_i <comp op> QV_i" is true for the applicable <comp op>. If PV_i and QV_i are not null and the result of "PV_i <comp op> QV_i" is *unknown*, then the relative ordering of PV_i and QV_i is implementation-dependent.

8. *Rationale: Use the final <sort specification>.*

Replace General Rule 2) d) with:

2) d) In *TS*, the relative position of row *P* is before row *Q* if PV_n precedes QV_n for some *n* greater than 0 (zero) and less than or equal to the number of <sort specification>s and $PV_i = QV_i$ for all $i < n$. The relative order of two rows that are not distinct with respect to the <sort specification>s are implementation-dependent.

9. *Rationale: Provide consistent Conformance Rules for comparison operations.*

Replace Conformance Rule 9) with:

9) Without Feature S024, "Enhanced structured types", a <value expression> that is a <sort key> shall not be of an ST-ordered declared type.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

14.3 <fetch statement>

1. Rationale: Correct use of non-terminals.

Replace Syntax Rule 6) a) i) with:

6) i) If *TS* is an <SQL parameter reference>, then the Syntax Rules of Subclause 9.2, "Store assignment", apply to *TS* and the row type of table *T* as *TARGET* and *VALUE*, respectively.

Replace Syntax Rule 6) b) ii) with:

6) ii) For each <target specification> *TS1* that is an <SQL parameter reference>, the Syntax Rules of Subclause 9.2, "Store assignment", apply to *TS1* and the corresponding column of table *T* as *TARGET* and *VALUE*, respectively.

Replace Syntax Rule 6) b) iii) with:

6) iii) For each <target specification> *TS2* that is a <host parameter specification>, the Syntax Rules of Subclause 9.1, "Retrieval assignment", apply to each *TS2* and the corresponding column of table *T*, as *TARGET* and *VALUE*, respectively.

Replace General Rule 7) a) i) with:

7) i) If *TS* is an <SQL parameter reference>, then the General Rules of Subclause 9.2, "Store assignment", apply to *TS* and the current row as *TARGET* and *VALUE*, respectively.

Replace General Rule 7) b) i) with:

7) i) If *TV* is an <SQL parameter reference>, then the General Rules of Subclause 9.2, "Store assignment", apply to *TS* and *SV* as *TARGET* and *VALUE*, respectively.

2. Rationale: Add missing Syntax and General Rules for <fetch statement>.

Replace General rule 7) b) ii) with:

7) b) ii) If *TV* is a <host parameter specification>, then the General Rules of Subclause 9.1, "Retrieval assignment" are applied to *TV* and *SV*, as *TARGET* and *VALUE*, respectively.

14.5 <select statement: single row>

1. Rationale: Correct use of non-terminals.

Replace Syntax Rules 3) and 4) with:

3) For each <target specification> *TS* that is an <SQL parameter reference>, the Syntax Rules of Subclause 9.2, "Store assignment", apply to *TS* and the corresponding element of the <select list>, as *TARGET* and *VALUE*, respectively.

4) For each <target specification> *TS* that is a <host parameter specification>, the Syntax Rules of Subclause 9.1, "Retrieval assignment", apply to *TS* and the corresponding element of the <select list>, as *TARGET* and *VALUE*, respectively.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

2. *Rationale: Supply the correct definition for non-determinism.*

Delete Syntax Rule 5).

Add the following Syntax Rule:

6.1) The <select statement: single row> is *possibly non-deterministic* if *S* is possibly non-deterministic.

3. *Rationale: Correct use of non-terminals.*

Replace Syntax Rules 4) and 5) with:

4) For each <target specification> *TS* that is an <SQL parameter reference>, the corresponding value in the row of *Q* is assigned to *TS* according to the General Rules of Subclause 9.2, "Store assignment", as *VALUE* and *TARGET*, respectively. The assignment of values to targets in the <select target list> is in an implementation-dependent order.

5) For each <target specification> *TS* that is a <host parameter specification>, the corresponding value in the row of *Q* is assigned to *TS* according to the General Rules of Subclause 9.1, "Retrieval assignment", as *VALUE* and *TARGET*, respectively. The assignment of values to targets in the <select target list> is in an implementation-dependent order.

14.6 <delete statement: positioned>

1. *Rationale: If ONLY is specified, then parentheses in <target table> are required, otherwise they are optional.*

In the Format, replace the production for <target table> with:

```
<target table> ::=
      <table name>
    | ONLY <left paren> <table name> <right paren>
```

2. *Rationale: Use the notion of identical to make specification of effect of <delete statement: positioned> more precise.*

Replace General Rule 7) with:

7) Let *R* be the current row of *CR*. Exactly one row *R1* in *LUT* such that each field in *R* is identical to the corresponding field in *R1* is identified for deletion from *LUT*.

3. *Rationale: Use of ONLY requires Feature S111, "ONLY in query expressions".*

Add the following Conformance Rule:

1) Without Feature S111, "ONLY in query expressions", a <target table> shall not specify ONLY.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0860049 969

14.8 <insert statement>

1. Rationale: Clean up typed table insertability property for non-instantiable types.

Delete Syntax Rule 3)

2. Rationale: Remove a syntactic ambiguity, that VALUES (1) may be parsed either as <insert columns and source> ::= <from subquery> ::= <query expression> ::= <table value constructor> ::= VALUES (1) or <insert columns and source> ::= <from constructor> ::= <contextually typed table value constructor> ::= VALUES (1).

Add the following Syntax Rule:

8.1) A <query expression> simply contained in a <from subquery> shall not be a <table value constructor>.

3. Rationale: Use the correct non-terminal symbols.

Replace Syntax Rule 9) with:

9) An <insert columns and source> that specifies DEFAULT VALUES is implicitly replaced by an <insert columns and source> that specifies a <contextually typed table value constructor> of the form
`VALUES (DEFAULT, DEFAULT, . . . , DEFAULT)`
where the number of "DEFAULT" entries is equal to the number of columns of T.

4. Rationale: Cater for self-referencing column in underlying table of T.

Replace General Rules 7) b), c) and d) with:

7) b) If *T* has a column *RC* of which some underlying column is a self-referencing column, then

Case:

i) If *RC* is a system-generated self-referencing column, then the value of *RC* is effectively replaced by the REF value of the candidate row.

ii) If *RC* is a derived self-referencing column, then the value of *RC* is effectively replaced by a value derived from the columns in the candidate row that correspond to the list of attributes of the derived representation of the reference type of *RC* in an implementation-dependent manner.

c) For each object column in the candidate row, let C_i be the object column identified by the *i*-th <column name> in the <insert column list> and let $_{SVi}$ be the *i*-th value of *R*.

d) For every C_i for which one of the following conditions is true:

i) C_i is not a self-referencing column of *T;*

ii) C_i is a user-generated self-referencing column of *T;*

iii) C_i is a self-referencing column of *T* and OVERRIDING SYSTEM VALUE is specified;

the General Rules of Subclause 9.2, "Store assignment", are applied to C_i and $_{SVi}$ as *TARGET* and *SOURCE*, respectively.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

14.9 <update statement: positioned>

1. Rationale: Remove undefined SET ROW functionality.

In the Format, replace the production for <update target> with:

```
<update target> ::=
      <object column>
    | <object column>
        <left bracket or trigraph>
        <simple value specification>
        <right bracket or trigraph>
```

2. Rationale: Remove undefined SET ROW functionality.

Delete Syntax Rule 9).

3. Rationale: Remove undefined SET ROW functionality.

Replace General Rule 15) with:

15) Let *CL* be the columns of *T* identified by the <object columns> contained in the <set clause list>.

4. Rationale: Use the notion of identical to make specification of <update statement: positioned> more precise.

Replace General Rule 16) with:

16) Let *RI* be the candidate new row. The current row *R* of *CR* is replaced by *RI*. Exactly one row *TR* in *T* such that each field in *R* is identical to the corresponding field in *TR* is identified for replacement in *T*. Let *TRI* be a row consisting of the fields of *RI* and the fields of *TR* that have no corresponding fields in *RI*, ordered according to the order of their corresponding columns in *T*. *TRI* is the replacement row for *TR* and { (*TR*, *TRI*) } is the replacement set for *T*.

5. Rationale: Remove undefined SET ROW functionality.

Delete Conformance Rule 3).

14.10 <update statement: searched>

1. Rationale: Remove undefined SET ROW functionality.

Delete Syntax Rule 4).

2. Rationale: Remove undefined SET ROW functionality.

Replace General Rule 13) with:

13) Let *CL* be the columns of *T* identified by the <object columns> contained in the <set clause list>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

14.11<temporary table declaration>

1. Rationale: Correct privileges for <temporary tables>s.

Replace General Rule 4) with:

4) A set of privilege descriptors is created that define the privileges INSERT, SELECT, UPDATE, DELETE, and REFERENCES on this table and INSERT, SELECT, UPDATE, and REFERENCES for every <column definition> in the table definition to *A*. These privileges are not grantable. The grantee is "PUBLIC".

14.18 Effect of inserting a table into a derived table

1. Rationale: Editorial.

Replace General Rule 2) b) iii) with:

2) b) iii) The candidate rows are added to the corresponding S_i.

15.2 <return statement>

1. Rationale: Move the check for the most specific type of the returned value to <routine invocation>.

Replace General Rules 1) and 2) with:

1) Let *RV* be the value of *VE*. Let *RT* be an item of data type *RDT*, arbitrarily chosen.

16.4 <savepoint statement>

1. Rationale: Remove the anomaly in <savepoint specifier>.

In the Format, replace the production for <savepoint specifier> with:

```
<savepoint specifier> ::= <savepoint name>
```

Delete Syntax Rule 1)

Replace General Rules 1) and 2) with:

1) Let *S* be the <savepoint name>.

Delete General Rule 5)

16.5 <release savepoint statement>

1. Rationale: Remove the anomaly in <savepoint specifier>.

Delete Syntax Rule 1)

Replace General Rule 1) with:

1) Let *S* be the <savepoint name>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

16.7 <rollback statement>

1. Rationale: Remove the anomaly in <savepoint specifier>.

Delete Syntax Rule 1)

Replace General Rule 3) a) with:

3) a) Let *S* be the <savepoint name>.

2. Rationale: Correct the specification of which locators are marked invalid when an SQL-transaction ends.

Replace General Rule 3) f) with:

3) f) Every valid locator generated in the current SQL-transaction subsequent to the establishment of *S* is marked invalid.

19.1 <get diagnostics statement>

1. Rationale: Editorial. Non-reserved word - DYNAMIC_FUNCTION - not defined. Only defined in Part 5.

Replace General Rule 3) p) with:

3) p) The values of CONNECTION_NAME and SERVER_NAME are respectively

Case:

i) If COMMAND_FUNCTION identifies an <SQL connection statement>, then the <connection name> and the <SQL-server name> specified by or implied by the <SQL connection statement>.

ii) Otherwise, the <connection name> and <SQL-server name> of the SQL-session in which the condition was raised.

2. Rationale: Correct the classification of SQL-statements.

Replace the following row in Table 26, "SQL-statement codes":

SQL-statement	Identifier	Code
<grant statement>	GRANT	48

with:

SQL-statement	Identifier	Code
<grant privilege statement>	GRANT	48

3. Rationale: Correct the classification of SQL-statements.

Add the following rows to Table 26, "SQL-statement codes":

SQL-statement	Identifier	Code
<user-defined cast definition>	CREATE CAST	52

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

<drop-user-defined cast definition>	DROP CAST	78
<temporary table declaration>	TEMPORARY TABLE	93

20.11 ATTRIBUTES view

1. Rationale: Add missing column to the definition of the ATTRIBUTES view.

In the Definition, replace the view definition with:

```
CREATE VIEW ATTRIBUTES AS
SELECT DISTINCT
       UDT_CATALOG, UDT_SCHEMA, UDT_NAME, A.ATTRIBUTE_NAME, ORDINAL_POSITION,
       CASE WHEN EXISTS ( SELECT *
                            FROM DEFINITION_SCHEMA.SCHEMATA AS S
                            WHERE ( UDT_CATALOG, UDT_SCHEMA )
                                    = ( S.CATALOG_NAME, S.SCHEMA_NAME )
                                  AND
                                  ( SCHEMA_OWNER IN ( 'PUBLIC', CURRENT_USER )
                                    OR
                                    SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                                        FROM ENABLED_ROLES ) ) )
              THEN ATTRIBUTE_DEFAULT
            ELSE NULL
         END AS ATTRIBUTE_DEFAULT,
       IS_NULLABLE, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH,
       CHARACTER_OCTET_LENGTH, C1.CHARACTER_SET_CATALOG,
       C1.CHARACTER_SET_SCHEMA, C1.CHARACTER_SET_NAME,
       D1.COLLATION_CATALOG, D1.COLLATION_SCHEMA, D1.COLLATION_NAME,
       NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
       DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
       D1.USER_DEFINED_TYPE_CATALOG AS ATTRIBUTE_UDT_CATALOG,
       D1.USER_DEFINED_TYPE_SCHEMA AS ATTRIBUTE_UDT_SCHEMA,
       D1.USER_DEFINED_TYPE_NAME AS ATTRIBUTE_UDT_NAME,
       D1.SCOPE_CATALOG, D1.SCOPE_SCHEMA, D1.SCOPE_NAME,
       MAXIMUM_CARDINALITY, A.DTD_IDENTIFIER, CHECK_REFERENCES,
       IS_DERIVED_REFERENCE_ATTRIBUTE
  FROM ( DEFINITION_SCHEMA.ATTRIBUTES AS A
         LEFT JOIN ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D1
                     LEFT JOIN DEFINITION_SCHEMA.COLLATIONS AS C1
                       ON ( ( C1.COLLATION_CATALOG, C1.COLLATION_SCHEMA,
                              C1.COLLATION_NAME )
                              = ( D1.COLLATION_CATALOG, D1.COLLATION_SCHEMA,
                                  D1.COLLATION_NAME ) ) )
           ON ( ( A.UDT_CATALOG, A.UDT_SCHEMA, A.UDT_NAME,
                  'USER-DEFINED TYPE', A.DTD_IDENTIFIER )
                = ( D1.OBJECT_CATALOG, D1.OBJECT_SCHEMA, D1.OBJECT_NAME,
                    D1.OBJECT_TYPE, D1.DTD_IDENTIFIER ) ) )
 WHERE ( A.UDT_CATALOG, A.UDT_SCHEMA, A.UDT_NAME )
         IN ( SELECT UDT_CATALOG, UDT_SCHEMA, UDT_NAME
                FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES
               WHERE ( SCHEMA_OWNER IN ( 'PUBLIC', CURRENT_USER )
                       OR
                       SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                           FROM ENABLED_ROLES ) ) )
       AND
       A.UDT_CATALOG = ( SELECT CATALOG_NAME
                           FROM INFORMATION_SCHEMA_CATALOG_NAME );
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

20.18 COLUMNS view

1. Rationale: Editorial - typographical errors.

```
CREATE VIEW COLUMNS AS
SELECT DISTINCT
       TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
       C.COLUMN_NAME, ORDINAL_POSITION,
       CASE WHEN EXISTS ( SELECT *
                            FROM DEFINITION_SCHEMA.SCHEMATA AS S
                           WHERE ( TABLE_CATALOG, TABLE_SCHEMA )
                                   = (S.CATALOG_NAME, S.SCHEMA_NAME )
                                 AND
                                 ( SCHEMA_OWNER IN ( 'PUBLIC', CURRENT_USER )
                                   OR
                                   SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                                       FROM ENABLED_ROLES ) ) )
              THEN COLUMN_DEFAULT
            ELSE NULL
         END AS COLUMN_DEFAULT,
       IS_NULLABLE,
       COALESCE ( D1.DATA_TYPE, D2.DATA_TYPE ) AS DATA_TYPE,
       COALESCE ( D1.CHARACTER_MAXIMUM_LENGTH, D2.CHARACTER_MAXIMUM_LENGTH )
         AS CHARACTER_MAXIMUM_LENGTH,
       COALESCE ( D1.CHARACTER_OCTET_LENGTH, D2.CHARACTER_OCTET_LENGTH )
         AS CHARACTER_OCTET_LENGTH,
       COALESCE ( D1.NUMERIC_PRECISION, D2.NUMERIC_PRECISION )
         AS NUMERIC_PRECISION,
       COALESCE ( D1.NUMERIC_PRECISION_RADIX, D2.NUMERIC_PRECISION_RADIX )
         AS NUMERIC_PRECISION_RADIX,
       COALESCE ( D1.NUMERIC_SCALE, D2.NUMERIC_SCALE )
         AS NUMERIC_SCALE,
       COALESCE ( D1.DATETIME_PRECISION, D2.DATETIME_PRECISION )
         AS DATETIME_PRECISION,
       COALESCE ( D1.INTERVAL_TYPE, D2.INTERVAL_TYPE )
         AS INTERVAL_TYPE,
       COALESCE ( D1.INTERVAL_PRECISION, D2.INTERVAL_PRECISION )
         AS INTERVAL_PRECISION,
       COALESCE ( C1.CHARACTER_SET_CATALOG, C2.CHARACTER_SET_CATALOG )
         AS CHARACTER_SET_CATALOG,
       COALESCE ( C1.CHARACTER_SET_SCHEMA, C2.CHARACTER_SET_SCHEMA )
         AS CHARACTER_SET_SCHEMA,
       COALESCE ( C1.CHARACTER_SET_NAME, C2.CHARACTER_SET_NAME )
         AS CHARACTER_SET_NAME,
       COALESCE ( D1.COLLATION_CATALOG, D2.COLLATION_CATALOG )
         AS COLLATION_CATALOG,
       COALESCE ( D1.COLLATION_SCHEMA, D2.COLLATION_SCHEMA )
         AS COLLATION_SCHEMA,
       COALESCE ( D1.COLLATION_NAME, D2.COLLATION_NAME )
         AS COLLATION_NAME,
       DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
       COALESCE ( D1.USER_DEFINED_TYPE_CATALOG, D2.USER_DEFINED_TYPE_CATALOG )
         AS UDT_CATALOG,
       COALESCE ( D1.USER_DEFINED_TYPE_SCHEMA, D2.USER_DEFINED_TYPE_SCHEMA )
         AS UDT_SCHEMA,
       COALESCE ( D1.USER_DEFINED_TYPE_NAME, D2.USER_DEFINED_TYPE_NAME )
         AS UDT_NAME,
       COALESCE ( D1.SCOPE_CATALOG, D2.SCOPE_CATALOG ) AS SCOPE_CATALOG,
       COALESCE ( D1.SCOPE_SCHEMA, D2.SCOPE_SCHEMA ) AS SCOPE_SCHEMA,
       COALESCE ( D1.SCOPE_NAME, D2.SCOPE_NAME ) AS SCOPE_NAME,
       COALESCE ( D1.MAXIMUM_CARDINALITY, D2.MAXIMUM_CARDINALITY )
         AS MAXIMUM_CARDINALITY,
       COALESCE ( D1.DTD_IDENTIFIER, D2.DTD_IDENTIFIER ) AS DTD_IDENTIFIER,
       IS_SELF_REFERENCING
  FROM ( DEFINITION_SCHEMA.COLUMNS AS C
         LEFT JOIN ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D1
                     LEFT JOIN DEFINITION_SCHEMA.COLLATIONS AS C1
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
                    ON ( ( C1.COLLATION_CATALOG, C1.COLLATION_SCHEMA,
                           C1.COLLATION_NAME )
                           = ( D1.COLLATION_CATALOG, D1.COLLATION_SCHEMA,
                               D1.COLLATION_NAME ) ) )
          ON ( ( C.TABLE_CATALOG, C.TABLE_SCHEMA, C.TABLE_NAME,
                 'TABLE', C.DTD_IDENTIFIER )
               = ( D1.OBJECT_CATALOG, D1.OBJECT_SCHEMA, D1.OBJECT_NAME,
                   D1.OBJECT_TYPE, D1.DTD_IDENTIFIER ) ) )
      LEFT JOIN ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D2
                  LEFT JOIN DEFINITION_SCHEMA.COLLATIONS AS C2
                    ON ( ( C2.COLLATION_CATALOG, C2.COLLATION_SCHEMA,
                           C2.COLLATION_NAME )
                           = ( D2.COLLATION_CATALOG, D2.COLLATION_SCHEMA,
                               D2.COLLATION_NAME ) ) )
        ON ( ( C.DOMAIN_CATALOG, C.DOMAIN_SCHEMA, C.DOMAIN_NAME,
               'DOMAIN', C.DTD_IDENTIFIER )
             = ( D2.OBJECT_CATALOG, D2.OBJECT_SCHEMA, D2.OBJECT_NAME,
                 D2.OBJECT_TYPE, D2.DTD_IDENTIFIER ) )
WHERE ( C.TABLE_CATALOG, C.TABLE_SCHEMA, C.TABLE_NAME, C.COLUMN_NAME )
       IN ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
              FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES
             WHERE ( SCHEMA_OWNER IN ( 'PUBLIC', CURRENT_USER )
                     OR
                     SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                         FROM ENABLED_ROLES ) ) )
      AND
      C.TABLE_CATALOG = ( SELECT CATALOG_NAME
                            FROM INFORMATION_SCHEMA_CATALOG_NAME );
```

20.19 CONSTRAINT_COLUMN_USAGE view

1. Rationale: Editorial - typographical errors.

In the Definition, replace the <from clause> with:

```
FROM ( ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,
                CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
           FROM DEFINITION_SCHEMA.CHECK_COLUMN_USAGE )
         UNION
         ( SELECT PK.TABLE_CATALOG, PK.TABLE_SCHEMA, PK.TABLE_NAME,
                  PK.COLUMN_NAME,
                  FK.CONSTRAINT_CATALOG, FK.CONSTRAINT_SCHEMA, FK.CONSTRAINT_NAME
             FROM DEFINITION_SCHEMA.REFERENTIAL_CONSTRAINTS AS FK
                  JOIN DEFINITION_SCHEMA.KEY_COLUMN_USAGE AS PK
                    ON ( FK.UNIQUE_CONSTRAINT_CATALOG,
                         FK.UNIQUE_CONSTRAINT_SCHEMA,
                         FK.UNIQUE_CONSTRAINT_NAME )
                       = ( PK.CONSTRAINT_CATALOG, PK.CONSTRAINT_SCHEMA,
                           PK.CONSTRAINT_NAME ) )
         UNION
         ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,
                  CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
             FROM DEFINITION_SCHEMA.KEY_COLUMN_USAGE
                  NATURAL JOIN         DEFINITION_SCHEMA.TABLE_CONSTRAINTS
            WHERE CONSTRAINT_TYPE IN ( 'UNIQUE', 'PRIMARY KEY' ) ) )
       JOIN DEFINITION_SCHEMA.SCHEMATA
              ON ( ( TABLE_CATALOG, TABLE_SCHEMA )
                   = ( CATALOG_NAME, SCHEMA_NAME ) )
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4851903 0860056 0T9

20.21 DATA_TYPE_PRIVILEGES view

1. Rationale: Editorial - typographical error and missing grant statement.

Replace the Definition with:

```
CREATE VIEW DATA_TYPE_PRIVILEGES
( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
  OBJECT_TYPE, DTD_IDENTIFIER )
AS
SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
       USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE', DTD_IDENTIFIER
  FROM ATTRIBUTES
UNION
SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
       'TABLE', DTD_IDENTIFIER
  FROM COLUMNS
UNION
SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
       'DOMAIN', DTD_IDENTIFIER
  FROM DOMAINS
UNION
SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
       USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE', DTD_IDENTIFIER
  FROM METHOD_SPECIFICATIONS
UNION
SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
       USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE', DTD_IDENTIFIER
  FROM METHOD_SPECIFICATION_PARAMETERS
UNION
SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
       'ROUTINE', DTD_IDENTIFIER
  FROM PARAMETERS
UNION
SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
       'ROUTINE', DTD_IDENTIFIER
  FROM ROUTINES
UNION
SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
       USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE', SOURCE_DTD_IDENTIFIER
  FROM USER_DEFINED_TYPES
UNION
SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
       USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE', REF_DTD_IDENTIFIER
  FROM USER_DEFINED_TYPES;

GRANT SELECT ON TABLE DATA_TYPE_PRIVILEGES
  TO PUBLIC WITH GRANT OPTION;
```

20.26 DOMAINS view

1. Rationale: Editorial - typographical error.

In the Definition, replace the <where clause> with:

```
WHERE ( ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME, 'DOMAIN' )
        IN ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE
               FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES
              WHERE ( SCHEMA_OWNER IN ( 'PUBLIC', CURRENT_USER )
                      OR
                      SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                          FROM ENABLED_ROLES ) ) )
        OR
        ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME )
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
        IN ( SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME
               FROM COLUMNS ) )
    AND
    DOMAIN_CATALOG = ( SELECT CATALOG_NAME
                         FROM INFORMATION_SCHEMA_CATALOG_NAME );
```

20.27 ELEMENT_TYPES view

1. Rationale: Remove irrelevant column from the definition of the ELEMENT_TYPES view.

In the Definition, replace the view definition with:

```
CREATE VIEW ELEMENT_TYPES AS
  SELECT DISTINCT
         E.OBJECT_CATALOG, E.OBJECT_SCHEMA, E.OBJECT_NAME,
         E.OBJECT_TYPE, ARRAY_TYPE_IDENTIFIER, DATA_TYPE,
         CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
         CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
         COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
         NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
         DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
         USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
         USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
         USER_DEFINED_TYPE_NAME AS UDT_NAME,
         SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
         MAXIMUM_CARDINALITY, E.DTD_IDENTIFIER
    FROM DEFINITION_SCHEMA.ELEMENT_TYPES AS E
         JOIN ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D
                LEFT JOIN DEFINITION_SCHEMA.COLLATIONS AS S
                  USING ( COLLATION_CATALOG, COLLATION_SCHEMA,
                          COLLATION_NAME ) )
           ON ( ( E.OBJECT_CATALOG, E.OBJECT_SCHEMA, E.OBJECT_NAME,
                  E.OBJECT_TYPE, E.DTD_IDENTIFIER )
                  = ( D.OBJECT_CATALOG, D.OBJECT_SCHEMA, D.OBJECT_NAME,
                      D.OBJECT_TYPE, D.DTD_IDENTIFIER ) )
   WHERE ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
           OBJECT_TYPE, ROOT_DTD_IDENTIFIER )
           IN ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                       OBJECT_TYPE, DTD_IDENTIFIER
                  FROM INFORMATION_SCHEMA.DATA_TYPE_PRIVILEGES );
```

20.29 FIELDS view

1. Rationale: Remove irrelevant columns from the definition of the FIELDS view.

In the Definition, replace the view definition with:

```
CREATE VIEW FIELDS AS
SELECT DISTINCT
       F.OBJECT_CATALOG, F.OBJECT_SCHEMA, F.OBJECT_NAME,
       F.OBJECT_TYPE, ROW_IDENTIFIER, FIELD_NAME,
       ORDINAL_POSITION, DATA_TYPE,
       CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
       CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
       COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
       NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
       DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
       USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
       USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
       USER_DEFINED_TYPE_NAME AS UDT_NAME,
       SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
       MAXIMUM_CARDINALITY, F.DTD_IDENTIFIER
  FROM DEFINITION_SCHEMA.FIELDS AS F
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
      JOIN ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D
             LEFT JOIN DEFINITION_SCHEMA.COLLATIONS AS S
               USING ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) )
        ON ( ( F.OBJECT_CATALOG, F.OBJECT_SCHEMA, F.OBJECT_NAME,
               F.OBJECT_TYPE, F.DTD_IDENTIFIER )
               = ( D.OBJECT_CATALOG, D.OBJECT_SCHEMA, D.OBJECT_NAME,
                   D.OBJECT_TYPE, D.DTD_IDENTIFIER ) )
WHERE ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
        OBJECT_TYPE, ROOT_DTD_IDENTIFIER )
        IN ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                    OBJECT_TYPE, DTD_IDENTIFIER
               FROM INFORMATION_SCHEMA.DATA_TYPE_PRIVILEGES );
```

20.31 METHOD_SPECIFICATION_PARAMETERS view

1. Rationale: Editorial - typographical error and missing grant statement.

In the Definition, replace the <where clause> with:

```
WHERE ( M.USER_DEFINED_TYPE_CATALOG, M.USER_DEFINED_TYPE_SCHEMA,
        M.USER_DEFINED_TYPE_NAME )
       IN ( SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                   USER_DEFINED_TYPE_NAME
              FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES
             WHERE ( SCHEMA_OWNER IN ( 'PUBLIC', CURRENT_USER )
                     OR
                     SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                         FROM ENABLED_ROLES ) ) )
      AND
      M.USER_DEFINED_TYPE_CATALOG = ( SELECT CATALOG_NAME
                                        FROM INFORMATION_SCHEMA_CATALOG_NAME );
```

In the Definition, replace the <grant statement> with:

```
GRANT SELECT ON TABLE METHOD_SPECIFICATION_PARAMETERS
  TO PUBLIC WITH GRANT OPTION;
```

20.32 METHOD_SPECIFICATIONS view

1. Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.

In the Definition, replace the view definition with:

```
CREATE VIEW METHOD_SPECIFICATIONS AS
  SELECT M.SPECIFIC_CATALOG, M.SPECIFIC_SCHEMA, M.SPECIFIC_NAME,
         M.USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
         M.USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
         M.USER_DEFINED_TYPE_NAME AS UDT_NAME,
         M.METHOD_NAME, IS_STATIC, IS_OVERRIDING,
         IS_CONSTRUCTOR,
         D.DATA_TYPE, D.CHARACTER_MAXIMUM_LENGTH, D.CHARACTER_OCTET_LENGTH,
         C.CHARACTER_SET_CATALOG, C.CHARACTER_SET_SCHEMA,
         C.CHARACTER_SET_NAME,
         D.COLLATION_CATALOG, D.COLLATION_SCHEMA, D.COLLATION_NAME,
         D.NUMERIC_PRECISION, D.NUMERIC_PRECISION_RADIX, D.NUMERIC_SCALE,
         D.DATETIME_PRECISION, D.INTERVAL_TYPE, D.INTERVAL_PRECISION,
         D.USER_DEFINED_TYPE_CATALOG AS RETURN_UDT_CATALOG,
         D.USER_DEFINED_TYPE_SCHEMA AS RETURN_UDT_SCHEMA,
         D.USER_DEFINED_TYPE_NAME AS RETURN_UDT_NAME,
         D.SCOPE_CATALOG, D.SCOPE_SCHEMA, D.SCOPE_NAME,
         D.MAXIMUM_CARDINALITY, D.DTD_IDENTIFIER, M.METHOD_LANGUAGE,
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
        M.PARAMETER_STYLE, M.IS_DETERMINISTIC, M.SQL_DATA_ACCESS,
        M.IS_NULL_CALL,
        M.TO_SQL_SPECIFIC_CATALOG, M.TO_SQL_SPECIFIC_SCHEMA,
        M.TO_SQL_SPECIFIC_NAME,
        M.CREATED, M.LAST_ALTERED
    FROM ( DEFINITION_SCHEMA.METHOD_SPECIFICATIONS M
           JOIN ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR D
                  LEFT JOIN DEFINITION_SCHEMA.COLLATIONS AS C
                    ON ( C.COLLATION_CATALOG, C.COLLATION_SCHEMA,
                         C.COLLATION_NAME )
                         = ( D.COLLATION_CATALOG, D.COLLATION_SCHEMA,
                             D.COLLATION_NAME ) )
             ON ( M.USER_DEFINED_TYPE_CATALOG, M.USER_DEFINED_TYPE_SCHEMA,
                  M.USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE',
                  DTD_IDENTIFIER )
                  = ( D.OBJECT_CATALOG, D.OBJECT_SCHEMA, D.OBJECT_NAME,
                      D.OBJECT_TYPE, D.DTD_IDENTIFIER ) )
    WHERE ( M.USER_DEFINED_TYPE_CATALOG, M.USER_DEFINED_TYPE_SCHEMA,
            M.USER_DEFINED_TYPE_NAME )
            IN ( SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                        USER_DEFINED_TYPE_NAME
                   FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES
                  WHERE ( SCHEMA_OWNER IN ( 'PUBLIC', CURRENT_USER )
                        OR
                        SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                            FROM ENABLED_ROLES ) ) )
          AND
          M.USER_DEFINED_TYPE_CATALOG
          = ( SELECT CATALOG_NAME
                FROM INFORMATION_SCHEMA_CATALOG_NAME );
```

20.33 PARAMETERS view

1. Rationale: Editorial - typographical error.

In the Definition, replace the <from clause> with:

```
FROM ( DEFINITION_SCHEMA.PARAMETERS P1
       LEFT JOIN     ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR D1
                       LEFT JOIN DEFINITION_SCHEMA.COLLATIONS C1
                         ON ( ( C1.COLLATION_CATALOG, C1.COLLATION_SCHEMA,
                                C1.COLLATION_NAME )
                              = ( D1.COLLATION_CATALOG, D1.COLLATION_SCHEMA,
                                  D1.COLLATION_NAME ) ) )
         ON ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
              'ROUTINE', P1.DTD_IDENTIFIER )
            = ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                OBJECT_TYPE, D1.DTD_IDENTIFIER ) )
       JOIN DEFINITION_SCHEMA.ROUTINES R1
         ON ( ( P1.SPECIFIC_CATALOG, P1.SPECIFIC_SCHEMA, P1.SPECIFIC_NAME )
              = ( R1.SPECIFIC_CATALOG, R1.SPECIFIC_SCHEMA, R1.SPECIFIC_NAME ) )
```

20.34 REFERENCED_TYPES view

1. Rationale: Remove irrelevant column from the definition of the REFERENCED_TYPES view.

In the Definition, replace the view definition with:

```
CREATE VIEW REFERENCED_TYPES AS
  SELECT DISTINCT
         R.OBJECT_CATALOG, R.OBJECT_SCHEMA, R.OBJECT_NAME,
         R.OBJECT_TYPE, REFERENCE_TYPE_IDENTIFIER, DATA_TYPE,
         CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
        CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
        COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
        NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
        DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
        USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
        USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
        USER_DEFINED_TYPE_NAME AS UDT_NAME,
        SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
        MAXIMUM_CARDINALITY, R.DTD_IDENTIFIER
    FROM ( DEFINITION_SCHEMA.REFERENCED_TYPES AS R
           JOIN ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D
                  LEFT JOIN DEFINITION_SCHEMA.COLLATIONS AS S
                    USING ( COLLATION_CATALOG, COLLATION_SCHEMA,
                            COLLATION_NAME ) )
             ON ( ( R.OBJECT_CATALOG, R.OBJECT_SCHEMA,
                    R.OBJECT_NAME,R.OBJECT_TYPE, R.DTD_IDENTIFIER )
                    = ( D.OBJECT_CATALOG, D.OBJECT_SCHEMA, D.OBJECT_NAME,
                        D.OBJECT_TYPE, D.DTD_IDENTIFIER ) ) )
    WHERE ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
            OBJECT_TYPE, ROOT_DTD_IDENTIFIER )
            IN ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                        OBJECT_TYPE, DTD_IDENTIFIER
                   FROM INFORMATION_SCHEMA.DATA_TYPE_PRIVILEGES );
```

20.38 ROLE_TABLE_GRANTS view

1. Rationale: Editorial - typographical error.

In the Definition, replace the <where clause> with:

```
WHERE ( GRANTEE IN ( SELECT ROLE_NAME
                       FROM ENABLED_ROLES )
        OR
        GRANTOR IN ( SELECT ROLE_NAME
                       FROM ENABLED_ROLES ) )
      AND TABLE_CATALOG = ( SELECT CATALOG_NAME
                              FROM INFORMATION_SCHEMA_CATALOG_NAME );
```

20.56 TABLES view

1. Rationale: Clean up typed table insertability property for non-instantiable types.

Replace the Definition with:

```
CREATE VIEW TABLES AS
  SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
         TABLE_TYPE, SELF_REFERENCING_COLUMN_NAME, REFERENCE_GENERATION,
         USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
         USER_DEFINED_TYPE_NAME, IS_INSERTABLE_INTO
    FROM DEFINITION_SCHEMA.TABLES
    WHERE ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
          IN ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                 FROM DEFINITION_SCHEMA.TABLE_PRIVILEGES
                WHERE GRANTEE IN ( 'PUBLIC', CURRENT_USER )
               UNION
               SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                 FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES
                WHERE ( SCHEMA_OWNER IN ( 'PUBLIC', CURRENT_USER )
                        OR
                        SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                            FROM ENABLED_ROLES ) ) )
          AND
          TABLE_CATALOG = ( SELECT CATALOG_NAME
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
                                FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE TABLES
  TO PUBLIC WITH GRANT OPTION;
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4851903 0860062 3T2

20.62 TRIGGERS view

1. *Rationale: Extend TRIGGERS view to allow for the representation of all possible elements of <old or new values alias list>, and delete stray reference to " CONDITION_TIMING," and editorially remove erroneous text.*

Replace the Definition with:

```
CREATE VIEW TRIGGERS AS
  SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
         EVENT_MANIPULATION,
         EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA, EVENT_OBJECT_TABLE,
         ACTION_ORDER, ACTION_CONDITION, ACTION_STATEMENT,
         ACTION_ORIENTATION, ACTION_TIMING,
         ACTION_REFERENCE_OLD_TABLE, ACTION_REFERENCE_NEW_TABLE,
         ACTION_REFERENCE_OLD_ROW, ACTION_REFERENCE_NEW_ROW, CREATED
    FROM DEFINITION_SCHEMA.TRIGGERS
         JOIN DEFINITION_SCHEMA.SCHEMATA S
           ON ( TRIGGER_CATALOG, TRIGGER_SCHEMA )
                = ( S.CATALOG_NAME, S.SCHEMA_NAME )
   WHERE ( SCHEMA_OWNER = CURRENT_USER
           OR
           SCHEMA_OWNER IN ( SELECT ROLE_NAME
                               FROM ENABLED_ROLES ) )
         AND
         TRIGGER_CATALOG = ( SELECT CATALOG_NAME
                               FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE TRIGGERS
  TO PUBLIC WITH GRANT OPTION;
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

20.68 VIEWS view

1. Rationale: Clean up typed table insertability property for non-instantiable types.

Replace the Definition with:

```
CREATE VIEW VIEWS AS
  SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
         CASE WHEN EXISTS ( SELECT *
                              FROM DEFINITION_SCHEMA.SCHEMATA AS S
                             WHERE ( TABLE_CATALOG, TABLE_SCHEMA )
                                     = ( S.CATALOG_NAME, S.SCHEMA_NAME )
                                   AND
                                   ( SCHEMA_OWNER = CURRENT_USER
                                     OR
                                     SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                                         FROM ENABLED_ROLES )
                                   ) )
                THEN VIEW_DEFINITION
              ELSE NULL
           END AS VIEW_DEFINITION,
         CHECK_OPTION, IS_UPDATABLE
    FROM DEFINITION_SCHEMA.VIEWS
   WHERE ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
           IN ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                  FROM TABLES )
         AND
         TABLE_CATALOG = ( SELECT CATALOG_NAME
                             FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE VIEWS
  TO PUBLIC WITH GRANT OPTION;
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

20.69 Short name views

1. Rationale: Add missing column to the definition of the ATTRIBUTES view.

In the Definition, replace the view definition for ATTRIBUTES_S with:

```
CREATE VIEW ATTRIBUTES_S
  ( UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
    ATTRIBUTE_NAME, ORDINAL_POSITION, ATTRIBUTE_DEFAULT,
    IS_NULLABLE, DATA_TYPE, CHAR_MAX_LENGTH,
    CHAR_OCTET_LENGTH, CHAR_SET_CATALOG, CHAR_SET_SCHEMA,
    CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,
    COLLATION_NAME, NUMERIC_PRECISION, NUMERIC_PREC_RADIX,
    NUMERIC_SCALE, DATETIME_PRECISION, INTERVAL_TYPE,
    INTERVAL_PRECISION, DOMAIN_CATALOG, DOMAIN_SCHEMA,
    DOMAIN_NAME, ATT_UDT_CAT, ATT_UDT_SCHEMA,
    ATT_UDT_NAME, SCOPE_CATALOG, SCOPE_SCHEMA,
    SCOPE_NAME, MAX_CARDINALITY, DTD_IDENTIFIER,
    CHECK_REFERENCES, IS_DERIVED_REF_ATT ) AS
  SELECT UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
         ATTRIBUTE_NAME, ORDINAL_POSITION, COLUMN_DEFAULT,
         IS_NULLABLE, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH,
         CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
         CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,
         COLLATION_NAME, NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
         NUMERIC_SCALE, DATETIME_PRECISION, INTERVAL_TYPE,
         INTERVAL_PRECISION, DOMAIN_CATALOG, DOMAIN_SCHEMA,
         DOMAIN_NAME, ATTRIBUTE_UDT_CATALOG, ATTRIBUTE_UDT_SCHEMA,
         ATTRIBUTE_UDT_NAME, SCOPE_CATALOG, SCOPE_SCHEMA,
         SCOPE_NAME, MAXIMUM_CARDINALITY, DTD_IDENTIFIER, CHECK_REFERENCES,
         IS_DERIVED_REFERENCE_ATTRIBUTE
    FROM INFORMATION_SCHEMA.ATTRIBUTES;
```

2. Rationale: Remove irrelevant column from the definition of the ELEMENT_TYPES view.

In the Definition, replace the view definition for ELEMENT_TYPES_S with:

```
CREATE VIEW ELEMENT_TYPES_S
  ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
    OBJECT_TYPE, ARRAY_TYPE_ID, DATA_TYPE,
    CHAR_MAX_LENGTH, CHAR_OCTET_LENGTH, CHAR_SET_CATALOG,
    CHAR_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
    COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
    NUMERIC_PREC_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
    INTERVAL_TYPE, INTERVAL_PRECISION,
    UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
    SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
    MAX_CARDINALITY, DTD_IDENTIFIER ) AS
  SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
         OBJECT_TYPE, ARRAY_TYPE_IDENTIFIER, DATA_TYPE,
         CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
         CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
         COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
         NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
         DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
         UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
         SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
         MAXIMUM_CARDINALITY, DTD_IDENTIFIER
    FROM INFORMATION_SCHEMA.ELEMENT_TYPES;
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3. *Rationale: Remove irrelevant columns from the definition of the FIELDS view.*

In the Definition, replace the view definition for the FIELDS_S view with:

```
CREATE VIEW FIELDS_S
  ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
    OBJECT_TYPE, ROW_IDENTIFIER, FIELD_NAME,
    ORDINAL_POSITION, DATA_TYPE,
    CHAR_MAX_LENGTH, CHAR_OCTET_LENGTH, CHAR_SET_CATALOG,
    CHAR_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
    COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
    NUMERIC_PREC_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
    INTERVAL_TYPE, INTERVAL_PRECISION,
    UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
    SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
    MAX_CARDINALITY, DTD_IDENTIFIER ) AS
  SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
         OBJECT_TYPE, ROW_IDENTIFIER, FIELD_NAME,
         ORDINAL_POSITION, DATA_TYPE,
         CHARACTER_MAX_LENGTH, CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG,
         CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
         COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
         NUMERIC_PRECISION_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
         INTERVAL_TYPE, INTERVAL_PRECISION, DOMAIN_DEFAULT,
         UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
         SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
         MAXIMUM_CARDINALITY, DTD_IDENTIFIER
    FROM INFORMATION_SCHEMA.FIELDS;
```

4. *Rationale: Editorial - remove erroneous column and other text.*

Replace the Definition of the REFERENCED_TYPES_S view with:

```
CREATE VIEW REFERENCED_TYPES_S
  ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
    OBJECT_TYPE, REFERENCE_TYPE_ID, DATA_TYPE,
    CHAR_MAX_LENGTH, CHAR_OCTET_LENGTH, CHAR_SET_CATALOG,
    CHAR_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
    COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
    NUMERIC_PREC_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
    INTERVAL_TYPE, INTERVAL_PRECISION,
    UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
    SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
    MAX_CARDINALITY, DTD_IDENTIFIER ) AS
  SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
         OBJECT_TYPE, REFERENCE_TYPE_IDENTIFIER, DATA_TYPE,
         CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
         CHARACTER_SET_CATALOG,
         CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
         COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
         NUMERIC_PRECISION_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
         INTERVAL_TYPE, INTERVAL_PRECISION,
         UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
         SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
         MAXIMUM_CARDINALITY, DTD_IDENTIFIER
    FROM INFORMATION_SCHEMA.REFERENCED_TYPES;
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

5. *Rationale: Editorial.*

Replace the Definition of the ROL_TAB_METH_GRNTS view with:

```
CREATE VIEW ROL_TAB_METH_GRNTS
  ( GRANTOR, GRANTEE, TABLE_CATALOG,
    TABLE_SCHEMA, TABLE_NAME, SPECIFIC_CATALOG,
    SPECIFIC_SCHEMA, SPECIFIC_NAME, IS_GRANTABLE ) AS
  SELECT GRANTOR, GRANTEE, TABLE_CATALOG,
         TABLE_SCHEMA, TABLE_NAME, SPECIFIC_CATALOG,
         SPECIFIC_SCHEMA, SPECIFIC_NAME, IS_GRANTABLE
    FROM DEFINITION_SCHEMA.ROLE_TABLE_METHOD_GRANTS;
```

6. *Rationale: Replace use of reserved word as column name.*

Replace the Definition of the SQL_LANGUAGES_S view with:

```
CREATE VIEW SQL_LANGUAGES_S
  ( SOURCE, SQL_LANGUAGE_YEAR, CONFORMANCE,
    INTEGRITY, IMPLEMENTATION, BINDING_STYLE,
    PROGRAMMING_LANGUAGE ) AS
  SELECT SQL_LANGUAGE_SOURCE, SQL_LANGUAGE_YEAR, SQL_LANGUAGE_CONFORMANCE,
         SQL_LANGUAGE_INTEGRITY, SQL_LANGUAGE_IMPLEMENTATION,
         SQL_LANGUAGE_BINDING_STYLE, SQL_LANGUAGE_PROGRAMMING_LANGUAGE
    FROM INFORMATION_SCHEMA.SQL_LANGUAGES;
```

7. *Rationale: Clean up typed table insertability property for non-instantiable types.*

Replace the Definition of the TABLES_S view with:

```
CREATE VIEW TABLES_S
  ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
    TABLE_TYPE, SELF_REF_COLUMN, REF_GENERATION,
    UDT_CATALOG, UDT_SCHEMA, UDT_NAME, IS_INSERTABLE_INTO) AS
  SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
         TABLE_TYPE, SELF_REFERENCING_COLUMN, REFERENCE_GENERATION,
         USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
         USER_DEFINED_TYPE_NAME, IS_INSERTABLE_INTO
    FROM INFORMATION_SCHEMA.TABLES;
```

21.3 EQUAL_KEY_DEGREES assertion

1. *Rationale: Replace use of reserved word as correlation name.*

Replace the Definition with:

```
CREATE ASSERTION EQUAL_KEY_DEGREES
  CHECK     ( NOT EXISTS ( SELECT *
                           FROM ( SELECT COUNT ( DISTINCT FK.COLUMN_NAME ),
                                         COUNT ( DISTINCT PK.COLUMN_NAME )
                                    FROM KEY_COLUMN_USAGE AS FK,
                                         REFERENTIAL_CONSTRAINTS AS RF,
                                         KEY_COLUMN_USAGE AS PK
                                   WHERE ( FK.CONSTRAINT_CATALOG,
                                           FK.CONSTRAINT_SCHEMA,
                                           FK.CONSTRAINT_NAME )
                                         = ( RF.CONSTRAINT_CATALOG,
                                             RF.CONSTRAINT_SCHEMA,
                                             RF.CONSTRAINT_NAME )
                                         AND
                                         ( PK.CONSTRAINT_CATALOG,
                                           PK.CONSTRAINT_SCHEMA,
                                           PK.CONSTRAINT_NAME )
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
                    = ( RF.UNIQUE_CONSTRAINT_CATALOG,
                        RF.UNIQUE_CONSTRAINT_SCHEMA,
                        RF.UNIQUE_CONSTRAINT_NAME )
          GROUP BY RF.CONSTRAINT_CATALOG,
                   RF.CONSTRAINT_SCHEMA,
                   RF.CONSTRAINT_NAME )
        AS R ( FK_DEGREE, PK_DEGREE )
WHERE FK_DEGREE <> PK_DEGREE ) )
```

21.6 ASSERTIONS base table

1. Rationale: Remove irrelevant column from the definition of the ASSERTIONS base table.

Delete the column definition for CHECK_TIME.

2. Rationale: Editorial.

Replace the check constraint ASSERTIONS_DEFERRED_CHECK with:

```
CONSTRAINT ASSERTIONS_DEFERRED_CHECK
  CHECK ( ( IS_DEFERRABLE, INITIALLY_DEFERRED )
          IN ( VALUES ( 'NO', 'NO' ),
                      ( 'YES', 'NO' ),
                      ( 'YES', 'YES' ) ) )
```

21.7 ATTRIBUTES base table

1. Rationale: Editorial.

Replace the check constraint ATTRIBUTES_IS_DERIVED_REFERENCE_ATTRIBUTE_CHECK with:

```
CONSTRAINT ATTRIBUTES_IS_DERIVED_REFERENCE_ATTRIBUTE_CHECK
  CHECK ( IS_DERIVED_REFERENCE_ATTRIBUTE IN ( 'YES', 'NO' ) ),
```

Replace the check constraint ATTRIBUTES_CHECK_DATA_TYPE with:

```
CONSTRAINT ATTRIBUTES_CHECK_DATA_TYPE
  CHECK ( ( UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
            'USER-DEFINED TYPE', DTD_IDENTIFIER )
          IN ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                      OBJECT_TYPE, DTD_IDENTIFIER
                 FROM DATA_TYPE_DESCRIPTOR ) ),
```

21.8 CHARACTER_SETS base table

1. Rationale: Correct too hasty amendment of a change proposal.
Source: BHX-144

Replace Description 2) with:

2) The value of FORM_OF_USE is the null value.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

2. *Rationale: Correct misapplication of a change proposal.*
Source: BHX-144

Replace Description 3) with:

3) The value of NUMBER_OF_CHARACTERS is the null value.

3. *Rationale: Add missing Description.*
Source: GBR-STC-028

Insert the following Description:

7) There is a row in this table for the character set INFORMATION_SCHEMA.SQL_CHARACTER. In that row:

a) CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_IDENTIFIER', respectively.

b) DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA, and DEFAULT_COLLATE_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_CHARACTER', respectively.

21.12 COLLATIONS base table

1. *Rationale: Editorial.*

Replace the primary key constraint COLLATIONS_PAD_PRIMARY_KEY with:

```
CONSTRAINT COLLATIONS_PRIMARY_KEY
     PRIMARY KEY ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ),
```

Replace the foreign key constraint COLLATIONS_PAD_FOREIGN_KEY_SCHEMATA with:

```
CONSTRAINT COLLATIONS_FOREIGN_KEY_SCHEMATA
     FOREIGN KEY ( COLLATION_CATALOG, COLLATION_SCHEMA )
        REFERENCES SCHEMATA,
```

2. *Rationale: Correct too hasty amendment of a change proposal.*
Source: BHX-144

Replace Description 3) with:

3) The values of COLLATION_TYPE, COLLATION_DICTIONARY, and COLLATION_DEFINITION are the null value.

21.14 COLUMNS base table

1. *Rationale: Editorial - typographical error.*

Replace the check constraint COLUMN_CHECK_DATA TYPE with:

```
CONSTRAINT COLUMNS_CHECK_DATA_TYPE
  CHECK ( DOMAIN_CATALOG NOT IN( SELECT CATALOG_NAME
                                   FROM SCHEMATA )
          OR
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
( ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME ) IS NOT NULL
  AND
  ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
    'TABLE', DTD_IDENTIFIER )
    NOT IN ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                    OBJECT_TYPE, DTD_IDENTIFIER
               FROM DATA_TYPE_DESCRIPTOR ) )
OR
( ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME ) IS NULL
  AND
  ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
    'COLUMN', COLUMN_NAME )
    IN ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                OBJECT_TYPE, DTD_IDENTIFIER
           FROM DATA_TYPE_DESCRIPTOR ) ) )
```

21.15 DATA_TYPE_DESCRIPTOR base table

1. *Rationale: Provide missing text in Function section.*

Replace the Function with:

The DATA_TYPE_DESCRIPTOR table has one row for each domain, one row for each column (in each table) and for each attribute (in each structured type) that is defined as having a data type rather than a domain, one row for each distinct type, one row for the result type of each SQL-invoked function, one row for each SQL parameter of each SQL-invoked routine, one row for the result type of each method specification, one row for each parameter of each method specification, and one row for each structured type whose associated reference type has a user-defined representation. It effectively contains a representation of the data type descriptors.

2. *Rationale: Correct definition of the length of <reference type>s and typographical errors.*

Replace the check constraint DATA_TYPE_DESCRIPTOR_DATA_TYPE_CHECK_COMBINATIONS with:

```
CONSTRAINT DATA_TYPE_DESCRIPTOR_DATA_TYPE_CHECK_COMBINATIONS
  CHECK ( ( DATA_TYPE IN ( 'CHARACTER', 'CHARACTER VARYING',
                           'CHARACTER LARGE OBJECT' )
            AND
            ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
              COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NOT NULL
            AND
            ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
              DATETIME_PRECISION, USER_DEFINED_TYPE_CATALOG,
              USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME ) IS NULL
            AND
            ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
            AND
            ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
            AND
            MAXIMUM_CARDINALITY IS NULL )
          OR
          ( DATA_TYPE IN ( 'BIT', 'BIT VARYING' )
            AND
            ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH ) IS NOT NULL
            AND
            ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
            AND
            ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
              DATETIME_PRECISION, USER_DEFINED_TYPE_CATALOG,
              USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME ) IS NULL
            AND
            ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
            AND
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
    MAXIMUM_CARDINALITY IS NULL )
  OR
  ( DATA_TYPE IN ( 'BINARY LARGE OBJECT' )
    AND
    ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH ) IS NOT NULL
    AND
    ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
      NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
      DATETIME_PRECISION, USER_DEFINED_TYPE_CATALOG,
      USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME ) IS NULL
    AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
    MAXIMUM_CARDINALITY IS NULL )
  OR
  ( DATA_TYPE IN ( 'INTEGER', 'SMALLINT' )
    AND
    ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
      COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
    NUMERIC_PRECISION_RADIX IN (2,10 )
    AND
    NUMERIC_PRECISION IS NOT NULL
    AND
    NUMERIC_SCALE = 0
    AND
    DATETIME_PRECISION IS NULL
    AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
    MAXIMUM_CARDINALITY IS NULL )
  OR
  ( DATA_TYPE IN ( 'NUMERIC', 'DECIMAL' )
    AND
    ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
      COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
    NUMERIC_PRECISION_RADIX = 10
    AND
    ( NUMERIC_PRECISION, NUMERIC_SCALE ) IS NOT NULL
    AND
    DATETIME_PRECISION IS NULL
    AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
    MAXIMUM_CARDINALITY IS NULL )
  OR
  ( DATA_TYPE IN ( 'REAL', 'DOUBLE PRECISION', 'FLOAT' )
    AND
    ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
      COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
    NUMERIC_PRECISION IS NOT NULL
    AND
    NUMERIC_PRECISION_RADIX = 2
    AND
    NUMERIC_SCALE IS NULL
    AND
    DATETIME_PRECISION IS NULL
    AND
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

```
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NULL
    AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
    MAXIMUM_CARDINALITY IS NULL )
  OR
  ( DATA_TYPE IN ( 'DATE', 'TIME', 'TIMESTAMP', 'TIME WITH TIME ZONE',
                   'TIMESTAMP WITH TIME ZONE' )
    AND
    ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
      COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
    ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX ) IS NOT NULL
    AND
    NUMERIC_SCALE IS NULL
    AND
    DATETIME_PRECISION IS NOT NULL
    AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NULL
    AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
    MAXIMUM_CARDINALITY IS NULL )
  OR
  ( DATA_TYPE = 'INTERVAL'
    AND
    ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
      COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
    ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX ) IS NOT NULL
    AND
    NUMERIC_SCALE IS NULL
    AND
    DATETIME_PRECISION IS NOT NULL
    AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NULL
    AND
    INTERVAL_TYPE IN ( 'YEAR', 'MONTH', 'DAY', 'HOUR', 'MINUTE',
                       'SECOND', 'YEAR TO MONTH', 'DAY TO HOUR',
                       'DAY TO MINUTE', 'DAY TO SECOND',
                       'HOUR TO MINUTE', 'HOUR TO SECOND',
                       'MINUTE TO SECOND' )
    AND
    INTERVAL_PRECISION IS NOT NULL
    AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
    MAXIMUM_CARDINALITY IS NULL )
  OR
  ( DATA_TYPE = 'BOOLEAN'
    AND
    ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
      COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
    ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX ) IS NULL
    AND
    NUMERIC_SCALE IS NULL
    AND
    DATETIME_PRECISION IS NULL
    AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

```
  AND
  ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
    USER_DEFINED_TYPE_NAME ) IS NULL
  AND
  ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
  AND
  MAXIMUM_CARDINALITY IS NULL )
OR
( DATA_TYPE = 'USER-DEFINED'
  AND
  ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
    DATETIME_PRECISION, CHARACTER_OCTET_LENGTH,
    CHARACTER_MAXIMUM_LENGTH, INTERVAL_TYPE, INTERVAL_PRECISION,
    SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
  AND
  ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
    USER_DEFINED_TYPE_NAME ) IS NOT NULL
  AND
  MAXIMUM_CARDINALITY IS NULL )
OR
( DATA_TYPE = 'REF'
  AND
  ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH ) IS NOT NULL
    AND
    ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
      DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NOT NULL
    AND
    MAXIMUM_CARDINALITY IS NULL )
  OR
  ( DATA_TYPE = 'ARRAY'
    AND
    ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
      DATETIME_PRECISION, CHARACTER_OCTET_LENGTH,
      CHARACTER_MAXIMUM_LENGTH, INTERVAL_TYPE,
      INTERVAL_PRECISION )          IS NULL
    AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NULL
    AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
    MAXIMUM_CARDINALITY IS NOT NULL )
  OR
  ( DATA_TYPE = 'ROW'
    AND
    ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
      DATETIME_PRECISION, CHARACTER_OCTET_LENGTH,
      CHARACTER_MAXIMUM_LENGTH, INTERVAL_TYPE,
      INTERVAL_PRECISION )          IS NULL
    AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NULL
    AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
    MAXIMUM_CARDINALITY IS NULL )
  OR
  ( DATA_TYPE NOT IN ( 'CHARACTER', 'CHARACTER VARYING',
                       'CHARACTER LARGE OBJECT',
                       'BINARY LARGE OBJECT', 'BIT', 'BIT VARYING',
                       'INTEGER', 'SMALLINT', 'NUMERIC', 'DECIMAL',
                       'REAL', 'DOUBLE PRECISION', 'FLOAT',
                       'DATE', 'TIME', 'TIMESTAMP',
                       'INTERVAL', 'BOOLEAN', 'USER-DEFINED',
                       'REF', 'ARRAY', 'ROW' ) ) ),
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

3. *Rationale: Align Description 6) with the Definition section.*

Replace Description 6) with:

6) Case:

a) If DATA_TYPE is 'USER-DEFINED', then the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the qualified name of the user-defined type being described.

b) If the DATA_TYPE is 'REF', then the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the qualified name of the referenced structured type of the reference type being described.

c) Otherwise, the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the null value.

21.16 DIRECT_SUPERTABLES

1. *Rationale: Correct DIRECT_SUPERTABLES_CHECK_NO_REFLEXITIVITY constraint.*

Replace the DIRECT_SUPERTABLE_CHECK_NO_REFLEXITIVTY constraint with the following:

```
CONSTRAINT DIRECT_SUPERTABLES_CHECK_NO_REFLEXITIVITY
  CHECK ( ( TABLE_CATALOG, TABLE_SCHEMA,
            SUPERTABLE_NAME, TABLE_NAME )
          NOT IN ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                          SUPERTABLE_NAME
                     FROM DIRECT_SUPERTABLES ) ),
```

21.23 METHOD_SPECIFICATION_PARAMETERS

1. *Rationale: Delete reference to METHOD_NAME column.*

Delete the column definition for METHOD_NAME.

2. *Rationale: Correct METHOD_SPECIFICATION_PARAMETERS_PRIMARY_KEY constraint.*

Replace the METHOD_SPECIFICATION_PARAMETERS_PRIMARY_KEY constraint with:

```
CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_PRIMARY_KEY
  PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                                  ORDINAL_POSITION ),
```

3. *Rationale: Correct METHOD_SPECIFICATION_PARAMETERS_FOREIGN_KEY constraint.*

Replace the METHOD_SPECIFICATION_PARAMETERS_PRIMARY_KEY constraint with:

```
CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_FOREIGN_KEY
  FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
    REFERENCES METHOD_SPECIFICATIONS,
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0860074 014

21.24 METHOD_SPECIFICATIONS base table

1. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Add the following column definition after the column IS_OVERRIDING:

```
IS_CONSTRUCTOR                              INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT METHOD_SPECIFICATION_IS_CONSTRUCTOR_CHECK
    CHECK ( IS_CONSTRUCTOR IN ('YES', 'NO') )
  CONSTRAINT METHOD_SPECIFICATION_IS_CONSTRUCTOR_COMBINATION_CHECK
    CHECK ( IS_CONSTRUCTOR = 'YES'
            AND
            IS_OVERRIDING = 'NO' ),
```

2. *Rationale: The specification of SQL_DATA_ACCESS is incorrect.*

Replace the definition of SQL_DATA_ACCESS with:

```
SQL_DATA_ACCESS                             INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT METHOD_SPECIFICATIONS_SQL_DATA_ACCESS_NOT_NULL NOT NULL
  CONSTRAINT METHOD_SPECIFICATIONS_SQL_DATA_ACCESS_CHECK
    CHECK ( SQL_DATA_ACCESS IN ( 'NO SQL', 'CONTAINS SQL',
                                 'READS SQL DATA', 'MODIFIES SQL DATA' ) ),
```

3. *Rationale: Correct METHOD_SPECIFICATIONS_LANGUAGE_CHECK constraint.*

Replace the check constraint METHOD_SPECIFICATIONS_LANGUAGE_CHECK with:

```
CONSTRAINT METHOD_SPECIFICATIONS_LANGUAGE_CHECK
  CHECK ( ROUTINE_BODY IN ( 'SQL', 'ADA', 'C', 'COBOL', 'FORTRAN', 'MUMPS',
                            'PASCAL', 'PLI' ) ),
```

4. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Add the following Description Rule:

5.1) The values of IS_CONSTRUCTOR have the following meanings:

YES	The SQL-invoked method is an SQL-invoked constructor method.
NO	The SQL-invoked method is not an SQL-invoked constructor method.

5. *Rationale: The specification of SQL_DATA_ACCESS is incorrect.*

Replace Description Rule 10) with:

10) The values of SQL_DATA_ACCESS have the following meanings:

NO SQL	The SQL-invoked routine does not possibly contain SQL.
CONTAINS SQL	The SQL-invoked routine possibly contains SQL.
READS SQL DATA	The SQL-invoked routine possibly reads SQL-data.
MODIFIES SQL DATA	The SQL-invoked routine possibly modifies SQL-data.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

21.25 PARAMETERS base table

1. Rationale: Editorial - typographical error.

Replace the foreign key constraint PARAMETERS_FOREIGN_KEY_SCHEMATA with:

```
CONSTRAINT PARAMETERS_FOREIGN_KEY_SCHEMATA
  FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA )
    REFERENCES SCHEMATA,
```

21.33 ROUTINES base table

1. Rationale: The specification of SQL_DATA_ACCESS is incorrect.

Replace the definition of SQL_DATA_ACCESS with:

```
SQL_DATA_ACCESS                          INFORMATION_SCHEMA.CHARACTER_ DATA
  CONSTRAINT ROUTINES_SQL_DATA_ACCESS_NOT_NULL NOT NULL
  CONSTRAINT ROUTINES_SQL_DATA_ACCESS_CHECK
    CHECK ( SQL_DATA_ACCESS IN ( 'NO SQL', 'CONTAINS SQL',
                                 'READS SQL DATA', 'MODIFIES SQL DATA' ) ),
```

Replace Description Rule 8) with:

8) The values of SQL_DATA_ACCESS have the following meanings:

NO SQL	The SQL-invoked routine does not possibly contain SQL.
CONTAINS SQL	The SQL-invoked routine possibly contains SQL.
READS SQL DATA	The SQL-invoked routine possibly reads SQL-data.
MODIFIES SQL DATA	The SQL-invoked routine possibly modifies SQL-data.

21.34 SCHEMATA base table

1. Rationale: Editorial - typographical error.

Replace the Definition with:

```
CREATE TABLE SCHEMATA (
  CATALOG_NAME                           INFORMATION_SCHEMA.SQL_IDENTIFIER,
  SCHEMA_NAME                            INFORMATION_SCHEMA.SQL_IDENTIFIER,
  SCHEMA_OWNER                           INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT SCHEMA_OWNER_NOT_NULL NOT NULL,
  DEFAULT_CHARACTER_SET_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT DEFAULT_CHARACTER_SET_CATALOG_NOT_NULL NOT NULL,
  DEFAULT_CHARACTER_SET_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT DEFAULT_CHARACTER_SET_SCHEMA_NOT_NULL NOT NULL,
  DEFAULT_CHARACTER_SET_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT DEFAULT_CHARACTER_SET_NAME_NOT_NULL NOT NULL,
  SQL_PATH                               INFORMATION_SCHEMA.CHARACTER_DATA,
  CONSTRAINT SCHEMATA_PRIMARY_KEY
    PRIMARY KEY ( CATALOG_NAME, SCHEMA_NAME ),
  CONSTRAINT SCHEMATA_FOREIGN_KEY
    FOREIGN KEY ( SCHEMA_OWNER )
      REFERENCES USERS
  )
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4851903 0860076 997

21.35 SQL_FEATURES base table

1. Rationale: Editorial - typographical errors.

Replace the Definition with:

```
CREATE TABLE SQL_FEATURES (
  FEATURE_ID                          INFORMATION_SCHEMA.CHARACTER_DATA,
  FEATURE_NAME                        INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SQL_FEATURES_FEATURE_NAME_NOT_NULL NOT NULL,
  /* Zero for SUB_FEATURE_ID indicates a feature or a package*/
  SUB_FEATURE_ID                      INFORMATION_SCHEMA.CHARACTER_DATA,
  /* Zero-length string for SUB_FEATURE_NAME indicates a feature or a package */
  SUB_FEATURE_NAME                    INFORMATION_SCHEMA.CHARACTER_DATA,
  FEATURE_SUBFEATURE_PACKAGE_CODE     INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT_SQL_FEATURES_FEATURE_SUBFEATURE_PACKAGE_CODE_CHECK
      CHECK ( FEATURE_SUBFEATURE_PACKAGE_CODE IN ( 'FEATURE', 'SUBFEATURE',
                                                   'PACKAGE' ) )
    CONSTRAINT SQL_FEATURES_SUB_FEATURE_NAME_NOT_NULL NOT NULL,
  IS_SUPPORTED                        INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SQL_FEATURES_IS_SUPPORTED_NOT_NULL NOT NULL
    CONSTRAINT SQL_FEATURES_IS_SUPPORTED_CHECK
      CHECK ( IS_SUPPORTED IN ( 'YES', 'NO' ) ),
  IS_VERIFIED_BY                      INFORMATION_SCHEMA.CHARACTER_DATA,
  COMMENTS                            INFORMATION_SCHEMA.CHARACTER_DATA,
  CONSTRAINT SQL_FEATURES_PRIMARY_KEY
    PRIMARY KEY ( FEATURE_ID, SUB_FEATURE_ID ),
  CONSTRAINT SQL_FEATURES_CHECK_SUPPORTED_VERIFIED
    CHECK ( IS_SUPPORTED = 'YES'
            OR
            IS_VERIFIED_BY IS NULL )
  )
```

21.36 SQL_IMPLEMENTATION_INFO base table

1. Rationale: Editorial - typographical error.

Replace the Definition with:

```
CREATE TABLE SQL_IMPLEMENTATION_INFO (
IMPLEMENTATION_INFO_ID                INFORMATION_SCHEMA.CHARACTER_DATA,
IMPLEMENTATION_INFO_NAME              INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT SQL_IMPLEMENTATION_INFO_NAME_NOT_NULL NOT NULL,
INTEGER_VALUE                         INFORMATION_SCHEMA.CARDINAL_NUMBER,
CHARACTER_VALUE                       INFORMATION_SCHEMA.CHARACTER_DATA,
COMMENTS                              INFORMATION_SCHEMA.CHARACTER_DATA,
CONSTRAINT SQL_IMPLEMENTATION_INFO_PRIMARY_KEY
  PRIMARY KEY ( IMPLEMENTATION_INFO_ID )
)
```

21.38 SQL_SIZING base table

1. Rationale: Editorial - typographical error.

Replace the Definition with:

```
CREATE TABLE SQL_SIZING (
  SIZING_ID                       INFORMATION_SCHEMA.CARDINAL_NUMBER,
  SIZING_NAME                     INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SQL_SIZING_SIZING_NAME_NOT_NULL NOT NULL,
  /* If SUPPORTED_VALUE is the null value, that means that the item */
  /* being described is not applicable in the implementation. */
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
/* If SUPPORTED_VALUE is 0 (zero), that means that the item */
/* being described has no limit or the limit cannot be determined.*/
SUPPORTED_VALUE                 INFORMATION_SCHEMA.CARDINAL_NUMBER,
COMMENTS                        INFORMATION_SCHEMA.CHARACTER_DATA,
CONSTRAINT SQL_SIZING_PRIMARY_KEY
  PRIMARY KEY (SIZING_ID )
)
```

21.39 SQL_SIZING_PROFILES base table

1. Rationale: Editorial - typographical errors.

Replace the Function with:

The SQL_SIZING base table has one row for each sizing item defined in ISO/IEC 9075.

Replace the Definition with:

```
CREATE TABLE SQL_SIZING_PROFILES (
SIZING_ID                       INFORMATION_SCHEMA.CARDINAL_NUMBER,
SIZING_NAME                     INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT SQL_SIZING_SIZING_NAME_NOT_NULL NOT NULL,
PROFILE_ID                      INFORMATION_SCHEMA.CHARACTER_DATA,
/* If REQUIRED_VALUE is the null value, that means that the item */
/* being described is not applicable in the profile. */
/* If REQUIRED_VALUE is 0 (zero), that means that the profile */
/* does not set a limit for this sizing item. */
REQUIRED_VALUE                  INFORMATION_SCHEMA.CARDINAL_NUMBER,
COMMENTS                        INFORMATION_SCHEMA.CHARACTER_DATA,
CONSTRAINT SQL_SIZING_PROFILE_PRIMARY_KEY
  PRIMARY KEY ( SIZING_ID, PROFILE_ID )
)
```

21.43 TABLES base table

1. Rationale: Clean up typed table insertability property for non-instantiable types.

In the Definition, add the following column definition:

```
IS_INSERTABLE_INTO                  INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT IS_INSERTABLE_INTO_NOT_NULL NOT NULL
  CONSTRAINT IS_INSERTABLE_INTO_CHECK
    CHECK ( IS_INSERTABLE_INTO IN ( 'YES', 'NO' ) ),
```

2. Rationale: Clean up typed table insertability property for non-instantiable types.

Add the following Description Rule:

6.1) The values of IS_INSERTABLE_INTO have the following meanings:

YES	The table is insertable-into.
NO	The table is not insertable-into

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

21.44 TRANSFORMS base table

1. Rationale: Editorial - typographical error.

Replace the column definition TRANSFORM_TYPE with:

```
TRANSFORM_TYPE                          INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT TRANSFORM_TYPE_NOT_NULL NOT NULL
  CONSTRAINT TRANSFORM_TYPE_CHECK
    CHECK ( TRANSFORM_TYPE IN ('TO SQL', 'FROM SQL') ),
```

21.45 TRANSLATIONS base table

1. Rationale: Correct too hasty amendment of a change proposal.

In the Definition, delete the column constraint:

```
CONSTRAINT TRANSLATION_DEFINITION_NOT_NULL NOT NULL
```

Replace Description 4) with:

4) The value of TRANSLATION_DEFINITION is the null value.

21.47 TRIGGER_COLUMN_USAGE base table

1. Rationale: Enable recording of column dependencies for UPDATE triggers.

Replace the Function with:

The TRIGGER_COLUMN_USAGE base table has one row for each column of a table identified by a <table name> contained in a <table reference> that is contained in the <search condition> of a <triggered action> or explicitly or implicitly referenced in a <triggered SQL statement> of a <trigger definition> of the trigger being described.

2. Rationale: Enable recording of column dependencies for UPDATE triggers.

In the Definition, delete the constraint TRIGGER_COLUMN_USAGE_EVENT_NOT_UPDATE_CHECK.

3. Rationale: Enable recording of column dependencies for UPDATE triggers.

Replace Description 3) with:

3) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME are the catalog name, unqualified schema name, qualified identifier, and column name, respectively, of a column of a table identified by a <table name> contained in the <search condition> of a <triggered action>, or explicitly or implicitly referenced in a <triggered SQL statement> of a <trigger definition> of the trigger being described.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

21.49 TRIGGERS base table

1. *Rationale: Extend TRIGGERS view to allow for the representation of all possible elements of <old or new values alias list>, and delete stray reference to " CONDITION_TIMING,".*

Replace the definitions of the columns CONDITION_TIMING, CONDITION_REFERENCE_OLD_TABLE and CONDITION_REFERENCE_NEW_TABLE with:

```
ACTION_TIMING                           INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT TRIGGERS_ACTION_TIMING_CHECK
    CHECK ( ACTION_TIMING IN ( 'BEFORE', 'AFTER' ) ),
ACTION_REFERENCE_OLD_TABLE              INFORMATION_SCHEMA.SQL_IDENTIFIER,
ACTION_REFERENCE_NEW_TABLE              INFORMATION_SCHEMA.SQL_IDENTIFIER,
ACTION_REFERENCE_OLD_ROW                INFORMATION_SCHEMA.SQL_IDENTIFIER,
ACTION_REFERENCE_NEW_ROW                INFORMATION_SCHEMA.SQL_IDENTIFIER,
```

2. *Rationale: Remove references to non-existing column.*

Delete the constraint EVENT_MANIPULATION_UPDATE_CHECK.

3. *Rationale: Editorial.*

Replace the definition of EVENT_MANIPULATION with:

```
EVENT_MANIPULATION                      INFORMATION_SCHEMA.CHARACTER_DATA
CONSTRAINT TRIGGERS_EVENT_MANIPULATION_CHECK
  CHECK ( EVENT_MANIPULATION IN
          ( 'INSERT', 'DELETE', 'UPDATE' ) ),
```

4. *Rationale: Extend TRIGGERS view to allow for the representation of all possible elements of <old or new values alias list>, and delete stray reference to " CONDITION_TIMING,".*

Replace Descriptions 4), 5) and 6) with:

4) The values of ACTION_TIMING have the following meaning:

BEFORE	The <trigger action time> is BEFORE.
AFTER	The <trigger action time> is AFTER.

5) The value of ACTION_REFERENCE_OLD_TABLE is the <old values table alias> of the trigger being described.

6) The value of ACTION_REFERENCE_NEW_TABLE is the <new values table alias> of the trigger being described.

6.1) The value of ACTION_REFERENCE_OLD_ROW is the <old values correlation name> of the trigger being described.

6.2) The value of ACTION_REFERENCE_NEW_ROW is the <new values correlation name> of the trigger being described.

5. *Rationale: Editorial.*

Replace Description Rule 7) with:

7) The value of ACTION_ORDER is the ordinal position of the trigger in the list of triggers with the same EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA, EVENT_OBJECT_TABLE, EVENT_MANIPULATION, ACTION_TIMING, and ACTION_ORIENTATION.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

6. *Rationale: Description Rule 9) refers to a non-existent BNF non-terminal, <triggered SQL statement list>.*

Replace Description Rule 9) with:

9) ACTION_STATEMENT is a character representation of the <triggered SQL statement> in the <triggered action> of the trigger being described.

7. *Rationale: Editorial.*

Replace Description Rule 10) with:

10) The values of ACTION_ORIENTATION have the following meanings:
ROW The <triggered action> specifies FOR EACH ROW.
STATEMENT The <triggered action> specifies FOR EACH STATEMENT.

21.52 USER_DEFINED_TYPES base table

1. *Rationale: Editorial - typographical error.*

Replace the column definition ORDERING_FORM with:

```
ORDERING_FORM                              INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT USER_DEFINED_TYPES_ORDERING_FORM_NOT_NULL NOT NULL
  CONSTRAINT USER_DEFINED_TYPES_ORDERING_FORM_CHECK
    CHECK ( ORDERING_FORM IN ( 'NONE', 'FULL', 'EQUALS' ) ),
```

2. *Rationale: Editorial - typographical error.*

Replace the column definition REFERENCE_TYPE with:

```
REFERENCE_TYPE                             INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT USER_DEFINED_TYPES_REFERENCE_TYPE_CHECK
    CHECK ( REFERENCE_TYPE IN ( 'SYSTEM GENERATED', 'USER GENERATED',
                                'DERIVED' ) ),
```

3. *Rationale: Editorial*

Replace the from clause with:

```
FROM ( DEFINITION_SCHEMA.USER_DEFINED_TYPES AS U
       LEFT JOIN ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D1
                   LEFT JOIN DEFINITION_SCHEMA.COLLATIONS AS C1
                     ON ( ( C1.COLLATION_CATALOG, C1.COLLATION_SCHEMA,
                            C1.COLLATION_NAME )
                          = ( D1.COLLATION_CATALOG, D1.COLLATION_SCHEMA,
                              D1.COLLATION_NAME ) ) )
         ON ( ( U.USER_DEFINED_TYPE_CATALOG, U.USER_DEFINED_TYPE_SCHEMA,
                U.USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE',
                U.SOURCE_DTD_IDENTIFIER )
              = ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                  OBJECT_TYPE, D1.DTD_IDENTIFIER )
             OR
             ( U.USER_DEFINED_TYPE_CATALOG, U.USER_DEFINED_TYPE_SCHEMA,
               U.USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE',
               U.REF_DTD_IDENTIFIER )
             = ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                 OBJECT_TYPE, D1.DTD_IDENTIFIER ) ) )
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

21.56 VIEWS base table

1. *Rationale: Clean up typed table insertability property for non-instantiable types.*

In the Definition, delete the column definition for IS_INSERTABLE_INTO.

2. *Rationale: Clean up typed table insertability property for non-instantiable types.*

Delete Description 6)

22.1 SQLSTATE

1. *Rationale: The SQLSTATE Class codes allocated for SQL/MM no longer needed since they are not used by ISO/IEC 13249.*

Delete the rows in Table 27, "SQLSTATE class and subclass values" for the Class values "H1", "H2", "H3", "H4", "H5", "H6", "H7", "H8", "H9", "HA", "HB", "HC", "HD", "HE" and "HF".

22.3 SQL Multimedia and Application Package SQLSTATE Subclasses

1. *Rationale: The SQLSTATE Class codes allocated for SQL/MM no longer needed since they are not used by ISO/IEC 13249.*

Delete Subclause 22.3, "SQL Multimedia and Application Package SQLSTATE Subclasses" including Table 29, "SQLSTATE class codes for SQL/MM)".

23.1 General conformance requirements

1. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".*

Add the following row to "Table 30 — Implied feature relationships":

Feature ID	Feature Description	Implied Feature ID	Implied Feature Description
F451	Character set definition	F461	Named character sets

Annex A SQL conformance summary

1. *Rationale: Editorial.*

Replace Item 4) d) i) with:

4) d) i) Without Feature F052, "Intervals and datetime arithmetic", conforming SQL shall contain no <interval value function>.

2. *Rationale: Create a separate Feature for <overlaps predicate>.*

Delete Items 4) h) and 4) i).

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Add the following Item:

4.1) Specifications for Feature F053, "OVERLAPS predicate":

a) Subclause 8.1, "<predicate>":

i) Without Feature F053, "OVERLAPS predicate", conforming SQL language shall not contain any <overlaps predicate>.

b) Subclause 8.12, "<overlaps predicate>":

i) Without Feature F053, "OVERLAPS predicate", conforming SQL language shall not contain any <overlaps predicate>.

3. Rationale: There is no syntax "OF NEW TYPE" in <table definition>.

Delete Item 20)

4. Rationale: Editorial.

Replace Item 23) a) i) with:

23) a) i) Without Feature F411, "Time zone specification", conforming SQL shall not specify a <time zone interval>.

5. Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".

Replace Items 27) and 28) with:

27) Specifications for Feature F451, "Character set definition":

a) Subclause 11.1, "<schema definition>":

i) Without Feature F451, "Character set definition", conforming SQL language shall not contain any <character set definition>.

b) Subclause 11.30, "<character set definition>":

i) Without Feature F451, "Character set definition", conforming SQL language shall not specify any <character set definition>.

ii) Without Feature F451, "Character set definition", and Feature F691, "Collation and translation", <collation source> shall specify DEFAULT.

c) Subclause 11.31, "<drop character set statement>":

i) Without Feature F451, "Character set definition", conforming SQL language shall contain no <drop character set statement>.

d) Subclause 13.5, "<SQL procedure statement>":

i) Without Feature F451, "Character set definition", an <SQL schema definition statement> shall not be a <character set definition>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ii) Without Feature F451, "Character set definition", an <SQL schema definition statement> shall not be a <drop character set statement>.

28) Specifications for Feature F461, "Named character sets":

a) Subclause 5.3, "<literal>":

i) Without Feature F461, "Named character sets", a <character string literal> shall not specify a <character set specification>.

b) Subclause 5.4, "Names and identifiers":

i) Without Feature F461, "Named character sets", conforming SQL language shall not contain any <character set name>.

c) Subclause 6.1, "<data type>":

i) Without Feature F461, "Named character sets", a <data type> shall not specify CHARACTER SET.

d) Subclause 10.5, "<privileges>":

i) Without Feature F461, "Named character sets", in conforming SQL language, an <object name> shall not specify CHARACTER SET.

e) Subclause 10.6, "<character set specification>":

i) Without Feature F461, "Named character sets", conforming SQL language shall not contain a <character set specification>.

f) Subclause 11.1, "<schema definition>":

i) Without Feature F461, "Named character sets", a <schema character set specification> shall not be specified.

g) Subclause 13.2, "<module name clause>":

i) Without Feature F461, "Named character sets", <module character set specification> shall not be specified.

6. Rationale: Supply missing Conformance Rules for <contextually typed row value constructor>

Add Item 38) a) iii):

38) a) iii) Without Feature F641, "Row and table constructors", a <contextually typed row value constructor> that is not simply contained in a <contextually typed table value constructor> shall not contain more than one <row value constructor element>.

Add Item 38) a) iv):

38) a) iv) Without Feature F641, "Row and table constructors", a <contextually typed row value constructor> shall not be a <row subquery>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

7. *Rationale: Use the correct non-terminal symbols.*

Replace Item 38) b) i) with:

38) b) i) Without Feature F641, "Row and table constructors", the <contextually typed row value expression list> of a <contextually typed table value constructor> shall contain exactly one <contextually typed row value constructor> *RVE*. *RVE* shall be of the form "(<contextually typed row value constructor element list>)".

8. *Rationale: Supply missing Conformance Rules for <contextually typed table value constructor>*

Add Item 38) b) iii):

38) b) iii) Without Feature F641, "Row and table constructors", the <contextually typed row value expression list> of a <contextually typed table value constructor> shall contain exactly one <contextually typed row value constructor> *RVE*. *RVE* shall be of the form "(<contextually typed row value constructor element list>)".

Add Item 38) b) iv):

38) b) iv) Without Feature F641, "Row and table constructors", conforming SQL language shall not contain any <contextually typed table value constructor>.

9. *Rationale: Editorial.*

Replace Item 56) d) i) with:

56) d) i) Without Feature S023, "Basic structured types", conforming SQL language shall not contain any <new specification>.

10. *Rationale: Named column joins implicitly perform <comparison predicate>s, and should be subject to the same Conformance Rules.*

Add the following Conformance Rules to Item 57):

57) b.1) Without Feature S024, "Enhanced structured types", if NATURAL or <named column join> is specified, and if *C* is a corresponding join column, then the declared type of *C* shall not be based on a structured type.

11. *Rationale: Correct the references to <grant statement>.*

Replace Item 57) t) with:

57) t) Subclause 12.2, "< grant privilege statement> ":

i) Without Feature S024, " Enhanced structured types", a <specific routine designator> contained in a <grant privilege statement> shall not identify a method.

12. *Rationale: Clarify that Feature S023 also comprises <method specification list>.*

Add the following Item:

56) g) i.1) Without Feature S023, "Basic structured types", conforming SQL language shall not specify <method specification list>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

13. Rationale: permit tables of structured type without requiring Feature S043, "Enhanced reference types".

Replace Item 59) f) ii) with:

59) f) ii) Without Feature S043, "Enhanced reference types", a <self-referencing column specification> shall specify SYSTEM GENERATED.

14. Rationale: Use of ONLY requires Feature S111, "ONLY in query expressions".

Add the following Item:

66) a.1) Subclause 14.6, "<delete statement: positioned>":

i) Without Feature S111, "ONLY in query expressions", a <target table> shall not specify ONLY.

15. Rationale: Editorial.

Replace Item 68) b) i) with:

68) b) i) Without Feature S161, "Subtype treatment", conforming SQL Language shall contain no <subtype treatment>.

16. Rationale: Editorial.

Replace Item 71) a) i) with:

71) a) i) Without Feature S241, "Transform functions", conforming SQL language shall not specify <transform group specification>.

17. Rationale: Editorial.

Replace Item 72) a) i) with:

72) a) i) Without Feature S251, "User-defined orderings", conforming SQL shall contain no <user-defined ordering definition>.

18. Rationale: Editorial.

Replace Item 76) a) i) with:

76) a) i) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain any <binary string literal>.

19. Rationale: Editorial.

Replace Item 77) a) ii) with:

77) a) ii) Without Feature T042, "Extended LOB data type support", conforming SQL language shall not contain any <blob value function>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

20. Rationale: Editorial.

Replace Item 77) d) ii) with:

77) d) ii) Without Feature F421, "National character", and Feature T042, "Extended LOB data type support", neither operand of <concatenation> shall be of declared type NATIONAL CHARACTER LARGE OBJECT.

21. Rationale: Editorial.

Replace Item 87) b) i) with:

87) b) i) Without Feature T211, "Basic trigger capability", conforming SQL language shall not contain a <trigger definition>.

22. Rationale: Supply missing application of functional dependencies to <having clause>.

Add the following to Item:

95) a.0) Subclause 7.10, "<having clause>"

a) Without Feature T301, "Functional dependencies", each column reference directly contained in the <search condition> shall be one of the following:

i) an unambiguous reference to a grouping column of *T*, or

ii) an outer reference.

b) Without Feature T301, "Functional dependencies", each column reference contained in a <subquery> in the <search condition> that references a column of *T* shall be one of the following:

i) an unambiguous reference to a grouping column of *T*, or

ii) contained in a <set function specification>.

23. Rationale: Remove redundant item. <grant role statement> is defined in <grant statement>.

Delete Item 99) b) ii).

24. Rationale: Remove undefined SET ROW functionality.

Delete Item 102).

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

25. Rationale: Adapting the Conformance Rules to the Feature name and broader scope of Feature T431, "Extended grouping capabilities".

Replace Item 103) with:

103) Specifications for Feature T431, "Extended grouping capabilities":

a) Sublause 6.16, "<set function specification>":

i) Without Feature T431, "Extended grouping capabilities", conforming SQL language shall not contain a <set function specification> that is a <grouping operation>.

b) Subclause 7.9, "<group by clause>":

i) Without Feature T431, "Extended grouping capabilities", conforming SQL language shall not specify ROLLUP, CUBE, GROUPING SETS, or <grand total>.

ii) Without Feature T431, "Extended grouping capabilities", an <ordinary grouping set> shall be a <grouping column reference>.

26. Rationale: Editorial.

Replace Item 106) a) i) with:

106) a) i) Without Feature T471, "Result sets return value", conforming SQL language shall not specify <dynamic result sets characteristic>.

Annex B Implementation-defined elements

1. Rationale: Insert missing item.

Add the following Item:

1.1) Subclause 4.2.4 "Named character sets":

a) The collation and form-of-use of the named character set SQL_CHARACTER are implementation-defined.

2. Rationale: The implementation-defined collation used to compare identifiers has not been noted.

Add the following Item:

12.1) Subclause 5.2, "<token> and <separator>": equivalence of two <regular identifier>s, or a <regular identifier> and a <delimited identifier>, is determined using an implementation-defined collation that is sensitive to case.

3. Rationale: Remove undefined non-terminals .

Delete Item 30)

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4851903 0860088 609

4. *Rationale: Missing entry for SQL-data access indication.*

Add the following Item:

33.1) Subclause 11.49 "<SQL-invoked routine>":

a) If READS SQL DATA is specified, then it is implementation-defined whether the <SQL routine body> shall not contain an <SQL procedure statement> *S* that satisfies at least one of the following:

i) *S* is an <SQL data change statement>.

ii) *S* contains a <routine invocation> whose subject routine is an SQL-invoked routine that possibly modifies SQL-data.

iii) *S* contains an <SQL procedure statement> that is an <SQL data change statement>.

b) If CONTAINS SQL is specified, then it is implementation-defined whether the <SQL routine body> shall not contain an <SQL procedure statement> *S* that satisfies at least one of the following:

i) *S* is an <SQL data statement> other than <free locator statement> and <hold locator statement>.

ii) *S* contains a <routine invocation> whose subject routine is an SQL-invoked routine that possibly modifies SQL-data or possibly reads SQL-data.

iii) *S* contains an <SQL procedure statement> that is an <SQL data statement> other than <free locator statement> and <hold locator statement>.

5. *Rationale: Delete spurious items.*

Delete Items 51) and 52).

Annex C Implementation-dependent elements

1. *Rationale: The notion of distinct may be implementation-dependent if a RELATIVE or MAP ordering function is incorrectly defined.*

Add the following item:

0.1) Subclause 3.1.5, "Definitions provided in Part 2"

a) Whether two nonnull values of user-defined type whose comparison form is RELATIVE or MAP that result in ***unknown*** when tested for equality according to the rules of Subclause 8.2, "<comparison predicate>" are distinct or not is implementation-dependent.

2. *Rationale: There is no syntax "OF NEW TYPE" in <table definition>. Fixing bugs connected with the use of <user-defined type> syntax.*

Delete Item 20)

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0860089 545

Annex E Incompatibilities with ISO/IEC 9075:1992 and ISO/IEC 9075-4:1996

1. Rationale: Editorial.

Replace point 1) with:

1) In ISO/IEC 9075:1992, Subclause 12.3, "<procedure>", a <parameter declaration list> had an alternative "<parameter declaration> . . . " (that is, a parameter list not surrounded by parentheses and with the individual component parameter declarations not separated by commas). This option was listed in ISO/IEC 9075:1992, as a deprecated feature. In ISO/IEC 9075-2:1999, the equivalent Subclause 13.3, "<externally-invoked procedure>", does not contain this option. SQL-client modules that used this deprecated feature may be converted to conforming SQL by inserting a comma between each pair of <parameter declaration>s and placing a left parenthesis before and a right parenthesis after the entire parameter list.

2. Rationale: Editorial. Correct the list of incompatible new <reserved word>s.

In point 14) delete the texts:

— ABS
— ACTION
— AGGREGATE
— ALIAS
— CARDINALITY
— COMPLETION
— DESTROY
— DICTIONARY
— EVERY
— FACTOR
— HOST
— IGNORE
— INITIALIZE
— ITERATE
— LESS
— LIMIT
— MOD
— MODIFY
— NO
— OFF
— OPERATION
— OPERATOR
— OVERLAY
— PARAMETERS
— PREORDER
— RELATIVE
— REPLACE
— SENSITIVE
— SEQUENCE
— SESSION
— SPACE
— STRUCTURE
— SUBLIST
— SYMBOL
— TERM
— TERMINATE

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

— THE
— TYPE
— VARIABLE

In Item 14) add the following list elements:

— CURRENT_ROLE
— DYNAMIC
— FUNCTION
— INOUT
— LATERAL
— LOCALTIME
— LOCALTIMESTAMP
— METHOD
— OUT
— RELEASE
— UNNEST

Annex F SQL feature and package taxonomy

1. *Rationale: Editorial - typographic error.*

Replace row 20 in "Table 31 — SQL/Foundation feature taxonomy and definition for Core SQL" with:

Feature	ID	Feature Name	Feature Description
20	E021-12	Character comparison	— Subclause 8.2, "<comparison predicate>": For the CHARACTER and CHARACTER VARYING data types, without support for <table subquery> and without support for Feature F131, "Grouped operations"

2. *Rationale: Feature F031-13 should reference <drop table statement> rather than <drop column definition>.*

Replace row 118 in "Table 31 — SQL/Foundation feature taxonomy and definition for Core SQL" with:

Feature	ID	Feature Name	Feature Description
118	F031-13	DROP TABLE statement: RESTRICT clause	— Subclause 11.20 <drop table statement>: With a <drop behavior> of RESTRICT

3. *Rationale: Create a separate Feature for <overlaps predicate>.*

Add the following row into "Table 32—SQL/Foundation feature taxonomy for features outside Core SQL"

Feature	ID	Feature Name
7.1	F053	OVERLAPS predicate

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4. *Rationale: Remove undefined SET ROW functionality.*

In Table 32, delete row 158, Feature ID T411, Feature Name "UPDATE STATEMENT: SET ROW option

5. *Rationale: Renaming of Feature T431 is necessary to reflect its broader scope.*

In Table 32 replace row 159 with:

Feature	ID	Feature Name
159	T431	Extended grouping capabilities

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

ISO/IEC 9075-3:1999
Database Languages - SQL-Part 3:Call Level Interface

3.1.1.Definitions provided in Part 3

1. Rationale: Provide missing definition

Add the following item:

a.0) **data source**: A synonym for the SQL-server that is part of the current connection.

3.3.2.1 Clause, subclause and table relationships

1. Rationale: Separating data type correspondence tables in Foundation and CLI.

Replace the following rows from Table 1 "Clause, Subclause, and Table relationships":

Clause, Subclause, or Table in this part of /ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 5.15, "Data type correspondences"	Subclause 13.6, "Data type correspondences"	ISO/IEC 9075-2
Table 44, "Data type correspondences for Ada"	Table 18, "Data type correspondences for Ada"	ISO/IEC 9075-2
Table 45, "Data type correspondences for C"	Table 18, "Data type correspondences for C"	ISO/IEC 9075-2
Table 46, "Data type correspondences for COBOL"	Table 18, "Data type correspondences for COBOL"	ISO/IEC 9075-2
Table 47, "Data type correspondences for Fortran"	Table 18, "Data type correspondences for Fortran"	ISO/IEC 9075-2
Table 48, "Data type correspondences for MUMPS"	Table 18, "Data type correspondences for MUMPS"	ISO/IEC 9075-2
Table 49, "Data type correspondences for Pascal"	Table 18, "Data type correspondences for Pascal"	ISO/IEC 9075-2
Table 50, "Data type correspondences for PL/I"	Table 18, "Data type correspondences for PL/I"	ISO/IEC 9075-2

with the following rows:

Clause, Subclause, or Table in this part of /ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 5.15, "SQL/CLI data type correspondences"	*none*	*none*
Table 44, "SQL/CLI data type correspondences for Ada"	*none*	*none*
Table 45, "SQL/CLI data type correspondences for C"	*none*	*none*
Table 46, "SQL/CLI data type	*none*	*none*

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

correspondences for COBOL"		
Table 47, "SQL/CLI data type correspondences for Fortran"	*none*	*none*
Table 48, "SQL/CLI data type correspondences for MUMPS"	*none*	*none*
Table 49, "SQL/CLI data type correspondences for Pascal"	*none*	*none*
Table 50, "SQL/CLI data type correspondences for PL/I"	*none*	*none*

4.4.5 Connection attributes

1. Rationale: Remove the anomaly in <savepoint specifier> and correct the specification of which locators are marked invalid when an SQL-transaction ends.

Replace the 5th paragraph with:

The SAVEPOINT NAME connection attribute specifies the savepoint to be referenced in an invocation of the EndTran routine that uses the SAVEPOINT NAME ROLLBACK or SAVEPOINT NAME RELEASE CompletionType. The SAVEPOINT NAME attribute is set to a zero-length string when the SQL-connection is allocated.

5.1 <CLI routine>

1. Rationale: Add missing rule.

Add the following Syntax Rule:

8.1) There shall be no <separator> between the <CLI name prefix> and the <CLI generic name>

2. Rationale: Separating data type correspondence tables in Foundation and CLI.

Replace Syntax Rule 12) with:

12) Let *operative data type correspondence table* be the data type correspondence table for *HL* as specified in Subclause 5.15, "SQL/CLI data type correspondences". Refer to the two columns of the operative data type correspondence table as the "SQL data type column" and the "host data type column".

5.3 Description of CLI item descriptor areas

1. Rationale: Separating data type correspondence tables in Foundation and CLI.

Replace General Rule 6) with:

6) Let *HL* be the standard programming language of the invoking host program. Let *operative data type correspondence table* be the data type correspondence table for *HL* as specified in Subclause 5.15, "SQL/CLI data type correspondences". Refer to the two columns of the operative data type correspondence table as the *SQL data type column* and the *host data type column*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

5.5 Implicit DESCRIBE USING clause

1. Rationale: Editorial - typographic error.

Replace General Rule 5) c) iv) 1) with:

5) c) iv) 1) If TYPE indicates a <character string type>, then LENGTH is set to the length or maximum length in characters of the character string. OCTET_LENGTH is set to the maximum possible length in octets of the character string. If *HL* is C, then the lengths specified in LENGTH and OCTET_LENGTH do not include the implementation-defined null character that terminates a C character string. CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are set to the <character set name> of the character string's character set. COLLATION_CATALOG, COLLATION_SCHEMA, and COLLATION_NAME are set to the <collation name> of the character string's collation.

5.11 Deferred parameter check

1. Rationale: Editorial - style error.

Replace General Rule 4) with:

4) Let *L2* be the set of all allocated SQL-statements in *L1* that have an associated deferred parameter number.

5.13 Description of CLI item descriptor areas

1. Rationale: Remove mangled text

Replace Syntax Rule 5) c) xiv) with:

5) c) xiv) TYPE indicates ARRAY or ARRAY LOCATOR, the value of CARDINALITY is a valid value for the cardinality of an array, there is exactly one immediately subordinate descriptor area of IDA, and that item descriptor area is valid.

2. Rationale: Editorial - misplaced row.

Delete the following row from Table 6 — Fields in SQL/CLI row and parameter descriptor areas

Field	Data Type
Fields in item descriptor areas	
COUNT	SMALLINT

Add the following row into Table 6 — Fields in SQL/CLI row and parameter descriptor areas

Field	Data Type
Header fields	
COUNT	SMALLINT

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5.14 Other tables associated with CLI

1. Rationale: Correct the specification of which locators are marked invalid when an SQL-transaction ends.

In Table 14 — Codes used for transaction termination, replace the row:

Termination type	Code
SAVEPOINT NAME COMMIT	2

with:

Termination type	Code
SAVEPOINT NAME ROLLBACK	2

2. Rationale: Remove the anomaly in <savepoint specifier>.

In Table 14 — Codes used for transaction termination, delete the following rows:

Termination type	Code
SAVEPOINT NUMBER COMMIT	3
SAVEPOINT NUMBER RELEASE	5

In Table 16 — Codes used for connection attributes, delete the following row:

Attribute	Code	May be set
SAVEPOINT NUMBER	10028	Yes

In Table 19 — Data types of attributes, delete the following row:

Attribute	Data type	Values
Statement attributes		
SAVEPOINT NUMBER	INTEGER	Not specified

3. Rationale: Correct the contents of the Type column of Table 20 "Codes used for descriptor fields".

In Table 20 — Codes used for descriptor fields, delete the following rows:

Field	Code	SQL Item Descriptor Name	Type
CARDINALITY	1040	CARDINALITY	Status
CURRENT_TRANSFORM_GROUP	1039	(Not applicable)	Status
DEGREE	1041	DEGREE	Status
RETURNED_CARDINALITY_ POINTER	1043	RETURNED_CARDINALITY	Status

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

SCOPE_CATALOG	1033	SCOPE_CATALOG	Status
SCOPE_NAME	1035	SCOPE_NAME	Status
SCOPE_SCHEMA	1034	SCOPE_SCHEMA	Status
SPECIFIC_TYPE_CATALOG	1036	(Not applicable)	Status
SPECIFIC_TYPE_NAME	1038	(Not applicable)	Status
SPECIFIC_TYPE_SCHEMA	1037	(Not applicable)	Status

In Table 20 — Codes used for descriptor fields, add the following rows:

Field	Code	SQL Item Descriptor Name	Type
CARDINALITY	1040	CARDINALITY	Item
CURRENT_TRANSFORM_GROUP	1039	(Not applicable)	Item
DEGREE	1041	DEGREE	Item
RETURNED_CARDINALITY_ POINTER	1043	RETURNED_CARDINALITY	Item
SCOPE_CATALOG	1033	SCOPE_CATALOG	Item
SCOPE_NAME	1035	SCOPE_NAME	Item
SCOPE_SCHEMA	1034	SCOPE_SCHEMA	Item
SPECIFIC_TYPE_CATALOG	1036	(Not applicable)	Item
SPECIFIC_TYPE_NAME	1038	(Not applicable)	Item
SPECIFIC_TYPE_SCHEMA	1037	(Not applicable)	Item

5.15 Data type correspondences

1. Rationale: Separating data type correspondence tables in Foundation and CLI.

Change the name of this subclause to "SQL/CLI data type correspondences".

2. Rationale: Separating data type correspondence tables in Foundation and CLI.

In the Function replace the 1st paragraph with:

Replace first paragraph Specify the SQL/CLI data type correspondences for SQL data types and host language types associated with the required parameter mechanisms, as shown in Table 3, "Supported calling conventions of SQL/CLI routines by language".

3. Rationale: Separating data type correspondence tables in Foundation and CLI.

In the Tables section:

Replace the title of Table 44 — "Data type correspondences for Ada" with:
Table 44 — "SQL/CLI data type correspondences for Ada".

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Replace the title of Table 45 — "Data type correspondences for C" with:
Table 45 — "SQL/CLI data type correspondences for C".

Replace the title of Table 46 — "Data type correspondences for COBOL" with:
Table 46 — "SQL/CLI data type correspondences for COBOL".

Replace the title of Table 47 — "Data type correspondences for Fortran" with:
Table 47 — "SQL/CLI data type correspondences for Fortran".

Replace the title of Table 48 — "Data type correspondences for MUMPS" with:
Table 48 — "SQL/CLI data type correspondences for MUMPS".

Replace the title of Table 49 — "Data type correspondences for Pascal" with:
Table 49 — "SQL/CLI data type correspondences for Pascal".

Replace the title of Table 50 — "Data type correspondences for PL/I" with:
Table 50 — "SQL/CLI data type correspondences for PL/I".

6.5 BindCol

1. Rationale: Separating data type correspondence tables in Foundation and CLI.

Replace General Rule 8) with:

8) Let *HL* be the standard programming language of the invoking host program. Let *operative data type correspondence table* be the data type correspondence table for *HL* as specified in Subclause 5.15, "SQL/CLI data type correspondences". Refer to the two columns of the *operative data type correspondence table* as the *SQL data type column* and the *host data type column*.

6.6 BindParameter

1. Rationale: Separating data type correspondence tables in Foundation and CLI.

Replace General Rule 9) with:

9) Let *HL* be the standard programming language of the invoking host program. Let *operative data type correspondence table* be the data type correspondence table for *HL* as specified in Subclause 5.15, "SQL/CLI data type correspondences". Refer to the two columns of the *operative data type correspondence table* as the *SQL data type column* and the *host data type column*.

6.9 ColAttribute

1. Rationale: Correct definition of parameter length.

Replace the Definition with:

```
ColAttribute (
     StatementHandle      IN  INTEGER,
     ColumnNumber         IN  SMALLINT,
     FieldIdentifier      IN  SMALLINT,
     CharacterAttribute   OUT CHARACTER(L),
     BufferLength         IN  SMALLINT,
     StringLength         OUT SMALLINT,
     NumericAttribute     OUT INTEGER )
     RETURNS SMALLINT
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.10 ColumnPrivileges

1. Rationale: Correct definition of parameter length.

Replace the Definition with:

```
ColumnPrivileges (
      StatementHandle      IN  INTEGER,
      CatalogName          IN  CHARACTER(L1),
      NameLength1          IN  SMALLINT,
      SchemaName           IN  CHARACTER(L2),
      NameLength2          IN  SMALLINT,
      TableName            IN  CHARACTER(L3),
      NameLength3          IN  SMALLINT,
      ColumnName           IN  CHARACTER(L4),
      NameLength4          IN  SMALLINT )
      RETURNS SMALLINT
```

where each of *L1*, *L2*, *L3*, and *L4* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.11 Columns

1. Rationale: Correct definition of parameter length.

Replace the Definition with:

```
Columns (
      StatementHandle      IN  INTEGER,
      CatalogName          IN  CHARACTER(L1),
      NameLength1          IN  SMALLINT,
      SchemaName           IN  CHARACTER(L2),
      NameLength2          IN  SMALLINT,
      TableName            IN  CHARACTER(L3),
      NameLength3          IN  SMALLINT,
      ColumnName           IN  CHARACTER(L4),
      NameLength4          IN  SMALLINT )
      RETURNS SMALLINT
```

where each of *L1*, *L2*, *L3*, and *L4* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.12 Connect

1. Rationale: Correct definition of parameter length.

Replace the Definition with:

```
Connect (
      ConnectionHandle     IN  INTEGER,
      ServerName           IN  CHARACTER(L1),
      NameLength1          IN  SMALLINT,
      UserName             IN  CHARACTER(L2),
      NameLength2          IN  SMALLINT,
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
      Authentication        IN  CHARACTER(L3),
      NameLength3           IN  SMALLINT )
      RETURNS SMALLINT
```

where

:

— *L1* has a maximum value of 128.

— *L2* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

— *L3* has an implementation-defined maximum value.

6.14 DataSources

1. Rationale: Correct definition of parameter length.

Replace the Definition with:

```
DataSources (
      EnvironmentHandle     IN  INTEGER,
      Direction             IN  SMALLINT,
      ServerName            OUT CHARACTER(L1),
      BufferLength1         IN  SMALLINT,
      NameLength1           OUT SMALLINT,
      Description           OUT CHARACTER(L2),
      BufferLength2         IN  SMALLINT,
      NameLength2           OUT SMALLINT )
      RETURNS SMALLINT
```

where *L1* and *L2* have maximum values equal to the implementation-defined maximum length of a variable-length character string.

6.15 DescribeCol

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
DescribeCol (
      StatementHandle       IN  INTEGER,
      ColumnNumber          IN  SMALLINT,
      ColumnName            OUT CHARACTER(L),
      BufferLength          IN  SMALLINT,
      NameLength            OUT SMALLINT,
      DataType              OUT SMALLINT,
      ColumnSize            OUT INTEGER,
      DecimalDigits         OUT SMALLINT,
      Nullable              OUT SMALLINT )
      RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

6.17 EndTran

1. *Rationale: Remove the anomaly in <savepoint specifier> and correct the specification of which locators are marked invalid when an SQL-transaction ends.*

Replace General Rule 13) with:

13) If *CT* indicates SAVEPOINT NAME RELEASE, then:

a) If *HT* is not CONNECTION HANDLE, then an exception condition is raised: *CLI-specific condition — invalid handle.*

b) Let *SP* be the value of the SAVEPOINT NAME connection attribute of *C*.

c) If *SP* does not specify a savepoint established within the current SQL-transaction, then an exception condition is raised: *savepoint exception — invalid specification.*

d) The savepoint identified by *SP* and all savepoints established by the current SQL-transaction subsequent to the establishment of *SP* are destroyed.

2. *Rationale: Correct the specification of which locators are marked invalid when an SQL-transaction ends.*

Replace General Rule 14) d) with:

14) d) Every valid locator value is marked invalid.

3. *Rationale: Remove the anomaly in <savepoint specifier> and correct the specification of which locators are marked invalid when an SQL-transaction ends.*

Replace General Rule 15) with:

15) If *CT* indicates SAVEPOINT NAME ROLLBACK, then:

a) If *HT* is not CONNECTION HANDLE, then an exception condition is raised: *CLI-specific condition — invalid handle.*

b) Let *SP* be the value of the SAVEPOINT NAME connection attribute of *C*.

c) If *SP* does not specify a savepoint established within the current SQL-transaction, then an exception condition is raised: *savepoint exception — invalid specification.*

d) If an atomic execution context is active and *SP* specifies a savepoint established before the beginning of the most recent atomic execution context, then an exception condition is raised: *savepoint exception — invalid specification.*

e) Any changes to SQL-data or schemas that were made by the current SQL-transaction subsequent to the establishment of *SP* are cancelled.

f) All savepoints established by the current SQL-transaction subsequent to the establishment of *SP* are destroyed.

g) Every valid locator generated in the current SQL-transaction subsequent to the establishment of *SP* is marked invalid.

h) For every open cursor *OC* in *L3* that was opened subsequent to the establishment of *SP*:

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Let *S* be the allocated SQL-statement with which *OC* is associated.

ii) *OC* is placed in the closed state and its copy of the select source is destroyed.

iii) Any fetched row associated with *OC* is removed from association with *S*.

i) The status of any open cursors in *L3* that were opened by the current SQL-transaction before the establishment of *SP* is implementation-defined.

NOTE 26 — The current SQL-transaction is not terminated, and there is no other effect on the SQL-data or schemas.

6.18 Error

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
Error (
     EnvironmentHandle     IN  INTEGER,
     ConnectionHandle      IN  INTEGER,
     StatementHandle       IN  INTEGER,
     Sqlstate              OUT CHARACTER(5),
     NativeError           OUT INTEGER,
     MessageText           OUT CHARACTER(L),
     BufferLength          IN  SMALLINT,
     TextLength            OUT SMALLINT )
     RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.19 ExecDirect

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
ExecDirect (
     StatementHandle       IN  INTEGER,
     StatementText         IN  CHARACTER(L),
     TextLength            IN  INTEGER )
     RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.21 Fetch

1. Rationale: Editorial.

Replace General Rule 13) with:

13) Let *RS* be the number of rows in *NR*. For *RN* ranging from 1 (one) to *RS*:

a) Let *RNR* be the *RN*-th row of *NR*. Let *ROWS_PROCESSED* be 0 (zero).

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4851903 0860102 809

Case:

i) If an exception condition is raised during derivation of any <derived column> associated with *RNR* and *ASP* is not a null pointer, then set the *RN*-th element of *ASP* to 5 (indicating **Row error**). For all diagnostic records that result from the application of this rule, the ROW_NUMBER field is set to RN and the COLUMN_NUMBER field is set to the appropriate column number, if any.

ii) Otherwise, the row *RNR* is fetched and *ROWS_PROCESSED* is incremented by 1 (one).

6.22 FetchScroll

1. Rationale: Editorial.

Replace General Rule 18) ii) with:

18) Let *RS* be the number of rows in *NR*. For *RN* ranging from 1 (one) to *RS*:

a) Let *R* be the *RN*-th row of *NR*. Let *ROWS_PROCESSED* be 0 (zero).

Case:

i) If an exception condition is raised during derivation of any <derived column> associated with *R* and *ASP* is not a null pointer, then set the *RN*-th element of *ASP* to 5 (indicating **Row error**). For all diagnostic records that result from the application of this Rule, the ROW_NUMBER field is set to *RN* and the COLUMN_NUMBER field is set to the appropriate column number, if any.

ii) Otherwise the row *R* is fetched and *ROWS_PROCESSED* is incremented by 1 (one).

6.23 ForeignKeys

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
ForeignKeys (
     StatementHandle      IN  INTEGER,
     PKCatalogName        IN  CHARACTER(L1),
     NameLength1          IN  SMALLINT,
     PKSchemaName         IN  CHARACTER(L2),
     NameLength2          IN  SMALLINT,
     PKTableName          IN  CHARACTER(L3),
     NameLength3          IN  SMALLINT,
     FKCatalogName        IN  CHARACTER(L4),
     NameLength4          IN  SMALLINT,
     FKSchemaName         IN  CHARACTER(L5),
     NameLength5          IN  SMALLINT,
     FKTableName          IN  CHARACTER(L6),
     NameLength6          IN  SMALLINT )
     RETURNS SMALLINT
```

where each of *L1*, *L2*, *L3*, *L4*, *L5*, and *L6* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.28 GetConnectAttr

1. Rationale: Remove the anomaly in <savepoint specifier>.

Delete General Rule 6)

6.29 GetCursorName

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
GetCursorName (
      StatementHandle       IN  INTEGER,
      CursorName            OUT CHARACTER(L),
      BufferLength          IN  SMALLINT,
      NameLength            OUT SMALLINT )
      RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.30 GetData

1. Rationale: Separating data type correspondence tables in Foundation and CLI.

Replace General Rule 17) with:

17) Let *HL* be the standard programming language of the invoking host program. Let *operative data type correspondence table* be the data type correspondence table for *HL* as specified in Subclause 5.15, "SQL/CLI data type correspondences". Refer to the two columns of the *operative data type correspondence table* as the *SQL data type column* and the *host data type column*.

6.32 GetDescRec

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
GetDescRec (
      DescriptorHandle      IN  INTEGER,
      RecordNumber          IN  SMALLINT,
      Name                  OUT CHARACTER(L),
      BufferLength          IN  SMALLINT,
      NameLength            OUT SMALLINT,
      Type                  OUT SMALLINT,
      SubType               OUT SMALLINT,
      Length                OUT INTEGER,
      Precision             OUT SMALLINT,
      Scale                 OUT SMALLINT,
      Nullable              OUT SMALLINT )
      RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

6.34 GetDiagRec

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
GetDiagRec (
     HandleType          IN  SMALLINT,
     Handle              IN  INTEGER,
     RecordNumber        IN  SMALLINT,
     Sqlstate            OUT CHARACTER(5),
     NativeError         OUT INTEGER,
     MessageText         OUT CHARACTER(L),
     BufferLength        IN  SMALLINT,
     TextLength          OUT SMALLINT )
     RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.36 GetFeatureInfo

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
GetFeatureInfo (
     ConnectionHandle     IN  INTEGER,
     FeatureType          IN  CHARACTER(L1),
     FeatureTypeLength    IN  SMALLINT,
     FeatureId            IN  CHARACTER(L2),
     FeatureIdLength      IN  SMALLINT,
     SubFeatureId         IN  CHARACTER(L3),
     SubFeatureIdLength   IN  SMALLINT,
     Supported            OUT SMALLINT )
     RETURNS SMALLINT
```

where each of *L1*, *L2*, and *L3* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.40 GetParamData

1. Rationale: Separating data type correspondence tables in Foundation and CLI.

Replace General Rule 17) with:

17) Let *HL* be the standard programming language of the invoking host program. Let *operative data type correspondence table* be the data type correspondence table for *HL* as specified in Subclause 5.15, "SQL/CLI data type correspondences". Refer to the two columns of the *operative data type correspondence table* as the *SQL data type column* and the *host data type column*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.50 Prepare

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
Prepare (
     StatementHandle       IN  INTEGER,
     StatementText         IN  CHARACTER(L),
     TextLength            IN  INTEGER )
     RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.51 PrimaryKeys

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
PrimaryKeys (
     StatementHandle       IN  INTEGER,
     CatalogName           IN  CHARACTER(L1),
     NameLength1           IN  SMALLINT,
     SchemaName            IN  CHARACTER(L2),
     NameLength2           IN  SMALLINT,
     TableName             IN  CHARACTER(L3),
     NameLength3           IN  SMALLINT )
     RETURNS SMALLINT
```

where each of *L1*, *L2*, and *L3* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.54 SetConnectAttr

1. Rationale: Remove the anomaly in <savepoint specifier>.

Delete General Rule 5)

6.55 SetCursorName

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
SetCursorName (
     StatementHandle       IN  INTEGER,
     CursorName            IN  CHARACTER(L),
     NameLength            IN  SMALLINT )
     RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

6.56 SetDescField

1. Rationale: Editorial - typographic error.

Replace General Rule 13) i) iii) 2) with:

13) i) iii) 2) Otherwise, let *FV* be the first *L* octets of Value and let *TFV* be the value of

```
TRIM ( BOTH ' ' FROM FV )
```

2. Rationale: Editorial - typographic error.

Replace General Rule 14) a) with:

14) a) If *RI* is 1 (one) and Value is not 0 (zero), then an exception condition is raised: *dynamic SQL error — invalid LEVEL value.*

3. Rationale: Editorial - typographic error.

Replace General Rule 14) b) i) with:

14) b) i) If Value is *K*+1 and TYPE in *PIDA* does not indicate ROW, ARRAY, or ARRAY LOCATOR, then an exception condition is raised: *dynamic SQL error — invalid LEVEL value.*

6.60 SpecialColumns

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
SpecialColumns (
     StatementHandle       IN  INTEGER,
     IdentifierType        IN  SMALLINT,
     CatalogName           IN  CHARACTER(L1),
     NameLength1           IN  SMALLINT,
     SchemaName            IN  CHARACTER(L2),
     NameLength2           IN  SMALLINT,
     TableName             IN  CHARACTER(L3),
     NameLength3           IN  SMALLINT,
     Scope                 IN  SMALLINT,
     Nullable              IN  SMALLINT )
     RETURNS SMALLINT
```

where each of *L1*, *L2*, and *L3* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.62 TablePrivileges

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
TablePrivileges (
     StatementHandle       IN  INTEGER,
     CatalogName           IN  CHARACTER(L1),
     NameLength1           IN  SMALLINT,
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
SchemaName              IN  CHARACTER(L2),
NameLength2             IN  SMALLINT,
TableName               IN  CHARACTER(L3),
NameLength3             IN  SMALLINT )
RETURNS SMALLINT
```

where each of *L1*, *L2*, and *L3* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.63 Tables

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
Tables (
     StatementHandle        IN  INTEGER,
     CatalogName            IN  CHARACTER(L1),
     NameLength1            IN  SMALLINT,
     SchemaName             IN  CHARACTER(L2),
     NameLength2            IN  SMALLINT,
     TableName              IN  CHARACTER(L3),
     NameLength3            IN  SMALLINT,
     TableType              IN  CHARACTER(L4),
     NameLength4            IN  SMALLINT )
     RETURNS SMALLINT
```

where each of *L1*, *L2*, *L3*, and *L4* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

A.1 C header file SQLCLI.H

1. Rationale: Remove the anomaly in <savepoint specifier>.

Delete the lines:

```
#define SQL_ATTR_SAVEPOINT_NUMBER                10028
#define SQL_SAVEPOINT_NUMBER_COMMIT                  3
#define SQL_SAVEPOINT_NUMBER_RELEASE                 5
```

2. Rationale: Correct the specification of which locators are marked invalid when an SQL-transaction ends.

Replace the line:

```
#define SQL_SAVEPOINT_NAME_COMMIT                    2
```

with:

```
#define SQL_SAVEPOINT_NAME_ROLLBACK                  2
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

A.2 COBOL library item SQLCLI

1. Rationale: Remove the anomaly in <savepoint specifier>.

Delete the line:

```
01 SQL-ATTR-SAVEPOINT-NUMBER PIC S9(9) BINARY VALUE IS 10028.
```

2. Rationale: Editorial - consistency with Annex A.1.

Replace the lines:

```
01 SQL-BIT-ARRAY PIC S9(4) BINARY VALUE IS 50.
01 SQL-BIT-ARRAY-LOCATOR PIC S9(4) BINARY VALUE IS 51.
```

with:

```
01 SQL-ARRAY PIC S9(4) BINARY VALUE IS 50.
01 SQL-ARRAY-LOCATOR PIC S9(4) BINARY VALUE IS 51.
```

3. Rationale: Correct the specification of which locators are marked invalid when an SQL-transaction ends.

Replace the line:

```
01 SQL-SAVEPOINT-NAME-COMMIT PIC S9(4) BINARY VALUE IS 2.
```

with:

```
01 SQL-SAVEPOINT-NAME-ROLLBACK PIC S9(4) BINARY VALUE IS 2.
```

4. Rationale: Remove the anomaly in <savepoint specifier>.

Delete the lines:

```
01 SQL-SAVEPOINT-NUMBER-COMMIT PIC S9(4) BINARY VALUE IS 3.
01 SQL-SAVEPOINT-NUMBER-RELEASE PIC S9(4) BINARY VALUE IS 5.
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ISO/IEC 9075-4:1999
Database Languages - SQL-Part 4:Persistent Stored Modules

3.3.2.1 Clause, Subclause, and Table relationships

1. Rationale: Correct the classification of SQL-statements.

Add the following row to Table 1, "Clause, Subclause, and Table relationships":

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
4.10.2a Preparable and immediately executable SQL-statements	4.6.5 Preparable and immediately executable SQL-statements	ISO/IEC 9075-5

4.8 Cursors

1. Rationale: Fix bug in processing results sets in PSM.

Replace the 1st paragraph with:

Insert this paragraph For every <declare cursor> in a <compound statement>, a cursor is effectively created each time the <compound statement> is executed, and destroyed when that execution completes, unless the cursor is an open result set cursor.

4.10 SQL-statements

1. Rationale: Correct the classification of SQL-statements.

Add the following Subclause after Subclause 4.10.2, "Embeddable SQL-statements":

4.10.2a Preparable and immediately executable SQL-statements

Insert this paragraph Consequently, the following SQL-control statements are not preparable:

— <compound statement>

— <case statement>

— <if statement>

— <iterate statement>

— <leave statement>

— <loop statement>

— <while statement>

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Previous page is blank.

Not for Resale

— <repeat statement>

— <for statement>

— <assignment statement>

Insert this paragraph Consequently, the following SQL-control declarations are not preparable:

— <condition declaration>

— <handler declaration>

— <SQL variable declaration>

4.10.1 SQL-statements classified by function

1. Rationale: Correct the classification of SQL-statements.

Delete the 2nd bullet from the 2nd paragraph.

5.1 <token> and <separator>

1. Rationale: Editorial - Correct reserved and non-reserved word lists.

In the Format, in the production for <reserved word> delete the alternatives:

| REDO

6.2 <identifier chain>

1. Rationale: Correct definition of possible scope tags.

Replace Syntax Rule 2) with:

2) Replace SR7)a) If $N = 1$, then *IC* shall be contained within the scope of one or more exposed <table or query name>s or <correlation name>s whose associated tables include a column whose <identifier> is equivalent to I_1 or within the scope of a <routine name> whose associated <SQL parameter declaration list> includes an SQL parameter whose <identifier> is equivalent to I_1 or within the scope of one or more <beginning label>s whose associated <local declaration list> includes an SQL variable whose <identifier> is equivalent to I_1. Let the phrase possible scope tags denote those exposed <table or query name>s, <correlation name>s, <routine name>s and <beginning label>s.

3) Insert after SR7)a)i)2) If the innermost possible scope tag is a <beginning label>, then let *SV* be the SQL variable whose <identifier> is equivalent to I_1. PIC_1 is the basis of *IC*, the basis length is 1 (one), the basis scope is the scope of *SP*, and the basis referent is *SV*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

9.18 <SQL-server module definition>

1. Rationale: Editorial.

In the Format, replace the production <SQL-server module definition> with:

```
<SQL-server module definition> ::=
     CREATE MODULE <SQL-server module name>
     [ <SQL-server module character set specification> ]
     [ <SQL-server module schema clause> ]
     [ <SQL-server module path specification> ]
     [ <temporary table declaration>... ]
     <SQL-server module contents>...
     END MODULE
```

11.2 <SQL procedure statement>

1. Rationale: Editorial - to correct reference.

Replace Conformance Rule 1) with:

1) Without Feature P001, "Stored modules", an <SQL procedure statement> shall not be an <SQL server module definition> or a <drop module statement>.

12.2 <fetch statement>

1. Rationale: Replace misleading text.

Replace Syntax Rules 2) and 3) with:

2) Replace SR6)a)i) If *TS* is an <SQL variable reference> or the <SQL parameter name> of an SQL parameter of an SQL-invoked routine, then the Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9072-2, apply to *TS* and the row type of table *T* as *TARGET* and *VALUE*, respectively.

3) Replace SR6)b)ii) For each <target specification> *TS1* that is an <SQL variable reference> or the <SQL parameter name> of an SQL parameter of an SQL-invoked routine, the Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9072-2, apply to *TS1* and the corresponding column of table *T* as *TARGET* and *VALUE*, respectively.

Replace General Rules 1) and 2) with:

1) Replace GR7)a)i) If *TS* is an <SQL variable reference> or an <SQL parameter reference>, then the General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9072-2, apply to *TS* and the current row as *TARGET* and *VALUE*, respectively.

2) Replace GR7)b)i) If *TV* is an <SQL variable reference> or an <SQL parameter reference>, then the General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9072-2, apply to *TS* and *SV* as *TARGET* and *VALUE*, respectively.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

13.1 <compound statement>

1. Rationale: Editorial.

Replace General Rule 3) b) with:

3) b) The General Rules of Subclause 13.5, "<SQL procedure statement>", in ISO 9075-2, are evaluated with S_1 as the *executing statement*.

13.6 <resignal statement>

1. Rationale: Editorial.

Replace General Rule 3) a) ii) with:

3) a) ii) If a handler *H* is the most appropriate handler for *S*, then *H* is activated.

13.7 <if statement>

1. Rationale: Use the correct form of containment

Replace Syntax Rule 1) with:

1) If one or more <if statement elseif clause>s are specified, then the <if statement> is equivalent to an <if statement> that does not contain ELSEIF by performing the following transformation recursively:

```
IF <search condition>
  <if statement then clause>
  <if statement elseif clause 1>
  [ <if statement elseif clause>... ]
  [ <if statement else clause> ]
END IF
```

is equivalent to

```
IF <search condition>
  <if statement then clause>
ELSE
  IF <search condition 1>
    THEN <statement list 1>
    [ <if statement elseif clause>... ]
    [ <if statement else clause> ]
  END IF
END IF
```

where <search condition 1> is the <search condition> simply contained in <if statement elseif clause 1> and <statement list 1> is the <SQL statement list> simply contained in <if statement elseif clause 1>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.1 <embedded SQL host program>

1. Rationale: Remove redundant rule that refers to a non-existent BNF term.

Delete Syntax Rule 4).

16.1 <get diagnostics statement>

1. Rationale: Correct the classification of SQL-statements.

Replace the following rows from Table 3, "SQL-statement codes for use in the diagnostics area":

SQL-statement	Identifier	Code
<temporary table declaration>	TEMPORARY TABLE	93
<alter module statement>	ALTER MODULE	95

with:

SQL-statement	Identifier	Code
<iterate statement>	ITERATE	102

16.3 <resignal statement>

1. Rationale: Editorial.

Replace General Rule 2) a) with:

2) a) If the first condition in the diagnostics area has no RETURNED_SQLSTATE value and the value of the CONDITION_IDENTIFIER is a zero-length string, then an exception condition is raised: *resignal when handler not active.*

17.2 MODULE_PRIVILEGES view

1. Rationale: Editorial.

In the Definition, replace the view definition with:

```
CREATE VIEW MODULE_PRIVILEGES AS
  SELECT GRANTOR, GRANTEE, MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
         PRIVILEGE_TYPE, IS_GRANTABLE
    FROM DEFINITION_SCHEMA.MODULE_PRIVILEGES
   WHERE ( GRANTEE IN ( 'PUBLIC', CURRENT_USER )
           OR
           GRANTEE = CURRENT_USER )
         AND
         MODULE_CATALOG = ( SELECT CATALOG_NAME
                              FROM INFORMATION_SCHEMA_CATALOG_NAME );
```

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

17.4 MODULES view

1. Rationale: Editorial.

In the Definition, replace the view definition with:

```
CREATE VIEW MODULES AS
  SELECT MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
         DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA,
         DEFAULT_CHARACTER_SET_NAME, DEFAULT_SCHEMA_CATALOG,
         DEFAULT_SCHEMA_NAME,
         CASE WHEN EXISTS ( SELECT *
                              FROM DEFINITION_SCHEMA.SCHEMATA AS S
                             WHERE ( MODULE_CATALOG, MODULE_SCHEMA )
                                     = ( S.CATALOG_NAME, S.SCHEMA_NAME )
                                   AND
                                   ( SCHEMA_OWNER IN ( 'PUBLIC',
                                                       CURRENT_USER )
                                     OR
                                     SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                                         FROM_ENABLED_ROLES
                                                     )
                                   )
                          )
              THEN MODULE_DEFINITION
            ELSE NULL
          END AS MODULE_DEFINITION,
         MODULE_AUTHORIZATION, SQL_PATH, MODULE_CREATED,
         MODULE_LAST_ALTERED
    FROM DEFINITION_SCHEMA.MODULES
   WHERE ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME )
           IN ( SELECT MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME
                  FROM DEFINITION_SCHEMA.MODULE_PRIVILEGES
                 WHERE ( SCHEMA_OWNER IN ( 'PUBLIC', CURRENT_USER )
                         OR
                         SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                             FROM ENABLED_ROLES ) ) )
         AND
         MODULE_CATALOG = ( SELECT CATALOG_NAME
                              FROM INFORMATION_SCHEMA_CATALOG_NAME );
```

Annex A SQL Conformance Summary

1. Rationale: Editorial - to correct reference.

Replace Point 1) with:

1) Specifications for Feature P001, "Stored modules":

   a) Subclause 11.2, "<SQL procedure statement>":

      i) Without Feature P001, "Stored modules", an <SQL procedure statement> shall not be an <SQL server module definition> or a <drop module statement>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Annex E Incompatibilities with ISO/IEC 9075:1992

1. Rationale: Editorial.

In Point 1) delete the following bullet:

— REDO

Annex F SQL Feature Taxonomy

1. Rationale: Editorial - to correct reference.

Replace the 4th paragraph, in order to correct the reference to the column, as follows:

The "Feature Name" column contains a brief description of the feature or subfeature associated with the Feature ID value.

2. Rationale: Editorial - to delete incorrect text.

Delete the 5th paragraph.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0860116 3T3

ISO/IEC 9075-5:1999
Database Languages - SQL-Part 5:Bindings

4.6.1 Classes of SQL-statements

1. Rationale: Clarify the semantics of SQL-data access indication.

Replace the 2nd paragraph with:

Insert this paragraph There are at least four additional ways of classifying SQL-statements:

— According to whether or not they may be embedded.

— According to whether they may be dynamically prepared and executed.

— According to whether or not they may be directly executed.

— According to whether they do not possibly contain SQL, possibly contain SQL, possibly read SQL-data, or possibly modify SQL-data.

4.6.4 Embeddable SQL-statement

1. Rationale: Correct the classification of SQL-statements.

Add the following sub-bullet to the 7th bullet of the 1st paragraph:

- <hold locator statement>

2. Rationale: Correct the classification of SQL-statements.

Replace the 8th bullet of the 1st paragraph with:

— The following SQL-control statements:

- <call statement>
- <return statement>

4.6.5 Preparable and immediately executable SQL-statements

1. Rationale: Correct the classification of SQL-statements.

Add the following sub-bullet to the 4th bullet of the 1st paragraph:

- <hold locator statement>

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Previous page is blank.

Not for Resale

2. *Rationale: Correct the classification of SQL-statements.*

Add the following bullet to the 2nd paragraph:

— <return statement>

4.6.6 Directly executable SQL-statements

1. *Rationale: Correct the classification of SQL-statements.*

Add the following bullet to the 1st paragraph:

— The following SQL-control statements:

- <call statement>
- <return statement.

2. *Rationale: Clarify the semantics of SQL-data access indication.*

Add the following Subclause after Subclause 4.6.6, "Directly executable SQL-statements":

4.6.6a SQL-statements and SQL-data access indication

Insert this paragraph The following are the other SQL-statements that possibly contain SQL:

— SQL embedded exception declaration

Insert this paragraph The following are the other SQL-statements that possibly read SQL-data:

— SQL-dynamic statements

5.1 <token> and <separator>

1. *Rationale: Editorial - Correct reserved and non-reserved word lists.*

In the Format, in the production for <non-reserved word> add the alternative:

```
| NESTING
```

2. *Rationale: Editorial. Reserved word in already included in Part 2.*

In the Format, in the production for <reserved word>, delete the text:

```
| DYNAMIC
| NESTING
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

10.5 <SQL-invoked routine>

1. Rationale: Clarify the semantics of SQL-data access indication.

Replace Syntax Rule 1) with:

1) Insert before SR 18) c) It is implementation-defined whether the <SQL routine body> shall not contain an <SQL dynamic statement>.

11.2 Calls to an <externally-invoked procedure>

1. Rationale: Editorial.

In Syntax Rule 1) replace the following constraints:

```
DYNAMIC_SQL_ERROR_UNDEFINED_DATA_TARGET:
     constant SQLSTATE_TYPE := "0700D";
DYNAMIC_SQL_ERROR_UNDEFINED_LEVEL_VALUE:
     constant SQLSTATE_TYPE := "0700E";
```

with:

```
DYNAMIC_SQL_ERROR_INVALID_DATA_TARGET:
     constant SQLSTATE_TYPE := "0700D";
DYNAMIC_SQL_ERROR_INVALID_LEVEL_VALUE:
     constant SQLSTATE_TYPE := "0700E";
```

11.3 <SQL procedure statement>

1. Rationale: Editorial.

In the Format, replace the production for <SQL procedure statement> with:

```
<SQL session statement> ::=
      !! All alternatives from ISO/IEC 9075-2
    | <set catalog statement>
    | <set schema statement>
    | <set names statement>
    | <set path statement>
    | <set transform group statement>
```

12.0 <fetch statement>

1. Rationale: Add missing Syntax and General Rules for <fetch statement>.

Add a new Subclause as follows:

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12.0 <fetch statement>

Function

Position a cursor on a specified row of a table and retrieve values from that row.

Format

No additional Format items.

Syntax Rules

1) Add after SR 6) b) iii) For each <target specification> *TS2* that is an <embedded variable name>, the Syntax Rules of Subclause 9.1, "Retrieval assignment", apply to each *TS2* and the corresponding column of table *T*, as *TARGET* and *VALUE*, respectively.

General Rules

1) Add after GR 7) b) ii) If *TV* is an <embedded variable name>, then the General Rules of Subclause 9.1, "Retrieval assignment" are applied to *TV* and *SV*, as *TARGET* and *VALUE*, respectively.

12.1 <select statement: single row>

1. Rationale: Replace incorrect non-terminal.

Replace Syntax Rule 1).

1) Insert after SR4) For each <target specification> *TS* that is an <embedded variable specification>, then the Syntax Rules of Subclause 9.1, "Retrieval assignment", shall apply to *TS* and the corresponding element of the <select list>, as *TARGET* and *VALUE*, respectively.

2. Rationale: Remove redundant and incorrect rule.

Delete General Rule 1).

3. Rationale: Replace incorrect non-terminal.

Replace General Rule 2).

2) Insert after GR5) For each <target specification> TS that is an <embedded variable specification>, the corresponding value in the row of *Q* is assigned to *TS* according to the General Rules of Subclause 9.1, "Retrieval assignment", as *VALUE* and *TARGET*, respectively. The assignment of values to targets in the <select target list> is in an implementation-dependent order.

12.2 <free locator statement>

1. Rationale: Editorial - Typographical error.

In the Format, replace the production for <locator reference> with:

```
<locator reference> ::=
      !! All alternatives from ISO/IEC 9075-2
    | <embedded variable name>
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

14.3 <set names statement>

1. Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".

Replace Conformance Rule 1) with:

1) Without Feature F761, "Session management" and Feature F461, "Named character sets", conforming SQL language shall not contain any <set names statement>.

15.1 Description of SQL descriptor areas

1. Rationale: Correct definition of the length of <reference type>s.

Replace Syntax Rule 6) n) with:

6) n) TYPE indicates REF, LENGTH is the length in octets for the REF type, and USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are a valid qualified user-defined type name, and SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME are a valid qualified table name.

15.6 <prepare statement>

1. Rationale: handle <dynamic parameter specification>s for <regular expression substring function>.

Add the following General Rule:

6) a) vii.1) If *DP* is either *X1*, *X2* or *X3* in a <string value function> of the form "SUBSTRING (*X1* SIMILAR *X2* ESCAPE *X3*)" then

1) Case:

a) If the declared type of *X1* is CHARACTER, CHARACTER VARYING or CHARACTER LARGE OBJECT, then let *CS* be the character set of *X1*.

b) If the declared type of *X2* is CHARACTER, CHARACTER VARYING or CHARACTER LARGE OBJECT, then let *CS* be the character set of *X1*.

c) If the declared type of *X3* is CHARACTER, CHARACTER VARYING or CHARACTER LARGE OBJECT, then let *CS* be the character set of *X1*.

d) Otherwise, the character set *CS* is undefined

2) If *CS* is defined, then

a) If *DP* is *X1* or *X2*, then *DT* is CHARACTER VARYING (*ML*) with character set *CS*.

b) If *DP* is *X3*, then *DT* is CHARACTER (1) with character set *CS*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

15.8 <describe statement>

1. Rationale: Correct definition of the length of <reference type>s.

Replace General Rule 8) d) ix) with:

8) d) ix) If TYPE indicates a <reference type>, then USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME are set to the <user-defined type name> of the referenced type and qualified name of the referenceable base table; LENGTH and OCTET_LENGTH are set to the length in octets of the <reference type>.

15.13 <dynamic declare cursor>

1. Rationale: Editorial - typographical errors.

Replace Conformance Rule 6) with:

6) Without Feature B031, "Basic dynamic SQL", and Feature F431 "Read-only scrollable cursors", and Feature F831, "Full cursor update", if an <updatability clause> of FOR UPDATE with or without a <column name list> is specified, then <cursor scrollability> shall not be specified.

2. Rationale: Editorial - typographical errors.

Replace Conformance Rule 8) with:

8) Without Feature B031, "Basic dynamic SQL", and Feature F431 "Read-only scrollable cursors", a <dynamic declare cursor> shall not specify <cursor scrollability>.

15.14 <allocate cursor statement>

1. Rationale: Ensure that the cursor is positioned in same place in returned result set as last set in the called procedure.

Replace General Rule 4) g) with:

4) g) Cursor *CR* is placed in the open state

Case:

i) If *CR* is scrollable then, let *CRCN* be the <cursor name> of *CR* in *P*. The position of *CR* in *T* is before the row that would be retrieved if the following SQL-statement were executed in *P*:

```
FETCH NEXT FROM CRCN INTO ...
```

ii) Otherwise, the position of *CR* is before the first row of *T*.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

16.1 <embedded SQL host program>

1. Rationale: Delete redundant syntax alternative.

In the Format, replace the production for <statement or declaration> with:

```
<statement or declaration> ::=
        <declare cursor>
      | <dynamic declare cursor>
      | <temporary table declaration>
      | <embedded authorization declaration>
      | <embedded path specification>
      | <embedded transform group specification>
      | <embedded exception declaration>
      | <handler declaration>
      | <SQL procedure statement>
```

2. Rationale: Correct use of undefined BNF term.

Replace Syntax Rule 9) with:

9) Case:

a) If <embedded transform group specification> is not specified, then an <embedded transform group specification> containing a <multiple group specification> with a <group specification> *GS* for each <host variable definition> that has an associated user-defined type *UDT*, but is not a user-defined locator variable is implicit. The <group name> of *GS* is implementation-defined and its <user-defined type> is *UDT*.

b) If <embedded transform group specification> contains a <single group specification> with a <group name> *GN*, then an <embedded transform group specification> containing a <multiple group specification> with a <group specification> *GS* for each <host variable definition> that has an associated user-defined type *UDT*, but is not a user-defined type locator variable is implicit. The <group name> of *GS* is *GN* and its <user-defined type> is *UDT*.

c) If <embedded transform group specification> contains a <multiple group specification> *MGS*, then an <embedded transform group specification> containing a <multiple group specification> that contains *MGS* extended with a <group specification> *GS* for each <host variable definition> that has an associated user-defined type *UDT*, but is not a user-defined locator variable and the <user-defined type name> of *UDT* is not contained in any <group specification> contained in *MGS* is implicit. The <group name> of *GS* is implementation-defined and its <user-defined type> is *UDT*.

3. Rationale: Correct use of undefined BNF term.

Replace Syntax Rule 10) with:

10) The implicit or explicit <embedded transform group specification> precedes in the text of the <embedded SQL host program> every <host variable definition>.

4. Rationale: Remove redundant rules that refer to a non-existent BNF term.

Delete Syntax Rules 15) and 16).

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5. Rationale: Editorial.

Replace Syntax Rule 22) k) i) 6) G) with:

22) k) i) 6) G) Let GNI_j, 1 (one) $\leq j \leq a$, be the <group name>s corresponding to the <user-defined type name> of TUI_j contained in the <group specification> contained in <embedded transform group specification>. Let GNO_k, 1 (one) $\leq k \leq b$, be the <group name>s corresponding to the <user-defined type name> of TUO_k contained in the <group specification> contained in <embedded transform group specification>. Let $GNIO_l$, 1 (one) $\leq l \leq c$, be the <group name>s corresponding to the <user-defined type name> of $TUIO_l$ contained in the <group specification> contained in <embedded transform group specification>.

6. Rationale: Editorial.

Replace Syntax Rule 22) k) i) 8) with:

22) k) i) 8) The <SQL procedure statement> of *PS* is:

```
BEGIN ATOMIC
    DECLARE SVI1 TUI1 ;
    .
    .
    .
    DECLARE SVIa TUIa ;
    DECLARE SVO1 TUO1 ;
    .
    .
    .
    DECLARE SVOb TUOb ;
    DECLARE SVIO1 TUIO1 ;
    .
    .
    .
    DECLARE SVIOc TUIOc ;
    SET SVI1 = TSIN1 ( CAST ( PNI1 AS TTI1 ));
    .
    .
    .
    SET SVIa = TSINa ( CAST ( PNIa AS TTIa ));
    SET SVIO1 = TSION1 ( CAST ( PNIO1 AS TTIO1 ));
    .
    .
    .
    SET SVIOc = TSIONc (CAST ( PNIOc AS TTIOc ));
    NES;
    SET PNO1 = CAST ( FSON1 ( SVO1 )AS TSO1 );
    .
    .
    .
    SET PNOb = CAST ( FSONb ( SVOb )AS TSOb );
    SET PNIO1 = CAST ( FSION1 ( SVIO1 )AS TSIO1 );
    .
    .
    .
    SET PNIOc = CAST ( FSIONc ( SVIOc )AS TSIOc );
END;
```

7. Rationale: Remove rule referring to a non-existent feature.

Delete General Rule 2).

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

8. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".*

Replace Conformance Rule 2) with:

2) Without Feature F461, "Named character sets", an <embedded SQL declare section> shall not contain an <embedded character set declaration>.

16.3 <embedded SQL Ada program>

1. *Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).*

In the Format, replace the production for <Ada user-defined type locator variable> with:

```
<Ada user-defined type locator variable> ::=
    SQL TYPE IS <user-defined type> AS LOCATOR
```

2. *Rationale: Correct definition of the length of <reference type>s.*

Replace Syntax Rule 5) i) with:

5) i) The syntax

```
SQL TYPE IS <reference type>
```

for a given <Ada host identifier> RTV shall be replaced by

```
RTV : Interfaces.SQL.CHAR(1..<length>)
```

in any <Ada REF variable>, where <length> is the length in octets of the <reference type>.

3. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".*

Replace Conformance Rule 3) with:

3) Without Feature F461, "Named character sets", an <Ada type specification> shall not contain a <character set specification>.

4. *Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).*

Replace Conformance Rule 7) with:

7) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in an <Ada user-defined type locator variable> shall not identify a structured type.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

16.4 <embedded SQL C program>

1. Rationale: Correct definition of the length of <reference type>s.

Replace Syntax Rule 5) n) with:

5) n) The syntax

```
SQL TYPE IS <reference type>
```

for a given <C host identifier> *hvn* shall be replaced by

```
unsigned.char hvn[L]
```

in any <C REF variable>, where *L* is the length in octets of the <reference type>.

2. Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".

Replace Conformance Rule 3) with:

3) Without Feature F461, "Named character sets", a <C variable definition> shall not contain a <character set specification>.

3. Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).

Replace Conformance Rule 7) with:

7) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in an <C user-defined type locator variable> shall not identify a structured type.

16.5 <embedded SQL COBOL program>

1. Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).

In the Format, replace the production for <COBOL user-defined type locator variable> with:

```
<COBOL user-defined type locator variable> ::=
     [ USAGE [ IS ] ] SQL TYPE IS <user-defined type> AS LOCATOR
```

2. Rationale: Correct definition of the length of <reference type>s.

Replace Syntax Rule 6) k) with:

6) k) The syntax

```
SQL TYPE IS <reference type>
```

for a given <COBOL host identifier> *HVN* shall be replaced by

```
01 HVN PICTURE X(L)
```

in any <COBOL REF variable>, where *L* is the length in octets of the <reference type>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

3. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".*

Replace Conformance Rule 3) with:

3) Without Feature F461, "Named character sets", a <COBOL type specification> shall not contain a <character set specification>.

4. *Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).*

Replace Conformance Rule 7) with:

7) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <COBOL user-defined type locator variable> shall not identify a structured type.

16.6 <embedded SQL FORTRAN program>

1. *Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).*

In the Format, replace the production for <Fortran user-defined type locator variable> with:

```
<Fortran user-defined type locator variable> ::=
    SQL TYPE IS <user-defined type> AS LOCATOR
```

2. *Rationale: Correct definition of the length of <reference type>s and the BNF term <Fortran REF variable>.*

Replace Syntax Rule 6) k) with:

6) k) The syntax

```
SQL TYPE IS <reference type>
```

for a given <Fortran host identifier> *HVN* shall be replaced by

```
CHARACTER HVN *<length>
```

in any <Fortran REF variable>, where <length> is the length in octets of the <reference type>.

3. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".*

Replace Conformance Rule 3) with:

3) 3) Without Feature F461, "Named character sets", a <Fortran type specification> shall not contain a <character set specification>.

4. *Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).*

Replace Conformance Rule 7) with:

7) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <Fortran user-defined type locator variable> shall not identify a structured type.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

16.7 <embedded SQL MUMPS program>

1. Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).

In the Format, replace the production for <MUMPS user-defined type locator variable> with:

```
<MUMPS user-defined type locator variable> ::=
    SQL TYPE IS <user-defined type> AS LOCATOR
```

2. Rationale: Correct definition of the length of <reference type>s and the BNF term <MUMPS REF variable>.

Replace Syntax Rule 9) h) with:

9) h) The syntax

```
SQL TYPE IS <reference type>
```

for a given <MUMPS host identifier> *HVN* shall be replaced by

```
VARCHAR HVN L
```

in any <MUMPS REF variable>, where *L* is the length in octets of the <reference type>.

3. Rationale: Add missing conformance rule.

Add Conformance Rule 7):

7) Without Feature F461, "Named character sets", a <MUMPS CLOB variable> shall not contain a <character set specification>.

4. Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).

Replace Conformance Rule 5) with:

7) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <MUMPS user-defined type locator variable> shall not identify a structured type.

16.8 <embedded SQL Pascal program>

1. Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).

In the Format, replace the production for <Pascal user-defined type locator variable> with:

```
<Pascal user-defined type locator variable> ::=
    SQL TYPE IS <user-defined type> AS LOCATOR
```


© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

2. *Rationale: Correct definition of the length of <reference type>s and the BNF term <Pascal REF variable>.*

Replace Syntax Rule 5) l) with:

5) l) The syntax

```
SQL TYPE IS <reference type>
```

for a given <Pascal host identifier> *HVN* shall be replaced by

```
HVN : PACKED ARRAY [1..<length>] of CHAR
```

in any <Pascal REF variable>, where <length> is the length in octets of the <reference type>.

3. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".*

Replace Conformance Rule 3) with:

3) Without Feature F461, "Named character sets", a <Pascal type specification> shall not contain a <character set specification>.

4. *Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).*

Replace Conformance Rule 7) with:

7) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <Pascal user-defined type locator variable> shall not identify a structured type.

16.9 <embedded SQL PL/I program>

1. *Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).*

In the Format, replace the production for <PL/I user-defined type locator variable> with:

```
<PL/I user-defined type locator variable> ::=
     SQL TYPE IS <user-defined type> AS LOCATOR
```

2. *Rationale: Correct definition of the length of <reference type>s and the BNF term <PL/I REF variable>.*

Replace Syntax Rule 5) i) with:

5) i) The syntax

```
SQL TYPE IS <reference type>
```

for a given <PL/I host identifier> *HVN* shall be replaced by

```
DCL HVN CHARACTER(<length>) VARYING
```

in any <PL/I REF variable>, where <length> is the length in octets of the <reference type>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".*

Replace Conformance Rule 3) with:

3) Without Feature F461, "Named character sets", a <PL/I type specification> shall not contain a <character set specification>.

4. *Rationale: Use the correct BNF (<user-defined type> instead of <user-defined type name>).*

Replace Conformance Rule 7) with:

7) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <PL/I user-defined type locator variable> shall not identify a structured type.

19.1 <get diagnostics statement>

1. *Rationale: Editorial. Non-reserved word - DYNAMIC_FUNCTION - not defined in Part 2.*

Add the following General Rule 3):

3) Replace GR 3) p) The values of CONNECTION_NAME and SERVER_NAME are respectively

Case:

i) If COMMAND_FUNCTION or DYNAMIC_FUNCTION identifies an <SQL connection statement>, then the <connection name> and the <SQL-server name> specified by or implied by the <SQL connection statement>.

ii) Otherwise, the <connection name> and <SQL-server name> of the SQL-session in which the condition was raised.

20.1 SQLSTATE

1. *Rationale: Editorial.*

In "Table 10—SQLSTATE class and subclass values", replace the rows:

Category	Condition	Class	Subcondition	Subclass
			undefined DATA target	00D
			undefined LEVEL value	0

with:

Category	Condition	Class	Subcondition	Subclass
			invalid DATA target	00D
			invalid LEVEL value	0

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex A SQL Conformance Summary

1. Rationale: Editorial - typographical errors.

Replace Item 9) o) ii) with:

9) o) ii) Without Feature B031, "Basic dynamic SQL", and Feature F431 "Read-only scrollable cursors", a <dynamic declare cursor> shall not specify <cursor scrollability>.

2. Rationale: Editorial - typographical errors.

Replace Item 9) o) iv) with:

9) o) iv) Without Feature B031, "Basic dynamic SQL", and Feature F431 "Read-only scrollable cursors", and Feature F831, "Full cursor update", if an <updatability clause> of FOR UPDATE with or without a <column name list> is specified, then <cursor scrollability> shall not be specified.

3. Rationale: Editorial - typographical errors.

Replace Item 13) with:

13) Specifications for Feature F431 "Read-only scrollable cursors":

a) Subclause 15.13, "<dynamic declare cursor>":

i) Without Feature B031, "Basic dynamic SQL", and Feature F431 "Read-only scrollable cursors", and Feature F831, "Full cursor update", if an <updatability clause> of FOR UPDATE with or without a <column name list> is specified, then <cursor scrollability> shall not be specified.

4. Rationale: Editorial - typographical errors.

Replace Item 15) with:

15) Specifications for Feature F431 "Read-only scrollable cursors":

a) Subclause 15.13, "<dynamic declare cursor>":

i) Without Feature B031, "Basic dynamic SQL", and Feature F431 "Read-only scrollable cursors", a <dynamic declare cursor> shall not specify <cursor scrollability>.

5. Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets".

Replace Item 16) with:

16) Specifications for Feature F451, "Character set definition":

a) Subclause 14.3, "<set names statement>":

i) Without Feature F761, "Session management" and Feature F461, "Named character sets", conforming SQL language shall not contain any <set names statement>.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) Subclause 16.1, "<embedded SQL host program>":

i) Without Feature F461, "Named character sets", an <embedded SQL declare section> shall not contain an <embedded character set declaration>.

c) Subclause 16.3, "<embedded SQL Ada program>":

i) Without Feature F461, "Named character sets", an <Ada type specification> shall not contain a <character set specification>.

d) Subclause 16.4, "<embedded SQL C program>":

i) Without Feature F461, "Named character sets", a <C variable definition> shall not contain a <character set specification>.

e) Subclause 16.5, "<embedded SQL COBOL program>":

i) Without Feature F461, "Named character sets", a <COBOL type specification> shall not contain a <character set specification>.

f) Subclause 16.6, "<embedded SQL Fortran program>":

i) Without Feature F461, "Named character sets", a <Fortran type specification> shall not contain a <character set specification>.

g) Subclause 16.7, "<embedded SQL MUMPS program>":

i) Without Feature F461, "Named character sets", a <MUMPS CLOB variable> shall not contain a <character set specification>.

h) Subclause 16.8, "<embedded SQL Pascal program>":

i) Without Feature F461, "Named character sets", a <Pascal type specification> shall not contain a <character set specification>.

i) Subclause 16.9, "<embedded SQL PL/I program>":

i) Without Feature F461, "Named character sets", a <PL/I type specification> shall not contain a <character set specification>.

6. *Rationale: Use the correct BNF (<user-defined type name> instead of <user-defined type>).*

Replace Item 26) with:

26) Specifications for Feature S231, "Structured type locators":

a) Subclause 16.3, "<embedded SQL Ada program>":

i) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in an <Ada user-defined type locator variable> shall not identify a structured type.

b) Subclause 16.4, "<embedded SQL C program>":

i) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <C user-defined type locator variable> shall not identify a structured type.

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4851903 0860132 546

c) Subclause 16.5, "<embedded SQL COBOL program>":

i) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <COBOL user-defined type locator variable> shall not identify a structured type.

d) Subclause 16.6, "<embedded SQL Fortran program>":

i) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <Fortran user-defined type locator variable> shall not identify a structured type.

e) Subclause 16.7, "<embedded SQL MUMPS program>":

i) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <MUMPS user-defined type locator variable> shall not identify a structured type.

f) Subclause 16.8, "<embedded SQL Pascal program>":

i) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <Pascal user-defined type locator variable> shall not identify a structured type.

g) Subclause 16.9, "<embedded SQL PL/I program>":

i) Without Feature S231, "Structured type locators", the <user-defined type> simply contained in a <PL/I user-defined type locator variable> shall not identify a structured type.

Annex E Incompatibilities with ISO/IEC 9075:1992

1. Rationale: Editorial.

Replace Point 2) with:

2) A number of additional <reserved word>s have been added to the language. These <reserved word>s are:

— CURRENT_DEFAULT_TRANSFORM_GROUP

— CURRENT_TRANSFORM_GROUP_FOR_TYPE

© ISO/IEC 2000 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

STD.ISO 9075-5-ENGL 1999 ■ 4851903 0829331 T94 ■

INTERNATIONAL STANDARD

ISO/IEC 9075-5

First edition
1999-12-01

Information technology — Database languages — SQL —

Part 5:
Host Language Bindings (SQL/Bindings)

Technologies de l'information — Langages de base de données — SQL —

Partie 5: Liants de langage d'hôte (SQL/Liants)

This material is reproduced from ISO documents under International Organization for Standardization (ISO) Copyright License Number HIS/CC/1996. Not for resale. No part of these ISO documents may be reproduced in any form, electronic retrieval system or otherwise, except as allowed in the copyright law of the country of use, or with the prior written consent of ISO (Case postale 56,1211 Geneva 20, Switzerland, Fax +41 22 734 10 79), IHS or the ISO Licensor's members

Reference number
ISO/IEC 9075-5:1999(E)

© ISO/IEC 1999

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO/IEC 1999

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

© ISO/IEC 1999 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Contents

Page

Foreword ... vii

Introduction ... ix

1 Scope ... 1

2 Normative references ... 3

3 Definitions, notations, and conventions ... 5
3.1 Definitions ... 5
3.2 Notation ... 5
3.3 Conventions ... 5
3.3.1 Use of terms ... 5
3.3.1.1 Other terms ... 5
3.3.2 Relationships to other parts of ISO/IEC 9075 ... 5
3.3.2.1 Clause, Subclause, and Table relationships ... 5
3.4 Object identifier for Database Language SQL ... 12

4 Concepts ... 15
4.1 Catalogs ... 15
4.2 SQL-client modules ... 15
4.3 SQL-invoked routines ... 15
4.4 Locators ... 15
4.5 Cursors ... 15
4.6 SQL-statements ... 16
4.6.1 Classes of SQL-statements ... 16
4.6.2 SQL-statements classified by function ... 16
4.6.3 SQL-statements and transaction states ... 18
4.6.4 Embeddable SQL-statements ... 19
4.6.5 Preparable and immediately executable SQL-statements ... 20
4.6.6 Directly executable SQL-statements ... 22
4.7 Standard programming languages ... 23
4.8 Embedded syntax ... 24
4.9 SQL dynamic statements ... 24
4.10 Direct invocation of SQL ... 27
4.11 Privileges and roles ... 27
4.12 SQL-transactions ... 27

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4.13 SQL-connections . . . 27
4.14 SQL-sessions . . . 28
4.15 Client-server operation . . . 29

5 Lexical elements . . . 31
5.1 <token> and <separator> . . . 31
5.2 <literal> . . . 33
5.3 Names and identifiers . . . 34

6 Scalar expressions . . . 37
6.1 <value specification> and <target specification> . . . 37
6.2 <column reference> . . . 39
6.3 <interval value expression> . . . 40

7 Query expressions . . . 41
7.1 <table reference> . . . 41
7.2 <query specification> . . . 42

8 Additional common elements . . . 43
8.1 <routine invocation> . . . 43

9 Data assignment rules and routine determination . . . 45
9.1 Retrieval assignment . . . 45
9.2 Store assignment . . . 46
9.3 Data types of results of aggregations . . . 47

10 Schema definition and manipulation . . . 49
10.1 <check constraint definition> . . . 49
10.2 <view definition> . . . 50
10.3 <assertion definition> . . . 51
10.4 <trigger definition> . . . 52
10.5 <SQL-invoked routine> . . . 53

11 SQL-client modules . . . 55
11.1 <SQL-client module definition> . . . 55
11.2 Calls to an <externally-invoked procedure> . . . 57
11.3 <SQL procedure statement> . . . 59

12 Data manipulation . . . 61
12.1 <select statement: single row> . . . 61
12.2 <free locator statement> . . . 62

13 Transaction management . . . 63
13.1 <commit statement> . . . 63

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

14 Session management 65
14.1 <set catalog statement> 65
14.2 <set schema statement> 66
14.3 <set names statement> 67
14.4 <set path statement> 68
14.5 <set transform group statement> 69

15 Dynamic SQL 71
15.1 Description of SQL descriptor areas 71
15.2 <allocate descriptor statement> 80
15.3 <deallocate descriptor statement> 82
15.4 <get descriptor statement> 83
15.5 <set descriptor statement> 86
15.6 <prepare statement> 91
15.7 <deallocate prepared statement> 100
15.8 <describe statement> 101
15.9 <input using clause> 107
15.10 <output using clause> 111
15.11 <execute statement> 116
15.12 <execute immediate statement> 118
15.13 <dynamic declare cursor> 119
15.14 <allocate cursor statement> 121
15.15 <dynamic open statement> 123
15.16 <dynamic fetch statement> 124
15.17 <dynamic single row select statement> 125
15.18 <dynamic close statement> 126
15.19 <dynamic delete statement: positioned> 127
15.20 <dynamic update statement: positioned> 129
15.21 <preparable dynamic delete statement: positioned> 131
15.22 <preparable dynamic update statement: positioned> 132

16 Embedded SQL 133
16.1 <embedded SQL host program> 133
16.2 <embedded exception declaration> 145
16.3 <embedded SQL Ada program> 149
16.4 <embedded SQL C program> 154
16.5 <embedded SQL COBOL program> 161
16.6 <embedded SQL Fortran program> 167
16.7 <embedded SQL MUMPS program> 172
16.8 <embedded SQL Pascal program> 176
16.9 <embedded SQL PL/I program> 181

17 Direct invocation of SQL 187
17.1 <direct SQL statement> 187
17.2 <direct select statement: multiple rows> 191

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

18 Diagnostics management 193
18.1 <get diagnostics statement> 193

19 Definition Schema 197
19.1 SQL_LANGUAGES base table 197

20 Status codes 199
20.1 SQLSTATE 199

21 Conformance 201
21.1 General conformance requirements 201
21.2 Claims of conformance 201

Annex A SQL Conformance Summary 203

Annex B Implementation-defined elements 217

Annex C Implementation-dependent elements 221

Annex D Deprecated features 223

Annex E Incompatibilities with ISO/IEC 9075:1992 225

Annex F SQL feature and package taxonomy 227

Index 229

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

TABLES

Tables **Page**

1	Clause, Subclause, and Table relationships	5
2	Data types of <key word>s used in the header of SQL descriptor areas	75
3	Data types of <key word>s used in SQL item descriptor areas	75
4	Codes used for SQL data types in Dynamic SQL	77
5	Codes associated with datetime data types in Dynamic SQL	78
6	Codes used for <interval qualifier>s in Dynamic SQL	78
7	Codes used for input/output SQL parameter modes in Dynamic SQL	79
8	<identifier>s for use with <get diagnostics statement>	193
9	SQL-statement codes	194
10	SQLSTATE class and subclass values	199
11	Implied feature relationships	201
12	SQL/Bindings feature taxonomy and definition for Core SQL	228
13	SQL/Bindings feature taxonomy for features outside Core SQL	228

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

International Standard ISO/IEC 9075-5 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

ISO/IEC 9075 consists of the following parts, under the general title *Information technology — Database languages — SQL*:

— *Part 1: Framework (SQL/Framework)*

— *Part 2: Foundation (SQL/Foundation)*

— *Part 3: Call-Level Interface (SQL/CLI)*

— *Part 4: Persistent Stored Modules (SQL/PSM)*

— *Part 5: Host Language Bindings (SQL/Bindings)*

Annexes A, B, C, D, E, and F of this part of ISO/IEC 9075 are for information only.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Introduction

The organization of this part of ISO/IEC 9075 is as follows:

1) Clause 1, "Scope", specifies the scope of this part of ISO/IEC 9075.

2) Clause 2, "Normative references", identifies additional standards that, through reference in this part of ISO/IEC 9075, constitute provisions of this part of ISO/IEC 9075.

3) Clause 3, "Definitions, notations, and conventions", defines the notations and conventions used in this part of ISO/IEC 9075.

4) Clause 4, "Concepts", presents concepts used in the definition of Persistent SQL modules.

5) Clause 5, "Lexical elements", defines the lexical elements of the language.

6) Clause 6, "Scalar expressions", defines the elements of the language that produce scalar values.

7) Clause 7, "Query expressions", defines the elements of the language that produce rows and tables of data.

8) Clause 8, "Additional common elements", defines additional language elements that are used in various parts of the language.

9) Clause 9, "Data assignment rules and routine determination", specifies the rules for assignments that retrieve data from or store data into SQL-data, and formation rules for set operations.

10) Clause 10, "Schema definition and manipulation", defines facilities for creating and managing a schema.

11) Clause 11, "SQL-client modules", defines modules and procedures.

12) Clause 12, "Data manipulation", defines the data manipulation statements.

13) Clause 13, "Transaction management", defines the SQL-transaction management statements.

14) Clause 14, "Session management", defines the SQL-session management statements.

15) Clause 15, "Dynamic SQL", defines the SQL dynamic statements.

16) Clause 16, "Embedded SQL", defines the host language embeddings.

17) Clause 17, "Direct invocation of SQL", defines direct invocation of SQL language.

18) Clause 18, "Diagnostics management", defines the diagnostics management facilities.

19) Clause 19, "Definition Schema", defines base tables on which the viewed tables containing schema information depend.

20) Clause 20, "Status codes", defines values that identify the status of the execution of SQL-statements and the mechanisms by which those values are returned.

21) Clause 21, "Conformance", defines the criteria for conformance to this part of ISO/IEC 9075.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

22) Annex A, "SQL Conformance Summary", is an informative Annex. It summarizes the conformance requirements of the SQL language.

23) Annex B, "Implementation-defined elements", is an informative Annex. It lists those features for which the body of this part of ISO/IEC 9075 states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or any other behavior is partly or wholly implementation-defined.

24) Annex C, "Implementation-dependent elements", is an informative Annex. It lists those features for which the body of this part of ISO/IEC 9075 states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or any other behavior is partly or wholly implementation-dependent.

25) Annex D, "Deprecated features", is an informative Annex. It lists features that the responsible Technical Committee intends will not appear in a future revised version of ISO/IEC 9075.

26) Annex E, "Incompatibilities with ISO/IEC 9075:1992", is an informative Annex. It lists the incompatibilities between this version of ISO/IEC 9075 and ISO/IEC 9075:1992.

27) Annex F, "SQL feature and package taxonomy", is an informative Annex. It identifies features of the SQL language specified in this part of ISO/IEC 9075 by a numeric identifier and a short descriptive name. This taxonomy is used to specify conformance to Core SQL and may be used to develop other profiles involving the SQL language.

In the text of this part of ISO/IEC 9075, Clauses begin a new odd-numbered page, and in Clause 5, "Lexical elements", through Clause 21, "Conformance", Subclauses begin a new page. Any resulting blank space is not significant.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Information technology — Database languages — SQL —

Part 5:
Host Language Bindings (SQL/Bindings)

1 Scope

This part of ISO/IEC 9075 specifies:

— Syntax for embedding SQL-statements in a compilation unit that otherwise conforms to the standard for a particular programming language (host language).

— How an equivalent compilation unit may be derived that conforms to the particular programming language standard. In that equivalent compilation unit, each embedded SQL-statement has been replaced by one or more statements in the host language, some of which invoke an SQL externally-invoked procedure that, when executed, has an effect equivalent to executing the SQL-statement.

— Syntax for direct invocation of SQL-statements.

— SQL language to support dynamic preparation and execution of SQL-statements.

--`,,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

--`,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2 Normative references

Insert this paragraph The following standards contain provisions that, through reference in this text, constitute provisions of this part of ISO/IEC 9075. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 9075 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 1539-1:1997, *Information technology — Programming languages — Fortran — Part 1: Base language.*

ISO 1989:1985, *Programming languages — COBOL.*
(Endorsement of ANSI X3.23-1985).

ISO 6160:1979, *Programming languages — PL/I*
(Endorsement of ANSI X3.53-1976).

ISO/IEC 7185:1990, *Information technology — Programming languages—Pascal.*

ISO 8652:1995, *Information technology — Programming languages — Ada.*

ISO/IEC 9075-1, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework).*

ISO/IEC 9075-2, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation).*

ISO/IEC 9075-3:1995, *Information technology — Database languages — SQL — Part 3: Call-Level Interface (SQL/CLI).*

ISO/IEC 9075-4, *Information technology — Database languages — SQL — Part 4: Persistent Stored Modules (SQL/PSM).*

ISO/IEC 9899:1990, *Programming languages — C.*

ISO/IEC 9899:1990/Amendment 1:1995, *Amendment to ISO/IEC 9899:1990 — C Integrity.*

ISO/IEC 10206:1991, *Information technology — Programming languages—Extended Pascal.*

ISO/IEC 11756:1992, *Information technology—Programming languages — MUMPS.*

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Information technology — Database languages — SQL — Part 5: Host Language Bindings (SQL/Bindings)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3 Definitions, notations, and conventions

3.1 Definitions

Insert this paragraph For the purposes of this part of ISO/IEC 9075, the definitions given in ISO/IEC 9075-1 and ISO/IEC 9075-2 apply.

3.2 Notation

Insert this paragraph The notation used in this part of ISO/IEC 9075 is defined in ISO/IEC 9075-1.

3.3 Conventions

Insert this paragraph The conventions used in this part of ISO/IEC 9075 is defined in ISO/IEC 9075-1.

3.3.1 Use of terms

3.3.1.1 Other terms

Insert this paragraph An SQL-statement *S1* is said to be executed as a *direct result of executing an SQL-statement* if *S1* is the value of an <SQL statement variable> referenced by an <execute immediate statement> contained in an <externally-invoked procedure> that has been executed, or if *S1* was the value of the <SQL statement variable> that was associated with an <SQL statement name> by a <prepare statement> and that same <SQL statement name> is referenced by an <execute statement> contained in an <externally-invoked procedure> that has been executed.

3.3.2 Relationships to other parts of ISO/IEC 9075

3.3.2.1 Clause, Subclause, and Table relationships

Table 1—Clause, Subclause, and Table relationships

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Clause 1, “Scope”	Clause 1, "Scope"	ISO/IEC 9075-2
Clause 2, “Normative references”	Clause 2, "Normative references"	ISO/IEC 9075-2
Clause 3, “Definitions, notations, and conventions”	Clause 3, "Definitions, notations, and conventions"	ISO/IEC 9075-2

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 3.1, “Definitions”	Subclause 3.1, "Definitions"	ISO/IEC 9075-1
Subclause 3.2, “Notation”	Subclause 6.1, "Notation"	ISO/IEC 9075-1
Subclause 3.3, “Conventions”	Subclause 6.2, "Conventions"	ISO/IEC 9075-1
Subclause 3.3.1, “Use of terms”	Subclause 6.2.3, "Use of terms"	ISO/IEC 9075-1
Subclause 3.3.1.1, “Other terms”	Subclause 6.2.3.7, "Other terms"	ISO/IEC 9075-1
Subclause 3.3.2, “Relationships to other parts of ISO/IEC 9075”	*(none)*	*(none)*
Subclause 3.3.2.1, “Clause, Subclause, and Table relationships”	*(none)*	*(none)*
Subclause 3.4, “Object identifier for Database Language SQL”	Subclause 6.3, "Object identifier for Database Language SQL"	ISO/IEC 9075-1
Clause 4, “Concepts”	Clause 4, "Concepts"	ISO/IEC 9075-2
Subclause 4.1, “Catalogs”	Subclause 4.2.6.1, "Catalogs"	ISO/IEC 9075-1
Subclause 4.2, “SQL-client modules”	Subclause 4.21, "SQL-client modules"	ISO/IEC 9075-2
Subclause 4.3, “SQL-invoked routines”	Subclause 4.23, "SQL-invoked routines"	ISO/IEC 9075-2
Subclause 4.4, “Locators”	Subclause 4.26.4, "Locators"	ISO/IEC 9075-2
Subclause 4.5, “Cursors”	Subclause 4.29, "Cursors"	ISO/IEC 9075-2
Subclause 4.6, “SQL-statements”	Subclause 4.30, "SQL-statements"	ISO/IEC 9075-2
Subclause 4.6.1, “Classes of SQL-statements”	Subclause 4.30.1, "Classes of SQL-statements"	ISO/IEC 9075-2
Subclause 4.6.2, “SQL-statements classified by function”	Subclause 4.30.2, "SQL-statements classified by function"	ISO/IEC 9075-2
Subclause 4.6.3, “SQL-statements and transaction states”	Subclause 4.30.3, "SQL-statements and transaction states"	ISO/IEC 9075-2
Subclause 4.6.4, “Embeddable SQL-statements”	*none*	*none*
Subclause 4.6.5, “Preparable and immediately executable SQL-statements”	*none*	*none*
Subclause 4.6.6, “Directly executable SQL-statements”	*none*	*none*
Subclause 4.7, “Standard programming languages”	Subclause 4.28, "Standard programming languages"	ISO/IEC 9075-2
Subclause 4.8, “Embedded syntax”	*none*	*none*

--`,,`--`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 4.9, “SQL dynamic statements”	*none*	*none*
Subclause 4.10, “Direct invocation of SQL”	*none*	*none*
Subclause 4.11, “Privileges and roles”	Subclause 4.31, "Basic security model"	ISO/IEC 9075-2
Subclause 4.12, “SQL-transactions”	Subclause 4.32, "SQL-transactions"	ISO/IEC 9075-2
Subclause 4.13, “SQL-connections”	Subclause 4.33, "SQL-connections"	ISO/IEC 9075-2
Subclause 4.14, “SQL-sessions”	Subclause 4.34, "SQL-sessions"	ISO/IEC 9075-2
Subclause 4.15, “Client-server operation”	Subclause 4.36, "Client-server operation"	ISO/IEC 9075-2
Clause 5, “Lexical elements”	Clause 5, "Lexical elements"	ISO/IEC 9075-2
Subclause 5.1, “<token> and <separator>”	Subclause 5.2, "<token> and <separator>"	ISO/IEC 9075-2
Subclause 5.2, “<literal>”	Subclause 5.3, "<literal>"	ISO/IEC 9075-2
Subclause 5.3, “Names and identifiers”	Subclause 5.4, "Names and identifiers"	ISO/IEC 9075-2
Clause 6, “Scalar expressions”	Clause 6, "Scalar expressions"	ISO/IEC 9075-2
Subclause 6.1, “<value specification> and <target specification>”	Subclause 6.3, "<value specification> and <target specification>"	ISO/IEC 9075-2
Subclause 7.1, “<table reference>”	Subclause 7.6, "<table reference>"	ISO/IEC 9075-2
Subclause 6.2, “<column reference>”	Subclause 6.6, "<column reference>"	ISO/IEC 9075-2
Subclause 6.3, “<interval value expression>”	Subclause 6.29, "<interval value expression>"	ISO/IEC 9075-2
Clause 7, “Query expressions”	Clause 7, "Query expressions"	ISO/IEC 9075-2
Subclause 7.2, “<query specification>”	Subclause 7.11, "<query specification>"	ISO/IEC 9075-2
Clause 8, “Additional common elements”	Clause 10, "Additional common elements"	ISO/IEC 9075-2
Subclause 8.1, “<routine invocation>”	Subclause 10.4, "<routine invocation>"	ISO/IEC 9075-2
Clause 9, “Data assignment rules and routine determination”	Clause 9, "Data assignment rules and routine determination"	ISO/IEC 9075-2
Subclause 9.1, “Retrieval assignment”	Subclause 9.1, "Retrieval assignment"	ISO/IEC 9075-2
Subclause 9.2, “Store assignment”	Subclause 9.2, "Store assignment"	ISO/IEC 9075-2

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Clause 10, “Schema definition and manipulation”	Clause 11, "Schema definition and manipulation"	ISO/IEC 9075-2
Subclause 10.1, “<check constraint definition>”	Subclause 11.9, "<check constraint definition>"	ISO/IEC 9075-2
Subclause 10.2, “<view definition>”	Subclause 11.21, "<view definition>"	ISO/IEC 9075-2
Subclause 10.3, “<assertion definition>”	Subclause 11.36, "<assertion definition>"	ISO/IEC 9075-2
Subclause 10.4, “<trigger definition>”	Subclause 11.38, "<trigger definition>"	ISO/IEC 9075-2
Subclause 10.5, “<SQL-invoked routine>”	Subclause 11.49, "<SQL-invoked routine>"	ISO/IEC 9075-2
Clause 11, “SQL-client modules”	Clause 13, "SQL-client modules"	ISO/IEC 9075-2
Subclause 11.1, “<SQL-client module definition>”	Subclause 13.1, "<SQL-client module definition>"	ISO/IEC 9075-2
Subclause 11.2, “Calls to an <externally-invoked procedure>”	Subclause 13.4, "Calls to an <externally-invoked procedure>"	ISO/IEC 9075-2
Subclause 11.3, “<SQL procedure statement>”	Subclause 13.5, "<SQL procedure statement>"	ISO/IEC 9075-2
Clause 12, “Data manipulation”	Clause 14, "Data manipulation"	ISO/IEC 9075-2
Subclause 12.1, “<select statement: single row>”	Subclause 14.5, "<select statement: single row>"	ISO/IEC 9075-2
Subclause 12.2, “<free locator statement>”	Subclause 14.12, "<free locator statement>"	ISO/IEC 9075-2
Clause 13, “Transaction management”	Clause 16, "Transaction management"	ISO/IEC 9075-2
Subclause 13.1, “<commit statement>”	Subclause 16.6, "<commit statement>"	ISO/IEC 9075-2
Clause 14, “Session management”	Clause 18, "Session management"	ISO/IEC 9075-2
Subclause 14.1, “<set catalog statement>”	*none*	*none*
Subclause 14.2, “<set schema statement>”	*none*	*none*
Subclause 14.3, “<set names statement>”	*none*	*none*
Subclause 14.4, “<set path statement>”	*none*	*none*
Subclause 14.5, “<set transform group statement>”	*none*	*none*

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Clause 15, "Dynamic SQL"	*none*	*none*
Subclause 15.1, "Description of SQL descriptor areas"	*none*	*none*
Subclause 15.2, "<allocate descriptor statement>"	*none*	*none*
Subclause 15.3, "<deallocate descriptor statement>"	*none*	*none*
Subclause 15.4, "<get descriptor statement>"	*none*	*none*
Subclause 15.5, "<set descriptor statement>"	*none*	*none*
Subclause 15.6, "<prepare statement>"	*none*	*none*
Subclause 15.7, "<deallocate prepared statement>"	*none*	*none*
Subclause 15.8, "<describe statement>"	*none*	*none*
Subclause 15.9, "<input using clause>"	*none*	*none*
Subclause 15.10, "<output using clause>"	*none*	*none*
Subclause 15.11, "<execute statement>"	*none*	*none*
Subclause 15.12, "<execute immediate statement>"	*none*	*none*
Subclause 15.13, "<dynamic declare cursor>"	*none*	*none*
Subclause 15.14, "<allocate cursor statement>"	*none*	*none*
Subclause 15.15, "<dynamic open statement>"	*none*	*none*
Subclause 15.16, "<dynamic fetch statement>"	*none*	*none*
Subclause 15.17, "<dynamic single row select statement>"	*none*	*none*
Subclause 15.18, "<dynamic close statement>"	*none*	*none*
Subclause 15.19, "<dynamic delete statement: positioned>"	*none*	*none*

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 15.20, "<dynamic update statement: positioned>"	*none*	*none*
Subclause 15.21, "<preparable dynamic delete statement: positioned>"	*none*	*none*
Subclause 15.22, "<preparable dynamic update statement: positioned>"	*none*	*none*
Clause 16, "Embedded SQL"	*none*	*none*
Subclause 16.1, "<embedded SQL host program>"	*none*	*none*
Subclause 16.2, "<embedded exception declaration>"	*none*	*none*
Subclause 16.3, "<embedded SQL Ada program>"	*none*	*none*
Subclause 16.4, "<embedded SQL C program>"	*none*	*none*
Subclause 16.5, "<embedded SQL COBOL program>"	*none*	*none*
Subclause 16.6, "<embedded SQL Fortran program>"	*none*	*none*
Subclause 16.7, "<embedded SQL MUMPS program>"	*none*	*none*
Subclause 16.8, "<embedded SQL Pascal program>"	*none*	*none*
Subclause 16.9, "<embedded SQL PL/I program>"	*none*	*none*
Clause 17, "Direct invocation of SQL"	*none*	*none*
Subclause 17.1, "<direct SQL statement>"	*none*	*none*
Subclause 17.2, "<direct select statement: multiple rows>"	*none*	*none*
Clause 18, "Diagnostics management"	Clause 19, "Diagnostics management"	ISO/IEC 9075-2
Subclause 18.1, "<get diagnostics statement>"	Subclause 19.1, "<get diagnostics statement>"	ISO/IEC 9075-2
Clause 19, "Definition Schema"	Clause 21, "Definition Schema"	ISO/IEC 9075-2
Subclause 19.1, "SQL_LANGUAGES base table"	Subclause 21.37, "SQL_LANGUAGES base table"	ISO/IEC 9075-2

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Clause 20, “Status codes”	Clause 22, "Status codes"	ISO/IEC 9075-2
Subclause 20.1, “SQLSTATE”	Subclause 22.1, "SQLSTATE"	ISO/IEC 9075-2
Clause 21, “Conformance”	Clause 23, "Conformance"	ISO/IEC 9075-2
Subclause 21.2, “Claims of conformance”	Subclause 8.1.5, "Claims of conformance"	ISO/IEC 9075-1
Annex A, “SQL Conformance Summary”	Appendix A, "SQL Conformance Summary"	ISO/IEC 9075-2
Annex B, “Implementation-defined elements”	Appendix B, "Implementation-defined elements"	ISO/IEC 9075-2
Annex C, “Implementation-dependent elements”	Appendix C, "Implementation-dependent elements"	ISO/IEC 9075-2
Annex D, “Deprecated features”	Appendix D, "Deprecated features"	ISO/IEC 9075-2
Annex E, “Incompatibilities with ISO/IEC 9075:1992”	Appendix E, "Incompatibilities with ISO/IEC 9075:1992 and ISO/IEC 9075-4:1996"	ISO/IEC 9075-2
Annex F, “SQL feature and package taxonomy”	Appendix F, "SQL feature and package taxonomy"	ISO/IEC 9075-2
Table 1, “Clause, Subclause, and Table relationships”	*none*	*none*
Table 2, “Data types of <key word>s used in the header of SQL descriptor areas”	*none*	*none*
Table 3, “Data types of <key word>s used in SQL item descriptor areas”	*none*	*none*
Table 4, “Codes used for SQL data types in Dynamic SQL”	*none*	*none*
Table 5, “Codes associated with datetime data types in Dynamic SQL”	*none*	*none*
Table 6, “Codes used for <interval qualifier>s in Dynamic SQL”	*none*	*none*
Table 7, “Codes used for input/output SQL parameter modes in Dynamic SQL”	*none*	*none*
Table 8, “<identifier>s for use with <get diagnostics statement>”	Table 25, "<identifier>s for use with <get diagnostics statement>"	ISO/IEC 9075-2
Table 9, “SQL-statement codes”	Table 26, "SQL-statement codes"	ISO/IEC 9075-2
Table 10, “SQLSTATE class and subclass values”	Table 27, "SQLSTATE class and subclass values"	ISO/IEC 9075-2

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Table 12, “SQL/Bindings feature taxonomy and definition for Core SQL”	Table 31, "SQL/Foundation feature taxonomy and definition for Core SQL"	ISO/IEC 9075-2
Table 13, “SQL/Bindings feature taxonomy for features outside Core SQL”	Table 32, "SQL/Foundation feature taxonomy for features outside Core SQL"	ISO/IEC 9075-2

3.4 Object identifier for Database Language SQL

The object identifier for Database Language SQL is defined in ISO/IEC 9075-1 in Subclause 6.3, "Object identifier for Database Language SQL", with the following additions:

Format

```
<Part 5 yes> ::=
       <Part 5 conformance>
       <Part 5 direct> <Part 5 embedded>

<Part 5 conformance> ::=
       5 | sqlbindings199x <left paren> 5 <right paren>

<Part 5 direct> ::=
         <Part 5 direct yes>
       | <Part 5 direct no>

<Part 5 direct yes> ::=
       1 | directyes <left paren> 1 <right paren>

<Part 5 direct no> ::=
       0 | directno <left paren> 0 <right paren>

<Part 5 embedded> ::=
         <Part 5 embedded no>
       | <Part 5 embedded languages>...

<Part 5 embedded no> ::=
       0 | embeddedno <left paren> 0 <right paren>

<Part 5 embedded languages> ::=
         <Part 5 embedded Ada>
       | <Part 5 embedded C>
       | <Part 5 embedded COBOL>
       | <Part 5 embedded Fortran>
       | <Part 5 embedded MUMPS>
       | <Part 5 embedded Pascal>
       | <Part 5 embedded PL/I>

<Part 5 embedded Ada> ::=
       1 | embeddedAda <left paren> 1 <right paren>

<Part 5 embedded C> ::=
```

--`,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
       2 | embeddedC <left paren> 2 <right paren>

<Part 5 embedded COBOL> ::=
       3 | embeddedCOBOL <left paren> 3 <right paren>

<Part 5 embedded Fortran> ::=
       4 | embeddedFortran <left paren> 4 <right paren>

<Part 5 embedded MUMPS> ::=
       5 | embeddedMUMPS <left paren> 5 <right paren>

<Part 5 embedded Pascal> ::=
       6 | embeddedPascal <left paren> 6 <right paren>

<Part 5 embedded PL/I> ::=
       7 | embeddedPLI <left paren> 7 <right paren>
```

Syntax Rules

1) Specification of <Part 5 Yes> implies that conformance to this part of ISO/IEC 9075 is claimed.

2) Specification of <Part 5 direct yes> implies that conformance to <direct SQL statement> is claimed.

3) Specification of <Part 5 direct no> implies that conformance to <direct SQL statement> is not claimed.

4) Specification of <Part 5 embedded *language*> implies that conformance to <embedded SQL host program> for the specified *language* is claimed.

5) Specification of <Part 5 embedded no> implies that conformance to <embedded SQL host program> is not claimed.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4 Concepts

4.1 Catalogs

Insert this paragraph The default catalog name for <preparable statement>s that are dynamically prepared in the current SQL-session through the execution of <prepare statement>s and <execute immediate statement>s is initially implementation-defined but may be changed by the use of <set catalog statement>s.

4.2 SQL-client modules

Insert this paragraph SQL-session modules are implicitly-created modules for prepared SQL-statements (see Subclause 4.34, "SQL-sessions", in ISO/IEC 9075-2).

Insert this paragraph Each <SQL-client module definition> *M* that is associated with an SQL-session, other than an SQL-session module, may have associated with it a *shadow module M1* that has an implementation-dependent name not equivalent to the <module name> of any other <SQL-client module definition> in the same SQL-session and whose <module authorization clause> specifies "SCHEMA *SN*", where *SN* is the explicit or implicit <schema name> of the <module authorization clause> of *M*. The <language clause>, the SQL-path, if specified, and the <module character set specification> of *M1* are identical to the corresponding characteristic of *M*. When *M* contains a <module authorization clause> that specifies "FOR STATIC ONLY", this shadow module effectively contains one <externally-invoked procedure> for each SQL-statement prepared by <prepared statement>s or <execute immediate statement>s contained in *M*.

4.3 SQL-invoked routines

Augment the 6th paragraph The <SQL procedure statement> forming the <routine body> must exclude <SQL dynamic statement>.

4.4 Locators

Augment the 1st paragraph A host variable may be specified to be a *locator* by specifying AS LOCATOR.

Augment the 1st paragraph A host variable specified as a locator may be further specified to be a *holdable locator*.

4.5 Cursors

Insert this paragraph A cursor is also specified by a <dynamic declare cursor> or an <allocate cursor statement>. A cursor specified by a <dynamic declare cursor> is a *declared dynamic cursor*. A cursor specified by an <allocate cursor statement> is an *extended dynamic cursor*. A *dynamic cursor* is either a declared dynamic cursor or an extended dynamic cursor.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Insert this paragraph For every <dynamic declare cursor> in an <SQL-client module definition>, a cursor is effectively created when an SQL-transaction (see Subclause 4.32, "SQL-transactions", in ISO/IEC 9075-2) referencing the <SQL-client module definition> is initiated. An extended dynamic cursor is also effectively created when an <allocate cursor statement> is executed within an SQL-session and destroyed when that SQL-session is terminated.

Insert this paragraph In addition, a dynamic cursor is destroyed when a <deallocate prepared statement> is executed that deallocates the prepared statement on which the dynamic cursor is based.

Insert this paragraph A cursor is also placed in the open state by a <dynamic open statement> and returned to the closed state by a <dynamic close statement>.

Insert this paragraph A <dynamic fetch statement> positions an open cursor on a specified row of the cursor's ordering and retrieves the values of the columns of that row. A <dynamic update statement: positioned> updates the current row of the cursor. A <dynamic delete statement: positioned> deletes the current row of the cursor.

4.6 SQL-statements

4.6.1 Classes of SQL-statements

Insert this paragraph Most SQL-statements can be prepared for execution and executed in additional ways. These are:

— In an embedded SQL host program, in which case they are prepared when the embedded SQL host program is preprocessed (see Subclause 4.8, "Embedded syntax").

— Being prepared and executed by the use of SQL-dynamic statements (which are themselves executed in an SQL-client module or an embedded SQL host program—see Subclause 4.9, "SQL dynamic statements").

— Direct invocation, in which case they are effectively prepared immediately prior to execution (see Subclause 4.10, "Direct invocation of SQL").

Insert this paragraph There are at least three additional ways of classifying SQL-statements:

— According to whether or not they may be embedded.

— According to whether they may be dynamically prepared and executed.

— According to whether or not they may be directly executed.

4.6.2 SQL-statements classified by function

Insert this paragraph The following are other main classes of SQL-statements:

— SQL-dynamic statements

— SQL embedded exception declaration

Insert this paragraph The following are additional SQL-data statements:

— <dynamic declare cursor>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

— <allocate cursor statement>

— <dynamic select statement>

— <dynamic open statement>

— <dynamic close statement>

— <dynamic fetch statement>

— <direct select statement: multiple rows>

— <dynamic single row select statement>

Insert this paragraph The following are additional SQL-data change statements:

— <dynamic delete statement: positioned>

— <preparable dynamic delete statement: positioned>

— <dynamic update statement: positioned>

— <preparable dynamic update statement: positioned>

Insert this paragraph The following are additional SQL-session statements:

— <set catalog statement>

— <set schema statement>

— <set names statement>

— <set path statement>

— <set transform group statement>

Insert this paragraph The following are the SQL-dynamic statements:

— <execute immediate statement>

— <allocate descriptor statement>

— <deallocate descriptor statement>

— <get descriptor statement>

— <set descriptor statement>

— <prepare statement>

— <deallocate prepared statement>

— <describe input statement>

— <describe output statement>

— <execute statement>

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Insert this paragraph The following is the SQL embedded exception declaration:

— <embedded exception declaration>

4.6.3 SQL-statements and transaction states

Insert this paragraph Whether or not an <execute immediate statement> starts a transaction depends on the content of the <SQL statement variable> referenced by the <execute immediate statement> at the time it is executed. Whether or not an <execute statement> starts a transaction depends on the content of the <SQL statement variable> referenced by the <prepare statement> at the time the prepared statement referenced by the <execute statement> was prepared. In both cases, if the content of the <SQL statement variable> was a transaction-initiating SQL-statement, then the <execute immediate statement> or <execute statement> is treated as a transaction-initiating statement; otherwise it is not treated as a transaction-initiating statement.

Insert this paragraph The following additional SQL-statements are transaction-initiating SQL-statements, *i.e.*, if there is no current transaction, and a statement of this class is executed, a transaction is initiated:

— <allocate cursor statement>

— <dynamic open statement>

— <dynamic close statement>

— <dynamic fetch statement>

— <direct select statement: multiple rows>

— <dynamic single row select statement>

— <dynamic delete statement: positioned>

— <preparable dynamic delete statement: positioned>

— <dynamic update statement: positioned>

— <preparable dynamic update statement: positioned>

— The following SQL-dynamic statements:

- <describe input statement>
- <describe output statement>
- <allocate descriptor statement>
- <deallocate descriptor statement>
- <get descriptor statement>
- <set descriptor statement>
- <prepare statement>
- <deallocate prepared statement>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Insert this paragraph The following additional SQL-statements are not transaction-initiating SQL-statements, *i.e.*, if there is no current transaction, and a statement of this class is executed, no transaction is initiated.

— SQL embedded exception declarations

— The following SQL-data statements:

- <dynamic declare cursor>
- <dynamic select statement>

4.6.4 Embeddable SQL-statements

The following SQL-statements are embeddable in an embedded SQL host program, and may be the <SQL procedure statement> in an <externally-invoked procedure> in an <SQL-client module definition>:

— All SQL-schema statements

— All SQL-transaction statements

— All SQL-connection statements

— All SQL-session statements

— All SQL-dynamic statements

— All SQL-diagnostics statements

— The following SQL-data statements:

- <allocate cursor statement>
- <open statement>
- <dynamic open statement>
- <close statement>
- <dynamic close statement>
- <fetch statement>
- <dynamic fetch statement>
- <select statement: single row>
- <insert statement>
- <delete statement: searched>
- <delete statement: positioned>
- <dynamic delete statement: positioned>
- <update statement: searched>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

- <update statement: positioned>
- <dynamic update statement: positioned>
- <free locator statement>

— The following SQL-control statements:

- <call statement>

The following SQL-statements are embeddable in an embedded SQL host program, and may occur in an <SQL-client module definition>, though not in an <externally-invoked procedure>:

— <temporary table declaration>

— <declare cursor>

— <dynamic declare cursor>

The following SQL-statements are embeddable in an embedded SQL host program, but may not occur in an <SQL-client module definition>:

— SQL embedded exception declarations

Consequently, the following SQL-data statements are not embeddable in an embedded SQL host program, nor may they occur in an <SQL-client module definition>, nor be the <SQL procedure statement> in an <externally-invoked procedure> in an <SQL-client module definition>:

— <dynamic select statement>

— <dynamic single row select statement>

— <direct select statement: multiple rows>

— <preparable dynamic delete statement: positioned>

— <preparable dynamic update statement: positioned>

4.6.5 Preparable and immediately executable SQL-statements

The following SQL-statements are preparable:

— All SQL-schema statements

— All SQL-transaction statements

— All SQL-session statements

— The following SQL-data statements:

- <delete statement: searched>
- <dynamic select statement>
- <dynamic single row select statement>
- <insert statement>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

- <update statement: searched>
- <preparable dynamic delete statement: positioned>
- <preparable dynamic update statement: positioned>
- <preparable implementation-defined statement>
- <free locator statement>

— The following SQL-control statements:

- <call statement>

Consequently, the following SQL-statements are not preparable:

— All SQL-connection statements

— All SQL-dynamic statements

— All SQL-diagnostics statements

— SQL embedded exception declarations

— The following SQL-data statements:

- <allocate cursor statement>
- <open statement>
- <dynamic open statement>
- <close statement>
- <dynamic close statement>
- <fetch statement>
- <dynamic fetch statement>
- <select statement: single row>
- <delete statement: positioned>
- <dynamic delete statement: positioned>
- <update statement: positioned>
- <dynamic update statement: positioned>
- <direct select statement: multiple rows>
- <temporary table declaration>
- <declare cursor>
- <dynamic declare cursor>

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Any preparable SQL-statement can be executed immediately, with the exception of:

— <dynamic select statement>

— <dynamic single row select statement>

4.6.6 Directly executable SQL-statements

The following SQL-statements may be executed directly:

— All SQL-schema statements

— All SQL-transaction statements

— All SQL-connection statements

— All SQL-session statements

— The following SQL-data statements:

- <temporary table declaration>
- <direct select statement: multiple rows>
- <insert statement>
- <delete statement: searched>
- <update statement: searched>

Consequently, the following SQL-statements may not be executed directly:

— All SQL-dynamic statements

— All SQL-diagnostics statements

— SQL embedded exception declarations

— The following SQL-data statements:

- <declare cursor>
- <dynamic declare cursor>
- <allocate cursor statement>
- <open statement>
- <dynamic open statement>
- <close statement>
- <dynamic close statement>
- <fetch statement>
- <dynamic fetch statement>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

- <select statement: single row>
- <dynamic select statement>
- <dynamic single row select statement>
- <delete statement: positioned>
- <dynamic delete statement: positioned>
- <preparable dynamic delete statement: positioned>
- <update statement: positioned>
- <dynamic update statement: positioned>
- <preparable dynamic update statement: positioned>

— <free locator statement>

— <hold locator statement>

4.7 Standard programming languages

Insert this paragraph This part of ISO/IEC 9075 specifies a mechanism whereby SQL language may be embedded in programs that otherwise conform to any of the same specified programming language standards.

NOTE 1 – In this part of ISO/IEC 9075, for the purposes of interfacing with programming languages, the data types DATE, TIME, TIMESTAMP, and INTERVAL must be converted to or from character strings in those programming languages by means of a <cast specification>. It is anticipated that future evolution of programming language standards will support data types corresponding to these four SQL data types; this standard will then be amended to reflect the availability of those corresponding data types.

The data types CHARACTER, CHARACTER VARYING, and CHARACTER LARGE OBJECT are also mapped to character strings in the programming languages. However, because the facilities available in the programming languages do not provide the same capabilities as those available in SQL, there must be agreement between the host program and SQL regarding the specific format of the character data being exchanged. Specific syntax for this agreement is provided in this part of ISO/IEC 9075. For standard programming languages C, COBOL, Fortran, and Pascal, bit strings are mapped to character variables in the host language in a manner described in Subclause 16.1, "<embedded SQL host program>". For standard programming languages Ada and PL/I, bit string variables are directly supported.

For standard programming languages C and COBOL, BOOLEAN values are mapped to integer variables in the host language. For standard programming languages Ada, Fortran, Pascal, and PL/I, BOOLEAN variables are directly supported.

For the purposes of interfacing with programming languages, the data type ARRAY must be converted to a locator (see Subclause 4.26.4, "Locators", in ISO/IEC 9075-2).

For the purposes of interfacing with programming languages, user-defined types must be handled with a locator (see Subclause 4.26.4, "Locators", in ISO/IEC 9075-2) or transformed to another SQL data type that has a defined mapping to the host language (see Subclause 4.8.5, "Transforms for user-defined types", in ISO/IEC 9075-2).

--`,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4.8 Embedded syntax

An <embedded SQL host program> (<embedded SQL Ada program>, <embedded SQL C program>, <embedded SQL COBOL program>, <embedded SQL Fortran program>, <embedded SQL MUMPS program>, <embedded SQL Pascal program>, or <embedded SQL PL/I program>) is a compilation unit that consists of programming language text and SQL text. The programming language text shall conform to the requirements of a specific standard programming language. The SQL text shall consist of one or more <embedded SQL statement>s and, optionally, one or more <embedded SQL declare section>s, as defined in this International Standard. This allows database applications to be expressed in a hybrid form in which SQL-statements are embedded directly in a compilation unit. Such a hybrid compilation unit is defined to be equivalent to a standard compilation unit in which the SQL-statements have been replaced by standard procedure or subroutine calls of <externally-invoked procedure>s in a separate SQL-client module, and in which each has been removed and the declarations contained therein have been suitably transformed into standard host-language syntax.

If an <embedded SQL host program> contains an <embedded authorization declaration>, then it shall be the first statement or declaration in the <embedded SQL host program>. The <embedded authorization declaration> is not replaced by a procedure or subroutine call of an <externally-invoked procedure>, but is removed and replaced by syntax associated with the <SQL-client module definition>'s <module authorization clause>.

An implementation may reserve a portion of the name space in the <embedded SQL host program> for the names of procedures or subroutines that are generated to replace SQL-statements and for program variables and branch labels that may be generated as required to support the calling of these procedures or subroutines; whether this reservation is made is implementation-defined. They may similarly reserve name space for the <module name> and <procedure name>s of the generated <SQL-client module definition> that may be associated with the resulting standard compilation unit. The portion of the name space to be so reserved, if any, is implementation-defined.

4.9 SQL dynamic statements

In many cases, the SQL-statement to be executed can be coded into an <SQL-client module definition> or into a compilation unit using the embedded syntax. In other cases, the SQL-statement is not known when the program is written and will be generated during program execution. An <execute immediate statement> can be used for a one-time preparation and execution of an SQL-statement. A <prepare statement> is used to prepare the generated SQL-statement for subsequent execution. A <deallocate prepared statement> is used to deallocate SQL-statements that have been prepared with a <prepare statement>. A description of the input dynamic parameters for a prepared statement can be obtained by execution of a <describe input statement>. A description of the resultant columns of a <dynamic select statement> or <dynamic single row select statement> can be obtained by execution of a <describe output statement>. A description of the output dynamic parameters of a statement that is neither a <dynamic select statement> nor a <dynamic single row select statement> can be obtained by execution of a <describe output statement>. For a statement other than a <dynamic select statement>, an <execute statement> is used to associate parameters with the prepared statement and execute it as though it had been coded when the program was written. For a <dynamic select statement>, the prepared <cursor specification> is associated with a cursor via a <dynamic declare cursor> or <allocate cursor statement>. The cursor can be opened and dynamic parameters can be associated with the cursor with a <dynamic open statement>. A <dynamic fetch statement> positions an open cursor on a specified row and retrieves the values of the columns of that row. A <dynamic close statement> closes a cursor that was opened with a <dynamic open statement>. A <dynamic delete statement: positioned> is used to delete rows through

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

a dynamic cursor. A <dynamic update statement: positioned> is used to update rows through a dynamic cursor. A <preparable dynamic delete statement: positioned> is used to delete rows through a dynamic cursor when the precise format of the statement isn't known until runtime. A <preparable dynamic update statement: positioned> is used to update rows through a dynamic cursor when the precise format of the statement isn't known until runtime.

The interface for input dynamic parameters and output dynamic parameters for a prepared statement and for the resulting values from a <dynamic fetch statement> or the execution of a prepared <dynamic single row select statement> can be either a list of dynamic parameters or embedded variables or an SQL descriptor area. An SQL descriptor area consists of one or more item descriptor areas, together with a header that includes a count of the number of those item descriptor areas. The header of an SQL descriptor area consists of the fields in Table 2, "Data types of <key word>s used in the header of SQL descriptor areas", in Subclause 15.1, "Description of SQL descriptor areas". Each item descriptor area consists of the fields specified in Table 3, "Data types of <key word>s used in SQL item descriptor areas", in Subclause 15.1, "Description of SQL descriptor areas". The SQL descriptor area is allocated and maintained by the system with the following statements: <allocate descriptor statement>, <deallocate descriptor statement>, <set descriptor statement>, and <get descriptor statement>.

An SQL descriptor area is identified by a <descriptor name>, which is a <simple value specification> whose value is an <identifier>. Two <descriptor name>s identify the same SQL descriptor area if their values, with leading and trailing <space>s removed, are equivalent according to the rules for <identifier> comparisons in Subclause 5.2, "<token> and <separator>", in ISO/IEC 9075-2.

Dynamic statements can be identified by <statement name>s or by <extended statement name>s. Similarly, dynamic cursors can be identified by <cursor name>s and by <extended cursor name>s. The non-extended names are <identifier>s. The extended names are <target specification>s whose values are <identifier>s used to identify the statement or cursor. Two extended names are equivalent if their values, with leading and trailing <space>s removed, are equivalent according to the rules for <identifier> comparison in Subclause 5.2, "<token> and <separator>", in ISO/IEC 9075-2.

An SQL descriptor area name may be defined as global or local. Similarly, an extended statement name or extended cursor name may be global or local. The scope of a global name is the SQL-session. The scope of a local name is the <SQL-client module definition> in which it appears. A reference to an entity in which one specifies a global scope is valid only if the entity was defined as global and if the reference is from the same SQL-session in which it was defined. A reference to an entity in which one specifies a local scope is valid only if the entity was defined as local and if the reference is from the same <SQL-client module definition> in which it was defined. (The scope of non-extended statement names and non-extended cursor names is always local.)

NOTE 2 – The namespace of non-extended statement names and non-extended cursor names is different from the namespace of extended statement names that are specified to be LOCAL and extended cursor names that are specified to be LOCAL, respectively.

The relationships between prepared statements and <SQL-client module definition>s are:

— Within an SQL-session, all global prepared statements (prepared statements with global statement names) belong to the SQL-session module.

— Within an SQL-session, each local prepared statement (a prepared statement with a local statement name) prepared in an <SQL-client module definition> containing a <module authorization clause> that does not specify "AUTHORIZATION <module authorization identifier>" belongs to the <SQL-client module definition> that contains the <prepare statement> or <execute immediate statement> with which it is prepared.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

— Within an SQL-session, each local prepared statement (a prepared statement with a local statement names) prepared in an <SQL-client module definition> containing a <module authorization clause> that contains "AUTHORIZATION <module authorization identifier>" and does not contain "FOR STATIC ONLY" belongs to the <SQL-client module definition> that contains the <prepare statement> or <execute immediate statement> with which it is prepared.

— Within an SQL-session, each local prepared statement (a prepared statement with a local statement names) prepared in an <SQL-client module definition> *M* containing a <module authorization clause> that contains "AUTHORIZATION <module authorization identifier> FOR STATIC ONLY" belongs to a *shadow module* that has an implementation-dependent name not equivalent to the <module name> of any other <SQL-client module definition> in the same SQL-session and whose <module authorization clause> specifies "SCHEMA *SN*", where *SN* is the explicit or implicit <schema name> of the <module authorization clause> of *M*.

NOTE 3 – The SQL-session module is defined in Subclause 4.34, "SQL-sessions", in ISO/IEC 9075-2.

Dynamic execution of SQL-statements can generally be accomplished in two different ways. Statements can be *prepared* for execution and then later executed one or more times; when the statement is no longer needed for execution, it can be *released* by the use of a <deallocate prepared statement>. Alternatively, a statement that is needed only once can be executed without the preparation step—it can be *executed immediately* (not all SQL-statements can be executed immediately).

Many SQL-statements can be written to use "parameters" (which are manifested in static execution of SQL-statements as host parameters in <SQL procedure statement>s contained in <externally-invoked procedure>s in <SQL-client module definition>s or as host variables in <embedded SQL statement>s contained in <embedded SQL host program>s). In SQL-statements that are executed dynamically, the parameters are called dynamic parameters (<dynamic parameter specification>s) and are represented in SQL language by a <question mark> (?).

In many situations, an application that generates an SQL-statement for dynamic execution knows in detail the required characteristics (*e.g.*, <data type>, <length>, <precision>, <scale>, *etc.*) of each of the dynamic parameters used in the statement; similarly, the application may also know in detail the characteristics of the values that will be returned by execution of the statement. However, in other cases, the application may not know this information to the required level of detail; it is possible in some cases for the application to ascertain the information from the Information Schema, but in other cases (*e.g.*, when a returned value is derived from a computation instead of simply from a column in a table, or when dynamic parameters are supplied) this information is not generally available except in the context of preparing the statement for execution.

To provide the necessary information to applications, SQL permits an application to request the SQL-server to *describe* a prepared statement. The description of a statement identifies the number of input dynamic parameters (*describe input*) and their data type information or it identifies the number of output dynamic parameters or values to be returned (*describe output*) and their data type information. The description of a statement is placed into the SQL descriptor areas already mentioned.

Many, but not all, SQL-statements can be prepared and executed dynamically.

NOTE 4 – The complete list of statements that may be dynamically prepared and executed is defined in Subclause 4.6.5, "Preparable and immediately executable SQL-statements".

Certain "set statements" (<set catalog statement>, <set schema statement>, <set names statement>, and <set path statement>) have no effect other than to set up default information (catalog name, schema name, character set, and SQL path, respectively) to be applied to other SQL-statements that are prepared or executed immediately or that are invoked directly.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Syntax errors and Access Rule violations caused by the preparation or immediate execution of <preparable statement>s are identified when the statement is prepared (by <prepare statement>) or when it is executed (by <execute statement> or <execute immediate statement>); executed (by <execute statement> or <execute immediate statement>); such violations are indicated by the raising of an exception condition.

4.10 Direct invocation of SQL

Direct invocation of SQL is a mechanism for executing direct SQL-statements, known as <direct SQL statement>s. In direct invocation of SQL, the method of invoking <direct SQL statement>s, the method of raising conditions that result from the execution of <direct SQL statement>s, the method of accessing the diagnostics information that results from the execution of <direct SQL statement>s, and the method of returning the results are implementation-defined.

4.11 Privileges and roles

Insert this paragraph For direct SQL, the SQL-session user identifier is always the current authorization identifier.

4.12 SQL-transactions

Insert this paragraph The operations comprising an SQL-transaction may also be performed by the direct invocation of SQL.

Insert this paragraph It is implementation-defined whether or not the dynamic execution of an <SQL dynamic data statement> is permitted to occur within the same SQL-transaction as the dynamic execution of an SQL-schema statement. If it does occur, then the effect on any open cursor, prepared dynamic statement, or deferred constraint is implementation-defined. There may be additional implementation-defined restrictions, requirements, and conditions. If any such restrictions, requirements, or conditions are violated, then an implementation-defined exception condition or a completion condition *warning* with an implementation-defined subclass code is raised.

Insert this paragraph Each direct invocation of SQL that executes an SQL-statement of an SQL-transaction is associated with that SQL-transaction. An SQL-transaction is initiated when no SQL-transaction is currently active by direct invocation of SQL that results in the execution of a transaction-initiating <direct SQL statement>.

4.13 SQL-connections

Insert this paragraph An SQL-connection may also be terminated by the last execution of a <direct SQL statement> through the direct invocation of SQL. The mechanism and rules by which an SQL-implementation determines whether a direct invocation of SQL is the last execution of a <direct SQL statement> are implementation-defined.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4.14 SQL-sessions

Replace 3rd paragraph Within an SQL-session, module local temporary tables are effectively created by <temporary table declaration>s. Module local temporary tables are accessible only to invocations of <externally-invoked procedure>s in the SQL-client module or SQL-session module in which they are created. The definitions of module local temporary tables persist until the end of the SQL-session.

Insert this paragraph An *SQL-session* also spans the execution of a sequence of consecutive SQL-statements invoked by the direct invocation of SQL.

Insert this paragraph An SQL-session has an SQL-session module that is different from any other <SQL-client module definition> that exists simultaneously in the SQL-environment. The SQL-session module contains the global prepared SQL-statements that belong to the SQL-session. The SQL-session module contains a <module authorization clause> that specifies SCHEMA <schema name>, where the value of <schema name> is implementation-dependent.

Insert this paragraph An SQL-session has a default catalog name that is used to effectively qualify unqualified <schema name>s that are contained in <preparable statement>s when those statements are prepared in the current SQL-session by either an <execute immediate statement> or a <prepare statement> or are contained in <direct SQL statement>s when those statements are invoked directly. The default catalog name is initially set to an implementation-defined value but can subsequently be changed by the successful execution of a <set catalog statement> or <set schema statement>.

Insert this paragraph An SQL-session has a default unqualified schema name that is used to effectively qualify unqualified <schema qualified name>s that are contained in <preparable statement>s when those statements are prepared in the current SQL-session by either an <execute immediate statement> or a <prepare statement> or are contained in <direct SQL statement>s when those statements are invoked directly. The default unqualified schema name is initially set to an implementation-defined value but can subsequently be changed by the successful execution of a <set schema statement>.

Insert this paragraph An SQL-session has a SQL-path that is used to effectively qualify unqualified <routine name>s that are immediately contained in <routine invocation>s that are contained in <preparable statement>s when those statements are prepared in the current SQL-session by either an <execute immediate statement> or a <prepare statement> or are contained in <direct SQL statement>s when those statements are invoked directly. The SQL-path is initially set to an implementation-defined value, but can subsequently be changed by the successful execution of a <set path statement>.

Insert this paragraph The text defining the SQL-path can be referenced by using the <general value specification> CURRENT_PATH.

Insert this paragraph An SQL-session has a default transform group name and one or more user-defined type name-transform group name pairs that are used to identify the group of transform functions for every user-defined type that is referenced in <preparable statement>s when those statements are prepared in the current SQL-session by either an <execute immediate statement> or a <prepare statement> or are contained in <direct SQL statement>s when those statements are invoked directly. The transform group name for a given user-defined type name is initially set to an implementation-defined value but can subsequently be changed by the successful execution of a <set transform group statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Insert this paragraph The text defining the transform group names associated with the SQL-session can be referenced using two mechanisms: the <general value specification> "CURRENT_TRANSFORM_GROUP_FOR_TYPE <user-defined type>", which evaluates to the name of the transform group associated with the specified data type, and the <general value specification> "CURRENT_DEFAULT_TRANSFORM_GROUP", which evaluates to the name of the transform group associated with all types that have no type-specific transform group specified for them.

Insert this paragraph An SQL-session has a default character set name that is used to identify the character set in which <preparable statement>s are represented when those statements are prepared in the current SQL-session by either an <execute immediate statement> or a <prepare statement>. The default character set name is initially set to an implementation-defined value but can subsequently be changed by the successful execution of a <set names statement>.

Augment the 4th paragraph Within an SQL-session, locators are effectively created whenever a host variable that is specified as a binary large object locator, character large object locator, array locator, or user-defined type locator is assigned a value of binary large object type, character large object type, array type, or user-defined type, respectively. A host variable that is a locator may be *holdable* or *nonholdable*.

Insert this paragraph The SQL-session context also comprises:

— The SQL-session module.

— The current default catalog name.

— The current default unqualified schema name.

— The current character set name substitution value.

— The text defining the SQL-path.

— The contents of all SQL dynamic descriptor areas.

— The text defining the default transform group name.

— The text defining the user-defined type name-transform group name pair for each user-defined type explicitly set by the user.

4.15 Client-server operation

Insert this paragraph <direct SQL statement>s containing an <SQL connection statement> or an <SQL diagnostics statement> are processed by the SQL-client. Other <direct SQL statement>s are processed by the SQL-server.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

5 Lexical elements

5.1 <token> and <separator>

Function

Specify lexical units (tokens and separators) that participate in SQL language.

Format

```
<delimiter token> ::=
       !! All alternatives from ISO/IEC 9075-2
       | <double period>

<double period> ::= ..

<non-reserved word> ::=
         !! All alternatives from ISO/IEC 9075-2

       | DEGREE | DYNAMIC_FUNCTION | DYNAMIC_FUNCTION_CODE

       | RETURNED_CARDINALITY

       | STRUCTURE

       | TOP_LEVEL_COUNT

<reserved word> ::=
         !! All alternatives from ISO/IEC 9075-2

       | CURRENT_DEFAULT_TRANSFORM_GROUP
       | CURRENT_TRANSFORM_GROUP_FOR_TYPE

       | DYNAMIC

       | NESTING
```

Syntax Rules

None.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Conformance Rules

No additional Conformance Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5.2 <literal>

Function

Specify a non-null value.

Format

No additional Format items.

Syntax Rules

1) Augments SR8) Case:

   a) If a <character set specification> is not specified in a <character string literal>, then the set of characters contained in the <character string literal> shall be wholly contained in the character set of the SQL-client module or SQL-session module that contains the <character string literal>.

   b) Otherwise, there shall be no <separator> between the <introducer> and the <character set specification>, and the set of characters contained in the <character string literal> shall be wholly contained in the character set specified by the <character set specification>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5.3 Names and identifiers

Function

Specify names.

Format

```
<SQL statement name> ::=
          <statement name>
        | <extended statement name>

<statement name> ::= <identifier>

<extended statement name> ::=
        [ <scope option> ] <simple value specification>

<dynamic cursor name> ::=
          <cursor name>
        | <extended cursor name>

<extended cursor name> ::=
        [ <scope option> ] <simple value specification>

<descriptor name> ::=
        [ <scope option> ] <simple value specification>

<scope option> ::=
          GLOBAL
        | LOCAL
```

Syntax Rules

1) Insert before SR4)a) If the <local or schema qualified name> is contained, without an intervening <schema definition>, in a <preparable statement> that is prepared in the current SQL-session by an <execute immediate statement> or a <prepare statement> or in a <direct SQL statement> that is invoked directly, then the default <unqualified schema name> for the SQL-session is implicit.

2) Insert before SR11)a) If the <schema qualified name> is contained, without an intervening <schema definition>, in a <preparable statement> that is prepared in the current SQL-session by an <execute immediate statement> or a <prepare statement> or in a <direct SQL statement> that is invoked directly, then the default <unqualified schema name> for the SQL-session is implicit.

3) Insert before SR12)a) If the <unqualified schema name> is contained in a <preparable statement> that is prepared in the current SQL-session by an <execute immediate statement> or a <prepare statement> or in a <direct SQL statement> that is invoked directly, then the default catalog name for the SQL-session is implicit.

4) Insert this SR The <simple value specification> of <extended statement name> or <extended cursor name> shall not be a <literal>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

5) Insert this SR The declared type of the <simple value specification> of <extended statement name> shall be character string with an implementation-defined character set and shall have an octet length of 128 octets or less.

6) Insert this SR The declared type of the <simple value specification> of <extended cursor name> shall be character string with an implementation-defined character set and shall have an octet length of 128 octets or less.

7) Insert this SR The declared type of the <simple value specification> of <descriptor name> shall be character string with an implementation-defined character set and shall have an octet length of 128 octets or less.

8) Insert this SR In a <descriptor name>, <extended statement name>, or <extended cursor name>, if a <scope option> is not specified, then a <scope option> of LOCAL is implicit.

Access Rules

No additional Access Rules.

General Rules

1) Insert this GR The value of an <extended statement name> identifies a statement prepared by the execution of a <prepare statement>. If a <scope option> of GLOBAL is specified, then the scope of the <extended statement name> is the current SQL-session. If a <scope option> of LOCAL is specified or implicit, then the scope of the <extended statement name> is further restricted to the <SQL-client module definition> in which the <extended statement name> appears.

2) Insert this GR A <dynamic cursor name> identifies a cursor in an <SQL dynamic statement>.

3) Insert this GR A <descriptor name> identifies a descriptor area.

4) Insert this GR A <statement name> identifies a statement prepared by the execution of a <prepare statement>. The scope of a <statement name> is the <SQL-client module definition> in which it appears and the current SQL-session.

5) Insert this GR The value of an <extended cursor name> identifies a cursor created by the execution of an <allocate cursor statement>. If a <scope option> of GLOBAL is specified, then the scope of the <extended cursor name> is the current SQL-session. If a <scope option> of LOCAL is specified or implicit, then the scope of the <extended cursor name> is further restricted to the <SQL-client module definition> in which the <extended cursor name> appears.

6) Insert this GR A <descriptor name> identifies an SQL descriptor area created by the execution of an <allocate descriptor statement>. If a <scope option> of GLOBAL is specified, then the scope of the <descriptor name> is the current SQL-session. If a <scope option> of LOCAL is specified or implicit, then the scope of the <descriptor name> is further restricted to the <SQL-client module definition> in which the <descriptor name> appears.

--`,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Conformance Rules

1) Insert this CR Without Feature B032, "Extended dynamic SQL", conforming SQL language shall not contain any <extended statement name> or <extended cursor name>.

2) Insert this CR Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <SQL statement name>, <dynamic cursor name>, or <descriptor name>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6 Scalar expressions

6.1 <value specification> and <target specification>

Function

Specify one or more values, host parameters, SQL parameters, dynamic parameters, or host variables.

Format

```
<general value specification> ::=
         !! All alternatives from ISO/IEC 9075-2
       | <dynamic parameter specification>
       | <embedded variable specification>
       | CURRENT_DEFAULT_TRANSFORM_GROUP
       | CURRENT_TRANSFORM_GROUP_FOR_TYPE <user-defined type>

<simple value specification> ::=
         !! All alternatives from ISO/IEC 9075-2
       | <embedded variable name>

<target specification> ::=
         !! All alternatives from ISO/IEC 9075-2
       | <dynamic parameter specification>
       | <embedded variable specification>

<simple target specification> ::=
         !! All alternatives from ISO/IEC 9075-2
       | <embedded variable name>

<dynamic parameter specification> ::= <question mark>

<embedded variable specification> ::=
       <embedded variable name> [ <indicator variable> ]

<indicator variable> ::=
       [ INDICATOR ] <embedded variable name>
```

Syntax Rules

1) Insert this SR The declared type of an <indicator variable> shall be exact numeric with a scale of 0 (zero).

2) Insert this SR Each <embedded variable name> shall be contained in an <embedded SQL statement>.

3) Insert this SR Each <dynamic parameter specification> shall be contained in a <preparable statement> that is dynamically prepared in the current SQL-session through the execution of a <prepare statement>.

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4) Insert this SR The data type of CURRENT_DEFAULT_TRANSFORM_GROUP and of CURRENT_TRANSFORM_GROUP_FOR_TYPE <user-defined type> is a character string. Whether the character string is fixed length or variable length, and its length if fixed length or maximum length if variable length, are implementation-defined. The character set of the character string is SQL_IDENTIFIER.

Access Rules

No additional Access Rules.

General Rules

1) Insert this GR A <dynamic parameter specification> identifies a parameter used by a dynamically prepared statement.

2) Insert this GR An <embedded variable specification> identifies a host variable or a host variable and an indicator variable.

3) Augments GR3) A <target specification> specifies a parameter used in a dynamically prepared statement or a host variable that can be assigned a value.

4) Insert this GR If an <embedded variable specification> contains an <indicator variable> and the value of the indicator variable is negative, then the value specified by the <embedded variable specification> is null; otherwise, the value specified by a <embedded variable specification> is the value of the host variable identified by the <embedded variable name>.

5) Insert this GR The value specified by CURRENT_DEFAULT_TRANSFORM_GROUP is the character string that represents the default transform group name associated with the SQL-session.

6) Insert this GR The value specified by CURRENT_TRANSFORM_GROUP_FOR_TYPE <user-defined type> is the character string that represents the transform group name associated with the data type specified by <user-defined type>.

Conformance Rules

1) Insert this CR Without Feature F611, "Indicator data types", the specific declared types of <indicator parameter>s and <indicator variable>s shall be the same implementation-defined data type.

2) Insert this CR Without Feature B031, "Basic dynamic SQL", a <general value specification> shall not be a <dynamic parameter specification>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.2 <column reference>

Function

Reference a column.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

1) Insert this AR If *CR* is a <column reference> whose qualifying table is a base table or a viewed derived table and that is contained in any of:

   — A <query expression> simply contained in a <direct select statement: multiple rows>.

   — A <sort specification list> contained in a <direct select statement: multiple rows>.

   — A <query specification> contained in a <dynamic single row select statement>.

   then let *C* be the column referenced by *CR*.

   Case:

   a) If <column reference> is contained in an <SQL schema statement>, then the applicable privileges of the <authorization identifier> that owns the containing schema shall include SELECT for *C*.

   b) Otherwise, the current privileges shall include SELECT for *C*.

   NOTE 5 – "User privileges" and "applicable privileges" are defined in Subclause 10.5, "<privileges>", in ISO/IEC 9075-2.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.3 <interval value expression>

Function

Specify an interval value.

Format

```
<interval primary> ::=
        <value expression primary> [ <interval qualifier> ]
      | <interval value function>
```

Syntax Rules

1) Insert this SR An <interval primary> shall specify <interval qualifier> only if the <interval primary> specifies a <dynamic parameter specification>.

Access Rules

No additional Access Rules.

General Rules

1) Insert before GR3)a) If *IP* immediately contains a <value expression primary> *VEP* and an explicit <interval qualifier> *IQ*, then the value of *IP* is computed by:

```
CAST (VEP AS INTERVAL IQ)
```

Conformance Rules

No additional Conformance Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7 Query expressions

7.1 <table reference>

Function

Reference a table.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

1) Insert this AR If the <table reference> is contained in a <direct select statement: multiple rows> then the current privileges shall include SELECT for at least one column of *T*.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

>

7.2 <query specification>

Function

Specify a table derived from the result of a <table expression>.

Format

No additional Format items.

Syntax Rules

1) Insert after SR16)a)ii) An <indicator variable>.

2) Insert after SR16)a)ii) A <dynamic parameter specification>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

--`,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

8 Additional common elements

8.1 <routine invocation>

Function

Invoke an SQL-invoked routine.

Format

No additional Format items.

Syntax Rules

1) Replace SR8)c)i)4)A) If A_i is an <embedded variable name> or a <host parameter specification>, then P_i shall be assignable to A_i, according to the Syntax Rules of Subclause 9.1, "Retrieval assignment", with A_i and P_i as *TARGET* and *VALUE*, respectively.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9 Data assignment rules and routine determination

9.1 Retrieval assignment

Function

Specify rules for assignments to targets that do not support null values or that support null values with indicator parameters (*e.g.*, assigning SQL-data to host variables).

Syntax Rules

1) Replace SR3)a) If *T* is either a locator parameter of an external routine, a locator variable, or a host parameter that is a character large object locator parameter, then the declared type of *V* shall be character large object type and the character string type of *T* and the declared type of *V* shall be mutually assignable.

Access Rules

No additional Access Rules.

General Rules

1) Insert before GR3) If *V* is the null value and *T* is a host variable, then

Case:

a) If an indicator variable is specified for *T*, then that indicator variable is set to −1.

b) If no indicator variable is specified for *T*, then an exception condition is raised: *data exception — null value, no indicator parameter.*

2) Insert after GR3) If *V* is not the null value, *T* is a host variable, and *T* has an indicator variable, then

Case:

a) If the declared type of *T* is character string, bit string, or binary string and the length in characters, bits, or octets, respectively, *M* of *V* is greater than the length in characters, bits, or octets, respectively, of *T*, then the indicator parameter is set to *M*. If *M* exceeds the maximum value that the indicator parameter can contain, then an exception condition is raised: *data exception — indicator overflow*.

b) Otherwise, the indicator variable is set to 0 (zero).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

9.2 Store assignment

Function

Specify rules for assignments where the target permits null without the use of indicator variables, such as storing SQL-data.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Insert before GR2)a)iii) If V is a host variable and contains an indicator variable, then

Case:

1) If the value of the indicator variable is equal to -1, then T is set to the null value.

2) If the value of the indicator variable is less than -1, then an exception condition is raised: *data exception — invalid indicator parameter value*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.3 Data types of results of aggregations

Function

Specify the result data type of the result of an aggregation over values of compatible data types, such as <case expression>s, <collection value expression>s, or a column in the result of a <query expression>.

Syntax Rules

1) Replace SR1) Let *IDTS* be a set of data types specified in an application of this Subclause. let *DTS* be the set of data types in *IDTS* excluding any data types that are undefined. If the cardinality of *DTS* is 0 (zero), then the result data type is undefined and no further Rules of this Subclause are evaluated.

NOTE 6 –

The notion of "undefined data type" is defined in Subclause 15.6, "<prepare statement>".

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10 Schema definition and manipulation

10.1 <check constraint definition>

Function

Specify a condition for the SQL-data.

Format

No additional Format items.

Syntax Rules

1) Insert this SR The <search condition> shall also not contain an <embedded variable specification> or a <dynamic parameter specification>.

Access Rules

1) No additional Access Rules.

General Rules

1) No additional General Rules.

Conformance Rules

No additional Conformance Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.2 <view definition>

Function

Define a viewed table.

Format

No additional Format items.

Syntax Rules

1) Insert this SR The <view definition> shall not contain an <embedded variable specification> or a <dynamic parameter specification>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.3 <assertion definition>

Function

Specify an integrity constraint by means of an assertion and specify when the assertion is to be checked.

Format

No additional Format items.

Syntax Rules

1) Augments SR4) The <search condition> shall also not contain an <embedded variable specification> or a <dynamic parameter specification>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.4 <trigger definition>

Function

Define triggered SQL-statements.

Format

No additional Format items.

Syntax Rules

1) Insert this SR The <triggered action> shall not contain a <dynamic parameter specification> or an <embedded variable name>.

2) Insert this SR The <triggered SQL statement> shall not generally contain an <SQL dynamic statement>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.5 <SQL-invoked routine>

Function

Define an SQL-invoked routine.

Format

No additional Format items.

Syntax Rules

1) Insert before SR18)c) The <SQL routine body> shall not contain an <SQL dynamic statement>.

2) Insert this SR An <SQL-invoked routine> shall not contain a <dynamic parameter specification> or an <embedded variable name>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

11 SQL-client modules

11.1 <SQL-client module definition>

Function

Define an SQL-client module.

Format

```
<module authorization clause> ::=
         SCHEMA <schema name>
       | AUTHORIZATION <module authorization identifier>
            [ FOR STATIC { ONLY | AND DYNAMIC } ]
       | SCHEMA <schema name> AUTHORIZATION <module authorization identifier>
            [ FOR STATIC { ONLY | AND DYNAMIC } ]

<module contents> ::=
         !! All alternatives from ISO/IEC 9075-2
       | <dynamic declare cursor>
```

Syntax Rules

1) Insert this SR A <dynamic declare cursor> shall precede in the text of the <SQL-client module definition> any <externally-invoked procedure> that references the <cursor name> of the <dynamic declare cursor>.

2) Insert this SR If neither FOR STATIC ONLY nor FOR STATIC AND DYNAMIC is specified, then FOR STATIC AND DYNAMIC is implicit.

Access Rules

No additional Access Rules.

General Rules

1) Augments GR5) After the last time that an SQL-agent performs a call of an <externally-invoked procedure>, following the effective execution of a <rollback statement> or a <commit statement>, a <deallocate descriptor statement> that specifies

```
DEALLOCATE DESCRIPTOR D
```

is effectively executed, where *D* is the <descriptor name> of any system descriptor area that is currently allocated within an SQL-session associated with the SQL-agent.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) Insert this GR If FOR STATIC ONLY is specified, then a *shadow module* is effectively created to contain <externally-invoked procedure>s corresponding to each SQL-statement that is prepared or executed dynamically by <prepare statement>s or <execute immediate statement>s contained in this <SQL-client module definition>.

Conformance Rules

1) Insert this CR Without Feature B051, "Enhanced execution rights", conforming SQL language shall not specify FOR STATIC ONLY or FOR STATIC AND DYNAMIC.

2) Insert this CR Without Feature B031, "Basic dynamic SQL", a <module contents> shall not be a <dynamic declare cursor>.

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.2 Calls to an <externally-invoked procedure>

Function

Define a routine.

Format

No additional Format items.

Syntax Rules

1) Augments SR2)e) The base type of any host parameter shall be an Ada data type declared in an Ada package named Interfaces.SQL, as specified in ISO/IEC 9075-2, with the following additional constants:

```
ATTEMPT_TO_ASSIGN_TO_NON_UPDATABLE_COLUMN_NO_SUBCLASS:
        constant SQLSTATE_TYPE := "0U000";
ATTEMPT_TO_ASSIGN_TO_ORDERING_COLUMN_NO_SUBCLASS:
        constant SQLSTATE_TYPE := "0V000";
DYNAMIC_SQL_ERROR_NO_SUBCLASS:
        constant SQLSTATE_TYPE := "07000";
DYNAMIC_SQL_ERROR_CURSOR_SPECIFICATION_CANNOT_BE_EXECUTED:
        constant SQLSTATE_TYPE := "07003";
DYNAMIC_SQL_ERROR_INVALID_DESCRIPTOR_COUNT:
        constant SQLSTATE_TYPE := "07008";
DYNAMIC_SQL_ERROR_INVALID_DESCRIPTOR_INDEX:
        constant SQLSTATE_TYPE := "07009";
DYNAMIC_SQL_ERROR_PREPARED_STATEMENT_NOT_A_CURSOR_SPECIFICATION:
        constant SQLSTATE_TYPE := "07005";
DYNAMIC_SQL_ERROR_RESTRICTED_DATA_TYPE_ATTRIBUTE_VIOLATION:
        constant SQLSTATE_TYPE := "07006";
DYNAMIC_SQL_ERROR_DATA_TYPE_TRANSFORM_FUNCTION_VIOLATION:
        constant SQLSTATE_TYPE := "0700B";
DYNAMIC_SQL_ERROR_UNDEFINED_DATA_VALUE:
        constant SQLSTATE_TYPE := "0700C";
DYNAMIC_SQL_ERROR_UNDEFINED_DATA_TARGET:
        constant SQLSTATE_TYPE := "0700D";
DYNAMIC_SQL_ERROR_UNDEFINED_LEVEL_VALUE:
        constant SQLSTATE_TYPE := "0700E";
DYNAMIC_SQL_ERROR_UNDEFINED_DATETIME_INTERVAL_CODE:
        constant SQLSTATE_TYPE := "0700F";
DYNAMIC_SQL_ERROR_USING_CLAUSE_DOES_NOT_MATCH_DYNAMIC_PARAMETER_SPEC:
        constant SQLSTATE_TYPE := "07001";
DYNAMIC_SQL_ERROR_USING_CLAUSE_DOES_NOT_MATCH_TARGET_SPEC:
        constant SQLSTATE_TYPE := "07002";
DYNAMIC_SQL_ERROR_USING_CLAUSE_REQUIRED_FOR_DYNAMIC_PARAMETERS:
        constant SQLSTATE_TYPE := "07004";
DYNAMIC_SQL_ERROR_USING_CLAUSE_REQUIRED_FOR_RESULT_FIELDS:
        constant SQLSTATE_TYPE := "07007";
INVALID_CHARACTER_SET_NAME_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="2C000";
INVALID_SCHEMA_NAME_LIST_SPECIFICATION_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="0E000";
INVALID_SQL_INVOKED_PROCEDURE_REFERENCE_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="0M000";
INVALID_TRANSFORM_GROUP_NAME_SPECIFICATION_NO_SUBCLASS:
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
        constant SQLSTATE_TYPE :="0S000";
TARGET_TABLE_DISAGREES_WITH_CURSOR_SPECIFICATION_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="0T000";
WARNING_INSUFFICIENT_ITEM_DESCRIPTOR_AREAS:
        constant SQLSTATE_TYPE := "01005";
```

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

11.3 <SQL procedure statement>

Function

Define all of the SQL-statements that are <SQL procedure statement>s.

Format

```
<SQL executable statement> ::=
         !! All alternatives from ISO/IEC 9075-2
       | <SQL dynamic statement>

<SQL session statement> ::=
         !! All alternatives from ISO/IEC 9075-2
       | <set session characteristics statement>
       | <set catalog statement>
       | <set schema statement>
       | <set names statement>
       | <set path statement>
       | <set transform group statement>

<SQL dynamic statement> ::=
         <system descriptor statement>
       | <prepare statement>
       | <deallocate prepared statement>
       | <describe statement>
       | <execute statement>
       | <execute immediate statement>
       | <SQL dynamic data statement>

<SQL dynamic data statement> ::=
         <allocate cursor statement>
       | <dynamic open statement>
       | <dynamic fetch statement>
       | <dynamic close statement>
       | <dynamic delete statement: positioned>
       | <dynamic update statement: positioned>

<system descriptor statement> ::=
         <allocate descriptor statement>
       | <deallocate descriptor statement>
       | <set descriptor statement>
       | <get descriptor statement>
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

General Rules

1) Replace GR2) If the non-dynamic or dynamic execution of an <SQL data statement>, <SQL dynamic data statement>, <dynamic select statement>, or <dynamic single row select statement> occurs within the same SQL-transaction as the non-dynamic or dynamic execution of an SQL-schema statement and this is not allowed by the SQL-implementation, then an exception condition is raised: *invalid transaction state — schema and data statement mixing not supported*.

2) Replace GR5)a)i)6) If *S* successfully initiated or resumed an SQL-session, then subsequent calls to an <externally-invoked procedure> and subsequent invocations of <direct SQL statement>s by the SQL-agent are associated with the SQL-session until the SQL-agent terminates the SQL-session or makes it dormant.

3) Replace GR5)a)iii)1)D) Subsequent calls to an <externally-invoked procedure> and subsequent invocations of <direct SQL statement>s by the SQL-agent are associated with the SQL-session until the SQL-agent terminates the SQL-session or makes it dormant.

4) Replace GR5)a)iii)3)A) An SQL-transaction is effectively initiated and associated with this call and with subsequent calls of any <externally-invoked procedure> by that SQL-agent and with this and subsequent invocations of <direct SQL statement>s by that SQL-agent until the SQL-agent terminates that SQL-transaction.

Conformance Rules

1) Insert this CR Without Feature B031, "Basic dynamic SQL", an <SQL procedure statement> shall not be an <SQL dynamic statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12 Data manipulation

12.1 <select statement: single row>

Function

Retrieve values from a specified row of a table.

Format

No additional Format items.

Syntax Rules

1) Insert after SR4) For each <target specification> *TS* that is an <embedded variable name>, then the Syntax Rules of Subclause 9.1, "Retrieval assignment", shall apply to *TS* and the corresponding element of the <select list>, as *TARGET* and *VALUE*, respectively.

Access Rules

None.

General Rules

1) Insert before GR6) If the <target specification> *TS* is an <embedded variable name>, then the first value in the row of *Q* is assigned to *TS* according to the General Rules of Subclause 9.1, "Retrieval assignment", as *VALUE* and *TARGET*, respectively.

2) Insert after GR5) For each <target specification> *TS* that is an <embedded variable name>, the corresponding value in the row of *Q* is assigned to *TS* according to the General Rules of Subclause 9.1, "Retrieval assignment", as *VALUE* and *TARGET*, respectively. The assignment of values to targets in the <select target list> is in an implementation-dependent order.

Conformance Rules

No additional Conformance Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12.2 <free locator statement>

Function

Remove the association between a locator variable and the value that is represented by that locator.

Format

```
<locator reference> ::=
         !! All alternatives from ISO/IEC 9075-2
       | <host variable name>
```

Syntax Rules

1) Insert after SR1) Each host variable identified by the <host variable name> immediately contained in <locator reference> shall be a binary object locator variable, a character large object locator variable, an array locator variable, or a user-defined type locator variable.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

13 Transaction management

13.1 <commit statement>

Function

Terminate the current SQL-transaction with commit.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Insert this GR The <statement name> or <extended statement name> of every held cursor remains valid.

Conformance Rules

No additional Conformance Rules.

--`,,`--`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

5 Language Bindings (SQL/Bindings)

14 Session management

14.1 <set catalog statement>

Function

Set the default catalog name for unqualified <schema name>s in <preparable statement>s that are prepared in the current SQL-session by an <execute immediate statement> or a <prepare statement> and in <direct SQL statement>s that are invoked directly.

Format

```
<set catalog statement> ::=
       SET <catalog name characteristic>

<catalog name characteristic> ::=
       CATALOG <value specification>
```

Syntax Rules

1) The declared type of the <value specification> shall be an SQL character data type.

Access Rules

None.

General Rules

1) Let *S* be <value specification> and let *V* be the character string that is the value of

   ```
   TRIM ( BOTH ' ' FROM S )
   ```

2) If *V* does not conform to the Format and Syntax Rules of a <catalog name>, then an exception condition is raised: *invalid catalog name*.

3) The default catalog name of the current SQL-session is set to *V*.

Conformance Rules

1) Without Feature F761, "Session management", and Feature F651, "Catalog name qualifiers", conforming SQL language shall not contain any <set catalog statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

14.2 <set schema statement>

Function

Set the default schema name for unqualified <schema qualified name>s in <preparable statement>s that are prepared in the current SQL-session by an <execute immediate statement> or a <prepare statement> and in <direct SQL statement>s that are invoked directly.

Format

```
<set schema statement> ::=
      SET <schema name characteristic>

<schema name characteristic> ::=
      SCHEMA <value specification>
```

Syntax Rules

1) The declared type of the <value specification> shall be an SQL character data type.

Access Rules

None.

General Rules

1) Let *S* be <value specification> and let *V* be the character string that is the value of

   ```
   TRIM ( BOTH ' ' FROM S )
   ```

2) If *V* does not conform to the Format and Syntax Rules of a <schema name>, then an exception condition is raised: *invalid schema name*.

3) Case:

   a) If *V* conforms to the Format and Syntax Rules for a <schema name> that contains a <catalog name>, then let *X* be the <catalog name> part and let *Y* be the <unqualified schema name> part of *V*. The following statement is implicitly executed:

      ```
      SET CATALOG 'X'
      ```

      and the <set schema statement> is effectively replaced by:

      ```
      SET SCHEMA 'Y'
      ```

   b) Otherwise, the default unqualified schema name of the current SQL-session is set to *V*.

Conformance Rules

1) Without Feature F761, "Session management", conforming SQL language shall not contain any <set schema statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

14.3 <set names statement>

Function

Set the default character set name for <character string literal>s in <preparable statement>s that are prepared in the current SQL-session by an <execute immediate statement> or a <prepare statement> and in <direct SQL statement>s that are invoked directly.

Format

```
<set names statement> ::=
      SET <character set name characteristic>

<character set name characteristic> ::=
      NAMES <value specification>
```

Syntax Rules

1) The declared type of the <value specification> shall be an SQL character data type.

Access Rules

None.

General Rules

1) Let *S* be <value specification> and let *V* be the character string that is the value of

   ```
   TRIM ( BOTH ' ' FROM S )
   ```

2) If *V* does not conform to the Format and Syntax Rules of a <character set name>, then an exception condition is raised: *invalid character set name*.

3) The default character set name of the current SQL-session is set to *V*.

Conformance Rules

1) Without Feature F761, "Session management", and either Feature F451, "Character set definition", or Feature F461, "Named character sets", conforming SQL language shall not contain any <set names statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

14.4 <set path statement>

Function

Set the SQL-path used to determine the subject routine of <routine invocation>s with unqualified <routine name>s in <preparable statement>s that are prepared in the current SQL-session by an <execute immediate statement> or a <prepare statement> and in <direct SQL statement>s, respectively, that are invoked directly. The SQL-path remains the current SQL-path of the SQL-session until another SQL-path is successfully set.

Format

```
<set path statement> ::=
      SET <SQL-path characteristic>

<SQL-path characteristic> ::=
      PATH <value specification>
```

Syntax Rules

1) The declared type of the <value specification> shall be an SQL character data type.

Access Rules

None.

General Rules

1) Let *S* be <value specification> and let *V* be the character string that is the value of

   ```
   TRIM ( BOTH ' ' FROM S )
   ```

   a) If *V* does not conform to the Format and Syntax Rules of a <schema name list>, then an exception condition is raised: *invalid schema name list specification*.

   b) The SQL-path of the current SQL-session is set to *V*.

   NOTE 7 – A <set path statement> that is executed between a <prepare statement> and an <execute statement> has no effect on the prepared statement.

Conformance Rules

1) Without Feature S071, "SQL paths in function and type name resolution", Conforming SQL language shall not contain any <set path statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

14.5 <set transform group statement>

Function

Set the group name that identifies the group of transform functions for mapping values of user-defined data types to predefined data types.

Format

```
<set transform group statement> ::=
       SET <transform group characteristic>

<transform group characteristic> ::=
         DEFAULT TRANSFORM GROUP <value specification>
       | TRANSFORM GROUP FOR TYPE <user-defined type> <value specification>
```

Syntax Rules

1) The data type of the <value specification> shall be an SQL character data type.

2) If "TRANSFORM GROUP FOR TYPE" is specified, then let *UDT* be the user-defined type identified by <user-defined type>.

Access Rules

None.

General Rules

1) Let *S* be <value specification> and let *V* be the character string that is the value of

   ```
   TRIM ( BOTH ' ' FROM S )
   ```

   a) If *V* does not conform to the Format and Syntax Rules of an <identifier>, then an exception condition is raised: *invalid transform group name specification*.

   b) Case:

      i) If "TRANSFORM GROUP FOR TYPE" is specified, then the transform group name corresponding to all subtypes of *UDT* for the current SQL-session is set to *V*.

      ii) Otherwise, the default transform group name for the current SQL-session is set to *V*.

   NOTE 8 – A <set transform group statement> that is executed after a <prepare statement> has no effect on the prepared statement.

Conformance Rules

1) Without Feature S241, "Transform functions", conforming SQL language shall not contain any <set transform group statement>.

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

--`,,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15 Dynamic SQL

15.1 Description of SQL descriptor areas

Function

Specify the identifiers, data types, and codes used in SQL item descriptor areas.

Syntax Rules

1) An SQL item descriptor area comprises the items specified in Table 3, "Data types of <key word>s used in SQL item descriptor areas".

2) An SQL descriptor area comprises the items specified in Table 2, "Data types of <key word>s used in the header of SQL descriptor areas", and one or more occurrences of an SQL item descriptor area.

3) Given an SQL item descriptor area *IDA* in which the value of LEVEL is N, the *immediately subordinate descriptor areas* of *IDA* are those SQL item descriptor areas in which the value of LEVEL is N+1 and whose position in the SQL descriptor area follows that of *IDA* and precedes that of any SQL item descriptor area in which the value of LEVEL is less than N+1.

   The *subordinate descriptor areas* of *IDA* are those SQL item descriptor areas that are immediately subordinate descriptor areas of *IDA* or that are subordinate descriptor areas of an SQL item descriptor area that is immediately subordinate to *IDA*.

4) Given a data type *DT* and its descriptor *DE*, the *immediately subordinate descriptors* of *DE* are defined to be:

   Case:

   a) If *DT* is a row type, then the field descriptors of the fields of *DT*. The *i*-th immediately subordinate descriptor is the descriptor of the *i*-th field of *DT*.

   b) If *DT* is a collection type, then the descriptor of the associated element type of *DT*.

   The *subordinate descriptors* of *DE* are those descriptors that are immediately subordinate descriptors of *DE* or that are subordinate descriptors of a descriptor that is immediately subordinate to *DE*.

5) Given a descriptor *DE*, let SDE_j represent its *j*-th immediately subordinate descriptor. There is an implied ordering of the subordinate descriptors of *DE*, such that:

   a) SDE_1 is in the first ordinal position.

   b) The ordinal position of SDE_{j+1} is $K+NS+1$, where K is the ordinal position of SDE_j and NS is the number of subordinate descriptors of SDE_j. The implicitly ordered subordinate descriptors of SDE_j occupy contiguous ordinal positions starting at position K+1.

6) An item descriptor area *IDA* is *valid* if and only if TYPE indicates a code defined in Table 4, "Codes used for SQL data types in Dynamic SQL", and one of the following is true:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Case:

a) TYPE indicates NUMERIC and PRECISION and SCALE are valid precision and scale values for the NUMERIC data type.

b) TYPE indicates DECIMAL and PRECISION and SCALE are valid precision and scale values for the DECIMAL data type.

c) TYPE indicates FLOAT and PRECISION is a valid precision value for the FLOAT data type.

d) TYPE indicates INTEGER, SMALLINT, REAL, or DOUBLE PRECISION.

e) TYPE indicates BOOLEAN.

f) TYPE indicates BIT or BIT VARYING and LENGTH is a valid length value for the BIT data type.

g) TYPE indicates CHARACTER, CHARACTER VARYING, or CHARACTER LARGE OBJECT, LENGTH is a valid length value for TYPE, and CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are a valid qualified character set name for TYPE.

h) TYPE indicates BINARY LARGE OBJECT and LENGTH is a valid length value for the BINARY LARGE OBJECT data type.

i) TYPE indicates CHARACTER LARGE OBJECT LOCATOR.

j) TYPE indicates BINARY LARGE OBJECT LOCATOR.

k) TYPE indicates USER-DEFINED TYPE LOCATOR and USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are a valid qualified user-defined type name.

l) TYPE indicates a <datetime type>, DATETIME_INTERVAL_CODE is a code specified in Table 5, "Codes associated with datetime data types in Dynamic SQL", and PRECISION is a valid value for the <time precision> or <timestamp precision> of the indicated datetime data type.

m) TYPE indicates an <interval type>, DATETIME_INTERVAL_CODE is a code specified in Table 6, "Codes used for <interval qualifier>s in Dynamic SQL", and DATETIME_INTERVAL_PRECISION and PRECISION are valid values for <interval leading field precision> and <interval fractional seconds precision> for an <interval qualifier>.

n) TYPE indicates REF, LENGTH is the implementation-defined length in octets for the REF type, and USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are a valid qualified user-defined type name, and SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME are a valid qualified table name.

o) TYPE indicates ROW, the value *N* of DEGREE is a valid value for the degree of a row type, there are exactly *N* immediately subordinate descriptor areas of *IDA* and those SQL item descriptor areas are valid.

p) TYPE indicates ARRAY or ARRAY LOCATOR, the value of CARDINALITY is a valid value for the cardinality of an array, there is exactly one immediately subordinate descriptor area of *IDA* and that SQL item descriptor area is valid.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

q) TYPE indicates an implementation-defined data type.

7) The declared type *T* of a <simple value specification> or a <simple target specification> *SVT* is said to *match* the data type specified by a valid item descriptor area *IDA* if and only if one of the following conditions is true.

Case:

a) TYPE indicates NUMERIC and *T* is specified by NUMERIC(*P*,*S*), where *P* is the value of PRECISION and *S* is the value of SCALE.

b) TYPE indicates DECIMAL and *T* is specified by DECIMAL(*P*,*S*), where *P* is the value of PRECISION and *S* is the value of SCALE.

c) TYPE indicates INTEGER and *T* is specified by INTEGER.

d) TYPE indicates SMALLINT and *T* is specified by SMALLINT.

e) TYPE indicates FLOAT and *T* is specified by FLOAT(*P*), where *P* is the value of PRECISION.

f) TYPE indicates REAL and *T* is specified by REAL.

g) TYPE indicates DOUBLE PRECISION and *T* is specified by DOUBLE PRECISION.

h) TYPE indicates BOOLEAN and *T* is specified by BOOLEAN.

i) TYPE indicates BIT and *T* is specified by BIT(*L*), where *L* is the value of LENGTH.

j) TYPE indicates BIT VARYING and *T* is specified by BIT VARYING(*L*), where

Case:

i) *SVT* is a <simple value specification> and *L* is the value of LENGTH.

ii) *SVT* is a <simple target specification> and *L* is not less than the value of LENGTH.

k) TYPE indicates CHARACTER and *T* is specified by CHARACTER(*L*), where *L* is the value of LENGTH and the <character set specification> formed by the values of CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME identifies the character set of *SVT*.

l) Either TYPE indicates CHARACTER VARYING and *T* is specified by CHARACTER VARYING(*L*) or type indicates CHARACTER LARGE OBJECT and *T* is specified by CHARACTER LARGE OBJECT(*L*), where the <character set specification> formed by the values of CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME identifies the character set of *SVT* and

Case:

i) *SVT* is a <simple value specification> and *L* is the value of LENGTH.

ii) *SVT* is a <simple target specification> and *L* is not less than the value of LENGTH.

m) TYPE indicates BINARY LARGE OBJECT and *T* is specified by BINARY LARGE OBJECT(*L*) and

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Case:

i) *STV* is a <simple value specification> and *L* is the value of LENGTH.

ii) *STV* is a <simple target specification> and *L* is not less than the value of LENGTH.

n) TYPE indicates CHARACTER LARGE OBJECT LOCATOR and *T* is specified by CHARACTER LARGE OBJECT LOCATOR.

o) TYPE indicates BINARY LARGE OBJECT LOCATOR and *T* is specified by BINARY LARGE OBJECT LOCATOR.

p) TYPE indicates USER-DEFINED TYPE and *T* is specified by USER-DEFINED TYPE LOCATOR, where the <user-defined type name> formed by the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME identifies the associated user-defined type of *SVT*.

q) TYPE indicates REF and *T* is specified by REF, where the <user-defined type name> formed by the values of USER_DEFINED_DATA_TYPE_CATALOG, USER_DEFINED_DATA_TYPE_SCHEMA, and USER_DEFINED_DATA_TYPE_NAME identifies the row type of *SVT*, and SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME identify the scope of the reference type.

r) TYPE indicates ROW, and *T* is a row type with degree *D*, where *D* is the value of DEGREE, and the data type of the *i*-th field of *SVT* matches the data type specified by the *i*-th immediately subordinate descriptor area of *IDA*.

s) TYPE indicates ARRAY and *T* is an array type with cardinality *C* and the data type of the element type of *T* matches the immediately subordinate descriptor area of *IDA*, and

Case:

i) *SVT* is a <simple value specification> and *C* is the value of CARDINALITY.

ii) *SVT* is a <simple target specification> and *C* is not less than the value of CARDINALITY.

t) TYPE indicates ARRAY LOCATOR and *T* is an array locator type whose associated array type has cardinality *C* and the data type of the element type of the associated array type of *T* matches the immediately subordinate descriptor area of *IDA*, and

Case:

i) *SVT* is a <simple value specification> and *C* is the value of CARDINALITY.

ii) *SVT* is a <simple target specification> and *C* is not less than the value of CARDINALITY.

u) TYPE indicates a data type from Table 4, "Codes used for SQL data types in Dynamic SQL", other than an implementation-defined data type and *T* satisfies the implementation-defined rules for matching that data type.

v) TYPE indicates an implementation-defined data type and *T* satisfies the implementation-defined rules for matching that data type.

8) A data type *DT* is said to be *represented* by an SQL item descriptor area if a <simply value specification> of type *DT* matches the SQL item descriptor area.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Table 2—Data types of <key word>s used in the header of SQL descriptor areas

<key word>	Data Type
COUNT	exact numeric with scale 0 (zero)
DYNAMIC_FUNCTION	character string with character set SQL_IDENTIFIER and length not less than 128 characters
DYNAMIC_FUNCTION_CODE	exact numeric with scale 0 (zero)
KEY_TYPE	exact numeric with scale 0 (zero)
TOP_LEVEL_COUNT	exact numeric with scale 0 (zero)

Table 3—Data types of <key word>s used in SQL item descriptor areas

<key word>	Data Type
CARDINALITY	exact numeric with scale 0 (zero)
CHARACTER_SET_CATALOG	character string with character set SQL_IDENTIFIER and length not less than 128 characters
CHARACTER_SET_NAME	character string with character set SQL_IDENTIFIER and length not less than 128 characters
CHARACTER_SET_SCHEMA	character string with character set SQL_IDENTIFIER and length not less than 128 characters
COLLATION_CATALOG	character string with character set SQL_IDENTIFIER and length not less than 128 characters
COLLATION_NAME	character string with character set SQL_IDENTIFIER and length not less than 128 characters
COLLATION_SCHEMA	character string with character set SQL_IDENTIFIER and length not less than 128 characters
DATA	matches the data type represented by the SQL item descriptor area
DATETIME_INTERVAL_CODE	exact numeric with scale 0 (zero)
DATETIME_INTERVAL_ PRECISION	exact numeric with scale 0 (zero)
DEGREE	exact numeric with scale 0 (zero)
INDICATOR	exact numeric with scale 0 (zero)
KEY_MEMBER	exact numeric with scale 0 (zero)
LENGTH	exact numeric with scale 0 (zero)
LEVEL	exact numeric with scale 0 (zero)
NAME	character string with character set SQL_IDENTIFIER and length not less than 128 characters
NULLABLE	exact numeric with scale 0 (zero)
OCTET_LENGTH	exact numeric with scale 0 (zero)
PARAMETER_MODE	exact numeric with scale 0 (zero)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 3—Data types of <key word>s used in SQL item descriptor areas (Cont.)

<key word>	Data Type
PARAMETER_ORDINAL_POSITION	exact numeric with scale 0 (zero)
PARAMETER_SPECIFIC_CATALOG	character string with character set SQL_IDENTIFIER and length not less than 128 characters
PARAMETER_SPECIFIC_NAME	character string with character set SQL_IDENTIFIER and length not less than 128 characters
PARAMETER_SPECIFIC_SCHEMA	character string with character set SQL_IDENTIFIER and length not less than 128 characters
PRECISION	exact numeric with scale 0 (zero)
RETURNED_CARDINALITY	exact numeric with scale 0 (zero)
RETURNED_LENGTH	exact numeric with scale 0 (zero)
RETURNED_OCTET_LENGTH	exact numeric with scale 0 (zero)
SCALE	exact numeric with scale 0 (zero)
SCOPE_CATALOG	character string with character set SQL_IDENTIFIER and length not less than 128 characters
SCOPE_NAME	character string with character set SQL_IDENTIFIER and length not less than 128 characters
SCOPE_SCHEMA	character string with character set SQL_IDENTIFIER and length not less than 128 characters
TYPE	exact numeric with scale 0 (zero)
UNNAMED	exact numeric with scale 0 (zero)
USER_DEFINED_TYPE_CATALOG	character string with character set SQL_IDENTIFIER and length not less than 128 characters
USER_DEFINED_TYPE_NAME	character string with character set SQL_IDENTIFIER and length not less than 128 characters
USER_DEFINED_TYPE_SCHEMA	character string with character set SQL_IDENTIFIER and length not less than 128 characters

NOTE 9 – "Matches" and "represented by", as applied to the relationship between a data type and an SQL item descriptor area are defined in the Syntax Rules of this Subclause.

Access Rules

None.

General Rules

1) Table 4, "Codes used for SQL data types in Dynamic SQL", specifies the codes associated with the SQL data types.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 4—Codes used for SQL data types in Dynamic SQL

Data Type	Code
Implementation-defined data types	< 0 (zero)
ARRAY	50
ARRAY LOCATOR	51
BIT	14
BIT VARYING	15
BLOB	30
BLOB LOCATOR	31
BOOLEAN	16
CHARACTER	1 (one)
CHARACTER VARYING	12
CLOB	40
CLOB LOCATOR	41
DATE, TIME WITHOUT TIME ZONE, TIME WITH TIME ZONE, TIMESTAMP WITHOUT TIME ZONE, or TIMESTAMP WITH TIME ZONE	9
DECIMAL	3
DOUBLE PRECISION	8
FLOAT	6
INTEGER	4
INTERVAL	10
NUMERIC	2
REAL	7
SMALLINT	5
USER-DEFINED TYPE LOCATOR	18
ROW TYPE	19
REF	20
User-defined types	17

2) Table 5, "Codes associated with datetime data types in Dynamic SQL", specifies the codes associated with the datetime data types.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 5—Codes associated with datetime data types in Dynamic SQL

Datetime Data Type	Code
DATE	1 (one)
TIME WITH TIME ZONE	4
TIME WITHOUT TIME ZONE	2
TIMESTAMP WITH TIME ZONE	5
TIMESTAMP WITHOUT TIME ZONE	3

3) Table 6, "Codes used for <interval qualifier>s in Dynamic SQL", specifies the codes associated with <interval qualifier>s for interval data types.

Table 6—Codes used for <interval qualifier>s in Dynamic SQL

Datetime Qualifier	Code
DAY	3
DAY TO HOUR	8
DAY TO MINUTE	9
DAY TO SECOND	10
HOUR	4
HOUR TO MINUTE	11
HOUR TO SECOND	12
MINUTE	5
MINUTE TO SECOND	13
MONTH	2
SECOND	6
YEAR	1 (one)
YEAR TO MONTH	7

4) The value of DYNAMIC_FUNCTION is a character string that identifies the type of the prepared or executed SQL-statement. Table 9, "SQL-statement codes", specifies the identifier of the SQL-statements.

5) The value of DYNAMIC_FUNCTION_CODE is a number that identifies the type of the prepared or executed SQL-statement. Table 9, "SQL-statement codes", specifies the identifier of the SQL-statements.

6) Table 7, "Codes used for input/output SQL parameter modes in Dynamic SQL", specifies the codes used for the PARAMETER_MODE item descriptor field when describing a <call statement>.

--`,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 7—Codes used for input/output SQL parameter modes in Dynamic SQL

Parameter mode	Code
PARAMETER_MODE_IN	1 (one)
PARAMETER_MODE_INOUT	2
PARAMETER_MODE_OUT	4

Conformance Rules

None.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.2 <allocate descriptor statement>

Function

Allocate an SQL descriptor area.

Format

```
<allocate descriptor statement> ::=
      ALLOCATE [ SQL ] DESCRIPTOR <descriptor name> [ WITH MAX <occurrences> ]

<occurrences> ::= <simple value specification>
```

Syntax Rules

1) The declared type of <occurrences> shall be exact numeric with scale 0 (zero).

2) If WITH MAX <occurrences> is not specified, then an implementation-defined default value for <occurrences> that is greater than 0 (zero) is implicit.

Access Rules

None.

General Rules

1) Let *S* be the <simple value specification> that is immediately contained in <descriptor name> and let *V* be the character string that is the result of

   ```
   TRIM ( BOTH ' ' FROM S )
   ```

   If *V* does not conform to the Format and Syntax Rules of an <identifier>, then an exception condition is raised: *invalid SQL descriptor name*.

2) The <allocate descriptor statement> allocates an SQL descriptor area whose name is *V* and whose scope is specified by the <scope option>. The descriptor area will have at least <occurrences> number of item descriptor areas. The value of LEVEL in each of the item descriptor areas is set to 0 (zero). The values of all other fields in the SQL descriptor area are initially undefined. If an SQL descriptor has already been allocated whose name is *V*, whose scope is specified by the <scope option>, and that has not yet been deallocated, then an exception condition is raised: *invalid SQL descriptor name*.

3) If <occurrences> is less than 1 (one) or is greater than an implementation-defined maximum value, then an exception condition is raised: *dynamic SQL error — invalid descriptor index*. The maximum number of SQL descriptor areas that can be allocated at one time is implementation-defined.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Conformance Rules

1) Without Feature B032, "Extended dynamic SQL", an <occurrences> and a <descriptor name> shall be a <literal>.

2) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <allocate descriptor statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.3 <deallocate descriptor statement>

Function

Deallocate an SQL descriptor area.

Format

```
<deallocate descriptor statement> ::=
       DEALLOCATE [ SQL ] DESCRIPTOR <descriptor name>
```

Syntax Rules

None.

Access Rules

None.

General Rules

1) The <deallocate descriptor statement> deallocates an SQL descriptor area whose name is the value of the <descriptor name>'s <simple value specification> and whose scope is specified by the <scope option>. If an SQL descriptor is not currently allocated whose name is the value of the <descriptor name>'s <simple value specification> and whose scope is specified by the <scope option>, then an exception condition is raised: *invalid SQL descriptor name.*

Conformance Rules

1) Without Feature B032, "Extended dynamic SQL", a <descriptor name> shall be a <literal>.

2) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <deallocate descriptor statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.4 <get descriptor statement>

Function

Get information from an SQL descriptor area.

Format

```
<get descriptor statement> ::=
       GET [ SQL ] DESCRIPTOR <descriptor name> <get descriptor information>

<get descriptor information> ::=
         <get header information> [ { <comma> <get header information> }... ]
       | VALUE <item number>
           <get item information> [ { <comma> <get item information> }... ]

<get header information> ::=
       <simple target specification 1> <equals operator> <header item name>

<header item name> ::=
         COUNT
       | KEY_TYPE
       | DYNAMIC_FUNCTION
       | DYNAMIC_FUNCTION_CODE
       | TOP_LEVEL_COUNT

<get item information> ::=
       <simple target specification 2> <equals operator> <descriptor item name>

<item number> ::= <simple value specification>

<simple target specification 1> ::= <simple target specification>

<simple target specification 2> ::= <simple target specification>

<descriptor item name> ::=
         CARDINALITY
       | CHARACTER_SET_CATALOG
       | CHARACTER_SET_NAME
       | CHARACTER_SET_SCHEMA
       | COLLATION_CATALOG
       | COLLATION_NAME
       | COLLATION_SCHEMA
       | DATA
       | DATETIME_INTERVAL_CODE
       | DATETIME_INTERVAL_PRECISION
       | DEGREE
       | INDICATOR
       | KEY_MEMBER
       | LENGTH
       | LEVEL
       | NAME
       | NULLABLE
       | OCTET_LENGTH
       | PARAMETER_MODE
       | PARAMETER_ORDINAL_POSITION
       | PARAMETER_SPECIFIC_CATALOG
       | PARAMETER_SPECIFIC_NAME
       | PARAMETER_SPECIFIC_SCHEMA
       | PRECISION
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
| RETURNED_CARDINALITY
| RETURNED_LENGTH
| RETURNED_OCTET_LENGTH
| SCALE
| SCOPE_CATALOG
| SCOPE_NAME
| SCOPE_SCHEMA
| TYPE
| UNNAMED
| USER_DEFINED_TYPE_CATALOG
| USER_DEFINED_TYPE_NAME
| USER_DEFINED_TYPE_SCHEMA
```

Syntax Rules

1) The declared type of <item number> shall be exact numeric with scale 0 (zero).

2) For each <get header information>, the declared type of <simple target specification 1> shall be that shown in the Data Type column of the row in Table 2, "Data types of <key word>s used in the header of SQL descriptor areas", whose <key word> column value is equivalent to <header item name>.

3) For each <get item information>, the declared type of <simple target specification 2> shall be that shown in the Data Type column of the row in Table 3, "Data types of <key word>s used in SQL item descriptor areas", whose <key word> column value is equivalent to <descriptor item name>.

Access Rules

None.

General Rules

1) If a <descriptor name> specified in a <get descriptor statement> identifies an SQL descriptor area that is not currently allocated, then an exception condition is raised: *invalid SQL descriptor name*.

2) If the <item number> specified in a <get descriptor statement> is greater than the value of <occurrences> specified when the <descriptor name> was allocated or less than 1 (one), then an exception condition is raised: *dynamic SQL error — invalid descriptor index*.

3) If the <item number> specified in a <get descriptor statement> is greater than the value of COUNT, then a completion condition is raised: *no data*.

4) If the declared type of the <simple target specification> associated with the keyword DATA does not match the data type represented by the item descriptor area, then an exception condition is raised: *data exception — error in assignment*.

   NOTE 10 – "Match" and "represented by" are defined in the Syntax Rules of Subclause 15.1, "Description of SQL descriptor areas".

--`,,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5) Let *i* be the value of the <item number> contained in <get descriptor information>. Let *IDA* be the *i*-th <item descriptor area>. If a <get item information> specifies DATA, then:

   a) If *IDA* is subordinate to an item descriptor area whose TYPE field indicates ARRAY or ARRAY LOCATOR, then an exception condition is raised: *dynamic SQL error — undefined DATA value.*

   b) If the value of TYPE in *IDA* indicates ROW, then an exception condition is raised: *dynamic SQL error — undefined DATA value.*

   c) If the value of INDICATOR is negative and no <get item information> specifies INDICATOR, then an exception condition is raised: *data exception — null value, no indicator parameter.*

6) If an exception condition is raised in a <get descriptor statement>, then the values of all targets specified by <simple target specification 1> and <simple target specification 2> are implementation-dependent.

7) A <get descriptor statement> retrieves values from the SQL descriptor area and item specified by <descriptor name>. For each item, the value that is retrieved is the one established by the most recently executed <allocate descriptor statement>, <set descriptor statement>, or <describe statement> that references the specified SQL descriptor area and item. The value retrieved by a <get descriptor statement> for any field whose value is undefined is implementation-dependent.

   Case:

   a) If <get descriptor information> contains one or more <get header information>s, then for each <get header information> specified, the value of <simple target specification 1> is set to the value *V* in the SQL descriptor area of the field identified by the <header item name> by applying the General Rules of Subclause 9.2, "Store assignment", to <simple target specification 1> and *V* as *TARGET* and *VALUE*, respectively.

   b) If <get descriptor information> contains one or more <get item information>s, then:

      i) Let *i* be the value of the <item number> contained in the <get descriptor information>.

      ii) For each <get item information> specified, the value of <simple target specification 2> is set to the value *V* in the *i*-th SQL item descriptor area of the field identified by the <descriptor item name> by applying the General Rules of Subclause 9.2, "Store assignment", to <simple target specification 2> and *V* as *TARGET* and *VALUE*, respectively.

Conformance Rules

1) Without Feature B032, "Extended dynamic SQL", a <descriptor name> shall be a <literal>.

2) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <get descriptor statement>.

3) Without Feature T301, "Functional dependencies", and without Feature B031, "Basic dynamic SQL", conforming SQL language shall not specify KEY_MEMBER.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.5 <set descriptor statement>

Function

Set information in an SQL descriptor area.

Format

```
<set descriptor statement> ::=
       SET [ SQL ] DESCRIPTOR <descriptor name> <set descriptor information>

<set descriptor information> ::=
         <set header information> [ { <comma> <set header information> }... ]
       | VALUE <item number>
           <set item information> [ { <comma> <set item information> }... ]

<set header information> ::=
       <header item name> <equals operator> <simple value specification 1>

<set item information> ::=
       <descriptor item name> <equals operator> <simple value specification 2>

<simple value specification 1> ::= <simple value specification>

<simple value specification 2> ::= <simple value specification>

<item number> ::= <simple value specification>
```

Syntax Rules

1) The declared type of <item number> shall be exact numeric with scale 0 (zero).

2) For each <set header information>, <header item name> shall not be KEY_TYPE, TOP_LEVEL_COUNT, DYNAMIC_FUNCTION, or DYNAMIC_FUNCTION_CODE, and the declared type of <simple value specification 1> shall be that in the Data Type column of the row of Table 2, "Data types of <key word>s used in the header of SQL descriptor areas", whose <key word> column value is equivalent to <header item name>.

3) For each <set information item>, the value of <descriptor item name> shall not be RETURNED_LENGTH, RETURNED_OCTET_LENGTH, RETURNED_CARDINALITY, OCTET_LENGTH, NULLABLE, KEY_MEMBER, COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME, NAME, UNNAMED, PARAMETER_MODE, PARAMETER_ORDINAL_POSITION, PARAMETER_SPECIFIC_CATALOG, PARAMETER_SPECIFIC_SCHEMA, or PARAMETER_SPECIFIC_NAME. Other alternatives for <descriptor item name> shall not be specified more than once in a <set descriptor statement>. The declared type of <simple value specification 2> shall be that shown in the Data Type column of the row in Table 3, "Data types of <key word>s used in SQL item descriptor areas", whose <key word> column value is equivalent to <descriptor item name>.

4) If the <descriptor item name> specifies DATA, then <simple value specification 2> shall not be a <literal>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Access Rules

None.

General Rules

1) If a <descriptor name> specified in a <set descriptor statement> identifies an SQL descriptor area that is not currently allocated, then an exception condition is raised: *invalid SQL descriptor name*.

2) If the <item number> specified in a <set descriptor statement> is greater than the value of <occurrences> specified when the <descriptor name> was allocated or less than 1 (one), then an exception condition is raised: *dynamic SQL error — invalid descriptor index*.

3) When more than one value is set in a single <set descriptor statement>, the values are effectively assigned in the following order: LEVEL, TYPE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, PRECISION, SCALE, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME, LENGTH, INDICATOR, DEGREE, CARDINALITY, and DATA.

   When any value other than DATA is set, the value of DATA becomes undefined.

4) For every <set item information> specified, let *DIN* be the <descriptor item name>, let *V* be the value of the <simple value specification 2>, let *N* be the value of <item number>, and let *IDA* be the *N*-th item descriptor area.

   a) If *DIN* is DATA, then:

      i) If *IDA* is subordinate to an item descriptor area whose TYPE field indicates ARRAY or ARRAY LOCATOR, then an exception condition is raised: *dynamic SQL error — invalid DATA target*.

      ii) If TYPE in *IDA* indicates ROW, then an exception condition is raised: *dynamic SQL error — invalid DATA target*.

      iii) If the most specific type of *V* does not match the data type specified by the item descriptor area, then an exception condition is raised: *data exception — error in assignment*.

      NOTE 11 – "Match" is defined in the Syntax Rules of Subclause 15.1, "Description of SQL descriptor areas".

      iv) The value of DATA in *IDA* is set to *V*.

   b) If *DIN* is LEVEL, then:

      i) If *N* is 1 (one) and *V* is not 0 (zero), then an exception condition is raised: *dynamic SQL error — invalid LEVEL value*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ii) If *N* is greater than 1 (one), then let *PIDA* be *IDA*'s immediately preceding item descriptor area and let *K* be its LEVEL value.

1) If *V* = *K*+1 and TYPE in *PIDA* does not indicate ROW, ARRAY, or ARRAY LOCATOR, then an exception condition is raised: *dynamic SQL error — invalid LEVEL value*.

2) If *V* > *K*+1, then an exception condition is raised: *dynamic SQL error — invalid LEVEL value*.

3) If *V* < *K*+1, then let $OIDA_i$ be the *i*-th item descriptor area to which *PIDA* is subordinate and whose TYPE field indicates ROW, let NS_i be the number of immediately subordinate descriptor areas of $OIDA_i$ between $OIDA_i$ and *IDA* and let D_i be the value of DEGREE in $OIDA_i$.

A) For each $OIDA_i$ whose LEVEL value is greater than *V*, if D_i is not equal to NS_i, then an exception condition is raised: *dynamic SQL error — invalid LEVEL value*.

B) If *K* is not 0 (zero), then let $OIDA_j$ be the $OIDA_i$ whose LEVEL value is *K*. If there exists no such $OIDA_j$ or D_j is not greater than NS_j, then an exception condition is raised: *dynamic SQL error — invalid LEVEL value*.

iii) The value of LEVEL in *IDA* is set to *V*.

c) If *DIN* is TYPE, then:

i) The value of TYPE in *IDA* is set to *V*.

ii) The value of all fields other than TYPE and LEVEL in *IDA* are set to implementation-dependent values.

iii) Case:

1) If *V* indicates CHARACTER, CHARACTER VARYING, or CHARACTER LARGE OBJECT, then CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA and CHARACTER_SET_NAME in *IDA* are set to the values for the default character set name for the SQL-session and LENGTH in *IDA* is set to 1 (one).

2) If *V* indicates CHARACTER LARGE OBJECT LOCATOR, then LENGTH in *IDA* is set to 1 (one).

3) If *V* indicates BIT or BIT VARYING, then LENGTH in *IDA* is set to 1 (one).

4) If *V* indicates BINARY LARGE OBJECT, then LENGTH in *IDA* is set to 1 (one).

5) If *V* indicates BINARY LARGE OBJECT LOCATOR, then LENGTH in *IDA* is set to 1 (one).

6) If *V* indicates DATETIME, then PRECISION in *IDA* is set to 0 (zero).

7) If *V* indicates INTERVAL, then DATETIME_INTERVAL_PRECISION in *IDA* is set to 2.

8) If *V* indicates NUMERIC or DECIMAL, then SCALE in *IDA* is set to 0 (zero) and PRECISION in *IDA* is set to the implementation-defined default value for the precision of NUMERIC or DECIMAL data types, respectively.

--`,,`--`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

9) If *V* indicates FLOAT, then PRECISION in *IDA* is set to the implementation-defined default value for the precision of the FLOAT data type.

d) If *DIN* is DATETIME_INTERVAL_CODE, then

Case:

i) If TYPE in *IDA* indicates DATETIME, then

Case:

1) If *V* indicates DATE, TIME, or TIME WITH TIME ZONE, then PRECISION in *IDA* is set to 0 (zero) and DATETIME_INTERVAL_CODE in *IDA* is set to *V*.

2) If *V* indicates TIMESTAMP or TIMESTAMP WITH TIME ZONE, then PRECISION in *IDA* is set to 6 and DATETIME_INTERVAL_CODE in *IDA* is set to *V*.

3) Otherwise, an exception condition is raised: *dynamic SQL error — invalid DATETIME_INTERVAL_CODE*.

ii) If TYPE in *IDA* indicates INTERVAL, then

Case:

1) If *V* indicates DAY TO SECOND, HOUR TO SECOND, MINUTE TO SECOND, or SECOND, then PRECISION in *IDA* is set to 6, DATETIME_INTERVAL_PRECISION in *IDA* is set to 2 and DATETIME_INTERVAL_CODE in *IDA* is set to *V*.

2) If *V* indicates YEAR, MONTH, DAY, HOUR, MINUTE, YEAR TO MONTH, DAY TO HOUR, DAY TO MINUTE, or HOUR TO MINUTE, then PRECISION in *IDA* is set to 0 (zero), DATETIME_INTERVAL_PRECISION in *IDA* is set to 2 and DATETIME_INTERVAL_CODE in *IDA* is set to *V*.

3) Otherwise, an exception condition is raised: *dynamic SQL error — invalid DATETIME_INTERVAL_CODE*.

iii) Otherwise, an exception condition is raised: *dynamic SQL error — invalid DATETIME_INTERVAL_CODE*.

e) Otherwise, the value of *DIN* in IDA is set to *V* by applying the General Rules of Subclause 9.2, "Store assignment", to the field of *IDA* identified by *DIN* and *V* as *TARGET* and *VALUE*, respectively. .

5) For each <set header information> specified, the value of the field identified by <header item name> is set to the value *V* of <simple value specification 1> by applying the General Rules of Subclause 9.2, "Store assignment", to the field identified by the <header item name> and *V* as *TARGET* and *VALUE*, respectively.

6) If an exception condition is raised in a <set descriptor statement>, then the values of all elements of the item descriptor area specified in the <set descriptor statement> are implementation-dependent.

7) Restrictions on changing TYPE, LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

SCHEMA, and SCOPE_NAME values resulting from the execution of a <describe statement> before execution of an <execute statement>, <dynamic open statement>, or <dynamic fetch statement> are implementation-defined.

Conformance Rules

1) Without Feature B032, "Extended dynamic SQL", a <descriptor name> shall be a <literal>.

2) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <set descriptor statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.6 <prepare statement>

Function

Prepare a statement for execution.

Format

```
<prepare statement> ::=
       PREPARE <SQL statement name> FROM <SQL statement variable>

<SQL statement variable> ::= <simple value specification>

<preparable statement> ::=
         <preparable SQL data statement>
       | <preparable SQL schema statement>
       | <preparable SQL transaction statement>
       | <preparable SQL control statement>
       | <preparable SQL session statement>
       | <preparable implementation-defined statement>

<preparable SQL data statement> ::=
         <delete statement: searched>
       | <dynamic single row select statement>
       | <insert statement>
       | <dynamic select statement>
       | <update statement: searched>
       | <preparable dynamic delete statement: positioned>
       | <preparable dynamic update statement: positioned>

<preparable SQL schema statement> ::=
         <SQL schema statement>

<preparable SQL transaction statement> ::=
         <SQL transaction statement>

<preparable SQL control statement> ::=
       <SQL control statement>

<preparable SQL session statement> ::=
         <SQL session statement>

<dynamic select statement> ::= <cursor specification>

<preparable implementation-defined statement> ::= !! See the Syntax Rules.
```

Syntax Rules

1) The <simple value specification> of <SQL statement variable> shall not be a <literal>.

2) The declared type of <SQL statement variable> shall be character string.

3) The Format and Syntax Rules for <preparable implementation-defined statement> are implementation-defined.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4) A <preparable SQL control statement> shall not contain an <SQL procedure statement> that is not a <preparable statement>, nor shall it contain a <dynamic single row select statement> or a <dynamic select statement>.

Access Rules

None.

General Rules

1) Let *P* be the contents of the <SQL statement variable>. If *P* is an <SQL control statement>, then let *PS* be an <SQL procedure statement> contained in *P*.

2) Two subfields *SF1* and *SF2* of row types *RT1* and *RT2* are *corresponding subfields* if either *SF1* or *SF2* are positionally corresponding fields of *RT1* and *RT2*, respectively, or *SF1* and *SF2* are positionally corresponding fields of *RT1SF1* and *RT2SF2* and *RT1SF1* and *RT2SF2* are the declared types of corresponding subfields of *RT1* and *RT2* respectively.

3) If *P* does not conform to the Format, Syntax Rules, and Access Rules of a <preparable statement>, or if *P* contains a <simple comment> then an exception condition is raised: *syntax error or access rule violation*.

4) Let *DTGN* be the default transform group name and let *TFL* be the list of {user-defined type name — transform group name} pairs used to identify the group of transform functions for every user-defined type that is referenced in *P*. *DTGN* and *TFL* are not affected by the execution of a <set transform group statement> after *P* is prepared.

5) Let *DPV* be a <value expression> that is either a <dynamic parameter specification> or a <dynamic parameter specification> immediately contained in any number of <left paren> <right paren> pairs. Initially, the declared type of such a <value expression> is, by definition, undefined. A data type is *undefined* if it is neither a data type defined in this standard nor a data type defined by the implementation.

6) Let *MP* be the implementation-defined maximum value of <precision> for the NUMERIC data type. Let *ML* be the implementation-defined maximum value of <length> for the CHARACTER VARYING data type. For each <value expression> *DP* in *P* or *PS* that meets the criteria for *DPV* let *DT* denote its declared type.

   a) Case:

      i) If *DP* is immediately followed by an <interval qualifier> *IQ*, then *DT* is INTERVAL *IQ*.

      ii) If *DP* is the <numeric value expression> simply contained in an <element reference>, then *DT* is NUMERIC (*MP*,0).

      iii) If *DP* is the <string value expression> simply contained in a <char length expression> or an <octet length expression>, then *DT* is CHARACTER VARYING(*ML*) with an implementation-defined character set.

      iv) If *DP* is the <string value expression> simply contained in a <bit length expression>, then *DT* is BIT VARYING(*L*) where *L* is the implementation-defined maximum value of length for BIT VARYING.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

v) If *DP* is either the <numeric value expression dividend> *X1* or the <numeric value expression divisor> *X2* simply contained in a <modulus expression>, then if *DP* is *X1* (*X2*), then *DT* is the declared type of *X2* (*X1*).

vi) If *DP* is either *X1* or *X2* in a <position expression> of the form "POSITION <left paren> *X1* IN *X2* <right paren>", and if *DP* is *X1* (*X2*), then

Case:

1) If the declared type of *X2* (*X1*) is CHARACTER or CHARACTER VARYING with character set *CS*, then *DT* is CHARACTER VARYING (*ML*) with character set *CS*.

2) Otherwise, *DT* is the declared type of *X2* (*X1*).

vii) If *DP* is either *X2* or *X3* in a <string value function> of the form "SUBSTRING <left paren> *X1* FROM *X2* FOR *X3* <right paren>" or "SUBSTRING <left paren> *X1* FROM *X2* <right paren>", then *DT* is NUMERIC (*MP*, 0).

viii) If *DP* is any of *X1*, *X2*, *X3*, or X4 in a <string value function> of the form "OVERLAY <left paren> *X1* PLACING *X2* FROM *X3* FOR X4 <right paren>" or "OVERLAY <left paren> *X1* PLACING *X2* FROM *X3* <right paren>", then

Case:

1) If *DP* is *X1* (*X2*), then

Case:

A) If the declared type of *X2* (*X1*) is CHARACTER or CHARACTER VARYING with character set *CS*, *DT* is CHARACTER VARYING (*ML*) with character set *CS*.

B) Otherwise, *DT* is the declared type of *X2* (*X1*).

2) Otherwise, *DT* is NUMERIC (*MP*, 0).

ix) If *DP* is either *X1* or *X2* in a <value expression> of the form "*X1* <concatenation operator> *X2*" and *DP* is *X1* (*X2*), then

Case:

1) If the declared type of *X2* (*X1*) is CHARACTER or CHARACTER VARYING with character set *CS*, then *DT* is CHARACTER VARYING (*ML*) with character set *CS*.

2) Otherwise, *DT* is the declared type of *X2* (*X1*).

x) If *DP* is either *X1* or *X2* in a <value expression> of the form "*X1* <asterisk> *X2*" or "*X1* <solidus> *X2*" and *DP* is *X1* (*X2*), then

Case:

1) If *DP* is *X1*, then *DT* is the declared type of *X2*.

2) Otherwise,

Case:

A) If the declared type of *X1* is an interval type, then *DT* is NUMERIC (*MP*, 0).

B) Otherwise, *DT* is the declared type of *X2* (*X1*).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

xi) If *DP* is either *X1* or *X2* in a <value expression> of the form "*X1* <plus sign> *X2*" or "*X1* <minus sign> *X2*", then

Case:

1) If *DP* is *X1* in an expression of the form "*X1* <minus sign> *X2*", then *DT* is the declared type of *X2*.

2) Otherwise, if *DP* is *X1* (*X2*), then

Case:

A) If the declared type of *X2* (*X1*) is date, then *DT* is INTERVAL YEAR (*PR*) TO MONTH, where *PR* is the implementation-defined maximum <interval leading field precision>.

B) If the declared type of *X2* (*X1*) is time or timestamp, then *DT* is INTERVAL DAY (*PR*) TO SECOND(*FR*), where *PR* and *FR* are the implementation-defined maximum <interval leading field precision> and maximum <interval fractional seconds precision>, respectively.

C) Otherwise, *DT* is the declared type of *X2* (*X1*).

xii) If *DP* is the <value expression primary> simply contained in a <boolean primary>, then *DT* is BOOLEAN.

xiii) If *DP* is an <array element> simply contained in an <array element list> *AEL* or *DP* represents the value of a subfield *SF* of the declared type of an <array element> simply contained in an <array element list> *AEL*, then let *ET* be the result of applying the Syntax Rules of Subclause 9.3, "Data types of results of aggregations", to the declared types of the <array element>s simply contained in *AEL*.

Case:

1) If *DP* is an <array element> of *AEL*, then *DT* is *ET*.

2) Otherwise, *DT* is the declared type of the subfield of *ET* that corresponds to *SF*.

xiv) If *DP* is the <cast operand> simply contained in a <cast specification> *CS* or *DP* represents the value of a subfield *SF* of the declared type of the <cast operand> simply contained in a <cast specification> *CS*, then let *CT* be the simply contained <cast target> of *CS*.

Case:

1) Let *RT* be a data type determined as follows:

Case:

A) If *CT* immediately contains ARRAY, then *RT* is undefined.

B) If *CT* immediately contains <data type>, then *RT* is that data type.

C) If *CT* simply contains <domain name> *D*, then *RT* is the declared type of the domain identified by *D*.

2) Case:

A) If *DP* is the <cast operand> of *CS*, *DT* is *RT*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B) Otherwise, *DT* is the declared type of the subfield of *RT* that corresponds to *SF*.

xv) If *DP* is a <value expression> simply contained in a <case abbreviation> *CA* or *DP* represents the value of a subfield, SF, of the declared type of such a <value expression>, then let *RT* be the result of applying the Syntax Rules of Subclause 9.3, "Data types of results of aggregations", to the declared types of the <value expression>s simply contained in *CA*.

Case:

1) If *DP* is a <value expression> simply contained in *CA*, then *DT* is *RT*.

2) Otherwise, *DT* is the declared type of the subfield of *RT* that corresponds to *SF*.

xvi) If *DP* is a <result expression> simply contained in a <case specification> *CE* or *DP* represents the value of a subfield *SF* of the declared type of such a <result expression>, then let *RT* be the result of applying the Syntax Rules of Subclause 9.3, "Data types of results of aggregations", to the declared types of the <result expression>s simply contained in *CE*.

Case:

1) If *DP* is a <result expression> simply contained in *CE*, then *DT* is *RT*.

2) Otherwise, *DT* is the declared type of the subfield of *RT* that corresponds to *SF*.

xvii) If *DP* is a <case operand> or <when operand> simply contained in a <simple case> *CE* or *DP* represents the value of a subfield, SF, of the declared type of such a <case operand> or <when operand>, then *RT* is the result of applying the Syntax Rules of Subclause 9.3, to the declared types of the <case operand> and <when operand>s simply contained in *CE*.

Case:

1) If *DP* is a <case operand> or <when operand> simply contained in *CE*, then *DT* is *RT*.

2) Otherwise, *DT* is the declared type of the subfield of *RT* that corresponds to *SF*.

xviii) If *DP* is a <row value expression> or <contextually typed row value expression> simply contained in a <table value constructor> or <contextually typed table value constructor> *TVC*, or if *DP* represents the value of a subfield *SF* of the declared type of such a <row value expression> or <contextually typed row value expression>, then

Case:

1) Let *RT* be a data type determined as follows.

Case:

A) If *TVC* is simply contained in a <query expression> that is simply contained in an <insert statement> *IS* or if *TVC* is immediately contained in the <insert column and source> of an <insert statement> *IS*, then *RT* is a row type in which the declared type of the *i*-th field is the declared type of the *i*-th column in the explicit or implicit <insert column list> of *IS* and the degree of *RT* is equal to the number of columns in the explicit or implicit <insert column list> of *IS*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B) Otherwise, *RT* is the result of applying the Syntax Rules of Subclause 9.3, "Data types of results of aggregations", to the declared types of the <row value expression>s or <contextually typed row value expression>s simply contained in *TVC*.

2) Case:

A) If *DP* is a <row value expression> or <contextually typed row value expression> simply contained in *TVC*, then *DT* is *RT*.

B) Otherwise, *DT* is the declared type of the subfield of *RT* that corresponds to *SF*.

xix) If *DP* is a <row value expression> simply contained in a <comparison predicate>, <distinct predicate> or <between predicate> *PR* or if *DP* represents the value of a subfield, SF, of the declared type of such a <row value expression>, then let *RT* be the result of applying the Syntax Rules of Subclause 9.3, "Data types of results of aggregations", to the declared types of the <row value expression>s simply contained in *PR*.

Case:

1) If *DP* is a <row value expression> simply contained in *PR*, then *DT* is *RT*.

2) Otherwise, *DT* is the declared type of the subfield of *RT* that corresponds to *SF*.

xx) If *DP* is a <row value expression> simply contained in a <quantified comparison predicate> or <match predicate> *PR* or *DP* represents the value of a subfield *SF* of the declared type of such a <row value expression>, then let *RT* be the declared type of the <table subquery> simply contained in *PR*.

Case:

1) If *DP* is a <row value expression> simply contained in *PR*, then *DT* is *RT*.

2) Otherwise, *DT* is the declared type of the subfield of *RT* that corresponds to *SF*.

xxi) If *DP* is a <row value expression> simply contained in an <in predicate> *PR* or if *DP* represents the value of a subfield *SF* of the declared type of such a <row value expression>, then let *RT* be the result of applying the Syntax Rules of Subclause 9.3, "Data types of results of aggregations", to the declared types of the <row value expression>s simply contained in *PR* and the declared row type of the <table subquery> (if any) simply contained in *PR*.

Case:

1) If *DP* is a <row value expression> simply contained in *PR*, then *DT* is *RT*.

2) Otherwise, *DT* is the declared type of the subfield of *RT* that corresponds to *SF*.

xxii) If *DP* is the first <row value constructor element> simply contained in either <row value expression 1> *RV1* or <row value expression 2> *RV2* in an <overlaps predicate> *PR*, then

Case:

1) If both *RV1* and *RV2* simply contain a <row value constructor> whose first <row value constructor element> meets the criteria for *DPV*, then *DT* is TIMESTAMP WITH TIME ZONE.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) Otherwise, if *DP* is simply contained in *RV1* (*RV2*), then *DT* is the declared type of the first <row value constructor element> of *RV2* (*RV1*).

xxiii) If *DP* is simply contained in a <like predicate> or a <similar predicate> *PR*, then let *X1* represent the <character match value> or the <octet match value>, let *X2* represent the <character pattern>, the <octet pattern> or the <similar pattern>, and let *X3* represent the <escape character> or the <escape octet>.

Case:

1) If all *X1*, *X2* and *X3* meet the criteria for *DPV*, then *DT* is CHARACTER VARYING (*ML*) with an implementation-defined character set.

2) Otherwise, let *RT* be the result of applying the Syntax Rules of Subclause 9.3, "Data types of results of aggregations", to the declared types of *X1*, *X2* and *X3*.

Case:

A) If *RT* is CHARACTER or CHARACTER VARYING with character set *CS*, then *DT* is CHARACTER VARYING(*ML*) with character set *CS*.

B) Otherwise, *DT* is *RT*.

xxiv) If *DP* is the <value expression> simply contained in an <update source> of a <set clause> *SC* or if *DP* represents the value of a subfield *SF* of the declared type of such a <value expression>, then let *RT* be the declared type of the <update target> or <mutated set clause> specified in *SC*.

Case:

1) If *DP* is the <value expression> simply contained in *SC*, then *DT* is *RT*.

2) Otherwise, *DT* is the declared type of the subfield of *RT* that corresponds to *SF*.

xxv) If *DP* is the <value specification> immediately contained in a <catalog name characteristic>, <schema name characteristic>, <character set name characteristic>, <SQL-path characteristic>, <transform group characteristic>, <role specification> or <set session authorization identifier statement>, then *DT* is CHARACTER VARYING (*ML*) with an implementation-defined character set.

xxvi) If *DP* is the <interval value expression> immediately contained in a <set local time zone statement>, then *DT* is INTERVAL HOUR TO MINUTE.

xxvii) If *DP* is an <SQL argument> of a <routine invocation> *RI* immediately contained in a <call statement> or if *DP* is the value of a subfield *SF* of the declared type of a <value expression> immediately contained in such an <SQL argument>, and if *DP* is the *i*-th <SQL argument> of *RI* or is contained in the *i*-th <SQL argument> of *RI*, then let *RT* denote the declared type of the *i*-th SQL parameter of the subject routine of *RI* determined by applying the Syntax Rules of Subclause 8.1, "<routine invocation>", to *RI*.

Case:

1) If *DP* is the *i*-th <SQL argument> of *RI*, then *DT* is *RT*.

2) Otherwise, *DT* is the declared type of the subfield of *RT* that corresponds to *SF*.

b) If *DT* is undefined, then an exception condition is raised: *syntax error or access rule violation*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7) Whether a <dynamic parameter specification> is an input argument, an output argument, or both an input and an output argument is determined as follows.

Case:

a) If *P* is a <call statement>, then:

i) Let *SR* be the subject routine of the <routine invocation> *RI* immediately contained in *P*. Let *n* be the number of <SQL argument>s in the <SQL argument list> immediately contained in *RI*.

ii) Let A_y, 1 (one) $\leq y \leq n$, be the *y*-th <SQL argument> of the <SQL argument list> immediately contained in *RI*.

iii) For each <dynamic parameter specification> *D* contained in some <SQL argument> A_k, 1 (one) $\leq k \leq n$:

1) *D* is an input <dynamic parameter specification> if the <SQL parameter mode> of the *k*-th SQL parameter of *SR* is INPUT or INOUT.

2) *D* is an output <dynamic parameter specification> if the <SQL parameter mode> of the *k*-th SQL parameter of *SR* is OUTPUT or INOUT.

b) Otherwise:

i) If a <dynamic parameter specification> is contained in a <target specification>, then it is an output <dynamic parameter specification>.

ii) If a <dynamic parameter specification> is contained in a <value specification>, then it is an input <dynamic parameter specification>.

8) If *P* or *PS* is a <preparable dynamic delete statement: positioned> or a <preparable dynamic update statement: positioned>, then let *CN* be the <cursor name> contained in *P* or *PS*, respectively.

Case:

a) If *P* or *PS* contains a <scope option> that specifies GLOBAL, then

Case:

i) If there exists an extended dynamic cursor *EDC* with an <extended cursor name> having a global scope and a <cursor name> that is equivalent to *CN*, then *EDC* is the cursor referenced by *P* or *PS*.

ii) Otherwise, an exception condition is raised: *invalid cursor name*.

b) If *P* or *PS* contains a <scope option> that specifies LOCAL, or if no <scope option> is specified, then the *potentially referenced cursors* of *P* or *PS* include every declared dynamic cursor whose <cursor name> is equivalent to *CN* and whose scope is the containing module and every extended dynamic cursor having an <extended cursor name> that has a scope of the containing module and whose <cursor name> is equivalent to *CN*.

Case:

i) If the number of potentially referenced cursors is greater than 1 (one), then an exception condition is raised: *ambiguous cursor name*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ii) If the number of potentially referenced cursors is less than 1 (one), then an exception condition is raised: *invalid cursor name*.

iii) Otherwise, *CN* refers to the single potentially referenced cursor of *P*.

9) If <extended statement name> is specified for the <SQL statement name>, then let *S* be <simple target specification> and let *V* be the character string that is the result of

```
TRIM ( BOTH ' ' FROM S )
```

If *V* does not conform to the Format and Syntax Rules of an <identifier>, then an exception condition is raised: *invalid SQL statement identifier*.

10) If <statement name> is specified for the <SQL statement name>, *P* is not a <cursor specification>, and <statement name> is associated with a cursor *C* through a <dynamic declare cursor>, then an exception condition is raised: *dynamic SQL error — prepared statement not a cursor specification*.

11) If the value of the <SQL statement name> identifies an existing prepared statement, then an implicit

```
DEALLOCATE PREPARE SSN
```

is executed, where *SSN* is the value of the <SQL statement name>.

12) *P* is prepared for execution.

13) Case:

a) If <extended statement name> is specified for the <SQL statement name>, then the value of the <extended statement name> is associated with the prepared statement. This value and explicit or implied <scope option> shall be specified for each <execute statement> or <allocate cursor statement> that is to be associated with this prepared statement.

b) If <statement name> is specified for the <SQL statement name>, then:

i) If <statement name> is not associated with a cursor and either *P* is not a <cursor specification> or *P* is a <cursor specification> that conforms to the Format and Syntax Rules of a <dynamic single row select statement>, then an equivalent <statement name> shall be specified for each <execute statement> that is to be associated with this prepared statement.

ii) If *P* is a <cursor specification> and <statement name> is associated with a cursor *C* through a <dynamic declare cursor>, then an association is made between *C* and *P*. The association is preserved until the prepared statement is destroyed.

14) The validity of an <extended statement name> value or a <statement name> that does not identify a held cursor in an SQL-transaction different from the one in which the statement was prepared is implementation-dependent.

Conformance Rules

1) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <prepare statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.7 <deallocate prepared statement>

Function

Deallocate SQL-statements that have been prepared with a <prepare statement>.

Format

```
<deallocate prepared statement> ::=
       DEALLOCATE PREPARE <SQL statement name>
```

Syntax Rules

1) If <SQL statement name> is a <statement name>, then the containing <SQL-client module definition> shall contain a <prepare statement> whose <statement name> is equivalent to the <statement name> of the <deallocate prepared statement>.

Access Rules

None.

General Rules

1) If the <SQL statement name> does not identify a statement prepared in the scope of the <SQL statement name>, then an exception condition is raised: *invalid SQL statement name.*

2) If the value of <SQL statement name> identifies an existing prepared statement that is the <cursor specification> of an open cursor, then an exception condition is raised: *invalid cursor state.*

3) The prepared statement identified by the <SQL statement name> is destroyed. Any cursor that was allocated with an <allocate cursor statement> that is associated with the prepared statement identified by the <SQL statement name> is destroyed. If the value of the <SQL statement name> identifies an existing prepared statement that is a <cursor specification>, then any prepared statements that reference that cursor are destroyed.

Conformance Rules

1) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall contain no <deallocate prepared statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.8 <describe statement>

Function

Obtain information about the <select list> columns or <dynamic parameter specification>s contained in a prepared statement or about the columns of the result set associated with a cursor.

Format

```
<describe statement> ::=
        <describe input statement>
      | <describe output statement>

<describe input statement> ::=
      DESCRIBE INPUT <SQL statement name> <using descriptor>
          [ <nesting option> ]

<describe output statement> ::=
      DESCRIBE [ OUTPUT ] <described object> <using descriptor>
          [ <nesting option> ]

<nesting option> ::=
        WITH NESTING
      | WITHOUT NESTING

<using descriptor> ::=
      USING [ SQL ] DESCRIPTOR <descriptor name>

<described object> ::=
        <SQL statement name>
      | CURSOR <extended cursor name> STRUCTURE
```

Syntax Rules

1) If <SQL statement name> is a <statement name>, then the containing <SQL-client module definition> shall contain a <prepare statement> whose <statement name> is equivalent to the <statement name> of the <describe statement>.

2) If <nesting option> is not specified, then WITHOUT NESTING is implicit.

Access Rules

None.

General Rules

1) If <describe input statement> is executed and the value of the <SQL statement name> does not identify a statement prepared in the scope of the <SQL statement name>, then an exception condition is *invalid SQL statement name*.

2) If <describe output statement> is executed, <SQL statement name> is specified, and the value of the <SQL statement name> does not identify a statement prepared in the scope of the <SQL statement name>, then an exception condition is *invalid SQL statement name*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3) If <describe output statement> is executed, <extended cursor name> is specified, and the value of the <extended cursor name> does not identify a known allocated cursor, then an exception condition is *invalid cursor name*.

4) An SQL system descriptor area shall have been allocated and not yet deallocated whose name is the value of the <descriptor name>'s <simple value specification> and whose scope is that specified by the <scope option>. Otherwise, an exception condition is raised: *invalid SQL descriptor name*.

5) Let *DA* be the descriptor area identified by <descriptor name>. Let *N* be the <occurrences> specified when *DA* was allocated.

6) Case:

   a) If the statement being executed is a <describe input statement>, then a descriptor for the input <dynamic parameter specification>s for the prepared statement is stored in *DA*. Let *D* be the number of input <dynamic parameter specification>s in the prepared statement. If WITH NESTING is specified, then let NS_i, 1 (one) $\leq i \leq D$, be the number of subordinate descriptors of the descriptor for the i-th input dynamic parameter; otherwise, let NS_i be 0 (zero).

   b) If the statement being executed is a <describe output statement> and the prepared statement that is being described is a <dynamic select statement> or a <dynamic single row select statement>, then a descriptor for the <select list> columns for the prepared statement is stored in *DA*. Let *T* be the table defined by the prepared statement and let *D* be the degree of *T*. If WITH NESTING is specified, then let NS_i, 1 (one) $\leq i \leq D$, be the number of subordinate descriptors of the descriptor for the i-th column of *T*; otherwise, let NS_i be 0 (zero).

   c) Otherwise, a descriptor for the output <dynamic parameter specification>s for the prepared statement is stored in *DA*. Let *D* be the number of output <dynamic parameter specification>s in the prepared statement. If WITH NESTING is specified, then let NS_i, 1 (one) $\leq i \leq D$, be the number of subordinate descriptors of the descriptor for the i-th output dynamic parameter; otherwise, let NS_i be 0 (zero).

7) *DA* is set as follows:

   a) Let *TD* be the value of $D+NS_1+NS_2+ \ldots +NS_D$. COUNT is set to *TD*.

   b) TOP_LEVEL_COUNT is set to *D*.

   c) DYNAMIC_FUNCTION and DYNAMIC_FUNCTION_CODE are set to the identifier and code, respectively, for the prepared statement as shown in Table 9, "SQL-statement codes".

   d) If the statement being executed is a <describe output statement> and the prepared statement that is being described is a <dynamic select statement> or a <dynamic single row select statement>:

      Case:

      i) If some subset of the columns of *T* is the primary key of *T*, then KEY_TYPE is set to 1 (one).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

ii) If some subset of the columns of *T* is the preferred candidate key of *T*, then KEY_TYPE is set to 2.

NOTE 12 – Primary keys and preferred candidate keys are defined in Subclause 4.18, "Functional dependencies", ISO/IEC 9075-2.

e) If *TD* is greater than *N*, then a completion condition is raised: *warning — insufficient item descriptor areas*.

f) If *TD* is 0 (zero) or *TD* is greater than *N*, then no item descriptor areas are set. Otherwise:

i) The first *TD* item descriptor areas are set with values from the descriptors and, optionally, subordinate descriptors for

Case:

1) If the statement being executed is a <describe input statement>, then the input <dynamic parameter specification>s.

2) If the statement being executed is a <describe output statement> and the statement being described is a <dynamic select statement> or a <dynamic single row select statement>, then the columns of *T*.

3) Otherwise, the output <dynamic parameter specification>s.

ii) The descriptor for the first such column or <dynamic parameter specification> is assigned to the first item descriptor area.

iii) If the descriptor for the *j*-th column or <dynamic parameter specification> is assigned to the *k*-th item descriptor area, then:

1) The descriptor for the (*j*+1)-th column or <dynamic parameter specification> is assigned to the ($k+NS_{j+1}$)-th item descriptor area.

2) If WITH NESTING is specified, then the implicitly ordered subordinate descriptors for the *j*-th column or <dynamic parameter specification> are assigned to contiguous item descriptor areas starting at the (*k*+1)-th item descriptor area.

8) An SQL item descriptor area, if set, consists of values for LEVEL, TYPE, NULLABLE, NAME, UNNAMED, PARAMETER_ORDINAL_POSITION, PARAMETER_SPECIFIC_CATALOG, PARAMETER_SPECIFIC_SCHEMA, PARAMETER_SPECIFIC_NAME, and other fields depending on the value of TYPE as described below. The DATA and INDICATOR fields are not relevant. Those fields and fields that are not applicable for a particular value of TYPE are set to implementation-dependent values.

a) If the SQL item descriptor area is set to a descriptor that is immediately subordinate to another whose LEVEL value is *K*, then LEVEL is set to *K*+1; otherwise, LEVEL is set to 0 (zero).

b) TYPE is set to a code, as shown in Table 4, "Codes used for SQL data types in Dynamic SQL", indicating the declared type of the column, <dynamic parameter specification>, or subordinate descriptor.

--`,,`--`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c) Case:

i) If the value of LEVEL is 0 (zero) and the item descriptor area describes a column, then:

1) If the column is possibly nullable, then NULLABLE is set to 1 (one); otherwise, NULLABLE is set to 0 (zero).

2) If the column name is implementation-dependent, then NAME is set to the implementation-dependent name of the column and UNNAMED is set to 1 (one); otherwise, NAME is set to the <derived column> name for the column and UNNAMED is set to 0 (zero).

3) If the column is a member of the primary key of *T* and KEY_TYPE was set to 1 (one) or if the column is a member of the preferred candidate key of *T* and KEY_TYPE was set to 2, then KEY_MEMBER is set to 1 (one); otherwise, KEY_MEMBER is set to 0 (zero).

ii) If the value of LEVEL is 0 (zero) and the item descriptor area describes a <dynamic parameter specification>, then:

1) NULLABLE is set to 1 (one).

NOTE 13 – This indicates that the <dynamic parameter specification> can have the null value.

2) UNNAMED is set to 1 (one) and NAME is set to an implementation-dependent value.

3) KEY_MEMBER is set to 0 (zero).

iii) Otherwise:

1) NULLABLE is set to 1 (one).

2) Case:

A) If the item descriptor area describes a field of a row, then

Case:

I) If the name of the field is implementation-dependent, then NAME is set to the implementation-dependent name of the field and UNNAMED is set to 1 (one).

II) Otherwise, NAME is set to the name of the field and UNNAMED is set to 0 (zero).

B) Otherwise, UNNAMED is set to 1 (one) and NAME is set to an implementation-defined value.

3) KEY_MEMBER is set to 0 (zero).

d) Case:

i) If TYPE indicates a <character string type>, then LENGTH is set to the length or maximum length in characters of the character string; OCTET_LENGTH is set to the maximum possible length in octets of the character string; CHARACTER_SET_

--`,,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

CATALOG, CHARACTER_SET_SCHEMA and CHARACTER_SET_NAME are set to the <character set name> of the character string's character set; and COLLATION_CATALOG, COLLATION_SCHEMA and COLLATION_NAME are set to the <collation name> of the character string's collation. If the subject <language clause> specifies C, then the lengths specified in LENGTH and OCTET_LENGTH do not include the implementation-defined null character that terminates a C character string.

ii) If TYPE indicates a <bit string type>, then LENGTH is set to the length or maximum length in bits of the bit string and OCTET_LENGTH is set to the maximum possible length in octets of the bit string.

iii) If TYPE indicates a <binary large object string type>, then LENGTH is set to the length or maximum length in octets of the binary string and OCTET_LENGTH is set to the maximum possible length in octets of the binary string.

iv) If TYPE indicates an <exact numeric type>, then PRECISION and SCALE are set to the precision and scale of the exact numeric.

v) If TYPE indicates an <approximate numeric type>, then PRECISION is set to the precision of the approximate numeric.

vi) If TYPE indicates a <datetime type>, then LENGTH is set to the length in positions of the datetime type, DATETIME_INTERVAL_CODE is set to a code as specified in Table 5, "Codes associated with datetime data types in Dynamic SQL", to indicate the specific datetime data type and PRECISION is set to the <time precision> or <timestamp precision>, if either is applicable.

vii) If TYPE indicates an <interval type>, then LENGTH is set to the length in positions of the interval type, DATETIME_INTERVAL_CODE is set to a code as specified in Table 6, "Codes used for <interval qualifier>s in Dynamic SQL", to indicate the <interval qualifier> of the interval data type, DATETIME_INTERVAL_PRECISION is set to the <interval leading field precision> and PRECISION is set to the <interval fractional seconds precision>, if applicable.

viii) If TYPE indicates a user-defined type, then USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are set to the <user-defined type name> of the user-defined type's name.

ix) If TYPE indicates a <reference type>, then USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME are set to the <user-defined type name> of the referenced type and qualified name of the referenceable base table; LENGTH and OCTET_LENGTH are set to the implementation-defined length in octets of a <reference type>.

x) If TYPE indicates ROW, then DEGREE is set to the degree of the row type.

xi) If TYPE indicates ARRAY, then CARDINALITY is set to the cardinality of the array type.

e) If LEVEL is 0 (zero) and the prepared statement is a <call statement>, then:

i) Let *SR* be the subject routine for the <routine invocation> of the <call statement>.

ii) Let D_x be the x-th <dynamic parameter specification> simply contained in an SQL argument A_y of the <call statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

iii) Let P_y be the y-th SQL parameter of SR.

NOTE 14 – A P whose <SQL parameter mode> is IN can be a <value expression> that contains zero, one, or more <dynamic parameter specification>s. Thus:

— Every D_x maps to one and only one P_y.

— Several D_x instances can map to the same P_y.

— There can be P_y instances that have no D_x instances that map to them.

iv) The PARAMETER_MODE value in the descriptor for each D_x is set to the value from Table 7, "Codes used for input/output SQL parameter modes in Dynamic SQL", that indicates the <SQL parameter mode> of P_y.

v) The PARAMETER_ORDINAL_POSITION value in the descriptor for each D_x is set to the ordinal position of P_y.

vi) The PARAMETER_SPECIFIC_CATALOG, PARAMETER_SPECIFIC_SCHEMA, and PARAMETER_SPECIFIC_NAME values in the descriptor for each D_x are set to the values that identify the catalog, schema, and specific name of SR.

Conformance Rules

1) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall not contain any <describe input statement>.

2) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <describe output statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

15.9 <input using clause>

Function

Supply input values for an <SQL dynamic statement>.

Format

```
<input using clause> ::=
        <using arguments>
      | <using input descriptor>

<using arguments> ::=
      USING <using argument> [ { <comma> <using argument> }... ]

<using argument> ::= <general value specification>

<using input descriptor> ::= <using descriptor>
```

Syntax Rules

1) The <general value specification> immediately contained in <using argument> shall be either a <host parameter specification> or an <embedded variable specification>.

Access Rules

None.

General Rules

1) If <using input descriptor> is specified, then an SQL system descriptor area shall have been allocated and not yet deallocated whose name is the value of the <descriptor name>'s <simple value specification> and whose scope is that specified by the <scope option>. Otherwise, an exception condition is raised: *invalid SQL descriptor name*.

2) When an <input using clause> is used in a <dynamic open statement> or as the <parameter using clause> in an <execute statement>, the <input using clause> describes the input <dynamic parameter specification> values for the <dynamic open statement> or the <execute statement>, respectively. Let *PS* be the prepared <dynamic select statement> referenced by the <dynamic open statement> or the prepared statement referenced by the <execute statement>, respectively.

3) Let *D* be the number of input <dynamic parameter specification>s in *PS*.

4) If <using arguments> is specified and the number of <using argument>s is not *D*, then an exception condition is raised: *dynamic SQL error — using clause does not match dynamic parameter specifications*.

5) If <using input descriptor> is specified, then:

   a) Let *N* be the value of COUNT.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) If *N* is greater than the value of <occurrences> specified when the SQL descriptor area identified by <descriptor name> was allocated or is less than zero, then an exception condition is raised: *dynamic SQL error — invalid descriptor count.*

c) If the first *N* item descriptor areas are not valid as specified in Subclause 15.1, "Description of SQL descriptor areas", then an exception condition is raised: *dynamic SQL error — using clause does not match dynamic parameter specifications.*

d) In the first *N* item descriptor areas:

i) If the number of item descriptor areas in which the value of LEVEL is 0 (zero) is not *D*, then an exception condition is raised: *dynamic SQL error — using clause does not match dynamic parameter specifications.*

ii) If the value of INDICATOR is not negative, TYPE does not indicate ROW, and the item descriptor area is not subordinate to an item descriptor area whose INDICATOR value is negative or whose TYPE field indicates ARRAY or ARRAY LOCATOR, and if the value of DATA is not a valid value of the data type represented by the item descriptor area, then an exception condition is raised: *dynamic SQL error — using clause does not match dynamic parameter specifications.*

6) For 1 (one) $\leq i \leq D$:

a) Let *TDT* be the effective declared type of the *i*-th input <dynamic parameter specification>, defined to be the type represented by the item descriptor area and its subordinate descriptor areas that would be set by a <describe input statement> to reflect the description of the *i*-th input <dynamic parameter specification> of *PS*.

NOTE 15 – See the General Rules of Subclause 15.8, "<describe statement>".

NOTE 16 – "Represented by", as applied to the relationship between a data type and an item descriptor area, is defined in the Syntax Rules of Subclause 15.1, "Description of SQL descriptor areas".

b) Case:

i) If <using input descriptor> is specified, then:

1) Let *IDA* be the *i*-th item descriptor area whose LEVEL value is 0 (zero).

2) Let *SDT* be the effective declared type represented by *IDA*.

NOTE 17 – "Represented by", as applied to the relationship between a data type and an item descriptor area, is defined in the Syntax Rules of Subclause 15.1, "Description of SQL descriptor areas".

3) Let *SV* be the *associated value* of *IDA*, which is defined to be

Case:

A) If the value of INDICATOR is negative, then *SV* is the null value.

B) Otherwise,

Case:

I) If TYPE indicates ROW, then *SV* is a row whose type is *SDT* and whose field values are the associated values of the immediately subordinate descriptor areas of *IDA*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

II) Otherwise, *SV* is the value of DATA with data type *SDT*.

ii) If <using arguments> is specified, then let *SDT* and *SV* be the declared type and value, respectively, of the *i*-th <using argument>.

c) Case:

i) If *SDT* is a locator type, then:

1) If *SV* is not the null value, then let the value of the *i*-th dynamic parameter be the value of *SV*.

2) Otherwise, let the value of the *i*-th dynamic parameter be the null value.

ii) If *SDT* and *TDT* are predefined data types, then

Case:

1) If the <cast specification>

```
CAST ( IV AS TDT )
```

does not conform to the Syntax Rules of Subclause 6.22, "<cast specification>", in ISO/IEC 9075-2, and there is an implementation-defined conversion from type *STD* to type *TDT*, then that implementation-defined conversion is effectively performed, converting *IV* to type *TDT*, and the result is the value *TV* of the *i*-th input dynamic parameter.

2) Otherwise:

A) If the <cast specification>

```
CAST ( IV AS TDT )
```

does not conform to the Syntax Rules of Subclause 6.22, "<cast specification>", in ISO/IEC 9075-2, then an exception condition is raised: *dynamic SQL error — restricted data type attribute violation*.

B) If the <cast specification>

```
CAST ( IV AS TDT )
```

does not conform to the General Rules of Subclause 6.22, "<cast specification>", in ISO/IEC 9075-2, then an exception condition is raised in accordance with the General Rules of Subclause 6.22, "<cast specification>", in ISO/IEC 9075-2.

C) The <cast specification>

```
CAST ( IV AS TDT )
```

is effectively performed and is the value of the *i*-th input dynamic parameter.

iii) If *SDT* is a predefined data type and *TDT* is a user-defined type, then:

1) Let *DT* be the data type identified by *TDT*.

2) If the current SQL-session has a group name corresponding to the user-defined name of *DT*, then let *GN* be that group name; Otherwise, let *GN* be the default transform group name associated with the current SQL-session.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3) The Syntax Rules of Subclause 10.17, "Determination of a to-sql function", in ISO/IEC 9075-2, are applied with *DT* and *GN* as *TYPE* and *GROUP*, respectively.

Case:

A) If there is an applicable to-sql function, then let *TSF* be that to-sql function. If *TSF* is an SQL-invoked method, then let *TSFPT* be the declared type of the second SQL parameter of *TSF*; otherwise, let *TSFPT* be the declared type of the first SQL parameter of *TSF*.

Case:

I) If *TSFPT* is compatible with *SDT*, then

Case:

1) If *TSF* is an SQL-invoked method, then *TSF* is effectively invoked with the value returned by the function invocation:

```
DT()
```

as the first parameter and *SV* as the second parameter. The <return value> is the value of the *i*-th input dynamic parameter.

2) Otherwise, *TSF* is effectively invoked with *SV* as the first parameter. The <return value> is the value of the *i*-th input dynamic parameter.

II) Otherwise, an exception condition is raised: *dynamic SQL error — restricted data type attribute violation*.

B) Otherwise, an exception condition is raised: *dynamic SQL error — data type transform function violation*.

Conformance Rules

1) Without Feature B032, "Extended dynamic SQL", a <descriptor name> shall be a <literal>.

2) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <using input clause>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.10 <output using clause>

Function

Supply output variables for an <SQL dynamic statement>.

Format

```
<output using clause> ::=
         <into arguments>
       | <into descriptor>

<into arguments> ::=
       INTO <into argument> [ { <comma> <into argument> }... ]

<into argument> ::= <target specification>

<into descriptor> ::=
       INTO [ SQL ] DESCRIPTOR <descriptor name>
```

Syntax Rules

None.

Access Rules

None.

General Rules

1) If <into descriptor> is specified, then an SQL system descriptor area shall have been allocated and not yet deallocated whose name is the value of the <descriptor name>'s <simple value specification> and whose scope is that specified by the <scope option>. Otherwise, an exception condition is raised: *invalid SQL descriptor name*.

2) When an <output using clause> is used in a <dynamic fetch statement> or as the <result using clause> of an <execute statement>, let *PS* be the prepared <dynamic select statement> referenced by the <dynamic fetch statement> or the prepared <dynamic single row select statement> referenced by the <execute statement>, respectively.

3) Case:

   a) If *PS* is a <prepared dynamic select statement> or a <dynamic single row select statement>, then the <output using clause> describes the <target specification>s for the <dynamic fetch statement> or the <execute statement>. Let *D* be the degree of the table specified by *PS*.

   b) Otherwise, the <output using clause> describes the <target specification>s for the output <dynamic parameter specification>s contained in *PS*. Let *D* be the number of such output <dynamic parameter specification>s.

4) If <into arguments> is specified and the number of <into argument>s is not *D*, then an exception condition is raised: *dynamic SQL error — using clause does not match target specifications*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5) If <into descriptor> is specified, then:

   a) Let *N* be the value of COUNT.

   b) If *N* is greater than the value of <occurrences> specified when the SQL descriptor area identified by <descriptor name> was allocated or less than zero, then an exception condition is raised: *dynamic SQL error — invalid descriptor count.*

   c) If the first *N* item descriptor areas are not valid as specified in Subclause 15.1, "Description of SQL descriptor areas", then an exception condition is raised: *dynamic SQL error — using clause does not match target specifications.*

   d) In the first *N* item descriptor areas, if the number of item descriptor areas in which the value of LEVEL is 0 (zero) is not *D*, then an exception condition is raised: *dynamic SQL error — using clause does not match target specifications.*

6) For 1 (one) ≤ *i* ≤ *D*:

   a) Let *SDT* be the effective declared type of the *i*-th <select list> column or output <dynamic parameter specification>, defined to be the type represented by the item descriptor area and its subordinate descriptor areas that would be set by

      Case:

      i) If *PS* is a <prepared dynamic select statement> or a <dynamic single row select statement>, then a <describe output statement> to reflect the description of the *i*-th <select list> column; let *SV* be the value of that <select list> column, with data type *SDT*.

      ii) Otherwise, a <describe output statement> to reflect the description of the *i*-th output <dynamic parameter specification>; let *SV* be the value of that <dynamic parameter specification>, with data type *SDT*.

      NOTE 18 – "Represented by", as applied to the relationship between a data type and an item descriptor area, is defined in the Syntax Rules of Subclause 15.1, "Description of SQL descriptor areas".

   b) Case:

      i) If <into descriptor> is specified, then let *TDT* be the declared type of the *i*-th <target specification> as represented by the *i*-th item descriptor area *IDA* whose LEVEL value is 0 (zero).

      NOTE 19 – "Represented by", as applied to the relationship between a data type and an item descriptor area, is defined in the Syntax Rules of Subclause 15.1, "Description of SQL descriptor areas".

      ii) If <into arguments> is specified, then let *TDT* be the data type of the *i*-th <into argument>.

   c) If the <output using clause> is used in a <dynamic fetch statement>, then let *LTDT* be the data type on the most recently executed <dynamic fetch statement>, if any, for the cursor *CR*. It is implementation-defined whether or not an exception condition is raised: *dynamic SQL error — restricted data type attribute violation* if any of the following are true:

      i) *LTDT* and *TDT* both identify a binary large object type and only one of *LTDT* and *TDT* is a binary large object locator.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ii) *LTDT* and *TDT* both identify a character large object type and only one of *LTDT* and *TDT* is a character large object locator.

iii) *LTDT* and *TDT* both identify an array type and only one of *LTDT* and *TDT* is an array locator.

iv) *LTDT* and *TDT* both identify a user-defined type and only one of *LTDT* and *TDT* is a user-defined type locator.

d) Case:

i) If *TDT* is a locator type, then:

1) If *SV* is not the null value, then a locator *L* that uniquely identifies *SV* is generated and is the value *TV* of the *i*-th <target specification>.

2) Otherwise, the value *TV* of the *i*-th <target specification> is the null value.

ii) If *STD* and *TDT* are predefined data types, then

Case:

1) If the <cast specification>

```
CAST ( SV AS TDT )
```

does not conform to the Syntax Rules of Subclause 6.22, "<cast specification>", in ISO/IEC 9075-2, and there is an implementation-defined conversion of type *STD* to type *TDT*, then that implementation-defined conversion is effectively performed, converting *SV* to type *TDT*, and the result is the value *TV* of the *i*-th <target specification>.

2) Otherwise:

A) If the <cast specification>

```
CAST ( SV AS TDT )
```

does not conform to the Syntax Rules of Subclause 6.22, "<cast specification>", in ISO/IEC 9075-2, then an exception condition is raised: *dynamic SQL error — restricted data type attribute violation*.

B) If the <cast specification>

```
CAST ( SV AS TDT )
```

does not conform to the General Rules of Subclause 6.22, "<cast specification>", in ISO/IEC 9075-2, then an exception condition is raised in accordance with the General Rules of Subclause 6.22, "<cast specification>", in ISO/IEC 9075-2.

C) The <cast specification>

```
CAST ( SV AS TDT )
```

is effectively performed, and is the value *TV* of the *i*-th <target specification>.

iii) If *SDT* is a user-defined type and *TDT* is a predefined data type, then:

1) Let *DT* be the data type identified by *SDT*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) If the current SQL-session has a group name corresponding to the user-defined type name of *DT*, then let *GN* be that group name; otherwise, let *GN* be the default transform group name associated with the current SQL-session.

3) Apply the Syntax Rules of Subclause 10.15, "Determination of a from-sql function", in ISO/IEC 9075-2, with *DT* and *GN* as *TYPE* and *GROUP*, respectively.

Case:

A) If there is an applicable from-sql function, then let *FSF* be that from-sql function and let *FSFRT* be the <returns data type> of *FSF*.

Case:

I) If *FSFRT* is compatible with *TDT*, then the from-sql function *FSF* is effectively invoked with *SV* as its input parameter and the <return value> is the value *TV* of the *i*-th <target specification>.

II) Otherwise, an exception condition is raised: *dynamic SQL error — restricted data type attribute violation*.

B) Otherwise, an exception condition is raised: *dynamic SQL error — data type transform function violation*.

e) Case:

i) If <into descriptor> is specified, then *IDA* is set to reflect the value of *TV* as follows:

Case:

1) If TYPE indicates ROW, then

Case:

A) If *TV* is the null value, then the value of INDICATOR in *IDA* and in all subordinate descriptor areas of *IDA* that are not subordinate to an item descriptor area whose TYPE indicates ARRAY or ARRAY LOCATOR is set to −1.

B) Otherwise, the *i*-th subordinate descriptor area of *IDA* is set to reflect the value of the *i*-th field of *TV* by applying this subrule (beginning with the outermost 'Case') to the *i*-th subordinate descriptor area of *IDA* as *IDA*, the value of the *i*-th field of *TV* as *TV*, the value of the *i*-th field of *SV* as *SV*, and the data type of the *i*-th field of *SV* as *SDT*.

2) Otherwise,

Case:

A) If *TV* is the null value, then the value of INDICATOR is set to −1.

B) If *TV* is not the null value, then:

I) The value of INDICATOR is set to 0 (zero).

II) Case:

1) If TYPE indicates a locator type, then a locator *L* that uniquely identifies *TV* is generated and the value of DATA is set to an implementation-dependent four-octet value that represents *L*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) Otherwise, the value of DATA is set to *TV*.

III) Case:

1) If TYPE indicates CHARACTER VARYING, BIT VARYING, or BINARY LARGE OBJECT, then RETURNED_LENGTH is set to the length in characters, bits, or octets, respectively, of *TV*, and RETURNED_OCTET_LENGTH is set to the length in octets of *TV*.

2) If *SDT* is CHARACTER VARYING, BIT VARYING, or BINARY LARGE OBJECT, then RETURNED_LENGTH is set to the length in characters, bits, or octets, respectively, of *SV*, and RETURNED_OCTET_LENGTH is set to the length in octets of *SV*.

3) If TYPE indicates ARRAY or ARRAY LOCATOR, then RETURNED_CARDINALITY is set to the cardinality of *TV*.

ii) If <into arguments> is specified, then the Rules in Subclause 9.1, "Retrieval assignment", are applied to *TV* and the *i*-th <into argument> as *VALUE* and *TARGET*, respectively.

NOTE 20 – All other values of the SQL descriptor area are unchanged.

Conformance Rules

1) Without Feature B032, "Extended dynamic SQL", a <descriptor name> shall be a <literal>.

2) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <output using clause>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.11 <execute statement>

Function

Associate input SQL parameters and output targets with a prepared statement and execute the statement.

Format

```
<execute statement> ::=
       EXECUTE <SQL statement name>
         [ <result using clause> ]
         [ <parameter using clause> ]

<result using clause> ::= <output using clause>

<parameter using clause> ::= <input using clause>
```

Syntax Rules

1) If <SQL statement name> is a <statement name>, then the containing <SQL-client module definition> shall contain a <prepare statement> whose <statement name> is equivalent to the <statement name> of the <execute statement>.

Access Rules

None.

General Rules

1) When the <execute statement> is executed, if the <SQL statement name> does not identify a statement *P* previously prepared in the scope of the <SQL statement name>, then an exception condition is raised: *invalid SQL statement name*.

2) Let *PS* be the statement previously prepared using <SQL statement name>.

3) If *PS* is a <dynamic select statement> that does not conform to the Format and Syntax Rules of a <dynamic single row select statement>, then an exception condition is raised: *dynamic SQL error — cursor specification cannot be executed*.

4) If *PS* contains the <table name> of a created or declared local temporary table and if the <execute statement> is not in the same <SQL-client module definition> as the <prepare statement> that prepared the prepared statement, then an exception condition is raised: *syntax error or access rule violation*.

5) If *PS* contains input <dynamic parameter specification>s and a <parameter using clause> is not specified, then an exception condition is raised: *dynamic SQL error — using clause required for dynamic parameters*.

6) If *PS* is a <dynamic single row select statement> or it contains output <dynamic parameter specification>s and a <result using clause> is not specified, then an exception condition is raised: *dynamic SQL error — using clause required for result fields*.

--`,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7) If a <parameter using clause> is specified, then the General Rules specified in Subclause 15.9, "<input using clause>", for a <parameter using clause> in an <execute statement> are applied.

8) *PS* is executed by applying the General Rules of Subclause 11.3, "<SQL procedure statement>", to *PS* as if it were immediately contained in an <externally-invoked procedure>. All General Rules are applied except those associated with input and output parameters.

9) If *PA* is a <preparable dynamic delete statement: positioned>, then when it is executed all General Rules in Subclause 15.21, "<preparable dynamic delete statement: positioned>", apply to the <preparable statement>.

10) If *PS* is a <preparable dynamic update statement: positioned>, then when it is executed, all General Rules in Subclause 15.22, "<preparable dynamic update statement: positioned>", apply to the <preparable statement>.

11) If a <result using clause> is specified, then the General Rules specified in Subclause 15.10, "<output using clause>", for a <result using clause> in an <execute statement> are applied.

Conformance Rules

1) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall contain no <result using clause>.

2) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <execute statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.12 <execute immediate statement>

Function

Dynamically prepare and execute a preparable statement.

Format

```
<execute immediate statement> ::=
      EXECUTE IMMEDIATE <SQL statement variable>
```

Syntax Rules

1) The declared type of <SQL statement variable> shall be character string.

Access Rules

None.

General Rules

1) Let *P* be the contents of the <SQL statement variable>.

2) If one or more of the following are true, then an exception condition is raised: *syntax error or access rule violation*.

   a) *P* is a <dynamic select statement> or a <dynamic single row select statement>.

   b) *P* contains a <dynamic parameter specification>.

3) Let *SV* be <SQL statement variable>. <execute immediate statement> is equivalent to the following:

```
PREPARE IMMEDIATE_STMT FROM SV ;
EXECUTE IMMEDIATE_STMT ;
DEALLOCATE PREPARE IMMEDIATE_STMT ;
```

   where *IMMEDIATE_STMT* is an implementation-defined <statement name> that is not equivalent to any other <statement name> in the containing <SQL-client module definition>.

   NOTE 21 – Exception condition or completion condition information resulting from the PREPARE or EXECUTE is reflected in the diagnostics area.

Conformance Rules

1) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <execute immediate statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.13 <dynamic declare cursor>

Function

Declare a cursor to be associated with a <statement name>, which may in turn be associated with a <cursor specification>.

Format

```
<dynamic declare cursor> ::=
       DECLARE <cursor name> [ <cursor sensitivity>  ]
         [ <cursor scrollability> ] CURSOR
         [ <cursor holdability> ]
         [ <cursor returnability> ]
           FOR <statement name>
```

Syntax Rules

1) The <cursor name> shall not be identical to the <cursor name> specified in any other <declare cursor> or <dynamic declare cursor> in the same <SQL-client module definition>.

2) The containing <SQL-client module definition> shall contain a <prepare statement> whose <statement name> is equivalent to the <statement name> of the <dynamic declare cursor>.

3) If <cursor scrollability is not specified, then NO SCROLL is implicit.

4) If <cursor holdability is not specified, then WITHOUT HOLD is implicit.

5) If <cursor returnability is not specified, then WITHOUT RETURN is implicit.

Access Rules

None.

General Rules

1) All General Rules of Subclause 14.1, "<declare cursor>", in ISO/IEC 9075-2, apply to <dynamic declare cursor>, replacing "<cursor specification>" with "prepared statement".

Conformance Rules

1) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <dynamic declare cursor>.

2) Without Feature B031, "Basic dynamic SQL", and Feature T231, "SENSITIVE cursors", a <dynamic declare cursor> shall not specify SENSITIVE.

3) Without Feature B031, "Basic dynamic SQL", and Feature F791, "Insensitive cursors", a <dynamic declare cursor> shall not specify INSENSITIVE.

4) Without Feature B031, "Basic dynamic SQL", and either Feature F791, "Insensitive cursors", or Feature T231, "SENSITIVE cursors", a <dynamic declare cursor> shall not specify ASENSITIVE.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5) Without Feature B031, "Basic dynamic SQL", and Feature T471, "Result sets return value", a <dynamic declare cursor> shall not specify <cursor returnability>.

6) Without Feature B031, "Basic dynamic SQL", and Feature F341, "Scrolled cursors", and Feature F831, "Full cursor update", if an <updatability clause> of FOR UPDATE with or without a <column name list> is specified, then <cursor scrollability> shall not be specified.

7) Without Feature B031, "Basic dynamic SQL", and Feature F831, "Full cursor update", if an <updatability clause> of FOR UPDATE with or without a <column name list> is specified, then ORDER BY shall not be specified.

8) Without Feature B031, "Basic dynamic SQL", and Feature F421, "Scrolled cursors", a <dynamic declare cursor> shall not specify <cursor scrollability>.

9) Without Feature T551, "Optional key words for default syntax", a <dynamic declare cursor> shall not specify WITHOUT HOLD.

10) Without Feature S024, "Enhanced structured types", a <value expression> that is a <sort key> shall not be of a structured type.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.14 <allocate cursor statement>

Function

Define a cursor based on a prepared statement for a <cursor specification> or assign a cursor to the ordered set of result sets returned from an SQL-invoked procedure.

Format

```
<allocate cursor statement> ::=
       ALLOCATE <extended cursor name> <cursor intent>

<cursor intent> ::=
         <statement cursor>
       | <result set cursor>

<statement cursor> ::=
       [ <cursor sensitivity> ] [ SCROLL ] CURSOR
         [ WITH HOLD ]
         [ WITH RETURN ]
           FOR <extended statement name>

<result set cursor> ::=
       FOR PROCEDURE <specific routine designator>
```

Syntax Rules

1) If <result set cursor> is specified, then the SQL-invoked routine identified by <specific routine designator> shall be an SQL-invoked procedure.

Access Rules

None.

General Rules

1) Let *S* be the <simple value specification> immediately contained in <extended cursor name>. Let *V* be the character string that is the result of

   ```
   TRIM ( BOTH ' ' FROM S )
   ```

   If *V* does not conform to the Format and Syntax Rules of an <identifier>, then an exception condition is raised: *invalid cursor name*.

2) If the value of the <extended cursor name> is identical to the value of the <extended cursor name> of any other cursor allocated in the scope of the <extended cursor name>, then an exception condition is raised: *invalid cursor name*.

3) If <statement cursor> is specified, then:

   a) When the <allocate cursor statement> is executed, if the value of the <extended statement name> does not identify a statement previously prepared in the scope of the <extended statement name>, then an exception condition is raised: *invalid SQL statement name*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) If the prepared statement associated with the <extended statement name> is not a <cursor specification>, then an exception condition is raised: *dynamic SQL error — prepared statement not a cursor specification*.

c) All General Rules of Subclause 14.1, "<declare cursor>", in ISO/IEC 9075-2, apply to <allocate cursor statement>, replacing "<open statement>" with "<dynamic open statement>" and "<cursor specification>" with "prepared statement".

d) An association is made between the value of the <extended cursor name> and the prepared statement in the scope of the <extended cursor name>. The association is preserved until the prepared statement is destroyed, at which time the cursor identified by <extended cursor name> is also destroyed.

4) If <result set cursor> is specified, then:

a) When the <allocate cursor statement> is executed, if the <specific routine designator> does not identify an SQL-invoked procedure *P* that has been previously invoked during the current SQL-session, an exception condition is raised: *invalid SQL-invoked procedure reference*.

b) If *P* did not return any result sets, then an exception condition is raised: *no data — no additional dynamic result sets returned*.

c) Let *RRS* be the ordered set of result sets returned by *P*.

d) When the <allocate cursor statement> is executed, an association is made between the <extended cursor name> and the first result set *FRS* in *RRS*. The definition of *FRS* is the definition of the <cursor specification> *CS* in *P* that created *FRS*. Let *CR* be the cursor declared by the <declare cursor> that contains *CS*.

e) Let *T* be the table specified by *CS*. *T* is the first result set returned from *P*.

f) A table descriptor for *T* is effectively created.

g) Cursor *CR* is placed in the open state and its position is before the first row of *T*.

Conformance Rules

1) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall not contain any <allocate cursor statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

15.15 <dynamic open statement>

Function

Associate input dynamic parameters with a <cursor specification> and open the cursor.

Format

```
<dynamic open statement> ::=
       OPEN <dynamic cursor name> [ <input using clause> ]
```

Syntax Rules

1) If <dynamic cursor name> *DCN* is a <cursor name> *CN*, then the containing <SQL-client module definition> shall contain a <dynamic declare cursor> whose <cursor name> is *CN*.

2) Let *CR* be the cursor identified by *DCN*.

Access Rules

1) The Access Rules for the <query expression> simply contained in the prepared statement associated with the <dynamic cursor name> are applied.

General Rules

1) If <dynamic cursor name> is a <cursor name> and the <statement name> of the associated <dynamic declare cursor> is not associated with a prepared statement, then an exception condition is raised: *invalid SQL statement name*.

2) If <dynamic cursor name> is an <extended cursor name> whose value does not identify a cursor allocated in the scope of the <extended cursor name>, then an exception condition is raised: *invalid cursor name*.

3) If the prepared statement associated with the <dynamic cursor name> contains <dynamic parameter specification>s and an <input using clause> is not specified, then an exception condition is raised: *dynamic SQL error — using clause required for dynamic parameters*.

4) The cursor specified by <dynamic cursor name> is updatable if and only if the associated <cursor specification> specified an updatable cursor.

   NOTE 22 – "updatable cursor" is defined in Subclause 14.1, "<declare cursor>", in ISO/IEC 9075-2.

5) If an <input using clause> is specified, then the General Rules specified in Subclause 15.9, "<input using clause>", for <dynamic open statement> are applied.

6) All General Rules of Subclause 14.2, "<open statement>", in ISO/IEC 9075-2, apply to the <dynamic open statement>.

Conformance Rules

1) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <dynamic open statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.16 <dynamic fetch statement>

Function

Fetch a row for a cursor declared with a <dynamic declare cursor>.

Format

```
<dynamic fetch statement> ::=
       FETCH [ [ <fetch orientation> ] FROM ] <dynamic cursor name> <output using clause>
```

Syntax Rules

1) If <fetch orientation> is omitted, then NEXT is implicit.

2) If <dynamic cursor name> *DCN* is a <cursor name> *CN*, then the containing <SQL-client module definition> shall contain a <dynamic declare cursor> whose <cursor name> is *CN*.

3) Let *CR* be the cursor identified by *DCN*.

4) If the implicit or explicit <fetch orientation> is not NEXT, then the <dynamic declare cursor> or <allocate cursor statement> associated with *CR* shall specify SCROLL.

Access Rules

None.

General Rules

1) All General Rules of Subclause 14.3, "<fetch statement>", in ISO/IEC 9075-2 are applied to cursor *CR*, <fetch orientation>, and an empty <fetch target list>.

2) The General Rules of Subclause 15.10, "<output using clause>", are applied to the <dynamic fetch statement> and the current row of *CR* as the retrieved row.

Conformance Rules

1) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <dynamic fetch statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.17 <dynamic single row select statement>

Function

Retrieve values from a dynamically-specified row of a table.

Format

```
<dynamic single row select statement> ::= <query specification>
```

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *Q* be the result of the <query specification>.

2) Case:

   a) If the cardinality of *Q* is greater than 1 (one), then an exception condition is raised: *cardinality violation*.

   b) If *Q* is empty, then a completion condition is raised: *no data*.

Conformance Rules

1) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall contain no <dynamic single row select statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.18 <dynamic close statement>

Function

Close a cursor.

Format

```
<dynamic close statement> ::=
       CLOSE <dynamic cursor name>
```

Syntax Rules

1) If <dynamic cursor name> *DCN* is a <cursor name> *CN*, then the containing <SQL-client module definition> shall contain a <dynamic declare cursor> whose <cursor name> is *CN*.

2) Let *CR* be the cursor identified by *DCN*.

Access Rules

None.

General Rules

1) All General Rules of Subclause 14.4, "<close statement>", in ISO/IEC 9075-2, apply to the <dynamic close statement>.

Conformance Rules

1) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <dynamic close statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.19 <dynamic delete statement: positioned>

Function

Delete a row of a table.

Format

```
<dynamic delete statement: positioned> ::=
       DELETE FROM <target table>
         WHERE CURRENT OF <dynamic cursor name>
```

Syntax Rules

1) If <dynamic cursor name> *DCN* is a <cursor name> *CN*, then the containing <SQL-client module definition> shall contain a <dynamic declare cursor> whose <cursor name> is *CN*.

2) Let *T* be the table identified by the <table name> contained in <target table>.

Access Rules

1) All Access Rules of Subclause 14.6, "<delete statement: positioned>", in ISO/IEC 9075-2, apply to the <dynamic delete statement: positioned>.

General Rules

1) If *DCN* is a <cursor name> and the <statement name> of the associated <dynamic declare cursor> is not associated with a prepared statement, then an exception condition is raised: *invalid SQL statement name*.

2) If *DCN* is an <extended cursor name> whose value does not identify a cursor allocated in the scope of the <extended cursor name>, then an exception condition is raised: *invalid cursor name*.

3) Let *CR* be the cursor identified by *DCN*.

4) If *CR* is not an updatable cursor, then an exception condition is raised: *invalid cursor name*.

5) Let *T* be the simply underlying table of *CR*. *T* is the subject table of the <dynamic delete statement: positioned>. *T* shall have exactly one leaf underlying table *LUT*. Let *LUTN* be a <table name> that identifies *LUT*.

6) Let *TN* be the <table name> contained in <target table>. If *TN* does not identify *LUTN*, or if ONLY is specified and the <table reference> in *T* that references *LUT* does not specify ONLY, or if ONLY is not specified and the <table reference> in *T* that references *LUT* does specify ONLY, then an exception condition is raised: *target table disagrees with cursor specification*.

7) All General Rules of Subclause 14.6, "<delete statement: positioned>", in ISO/IEC 9075-2, apply to the <dynamic delete statement: positioned>, replacing "<delete statement: positioned>" with "<dynamic delete statement: positioned>".

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.19 <dynamic delete statement: positioned>

Conformance Rules

1) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <dynamic delete statement: positioned>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.20 <dynamic update statement: positioned>

Function

Update a row of a table.

Format

```
<dynamic update statement: positioned> ::=
       UPDATE <target table>
         SET <set clause list>
           WHERE CURRENT OF <dynamic cursor name>
```

Syntax Rules

1) If <dynamic cursor name> *DCN* is a <cursor name> *CN*, then the containing <SQL-client module definition> shall contain a <dynamic declare cursor> whose <cursor name> is *CN*.

2) Let *T* be the table identified by the <table name> contained in <target table>.

3) The scope of the <table name> is the entire <dynamic update statement: positioned>.

Access Rules

1) All Access Rules of Subclause 14.9, "<update statement: positioned>", in ISO/IEC 9075-2, apply to the <dynamic update statement: positioned>.

General Rules

1) If *DCN* is a <cursor name> and the <statement name> of the associated <dynamic declare cursor> is not associated with a prepared statement, then an exception condition is raised: *invalid SQL statement name*.

2) If *DCN* is an <extended cursor name> whose value does not identify a cursor allocated in the scope of the <extended cursor name>, then an exception condition is raised: *invalid cursor name*.

3) Let *CR* be the cursor identified by *DCN*.

4) If *CR* is not an updatable cursor, then an exception condition is raised: *invalid cursor name*.

5) Let *T* be the simply underlying table of *CR*. *T* is the subject table of the <dynamic update statement: positioned>. *T* shall have exactly one leaf underlying table *LUT*. Let *LUTN* be a <table name> that identifies *LUT*.

6) Let *TN* be the <table name> contained in <target table>. If *TN* does not identify *LUTN*, or if ONLY is specified and the <table reference> in *T* that references *LUT* does not specify ONLY, or if ONLY is not specified and the <table reference> in *T* that references *LUT* does specify ONLY, then an exception condition is raised: *target table disagrees with cursor specification*.

7) If any object column is directly or indirectly referenced in the <order by clause> of the <cursor specification> for *CR*, then an exception condition is raised: *attempt to assign to ordering column*.

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8) If any object column identifies a column that is not identified by a <column name> contained in the explicit or implicit <column name list> of the explicit or implicit <updatability clause> of the <cursor specification> for *CR*, then an exception condition is raised: *attempt to assign to non-updatable column*.

9) All General Rules of Subclause 14.9, "<update statement: positioned>", in ISO/IEC 9075-2, apply to the <dynamic update statement: positioned>, replacing "<cursor name>" with "<dynamic cursor name>" and "<update statement: positioned>" with "<dynamic update statement: positioned>".

Conformance Rules

1) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <dynamic update statement: positioned>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.21 <preparable dynamic delete statement: positioned>

Function

Delete a row of a table through a dynamic cursor.

Format

```
<preparable dynamic delete statement: positioned> ::=
       DELETE [ FROM <target table> ]
         WHERE CURRENT OF [ <scope option> ] <cursor name>
```

Syntax Rules

1) If <target table> is not specified, then let *TN* be the name of the leaf underlying table *LUT* of the <cursor specification> identified by <cursor name>.

   Case:

   i) If the <table reference> that references *LUT* specifies ONLY, then the <target table>

      ```
      ONLY ( TN )
      ```

      is implicit.

   ii) Otherwise, the <target table>

      ```
      TN
      ```

      is implicit.

2) All Syntax Rules of Subclause 14.6, "<delete statement: positioned>", in ISO/IEC 9075-2, apply to the <preparable dynamic delete statement: positioned>, replacing "<declare cursor>" with "<dynamic declare cursor> or <allocate cursor statement>" and "<delete statement: positioned>" with "<preparable dynamic delete statement: positioned>".

Access Rules

1) All Access Rules of Subclause 14.6, "<delete statement: positioned>", in ISO/IEC 9075-2, apply to the <preparable dynamic delete statement: positioned>.

General Rules

1) All General Rules of Subclause 14.6, "<delete statement: positioned>", in ISO/IEC 9075-2, apply to the <preparable dynamic delete statement: positioned>, replacing "<delete statement: positioned>" with "<preparable dynamic delete statement: positioned>".

Conformance Rules

1) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall contain no <preparable dynamic delete statement: positioned>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.22 <preparable dynamic update statement: positioned>

Function

Update a row of a table through a dynamic cursor.

Format

```
<preparable dynamic update statement: positioned> ::=
       UPDATE [ <target table> ]
         SET <set clause list>
         WHERE CURRENT OF [ <scope option> ] <cursor name>
```

Syntax Rules

1) If <target table> is not specified, then let *TN* be the name of the leaf underlying table *LUT* of the <cursor specification> identified by <cursor name>.

   Case:

   i) If the <table reference> that references *LUT* specifies ONLY, then the <target table>

      ONLY (*TN*)

      is implicit.

   ii) Otherwise, the <target table>

      TN

      is implicit.

2) All Syntax Rules of Subclause 14.9, "<update statement: positioned>", in ISO/IEC 9075-2, apply to the <preparable dynamic update statement: positioned>, replacing "<declare cursor>" with "<dynamic declare cursor> or <allocate cursor statement>" and "<update statement: positioned>" with "<preparable dynamic update statement: positioned>".

Access Rules

1) All Access Rules of Subclause 14.9, "<update statement: positioned>", in ISO/IEC 9075-2, apply to the <preparable dynamic update statement: positioned>.

General Rules

1) All General Rules of Subclause 14.9, "<update statement: positioned>", in ISO/IEC 9075-2, apply to the <preparable dynamic update statement: positioned>, replacing "<update statement: positioned>" with "<preparable dynamic update statement: positioned>".

Conformance Rules

1) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall contain no <preparable dynamic update statement: positioned>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

16 Embedded SQL

16.1 <embedded SQL host program>

Function

Specify an <embedded SQL host program>.

Format

```
<embedded SQL host program> ::=
          <embedded SQL Ada program>
        | <embedded SQL C program>
        | <embedded SQL COBOL program>
        | <embedded SQL Fortran program>
        | <embedded SQL MUMPS program>
        | <embedded SQL Pascal program>
        | <embedded SQL PL/I program>

<embedded SQL statement> ::=
        <SQL prefix>
          <statement or declaration>
        [ <SQL terminator> ]

<statement or declaration> ::=
          <declare cursor>
        | <dynamic declare cursor>
        | <temporary table declaration>
        | <embedded authorization declaration>
        | <embedded path specification>
        | <embedded transform group specification>
        | <embedded exception declaration>
        | <handler declaration>
        | <SQL-invoked routine>
        | <SQL procedure statement>

<SQL prefix> ::=
          EXEC SQL
        | <ampersand>SQL<left paren>

<SQL terminator> ::=
          END-EXEC
        | <semicolon>
        | <right paren>

<embedded authorization declaration> ::=
        DECLARE <embedded authorization clause>

<embedded authorization clause> ::=
          SCHEMA <schema name>
        | AUTHORIZATION <embedded authorization identifier>
               [ FOR STATIC { ONLY | AND DYNAMIC } ]
        | SCHEMA <schema name> AUTHORIZATION <embedded authorization identifier>
               [ FOR STATIC { ONLY | AND DYNAMIC } ]
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

16.1 <embedded SQL host program>

```
<embedded authorization identifier> ::=
       <module authorization identifier>
       <embedded path specification> ::=
       <path specification>

<embedded transform group specification> ::=
       <transform group specification>

<embedded SQL declare section> ::=
         <embedded SQL begin declare>
           [ <embedded character set declaration> ]
           [ <host variable definition>... ]
         <embedded SQL end declare>
       | <embedded SQL MUMPS declare>

<embedded character set declaration> ::=
       SQL NAMES ARE <character set specification>

<embedded SQL begin declare> ::=
       <SQL prefix> BEGIN DECLARE SECTION [ <SQL terminator> ]

<embedded SQL end declare> ::=
       <SQL prefix> END DECLARE SECTION [ <SQL terminator> ]

<embedded SQL MUMPS declare> ::=
       <SQL prefix>
         BEGIN DECLARE SECTION
           [ <embedded character set declaration> ]
           [ <host variable definition>... ]
         END DECLARE SECTION
       <SQL terminator>

<host variable definition> ::=
         <Ada variable definition>
       | <C variable definition>
       | <COBOL variable definition>
       | <Fortran variable definition>
       | <MUMPS variable definition>
       | <Pascal variable definition>
       | <PL/I variable definition>

<embedded variable name> ::=
       <colon><host identifier>

<host identifier> ::=
         <Ada host identifier>
       | <C host identifier>
       | <COBOL host identifier>
       | <Fortran host identifier>
       | <MUMPS host identifier>
       | <Pascal host identifier>
       | <PL/I host identifier>
```

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Syntax Rules

1) An <embedded SQL host program> is a compilation unit that consists of programming language text and SQL text. The programming language text shall conform to the requirements of a specific standard programming language. The SQL text shall consist of one or more <embedded SQL statement>s and, optionally, one or more <embedded SQL declare section>s, as defined in this International Standard.

   NOTE 23 – "Compilation unit" is defined in Subclause 4.21, "SQL-client modules", in ISO/IEC 9075-2.

2) Case:

   a) An <embedded SQL statement> or <embedded SQL MUMPS declare> that is contained in an <embedded SQL MUMPS program> shall contain an <SQL prefix> that is "<ampersand>SQL<left paren>". There shall be no <separator> between the <ampersand> and "SQL" nor between "SQL" and the <left paren>.

   b) An <embedded SQL statement>, <embedded SQL begin declare>, or <embedded SQL end declare> that is not contained in an <embedded SQL MUMPS program> shall contain an <SQL prefix> that is "EXEC SQL".

3) Case:

   a) An <embedded SQL statement>, <embedded SQL begin declare>, or <embedded SQL end declare> contained in an <embedded SQL COBOL program> shall contain an <SQL terminator> that is END-EXEC.

   b) An <embedded SQL statement>, <embedded SQL begin declare>, or <embedded SQL end declare> contained in an <embedded SQL Fortran program> shall not contain an <SQL terminator>.

   c) An <embedded SQL statement>, <embedded SQL begin declare>, or <embedded SQL end declare> contained in an <embedded SQL Ada program>, <embedded SQL C program>, <embedded SQL Pascal program>, or <embedded SQL PL/I program> shall contain an <SQL terminator> that is a <semicolon>.

   d) An <embedded SQL statement> or <embedded SQL MUMPS declare> that is contained in an <embedded SQL MUMPS program> shall contain an <SQL terminator> that is a <right paren>.

4) Case:

   a) An <embedded SQL declare section> that is contained in an <embedded SQL MUMPS program> shall be an <embedded SQL MUMPS declare>.

   b) An <embedded SQL declare section> that is not contained in an <embedded SQL MUMPS program> shall not be an <embedded SQL MUMPS declare>.

   NOTE 24 – There is no restriction on the number of <embedded SQL declare section>s that may be contained in an <embedded SQL host program>.

5) The <token>s comprising an <SQL prefix>, <embedded SQL begin declare>, or <embedded SQL end declare> shall be separated by <space> characters and shall be specified on one line. Otherwise, the rules for the continuation of lines and tokens from one line to the next and for the placement of host language comments are those of the programming language of the containing <embedded SQL host program>.

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6) If an <embedded authorization declaration> appears in an <embedded SQL host program>, then it shall be contained in the first <embedded SQL statement> of that <embedded SQL host program>.

7) An <embedded SQL host program> shall not contain more than one <embedded path specification>.

8) An <embedded SQL host program> shall not contain more than one <embedded transform group specification>.

9) Case:

   i) If <embedded transform group specification> is not specified, then an <embedded transform group specification> containing a <multiple group specification> with a <group specification> *GS* for each <host variable declaration> that has an associated user-defined type *UDT*, but is not a user-defined locator variable is implicit. The <group name> of *GS* is implementation-defined and its <user-defined type> is *UDT*.

   ii) If <embedded transform group specification> contains a <single group specification> with a <group name> *GN*, then an <embedded transform group specification> containing a <multiple group specification> with a <group specification> *GS* for each <host variable declaration> that has an associated user-defined type *UDT*, but is not a user-defined type locator variable is implicit. The <group name> of *GS* is *GN* and its <user-defined type> is *UDT*.

   iii) If <embedded transform group specification> contains a <multiple group specification> *MGS*, then an <embedded transform group specification> containing a <multiple group specification> that contains *MGS* extended with a <group specification> *GS* for each <host variable declaration> that has an associated user-defined type *UDT*, but is not a user-defined locator variable and the <user-defined type name> of *UDT* is not contained in any <group specification> contained in *MGS* is implicit. The <group name> of *GS* is implementation-defined and its <user-defined type> is *UDT*.

10) The implicit or explicit <embedded transform group specification> precedes in the text of the <embedded SQL host program> every <host variable declaration>.

11) An <embedded SQL host program> shall contain no more than one <embedded character set declaration>. If an <embedded character set declaration> is not specified, then an <embedded character set declaration> that specifies an implementation-defined character set that contains at least every character that is in <SQL language character> is implicit.

12) A <temporary table declaration> that is contained in an <embedded SQL host program> shall precede in the text of that <embedded SQL host program> any SQL-statement or <declare cursor> that references the <table name> of the <temporary table declaration>.

13) A <declare cursor> that is contained in an <embedded SQL host program> shall precede in the text of that <embedded SQL host program> any SQL-statement that references the <cursor name> of the <declare cursor>.

14) A <dynamic declare cursor> that is contained in an <embedded SQL host program> shall precede in the text of that <embedded SQL host program> any SQL-statement that references the <cursor name> of the <dynamic declare cursor>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15) An <embedded exception declaration> that is contained in an <embedded SQL host program> shall precede in the text of that <embedded SQL host program> any SQL-statement that references the <exception name> contained in the <embedded exception declaration>.

16) An <exception name> contained in an <embedded exception declaration> that is immediately contained in an <embedded SQL host program> shall not be equivalent to any other <exception name> contained in any other <embedded exception declaration> that is immediately contained in the <embedded SQL host program>.

17) Any <host identifier> that is contained in an <embedded SQL statement> in an <embedded SQL host program> shall be defined in exactly one <host variable definition> contained in that <embedded SQL host program>. In programming languages that support <host variable definition>s in subprograms, two <host variable definition>s with different, non-overlapping scope in the host language are to be regarded as defining different host variables, even if they specify the same variable name. That <host variable definition> shall appear in the text of the <embedded SQL host program> prior to any <embedded SQL statement> that references the <host identifier>. The <host variable definition> shall be such that a host language reference to the <host identifier> is valid at every <embedded SQL statement> that contains the <host identifier>.

18) A <host variable definition> defines the host language data type of the <host identifier>. For every such host language data type an equivalent SQL <data type> is specified in Subclause 16.3, "<embedded SQL Ada program>", Subclause 16.4, "<embedded SQL C program>", Subclause 16.5, "<embedded SQL COBOL program>", Subclause 16.6, "<embedded SQL Fortran program>", Subclause 16.7, "<embedded SQL MUMPS program>", Subclause 16.8, "<embedded SQL Pascal program>", and Subclause 16.9, "<embedded SQL PL/I program>".

19) An <embedded SQL host program> shall contain a <host variable definition> that specifies SQLSTATE.

20) If one or more <host variable definition>s that specify SQLSTATE appear in an <embedded SQL host program>, then the <host variable definition>s shall be such that a host language reference to SQLSTATE is valid at every <embedded SQL statement>, including <embedded SQL statement>s that appear in any subprograms contained in that <embedded SQL host program>. The first such <host variable definition> of SQLSTATE shall appear in the text of the <embedded SQL host program> prior to any <embedded SQL statement>.

21) Given an <embedded SQL host program> *H*, there is an implied standard-conforming SQL-client module *M* and an implied standard-conforming host program *P* derived from *H*. The derivation of the implied program *P* and the implied <SQL-client module definition> *M* of an <embedded SQL host program> *H* effectively precedes the processing of any host language program text manipulation commands such as inclusion or copying of text.

    Given an <embedded SQL host program> *H* with an implied <SQL-client module definition> *M* and an implied program *P* defined as above:

    a) The implied <SQL-client module definition> *M* of *H* shall be a standard-conforming <SQL-client module definition>.

    b) If *H* is an <embedded SQL Ada program>, an <embedded SQL C program>, an <embedded SQL COBOL program>, an <embedded SQL Fortran program>, an <embedded SQL MUMPS program>, an <embedded SQL Pascal program>, or an <embedded SQL PL/I program>, then the implied program *P* shall be a standard-conforming Ada program, a

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

standard-conforming C program, a standard-conforming COBOL program, a standard-conforming Fortran program, a standard-conforming MUMPS program, a standard-conforming Pascal program, or standard-conforming PL/I program, respectively.

22) *M* is derived from *H* as follows:

a) *M* contains a <module name clause> whose <module name> is either implementation-dependent or is omitted.

b) *M* contains a <module character set specification> that is identical to the explicit or implicit <embedded character set declaration> with the keyword "SQL" removed.

c) *M* contains a <language clause> that specifies either ADA, C, COBOL, FORTRAN, MUMPS, PASCAL, or PLI, where *H* is respectively an <embedded SQL Ada program>, an <embedded SQL C program>, an <embedded SQL COBOL program>, an <embedded SQL Fortran program>, an <embedded SQL MUMPS program>, an <embedded SQL Pascal program>, or an <embedded SQL PL/I program>.

d) Case:

i) If *H* contains an <embedded authorization declaration> *EAD*, then let *EAC* be the <embedded authorization clause>contained in *EAD*; *M* contains a <module authorization clause> that specifies *EAC*.

ii) Otherwise, let *SN* be an implementation-defined <schema name>; *M* contains a <module authorization clause> that specifies "SCHEMA *SN*".

e) Case:

i) If *H* contains an <embedded path specification> *EPS*, then *M* contains the <module path specification> *EPS*.

ii) Otherwise, *M* contains am implementation-defined <module path specification>.

f) *M* contains a <module transform group specification> that is identical to the explicit or implicit <embedded transform group specification>.

g) For every <declare cursor> *EC* contained in *H*, *M* contains one <declare cursor> *PC* and one <externally-invoked procedure> *PS* that contains an <open statement> that references *PC*.

i) The <procedure name> of *PS* is implementation-dependent. *PS* contains a <host parameter declaration> *PD* for each distinct <embedded variable name> *EVN* contained in *PC* with an implementation-dependent <host parameter name> *PN* and the <host parameter data type> *PT*, determined as follows:

Case:

1) If *EVN* identifies a binary large object locator variable, then *PT* is BLOB AS LOCATOR.

2) If *EVN* identifies a character large object locator variable, then *PT* is CLOB AS LOCATOR.

3) If *EVN* identifies an array locator variable, then *PT* is *AAT* AS LOCATOR, where *AAT* is the associated array type of *V*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4) If *EVN* identifies a user-defined type locator variable, then *PT* is *UDT* AS LOCATOR, where *UDT* is the associated user-defined type of *V*.

5) Otherwise, *PT* is the SQL data type that corresponds to the host language data type of *EVN* as specified in Subclause 13.6, "Data type correspondences", in ISO/IEC 9075-2.

ii) *PS* contains a <host parameter declaration> that specifies SQLSTATE . The order of <host parameter declaration>s in *PS* is implementation-dependent. *PC* is a copy of *EC* in which each *EVN* has been replaced as follows:

Case:

1) If *EVN* does not identify user-defined type locator variable, but *EVN* identifies a host variable that has an associated user-defined type *UT*, then:

A) Let *GN* be the <group name> corresponding to the <user-defined type name> of *UT* contained in <group specification> contained in <embedded group specification>.

B) Apply the Syntax Rules of Subclause 10.17, "Determination of a to-sql function", in ISO/IEC 9075-2, with *DT* and *GN* as *TYPE* and *GROUP*, respectively. There shall be an applicable to-sql function *TSF*.

C) Let the declared type of the single SQL parameter of *TSF* be *TPT*. *PT* and *TPT* shall be mutually assignable.

D) *EVN* is replaced by:

```
TSFN(CAST(PN AS TPT))
```

2) Otherwise, *EVN* is replaced by:

```
PN
```

h) For every <dynamic declare cursor> *EC* in *H*, *M* contains one <dynamic declare cursor> *PC* that is a copy of *EC*.

i) *M* contains one <temporary table declaration> for each <temporary table declaration> contained in *H*. Each <temporary table declaration> of *M* is a copy of the corresponding <temporary table declaration> of *H*.

j) *M* contains one <embedded exception declaration> for each <embedded exception declaration> contained in *H*. Each <embedded exception declaration> of *M* is a copy of the corresponding <embedded exception declaration> of *H*.

k) *M* contains an <externally-invoked procedure> for each <SQL procedure statement> contained in *H*. The <externally-invoked procedure> *PS* of *M* corresponding with an <SQL procedure statement> *ES* of *H* is defined as follows.

Case:

i) If *ES* is not an <open statement>, then:

1) The <procedure name> of *PS* is implementation-dependent.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) Let n be the number of distinct <embedded variable name>s contained in *ES*. Let HVN_i, 1 (one) $\leq i \leq n$, be the i-th such <embedded variable name> and let HV_i be the host variable identified by HVN_i.

3) For each HVN_i, 1 (one) $\leq i \leq n$, *PS* contains a <host parameter declaration> PD_i defining a host parameter P_i such that:

   A) The <host parameter name> PN_i of PD_i is implementation-dependent.

   B) The <host parameter data type> PT_i of PD_i is determined as follows:

      I) If HV_i is a binary large object locator variable, then PT_i is BLOB AS LOCATOR.

      II) If HV_i is a character large object locator variable, then PT_i is CLOB AS LOCATOR.

      III) If HV_i is an array locator variable, then PT_i is *AAT* AS LOCATOR, where *AAT* is the associated array type of HV_i.

      IV) If HV_i is user-defined type locator variable, then PT_i is *UDT* AS LOCATOR, where *UDT* is the associated user-defined type of HV_i.

      V) Otherwise, PT_i is the SQL data type that corresponds to the host language data type of HV_i as specified in Subclause 13.6, "Data type correspondences", in ISO/IEC 9075-2.

4) *PS* contains a <host parameter declaration> that specifies SQLSTATE.

5) The order of the <host parameter declaration>s PD_i, 1 (one) $\leq i \leq n$, is implementation-dependent.

6) For each HVN_i, 1 (one) $\leq i \leq n$, that identifies some HV_i that has an associated user-defined type, but is not a user-defined type locator variable, apply the Syntax Rules of Subclause 9.6, "Host parameter mode determination", in ISO/IEC 9075-2, with the PD_i corresponding to HVN_i and *ES* as <host parameter declaration> and <SQL procedure statement>, respectively, to determine whether the corresponding P_i is an input host parameter, an output host parameter, or both an input host parameter and an output host parameter.

   A) Among P_i, 1 (one) $\leq i \leq n$, let a be the number of input host parameters, b be the number of output host parameters, and let c be the number of host parameters that are both input host parameters and output host parameters.

   B) Among P_i, 1 (one) $\leq i \leq n$, let PI_j, 1 (one) $\leq j \leq a$, be the input host parameters, let PO_k, 1 (one) $\leq k \leq b$, be the output host parameters, and let PIO_l, 1 (one) $\leq l \leq c$, be the host parameters that are both input host parameters and output host parameters.

   C) Let PNI_j, 1 (one) $\leq j \leq a$, be the <host parameter name> of PI_j. Let PNO_k, 1 (one) $\leq k \leq b$, be the <host parameter name> of PO_k. Let $PNIO_l$, 1 (one) $\leq l \leq c$, be the <host parameter name> of PIO_l.

   D) Let HVI_j, 1 (one) $\leq j \leq a$, be the host variable corresponding to PI_j. Let HVO_k, 1 (one) $\leq k \leq b$, be the host variable corresponding to PO_k. Let $HVIO_l$, 1 (one) $\leq l \leq c$, be the host variable corresponding to PIO_l.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

E) Let TSI_j, 1 (one) $\leq j \leq a$, be the associated SQL data types of HVI_j. Let TSO_k, 1 (one) $\leq k \leq b$, be the associated SQL data types of HVO_k. Let $TSIO_l$, 1 (one) $\leq l \leq c$, be the associated SQL data types of $HVIO_l$.

F) Let TUI_j, 1 (one) $\leq j \leq a$, be the associated user-defined types of HVI_j. Let TUO_k, 1 (one) $\leq k \leq b$, be the associated user-defined types of HVO_k. Let $TUIO_l$, 1 (one) $\leq l \leq c$, be the associated user-defined types of $HVIO_l$.

G) Let GNI_j, 1 (one) $\leq j \leq a$, be the <group name>s corresponding to the <user-defined type name> of TUI_j contained in the <group specification> contained in <embedded group specification>. Let GNO_k, 1 (one) $\leq k \leq b$, be the <group name>s corresponding to the <user-defined type name> of TUO_k contained in the <group specification> contained in <embedded group specification>. Let $GNIO_l$, 1 (one) $\leq l \leq c$, be the <group name>s corresponding to the <user-defined type name> of $TUIO_l$ contained in the <group specification> contained in <embedded group specification>.

H) For every j, 1 (one) $\leq j \leq a$, apply the Syntax Rules of Subclause 10.17, "Determination of a to-sql function", in ISO/IEC 9075-2, with TUI_j and GNI_j as *TYPE* and *GROUP*, respectively. There shall be an applicable to-sql function $TSFI_j$ identified by <routine name> $TSIN_j$. Let TTI_j be the data type of the single SQL parameter of $TSFI_j$. TTI_j and TSI_j shall be mutually assignable.

I) For every l, 1 (one) $\leq l \leq c$, apply the Syntax Rules of Subclause 10.17, "Determination of a to-sql function", in ISO/IEC 9075-2, with $TUIO_l$ and $GNIO_l$ as *TYPE* and Subclause 7.9, "<group by clause>", respectively. There shall be an applicable to-sql function $TSFIO_l$ identified by <routine name> $TSION_l$. Let $TTIO_l$ be the data type of the single SQL parameter of $TSFIO_l$. $TTIO_l$ and $TSIO_l$ shall be mutually assignable.

J) For every k, 1 (one) $\leq k \leq b$, apply the Syntax Rules of Subclause 10.15, "Determination of a from-sql function", in ISO/IEC 9075-2, with TUO_k and GNO_k as *TYPE* and *GROUP*, respectively. There shall be an applicable from-sql function $FSFO_k$ identified by <routine name> $FSON_k$. Let TRO_k be the result data type of $FSFO_k$. TRO_k and TSO_k shall be mutually assignable.

K) For every l, 1 (one) $\leq l \leq c$, apply the Syntax Rules of Subclause 10.15, "Determination of a from-sql function", in ISO/IEC 9075-2, with $TUIO_l$ and $GNIO_l$ as *TYPE* and *GROUP*, respectively. There shall be an applicable from-sql function $FSFIO_l$ identified by <routine name> $FSION_l$. Let $TRIO_l$ be the result data type of $FSFIO_l$. $TRIO_l$ and $TSIO_l$ shall be mutually assignable.

L) Let SVI_j, 1 (one) $\leq j \leq a$, SVO_k, 1 (one) $\leq k \leq b$, and $SVIO_l$, 1 (one) $\leq l \leq c$, be implementation-dependent <SQL variable name>s, each of which is not equivalent to any other <SQL variable name> contained in *ES*, to any <SQL parameter name> contained in *ES*, or to any <column name> contained in *ES*.

7) Let *NES* be an <SQL procedure statement> that is a copy of *ES* in which every HVN_i, 1 (one) $\leq i \leq n$, is replaced as follows.

Case:

A) If HV_i has an associated user-defined type but is not a user-defined type locator variable, then

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Case:

I) If P_i is an input host parameter, then let PI_j, 1 (one) $\leq j \leq a$, be the input host parameter that corresponds to P_i; HVN_i is replaced by SVI_j.

II) If P_i is an output host parameter, then let PO_k, 1 (one) $\leq k \leq b$, be the output host parameter that corresponds to P_i; HVN_i is replaced by SVO_k.

III) Otherwise, let PIO_l, 1 (one) $\leq l \leq c$, be the intput host parameter and the output host parameter that corresponds to P_i; HVN_i is replaced by $SVIO_l$.

B) Otherwise, HVN_i is replaced by PN_i.

8) The <SQL procedure statement> of *PS* is:

```
BEGIN ATOMIC
    DECLARE SVI1 TUI1;
    .
    .
    .
    DECLARE SVIa TUIa;
    DECLARE SVO1 TUO1;
    .
    .
    .
    DECLARE SVOa TUOa;
    DECLARE SVIO1 TUIO1;
    .
    .
    .
    DECLARE SVIOa TUIOa;

    SET SVI1 = TSIN1 (CAST (PNI1 AS TTI1));
    .
    .
    .
    SET SVIa = TSINa (CAST (PNIa AS TTIa));
    SET SVIO1 = TSION1 (CAST (PNIO1 AS TTIO1));
    .
    .
    .
    SET SVIOc = TSIONc (CAST (PNIOc AS TTIOc));

    NES;

    SET PNO1 = CAST ( FSON1 (SVO1) AS TSO1);
    .
    .
    .
    SET PNOb = CAST ( FSONb (SVOb) AS TSOb);
    SET PNIO1 = CAST ( FSION1 (SVIO1) AS TSIO1);
    .
    .
    .
    SET PNIOc = CAST ( FSIONc (SVIOc) AS TSIOc);
END;
```

9) Whether one <externally-invoked procedure> of *M* can correspond to more than one <SQL procedure statement> of *H* is implementation-dependent.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ii) If *ES* is an <open statement>, then:

1) Let *EC* be the <declare cursor> in *H* referenced by *ES*.

2) *PS* is the <externally-invoked procedure> in *M* that contains an <open statement> that references the <declare cursor> in M corresponding to *EC*.

23) *P* is derived from *H* as follows:

a) Each <embedded SQL begin declare>, <embedded SQL end declare>, and <embedded character set declaration> has been deleted. If the embedded host language is MUMPS, then each <embedded SQL MUMPS declare> has been deleted.

b) Each <host variable definition> in an <embedded SQL declare section> has been replaced by a valid data definition in the target host language according to the Syntax Rules specified in an <embedded SQL Ada program>, <embedded SQL C program>, <embedded SQL COBOL program>, <embedded SQL Fortran program>, <embedded SQL Pascal program>, or an <embedded SQL PL/I program> clause.

c) Each <embedded SQL statement> that contains a <declare cursor>, a <dynamic declare cursor>, an <SQL-invoked routine>, or a <temporary table declaration> has been deleted, and every <embedded SQL statement> that contains an <embedded exception declaration> has been replaced with statements of the host language that will have the effect specified by the General Rules of Subclause 16.2, "<embedded exception declaration>".

d) Each <embedded SQL statement> that contains an <SQL procedure statement> has been replaced by host language statements that perform the following actions:

i) A host language procedure or subroutine call of the <externally-invoked procedure> of the implied <SQL-client module definition> *M* of *H* that corresponds with the <SQL procedure statement>.

If the <SQL procedure statement> is not an <open statement>, then the arguments of the call include each distinct <host identifier> contained in the <SQL procedure statement> together with the SQLSTATE <host identifier>. If the <SQL procedure statement> is an <open statement>, then the arguments of the call include each distinct <host identifier> contained in the corresponding <declare cursor> of *H* together with the SQLSTATE <host identifier>.

The order of the arguments in the call corresponds with the order of the corresponding <host parameter declaration>s in the corresponding <externally-invoked procedure>.

NOTE 25 – In an <embedded SQL Fortran program>, the "SQLSTATE" variable may be abbreviated to "SQLSTA". See the Syntax Rules of Subclause 16.6, "<embedded SQL Fortran program>".

ii) Exception actions, as specified in Subclause 16.2, "<embedded exception declaration>".

e) Each <statement or declaration> that contains an <embedded authorization declaration> is deleted.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Access Rules

1) For every host variable whose <embedded variable name> is contained in <statement or declaration> and has an associated user-defined type, the current privileges shall include EXECUTE privilege on all from-sql functions (if any) and all to-sql functions (if any) referenced in the corresponding SQL-client module.

General Rules

1) The interpretation of an <embedded SQL host program> *H* is defined to be equivalent to the interpretation of the implied program *P* of *H* and the implied <SQL-client module definition> *M* of *H*.

2) If the cursor mode of the current SQL-transaction is *cascade off*, and the <embedded SQL statement> is other than a <close statement>, a <delete statement: positioned>, a <fetch statement>, an <open statement>, an <update statement: positioned>, a <dynamic close statement>, a <dynamic delete statement: positioned>, a <dynamic fetch statement>, a <dynamic open statement>, or a <dynamic update statement: positioned>, then an exception condition is raised: *invalid SQL statement*.

Conformance Rules

1) Without Feature B051, "Enhanced execution rights", conforming SQL language shall not contain any <embedded authorization declaration>.

2) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", an <embedded SQL declare section> shall not contain an <embedded character set declaration>.

3) Without Feature F361, "Subprogram support", no two <host variable definition>s shall specify the same variable name.

4) Without Feature S071, "SQL paths in function and type name resolution", <embedded path specification> shall not be specified.

5) Without Feature S241, "Transform functions", <embedded transform group specification> shall not be specified.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

16.2 <embedded exception declaration>

Function

Specify the action to be taken when an SQL-statement causes a specific class of condition to be raised.

Format

```
<embedded exception declaration> ::=
       WHENEVER <condition> <condition action>

<condition> ::=
       <SQL condition>

<SQL condition> ::=
         <major category>
       | SQLSTATE ( <SQLSTATE class value> [ , <SQLSTATE subclass value> ] )
       | CONSTRAINT <constraint name>

<major category> ::=
         SQLEXCEPTION
       | SQLWARNING
       | NOT FOUND

<SQLSTATE class value> ::=
       <SQLSTATE char><SQLSTATE char> !! See the Syntax Rules.

<SQLSTATE subclass value> ::=
       <SQLSTATE char><SQLSTATE char><SQLSTATE char> !! See the Syntax Rules.

<SQLSTATE char> ::= <simple Latin upper case letter> | <digit>

<condition action> ::=
       CONTINUE | <go to>

<go to> ::=
       { GOTO | GO TO } <goto target>

<goto target> ::=
         <host label identifier>
       | <unsigned integer>
       | <host PL/I label variable>

<host label identifier> ::= !! See the Syntax Rules.

<host PL/I label variable> ::= !! See the Syntax Rules.
```

Syntax Rules

1) SQLWARNING, NOT FOUND, and SQLEXCEPTION correspond to SQLSTATE class values corresponding to categories W, N, and X in Table 10, “SQLSTATE class and subclass values”, respectively.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) An <embedded exception declaration> contained in an <embedded SQL host program> applies to an <SQL procedure statement> contained in that <embedded SQL host program> if and only if the <SQL procedure statement> appears after the <embedded exception declaration> that has condition *C* in the text sequence of the <embedded SQL host program> and no other <embedded exception declaration> *E* that satisfies one of the following conditions appears between the <embedded exception declaration> and the <SQL procedure statement> in the text sequence of the <embedded SQL host program>.

   Let *D* be the <condition> contained in *E*.

   Case:

   a) *D* is the same as *C*.

   b) *D* is a <major category> and belongs to the same class to which *C* belongs.

   c) *D* contains an <SQLSTATE class value>, but does not contain an <SQLSTATE subclass value>, and *E* contains the same <SQLSTATE class value> that *C* contains.

   d) *D* contains the <SQLSTATE class value> that corresponds to *integrity constraint violation* and *C* contains CONSTRAINT.

3) In the values of <SQLSTATE class value> and <SQLSTATE subclass value>, there shall be no <separator> between the <SQLSTATE char>s.

4) The values of <SQLSTATE class value> and <SQLSTATE subclass value> shall correspond to class values and subclass values, respectively, specified in Table 10, "SQLSTATE class and subclass values".

5) If an <embedded exception declaration> specifies a <go to>, then the <host label identifier>, <host PL/I label variable>, or <unsigned integer> of the <go to> shall be such that a host language GO TO statement specifying that <host label identifier>, <host PL/I label variable>, or <unsigned integer> is valid at every <SQL procedure statement> to which the <embedded exception declaration> applies.

   NOTE 26 –

   If an <embedded exception declaration> is contained in an <embedded SQL Ada program>, then the <goto target> of a <go to> should specify a <host label identifier> that is a label_name in the containing <embedded SQL Ada program>.

   If an <embedded exception declaration> is contained in an <embedded SQL C program>, then the <goto target> of a <go to> should specify a <host label identifier> that is a label in the containing <embedded SQL C program>.

   If an <embedded exception declaration> is contained in an <embedded SQL COBOL program>, then the <goto target> of a <go to> should specify a <host label identifier> that is a section-name or an unqualified paragraph-name in the containing <embedded SQL COBOL program>.

   If an <embedded exception declaration> is contained in an <embedded SQL Fortran program>, then the <goto target> of a <go to> should be an <unsigned integer> that is the statement label of an executable statement that appears in the same program unit as the <go to>.

   If an <embedded exception declaration> is contained in an <embedded SQL MUMPS program>, then the <goto target> of a <go to> should be a gotoargument that is the statement label of an executable statement that appears in the same <embedded SQL MUMPS program>.

   If an <embedded exception declaration> is contained in an <embedded SQL Pascal program>, then the <goto target> of a <go to> should be an <unsigned integer> that is a label.

   If an <embedded exception declaration> is contained in an <embedded SQL PL/I program>, then the <goto target> of a <go to> should specify either a <host label identifier> or a <host PL/I label variable>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Case:

— If <host label identifier> is specified, then the <host label identifier> should be a label constant in the containing <embedded SQL PL/I program>.

— If <host PL/I label variable> is specified, then the <host PL/I label variable> should be a PL/I label variable declared in the containing <embedded SQL PL/I program>.

Access Rules

None.

General Rules

1) Immediately after the execution of an <SQL procedure statement> *STMT* in an <embedded SQL host program> that returns an SQLSTATE value other than *successful completion*:

   a) Let *E* be the set of <embedded exception declaration>s that are contained in the <embedded SQL host program> containing *STMT*, that applies to *STMT*, and that specifies an <condition action> that is <go to>.

   b) Let *CV* and *SCV* be respectively the values of the class and subclass of the SQLSTATE value that indicates the result of the <SQL procedure statement>.

   c) If the execution of the <SQL procedure statement> caused the violation of one or more constraints or assertions, then:

      i) Let *ECN* be the set of <embedded exception declaration>s in *E* that specify CONSTRAINT and the <constraint name> of a constraint that was violated by execution of *STMT*.

      ii) If *ECN* contains more than one <embedded exception declaration>, then an implementation-dependent <embedded exception declaration> is chosen from *ECN*; otherwise, the single <embedded exception declaration> in *ECN* is chosen.

      iii) A GO TO statement of the host language is performed, specifying the <host label identifier>, <host PL/I label variable>, or <unsigned integer> of the <go to> specified in the <embedded exception declaration> chosen from *ECN*.

   d) Otherwise:

      i) Let *ECS* be the set of <embedded exception declaration>s in *E* that specify SQLSTATE, an <SQLSTATE class value>, and an <SQLSTATE subclass value>.

      ii) If *ECS* contains an <embedded exception declaration> *EY* that specifies an <SQLSTATE class value> identical to *CV* and an <SQLSTATE subclass value> identical to *SCV*, then a GO TO statement of the host language is performed, specifying the <host label identifier>, <host PL/I label variable>, or <unsigned integer> of the <go to> specified in the <embedded exception declaration> *EY*.

      iii) Otherwise:

         1) Let *EC* be the set of <embedded exception declaration>s in *E* that specify SQLSTATE and an <SQLSTATE class value> without an <SQLSTATE subclass value>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) If *EC* contains an <embedded exception declaration> *EY* that specifies an <SQLSTATE class value> identical to *CV*, then a GO TO statement of the host language is performed, specifying the <host label identifier>, <host PL/I label variable>, or <unsigned integer> of the <go to> specified in the <embedded exception declaration> *EY*.

3) Otherwise:

   A) Let *EX* be the set of <embedded exception declaration>s in *E* that specify SQLEXCEPTION.

   B) If *EX* contains an <embedded exception declaration> *EY* and *CV* belongs to Category X in Table 10, "SQLSTATE class and subclass values", then a GO TO statement of the host language is performed, specifying the <host label identifier>, <host PL/I label variable>, or <unsigned integer> of the <go to> specified in the <embedded exception declaration> *EY*.

   C) Otherwise:

      I) Let *EW* be the set of <embedded exception declaration>s in *E* that specify SQLWARNING.

      II) If *EW* contains an <embedded exception declaration> *EY* and *CV* belongs to Category W in Table 10, "SQLSTATE class and subclass values", then a GO TO statement of the host language is performed, specifying the <host label identifier>, <host PL/I label variable>, or <unsigned integer> of the <go to> specified in the <embedded exception declaration> *EY*.

      III) Otherwise, let *ENF* be the set of <embedded exception declaration>s in *E* that specify NOT FOUND. If *ENF* contains an <embedded exception declaration> *EY* and *CV* belongs to Category N in Table 10, "SQLSTATE class and subclass values", then a GO TO statement of the host language is performed, specifying the <host label identifier>, <host PL/I label variable>, or <unsigned integer> of the <go to> specified in the <embedded exception declaration> *EY*.

Conformance Rules

1) Without Feature B041, "Extensions to embedded SQL exception declarations", an <SQL condition> shall not specify SQLSTATE or CONSTRAINT.

2) Without Feature B041, "Extensions to embedded SQL exception declarations", a <major category> shall not specify SQLEXCEPTION or SQLWARNING.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

16.3 <embedded SQL Ada program>

Function

Specify an <embedded SQL Ada program>.

Format

```
<embedded SQL Ada program> ::= !! See the Syntax Rules.

<Ada variable definition> ::=
      <Ada host identifier> [ { <comma> <Ada host identifier> }... ] <colon>
      <Ada type specification> [ <Ada initial value> ]

<Ada initial value> ::=
      <Ada assignment operator> <character representation>...

<Ada assignment operator> ::= <colon><equals operator>

<Ada host identifier> ::= !! See the Syntax Rules.

<Ada type specification> ::=
        <Ada qualified type specification>
      | <Ada unqualified type specification>
      | <Ada derived type specification>

<Ada qualified type specification> ::=
        Interfaces.SQL <period> CHAR [ CHARACTER SET [ IS ] <character set specification> ]

            <left paren> 1 <double period> <length> <right paren>
      | Interfaces.SQL <period> BIT <left paren> 1 <double period> <length> <right paren>

      | Interfaces.SQL <period> SMALLINT
      | Interfaces.SQL <period> INT
      | Interfaces.SQL <period> REAL
      | Interfaces.SQL <period> DOUBLE_PRECISION
      | Interfaces.SQL <period> BOOLEAN
      | Interfaces.SQL <period> SQLSTATE_TYPE
      | Interfaces.SQL <period> INDICATOR_TYPE

<Ada unqualified type specification> ::=
        CHAR <left paren> 1 <double period> <length> <right paren>
      | BIT <left paren> 1 <double period> <length> <right paren>
      | SMALLINT
      | INT
      | REAL
      | DOUBLE_PRECISION
      | BOOLEAN
      | SQLSTATE_TYPE
      | INDICATOR_TYPE

<Ada derived type specification> ::=
        <Ada CLOB variable>
      | <Ada BLOB variable>
      | <Ada user-defined type variable>
      | <Ada CLOB locator variable>
      | <Ada BLOB locator variable>
      | <Ada user-defined type locator variable>
      | <Ada array locator variable>
      | <Ada REF variable>
```

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

16.3 <embedded SQL Ada program>

```
<Ada CLOB variable> ::=
       SQL TYPE IS CLOB <left paren> <large object length> <right paren>
         [ CHARACTER SET [ IS ] <character set specification> ]

<Ada BLOB variable> ::=
       SQL TYPE IS BLOB <left paren> <large object length> <right paren>

<Ada user-defined type variable> ::=
       SQL TYPE IS <user-defined type> AS <predefined type>

<Ada CLOB locator variable> ::=
       SQL TYPE IS CLOB AS LOCATOR

<Ada BLOB locator variable> ::=
       SQL TYPE IS BLOB AS LOCATOR

<Ada user-defined type locator variable> ::=
       SQL TYPE IS <user-defined type name> AS LOCATOR

<Ada array locator variable> ::=
       SQL TYPE IS <collection type> AS LOCATOR

<Ada REF variable> ::=
       SQL TYPE IS <reference type>
```

Syntax Rules

1) An <embedded SQL Ada program> is a compilation unit that consists of Ada text and SQL text. The Ada text shall conform to the Ada standard ISO/IEC 8652. The SQL text shall consist of one or more <embedded SQL statement>s and, optionally, one or more <embedded SQL declare section>s.

2) An <embedded SQL statement> may be specified wherever an Ada statement may be specified. An <embedded SQL statement> may be prefixed by an Ada label.

3) An <Ada host identifier> is any valid Ada identifier. An <Ada host identifier> shall be contained in an <embedded SQL Ada program>.

4) An <Ada variable definition> defines one or more host variables.

5) An <Ada variable definition> shall be modified as follows before it is placed into the program derived from the <embedded SQL Ada program> (see the Syntax Rules of Subclause 16.1, "<embedded SQL host program>"):

   a) Any optional CHARACTER SET specification shall be removed from an <Ada qualified type specification> and <Ada derived type specification>.

   b) The <length> specified in a CHAR declaration of any <Ada qualified type specification> or <Ada derived type specification> that contains a CHARACTER SET specification shall be replaced by a length equal to the length in octets of *PN*, where *PN* is the <Ada host identifier> specified in the containing <Ada variable definition>.

   c) The syntax

      ```
      SQL TYPE IS CLOB ( L )
      ```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

and the syntax

```
SQL TYPE IS BLOB ( L )
```

for a given <Ada host identifier> *HVN* shall be replaced by

```
TYPE HVN IS RECORD
  HVN_RESERVED : Interfaces.SQL.INT;
  HVN_LENGTH : Interfaces.SQL.INT;
  HVN_DATA : Interfaces.SQL.CHAR(1..L);
END RECORD;
```

in any <Ada CLOB variable> or <Ada BLOB variable>, where *L* is the numeric value of <large object length> as specified in Subclause 5.2, "<token> and <separator>", in ISO/IEC 9075-2.

d) The syntax

```
SQL TYPE IS UDTN AS PDT
```

shall be replaced by

```
ADT
```

in any <Ada user-defined type variable>, where *ADT* is the data type listed in the "Ada data type" column corresponding to the row for SQL data type *PDT* in Table 18, "Data type correspondences for Ada", in ISO/IEC 9075-2. *ADT* shall not be "none". The data type identified by *UDTN* is called the *associated user-defined type* of the host variable and the data type identified by *PDT* is called the *associated SQL data type* of the host variable.

e) The syntax

```
SQL TYPE IS BLOB AS LOCATOR
```

shall be replaced by

```
Interfaces.SQL.INT
```

in any <Ada BLOB locator variable>. The host variable defined by <Ada BLOB locator variable> is called a *binary large object locator variable*.

f) The syntax

```
SQL TYPE IS CLOB AS LOCATOR
```

shall be replaced by

```
Interfaces.SQL.INT
```

in any <Ada CLOB locator variable>. The host variable defined by <Ada CLOB locator variable> is called a *character large object locator variable*.

g) The syntax

```
SQL TYPE IS <user-defined type> AS LOCATOR
```

shall be replaced by

```
Interfaces.SQL.INT
```

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

in any <Ada user-defined type locator variable>. The host variable defined by <Ada user-defined type locator variable> is called a *user-defined type locator variable*. The data type identified by <user-defined type> is called the *associated user-defined type* of the host variable.

h) The syntax

```
SQL TYPE IS <collection type> AS LOCATOR
```

shall be replaced by

```
Interfaces.SQL.INT
```

in any <Ada array locator variable>. The host variable defined by <Ada array locator variable> is called an *array locator variable*. The data type identified by <collection type> is called the *associated array type* of the host variable.

i) The syntax

```
SQL TYPE IS <reference type>
```

for a given <Ada host identifier> *RTV* shall be replaced by

```
RTV : Interfaces.SQL.CHAR(1..<length>)
```

in any <Ada reference type>, where <length> is the implementation-defined length in octets of a reference type.

The modified <Ada variable definition> shall be a valid Ada object-declaration in the program derived from the <embedded SQL Ada program>.

6) The reference type identified by <reference type> contained in an <Ada REF variable> is called the *referenced type* of the reference.

7) An <Ada variable definition> shall be specified within the scope of Ada **with** and **use** clauses that specify the following:

with Interfaces.SQL;

use Interfaces.SQL;

use Interfaces.SQL.CHARACTER_SET;

8) The <character representation> sequence in an <Ada initial value> specifies an initial value to be assigned to the Ada variable. It shall be a valid Ada specification of an initial value.

9) CHAR describes a character string variable whose equivalent SQL data type is CHARACTER with the same length and character set specified by <character set specification>. If <character set specification> is not specified, then an implementation-defined <character set specification> is implicit.

10) BIT describes a bit string variable. The equivalent SQL data type is BIT with the same length.

11) INT and SMALLINT describe exact numeric variables. The equivalent SQL data types are INTEGER and SMALLINT, respectively.

12) REAL and DOUBLE_PRECISION describe approximate numeric variables. The equivalent SQL data types are REAL and DOUBLE PRECISION, respectively.

13) BOOLEAN describes a boolean variable. The equivalent SQL data type is BOOLEAN.

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

14) SQLSTATE_TYPE describes a character string variable whose length is the length of the SQLSTATE parameter, five characters.

15) INDICATOR_TYPE describes an exact numeric variable whose specific data type is any <exact numeric type> with a scale of 0 (zero).

Access Rules

None.

General Rules

1) See Subclause 16.1, "<embedded SQL host program>".

Conformance Rules

1) Without Feature B011, "Embedded Ada", <embedded SQL Ada program> shall not be specified.

2) Without Feature F511, "BIT data type", an <Ada variable definition> shall not specify a bit string variable.

3) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", an <Ada qualified type specification> shall not contain a <character set specification>.

4) Without Feature S041, "Basic reference types", an <Ada derived type specification> shall not be an <Ada REF variable>.

5) Without Feature S241, "Transform functions", an <Ada derived type specification> shall not be an <Ada user-defined type variable>.

6) Without Feature S232, "Array locators", an <Ada derived type specification> shall not be an <Ada array locator variable>.

7) Without Feature S231, "Structured type locators", the <user-defined type name> simply contained in an <Ada user-defined type locator variable> shall not identify a structured type.

8) Without Feature T041, "Basic LOB data type support", an <Ada BLOB variable>, <Ada CLOB variable>, <Ada BLOB locator variable>, and <Ada CLOB locator variable> shall not be specified.

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

16.4 <embedded SQL C program>

Function

Specify an <embedded SQL C program>.

Format

```
<embedded SQL C program> ::= !! See the Syntax Rules.

<C variable definition> ::=
        [ <C storage class> ]
        [ <C class modifier> ]
        <C variable specification>
      <semicolon>

<C variable specification> ::=
        <C numeric variable>
      | <C character variable>
      | <C derived variable>

<C storage class> ::=
        auto
      | extern
      | static

<C class modifier> ::=
        const
      | volatile

<C numeric variable> ::=
      { long | short | float | double }
        <C host identifier> [ <C initial value> ]
              [ { <comma> <C host identifier> [ <C initial value> ] }... ]

<C character variable> ::=
      <C character type> [ CHARACTER SET [ IS ] <character set specification> ]
        <C host identifier> <C array specification> [ <C initial value> ]
          [ { <comma> <C host identifier> <C array specification>
                    [ <C initial value> ] }... ]

<C character type> ::=
        char
      | unsigned char
      | unsigned short

<C array specification> ::=
      <left bracket> <length> <right bracket>

<C host identifier> ::= !! See the Syntax Rules.

<C derived variable> ::=
        <C VARCHAR variable>
      | <C NCHAR variable>
      | <C NCHAR VARYING variable>
      | <C CLOB variable>
      | <C NCLOB variable>
      | <C BLOB variable>
      | <C bit variable>
```

--`,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
      | <C user-defined type variable>
      | <C CLOB locator variable>
      | <C BLOB locator variable>
      | <C array locator variable>
      | <C user-defined type locator variable>
      | <C REF variable>

<C VARCHAR variable> ::=
      VARCHAR [ CHARACTER SET [ IS ] <character set specification> ]
        <C host identifier> <C array specification> [ <C initial value> ]
          [ { <comma> <C host identifier> <C array specification> [ <C initial value> ] }... ]


<C NCHAR variable> ::=
      NCHAR [ CHARACTER SET [ IS ] <character set specification> ]
        <C host identifier> <C array specification> [ <C initial value> ]
          [ { <comma> <C host identifier> <C array specification> [ <C initial value> ] } ... ]


<C NCHAR VARYING variable> ::=
      NCHAR VARYING [ CHARACTER SET [ IS ] <character set specification> ]
        <C host identifier> <C array specification> [ <C initial value> ]
          [ { <comma> <C host identifier> <C array specification> [ <C initial value> ] } ... ]


<C CLOB variable> ::=
      SQL TYPE IS CLOB <left paren> <large object length> <right paren>
        [ CHARACTER SET [ IS ] <character set specification> ]
        <C host identifier> [ <C initial value> ]
          [ { <comma> <C host identifier> [ <C initial value> ] }... ]

<C NCLOB variable> ::=
      SQL TYPE IS NCLOB <left paren> <large object length> <right paren>
        [ CHARACTER SET [ IS ] <character set specification> ]
        <C host identifier> [ <C initial value> ]
          [ { <comma> <C host identifier> [ <C initial value> ] }... ]

<C bit variable> ::=
      BIT <C host identifier> <C array specification> [ <C initial value> ]
        [ { <comma> <C host identifier> <C array specification>
                    [ <C initial value> ] }... ]

<C user-defined type variable> ::=
      SQL TYPE IS <user-defined type> AS <predefined type>
        <C host identifier> [ <C initial value> ]
          [ { <comma> <C host identifier> [ <C initial value> ] } ... ]

<C BLOB variable> ::=
      SQL TYPE IS BLOB <left paren> <large object length> <right paren>
        <C host identifier> [ <C initial value> ]
          [ { <comma> <C host identifier> [ <C initial value> ] } ... ]

<C CLOB locator variable> ::=
      SQL TYPE IS CLOB AS LOCATOR
        <C host identifier> [ <C initial value> ]
          [ { <comma> <C host identifier> [ <C initial value> ] } ... ]

<C BLOB locator variable> ::=
        SQL TYPE IS BLOB AS LOCATOR
        <C host identifier> [ <C initial value> ]
          [ { <comma> <C host identifier> [ <C initial value> ] } ... ]
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

16.4 <embedded SQL C program>

```
<C array locator variable> ::=
        SQL TYPE IS <collection type> AS LOCATOR
        <C host identifier> [ <C initial value> ]
          [ { <comma> <C host identifier> [ <C initial value> ] } ... ]

<C user-defined type locator variable> ::=
      SQL TYPE IS
          <user-defined type> AS LOCATOR
      <C host identifier> [ <C initial value> ]
      [ { <comma> <C host identifier> [ <C initial value> ] }... ]

<C REF variable> ::=
        SQL TYPE IS <reference type>

<C initial value> ::=
      <equals operator> <character representation>...
```

Syntax Rules

1) An <embedded SQL C program> is a compilation unit that consists of C text and SQL text. The C text shall conform to the C standard ISO/IEC 9899. The SQL text shall consist of one or more <embedded SQL statement>s and, optionally, one or more <embedded SQL declare section>s.

2) An <embedded SQL statement> may be specified wherever a C statement may be specified within a function block. If the C statement could include a label prefix, then the <embedded SQL statement> may be immediately preceded by a label prefix.

3) A <C host identifier> is any valid C variable identifier. A <C host identifier> shall be contained in an <embedded SQL C program>.

4) A <C variable definition> defines one or more host variables.

5) The <collection type> contained in a <C array locator variable> shall specify an array data type. The array data type specified by the <collection type> contained in a <C array locator variable> is called the *associated array data type* of the array locator.

6) A <C variable definition> shall be modified as follows before it is placed into the program derived from the <embedded SQL C program> (see the Syntax Rules of Subclause 16.1, "<embedded SQL host program>"):

   a) Any optional CHARACTER SET specification shall be removed from a <C VARCHAR variable>, a <C character variable>, a <C CLOB variable>, a <C NCHAR variable>, <C NCHAR VARYING variable>, or a <C NCLOB variable>.

   b) The syntax "VARCHAR" shall be replaced by "char" in any <C VARCHAR variable>.

   c) The syntax "BIT" shall be replaced by "char" in any <C bit variable>.

   d) The <length> specified in a <C array specification> in any <C bit variable> shall be replaced by a length equal to the smallest integer not less than *L*/*B*, as defined in the Syntax Rules of this Subclause.

   e) The <length> specified in a <C array specification> in any <C character variable> whose <C character type> specifies "`char`" or "`unsigned char`", in any <C VARCHAR variable>, in any <C NCHAR variable>, or in any <C NCHAR VARYING variable>, and the <large object length> specified in a <C CLOB variable> that contains a CHARACTER SET specification

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

or <C NCLOB variable> shall be replaced by a length equal to the length in octets of *PN*, where *PN* is the <C host identifier> specified in the containing <C variable definition>.

NOTE 27 – The <length> does not have to be adjusted for <C character type>s that specify "`unsigned short`" because the units of <length> are already the same units as used by the underlying character set.

f) The syntax "`NCHAR`" in any <C NCHAR variable> and the syntax "`NCHAR VARYING` " in any <C NCHAR VARYING variable> shall be replaced by "`char`".

g) The syntax

```
SQL TYPE IS NCLOB ( L )
```

for a given <C host identifier> *hvn* shall be replaced by

```
struct {
  long          hvn_reserved;
  unsigned long hvn_length;
  char          hvn_data[L];
  } hvn
```

in any <CNCLOB variable>, where *L* is the numeric value of <large object length> as specified in Subclause 5.2, "<token> and <separator>", in ISO/IEC 9075-2.

h) The syntax

```
SQL TYPE IS CLOB ( L )
```

or the syntax

```
SQL TYPE IS BLOB ( L )
```

for a given <C host identifier> *hvn* shall be replaced by:

```
struct {
  long          hvn_reserved;
  unsigned long hvn_length;
  char          hvn_data[L];
  } hvn
```

in any <C CLOB variable> or <C BLOB variable>, where *L* is the numeric value of <large object length> as specified in Subclause 5.2, "<token> and <separator>", in ISO/IEC 9075-2.

i) The syntax

```
SQL TYPE IS UDTN AS PDT
```

shall be replaced by

```
ADT
```

in any <C user-defined type variable>, where *ADT* is the data type listed in the "C data type" column corresponding to the row for SQL data type *PDT* in Table 19, "Data type correspondences for C", in ISO/IEC 9075-2. *ADT* shall not be "none". The data type identified by *UDTN* is called the *associated user-defined type* of the host variable and the data type identified by *PDT* is called the *associated SQL data type* of the host variable.

j) The syntax

```
SQL TYPE IS BLOB AS LOCATOR
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

shall be replaced by

```
unsigned long
```

in any <C BLOB locator variable>. The host variable defined by <C BLOB locator variable> is called a *binary large object locator variable*.

k) The syntax

```
SQL TYPE IS CLOB AS LOCATOR
```

shall be replaced by

```
unsigned long
```

in any <C CLOB locator variable>. The host variable defined by <C CLOB locator variable> is called a *character large object locator variable*.

l) The syntax

```
SQL TYPE IS <collection type> AS LOCATOR
```

shall be replaced by

```
unsigned long
```

in any <C array locator variable>. The host variable defined by <C array locator variable> is called an *array locator variable*. The data type identified by <collection type> is called the *associated array type* of the host variable.

m) The syntax

```
SQL TYPE IS <user-defined type> AS LOCATOR
```

shall be replaced by

```
unsigned long
```

in any <C user-defined type locator variable>. The host variable defined by <C user-defined type locator variable> is called a *user-defined type locator variable*. The data type identified by <user-defined type> is called the *associated user-defined type* of the host variable.

n) The syntax

```
SQL TYPE IS <reference type>
```

for a given <C host identifier> *hvn* shall be replaced by

```
unsigned char hvn[L]
```

in any <C REF variable>, where L is the implementation-defined length in octets of a reference type.

The modified <C variable definition> shall be a valid C data declaration in the program derived from the <embedded SQL C program>.

7) The reference type identified by <reference type> contained in a <C REF variable> is called the *referenced type* of the reference.

8) The <character representation> sequence contained in a <C initial value> specifies an initial value to be assigned to the C variable. It shall be a valid C specification of an initial value.

9) Except for array specifications for character strings and bit strings, a <C variable definition> shall specify a scalar type.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

10) In a <C variable definition>, the words "VARCHAR", "CHARACTER", "SET", "IS", "BIT", "VARYING", "BLOB", "CLOB", "NCHAR", "NCLOB", "AS", "LOCATOR", and "REF" may be specified in any combination of upper-case and lower-case letters (see the Syntax Rules of Subclause 5.2, "<token> and <separator>", in ISO/IEC 9075-2).

11) In a <C character variable>, a <C VARCHAR variable>, or a <C CLOB variable>, if a <character set specification> is specified, then the equivalent SQL data type is CHARACTER, CHARACTER VARYING, or CHARACTER LARGE OBJECT whose character repertoire is the same as the repertoire specified by the <character set specification>. In a <C NCHAR variable>, a <C NCHAR VARYING variable>, or a <C NCLOB variable>, if a <character set specification> is specified, then the equivalent SQL data type is NATIONAL CHARACTER, NATIONAL CHARACTER VARYING, or NATIONAL CHARACTER LARGE OBJECT whose character repertoire is the same as the repertoire specified by the <character set specification>. If <character set specification> is not specified, then an implementation-defined <character set specification> is implicit.

12) Each <C host identifier> specified in a <C character variable> or a <C NCHAR variable> describes a fixed-length character string. The length is specified by the <length> of the <C array specification>. The value in the host variable is terminated by a null character and the position occupied by this null character is included in the length of the host variable. The equivalent SQL data type is CHARACTER or NATIONAL CHARACTER, respectively, whose length is one less than the <length> of the <C array specification> and whose value does not include the terminating null character. The <length> shall be greater than 1 (one).

13) Each <C host identifier> specified in a <C VARCHAR variable> or a <C NCHAR VARYING variable> describes a variable-length character string. The maximum length is specified by the <length> of the <C array specification>. The value in the host variable is terminated by a null character and the position occupied by this null character is included in the maximum length of the host variable. The equivalent SQL data type is CHARACTER VARYING or NATIONAL CHARACTER VARYING, respectively, whose maximum length is 1 (one) less than the <length> of the <C array specification> and whose value does not include the terminating null character. The <length> shall be greater than 1 (one).

14) Each <C host identifier> specified in a <C bit variable> describes a fixed-length bit string. The value in the host variable has a BIT_LENGTH of <length>. Let *B* be the number of bits in a C **char** and let *L* be the <length> of the <C array specification>. The length of an equivalent C **char** variable is the smallest integer that is not less than the result of *L*/*B*. The equivalent SQL data type is BIT whose length is *L*.

15) "long" describes an exact numeric variable. The equivalent SQL data type is INTEGER or BOOLEAN.

16) "short" describes an exact numeric variable. The equivalent SQL data type is SMALLINT.

17) "float" describes an approximate numeric variable. The equivalent SQL data type is REAL.

18) "double" describes an approximate numeric variable. The equivalent SQL data type is DOUBLE PRECISION.

Access Rules

None.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

16.4 <embedded SQL C program>

General Rules

1) See Subclause 16.1, "<embedded SQL host program>".

Conformance Rules

1) Without Feature B012, "Embedded C", <embedded SQL C program> shall not be specified.

2) Without Feature F511, "BIT data type", a <C derived variable> shall not be a <C bit variable>.

3) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", a <C variable definition> shall not contain a <character set specification>.

4) Without Feature S041, "Basic reference types", a <C derived variable> shall not be a <C REF variable>.

5) Without Feature S241, "Transform functions", a <C derived type specification> shall not be a <C user-defined type variable>.

6) Without Feature S232, "Array locators", a <C derived variable> shall not be a <C array locator variable>.

7) Without Feature S231, "Structured type locators", the <user-defined type name> simply contained in a <C user-defined type locator variable> shall not identify a structured type.

8) Without Feature T041, "Basic LOB data type support", a <C BLOB variable>, <C CLOB variable>, <C BLOB locator variable>, and <C CLOB locator variable> shall not be specified.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

16.5 <embedded SQL COBOL program>

Function

Specify an <embedded SQL COBOL program>.

Format

```
<embedded SQL COBOL program> ::= !! See the Syntax Rules.

<COBOL variable definition> ::=
       {01|77} <COBOL host identifier> <COBOL type specification>
         [ <character representation>... ] <period>

<COBOL host identifier> ::= !! See the Syntax Rules.

<COBOL type specification> ::=
         <COBOL character type>
       | <COBOL national character type>
       | <COBOL bit type>
       | <COBOL numeric type>
       | <COBOL integer type>
       | <COBOL derived type specification>

<COBOL derived type specification> ::=
         <COBOL CLOB variable>
       | <COBOL NCLOB variable>
       | <COBOL BLOB variable>
       | <COBOL user-defined type variable>
       | <COBOL CLOB locator variable>
       | <COBOL BLOB locator variable>
       | <COBOL array locator variable>
       | <COBOL user-defined type locator variable>
       | <COBOL REF variable>

<COBOL character type> ::=
       [ CHARACTER SET [ IS ] <character set specification> ]
       { PIC | PICTURE } [ IS ] { X [ <left paren> <length> <right paren> ] }...

<COBOL national character type> ::=
       [ CHARACTER SET [ IS ] <character set specification> ]
       { PIC | PICTURE } [ IS ] { N [ <left paren> <length> <right paren> ] }...

<COBOL CLOB variable> ::=
       [ USAGE [ IS ] ]
         SQL TYPE IS CLOB <left paren> <large object length> <right paren>
         [ CHARACTER SET [ IS ] <character set specification> ]

<COBOL NCLOB variable> ::=
       [ USAGE [ IS ] ]
         SQL TYPE IS NCLOB <left paren> <large object length> <right paren>
         [ CHARACTER SET [ IS ] <character set specification> ]

<COBOL bit type> ::=
       { PIC | PICTURE } [ IS ] { X [ <left paren> <length> <right paren> ] }...
         USAGE [ IS ] BIT

<COBOL BLOB variable> ::=
       [ USAGE [ IS ] ]
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
        SQL TYPE IS BLOB <left paren> <large object length> <right paren>

<COBOL user-defined type variable> ::=
       [ USAGE [ IS ] ]
         SQL TYPE IS <user-defined type> AS <predefined type>

<COBOL CLOB locator variable> ::=
       [ USAGE [ IS ] ]
         SQL TYPE IS CLOB AS LOCATOR

<COBOL BLOB locator variable> ::=
       [ USAGE [ IS ] ]
         SQL TYPE IS BLOB AS LOCATOR

<COBOL array locator variable> ::=
       [ USAGE [ IS ] ]
         SQL TYPE IS <collection type> AS LOCATOR

<COBOL user-defined type locator variable> ::=
       [ USAGE [ IS ] ]
         SQL TYPE IS <user-defined type name> AS LOCATOR

<COBOL REF variable> ::=
         [ USAGE [ IS ] ]
         SQL TYPE IS <reference type>

<COBOL numeric type> ::=
       { PIC | PICTURE } [ IS ]
         S <COBOL nines specification>
       [ USAGE [ IS ] ] DISPLAY SIGN LEADING SEPARATE

<COBOL nines specification> ::=
         <COBOL nines> [ V [ <COBOL nines> ] ]
       | V <COBOL nines>

<COBOL integer type> ::=
          <COBOL binary integer>

<COBOL binary integer> ::=
       { PIC | PICTURE } [ IS ] S<COBOL nines>
         [ USAGE [ IS ] ] BINARY

<COBOL nines> ::= { 9 [ <left paren> <length> <right paren> ] }...
```

NOTE 28 – The syntax "N(*L*)" is not part of the current COBOL standard, so its use is merely a recommendation; therefore, the production <COBOL national character type> is not normative in ISO/IEC 9075.

Syntax Rules

1) An <embedded SQL COBOL program> is a compilation unit that consists of COBOL text and SQL text. The COBOL text shall conform to the COBOL standard ISO 1989. The SQL text shall consist of one or more <embedded SQL statement>s and, optionally, one or more <embedded SQL declare section>s.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) An <embedded SQL statement> in an <embedded SQL COBOL program> may be specified wherever a COBOL statement may be specified in the Procedure Division of the <embedded SQL COBOL program>. If the COBOL statement could be immediately preceded by a paragraph-name, then the <embedded SQL statement> may be immediately preceded by a paragraph-name.

3) A <COBOL host identifier> is any valid COBOL data-name. A <COBOL host identifier> shall be contained in an <embedded SQL COBOL program>.

4) A <COBOL variable definition> is a restricted form of COBOL data description entry that defines a host variable.

5) The <collection type> contained in a <COBOL array locator variable> shall specify an array data type. The array data type specified by the <collection type> contained in a <COBOL array locator variable> is called the *associated array data type* of the array locator.

6) A <COBOL variable definition> shall be modified as follows before it is placed into the program derived from the <embedded SQL COBOL program> (see the Syntax Rules of Subclause 16.1, "<embedded SQL host program>").

   a) Any optional CHARACTER SET specification shall be removed from a <COBOL character type>, a <COBOL national character type>, a <COBOL CLOB variable>, and a <COBOL NCLOB variable>.

   b) The syntax

      ```
      USAGE IS BIT
      ```

      shall be deleted.

   c) The <length> specified in any <COBOL bit type> shall be replaced by a length equal to the smallest integer not less than *L*/*B*, as defined in the Syntax Rules of this Subclause.

   d) The <length> specified in any <COBOL character type> and the <large object length> specified in any <COBOL CLOB variable> or <COBOL NCLOB variable> that contains a CHARACTER SET specification shall be replaced by a length equal to the length in octets of *PN*, where *PN* is the <COBOL host identifier> specified in the containing <COBOL variable definition>.

      NOTE 29 – The <length> specified in a <COBOL national character type> does not have to be adjusted, because the units of <length> are already the same units as used by the underlying character set.

      NOTE 30 – The syntax "N(*L*)" is not part of the current COBOL standard, so its use is merely a recommendation; therefore, the production <COBOL national character type> is not normative in ISO/IEC 9075.

   e) The syntax

      ```
      SQL TYPE IS CLOB ( L )
      ```

      or the syntax

      ```
      SQL TYPE IS NCLOB ( L )
      ```

      or the syntax

      ```
      SQL TYPE IS BLOB ( L )
      ```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

for a given <COBOL host identifier> *HVN* shall be replaced by:

```
49 HVN-RESERVED PIC S9(9) USAGE IS BINARY.
49 HVN-LENGTH PIC S9(9) USAGE IS BINARY.
49 HVN-DATA PIC X(L).
```

in any <COBOL CLOB variable> or <COBOL BLOB variable>.

f) The syntax

```
SQL TYPE IS UDTN AS PDT
```

shall be replaced by

```
ADT
```

in any <COBOL user-defined type variable>, where *ADT* is the data type listed in the "COBOL data type" column corresponding to the row for SQL data type *PDT* in Table 20, "Data type correspondences for COBOL", in ISO/IEC 9075-2. *ADT* shall not be "none". The data type identified by *UDTN* is called the *associated user-defined type* of the host variable and the data type identified by *PDT* is called the *associated SQL data type* of the host variable.

g) The syntax

```
SQL TYPE IS BLOB AS LOCATOR
```

shall be replaced by

```
PIC S9(9) USAGE IS BINARY
```

in any <COBOL BLOB locator variable>. The host variable defined by <COBOL BLOB locator variable> is called a *binary large object locator variable*.

h) The syntax

```
SQL TYPE IS CLOB AS LOCATOR
```

shall be replaced by

```
PIC S9(9) USAGE IS BINARY
```

in any <COBOL CLOB locator variable>. The host variable defined by <COBOL CLOB locator variable> is called a *character large object locator variable*.

i) The syntax

```
SQL TYPE IS <collection type> AS LOCATOR
```

shall be replaced by

```
PIC S9(9) USAGE IS BINARY
```

in any <COBOL array locator variable>. The host variable defined by <COBOL array locator variable> is called an *array locator variable*. The data type identified by <collection type> is called the *associated array type* of the host variable.

j) The syntax

```
SQL TYPE IS <user-defined type> AS LOCATOR
```

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

shall be replaced by

```
PIC S9(9) USAGE IS BINARY
```

in any <COBOL user-defined type locator variable>. The host variable defined by <COBOL user-defined type locator variable> is called a *user-defined type locator variable*. The data type identified by <user-defined type> is called the *associated user-defined type* of the host variable.

k) The syntax

```
SQL TYPE IS <reference type>
```

for a given <COBOL host identifier> *HVN* shall be replaced by

```
01 HVN PICTURE X(L)
```

in any <COBOL REF variable>, where *L* is the implementation-defined length in octets of a reference type.

The modified <COBOL variable definition> shall be a valid data description entry in the Data Division of the program derived from the <embedded SQL COBOL program>.

7) The reference type identified by <reference type> contained in a <COBOL REF variable> is called the *referenced type* of the reference.

8) The optional <character representation> sequence in a <COBOL variable definition> may specify a VALUE clause. Whether other clauses may be specified is implementation-defined. The <character representation> sequence shall be such that the <COBOL variable definition> is a valid COBOL data description entry.

9) A <COBOL character type> describes a character string variable whose equivalent SQL data type is CHARACTER with the same length and character set specified by <character set specification>. If <character set specification> is not specified, then an implementation-defined <character set specification> is implicit.

10) A <COBOL bit type> describes a bit string variable. Let *B* be the number of bits in a COBOL character and let *L* be the <length> of the <COBOL bit type>. The length of an equivalent COBOL character variable is the smallest integer not less than *L*/*B*. The equivalent SQL data type is BIT whose length is the <length> of the <COBOL bit type>. If the length of the <COBOL bit type> is 1 (one), then the equivalent SQL data type may be BOOLEAN.

11) A <COBOL numeric type> describes an exact numeric variable. The equivalent SQL data type is NUMERIC of the same precision and scale.

12) A <COBOL binary integer> describes an exact numeric variable. The equivalent SQL data type is SMALLINT or INTEGER.

Access Rules

None.

General Rules

1) See Subclause 16.1, "<embedded SQL host program>".

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Conformance Rules

1) Without Feature B013, "Embedded COBOL", <embedded SQL COBOL program> shall not be specified.

2) Without Feature F511, "BIT data type", a <COBOL type specification> shall not be a <COBOL bit type>.

3) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", a <COBOL character type> shall not contain a <character set specification>.

4) Without Feature S041, "Basic reference types", a <COBOL type specification> shall not be a <COBOL REF variable>.

5) Without Feature S241, "Transform functions", a <COBOL derived type specification> shall not be a <COBOL user-defined type variable>.

6) Without Feature S232, "Array locators", a <COBOL derived type specification> shall not be a <COBOL array locator variable>.

7) Without Feature S231, "Structured type locators", the <user-defined type name> simply contained in a <COBOL user-defined type locator variable> shall not identify a structured type.

8) Without Feature T041, "Basic LOB data type support", a <COBOL BLOB variable>, <COBOL CLOB variable>, <COBOL BLOB locator variable>, and <COBOL CLOB locator variable> shall not be specified.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

16.6 <embedded SQL Fortran program>

Function

Specify an <embedded SQL Fortran program>.

Format

```
<embedded SQL Fortran program> ::= !! See the Syntax Rules.

<Fortran variable definition> ::=
       <Fortran type specification>
       <Fortran host identifier> [ { <comma> <Fortran host identifier> }... ]

<Fortran host identifier> ::= !! See the Syntax Rules.

<Fortran type specification> ::=
         CHARACTER [ <asterisk> <length> ]
             [ CHARACTER SET [ IS ] <character set specification> ]
       | CHARACTER KIND = n [ <asterisk> <length> ]
             [ CHARACTER SET [ IS ] <character set specification> ]
       | BIT [ <asterisk> <length> ]
       | INTEGER
       | REAL
       | DOUBLE PRECISION
       | LOGICAL
       | <Fortran derived type specification>

<Fortran derived type specification> ::=
         <Fortran CLOB variable>
       | <Fortran BLOB variable>
       | <Fortran user-defined type variable>
       | <Fortran CLOB locator variable>
       | <Fortran BLOB locator variable>
       | <Fortran user-defined type locator variable>
       | <Fortran array locator variable>
       | <Fortran REF variable>

<Fortran CLOB variable> ::=
       SQL TYPE IS CLOB <left paren> <large object length> <right paren>
         [ CHARACTER SET [ IS ] <character set specification> ]

<Fortran BLOB variable> ::=
       SQL TYPE IS BLOB <left paren> <large object length> <right paren>

<Fortran user-defined type variable> ::=
       SQL TYPE IS <user-defined type> AS <predefined type>

<Fortran CLOB locator variable> ::=
       SQL TYPE IS CLOB AS LOCATOR

<Fortran BLOB locator variable> ::=
       SQL TYPE IS BLOB AS LOCATOR

<Fortran user-defined type locator variable> ::=
       SQL TYPE IS <user-defined type name> AS LOCATOR

<Fortran array locator variable> ::=
       SQL TYPE IS <collection type> AS LOCATOR
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

16.6 <embedded SQL Fortran program>

```
<Fortran REF variable> ::=
      SQL TYPE IS <reference type>
```

Syntax Rules

1) An <embedded SQL Fortran program> is a compilation unit that consists of Fortran text and SQL text. The Fortran text shall conform to the Fortran standard ISO 1539. The SQL text shall consist of one or more <embedded SQL statement>s and, optionally, one or more <embedded SQL declare section>s.

2) An <embedded SQL statement> may be specified wherever an executable Fortran statement may be specified. An <embedded SQL statement> that precedes any executable Fortran statement in the containing <embedded SQL Fortran program> shall not have a Fortran statement number. Otherwise, if the Fortran statement could have a statement number then the <embedded SQL statement> can have a statement number.

3) Blanks are significant in <embedded SQL statement>s. The rules for <separator>s in an <embedded SQL statement> are as specified in Subclause 5.2, "<token> and <separator>", in ISO/IEC 9075-2.

4) A <Fortran host identifier> is any valid Fortran variable name with all <space> characters removed. A <Fortran host identifier> shall be contained in an <embedded SQL Fortran program>.

5) A <Fortran variable definition> is a restricted form of Fortran type-statement that defines one or more host variables.

6) A <Fortran variable definition> shall be modified as follows before it is placed into the program derived from the <embedded SQL Fortran program> (see the Syntax Rules Subclause 16.1, "<embedded SQL host program>").

   a) Any optional CHARACTER SET specification shall be removed from the CHARACTER and the CHARACTER KIND=*n* alternatives in a <Fortran type specification>.

   b) The <length> specified in the CHARACTER alternative of any <Fortran type specification> and the <large object length> specified in any <Fortran CLOB variable> that contains a CHARACTER SET specification shall be replaced by a length equal to the length in octets of *PN*, where *PN* is the <Fortran host identifier> specified in the containing <Fortran variable definition>.

      NOTE 31 – The <length> does not have to be adjusted for CHARACTER KIND=*n* alternatives of any <Fortran type specification>, because the units of <length> are already the same units as used by the underlying character set.

   c) The syntax "BIT" shall be replaced by "CHARACTER" in any BIT alternative of a <Fortran type specification>.

   d) The <length> specified in any BIT alternative of a <Fortran type specification> shall be replaced by a length equal to the smallest integer not less than *L*/*B*, as defined in the Syntax Rules of this Subclause.

   e) The syntax

      ```
      SQL TYPE IS CLOB ( L )
      ```

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

and the syntax

```
SQL TYPE IS BLOB ( L )
```

for a given <Fortran host identifier> *HVN* shall be replaced by

```
INTEGER HVN_RESERVED
INTEGER HVN_LENGTH
CHARACTER HVN_DATA [ <asterisk> L ]
```

in any <Fortran CLOB variable> or <Fortran BLOB variable>, where *L* is the numeric value of <large object length> as specified in Subclause 5.2, "<token> and <separator>", of ISO/IEC 9075-2.

f) The syntax

```
SQL TYPE IS UDTN AS PDT
```

shall be replaced by

```
ADT
```

in any <Fortran user-defined type variable>, where *ADT* is the data type listed in the "Fortran data type" column corresponding to the row for SQL data type *PDT* in Table 21, "Data type correspondences for Fortran", in ISO/IEC 9075-2. *ADT* shall not be "none". The data type identified by *UDTN* is called the *associated user-defined type* of the host variable and the data type identified by *PDT* is called the *associated SQL data type* of the host variable.

g) The syntax

```
SQL TYPE IS BLOB AS LOCATOR
```

shall be replaced by

```
INTEGER
```

in any <Fortran BLOB locator variable>. The host variable defined by <Fortran BLOB locator variable> is called a *binary large object locator variable*.

h) The syntax

```
SQL TYPE IS CLOB AS LOCATOR
```

shall be replaced by

```
INTEGER
```

in any <Fortran CLOB locator variable>. The host variable defined by <Fortran CLOB locator variable> is called a *character large object locator variable*.

i) The syntax

```
SQL TYPE IS <user-defined type> AS LOCATOR
```

shall be replaced by

```
INTEGER
```

in any <Fortran user-defined type locator variable>. The host variable defined by <Fortran user-defined type locator variable> is called a *user-defined type locator variable*. The data type identified by <user-defined type> is called the *associated user-defined type* of the host variable.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

j) The syntax

```
SQL TYPE IS <collection type> AS LOCATOR
```

shall be replaced by

```
INTEGER
```

in any <Fortran array locator variable>. The host variable defined by <Fortran array locator variable> is called an *array locator variable*. The data type identified by <collection type> is called the *associated array type* of the host variable.

k) The syntax

```
SQL TYPE IS <reference type>
```

for a given <Fortran host identifier> *HVN* shall be replaced by

```
CHARACTER HVN * <length>
```

in any <Fortran reference type>, where <length> is the implementation-defined in octets of a reference type.

The modified <Fortran variable definition> shall be a valid Fortran type-statement in the program derived from the <embedded SQL Fortran program>.

7) The reference type identified by <reference type> contained in an <Fortran REF variable> is called the *referenced type* of the reference.

8) CHARACTER without "KIND=*n*" describes a character string variable whose equivalent SQL data type is CHARACTER with the same length and character set specified by <character set specification>. If <character set specification> is not specified, then an implementation-defined <character set specification> is implicit.

9) CHARACTER KIND=*n* describes a character string variable whose equivalent SQL data type is either CHARACTER or NATIONAL CHARACTER with the same length and character set specified by <character set specification>. If <character set specification> is not specified, then an implementation-defined <character set specification> is implicit. The value of *n* determines implementation-defined characteristics of the Fortran variable; values of *n* are implementation-defined.

10) BIT describes a bit string variable. Let *B* be the number of bits in a Fortran character and let *L* be the <length> of the bit string variable. The length of an equivalent Fortran character variable is the smallest integer not less than *L*/*B*. The equivalent SQL data type is BIT whose length is the <length> of the bit string variable.

11) INTEGER describes an exact numeric variable. The equivalent SQL data type is INTEGER.

12) REAL describes an approximate numeric variable. The equivalent SQL data type is REAL.

13) DOUBLE PRECISION describes an approximate numeric variable. The equivalent SQL data type is DOUBLE PRECISION.

14) LOGICAL describes a boolean variable. The equivalent SQL data type is BOOLEAN.

Access Rules

None.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

General Rules

1) See Subclause 16.1, "<embedded SQL host program>".

Conformance Rules

1) Without Feature B014, "Embedded Fortran", <embedded SQL Fortran program> shall not be specified.

2) Without Feature F511, "BIT data type", a <Fortran type specification> shall not specify BIT.

3) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", a <Fortran type specification> shall not contain a <character set specification>.

4) Without Feature S041, "Basic reference types", a <Fortran derived type specification> shall not be a <Fortran REF variable>.

5) Without Feature S241, "Transform functions", a <Fortran derived type specification> shall not be a <Fortran user-defined type variable>.

6) Without Feature S232, "Array locators", a <Fortran derived type specification> shall not be a <Fortran array locator variable>.

7) Without Feature S231, "Structured type locators", the <user-defined type name> simply contained in a <Fortran user-defined type locator variable> shall not identify a structured type.

8) Without Feature T041, "Basic LOB data type support", a <Fortran BLOB variable>, <Fortran CLOB variable>, <Fortran BLOB locator variable>, and <Fortran CLOB locator variable> shall not be specified.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

16.7 <embedded SQL MUMPS program>

Function

Specify an <embedded SQL MUMPS program>.

Format

```
<embedded SQL MUMPS program> ::= !! See the Syntax Rules.

<MUMPS variable definition> ::=
        <MUMPS numeric variable> <semicolon>
      | <MUMPS character variable> <semicolon>
      | <MUMPS derived type specification> <semicolon>

<MUMPS character variable> ::=
      VARCHAR <MUMPS host identifier> <MUMPS length specification>
        [ { <comma> <MUMPS host identifier> <MUMPS length specification> }... ]

<MUMPS host identifier> ::= !! See the Syntax Rules.

<MUMPS length specification> ::=
      <left paren> <length> <right paren>

<MUMPS numeric variable> ::=
      <MUMPS type specification>
        <MUMPS host identifier> [ { <comma> <MUMPS host identifier> }... ]

<MUMPS type specification> ::=
        INT
      | DEC [ <left paren> <precision> [ <comma> <scale> ] <right paren> ]
      | REAL

<MUMPS derived type specification> ::=
        <MUMPS CLOB variable>
      | <MUMPS BLOB variable>
      | <MUMPS user-defined type variable>
      | <MUMPS CLOB locator variable>
      | <MUMPS BLOB locator variable>
      | <MUMPS user-defined type locator variable>
      | <MUMPS array locator variable>
      | <MUMPS REF variable>

<MUMPS CLOB variable> ::=
      SQL TYPE IS CLOB <left paren> <large object length> <right paren>
        [ CHARACTER SET [ IS ] <character set specification> ]


<MUMPS BLOB variable> ::=
      SQL TYPE IS BLOB <left paren> <large object length> <right paren>

<MUMPS user-defined type variable> ::=
      SQL TYPE IS <user-defined type> AS <predefined type>

<MUMPS CLOB locator variable> ::=
      SQL TYPE IS CLOB AS LOCATOR


<MUMPS BLOB locator variable> ::=
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
      SQL TYPE IS BLOB AS LOCATOR


<MUMPS user-defined type locator variable> ::=
      SQL TYPE IS <user-defined type name> AS LOCATOR

<MUMPS array locator variable> ::=
      SQL TYPE IS <collection type> AS LOCATOR

<MUMPS REF variable> ::=
      SQL TYPE IS <reference type>
```

Syntax Rules

1) An <embedded SQL MUMPS program> is a compilation unit that consists of MUMPS text and SQL text. The MUMPS text shall conform to the MUMPS standard ISO/IEC 11756. The SQL text shall consist of one or more <embedded SQL statement>s and, optionally, one or more <embedded SQL declare section>s.

2) A <MUMPS host identifier> is any valid MUMPS variable name. A <MUMPS host identifier> shall be contained in an <embedded SQL MUMPS program>.

3) An <embedded SQL statement> may be specified wherever a MUMPS command may be specified.

4) A <MUMPS variable definition> defines one or more host variables.

5) The <MUMPS character variable> describes a variable-length character string. The equivalent SQL data type is CHARACTER VARYING whose maximum length is the <length> of the <MUMPS length specification> and whose character set is implementation-defined.

6) INT describes an exact numeric variable. The equivalent SQL data type is INTEGER.

7) DEC describes an exact numeric variable. The <scale> shall not be greater than the <precision>. The equivalent SQL data type is DECIMAL with the same <precision> and <scale>.

8) REAL describes an approximate numeric variable. The equivalent SQL data type is REAL.

9) A <MUMPS derived type specification> shall be modified as follows before it is placed into the program derived from the <embedded SQL MUMPS program> (see the Syntax Rules of Subclause 16.1, "<embedded SQL host program>").

   a) Any optional CHARACTER SET specification shall be removed from a <MUMPS CLOB variable>.

   b) The syntax

      ```
      SQL TYPE IS CLOB ( L )
      ```

      and the syntax

      ```
      SQL TYPE IS BLOB ( L )
      ```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

for a given <MUMPS host identifier> *HVN* shall be replaced by

```
INT HVN_RESERVED
INT HVN_LENGTH
VARCHAR HVN_DATA L
```

in any <MUMPS CLOB variable> or <MUMPS BLOB variable>, where *L* is the numeric value of <large object length> as specified in Subclause 5.2, "<token> and <separator>", of ISO/IEC 9075-2.

c) The syntax

```
SQL TYPE IS UDTN AS PDT
```

shall be replaced by

```
ADT
```

in any <MUMPS user-defined type variable>, where *ADT* is the data type listed in the "MUMPS data type" column corresponding to the row for SQL data type *PDT* in Table 22, "Data type correspondences for MUMPS", in ISO/IEC 9075-2. *ADT* shall not be "none". The data type identified by *UDTN* is called the *associated user-defined type* of the host variable and the data type identified by *PDT* is called the *associated SQL data type* of the host variable.

d) The syntax

```
SQL TYPE IS BLOB AS LOCATOR
```

shall be replaced by

```
INT
```

in any or <MUMPS BLOB locator variable>. The host variable defined by <MUMPS BLOB locator variable> is called a *binary large object locator variable*.

e) The syntax

```
SQL TYPE IS CLOB AS LOCATOR
```

shall be replaced by

```
INT
```

in any <MUMPS CLOB locator variable>. The host variable defined by <MUMPS CLOB locator variable> is called a *character large object locator variable*.

f) The syntax

```
SQL TYPE IS <user-defined type> AS LOCATOR
```

shall be replaced by

```
INT
```

in any <MUMPS user-defined type locator variable>. The host variable defined by <MUMPS user-defined type locator variable> is called a *user-defined type locator variable*. The data type identified by <user-defined type> is called the *associated user-defined type* of the host variable.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

g) The syntax

```
SQL TYPE IS <collection type> AS LOCATOR
```

shall be replaced by

```
INT
```

in any <MUMPS array locator variable>. The host variable defined by <MUMPS array locator variable> is called an *array locator variable*. The data type identified by <collection type> is called the *associated array type* of the host variable.

h) The syntax

```
SQL TYPE IS <reference type>
```

for a given <MUMPS host identifier> *HVN* shall be replaced by

```
VARCHAR HVN L
```

in any <MUMPS reference type>, where *L* is the implementation-defined length in octets of a reference type.

The modified <MUMPS variable definition> shall be a valid MUMPS variable in the program derived from the <embedded SQL MUMPS program>.

10) The reference type identified by <reference type> contained in an <MUMPS REF variable> is called the *referenced type* of the reference.

Access Rules

None.

General Rules

1) See Subclause 16.1, "<embedded SQL host program>".

Conformance Rules

1) Without Feature B015, "Embedded MUMPS", <embedded SQL MUMPS program> shall not be specified.

2) Without Feature S041, "Basic reference types", a <MUMPS derived type specification> shall not be a <MUMPS REF variable>.

3) Without Feature S241, "Transform functions", a <MUMPS derived type specification> shall not be a <MUMPS user-defined type variable>.

4) Without Feature S232, "Array locators", a <MUMPS derived type specification> shall not be a <MUMPS array locator variable>.

5) Without Feature S231, "Structured type locators", the <user-defined type name> simply contained in a <MUMPS user-defined type locator variable> shall not identify a structured type.

6) Without Feature T041, "Basic LOB data type support", a <MUMPS BLOB variable>, <MUMPS CLOB variable>, <MUMPS BLOB locator variable>, and <MUMPS CLOB locator variable> shall not be specified.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

16.8 <embedded SQL Pascal program>

Function

Specify an <embedded SQL Pascal program>.

Format

```
<embedded SQL Pascal program> ::= !! See the Syntax Rules.

<Pascal variable definition> ::=
       <Pascal host identifier> [ { <comma> <Pascal host identifier> }... ] <colon>
         <Pascal type specification> <semicolon>

<Pascal host identifier> ::= !! See the Syntax Rules.

<Pascal type specification> ::=
         PACKED ARRAY <left bracket> 1 <double period> <length> <right bracket>
           OF CHAR
             [ CHARACTER SET [ IS ] <character set specification> ]
       | PACKED ARRAY <left bracket> 1 <double period> <length> <right bracket>
           OF BIT
       | INTEGER
       | REAL
       | CHAR [ CHARACTER SET [ IS ] <character set specification> ]
       | BIT
       | BOOLEAN
       | <Pascal derived type specification>

<Pascal derived type specification> ::=
         <Pascal CLOB variable>
       | <Pascal BLOB variable>
       | <Pascal user-defined type variable>
       | <Pascal CLOB locator variable>
       | <Pascal BLOB locator variable>
       | <Pascal user-defined type locator variable>
       | <Pascal array locator variable>
       | <Pascal REF variable>

<Pascal CLOB variable> ::=
       SQL TYPE IS CLOB <left paren> <large object length> <right paren>
         [ CHARACTER SET [ IS ] <character set specification> ]

<Pascal BLOB variable> ::=
       SQL TYPE IS BLOB <left paren> <large object length> <right paren>

<Pascal CLOB locator variable> ::=
       SQL TYPE IS CLOB AS LOCATOR

<Pascal user-defined type variable> ::=
       SQL TYPE IS <user-defined type> AS <predefined type>

<Pascal BLOB locator variable> ::=
       SQL TYPE IS BLOB AS LOCATOR

<Pascal user-defined type locator variable> ::=
       SQL TYPE IS <user-defined type name> AS LOCATOR

<Pascal array locator variable> ::=
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
      SQL TYPE IS <collection type> AS LOCATOR

<Pascal REF variable> ::=
      SQL TYPE IS <reference type>
```

Syntax Rules

1) An <embedded SQL Pascal program> is a compilation unit that consists of Pascal text and SQL text. The Pascal text shall conform to one of the Pascal standards ISO/IEC 7185 and ISO/IEC 10206. The SQL text shall consist of one or more <embedded SQL statement>s and, optionally, one or more <embedded SQL declare section>s.

2) An <embedded SQL statement> may be specified wherever a Pascal statement may be specified. An <embedded SQL statement> may be prefixed by a Pascal label.

3) A <Pascal host identifier> is a Pascal variable-identifier whose applied instance denotes a defining instance within an <embedded SQL begin declare> and an <embedded SQL end declare>.

4) A <Pascal variable definition> defines one or more <Pascal host identifier>s.

5) A <Pascal variable definition> shall be modified as follows before it is placed into the program derived from the <embedded SQL Pascal program> (see the Syntax Rules of Subclause 16.1, "<embedded SQL host program>").

   a) Any optional CHARACTER SET specification shall be removed from the PACKED ARRAY OF CHAR or CHAR alternatives of a <Pascal type specification> and a <Pascal CLOB variable>.

   b) The <length> specified in the PACKED ARRAY OF CHAR alternative of any <Pascal type specification> that contains a CHARACTER SET specification and the <large object length> specified in a <Pascal CLOB variable> that contains a CHARACTER SET specification shall be replaced by a length equal to the length in octets of *PN*, where *PN* is the <Pascal host identifier> specified in the containing <Pascal variable definition>.

   c) If any <Pascal type specification> specifies the syntax "CHAR" and contains a CHARACTER SET specification, then let *L* be a length equal to the length in octets of *PN* and *PN* be the <Pascal host identifier> specified in the containing <Pascal variable definition>. If *L* is greater than 1 (one), then "CHAR" shall be replaced by "PACKED ARRAY [1..*L*] OF CHAR".

   d) The syntax "BIT" shall be replaced by "CHAR" in any PACKED ARRAY OF BIT or BIT alternatives of a <Pascal type specification>.

   e) The <length> specified in any PACKED ARRAY OF BIT alternative in a <Pascal type specification> shall be replaced by a length equal to the smallest integer not less than *L*/*B*, as defined in the Syntax Rules of this Subclause.

   f) The syntax

      ```
      SQL TYPE IS CLOB ( L )
      ```

      and the syntax

      ```
      SQL TYPE IS BLOB ( L )
      ```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

for a given <Pascal host identifier> *HVN* shall be replaced by

```
VAR HVN = RECORD
  HVN_RESERVED : INTEGER;
  HVN_LENGTH : INTEGER;
  HVN_DATA : PACKED ARRAY [ 1..L ] OF CHAR;
END;
```

in any <Pascal CLOB variable> or <Pascal BLOB variable>, where *L* is the numeric value of <large object length> as specified in Subclause 5.2, "<token> and <separator>", of ISO/IEC 9075-2.

g) The syntax

```
SQL TYPE IS UDTN AS PDT
```

shall be replaced by

```
ADT
```

in any <Pascal user-defined type variable>, where *ADT* is the data type listed in the "Pascal data type" column corresponding to the row for SQL data type *PDT* in Table 23, "Data type correspondences for Pascal", in ISO/IEC 9075-2. *ADT* shall not be "none". The data type identified by *UDTN* is called the *associated user-defined type* of the host variable and the data type identified by *PDT* is called the *associated SQL data type* of the host variable.

h) The syntax

```
SQL TYPE IS BLOB AS LOCATOR
```

shall be replaced by

```
INTEGER
```

in any <Pascal BLOB locator variable>. The host variable defined by <Pascal BLOB locator variable> is called a *binary large object locator variable*.

i) The syntax

```
SQL TYPE IS CLOB AS LOCATOR
```

shall be replaced by

```
INTEGER
```

in any <Pascal CLOB locator variable>. The host variable defined by <Pascal CLOB locator variable> is called a *character large object locator variable*.

j) The syntax

```
SQL TYPE IS <user-defined type> AS LOCATOR
```

shall be replaced by

```
INTEGER
```

in any <Pascal user-defined type locator variable>. The host variable defined by <Pascal user-defined type locator variable> is called a *user-defined type locator variable*. The data type identified by <user-defined type> is called the *associated user-defined type* of the host variable.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

k) The syntax

```
SQL TYPE IS <collection type> AS LOCATOR
```

shall be replaced by

```
INTEGER
```

in any <Pascal array locator variable>. The host variable defined by <Pascal array locator variable> is called an *array locator variable*. The data type identified by <collection type> is called the *associated array type* of the host variable.

l) The syntax

```
SQL TYPE IS <reference type>
```

for a given <Pascal host identifier> *HVN* shall be replaced by

```
HVN : PACKED ARRAY [1..<length>] OF CHAR
```

in any <Pascal reference type>, where <length> is the implementation-defined length in octets of a reference type.

The modified <Pascal variable definition> shall be a valid Pascal variable-declaration in the program derived from the <embedded SQL Pascal program>.

6) The reference type identified by <reference type> contained in an <Pascal REF variable> is called the *referenced type* of the reference.

7) CHAR specified without a CHARACTER SET specification is the ordinal-type-identifier of PASCAL. The equivalent SQL data type is CHARACTER with length 1 (one).

8) BIT describes a single-bit variable. It is mapped to a Pascal CHAR ordinal-type-identifier whose most significant bit contains either 0 (zero) or 1 (one) and whose least significant bits contain 0 (zero). The equivalent SQL data type is BIT with length 1 (one).

9) PACKED ARRAY [1..<length>] OF CHAR describes a character string having 2 or more components of the simple type CHAR. The equivalent SQL data type is CHARACTER with the same length and character set specified by <character set specification>. If <character set specification> is not specified, then an implementation-defined <character set specification> is implicit.

10) PACKED ARRAY [1..<length>] OF BIT describes a bit string variable. Let B be the number of bits in a Pascal CHAR and let L be the <length> of the bit string variable. The length of an equivalent Pascal character variable is the smallest integer not less than L/B. The equivalent SQL data type is BIT whose length is the <length> of the bit string variable.

11) INTEGER describes an exact numeric variable. The equivalent SQL data type is INTEGER.

12) REAL describes an approximate numeric variable. The equivalent SQL data type is REAL.

13) BOOLEAN describes a boolean variable. The equivalent SQL data type is BOOLEAN.

Access Rules

None.

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

16.8 <embedded SQL Pascal program>

General Rules

1) See Subclause 16.1, "<embedded SQL host program>".

Conformance Rules

1) Without Feature B016, "Embedded Pascal", <embedded SQL Pascal program> shall not be specified.

2) Without Feature F511, "BIT data type", a <Pascal type specification> shall not specify BIT or PACKED ARRAY [1..<length>] OF BIT.

3) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", a <Pascal type specification> shall not contain a <character set specification>.

4) Without Feature S041, "Basic reference types", a <Pascal derived type specification> shall not be a <Pascal REF variable>.

5) Without Feature S241, "Transform functions", a <Pascal derived type specification> shall not be a <Pascal user-defined type variable>.

6) Without Feature S232, "Array locators", a <Pascal derived type specification> shall not be a <Pascal array locator variable>.

7) Without Feature S231, "Structured type locators", the <user-defined type name> simply contained in a <Pascal user-defined type locator variable> shall not identify a structured type.

8) Without Feature T041, "Basic LOB data type support", a <Pascal BLOB variable>, <Pascal CLOB variable>, <Pascal BLOB locator variable>, and <Pascal CLOB locator variable> shall not be specified.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

16.9 <embedded SQL PL/I program>

Function

Specify an <embedded SQL PL/I program>.

Format

```
<embedded SQL PL/I program> ::= !! See the Syntax Rules.

<PL/I variable definition> ::=
       {DCL | DECLARE}
          {   <PL/I host identifier>
            | <left paren> <PL/I host identifier>
                  [ { <comma> <PL/I host identifier> }... ] <right paren> }
       <PL/I type specification>
       [ <character representation>... ] <semicolon>

<PL/I host identifier> ::= !! See the Syntax Rules.

<PL/I type specification> ::=
         { CHAR | CHARACTER } [ VARYING ] <left paren><length><right paren>
             [ CHARACTER SET [ IS ] <character set specification> ]
       | BIT [ VARYING ] <left paren><length><right paren>
       | <PL/I type fixed decimal> <left paren> <precision>
             [ <comma> <scale> ] <right paren>
       | <PL/I type fixed binary> [ <left paren> <precision> <right paren> ]
       | <PL/I type float binary> <left paren> <precision> <right paren>
       | <PL/I derived type specification>

<PL/I derived type specification> ::=
         <PL/I CLOB variable>
       | <PL/I BLOB variable>
       | <PL/I user-defined type variable>
       | <PL/I CLOB locator variable>
       | <PL/I BLOB locator variable>
       | <PL/I user-defined type locator variable>
       | <PL/I array locator variable>
       | <PL/I REF variable>

<PL/I CLOB variable> ::=
       SQL TYPE IS CLOB <left paren> <large object length> <right paren>
         [ CHARACTER SET [ IS ] <character set specification> ]

<PL/I BLOB variable> ::=
       SQL TYPE IS BLOB <left paren> <large object length> <right paren>

<PL/I user-defined type variable> ::=
       SQL TYPE IS <user-defined type> AS <predefined type>

<PL/I CLOB locator variable> ::=
       SQL TYPE IS CLOB AS LOCATOR

<PL/I BLOB locator variable> ::=
       SQL TYPE IS BLOB AS LOCATOR

<PL/I user-defined type locator variable> ::=
       SQL TYPE IS <user-defined type name> AS LOCATOR
```

--`,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

16.9 <embedded SQL PL/I program>

```
<PL/I array locator variable> ::=
      SQL TYPE IS <collection type> AS LOCATOR

<PL/I REF variable> ::=
      SQL TYPE IS <reference type>

<PL/I type fixed decimal> ::=
        { DEC | DECIMAL } FIXED
      | FIXED { DEC | DECIMAL }

<PL/I type fixed binary> ::=
        { BIN | BINARY } FIXED
      | FIXED { BIN | BINARY }

<PL/I type float binary> ::=
        { BIN | BINARY } FLOAT
      | FLOAT { BIN | BINARY }
```

Syntax Rules

1) An <embedded SQL PL/I program> is a compilation unit that consists of PL/I text and SQL text. The PL/I text shall conform to the PL/I standard ISO 6160. The SQL text shall consist of one or more <embedded SQL statement>s and, optionally, one or more <embedded SQL declare section>s.

2) An <embedded SQL statement> may be specified wherever a PL/I statement may be specified within a procedure block. If the PL/I statement could include a label prefix, the <embedded SQL statement> may be immediately preceded by a label prefix.

3) A <PL/I host identifier> is any valid PL/I variable identifier. A <PL/I host identifier> shall be contained in an <embedded SQL PL/I program>.

4) A <PL/I variable definition> defines one or more host variables.

5) A <PL/I variable definition> shall be modified as follows before it is placed into the program derived from the <embedded SQL PL/I program> (see the Syntax Rules of Subclause 16.1, "<embedded SQL host program>").

   a) Any optional CHARACTER SET specification shall be removed from the CHARACTER or CHARACTER VARYING alternatives of a <PL/I type specification>.

   b) The <length> specified in the CHARACTER or CHARACTER VARYING alternatives of any <PL/I type specification> or a <PL/I CLOB variable> that contains a CHARACTER SET specification shall be replaced by a length equal to the length in octets of *PN*, where *PN* is the <PL/I host identifier> specified in the containing <PL/I variable definition>.

   c) The syntax

      ```
      SQL TYPE IS CLOB ( L )
      ```

      and the syntax

      ```
      SQL TYPE IS BLOB ( L )
      ```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

for a given <PL/I host identifier> *HVN* shall be replaced by

```
DCL 1 HVN
  2 HVN_RESERVED FIXED BINARY(31),
  2 HVN_LENGTH FIXED BINARY(31),
  2 HVN_DATA CHARACTER(<length>);
```

in any <PL/I CLOB variable> or <PL/I BLOB variable>, where *L* is the numeric value of <large object length> as specified in Subclause 5.2, "<token> and <separator>", of ISO/IEC 9075-2.

d) The syntax

```
SQL TYPE IS UDTN AS PDT
```

shall be replaced by

```
ADT
```

in any <PL/I user-defined type variable>, where *ADT* is the data type listed in the "PL/I data type" column corresponding to the row for SQL data type *PDT* in Table 24, "Data type correspondences for PL/I", in ISO/IEC 9075-2. *ADT* shall not be "none". The data type identified by *UDTN* is called the *associated user-defined type* of the host variable and the data type identified by *PDT* is called the *associated SQL data type* of the host variable.

e) The syntax

```
SQL TYPE IS BLOB AS LOCATOR
```

shall be replaced by

```
FIXED BINARY(31)
```

in any <PL/I BLOB locator variable>. The host variable defined by <PL/I BLOB locator variable> is called a *binary large object locator variable*.

f) The syntax

```
SQL TYPE IS CLOB AS LOCATOR
```

shall be replaced by

```
FIXED BINARY(31)
```

in any <PL/I CLOB locator variable>. The host variable defined by <PL/I CLOB locator variable> is called a *character large object locator variable*.

g) The syntax

```
SQL TYPE IS <user-defined type> AS LOCATOR
```

shall be replaced by

```
FIXED BINARY(31)
```

in any <PL/I user-defined type locator variable>. The host variable defined by <PL/I user-defined type locator variable> is called a *user-defined type locator variable*. The data type identified by <user-defined type> is called the *associated user-defined type* of the host variable.

h) The syntax

```
SQL TYPE IS <collection type> AS LOCATOR
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

shall be replaced by

```
FIXED BINARY(31)
```

in any <PL/I array locator variable>. The host variable defined by <PL/I array locator variable> is called an *array locator variable*. The data type identified by <collection type> is called the *associated array type* of the host variable.

i) The syntax

```
SQL TYPE IS <reference type>
```

for a given <PL/I host identifier> *HVN* shall be replaced by

```
DCL HVN CHARACTER(<length>) VARYING
```

in any <PL/I reference type>, where <length> is the implementation-defined length in octets of a reference type.

The modified <PL/I variable definition> shall be a valid PL/I data declaration in the program derived from the <embedded SQL PL/I program>.

6) The reference type identified by <reference type> contained in an <PL/I REF variable> is called the *referenced type* of the reference.

7) A <PL/I variable definition> shall specify a scalar variable, not an array or structure.

8) The optional <character representation> sequence in a <PL/I variable definition> may specify an INITIAL clause. Whether other clauses may be specified is implementation-defined. The <character representation> sequence shall be such that the <PL/I variable definition> is a valid PL/I DECLARE statement.

9) CHARACTER describes a character string variable whose equivalent SQL data type has the character set specified by <character set specification>. If <character set specification> is not specified, then an implementation-defined <character set specification> is implicit.

Case:

a) If VARYING is not specified, then the length of the variable is fixed. The equivalent SQL data type is CHARACTER with the same length.

b) If VARYING is specified, then the variable is of variable length, with maximum size the value of <length>. The equivalent SQL data type is CHARACTER VARYING with the same maximum length.

10) BIT describes a bit string variable.

Case:

a) If VARYING is not specified, then the length of the variable is fixed. The equivalent SQL data type is BIT with the same length. If the length of the variable is 1 (one), then the equivalent SQL data type may be BOOLEAN.

b) If VARYING is specified, then the variable is of variable length with maximum size of the value of <length>. The equivalent SQL data type is BIT VARYING with the same maximum length.

11) FIXED DECIMAL describes an exact numeric variable. The <scale> shall not be greater than the <precision>. The equivalent SQL data type is DECIMAL with the same <precision> and <scale>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

12) FIXED BINARY describes an exact numeric variable. The equivalent SQL data type is SMALLINT or INTEGER.

13) FLOAT BINARY describes an approximate numeric variable. The equivalent SQL data type is FLOAT with the same <precision>.

Access Rules

None.

General Rules

1) See Subclause 16.1, "<embedded SQL host program>"

Conformance Rules

1) Without Feature B017, "Embedded PL/I", <embedded SQL PL/I program> shall not be specified.

2) Without Feature F511, "BIT data type", a <PL/I type specification> shall not specify BIT or BIT VARYING.

3) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", a <PL/I type specification> shall not contain a <character set specification>.

4) Without Feature S041, "Basic reference types", a <PL/I derived type specification> shall not be a <PL/I REF variable>.

5) Without Feature S241, "Transform functions", a <PL/I derived type specification> shall not be a <PL/I user-defined type variable>.

6) Without Feature S232, "Array locators", a <PL/I derived type specification> shall not be a <PL/I array locator variable>.

7) Without Feature S231, "Structured type locators", the <user-defined type name> simply contained in a <PL/I user-defined type locator variable> shall not identify a structured type.

8) Without Feature T041, "Basic LOB data type support", a <PL/I BLOB variable>, <PL/I CLOB variable>, <PL/I BLOB locator variable>, and <PL/I CLOB locator variable> shall not be specified.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

17 Direct invocation of SQL

17.1 <direct SQL statement>

Function

Specify direct execution of SQL.

Format

```
<direct SQL statement> ::=
       <directly executable statement> <semicolon>

<directly executable statement> ::=
         <direct SQL data statement>
       | <SQL schema statement>
       | <SQL transaction statement>
       | <SQL connection statement>
       | <SQL session statement>
       | <direct implementation-defined statement>

<direct SQL data statement> ::=
         <delete statement: searched>
       | <direct select statement: multiple rows>
       | <insert statement>
       | <update statement: searched>
       | <temporary table declaration>

<direct implementation-defined statement> ::= !! See the Syntax Rules
```

Syntax Rules

1) The <direct SQL data statement> shall not contain any SQL parameter reference, SQL variable reference, <dynamic parameter specification>, or <embedded variable specification>.

2) The <value specification> that represents the null value is implementation-defined.

3) The Format and Syntax Rules for <direct implementation-defined statement> are implementation-defined.

Access Rules

1) The Access Rules for <direct implementation-defined statement> are implementation-defined.

General Rules

1) The following <direct SQL statement>s are transaction-initiating <direct SQL statement>s:

   a) <direct SQL statement>s that are transaction-initiating <SQL procedure statement>s;

   b) <direct select statement: multiple rows>; and

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c) <direct implementation-defined statement>s that are transaction-initiating.

2) After the last invocation of an SQL-statement by an SQL-agent in an SQL-session:

a) A <rollback statement> or a <commit statement> is effectively executed. If an unrecoverable error has occurred, or if the direct invocation of SQL terminated unexpectedly, or if any constraint is not satisfied, then a <rollback statement> is performed. Otherwise, the choice of which of these SQL-statements to perform is implementation-dependent. The determination of whether a direct invocation of SQL has terminated unexpectedly is implementation-dependent.

b) Let *D* be the <descriptor name> of any system descriptor area that is currently allocated within the current SQL-session. A <deallocate descriptor statement> that specifies

```
DEALLOCATE DESCRIPTOR D
```

is effectively executed.

c) All SQL-sessions associated with the SQL-agent are terminated.

3) Let *S* be the <direct SQL statement>.

4) The current authorization identifier for privilege determination for the execution of *S* is the SQL-session user identifier.

5) If *S* does not conform to the Format, Syntax Rules, and Access Rules for a <direct SQL statement>, then an exception condition is raised: *syntax error or access rule violation*.

6) When *S* is invoked by the SQL-agent:

Case:

a) If *S* is an <SQL connection statement>, then:

i) The diagnostics area is emptied.

ii) *S* is executed.

iii) If *S* successfully initiated or resumed an SQL-session, then subsequent invocations of a <direct SQL statement> by the SQL-agent are associated with that SQL-session until the SQL-agent terminates the SQL-session or makes it dormant.

b) Otherwise:

i) If no SQL-session is current for the SQL-agent, then

Case:

1) If the SQL-agent has not executed an <SQL connection statement> and there is no default SQL-session associated with the SQL-agent, then the following <connect statement> is effectively executed:

```
CONNECT TO DEFAULT
```

2) If the SQL-agent has not executed an <SQL connection statement> and there is a default SQL-session associated with the SQL-agent, then the following <set connection statement> is effectively executed:

```
SET CONNECTION DEFAULT
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3) Otherwise, an exception condition is raised: *connection exception — connection does not exist*.

Subsequent calls to an <externally-invoked procedure> or invocations of a <direct SQL statement> by the SQL-agent are associated with the SQL-session until the SQL-agent terminates the SQL-session or makes it dormant.

ii) If an SQL-transaction is active for the SQL-agent, then *S* is associated with that SQL-transaction. If *S* is a <direct implementation-defined statement>, then it is implementation-defined whether or not *S* may be associated with an active SQL-transaction; if not, then an exception condition is raised: *invalid transaction state — active SQL-transaction*.

iii) If no SQL-transaction is active for the SQL-agent, then

1) Case:

A) If *S* is a transaction-initiating <direct SQL statement>, then an SQL-transaction is initiated.

B) If *S* is a <direct implementation-defined statement>, then it is implementation-defined whether or not *S* initiates an SQL-transaction. If an implementation defines *S* to be transaction-initiating, then an SQL-transaction is initiated.

2) If *S* initiated an SQL-transaction, then:

A) Let *T* be the SQL-transaction initiated by *S*.

B) *T* is associated with this invocation and any subsequent invocations of <direct SQL statement>s or calls to an <externally-invoked procedure> by the SQL-agent until the SQL-agent terminates *T*.

C) If *S* is not a <start transaction statement>, then

Case:

I) If a <set transaction statement> has been executed since the termination of the last SQL-transaction in the SQL-session (or if there has been no previous SQL-transaction in the SQL-session and a <set transaction statement> has been executed), then the access mode, constraint mode, and isolation level of *T* are set as specified by the <set transaction statement>.

II) Otherwise, the access mode, constraint mode for all constraints, and isolation level for *T* are *read-write*, *immediate*, and SERIALIZABLE, respectively.

D) *T* is associated with the SQL-session.

iv) If *S* contains an <SQL schema statement> and the access mode of the current SQL-transaction is *read-only*, then an exception condition is raised: *invalid transaction state — read-only SQL-transaction*.

v) The diagnostics area is emptied.

vi) *S* is executed.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7) If the execution of a <direct SQL data statement> occurs within the same SQL-transaction as the execution of an SQL-schema statement and this is not allowed by the SQL-implementation, then an exception condition is raised: *invalid transaction state — schema and data statement mixing not supported.*

8) Case:

   a) If *S* executed successfully, then either a completion condition is raised: *successful completion*, or a completion condition is raised: *warning*, or a completion condition is raised: *no data*.

   b) If *S* did not execute successfully, then all changes made to SQL-data or schemas by the execution of *S* are canceled and an exception condition is raised.

   NOTE 32 – The method of raising a condition is implementation-defined.

9) Diagnostics information resulting from the execution of *S* is placed into the diagnostics area as specified in Clause 18, "Diagnostics management".

   NOTE 33 – The method of accessing the diagnostics information is implementation-defined, but does not alter the contents of the diagnostics area.

Conformance Rules

1) Without Feature B021, "Direct SQL", <direct SQL statement> shall not be specified.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

17.2 <direct select statement: multiple rows>

Function

Specify a statement to retrieve multiple rows from a specified table.

Format

```
<direct select statement: multiple rows> ::=
      <query expression> [ <order by clause> ]
```

Syntax Rules

1) All Syntax Rules of Subclause 7.12, "<query expression>", in ISO/IEC 9075-2, apply to the <direct select statement: multiple rows>.

2) The <query expression> or <order by clause> of a <direct select statement: multiple rows> shall not contain any <value specification> other than a <literal>, CURRENT_USER, CURRENT_ROLE, SESSION_USER, SYSTEM_USER, or CURRENT_PATH.

3) Let *T* be the table specified by the <query expression>.

4) If ORDER BY is specified, then each <sort specification> in the <order by clause> shall contain a <value expression> *VE*, which shall not contain a <subquery> or a <set function specification>, but shall contain a <column reference>.

   a) Let *X* be any <column reference> directly contained in *VE*.

   b) If *X* does not contain an explicit <table or query name> or <correlation name>, then *VE* shall be a <column name> that shall be equivalent to the name of exactly one column of *ST*.

   NOTE 34 – A previous version of ISO/IEC 9075 allows <sort specification> to simply contain a <signed integer> to denote a column reference of a column of *T*. That facility no longer exists. See Appendix E, "Incompatibilities with ISO/IEC 9075:1992 and ISO/IEC 9075-4:1996", in ISO/IEC 9075.

Access Rules

None.

General Rules

1) All General Rules of Subclause 7.12, "<query expression>", in ISO/IEC 9075-2, apply to the <direct select statement: multiple rows>.

2) Let *Q* be the result of the <query expression>.

3) If *Q* is empty, then a completion condition is raised: *no data*.

4) If an <order by clause> is not specified, then the ordering of the rows of *Q* is implementation-dependent.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5) If an <order by clause> is specified, then the ordering of rows of the result is effectively determined by the <order by clause> as follows:

   a) Each <sort specification> specifies the sort direction for the corresponding sort key K_i. If ASC is specified or implied in the i-th <sort specification>, then the sort direction for K_i is ascending and the applicable <comp op> is the <less than operator>. Otherwise, the sort direction for K_i is descending and the applicable <comp op> is the <greater than operator>.

   b) Let X and Y be distinct rows in the result table, and let XV_i and YV_i be the values of K_i in these rows, respectively. The relative position of rows X and Y in the result is determined by comparing XV_i and YV_i according to the rules of Subclause 8.2, "<comparison predicate>", in ISO/IEC 9075-2, where the <comp op> is the applicable <comp op> for K_i, with the following special treatment of null values. Whether a sort key value that is null is considered greater or less than a non-null value is implementation-defined, but all sort key values that are null shall either be considered greater than all non-null values or be considered less than all non-null values. XV_i is said to *precede* YV_i if the value of the <comparison predicate> "XV_i <comp op> YV_i" is true for the applicable <comp op>.

   c) In the result table, the relative position of row X is before row Y if and only if XV_n precedes YV_n for some n greater than 0 (zero) and less than the number of <sort specification>s and $XV_i = YV_i$ for all $i < n$. The relative order of two rows for which $XV_i = YV_i$ for all i is implementation-dependent.

6) If Q is not empty, then Q is returned. The method of returning Q is implementation-defined.

Conformance Rules

None.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

18 Diagnostics management

18.1 <get diagnostics statement>

Function

Get exception or completion condition information from the diagnostics area.

Format

```
<statement information item name> ::=
         !! All alternatives from ISO/IEC 9075-2
       | DYNAMIC_FUNCTION
       | DYNAMIC_FUNCTION_CODE
```

Syntax Rules

Table 8—<identifier>s for use with <get diagnostics statement>

<identifier>	Data Type
<statement information item name>s	
All alternatives from ISO/IEC 9075-2	
DYNAMIC_FUNCTION	character varying (*L*)
DYNAMIC_FUNCTION_CODE	exact numeric with scale 0 (zero)

Access Rules

No additional Access Rules.

General Rules

1) Augments GR1) Specification of <statement information item> retrieves information about the statement execution recorded in the diagnostics area into <simple target specification>.

   a) The value of DYNAMIC_FUNCTION is a character string that identifies the type of the SQL-statement being prepared or executed dynamically. Table 9, "SQL-statement codes", specifies the identifier of the SQL-statements.

   b) The value of DYNAMIC_FUNCTION_CODE is a number that identifies the type of the SQL-statement being prepared or executed dynamically. Table 9, "SQL-statement codes", specifies the code for the SQL-statements. Positive values are reserved for SQL-statements defined by ISO/IEC 9075; negative values are reserved for implementation-defined SQL-statements.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 9—SQL-statement codes

SQL-statement	Identifier	Code
All alternatives from ISO/IEC 9075-2		
<allocate cursor statement>	ALLOCATE CURSOR	1 (one)
<allocate descriptor statement>	ALLOCATE DESCRIPTOR	2
<deallocate descriptor statement>	DEALLOCATE DESCRIPTOR	15
<deallocate prepared statement>	DEALLOCATE PREPARE	16
<describe statement>	DESCRIBE	20
<direct select statement: multiple rows>	SELECT	21
<dynamic close statement>	DYNAMIC CLOSE	37
<dynamic delete statement: positioned>	DYNAMIC DELETE CURSOR	38
<dynamic fetch statement>	DYNAMIC FETCH	39
<dynamic open statement>	DYNAMIC OPEN	40
<dynamic select statement>	SELECT CURSOR	85
<dynamic single row select statement>	SELECT	41
<dynamic update statement: positioned>	DYNAMIC UPDATE CURSOR	42
<execute immediate statement>	EXECUTE IMMEDIATE	43
<execute statement>	EXECUTE	44
<get descriptor statement>	GET DESCRIPTOR	47
<preparable dynamic delete statement: positioned>	DYNAMIC DELETE CURSOR	54
<preparable dynamic update statement: positioned>	DYNAMIC UPDATE CURSOR	55
<prepare statement>	PREPARE	56
<set catalog statement>	SET CATALOG	66
<set descriptor statement>	SET DESCRIPTOR	70
<set path statement>	SET CURRENT_PATH	69
<set names statement>	SET NAMES	72
<set schema statement>	SET SCHEMA	74
<set transform group statement>	SET TRANSFORM GROUP	118
Implementation-defined statements	An implementation-defined character string value different from the value associated with any other SQL-statement	*x*[1]
Unrecognized statements	A zero-length string	0 (zero)

[1]An implementation-defined negative number different from the value associated with any other SQL-statement.

2) Augments GR1) It is implementation-defined whether the identifier and code from Table 9, "SQL-statement codes", for <dynamic select statement> or <dynamic single row select statement> is used to describe a <dynamic select statement> or <dynamic single row select statement> that has been prepared but has not yet been executed dynamically.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Conformance Rules

No additional Conformance Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

--`,,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

19 Definition Schema

19.1 SQL_LANGUAGES base table

Function

The SQL_LANGUAGES table has one row for each ISO and implementation-defined SQL language binding and programming language for which conformance is claimed.

Definition

Add a new table constraint

```
CONSTRAINT SQL_LANGUAGE_BINDING_STYLE_ISO_1999
  CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
        OR
          ( SQL_LANGUAGE_YEAR <> '1999'
          OR
            ( SQL_LANGUAGE_BINDING_STYLE IS NOT NULL
            AND
              SQL_LANGUAGE_BINDING_STYLE IN
              ( 'DIRECT', 'EMBEDDED', 'MODULE' ) ) ) ),
```

Add a new table constraint

```
CONSTRAINT SQL_LANGUAGES_STANDARD_VALID_CHECK_ISO_1999
  CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
        OR
          ( SQL_LANGUAGE_YEAR <> '1999'
          OR
            ( SQL_LANGUAGE_BINDING_STYLE = 'DIRECT'
            AND
              SQL_LANGUAGE_PROGRAMMING_LANGUAGE IS NULL )
          OR
            ( SQL_LANGUAGE_BINDING_STYLE IN
              ( 'EMBEDDED', 'MODULE' )
            AND
              SQL_LANGUAGE_PROGRAMMING_LANGUAGE IN
              ( 'ADA', 'C', 'COBOL', 'FORTRAN',
                'MUMPS', 'PASCAL', 'PLI' ) ) ) )
```

Description

No additional Descriptions.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

--`,,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

20 Status codes

20.1 SQLSTATE

Table 10—SQLSTATE class and subclass values

Category	Condition	Class	Subcondition	Subclass
	All alternatives from ISO/IEC 9075-2			
X	attempt to assign to non-updatable column	0U	(*no subclass*)	000
X	attempt to assign to ordering column	0V	(*no subclass*)	000
X	dynamic SQL error	07	(*no subclass*)	000
			cursor specification cannot be executed	003
			invalid descriptor count	008
			invalid descriptor index	009
			prepared statement not a cursor specification	005
			restricted data type attribute violation	006
			undefined DATA value	00C
			undefined DATA target	00D
			undefined LEVEL value	00E
			undefined DATETIME_INTERVAL_CODE	00F
			using clause does not match dynamic parameter specifications	001
			using clause does not match target specifications	002
			using clause required for dynamic parameters	004
			using clause required for result fields	007
			data type transform function violation	00B
X	invalid SQL-invoked procedure reference	0M	(*no subclass*)	000
X	invalid character set name	2C	(*no subclass*)	000

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 10—SQLSTATE class and subclass values (Cont.)

Category	Condition	Class	Subcondition	Subclass
X	invalid schema name list specification	0E	(*no subclass*)	000
X	invalid transform group name specification	0S	(*no subclass*)	000
X	target table disagrees with cursor specification	0T	(*no subclass*)	000
W	warning	01	(*no subclass*)	000
			insufficient item descriptor areas	005

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

21 Conformance

21.1 General conformance requirements

Insert this paragraph The list of features that are implied by other features is shown in Table 11, “Implied feature relationships”.

Table 11—Implied feature relationships

Feature ID	Feature Description	Implied Feature ID	Implied Feature Description
B032	Extended dynamic SQL	B031	Basic dynamic SQL

21.2 Claims of conformance

Insert this paragraph In addition to the requirements of Subclause 8.1.5, "Claims of conformance", in ISO/IEC 9075-1, a claim of conformance to this part of ISO/IEC 9075 shall state:

1) Whether direct invocation (<direct SQL statement>) is supported.

2) Whether embedded SQL is supported and, if so, for which of the following programming languages:

   a) Ada

   b) C

   c) COBOL

   d) Fortran

   e) MUMPS

   f) Pascal

   g) PL/I

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

--`,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex A
(informative)

SQL Conformance Summary

Replicated paragraph The contents of this Annex summarizes all Conformance Rules, ordered by Feature ID and by Subclause.

1) Specifications for Feature B011, "Embedded Ada":

   a) Subclause 16.3, "<embedded SQL Ada program>":

      i) Without Feature B011, "Embedded Ada", <embedded SQL Ada program> shall not be specified.

2) Specifications for Feature B012, "Embedded C":

   a) Subclause 16.4, "<embedded SQL C program>":

      i) Without Feature B012, "Embedded C", <embedded SQL C program> shall not be specified.

3) Specifications for Feature B013, "Embedded COBOL":

   a) Subclause 16.5, "<embedded SQL COBOL program>":

      i) Without Feature B013, "Embedded COBOL", <embedded SQL COBOL program> shall not be specified.

4) Specifications for Feature B014, "Embedded Fortran":

   a) Subclause 16.6, "<embedded SQL Fortran program>":

      i) Without Feature B014, "Embedded Fortran", <embedded SQL Fortran program> shall not be specified.

5) Specifications for Feature B015, "Embedded MUMPS":

   a) Subclause 16.7, "<embedded SQL MUMPS program>":

      i) Without Feature B015, "Embedded MUMPS", <embedded SQL MUMPS program> shall not be specified.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6) Specifications for Feature B016, "Embedded Pascal":

   a) Subclause 16.8, "<embedded SQL Pascal program>":

      i) Without Feature B016, "Embedded Pascal", <embedded SQL Pascal program> shall not be specified.

7) Specifications for Feature B017, "Embedded PL/I":

   a) Subclause 16.9, "<embedded SQL PL/I program>":

      i) Without Feature B017, "Embedded PL/I", <embedded SQL PL/I program> shall not be specified.

8) Specifications for Feature B021, "Direct SQL":

   a) Subclause 17.1, "<direct SQL statement>":

      i) Without Feature B021, "Direct SQL", <direct SQL statement> shall not be specified.

9) Specifications for Feature B031, "Basic dynamic SQL":

   a) Subclause 5.3, "Names and identifiers":

      i) Insert this CR Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <SQL statement name>, <dynamic cursor name>, or <descriptor name>.

   b) Subclause 6.1, "<value specification> and <target specification>":

      i) Insert this CR Without Feature B031, "Basic dynamic SQL", a <general value specification> shall not be a <dynamic parameter specification>.

   c) Subclause 11.1, "<SQL-client module definition>":

      i) Insert this CR Without Feature B031, "Basic dynamic SQL", a <module contents> shall not be a <dynamic declare cursor>.

   d) Subclause 11.3, "<SQL procedure statement>":

      i) Insert this CR Without Feature B031, "Basic dynamic SQL", an <SQL procedure statement> shall not be an <SQL dynamic statement>.

   e) Subclause 15.2, "<allocate descriptor statement>":

      i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <allocate descriptor statement>.

   f) Subclause 15.3, "<deallocate descriptor statement>":

      i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <deallocate descriptor statement>.

   g) Subclause 15.4, "<get descriptor statement>":

      i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <get descriptor statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

ii) Without Feature T301, "Functional dependencies", and without Feature B031, "Basic dynamic SQL", conforming SQL language shall not specify KEY_MEMBER.

h) Subclause 15.5, "<set descriptor statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <set descriptor statement>.

i) Subclause 15.6, "<prepare statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <prepare statement>.

j) Subclause 15.8, "<describe statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <describe output statement>.

k) Subclause 15.9, "<input using clause>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <using input clause>.

l) Subclause 15.10, "<output using clause>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <output using clause>.

m) Subclause 15.11, "<execute statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <execute statement>.

n) Subclause 15.12, "<execute immediate statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <execute immediate statement>.

o) Subclause 15.13, "<dynamic declare cursor>":

i) Without Feature B031, "Basic dynamic SQL", and Feature F791, "Insensitive cursors", a <dynamic declare cursor> shall not specify INSENSITIVE.

ii) Without Feature B031, "Basic dynamic SQL", and Feature F421, "Scrolled cursors", a <dynamic declare cursor> shall not specify <cursor scrollability>.

iii) Without Feature B031, "Basic dynamic SQL", and Feature F831, "Full cursor update", if an <updatability clause> of FOR UPDATE with or without a <column name list> is specified, then ORDER BY shall not be specified.

iv) Without Feature B031, "Basic dynamic SQL", and Feature F341, "Scrolled cursors", and Feature F831, "Full cursor update", if an <updatability clause> of FOR UPDATE with or without a <column name list> is specified, then <cursor scrollability> shall not be specified.

v) Without Feature B031, "Basic dynamic SQL", and Feature T471, "Result sets return value", a <dynamic declare cursor> shall not specify <cursor returnability>.

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

vi) Without Feature B031, "Basic dynamic SQL", and either Feature F791, "Insensitive cursors", or Feature T231, "SENSITIVE cursors", a <dynamic declare cursor> shall not specify ASENSITIVE.

vii) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <dynamic declare cursor>.

viii) Without Feature B031, "Basic dynamic SQL", and Feature T231, "SENSITIVE cursors", a <dynamic declare cursor> shall not specify SENSITIVE.

p) Subclause 15.15, "<dynamic open statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <dynamic open statement>.

q) Subclause 15.16, "<dynamic fetch statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <dynamic fetch statement>.

r) Subclause 15.17, "<dynamic single row select statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall contain no <dynamic single row select statement>.

s) Subclause 15.18, "<dynamic close statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <dynamic close statement>.

t) Subclause 15.19, "<dynamic delete statement: positioned>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <dynamic delete statement: positioned>.

u) Subclause 15.20, "<dynamic update statement: positioned>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain any <dynamic update statement: positioned>.

10) Specifications for Feature B032, "Extended dynamic SQL":

a) Subclause 5.3, "Names and identifiers":

i) Insert this CR Without Feature B032, "Extended dynamic SQL", conforming SQL language shall not contain any <extended statement name> or <extended cursor name>.

b) Subclause 15.2, "<allocate descriptor statement>":

i) Without Feature B032, "Extended dynamic SQL", an <occurrences> and a <descriptor name> shall be a <literal>.

c) Subclause 15.3, "<deallocate descriptor statement>":

i) Without Feature B032, "Extended dynamic SQL", a <descriptor name> shall be a <literal>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

d) Subclause 15.4, "<get descriptor statement>":

i) Without Feature B032, "Extended dynamic SQL", a <descriptor name> shall be a <literal>.

e) Subclause 15.5, "<set descriptor statement>":

i) Without Feature B032, "Extended dynamic SQL", a <descriptor name> shall be a <literal>.

f) Subclause 15.7, "<deallocate prepared statement>":

i) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall contain no <deallocate prepared statement>.

g) Subclause 15.8, "<describe statement>":

i) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall not contain any <describe input statement>.

h) Subclause 15.9, "<input using clause>":

i) Without Feature B032, "Extended dynamic SQL", a <descriptor name> shall be a <literal>.

i) Subclause 15.10, "<output using clause>":

i) Without Feature B032, "Extended dynamic SQL", a <descriptor name> shall be a <literal>.

j) Subclause 15.11, "<execute statement>":

i) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall contain no <result using clause>.

k) Subclause 15.14, "<allocate cursor statement>":

i) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall not contain any <allocate cursor statement>.

l) Subclause 15.21, "<preparable dynamic delete statement: positioned>":

i) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall contain no <preparable dynamic delete statement: positioned>.

m) Subclause 15.22, "<preparable dynamic update statement: positioned>":

i) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall contain no <preparable dynamic update statement: positioned>.

11) Specifications for Feature B041, "Extensions to embedded SQL exception declarations":

a) Subclause 16.2, "<embedded exception declaration>":

i) Without Feature B041, "Extensions to embedded SQL exception declarations", a <major category> shall not specify SQLEXCEPTION or SQLWARNING.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ii) Without Feature B041, "Extensions to embedded SQL exception declarations", an <SQL condition> shall not specify SQLSTATE or CONSTRAINT.

12) Specifications for Feature B051, "Enhanced execution rights":

a) Subclause 11.1, "<SQL-client module definition>":

i) Insert this CR Without Feature B051, "Enhanced execution rights", conforming SQL language shall not specify FOR STATIC ONLY or FOR STATIC AND DYNAMIC.

b) Subclause 16.1, "<embedded SQL host program>":

i) Without Feature B051, "Enhanced execution rights", conforming SQL language shall not contain any <embedded authorization declaration>.

13) Specifications for Feature F341, "Scrolled cursors":

a) Subclause 15.13, "<dynamic declare cursor>":

i) Without Feature B031, "Basic dynamic SQL", and Feature F341, "Scrolled cursors", and Feature F831, "Full cursor update", if an <updatability clause> of FOR UPDATE with or without a <column name list> is specified, then <cursor scrollability> shall not be specified.

14) Specifications for Feature F361, "Subprogram support":

a) Subclause 16.1, "<embedded SQL host program>":

i) Without Feature F361, "Subprogram support", no two <host variable definition>s shall specify the same variable name.

15) Specifications for Feature F421, "Scrolled cursors":

a) Subclause 15.13, "<dynamic declare cursor>":

i) Without Feature B031, "Basic dynamic SQL", and Feature F421, "Scrolled cursors", a <dynamic declare cursor> shall not specify <cursor scrollability>.

16) Specifications for Feature F451, "Character set definition":

a) Subclause 14.3, "<set names statement>":

i) Without Feature F761, "Session management", and either Feature F451, "Character set definition", or Feature F461, "Named character sets", conforming SQL language shall not contain any <set names statement>.

b) Subclause 16.1, "<embedded SQL host program>":

i) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", an <embedded SQL declare section> shall not contain an <embedded character set declaration>.

c) Subclause 16.3, "<embedded SQL Ada program>":

i) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", an <Ada qualified type specification> shall not contain a <character set specification>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

d) Subclause 16.4, "<embedded SQL C program>":

i) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", a <C variable definition> shall not contain a <character set specification>.

e) Subclause 16.5, "<embedded SQL COBOL program>":

i) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", a <COBOL character type> shall not contain a <character set specification>.

f) Subclause 16.6, "<embedded SQL Fortran program>":

i) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", a <Fortran type specification> shall not contain a <character set specification>.

g) Subclause 16.8, "<embedded SQL Pascal program>":

i) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", a <Pascal type specification> shall not contain a <character set specification>.

h) Subclause 16.9, "<embedded SQL PL/I program>":

i) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", a <PL/I type specification> shall not contain a <character set specification>.

17) Specifications for Feature F461, "Named character sets":

a) Subclause 16.1, "<embedded SQL host program>":

i) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", an <embedded SQL declare section> shall not contain an <embedded character set declaration>.

b) Subclause 16.3, "<embedded SQL Ada program>":

i) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", an <Ada qualified type specification> shall not contain a <character set specification>.

c) Subclause 16.4, "<embedded SQL C program>":

i) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", a <C variable definition> shall not contain a <character set specification>.

d) Subclause 16.5, "<embedded SQL COBOL program>":

i) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", a <COBOL character type> shall not contain a <character set specification>.

e) Subclause 16.6, "<embedded SQL Fortran program>":

i) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", a <Fortran type specification> shall not contain a <character set specification>.

f) Subclause 16.8, "<embedded SQL Pascal program>":

i) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", a <Pascal type specification> shall not contain a <character set specification>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

g) Subclause 16.9, "<embedded SQL PL/I program>":

i) Without Feature F451, "Character set definition", or Feature F461, "Named character sets", a <PL/I type specification> shall not contain a <character set specification>.

18) Specifications for Feature F511, "BIT data type":

a) Subclause 16.3, "<embedded SQL Ada program>":

i) Without Feature F511, "BIT data type", an <Ada variable definition> shall not specify a bit string variable.

b) Subclause 16.4, "<embedded SQL C program>":

i) Without Feature F511, "BIT data type", a <C derived variable> shall not be a <C bit variable>.

c) Subclause 16.5, "<embedded SQL COBOL program>":

i) Without Feature F511, "BIT data type", a <COBOL type specification> shall not be a <COBOL bit type>.

d) Subclause 16.6, "<embedded SQL Fortran program>":

i) Without Feature F511, "BIT data type", a <Fortran type specification> shall not specify BIT.

e) Subclause 16.8, "<embedded SQL Pascal program>":

i) Without Feature F511, "BIT data type", a <Pascal type specification> shall not specify BIT or PACKED ARRAY [1..<length>] OF BIT.

f) Subclause 16.9, "<embedded SQL PL/I program>":

i) Without Feature F511, "BIT data type", a <PL/I type specification> shall not specify BIT or BIT VARYING.

19) Specifications for Feature F611, "Indicator data types":

a) Subclause 6.1, "<value specification> and <target specification>":

i) Insert this CR Without Feature F611, "Indicator data types", the specific declared types of <indicator parameter>s and <indicator variable>s shall be the same implementation-defined data type.

20) Specifications for Feature F761, "Session management":

a) Subclause 14.1, "<set catalog statement>":

i) Without Feature F761, "Session management", and Feature F651, "Catalog name qualifiers", conforming SQL language shall not contain any <set catalog statement>.

b) Subclause 14.2, "<set schema statement>":

i) Without Feature F761, "Session management", conforming SQL language shall not contain any <set schema statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c) Subclause 14.3, "<set names statement>":

i) Without Feature F761, "Session management", and either Feature F451, "Character set definition", or Feature F461, "Named character sets", conforming SQL language shall not contain any <set names statement>.

21) Specifications for Feature F791, "Insensitive cursors":

a) Subclause 15.13, "<dynamic declare cursor>":

i) Without Feature B031, "Basic dynamic SQL", and either Feature F791, "Insensitive cursors", or Feature T231, "SENSITIVE cursors", a <dynamic declare cursor> shall not specify ASENSITIVE.

ii) Without Feature B031, "Basic dynamic SQL", and Feature F791, "Insensitive cursors", a <dynamic declare cursor> shall not specify INSENSITIVE.

22) Specifications for Feature F831, "Full cursor update":

a) Subclause 15.13, "<dynamic declare cursor>":

i) Without Feature B031, "Basic dynamic SQL", and Feature F831, "Full cursor update", if an <updatability clause> of FOR UPDATE with or without a <column name list> is specified, then ORDER BY shall not be specified.

23) Specifications for Feature S024, "Enhanced structured types":

a) Subclause 15.13, "<dynamic declare cursor>":

i) Without Feature S024, "Enhanced structured types", a <value expression> that is a <sort key> shall not be of a structured type.

24) Specifications for Feature S041, "Basic reference types":

a) Subclause 16.3, "<embedded SQL Ada program>":

i) Without Feature S041, "Basic reference types", an <Ada derived type specification> shall not be an <Ada REF variable>.

b) Subclause 16.4, "<embedded SQL C program>":

i) Without Feature S041, "Basic reference types", a <C derived variable> shall not be a <C REF variable>.

c) Subclause 16.5, "<embedded SQL COBOL program>":

i) Without Feature S041, "Basic reference types", a <COBOL type specification> shall not be a <COBOL REF variable>.

d) Subclause 16.6, "<embedded SQL Fortran program>":

i) Without Feature S041, "Basic reference types", a <Fortran derived type specification> shall not be a <Fortran REF variable>.

e) Subclause 16.7, "<embedded SQL MUMPS program>":

i) Without Feature S041, "Basic reference types", a <MUMPS derived type specification> shall not be a <MUMPS REF variable>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

f) Subclause 16.8, "<embedded SQL Pascal program>":

i) Without Feature S041, "Basic reference types", a <Pascal derived type specification> shall not be a <Pascal REF variable>.

g) Subclause 16.9, "<embedded SQL PL/I program>":

i) Without Feature S041, "Basic reference types", a <PL/I derived type specification> shall not be a <PL/I REF variable>.

25) Specifications for Feature S071, "SQL paths in function and type name resolution":

a) Subclause 14.4, "<set path statement>":

i) Without Feature S071, "SQL paths in function and type name resolution", Conforming SQL language shall not contain any <set path statement>.

b) Subclause 16.1, "<embedded SQL host program>":

i) Without Feature S071, "SQL paths in function and type name resolution", <embedded path specification> shall not be specified.

26) Specifications for Feature S231, "Structured type locators":

a) Subclause 16.3, "<embedded SQL Ada program>":

i) Without Feature S231, "Structured type locators", the <user-defined type name> simply contained in an <Ada user-defined type locator variable> shall not identify a structured type.

b) Subclause 16.4, "<embedded SQL C program>":

i) Without Feature S231, "Structured type locators", the <user-defined type name> simply contained in a <C user-defined type locator variable> shall not identify a structured type.

c) Subclause 16.5, "<embedded SQL COBOL program>":

i) Without Feature S231, "Structured type locators", the <user-defined type name> simply contained in a <COBOL user-defined type locator variable> shall not identify a structured type.

d) Subclause 16.6, "<embedded SQL Fortran program>":

i) Without Feature S231, "Structured type locators", the <user-defined type name> simply contained in a <Fortran user-defined type locator variable> shall not identify a structured type.

e) Subclause 16.7, "<embedded SQL MUMPS program>":

i) Without Feature S231, "Structured type locators", the <user-defined type name> simply contained in a <MUMPS user-defined type locator variable> shall not identify a structured type.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

f) Subclause 16.8, "<embedded SQL Pascal program>":

i) Without Feature S231, "Structured type locators", the <user-defined type name> simply contained in a <Pascal user-defined type locator variable> shall not identify a structured type.

g) Subclause 16.9, "<embedded SQL PL/I program>":

i) Without Feature S231, "Structured type locators", the <user-defined type name> simply contained in a <PL/I user-defined type locator variable> shall not identify a structured type.

27) Specifications for Feature S232, "Array locators":

a) Subclause 16.3, "<embedded SQL Ada program>":

i) Without Feature S232, "Array locators", an <Ada derived type specification> shall not be an <Ada array locator variable>.

b) Subclause 16.4, "<embedded SQL C program>":

i) Without Feature S232, "Array locators", a <C derived variable> shall not be a <C array locator variable>.

c) Subclause 16.5, "<embedded SQL COBOL program>":

i) Without Feature S232, "Array locators", a <COBOL derived type specification> shall not be a <COBOL array locator variable>.

d) Subclause 16.6, "<embedded SQL Fortran program>":

i) Without Feature S232, "Array locators", a <Fortran derived type specification> shall not be a <Fortran array locator variable>.

e) Subclause 16.7, "<embedded SQL MUMPS program>":

i) Without Feature S232, "Array locators", a <MUMPS derived type specification> shall not be a <MUMPS array locator variable>.

f) Subclause 16.8, "<embedded SQL Pascal program>":

i) Without Feature S232, "Array locators", a <Pascal derived type specification> shall not be a <Pascal array locator variable>.

g) Subclause 16.9, "<embedded SQL PL/I program>":

i) Without Feature S232, "Array locators", a <PL/I derived type specification> shall not be a <PL/I array locator variable>.

28) Specifications for Feature S241, "Transform functions":

a) Subclause 14.5, "<set transform group statement>":

i) Without Feature S241, "Transform functions", conforming SQL language shall not contain any <set transform group statement>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) Subclause 16.1, "<embedded SQL host program>":

i) Without Feature S241, "Transform functions", <embedded transform group specification> shall not be specified.

c) Subclause 16.3, "<embedded SQL Ada program>":

i) Without Feature S241, "Transform functions", an <Ada derived type specification> shall not be an <Ada user-defined type variable>.

d) Subclause 16.4, "<embedded SQL C program>":

i) Without Feature S241, "Transform functions", a <C derived type specification> shall not be a <C user-defined type variable>.

e) Subclause 16.5, "<embedded SQL COBOL program>":

i) Without Feature S241, "Transform functions", a <COBOL derived type specification> shall not be a <COBOL user-defined type variable>.

f) Subclause 16.6, "<embedded SQL Fortran program>":

i) Without Feature S241, "Transform functions", a <Fortran derived type specification> shall not be a <Fortran user-defined type variable>.

g) Subclause 16.7, "<embedded SQL MUMPS program>":

i) Without Feature S241, "Transform functions", a <MUMPS derived type specification> shall not be a <MUMPS user-defined type variable>.

h) Subclause 16.8, "<embedded SQL Pascal program>":

i) Without Feature S241, "Transform functions", a <Pascal derived type specification> shall not be a <Pascal user-defined type variable>.

i) Subclause 16.9, "<embedded SQL PL/I program>":

i) Without Feature S241, "Transform functions", a <PL/I derived type specification> shall not be a <PL/I user-defined type variable>.

29) Specifications for Feature T041, "Basic LOB data type support":

a) Subclause 16.3, "<embedded SQL Ada program>":

i) Without Feature T041, "Basic LOB data type support", an <Ada BLOB variable>, <Ada CLOB variable>, <Ada BLOB locator variable>, and <Ada CLOB locator variable> shall not be specified.

b) Subclause 16.4, "<embedded SQL C program>":

i) Without Feature T041, "Basic LOB data type support", a <C BLOB variable>, <C CLOB variable>, <C BLOB locator variable>, and <C CLOB locator variable> shall not be specified.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c) Subclause 16.5, "<embedded SQL COBOL program>":

i) Without Feature T041, "Basic LOB data type support", a <COBOL BLOB variable>, <COBOL CLOB variable>, <COBOL BLOB locator variable>, and <COBOL CLOB locator variable> shall not be specified.

d) Subclause 16.6, "<embedded SQL Fortran program>":

i) Without Feature T041, "Basic LOB data type support", a <Fortran BLOB variable>, <Fortran CLOB variable>, <Fortran BLOB locator variable>, and <Fortran CLOB locator variable> shall not be specified.

e) Subclause 16.7, "<embedded SQL MUMPS program>":

i) Without Feature T041, "Basic LOB data type support", a <MUMPS BLOB variable>, <MUMPS CLOB variable>, <MUMPS BLOB locator variable>, and <MUMPS CLOB locator variable> shall not be specified.

f) Subclause 16.8, "<embedded SQL Pascal program>":

i) Without Feature T041, "Basic LOB data type support", a <Pascal BLOB variable>, <Pascal CLOB variable>, <Pascal BLOB locator variable>, and <Pascal CLOB locator variable> shall not be specified.

g) Subclause 16.9, "<embedded SQL PL/I program>":

i) Without Feature T041, "Basic LOB data type support", a <PL/I BLOB variable>, <PL/I CLOB variable>, <PL/I BLOB locator variable>, and <PL/I CLOB locator variable> shall not be specified.

30) Specifications for Feature T231, "SENSITIVE cursors":

a) Subclause 15.13, "<dynamic declare cursor>":

i) Without Feature B031, "Basic dynamic SQL", and either Feature F791, "Insensitive cursors", or Feature T231, "SENSITIVE cursors", a <dynamic declare cursor> shall not specify ASENSITIVE.

ii) Without Feature B031, "Basic dynamic SQL", and Feature T231, "SENSITIVE cursors", a <dynamic declare cursor> shall not specify SENSITIVE.

31) Specifications for Feature T301, "Functional dependencies":

a) Subclause 15.4, "<get descriptor statement>":

i) Without Feature T301, "Functional dependencies", and without Feature B031, "Basic dynamic SQL", conforming SQL language shall not specify KEY_MEMBER.

32) Specifications for Feature T471, "Result sets return value":

a) Subclause 15.13, "<dynamic declare cursor>":

i) Without Feature B031, "Basic dynamic SQL", and Feature T471, "Result sets return value", a <dynamic declare cursor> shall not specify <cursor returnability>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

33) Specifications for Feature T551, "Optional key words for default syntax":

a) Subclause 15.13, "<dynamic declare cursor>":

i) Without Feature T551, "Optional key words for default syntax", a <dynamic declare cursor> shall not specify WITHOUT HOLD.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex B
(informative)

Implementation-defined elements

Insert this paragraph This Annex references those features that are identified in the body of this part of ISO/IEC 9075 as implementation-defined.

1) Subclause 4.1, "Catalogs": The default catalog name substitution value for execution of <preparable statement>s that are dynamically prepared in the current SQL-session through the execution of <prepare statement>s and <execute immediate statement>s are implementation-defined.

2) Subclause 4.8, "Embedded syntax": Whether a portion of the name space is reserved by an implementation for the names of procedures, subroutines, program variables, branch labels, <SQL-client module definition>s, or <externally-invoked procedure>s is implementation-defined; if a portion of the name space is so reserved, the portion reserved is also implementation-defined.

3) Subclause 4.9, "SQL dynamic statements": Within an SQL-session, all prepared statements belong to the same implementation-defined <SQL-client module definition> that is different from any other <SQL-client module definition> that exists simultaneously in the environment.

4) Subclause 4.10, "Direct invocation of SQL": The method of invoking <direct SQL statement>s, the method of raising conditions as a result of <direct SQL statement>s, the method of accessing diagnostic information, and the method of returning the results are all implementation-defined.

5) Subclause 4.12, "SQL-transactions": It is implementation-defined whether or not the dynamic execution of an <SQL dynamic data statement> is permitted to occur within the same SQL-transaction as the dynamic execution of an SQL-schema statement. If it does occur, then the effect on any prepared dynamic statement is also implementation-defined.

6) Subclause 4.14, "SQL-sessions": When an SQL-session is initiated, there is an implementation-defined default catalog whose name is used to effectively qualify all unqualified <schema name>s contained in <preparable statement>s that are dynamically prepared in the current SQL-session through the execution of <prepare statement>s and <execute immediate statement>s.

7) Subclause 4.14, "SQL-sessions": When an SQL-session is initiated, there is an implementation-defined default schema whose name is used to effectively qualify all unqualified <schema qualified name>s contained in <preparable statement>s that are dynamically prepared in the current SQL-session through the execution of <prepare statement>s and <execute immediate statement>s.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

8) Subclause 4.14, "SQL-sessions": The value of the current SQL-path before a successful execution of <set path statement> is implementation-defined.

9) Subclause 5.3, "Names and identifiers": If a <schema name> contained in a <preparable statement> that is dynamically prepared in the current SQL-session through the execution of a <prepare statement> or an <execute immediate statement> does not contain a <catalog name>, then the implementation-defined <catalog name> for the SQL-session is implicit.

10) Subclause 5.3, "Names and identifiers": If a <schema qualified name> contained in a <preparable statement> that is dynamically prepared in the current SQL-session through the execution of a <prepare statement> or an <execute immediate statement> does not contain a <schema name>, then the implementation-defined <schema name> for the SQL-session is implicit.

11) Subclause 6.1, "<value specification> and <target specification>": In Intermediate SQL, the specific data type of <indicator parameter>s and <indicator variable>s shall be the same implementation-defined data type.

12) Subclause 15.2, "<allocate descriptor statement>": If WITH MAX <occurrences> is not specified, then an implementation-defined default value for <occurrences> that is greater than 0 (zero) is implicit.

13) Subclause 15.2, "<allocate descriptor statement>": The maximum number of SQL descriptor areas and the maximum number of item descriptor areas for a single SQL descriptor area are implementation-defined.

14) Subclause 15.5, "<set descriptor statement>": Restrictions on changing TYPE, LENGTH, OCTET_LENGTH, SCALE, COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME values resulting from the execution of a <describe statement> before execution of an <execute statement>, <dynamic open statement>, or <dynamic fetch statement> are implementation-defined.

15) Subclause 15.6, "<prepare statement>": The Format and Syntax Rules for a <preparable implementation-defined statement> are implementation-defined.

16) Subclause 15.8, "<describe statement>":

    a) The character set of the data type of <descriptor name> is implementation-defined.

    b) If *SR* does not contain a single routine SQL-invoked *R*, then the values of PARAMETER_MODE, PARAMETER_ORDINAL_POSITION, PARAMETER_SPECIFIC_CATALOG, PARAMETER_SPECIFIC_SCHEMA, and PARAMETER_SPECIFIC_NAME in the descriptor for each <dynamic parameter specification> simply contained in the <call statement> are set to implementation-defined values.

17) Subclause 16.1, "<embedded SQL host program>":

    a) If *H* does not contain an <embedded authorization declaration> that specifies SCHEMA, then the <schema name> of the <module authorization clause> of *M* is implementation-defined.

    b) If *H* does not contain an <embedded authorization declaration>, then *M* contains a <module authorization clause> that specifies "SCHEMA *SN*", where *SN* is an implementation-defined <schema name>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c) If an <embedded character set declaration> is not specified, then an <embedded character set declaration> containing an implementation-defined <character set specification> is implicit.

d) Each <allocate cursor statement> is replaced with a host language procedure or subroutine call of an implementation-defined procedure that associates the <dynamic cursor name> with the prepared statement.

e) If an <embedded SQL host program> does not contain an <embedded path specification>, then the implied module contains an implementation-defined <module path specification>.

18) Subclause 16.5, "<embedded SQL COBOL program>": The COBOL data description clauses, in addition to the PICTURE, SIGN, USAGE and VALUE clauses, that may appear in a <COBOL variable definition> are implementation-defined.

19) Subclause 16.9, "<embedded SQL PL/I program>": The PL/I data description clauses, in addition to the <PL/I type specification> and the INITIAL clause, that may appear in a <PL/I variable definition> are implementation-defined.

20) Subclause 17.1, "<direct SQL statement>": The <value specification> that represents the null value is implementation-defined.

21) Subclause 17.1, "<direct SQL statement>": The Format, Syntax Rules, and Access Rules for <direct implementation-defined statement> are implementation-defined.

22) Subclause 17.1, "<direct SQL statement>": Whether a <direct implementation-defined statement> may be associated with an active transaction is implementation-defined.

23) Subclause 17.1, "<direct SQL statement>": Whether a <direct implementation-defined statement> initiates a transaction is implementation-defined.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Annex C
(informative)

Implementation-dependent elements

Insert this paragraph This Annex references those places where this part of ISO/IEC 9075 states explicitly that the actions of a conforming implementation are implementation-dependent.

1) Subclause 15.4, "<get descriptor statement>":

   a) If an exception condition is raised in a <get descriptor statement>, then the values of all targets specified by <simple target specification 1>, <simple target specification 2>, and <simple target specification 3> are implementation-dependent.

   b) For a <dynamic parameter specification>, the value of UNNAMED is 1 (one) and the value of NAME is implementation-dependent.

   c) The value retrieved by a <get descriptor statement> for any field whose value is undefined is implementation-dependent.

2) Subclause 15.5, "<set descriptor statement>": If an exception condition is raised in a <set descriptor statement>, then the values of all elements of the descriptor specified in the <set descriptor statement> are implementation-dependent.

3) Subclause 15.6, "<prepare statement>": The validity of an <extended statement name> value or a <statement name> in an SQL-transaction different from the one in which the statement was prepared is implementation-dependent.

4) Subclause 15.9, "<input using clause>": When a <describe input statement> is used, the values for NAME, DATA, and INDICATOR in the SQL dynamic descriptor area structure is implementation-dependent. If TYPE indicates a character string type, a bit string type, or a binary large object type, then the values of SCALE and PRECISION are implementation-dependent. If TYPE indicates an exact or approximate numeric type, then the values of LENGTH and OCTET_LENGTH are implementation-dependent. If TYPE indicates a boolean type, then the values of PRECISION, SCALE, LENGTH, and OCTET_LENGTH are implementation-dependent.

5) Subclause 15.10, "<output using clause>": When a <describe output statement> is executed, the values of DATA and INDICATOR are implementation-dependent. If TYPE indicates a character string type, a bit string type, or a binary large object type, then the values of SCALE and PRECISION are implementation-dependent. If TYPE indicates an exact or approximate numeric type, then the values of LENGTH and OCTET_LENGTH are implementation-dependent. If TYPE indicates a boolean type, then the values of PRECISION, SCALE, LENGTH, and OCTET_LENGTH are implementation-dependent.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

6) Subclause 16.1, "<embedded SQL host program>": The <module name> of the implied <SQL-client module definition> derived from an <embedded SQL host program> is implementation-dependent.

7) Subclause 16.1, "<embedded SQL host program>": If an <embedded SQL host program> does not contain an <embedded authorization statement>, then the <module authorization identifier> of the implied <SQL-client module definition> derived from the <embedded SQL host program> is implementation-dependent.

8) Subclause 16.1, "<embedded SQL host program>": In each <declare cursor> in the implied <SQL-client module definition> derived from an <embedded SQL host program>, each <embedded variable name> has been replaced consistently with a distinct <host parameter name> that is implementation-dependent.

9) Subclause 16.1, "<embedded SQL host program>": The <procedure name> of each <externally-invoked procedure> in the implied <SQL-client module definition> derived from an <embedded SQL host program> is implementation-dependent.

10) Subclause 16.1, "<embedded SQL host program>": In each <externally-invoked procedure> in the implied <SQL-client module definition> derived from an <embedded SQL host program>, each <embedded variable name> has been replaced consistently with a distinct <host parameter name> that is implementation-dependent.

11) Subclause 16.1, "<embedded SQL host program>":

    a) For <SQL procedure statement>s other than <open statement>s, whether one <externally-invoked procedure> in the implied <SQL-client module definition> derived from an <embedded SQL host program> can correspond to more than one <SQL procedure statement> in the <embedded SQL host program> is implementation-dependent.

    b) In each <externally-invoked procedure> in the implied <SQL-client module definition> derived from an <embedded SQL host program>, the order of the instances of <host parameter declaration> is

    c) implementation-dependent.

12) Subclause 17.1, "<direct SQL statement>": A <commit statement> or a <rollback statement> is executed. If an unrecoverable error has occurred, or if the direct invocation of SQL terminated unexpectedly, or if any constraint is not satisfied, then a <rollback statement> is performed. Otherwise, the choice of which of these SQL-statements to perform is implementation-dependent. The determination of whether a direct invocation of SQL has terminated unexpectedly is implementation-dependent.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex D
(informative)

Deprecated features

Insert this paragraph It is intended that the following features will be removed at a later date from a revised version of this part of ISO/IEC 9075:

No additional deprecated items.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex E
(informative)

Incompatibilities with ISO/IEC 9075:1992

Insert this paragraph This edition of this part of ISO/IEC 9075 introduces some incompatibilities with the earlier version of Database Language SQL as specified in ISO/IEC 9075:1992.

Insert this paragraph Except as specified in this Annex, every feature and capability of Database Language SQL as specified in this edition of this part of ISO/IEC 9075 is compatible with ISO/IEC 9075:1992.

1) In ISO/IEC 9075:1992, if a <row value expression> in the <in value list> of an <in predicate> was a <dynamic parameter>, its data type was either:

   a) If the <row value expression> was not itself a <dynamic parameter specification>, then that of the <row value expression> immediately contained in the <in predicate>.

   b) Otherwise, the result of applying aggregation to the other <row value expression>s in the <in value list>.

   In ISO/IEC 9075:1999, this special case is eliminated, so that the data type of a <dynamic parameter specification>, regardless of its location, is determined by applying aggregation to all <row value expression>s (and the <table subquery>, if present).

   This allows <in predicate>s to be consistent with other predicates in the way they deal with subfields that are dynamic parameters.

2) A number of additional <reserved word>s have been added to the language. These <reserved word>s are:

   — DYNAMIC

3) In ISO/IEC 9075:1992, <describe input statement> described both input and output <dynamic parameter specification>s. In ISO/IEC 9075-5:1999, <describe input statement> only describes input <dynamic parameter specification>s.

4) The following features that appeared in ISO/IEC 9075:1992 have been deleted from this version of ISO/IEC 9075:

   a) <COBOL computational integer>s in <embedded SQL COBOL program>s.

   b) The use of SQLERROR as a <condition> in an <embedded exception declaration>.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex F
(informative)

SQL feature and package taxonomy

Insert this paragraph This Annex describes a taxonomy of features and packages defined in this part of ISO/IEC 9075.

Insert this paragraph Table 12, "SQL/Bindings feature taxonomy and definition for Core SQL", contains a taxonomy of the features of Core SQL language that are specified in this part of ISO/IEC 9075. Table 13, "SQL/Bindings feature taxonomy for features outside Core SQL", contains a taxonomy of the features of the SQL language not in Core SQL that are specified in this part of ISO/IEC 9075.

Insert this paragraph In these tables, the first column contains a counter that may be used to quickly locate rows of the table; these values otherwise have no use and are not stable — that is, they are subject to change in future editions of or even Technical Corrigenda to ISO/IEC 9075 without notice.

Insert this paragraph The "Feature ID" column of Table 12, "SQL/Bindings feature taxonomy and definition for Core SQL", and of Table 13, "SQL/Bindings feature taxonomy for features outside Core SQL", specifies the formal identification of each feature and each subfeature contained in the table. The Feature ID is stable and can be depended on to remain constant. A Feature ID value comprises either a letter and three digits or a letter, three digits, a hyphen, and one or two additional digits. Feature ID values containing a hyphen and additional digits indicate "subfeatures" that help to define complete features, which are in turn indicated by Feature ID values without a hyphen. Only entire features are used to specify the contents of Core SQL and various packages.

Insert this paragraph The "Feature Description" column of Table 12, "SQL/Bindings feature taxonomy and definition for Core SQL", and of Table 13, "SQL/Bindings feature taxonomy for features outside Core SQL", contains a brief description of the feature or subfeature associated with the Feature ID value.

Insert this paragraph Table 12, "SQL/Bindings feature taxonomy and definition for Core SQL", provides the only definition of the features comprising the minimal conformance possibility for ISO/IEC 9075, called Core SQL. The final column of this table, labeled "Implies", contains indications of specific language elements supported in each feature, subject to the constraints of all Syntax Rules, Access Rules, and Conformance Rules.

Insert this paragraph In Table 12, "SQL/Bindings feature taxonomy and definition for Core SQL", unless otherwise stated, a feature or subfeature assumes the support of underlying elements of ISO/IEC 9075.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Table 12—SQL/Bindings feature taxonomy and definition for Core SQL

	Feature ID	Feature Name	Feature Description
1	**B011**	**Embedded Ada**[1]	Subclause 16.3, "<embedded SQL Ada program>"
2	**B012**	**Embedded C**[1]	Subclause 16.4, "<embedded SQL C program>"
3	**B013**	**Embedded COBOL**[1]	Subclause 16.5, "<embedded SQL COBOL program>"
4	**B014**	**Embedded Fortran**[1]	Subclause 16.6, "<embedded SQL Fortran program>"
5	**B015**	**Embedded MUMPS**[1]	Subclause 16.7, "<embedded SQL MUMPS program>"
6	**B016**	**Embedded Pascal**[1]	Subclause 16.8, "<embedded SQL Pascal program>"
7	**B017**	**Embedded PL/I**[1]	Subclause 16.9, "<embedded SQL PL/I program>"

[1]A conforming SQL-implementation is required (by Clause 8, "Conformance", in ISO/IEC 9075-1) to support at least one embedded language.

Table 13—SQL/Bindings feature taxonomy for features outside Core SQL

	Feature ID	Feature Name
1	B021	Direct SQL
2	B031	Basic dynamic SQL
3	B032	Extended dynamic SQL
4	B032-01	<describe input> statement
5	B041	Extensions to embedded SQL exception declarations
6	B051	Enhanced execution rights

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Index

Index entries appearing in **boldface** indicate the page where the word, phrase, or BNF nonterminal was defined; index entries appearing in *italics* indicate a page where the BNF nonterminal was used in a Format; and index entries appearing in roman type indicate a page where the word, phrase, or BNF nonterminal was used in a heading, Function, Syntax Rule, Access Rule, General Rule, Leveling Rule, Table, or other descriptive text.

— A —

access mode • 189
active • 27, 189, 219
active SQL-transaction • 189
Ada • 3, 12, 23, 24, 57, 133, 134, 135, 137, 138, 143, 146, 149, 150, 151, 152, 153, 201, 203, 208, 209, 210, 211, 212, 213, 214, 228
ADA • 138, 197
<Ada array locator variable> • *149*, **150**, 152, 153, 213
<Ada assignment operator> • ***149***
<Ada BLOB locator variable> • *149*, **150**, 151, 153, 214
<Ada BLOB variable> • *149*, **150**, 151, 153, 214
<Ada CLOB locator variable> • *149*, **150**, 151, 153, 214
<Ada CLOB variable> • *149*, **150**, 151, 153, 214
<Ada derived type specification> • ***149***, 150, 153, 211, 213, 214
<Ada host identifier> • *134*, ***149***, 150, 151, 152
<Ada initial value> • ***149***, 152
<Ada qualified type specification> • ***149***, 150, 153, 208, 209
<Ada REF variable> • *149*, **150**, 152, 153, 211
<Ada type specification> • ***149***
<Ada unqualified type specification> • ***149***
<Ada user-defined type locator variable> • *149*, **150**, 152, 153, 212
<Ada user-defined type variable> • *149*, **150**, 151, 153, 214
<Ada variable definition> • *134*, **149**, 150, 152, 153, 210
ALLOCATE • *80*, *121*
<allocate cursor statement> • 15, 16, 17, 18, 19, 21, 22, 24, 35, *59*, 99, 100, **121**, 122, 124, 131, 132, 194, 207, 219
<allocate descriptor statement> • 17, 18, 25, 35, *59*, **80**, 81, 85, 194, 204
ambiguous cursor name • 98
<ampersand> • *133*, 135
AND • *55*, 56, *133*, 197, 208
applicable from-sql function • 114
applicable privileges • 39
applicable to-sql function • 110
approximate numeric • 105, 152, 159, 170, 173, 179, 185, 221
<approximate numeric type> • 105
ARD • 31, 72, 74, 75, 76, 83, 84, 86, 87, 105, 115, 197
ARE • *134*
array • 29, 62, 72, 74, 94, 105, 113, 138, 140, 149, 150, 152, 153, 154, 155, 156, 158, 159, 160, 161, 162, 163, 164, 166, 167, 170, 171, 172, 173, 175, 176, 179, 180, 181, 182, 183, 184, 185, 213
ARRAY • 23, 72, 74, 77, 85, 87, 88, 94, 105, 108, 114, 115, *176*, 177, 178, *179*, 180, 210
<array element> • 94
<array element list> • 94
array locator variable • 138, 152, 153, 156, 158, 160, 163, 164, 166, 170, 171, 175, 179, 180, 184, 185, 213
array type • 29, 74, 105, 113, 140, 152, 158, 164, 170, 175, 179, 184
AS • 15, *40*, *109*, *113*, 138, *139*, 140, 142, *150*, *151*, *152*, *155*, *156*, *157*, *158*, *162*, *164*, *167*, *169*, *170*, *172*, *173*, *174*, *175*, *176*, *177*, *178*, *179*, *181*, *182*, *183*
ASC • 192
ASENSITIVE • 119, 206, 211, 215
assertion • 8, 51, 147
<assertion definition> • 51
assignable • 43, 45, 139, 141
assignment • 7, 45, 46, 61, 84, 87, 149
associated array data type • 156, 163
associated array type • 74, 140, 152, 158, 164, 170, 175, 179, 184
associated SQL data type • 141, 151, 157, 164, 169, 174, 178, 183
associated user-defined type • 74, 136, 139, 140, 141, 144, 151, 152, 157, 158, 164, 165, 169, 174, 178, 183
associated value • 108

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

associated with • 5, 24, 27, 29, 38, 55, 60, 76, 77, 78, 84, 99, 100, 101, 109, 114, 117, 119, 122, 123, 124, 127, 129, 188, 189, 194, 227
<asterisk> • 93, *167*, 169
ATOMIC • 142
attempt to assign to non-updatable column • 130, 199
attempt to assign to ordering column • 129, 199
attribute • 109, 110, 112, 113, 114, 199
ATTRIBUTES • 5, 75, 76, 77, 78, 193, 199, 201, 227, 228
AUTHORIZATION • 25, 26, *55*, *133*
<authorization identifier> • 39

— B —

based on • 3, 91, 121
base table • 39, 105, 197
BEGIN • *134*, 142
<between predicate> • 96
BINARY • 72, 73, 74, 88, 115, *162*, *164*, *165*, *182*, *183*, *184*, 185
binary large object • 29, 105, 112, 138, 140, 151, 158, 164, 169, 174, 178, 183, 221
binary large object locator variable • 138, 151, 158, 164, 169, 174, 178, 183
<binary large object string type> • 105
BIT • 72, 73, 77, 88, 92, 115, *149*, 152, 153, *155*, 156, 159, 160, *161*, *163*, 165, 166, *167*, 168, 170, 171, *176*, 177, 179, 180, *181*, 184, 185, 210
<bit length expression> • 92
bit string type • 105, 221
<bit string type> • 105
BIT_LENGTH • 159
BLOB • 77, 138, 140, *149*, *150*, *151*, 153, *154*, *155*, *157*, 158, 159, 160, *161*, *162*, *163*, *164*, 166, *167*, *169*, 171, *172*, *173*, *174*, 175, *176*, *177*, *178*, 180, *181*, *182*, *183*, 185, 214, 215
boolean • 94, 152, 170, 179, 221
BOOLEAN • 23, 72, 73, 77, 94, *149*, 152, 159, 165, 170, *176*, 179, 184
<boolean primary> • 94
BOTH • *65*, *66*, *67*, *80*, *99*, *121*
branch • 24, 217
BY • 191

— C —

C • 3, 23, 24, 105, *133*, *134*, 135, 137, 138, 143, 146, *154*, *155*, *156*, 157, 158, 159, 160, 197, 201, 203, 209, 210, 211, 212, 213, 214, 228
<call statement> • 20, 21, 78, 97, 98, 105, 218
candidate key • 103, 104
cardinality • 47, 72, 74, 105, 115, 125
CARDINALITY • *31*, 72, 74, 75, 76, *83*, *84*, 86, 87, 105, 115
cardinality violation • 125
<C array locator variable> • *155*, **156**, 158
<C array specification> • ***154***, *155*, 156, 159
cascade off • 144
<case abbreviation> • 95
<case expression> • 47
<case operand> • 95
<case specification> • 95
CAST • *40*, *109*, *113*, *139*, 142
<cast operand> • 94
<cast specification> • 23, 94, 109, 113
<cast target> • 94
catalog • 6, 8, 15, 17, 26, 28, 29, 34, 59, 65, 66, 97, 106, 194, 210, 217, 218
CATALOG • *66*, 194
<catalog name> • 65, 66, 218
<catalog name characteristic> • ***65***, 97
<C bit variable> • *154*, **155**, 156, 159, 160, 210
<C BLOB locator variable> • ***155***, 158, 160, 214
<C BLOB variable> • *154*, **155**, 160, 214
<C character type> • ***154***, 156, 157
<C character variable> • ***154***, 156, 159
<C class modifier> • ***154***
<C CLOB locator variable> • ***155***, 158, 160, 214
<C CLOB variable> • *154*, **155**, 156, 159, 160, 214
<C derived variable> • ***154***, 160, 210, 211, 213
CHAR • *149*, 150, 152, *176*, 177, 179, *181*
character • 15, 23, 26, 29, 33, 35, 38, 45, 59, 62, 65, 66, 67, 68, 69, 72, 73, 75, 76, 78, 80, 88, 91, 92, 93, 97, 99, 104, 105, 113, 115, 118, 121, 134, 135, 136, 138, 140, 143, 144, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 176, 178, 179, 180, 181, 183, 184, 185, 193, 194, 199, 208, 209, 210, 211, 218, 219, 221
CHARACTER • 23, 72, 73, 74, 77, 88, 92, 93, 97, 115, *149*, *150*, 152, *154*, *155*, 156, 157, 159, *161*, 163, 165, *167*, 168, 170, *172*, 173, *176*, 177, 179, *181*, 182, 184
character large object • 29, 45, 62, 113, 138, 140, 151, 158, 164, 169, 174, 178, 183
character large object locator variable • 62, 151, 158, 164, 169, 174, 178, 183
<character match value> • 97
<character pattern> • 97
character repertoire • 159
<character representation> • *149*, 152, *156*, 158, *161*, 165, *181*, 184
character set • 15, 26, 29, 33, 35, 38, 67, 72, 73, 75, 76, 88, 92, 93, 97, 105, 136, 138, 143, 144, 152, 153, 157, 159, 160, 163, 165, 166, 168, 170, 171, 173, 179, 180, 184, 185, 199, 208, 209, 210, 211, 218, 219
<character set name> • 67, 105
<character set name characteristic> • ***67***, 97
<character set specification> • 33, 73, *134*, *149*, *150*, 152, 153, *154*, *155*, 159, 160, *161*, 165, 166, *167*, 170, 171, *172*, *176*, 179, 180, *181*, 184, 185, 208, 209, 210, 219
<character string literal> • 33, 67
<character string type> • 104
character type • 156, 157, 162, 163, 165, 166, 209
CHARACTER_SET_CATALOG • 72, 73, 75, *83*, 87, 88, 89, 105, 218

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

CHARACTER_SET_NAME • *57*, 72, 73, 75, *83*, 87, 88, 89, 105, 218
CHARACTER_SET_SCHEMA • 72, 73, 75, *83*, 87, 88, 89, 105, 218
<char length expression> • 92
CHECK • 197
check constraint • 49
<check constraint definition> • 49
<C host identifier> • *134*, ***154***, *155*, *156*, 157, 158, 159
<C initial value> • *154*, *155*, ***156***, 158
CLASS • *57*, *58*
CLOSE • *126*, 194
<close statement> • 19, 21, 22, 144
<C NCHAR variable> • *154*, **155**, 156, 157, 159
<C NCHAR VARYING variable> • *154*, **155**, 156, 157, 159
<C NCLOB variable> • *154*, **155**, 156, 157, 159
<C numeric variable> • ***154***
COBOL • 3, *12*, *13*, 23, 24, *133*, *134*, 135, 137, 138, 143, 146, *161*, *162*, 163, 164, 165, 166, 197, 201, 203, 209, 210, 211, 212, 213, 214, 215, 219, 225, 228
<COBOL array locator variable> • *161*, **162**, 163, 164, 166, 213
<COBOL binary integer> • ***162***, 165
<COBOL bit type> • ***161***, 163, 165, 166, 210
<COBOL BLOB locator variable> • *161*, **162**, 164, 166, 215
<COBOL BLOB variable> • ***161***, 164, 166, 215
<COBOL character type> • ***161***, 163, 165, 166, 209
<COBOL CLOB locator variable> • *161*, **162**, 164, 166, 215
<COBOL CLOB variable> • ***161***, 163, 164, 166, 215
<COBOL derived type specification> • ***161***, 166, 213, 214
<COBOL host identifier> • *134*, ***161***, 163, 164, 165
<COBOL integer type> • *161*, **162**
<COBOL national character type> • ***161***, 162, 163
<COBOL NCLOB variable> • ***161***, 163
<COBOL nines> • ***162***
<COBOL nines specification> • ***162***
<COBOL numeric type> • *161*, **162**, 165
<COBOL REF variable> • *161*, **162**, 165, 166, 211
<COBOL type specification> • ***161***, 166, 210, 211
<COBOL user-defined type locator variable> • *161*, **162**, 165, 166, 212
<COBOL user-defined type variable> • *161*, **162**, 164, 166, 214
<COBOL variable definition> • *134*, **161**, 163, 165, 219
collation • 105
<collation name> • 105
COLLATION_CAT • 75, 83, 86, 105, 218
COLLATION_CATALOG • 75, *83*, 86, 105, 218
COLLATION_NAME • 75, *83*, 86, 105, 218
COLLATION_SCHEM • 75, 83, 86, 105, 218
COLLATION_SCHEMA • 75, *83*, 86, 105, 218
collection • 47, 71, 150, 152, 156, 158, 162, 163, 164, 167, 170, 173, 175, 177, 179, 182, 183, 184
collection type • 71, 152, 156, 158, 163, 164, 170, 175, 179, 183, 184
<collection type> • *150*, 152, *156*, 158, *162*, 163, 164, *167*, 170, *173*, 175, *177*, 179, *182*, 183, 184
<collection value expression> • 47
<colon> • *134*, *149*, *176*
column • 16, 24, 26, 39, 41, 47, 84, 86, 95, 101, 102, 103, 104, 112, 120, 129, 130, 141, 151, 157, 164, 169, 174, 178, 183, 191, 199, 205, 208, 211, 227
column list • 95
<column name> • 130, 141, 191
<column name list> • 120, 130, 205, 208, 211
<column reference> • 39, 191
<comma> • *83*, *86*, *107*, *111*, *149*, *154*, *155*, *156*, *167*, *172*, *176*, *181*
<commit statement> • 55, 63, 188, 222
<comparison predicate> • 96, 192
compatible • 47, 110, 114, 225
compilation unit • 1, 24, 135, 150, 156, 162, 168, 173, 177, 182
completion condition • 27, 84, 103, 118, 125, 190, 191, 193
<comp op> • 192
<concatenation operator> • 93
condition • 27, 45, 46, 49, 51, 60, 65, 66, 67, 68, 69, 73, 80, 82, 84, 85, 87, 88, 89, 92, 97, 98, 99, 100, 101, 102, 103, 107, 108, 109, 110, 111, 112, 113, 114, 116, 118, 121, 122, 123, 125, 127, 129, 130, 144, 145, 146, 147, 148, 188, 189, 190, 191, 193, 199, 208, 217, 221, 225
<condition> • ***145***, 146, 225
<condition action> • ***145***, 147
conforming SQL-implementation • 228
CONNECT • *188*
CONNECTION • *188*
connection does not exist • 189
connection exception • 189
<connect statement> • 188
CONSTRAINT • *145*, 146, 147, 148, 208
constraint mode • 189
<constraint name> • *145*, 147
constraint violation • 146
contain • 3, 5, 15, 24, 25, 26, 28, 29, 33, 34, 36, 37, 38, 39, 40, 41, 45, 46, 49, 50, 51, 52, 53, 56, 62, 65, 66, 67, 68, 69, 80, 81, 82, 85, 90, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 105, 106, 107, 110, 111, 115, 116, 117, 118, 119, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 135, 136, 137, 138, 139, 140, 141, 143, 144, 146, 147, 148, 150, 152, 153, 156, 157, 158, 160, 163, 165, 166, 168, 170, 171, 173, 175, 177, 179, 180, 182, 184, 185, 187, 189, 191, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 217, 218, 219, 222, 225, 227

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

contained in • 5, 15, 26, 28, 33, 34, 37, 39, 41, 56, 62, 80, 85, 92, 93, 94, 95, 96, 97, 98, 101, 105, 107, 111, 117, 121, 123, 127, 129, 130, 135, 136, 137, 138, 139, 140, 141, 146, 147, 150, 152, 153, 156, 158, 160, 163, 165, 166, 168, 170, 171, 173, 175, 179, 180, 182, 184, 185, 191, 212, 213, 217, 218, 225, 227
containing • 5, 25, 26, 29, 39, 98, 100, 101, 116, 118, 119, 123, 124, 126, 127, 129, 135, 136, 146, 147, 150, 157, 163, 168, 177, 182, 219, 227
containing schema • 39
contains • 15, 24, 25, 26, 28, 33, 38, 40, 46, 66, 85, 92, 94, 98, 106, 116, 118, 122, 123, 136, 137, 138, 139, 140, 143, 146, 147, 148, 150, 156, 163, 168, 177, 179, 182, 189, 218, 219, 227
<contextually typed row value expression> • 95, 96
CONTINUE • *145*
<correlation name> • 191
corresponding fields • 92
corresponding subfields • 92
corresponds to • 94, 95, 96, 97, 139, 140, 142
COUNT • *31*, *57*, 75, *83*, 84, 86, 102, 107, 112
created by • 28, 35
<C REF variable> • *155*, **156**, 158, 160, 211
<C storage class> • ***154***
current • 3, 15, 16, 18, 19, 27, 28, 29, 34, 35, 37, 39, 41, 55, 63, 65, 66, 67, 68, 69, 82, 84, 87, 109, 114, 122, 124, 144, 162, 163, 188, 189, 217, 218
CURRENT • *127*, *129*, *131*, *132*
current authorization identifier • 188
current privileges • 39, 41, 144
current SQL-session • 28, 29, 34, 35, 65, 66, 67, 68, 69, 109, 114, 122, 188
CURRENT_DEFAULT_TRANSFORM_GROUP • 29, *31*, *37*, 38
CURRENT_PATH • 28, 191, 194
CURRENT_ROLE • 191
CURRENT_TRANSFORM_GROUP_FOR_TYPE • 29, *31*, *37*, 38
CURRENT_USER • 191
CURSOR • *119*, *121*
<cursor holdability> • *119*
<cursor intent> • ***121***
<cursor name> • 25, *34*, 55, 98, *119*, 123, 124, 126, 127, 129, 130, *131*, *132*, 136
<cursor returnability> • *119*, 120, 205, 215
<cursor scrollability> • *119*, 120, 205, 208
<cursor sensitivity> • *119*, *121*
<cursor specification> • *24*, *91*, *99*, 100, 119, 121, 122, 123, 129, 130, 131, 132
cursor specification cannot be executed • 116, 199
<C user-defined type locator variable> • *155*, **156**, 158, 160, 212
<C user-defined type variable> • ***155***, 157, 160, 214
<C VARCHAR variable> • *154*, **155**, 156, 159
<C variable definition> • *134*, **154**, 156, 157, 158, 159, 160, 209
<C variable specification> • ***154***

— D —

Data • 3, 12, 45, 47, 61, 75, 77, 78, 84, 86, 165, 193, 225
DATA • 75, *83*, 84, 85, 86, 87, 103, 108, 109, 114, 115
data exception • 45, 46, 84, 85, 87
data type • 23, 26, 29, 38, 47, 57, 65, 66, 67, 68, 69, 71, 72, 73, 74, 75, 76, 77, 78, 87, 88, 89, 92, 94, 95, 105, 108, 109, 110, 112, 113, 114, 137, 138, 139, 140, 141, 151, 152, 153, 156, 157, 158, 159, 160, 163, 164, 165, 166, 169, 170, 171, 173, 174, 175, 178, 179, 180, 183, 184, 185, 199, 210, 214, 215, 218, 225
<data type> • 26, 94, 137
data type transform function violation • 110, 114, 199
DATA_TYPE • 57, 74
date • 11, 16, 17, 18, 19, 20, 21, 22, 23, 25, 59, 72, 77, 79, 91, 94, 97, 98, 103, 104, 105, 117, 120, 129, 130, 132, 144, 187, 194, 205, 206, 207, 208, 211, 223
DATE • 23, 77, 78
datetime type • 72, 105
<datetime type> • 72, 105
DATETIME_INTERVAL_CODE • *57*, 72, 75, *83*, 87, 89, 105, 199
DATETIME_INTERVAL_PRECISION • 72, 75, *83*, 87, 88, 89, 105
DAY • 78, 89, 94
DEALLOCATE • *55*, *82*, *99*, *100*, 118, *188*, 194
<deallocate descriptor statement> • 17, 18, 25, 55, *59*, **82**, 188, 194, 204
<deallocate prepared statement> • 16, 17, 18, 24, 26, *59*, **100**, 194, 207
DEC • *172*, 173, *182*
DECIMAL • 72, 73, 77, 88, 173, *182*, 184
DECLARE • *119*, *133*, *134*, 142, *181*, 184
<declare cursor> • 20, 21, 22, 119, 122, 131, 132, *133*, 136, 138, 143, 222
declared dynamic cursor • 15, 98
declared type • 35, 37, 45, 65, 66, 67, 68, 73, 80, 84, 86, 91, 92, 93, 94, 95, 96, 97, 103, 108, 109, 110, 112, 118, 139
DEFAULT • 29, *31*, *37*, 38, *69*, *188*
default SQL-session • 188
deferred • 27
DEFINED • *57*, 72, 74, 76, 77, *84*, 87, 89, 90, 105
defining instance • 177
DEGREE • *31*, 72, 74, 75, *83*, 87, 88, 105
DELETE • *127*, *131*, 194
<delete statement: positioned> • 19, 21, 23, 127, 131, 144
<delete statement: searched> • 19, 20, 22, *91*, *187*
<delimiter token> • ***31***
dependencies • 85, 205, 215
dependent • 15, 26, 28, 61, 85, 88, 89, 99, 103, 104, 114, 138, 139, 140, 141, 142, 147, 188, 191, 192, 221, 222
depends on • 18
<derived column> • 104

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

derived table • 39
describe • 9, 17, 18, 23, 24, 26, 59, 85, 90, 101, 102, 103, 104, 105, 106, 107, 108, 111, 112, 152, 153, 159, 165, 170, 173, 179, 184, 185, 194, 205, 207, 218, 221, 225, 227, 228
DESCRIBE • *101*, 194
<described object> • ***101***
describe input • 17, 18, 24, 26, 101, 102, 103, 106, 108, 207, 221, 225, 228
<describe input statement> • 17, 18, 24, ***101***, 102, 103, 106, 108, 207, 221, 225
describe output • 17, 18, 24, 26, 101, 102, 103, 106, 112, *205*, 221
<describe output statement> • 17, 18, 24, ***101***, 102, 103, 106, 112, 205, 221
<describe statement> • *59*, 85, 90, **101**, 194, 218
Description • 71, 197, 201, 227, 228
descriptor • 9, 17, 18, 25, 26, 29, 34, 35, 36, 55, 59, 71, 72, 73, 74, 75, 76, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 101, 102, 103, 104, 106, 107, 108, 110, 111, 112, 114, 115, 122, 188, 194, 199, 200, 204, 205, 206, 207, 218, 221
DESCRIPTOR • *55*, *57*, *58*, *80*, *82*, *83*, *86*, *101*, *111*, *188*, 194
descriptor area • 25, 26, 29, 35, 55, 71, 72, 73, 74, 75, 76, 80, 82, 83, 84, 85, 86, 87, 88, 89, 102, 103, 104, 107, 108, 112, 114, 115, 200, 218, 221
<descriptor item name> • ***83***, 84, 85, *86*, 87
<descriptor name> • 25, **34**, 35, 36, 55, *80*, 81, *82*, *83*, 84, 85, *86*, 87, 90, *101*, 102, 107, 108, 110, *111*, 112, 115, 188, 204, 206, 207, 218
diagnostics area • 118, 188, 189, 190, 193
<digit> • *145*
DIRECT • 197
<direct implementation-defined statement> • ***187***, 188, 189, 219
<directly executable statement> • ***187***
direct result of executing an SQL-statement • 5
<direct select statement: multiple rows> • 17, 18, 20, 21, 22, 39, 41, *187*, **191**, 194
<direct SQL data statement> • ***187***, 190
<direct SQL statement> • 13, 27, 28, 29, 34, 60, 65, 66, 67, 68, **187**, 188, 189, 190, 201, 204, 217
distinct • 96, 138, 140, 143, 192, 222
<distinct predicate> • 96
domain • 94
<domain name> • 94
dormant • 60, 188, 189
DOUBLE • 72, 73, 77, *149*, 152, 159, *167*, 170
<double period> • ***31***, *149*, *176*
DYNAMIC • *31*, *55*, 56, *57*, 75, 78, *83*, 86, 102, *133*, *193*, 194, 208, 225
<dynamic close statement> • 16, 17, 18, 19, 21, 22, 24, *59*, **126**, 144, 194, 206
dynamic cursor • 15, 16, 25, 35, 36, 98, 123, 124, 126, 127, 129, 130, 131, 132, 204, 219
<dynamic cursor name> • **34**, 35, 36, *123*, *124*, *126*, *127*, *129*, 130, 204, 219
<dynamic declare cursor> • 15, 16, 19, 20, 21, 22, 24, *55*, 56, 99, ***119***, 120, 123, 124, 126, 127, 129, 131, 132, *133*, 136, 139, 143, 204, 205, 206, 208, 211, 215, 216
<dynamic delete statement: positioned> • 16, 17, 18, 19, 21, 23, 25, *59*, **127**, 128, 144, 194, 206
<dynamic fetch statement> • 16, 17, 18, 19, 21, 22, 24, 25, *59*, 90, 111, 112, **124**, 144, 194, 206, 218
<dynamic open statement> • 16, 17, 18, 19, 21, 22, 24, *59*, 90, 107, 122, **123**, 144, 194, 206, 218
<dynamic parameter specification> • 26, ***37***, 38, 40, 42, 49, 50, 51, 52, 53, 92, 98, 101, 102, 103, 104, 105, 106, 107, 108, 111, 112, 116, 118, 123, 187, 204, 218, 221, 225
dynamic result sets returned • 122
<dynamic select statement> • 17, 19, 20, 22, 23, 24, 60, ***91***, 92, 102, 103, 107, 111, 116, 118, 194
<dynamic single row select statement> • 17, 18, 20, 22, 23, 24, 25, 39, 60, *91*, 92, 99, 102, 103, 111, 112, 116, 118, **125**, 194, 206
dynamic SQL error • 80, 84, 85, 87, 88, 89, 99, 107, 108, 109, 110, 111, 112, 113, 114, 116, 122, 123, 199
<dynamic update statement: positioned> • 16, 17, 18, 20, 21, 23, 25, *59*, **129**, 130, 144, 194, 206
DYNAMIC_FUNCTION • *31*, 75, 78, *83*, 86, 102, *193*
DYNAMIC_FUNCTION_CODE • *31*, 75, 78, *83*, 86, 102, *193*

— E —

Editor's Note • 97, 139
effective • 15, 16, 28, 29, 55, 56, 60, 66, 87, 108, 109, 110, 112, 113, 114, 122, 137, 188, 192, 217
effectively • 15, 16, 28, 29, 55, 56, 60, 66, 87, 109, 110, 113, 114, 122, 137, 188, 192, 217
<element reference> • 92
elements • 31, 43, 89, 217, 221, 227
element type • 71, 74
embedded • 1, 6, 12, 13, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 37, 38, 43, 49, 50, 51, 52, 53, 61, 107, 133, 134, 135, 136, 137, 138, 139, 140, 141, 143, 144, 145, 146, 147, 148, 149, 150, 152, 153, 154, 156, 158, 160, 161, 162, 163, 165, 166, 167, 168, 170, 171, 172, 173, 175, 176, 177, 179, 180, 181, 182, 184, 185, 187, 201, 203, 204, 207, 208, 209, 212, 214, 217, 218, 219, 222, 225, 228
EMBEDDED • 197
<embedded authorization clause> • ***133***, 138
<embedded authorization declaration> • 24, ***133***, 136, 138, 143, 144, 208, 218
<embedded authorization identifier> • *133*, **134**
<embedded character set declaration> • ***134***, 136, 138, 143, 144, 208, 209, 219
<embedded exception declaration> • 18, *133*, 137, 139, 143, **145**, 146, 147, 148, 225
<embedded path specification> • *133*, **134**, 136, 138, 144, 212, 219

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

<embedded SQL Ada program> • 24, *133*, 135, 137, 138, 143, 146, **149**, 150, 152, 153, 203
<embedded SQL begin declare> • ***134***, 135, 143, 177
<embedded SQL COBOL program> • 24, *133*, 135, 137, 138, 143, 146, **161**, 162, 163, 165, 166, 203, 225
<embedded SQL C program> • 24, *133*, 135, 137, 138, 143, 146, **154**, 156, 158, 160, 203
<embedded SQL declare section> • 24, **134**, 135, 143, 144, 150, 156, 162, 168, 173, 177, 182, 208, 209
<embedded SQL end declare> • ***134***, 135, 143, 177
<embedded SQL Fortran program> • 24, *133*, 135, 137, 138, 143, 146, **167**, 168, 170, 171, 203
<embedded SQL host program> • 13, 24, 26, **133**, 135, 136, 137, 144, 146, 147, 219, 222
<embedded SQL MUMPS declare> • ***134***, 135, 143
<embedded SQL MUMPS program> • 24, *133*, 135, 137, 138, 146, **172**, 173, 175, 203
<embedded SQL Pascal program> • 24, *133*, 135, 137, 138, 143, 146, **176**, 177, 179, 180, 204
<embedded SQL PL/I program> • 24, *133*, 135, 137, 138, 143, 146, 147, **181**, 182, 184, 185, 204
<embedded SQL statement> • 24, 26, 37, **133**, 135, 136, 137, 143, 144, 150, 156, 162, 163, 168, 173, 177, 182
<embedded transform group specification> • *133*, **134**, 136, 138, 144, 214
<embedded variable name> • *37*, 38, 43, 52, 53, 61, ***134***, 138, 140, 144, 222
<embedded variable specification> • ***37***, 38, 49, 50, 51, 107, 187
empty • 124, 125, 191, 192
END • *134*
END-EXEC • *133*, 135
<equals operator> • *83*, *86*, *149*, *156*
error in assignment • 84, 87
<escape character> • 97
<escape octet> • 97
exact numeric • 37, 75, 76, 80, 84, 86, 105, 152, 153, 159, 165, 170, 173, 179, 184, 185, 193
<exact numeric type> • 105, 153
exception • 16, 18, 19, 20, 21, 22, 27, 45, 46, 60, 65, 66, 67, 68, 69, 80, 82, 84, 85, 87, 88, 89, 92, 97, 98, 99, 100, 101, 102, 107, 108, 109, 110, 111, 112, 113, 114, 116, 118, 121, 122, 123, 125, 127, 129, 130, 133, 137, 139, 143, 144, 145, 146, 147, 148, 188, 189, 190, 193, 207, 208, 221, 225, 228
EXCEPTION • *145*, 148, 207
exception condition • 27, 45, 46, 60, 65, 66, 67, 68, 69, 80, 82, 84, 85, 87, 88, 89, 92, 97, 98, 99, 100, 101, 102, 107, 108, 109, 110, 111, 112, 113, 114, 116, 118, 121, 122, 123, 125, 127, 129, 144, 188, 189, 190, 221
EXEC • *133*, 135
EXECUTE • *57*, *116*, *118*, 144, 194
executed immediately • 22, 26
<execute immediate statement> • 5, 15, 17, 18, 24, 25, 26, 27, 28, 29, 34, 56, *59*, 65, 66, 67, 68, **118**, 194, 205, 217, 218
<execute statement> • 5, 17, 18, 24, 27, *59*, 68, 90, 99, 107, 111, **116**, 117, 194, 205, 218
<extended cursor name> • 25, ***34***, 35, 36, 98, *101*, 102, *121*, 122, 123, 127, 129, 206
extended dynamic cursor • 15, 16, 98
<extended statement name> • 25, ***34***, 35, 36, 63, 99, *121*, 122, 206, 221
externally-invoked procedure • 1, 5, 15, 19, 20, 24, 26, 28, 55, 56, 57, 60, 117, 138, 139, 142, 143, 189, 217, 222
<externally-invoked procedure> • 5, 15, 19, 20, 24, 26, 28, 55, 56, 57, 60, 117, 138, 139, 142, 143, 189, 217, 222
external routine • 45

— F —

Feature B011 • 153, 203
Feature B012 • 160, 203
Feature B013 • 166, 203
Feature B014 • 171, 203
Feature B015 • 175, 203
Feature B016 • 180, 204
Feature B017 • 185, 204
Feature B021 • 190, 204
Feature B031 • 36, 38, 56, 60, 81, 82, 85, 90, 99, 106, 110, 115, 117, 118, 119, 120, 123, 124, 125, 126, 128, 130, 204, 205, 206, 208, 211, 215
Feature B032 • 36, 81, 82, 85, 90, 100, 106, 110, 115, 117, 122, 131, 132, 206, 207
Feature B041 • 148, 207, 208
Feature B051 • 56, 144, 208
Feature F341 • 120, 205, 208
Feature F361 • 144, 208
Feature F421 • 120, 205, 208
Feature F451 • 67, 144, 153, 160, 166, 171, 180, 185, 208, 209, 210, 211
Feature F461 • 67, 208, 209, 210, 211
Feature F511 • 153, 160, 166, 171, 180, 185, 210
Feature F611 • 38, 210
Feature F651 • 65, 210
Feature F761 • 65, 66, 67, 208, 210, 211
Feature F791 • 119, 205, 206, 211, 215
Feature F831 • 120, 205, 208, 211
Feature S024 • 120, 211
Feature S041 • 153, 160, 166, 171, 175, 180, 185, 211, 212
Feature S071 • 68, 144, 212
Feature S231 • 153, 160, 166, 171, 175, 180, 185, 212, 213
Feature S232 • 153, 160, 166, 171, 175, 180, 185, 213
Feature S241 • 69, 144, 153, 160, 166, 171, 175, 180, 185, 213, 214
Feature T041 • 153, 160, 166, 171, 175, 180, 185, 214, 215
Feature T231 • 119, 206, 211, 215

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Feature T301 • 85, 205, 215
Feature T471 • 120, 205, 215
Feature T551 • 120, 216
Fetch • 124
FETCH • *124*, 194
<fetch orientation> • *124*
<fetch statement> • 19, 21, *22*, 144
<fetch target list> • 124
fields • 25, 71, 80, 88, 92, 103, 116, 199, 225
final • 227
fixed-length • 159
FLOAT • 72, 73, 77, 89, *182*, 185
FOR • 15, 26, *55*, 56, *69*, *119*, 120, *121*, *133*, 205, 208, 211
Fortran • 3, 12, 13, 23, 24, 133, 134, 135, 137, 138, 143, 146, 167, 168, 169, 170, 171, *201*, 203, 209, 210, 211, 212, 213, 214, 215, 228
FORTRAN • 138, 197
<Fortran array locator variable> • ***167***, 170, 171, 213
<Fortran BLOB locator variable> • ***167***, 169, 171, 215
<Fortran BLOB variable> • ***167***, 169, 171, 215
<Fortran CLOB locator variable> • ***167***, 169, 171, 215
<Fortran CLOB variable> • ***167***, 168, 169, 171, 215
<Fortran derived type specification> • ***167***, 171, 211, 213, 214
<Fortran host identifier> • *134*, ***167***, 168, 169, 170
<Fortran REF variable> • *167*, **168**, 170, 171, 211
<Fortran type specification> • ***167***, 168, 171, 209, 210
<Fortran user-defined type locator variable> • ***167***, 169, 171, 212
<Fortran user-defined type variable> • ***167***, 169, 171, 214
<Fortran variable definition> • *134*, **167**, 168, 170
FOUND • *145*, 148
<free locator statement> • 20, 21, 23, 62
FROM • *65*, *66*, *67*, *68*, *69*, *80*, *91*, 93, *99*, 118, *121*, *124*, *127*, *131*
from-sql function • 114
FS • 114, 141, 142
FUNCTION • *31*, *57*, 75, 78, *83*, 86, 102, *193*

— G —

G • *66*, 194
generally contain • 52
<general value specification> • 28, 29, ***37***, 38, *107*, 204
GET • *83*, 194
<get descriptor information> • ***83***, 85
<get descriptor statement> • 17, 18, 25, *59*, **83**, 84, 85, 194, 204, 221
<get diagnostics statement> • 193
<get header information> • ***83***, 84, 85
<get item information> • ***83***, 84, 85
GLOBAL • *34*, 35, 98
GO • *145*, 146, 147, 148
GOTO • *145*
<go to> • ***145***, 146, 147, 148
<goto target> • ***145***, 146
<greater than operator> • 192
<group name> • 136, 139, 141
<group specification> • 136, 139, 141
GROUP_NAME • 57

— H —

handle • 23, 133
<handler declaration> • *133*
<header item name> • ***83***, 84, 85, *86*, 89
HOLD • 119, *121*
holdable • 15, 29
holdable locator • 15
<hold locator statement> • 23
<host identifier> • ***134***, 137, 143
<host label identifier> • ***145***, 146, 147, 148
host language • 1, 23, 135, 137, 139, 140, 143, 146, 147, 148, 219
<host parameter data type> • 138, 140
<host parameter declaration> • 138, 139, 140, 143, 222
<host parameter name> • 138, 140, 222
<host parameter specification> • 43, 107
<host PL/I label variable> • ***145***, 146, 147, 148
<host variable definition> • ***134***, 137, 143, 144, 208
HOUR • 78, 89, 97

— I —

identified • 25, 27, 38, 62, 69, 85, 89, 94, 100, 102, 108, 109, 112, 113, 121, 122, 123, 124, 126, 127, 129, 130, 131, 132, 140, 141, 151, 152, 157, 158, 164, 165, 169, 170, 174, 175, 178, 179, 183, 184, 217
identifier • 12, 25, 26, 27, 34, 39, 55, 69, 71, 78, 80, 97, 99, 102, 121, 133, 134, 137, 143, 145, 146, 147, 148, 149, 150, 151, 152, 154, 155, 156, 157, 158, 159, 161, 163, 164, 165, 167, 168, 169, 170, 172, 173, 174, 175, 176, 177, 178, 179, 181, 182, 183, 184, 188, 193, 194, 222
<identifier> • 25, *34*, 69, 80, 99, 121, 193
identify • 25, 28, 29, 74, 92, 99, 100, 101, 102, 106, 112, 113, 116, 121, 122, 123, 127, 129, 139, 153, 160, 166, 171, 175, 180, 185, 212, 213
immediate • 5, 15, 16, 17, 18, 20, 22, 24, 25, 26, 27, 28, 29, 34, 40, 56, 59, 62, 65, 66, 67, 68, 71, 72, 74, 80, 88, 92, 94, 95, 97, 98, 103, 107, 108, 117, 118, 121, 137, 156, 163, 182, 189, 194, 205, 217, 218, 225
IMMEDIATE • *118*, 194
immediately contain • 28, 40, 80, 92, 94, 95, 97, 98, 107, 117, 137, 225
immediately subordinate descriptor areas • 71, 72, 88
immediately subordinate descriptors • 71
implementation-defined • 15, 21, 24, 27, 28, 29, 35, 38, 72, 73, 74, 80, 88, 89, 90, 91, 92, 94, 97, 104, 105, 109, 112, 113, 118, 136, 138, 152, 158, 159, 165, 170, 173, 175, 179, 184, 187, 188, 189, 190, 192, 193, 194, 197, 210, 217, 218, 219
implementation-dependent • 15, 26, 28, 61, 85, 88, 89, 99, 103, 104, 114, 138, 139, 140, 141, 142, 147, 188, 191, 192, 221, 222
IN • 93, 106, 197

--`,,`--`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

include • 25, 39, 41, 98, 105, 143, 144, 156, 159, 182
Indicator • 38, 210
INDICATOR • *37*, 75, *83*, 85, 87, 103, 108, 114, *149*, 153, 221
indicator overflow • 45
indicator parameter • 38, 45, 46, 85, 210, 218
<indicator parameter> • 38, 210, 218
<indicator variable> • ***37***, 38, 42, 210, 218
Information Schema • 26
INOUT • 79, 98
<in predicate> • 96, 225
INPUT • 98, *101*
input parameter • 114
input SQL parameter • 116
<input using clause> • **107**, *116*, *123*
INSENSITIVE • *119*, *121*, 205, 211
<insert column list> • 95
<insert statement> • 19, 20, 22, *91*, 95, *187*
instance • 106, 177, 222
insufficient item descriptor areas • 103, 200
INT • *149*, 152, *172*, 173
INTEGER • 72, 73, 77, 152, 159, 165, *167*, *169*, *170*, 173, *176*, *178*, *179*, 185
integrity constraint violation • 146
interface • 25
INTERVAL • 23, *40*, *57*, 72, 75, 77, *83*, 87, 88, 89, 92, 94, 97, 105, 199
<interval fractional seconds precision> • 72, 94, 105
<interval leading field precision> • 72, 94, 105
<interval primary> • ***40***
<interval qualifier> • *40*, 72, 78, 92, 105
interval type • 72, 93, 105
<interval type> • 72, 105
<interval value expression> • 40, 97
<interval value function> • *40*
INTERVAL_PRECISION • 72, 75, 83, 87, 88, 89, 105
INTO • *111*
<into argument> • ***111***, 112, 115
<into arguments> • ***111***, 112, 115
<into descriptor> • ***111***, 112, 114
<introducer> • 33
invalid • 46, 60, 65, 66, 67, 68, 69, 80, 82, 84, 87, 88, 89, 98, 99, 100, 101, 102, 107, 108, 111, 112, 116, 121, 122, 123, 127, 129, 144, 189, 190, 199, 200
invalid catalog name • 65
invalid character set name • 67, 199
invalid cursor name • 98, 99, 102, 121, 123, 127, 129
invalid cursor state • 100
invalid DATA target • 87
invalid DATETIME_INTERVAL_CODE • 89
invalid descriptor count • 108, 112, 199
invalid descriptor index • 80, 84, 87, 199
invalid indicator parameter value • 46
invalid LEVEL value • 87, 88
invalid schema name • 66, 68, 200
invalid schema name list specification • 68, 200
invalid SQL descriptor name • 80, 82, 84, 87, 102, 107, 111
invalid SQL-invoked procedure reference • 122, 199
invalid SQL statement • 144
invalid SQL statement identifier • 99
invalid SQL statement name • 100, 101, 116, 121, 123, 127, 129
invalid transaction state • 60, 189, 190
invalid transform group name specification • 69, 200
<in value list> • 225
IS • *149*, *150*, *151*, *152*, *154*, *155*, *156*, *157*, *158*, 159, *161*, *162*, *163*, *164*, *165*, *167*, *168*, *169*, *170*, *172*, *173*, *174*, *175*, *176*, *177*, *178*, *179*, *181*, *182*, *183*, *184*, 197
isolation level • 189
ITEM • *58*
<item number> • ***83***, 84, 85, ***86***, 87

— K —

KEY • 75, *83*, 85, 86, 102, 103, 104, 205, 215
<key word> • 75, 84, 86
KEY_MEMBER • 75, *83*, 85, 86, 104, 205, 215
KEY_TYPE • 75, *83*, 86, 102, 103, 104

— L —

LANGUAGE • 197
<language clause> • 15, 105, 138
LARGE • 23, 72, 73, 74, 88, 115, 159
<large object length> • *150*, 151, *155*, 156, 157, *161*, *162*, 163, *167*, 168, 169, *172*, 174, *176*, 177, 178, *181*, 183
large object string • 105
leaf underlying table • 127, 129, 131, 132
<left bracket> • *154*, *176*
<left paren> • *12*, *13*, 92, 93, *133*, 135, *149*, *150*, *155*, *161*, *162*, *167*, *172*, *176*, *181*
LENGTH • 72, 73, 74, 75, 76, *83*, *84*, 86, 87, 88, 89, 104, 105, 115, 151, 159, 164, 169, 174, 178, 183, 218, 221
<length> • 26, 92, *149*, 150, *152*, *154*, 156, 157, 159, *161*, *162*, 163, 165, *167*, 168, *170*, *172*, 173, *176*, 177, *179*, 180, *181*, 182, 183, *184*, 210
length in positions • 105
<less than operator> • 192
<like predicate> • 97
<literal> • 33, 34, 81, 82, 85, 86, 90, 91, 110, 115, 191, 206, 207
LOB • 153, 160, 166, 171, 175, 180, 185, 214, 215
LOCAL • *34*, 35, 98
<local or schema qualified name> • 34
local temporary table • 28, 116
local time • 97
locator • 6, 15, 20, 21, 23, 29, 45, 62, 74, 109, 112, 113, 114, 136, 138, 139, 140, 141, 149, 150, 151, 152, 153, 155, 156, 158, 160, 161, 162, 163, 164, 165, 166, 167, 169, 170, 171, 172, 173, 174, 175, 176, 178, 179, 180, 181, 182, 183, 184, 185, 212, 213, 214, 215
Locator • 15

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

LOCATOR • 72, 74, 77, 85, 87, 88, 108, 114, 115, 138, 139, 140, *150*, *151*, *152*, *155*, *156*, *157*, *158*, 159, *162*, *164*, *167*, *169*, *170*, *172*, *173*, *174*, *175*, *176*, *177*, *178*, *179*, *181*, *182*, *183*
<locator reference> • ***62***

— M —

<major category> • ***145***, 146, 148, 207
match • 73, 74, 75, 84, 87, 96, 97, 107, 108, 111, 112, 199
MATCH • *57*
<match predicate> • 96
MAX • *80*, 218
meets • 92, 96
method • 27, 110, 190, 192, 217
<minus sign> • 94
MINUTE • 78, 89, 97
modified • 150, 152, 156, 158, 163, 165, 168, 170, 173, 175, 177, 179, 182, 184
module • 6, 8, 15, 16, 19, 20, 24, 25, 26, 28, 29, 33, 35, 55, 56, 98, 100, 101, 116, 118, 119, 123, 124, 126, 127, 129, 134, 135, 137, 138, 143, 144, 204, 217, 218, 219, 222
MODULE • 197
<module authorization clause> • 15, 24, 25, 26, 28, ***55***, 138, 218
<module authorization identifier> • 25, 26, *55*, *134*, 222
<module character set specification> • 15, 138
<module contents> • ***55***, 56, 204
<module name> • 15, 24, 26, 138, 222
<module name clause> • 138
<module path specification> • 138, 219
<module transform group specification> • 138
<modulus expression> • 93
MONTH • 78, 89, 94
most specific type • 87
<multiple group specification> • 136
MUMPS • 3, *12*, *13*, 24, *133*, *134*, 135, 137, 138, 143, 146, *172*, *173*, 174, 175, 197, 201, 203, 211, 212, 213, 214, 215, 228
<MUMPS array locator variable> • *172*, **173**, 175, 213
<MUMPS BLOB locator variable> • ***172***, 174, 175, 215
<MUMPS BLOB variable> • ***172***, 174, 175, 215
<MUMPS character variable> • ***172***, 173
<MUMPS CLOB locator variable> • ***172***, 174, 175, 215
<MUMPS CLOB variable> • ***172***, 173, 174, 175, 215
<MUMPS derived type specification> • ***172***, 173, 175, 211, 213, 214
<MUMPS host identifier> • *134*, ***172***, 173, 174, 175
<MUMPS length specification> • ***172***, 173
<MUMPS numeric variable> • ***172***
<MUMPS REF variable> • *172*, **173**, 175, 211
<MUMPS type specification> • ***172***
<MUMPS user-defined type locator variable> • *172*, **173**, 174, 175, 212
<MUMPS user-defined type variable> • ***172***, 174, 175, 214
<MUMPS variable definition> • *134*, **172**, 173, 175
<mutated set clause> • 97

— N —

Name • 34, 67, 144, 153, 160, 166, 171, 180, 185, 208, 209, 210, 211, 228
NAME • 57, 67, 72, 73, 74, 75, 76, *83*, 84, 86, 87, 88, 89, 90, 103, 104, 105, 106, 134, 194, 218, 221
NAMES • *67*, *134*, 194
NATIONAL • 159, 170
NCHAR • *154*, *155*, 156, *157*, 159
NESTING • *31*, *101*, 102, 103
<nesting option> • ***101***
NEXT • 124
NO • 119
no additional dynamic result sets returned • 122
no data • 84, 122, 125, 190, 191
nonholdable • 29
<non-reserved word> • ***31***
no subclass • 199, 200
NOT • *145*, 148, 197
null • 33, 38, 42, 45, 46, 85, 104, 105, 108, 109, 113, 114, 159, 187, 192, 219
NULL • 197
NULLABLE • 75, *83*, 86, 103, 104
null value • 33, 45, 46, 85, 104, 108, 109, 113, 114, 187, 192, 219
null value, no indicator parameter • 45, 85
NUMERIC • 72, 73, 77, 88, 92, 93, 165
<numeric value expression> • 92
<numeric value expression dividend> • 93
<numeric value expression divisor> • 93

— O —

object • 6, 12, 13, 29, 45, 62, 101, 105, 112, 113, 129, 130, 138, 140, 150, 151, 152, 155, 156, 157, 158, 161, 162, 163, 164, 167, 168, 169, 172, 174, 176, 177, 178, 181, 183, 221
OBJECT • 23, 72, 73, 74, 88, 115, 159
object column • 129, 130
<occurrences> • ***80***, 81, 84, 87, 102, 108, 112, 206, 218
<octet length expression> • 92
<octet match value> • 97
<octet pattern> • 97
OCTET_LENGTH • 75, 76, *83*, *84*, 86, 104, 105, 115, 218, 221
OF • *127*, *129*, *131*, *132*, *176*, 177, 179, 180, 210
OLD • 119, *121*
ONLY • 15, 26, *55*, 56, 127, 129, *131*, *132*, *133*, 208
OPEN • *123*, 194
<open statement> • 19, 21, 22, 122, 138, 139, 143, 144, 222
Option • 120, 216
OR • 197
order • 16, 61, 71, 76, 87, 103, 121, 122, 129, 139, 140, 143, 191, 192, 199, 203, 222
ORDER • 191
<order by clause> • 129, *191*, 192
<ordinal element specification> • 92

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ORDINAL_POSITION • 76, 83, 86, 103, 106, 218
outermost • 114
OUTPUT • 98, *101*
output SQL parameter • 78
<output using clause> • **111**, 112, 115, *116*, *124*, 205
overlaps • 96
<overlaps predicate> • 96
OVERLAY • 93

— P —

package • 57, 227
parameter • 24, 25, 26, 37, 38, 40, 42, 43, 45, 46, 49, 50, 51, 52, 53, 57, 78, 85, 92, 97, 98, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 114, 116, 117, 118, 123, 138, 139, 140, 141, 142, 143, 153, 187, 199, 204, 210, 218, 221, 222, 225
PARAMETERS • *57*
<parameter using clause> • 107, ***116***, 117
PARAMETER_MODE • 75, 78, 79, *83*, 86, 106, 218
PARAMETER_ORDINAL_POSITION • 76, *83*, 86, 103, 106, 218
PARAMETER_SPECIFIC_CATALOG • 76, *83*, 86, 103, 106, 218
PARAMETER_SPECIFIC_NAME • 76, *83*, 86, 103, 106, 218
PARAMETER_SPECIFIC_SCHEMA • 76, *83*, 86, 103, 106, 218
Part 1 • 3
Part 2 • 3
Part 3 • 3
Part 4 • 3
Part 5 • 12, 13
<Part 5 conformance> • ***12***
<Part 5 direct> • ***12***
<Part 5 direct no> • ***12***, 13
<Part 5 direct yes> • ***12***, 13
<Part 5 embedded> • ***12***
<Part 5 embedded Ada> • ***12***
<Part 5 embedded C> • ***12***
<Part 5 embedded COBOL> • *12*, **13**
<Part 5 embedded Fortran> • *12*, **13**
<Part 5 embedded languages> • ***12***
<Part 5 embedded MUMPS> • *12*, **13**
<Part 5 embedded no> • ***12***, 13
<Part 5 embedded Pascal> • *12*, **13**
<Part 5 embedded PL/I> • *12*, **13**
<Part 5 yes> • **12**
part of • 1, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 23, 66, 123, 162, 163, 201, 217, 221, 223, 225, 227
Pascal • 3, 12, 13, 23, 24, 133, 134, 135, 137, 138, 143, 146, 176, 177, 178, 179, 180, 201, 204, 209, 210, 212, 213, 214, 215, 228
PASCAL • 138, 179, 197
<Pascal array locator variable> • ***176***, 179, 180, 213
<Pascal BLOB locator variable> • ***176***, 178, 180, 215
<Pascal BLOB variable> • ***176***, 178, 180, 215
<Pascal CLOB locator variable> • ***176***, 178, 180, 215
<Pascal CLOB variable> • ***176***, 177, 178, 180, 215
<Pascal derived type specification> • ***176***, 180, 212, 213, 214
<Pascal host identifier> • *134*, ***176***, 177, 178, 179
<Pascal REF variable> • *176*, **177**, 179, 180, 212
<Pascal type specification> • ***176***, 177, 180, 209, 210
<Pascal user-defined type locator variable> • ***176***, 178, 180, 213
<Pascal user-defined type variable> • ***176***, 178, 180, 214
<Pascal variable definition> • *134*, **176**, 177, 179
<path specification> • *134*
period • 31, 149, 161, 176
<period> • *149*, *161*
permitted • 27, 217
PL/I • 3, 12, 13, 23, 24, 133, 134, 135, 137, 138, 143, 145, 146, 147, 148, 181, 182, 183, 184, 185, 201, 204, 209, 210, 212, 213, 214, 215, 219, 228
<PL/I array locator variable> • *181*, **182**, 184, 185, 213
<PL/I BLOB locator variable> • ***181***, 183, 185, 215
<PL/I BLOB variable> • ***181***, 183, 185, 215
<PL/I CLOB locator variable> • ***181***, 183, 185, 215
<PL/I CLOB variable> • ***181***, 182, 183, 185, 215
<PL/I derived type specification> • ***181***, 185, 212, 213, 214
<PL/I host identifier> • *134*, ***181***, 182, 183, 184
<PL/I REF variable> • *181*, **182**, 184, 185, 212
<PL/I type fixed binary> • *181*, **182**
<PL/I type fixed decimal> • *181*, **182**
<PL/I type float binary> • *181*, **182**
<PL/I type specification> • ***181***, 182, 185, 209, 210, 219
<PL/I user-defined type locator variable> • ***181***, 183, 185, 213
<PL/I user-defined type variable> • ***181***, 183, 185, 214
<PL/I variable definition> • *134*, **181**, 182, 184, 219
PLACING • 93
PLI • *13*, 138, 197
<plus sign> • 94
POSITION • 76, *83*, 86, 93, 103, 106, 218
<position expression> • 93
possibly nullable • 104
potentially referenced cursors • 98, 99
precede • 55, 71, 136, 137, 156, 163, 168, 182, 192
precedes • 71, 136, 137, 168, 192
precision • 26, 72, 88, 89, 92, 94, 105, 165, 172, 173, 181, 184, 185
PRECISION • 72, 73, 75, 76, 77, *83*, 87, 88, 89, 105, *149*, 152, 159, *167*, 170, 221
<precision> • 26, 92, *172*, 173, *181*, 184, 185
predefined • 69, 109, 113, 150, 155, 162, 167, 172, 176, 181
predefined data types • 69, 109, 113
<predefined type> • *150*, *155*, *162*, *167*, *172*, *176*, *181*
preferred candidate key • 103

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

<preparable dynamic delete statement: positioned> • 17, 18, 20, 21, 23, 25, *91*, 98, 117, **131**, 194, 207
<preparable dynamic update statement: positioned> • 17, 18, 20, 21, 23, 25, *91*, 98, 117, **132**, 194, 207
<preparable implementation-defined statement> • 21, ***91***, 218
<preparable SQL control statement> • ***91***, 92
<preparable SQL data statement> • ***91***
<preparable SQL schema statement> • ***91***
<preparable SQL session statement> • ***91***
<preparable SQL transaction statement> • ***91***
<preparable statement> • 15, 27, 28, 29, 34, 37, 65, 66, 67, 68, **91**, 92, 117, 217, 218
Prepare • 91
PREPARE • *57*, *91*, *99*, *100*, 118, 194
prepared • 15, 16, 17, 18, 24, 25, 26, 27, 28, 29, 34, 35, 37, 38, 56, 59, 65, 66, 67, 68, 69, 78, 92, 99, 100, 101, 102, 105, 107, 111, 112, 116, 119, 121, 122, 123, 127, 129, 193, 194, 199, 207, 217, 218, 219, 221
prepared statement not a cursor specification • 99, 122, 199
<prepare statement> • 5, 15, 17, 18, 24, 25, 26, 27, 28, 29, 34, 35, 37, 56, *59*, 65, 66, 67, 68, 69, **91**, 99, 100, 101, 116, 119, 194, 205, 217, 218
primary key • 102, 104
privilege • 7, 27, 39, 41, 144, 188
PROCEDURE • *57*, 85, *121*
<procedure name> • 24, 138, 139, 222

— Q —

<quantified comparison predicate> • 96
query • 39, 42, 47, 95, 96, 123, 125, 191, 225
<query expression> • 39, 47, 95, 123, *191*
<query specification> • 39, 42, *125*
<question mark> • 26, *37*

— R —

read-only • 189
read-only SQL-transaction • 189
read-write • 189
REAL • 72, 73, 77, *149*, 152, 159, *167*, 170, *172*, 173, *176*, 179
REF • 72, 74, 77, 159
referenced type • 105, 152, 158, 165, 170, 175, 179, 184
reference type • 74, 105, 152, 153, 158, 160, 165, 166, 170, 171, 175, 179, 180, 184, 185, 211, 212
<reference type> • 105, *150*, 152, *156*, 158, *162*, 165, *168*, 170, *173*, 175, *177*, 179, *182*, 184
released • 26
repertoire • 159
represented • 26, 29, 62, 74, 75, 76, 84, 108, 112
<reserved word> • ***31***, 225
RESTRICT • *57*
restricted data type attribute violation • 109, 110, 112, 113, 114, 199
RESULT • *57*
result data type • 47, 141
<result expression> • 95
result set • 101, 121, 122
result set cursor • 121, 122
<result set cursor> • ***121***, 122
<result using clause> • 111, ***116***, 117, 207
Retrieval assignment • 45
RETURN • 119, *121*
returned value • 26
RETURNED_CARDINALITY • *31*, 76, *84*, 86, 115
RETURNED_LENGTH • 76, *84*, 86, 115
RETURNED_OCTET_LENGTH • 76, *84*, 86, 115
<returns data type> • 114
<return value> • 110, 114
<right bracket> • *154*, *176*
<right paren> • *12*, *13*, 92, 93, *133*, 135, *149*, *150*, *155*, *161*, *162*, *167*, *172*, *176*, *181*
role • 27, 97
<role specification> • 97
<rollback statement> • 55, 188, 222
<routine body> • 15
<routine invocation> • 28, 43, 68, 97, 98, 105
<routine name> • 28, 68, 141
row • 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 39, 41, 60, 61, 71, 72, 74, 84, 86, 91, 92, 95, 96, 97, 99, 102, 103, 104, 105, 108, 111, 112, 116, 118, 122, 124, 125, 127, 129, 131, 132, 151, 157, 164, 169, 174, 178, 183, 187, 191, 192, 194, 197, 206, 225, 227
ROW • 72, 74, 77, 85, 87, 88, 105, 108, 114
row type • 71, 72, 74, 92, 95, 96, 105
<row value constructor> • 96
<row value constructor element> • 96, 97
<row value expression 1> • 96
<row value expression 2> • 96
<row value expression> • 95, 96, 225

— S —

SCALE • 72, 73, 76, *84*, 87, 88, 89, 105, 218, 221
<scale> • 26, *172*, 173, *181*, 184
schema • 8, 15, 17, 19, 20, 22, 26, 27, 28, 29, 34, 39, 55, 59, 60, 65, 66, 68, 91, 97, 106, 133, 138, 187, 189, 190, 194, 200, 210, 217, 218
SCHEMA • 15, 26, 28, *55*, *57*, *66*, 72, 73, 74, 75, 76, *83*, *84*, 86, 87, 88, 89, 90, 103, 105, 106, *133*, 138, 194, 218
schema and data statement mixing not supported • 60, 190
<schema definition> • 34
<schema name> • 15, 26, 28, *55*, 65, 66, *133*, 138, 217, 218
<schema name characteristic> • ***66***, 97
<schema name list> • 68
<schema qualified name> • 28, 34, 66, 217, 218
SCHEMA_NAME • *57*
scope • 1, 5, 25, 34, 35, 74, 80, 82, 98, 99, 100, 101, 102, 107, 111, 116, 121, 122, 123, 127, 129, 131, 132, 137, 152
Scope • 1

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

SCOPE • 72, 74, 76, *84*, 87, 90, 105
<scope option> • ***34***, 35, 80, 82, 98, 99, 102, 107, 111, *131*, *132*
SCOPE_CATALOG • 72, 74, 76, 84, 87, 90, 105
SCOPE_NAME • 72, 74, 76, 84, 87, 90, 105
SCOPE_SCHEMA • 72, 74, 76, 84, 87, 90, 105
SCROLL • 119, *121*, 124
scrollability • 119, 120, 205, 208
search condition • 49, 51
<search condition> • 49, 51
SECOND • 78, 89, 94
SECTION • *134*
SELECT • 39, 41, 194
<select list> • 61, 101, 102, 112
<select statement: single row> • 19, 21, 23, 61
<select target list> • 61
<semicolon> • *133*, 135, *154*, *172*, *176*, *181*, *187*
sensitive • 119, 205, 206, 211, 215
sensitivity • 119, 121
<separator> • 31, 33, 135, 146, 168
sequence • 28, 146, 152, 158, 165, 184
SERIALIZABLE • 189
SESSION • 191
SESSION_USER • 191
SET • *65*, *66*, *67*, *68*, *69*, *86*, *129*, *132*, 142, *149*, *150*, *154*, *155*, 156, 157, 159, *161*, 163, *167*, 168, *172*, 173, *176*, 177, 179, *181*, 182, *188*, 194
<set catalog statement> • 15, 17, 26, 28, *59*, **65**, 194, 210
<set clause> • 97
<set clause list> • *129*, *132*
<set connection statement> • 188
<set descriptor information> • ***86***
<set descriptor statement> • 17, 18, 25, *59*, 85, **86**, 87, 89, 90, 194, 205, 221
<set function specification> • 191
<set header information> • ***86***, 89
<set item information> • ***86***, 87
<set local time zone statement> • 97
<set names statement> • 17, 26, 29, *59*, **67**, 194, 208, 211
<set path statement> • 17, 26, 28, *59*, **68**, 194, 212, *218*
<set schema statement> • 17, 26, 28, *59*, **66**, 194, 210
<set session characteristics statement> • *59*
<set transaction statement> • 189
<set transform group statement> • 17, 28, *59*, **69**, 92, 194, 213
shadow module • 15, 26, 56
shall • 24, 33, 34, 35, 36, 37, 38, 39, 40, 41, 43, 45, 49, 50, 51, *52*, 53, 55, 56, *57*, 60, 61, 62, 65, 66, 67, 68, 69, 80, 81, 82, 84, 85, 86, 90, 91, 92, 99, 100, 101, 102, 106, 107, 110, 111, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 135, 136, 137, 138, 139, 141, 144, 146, 148, 150, 151, 152, 153, 156, 157, 158, 159, 160, 162, 163, 164, 165, 166, 168, 169, 170, 171, 173, 174, 175, 177, 178, 179, 180, 182, 183, 184, 185, 187, 190, 191, 192, 201, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 218
<signed integer> • 191
significant • 168, 179
<similar pattern> • 97
<similar predicate> • 97
SIMPLE • 3, 40, 55, 65, 66, 67, 68, 69, 80, 99, 109, 110, 113, 118, 121, 131, 132, 139, 142, 146, 150, 151, 152, 157, 158, 163, 164, 165, 168, 169, 170, 173, 174, 175, 177, 178, 179, 182, 183, 184, 188
<simple case> • 95
<simple comment> • 92
<simple Latin upper case letter> • *145*
<simple target specification 1> • ***83***, 84, 85, 221
<simple target specification 2> • ***83***, 84, 85, 221
<simple target specification 3> • 221
<simple target specification> • ***37***, 73, 74, *83*, 84, 99, 193
<simple value specification 1> • ***86***, 89
<simple value specification 2> • ***86***, 87
<simple value specification> • 25, *34*, 35, ***37***, 73, 74, *80*, 82, *83*, *86*, *91*, 102, 107, 111, 121
simply contain • 39, 92, 93, 94, 95, 96, 97, 105, 123, 153, 160, 166, 171, 175, 180, 185, 191, 212, 213, 218
simply contained in • 39, 93, 94, 95, 96, 97, 105, 123, 153, 160, 166, 171, 175, 180, 185, 212, 213, 218
simply underlying table • 127, 129
<single group specification> • 136
SMALLINT • 72, 73, 77, *149*, 152, 159, 165, 185
<*solidus*> • 93
<sort key> • 120, 211
<sort specification> • 191, 192
<sort specification list> • 39
source • 95, 97
SOURCE • 197
<space> • 25, 135, 168
specific name • 106
<specific routine designator> • *121*, 122
SPECIFIC_CATALOG • 76, 83, 86, 103, 106, 218
SPECIFIC_NAME • 76, *83*, 86, 103, 106, 218
SPECIFIC_SCHEMA • 76, 83, 86, 103, 106, 218
specified by • 15, 33, 38, 73, 74, 80, 82, 85, 87, 102, 107, 111, 122, 123, 143, 152, 156, 159, 163, 165, 170, 179, 184, 191, 221

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

specify • 15, 25, 40, 51, 56, 85, 119, 120, 124, 127, 129, 137, 144, 146, 147, 148, 152, 153, 156, 157, 158, 163, 165, 171, 180, 184, 185, 205, 206, 207, 208, 210, 211, 215, 216, 227
SQL • *80*, *82*, *83*, *86*, *101*, *111*, *133*, *134*, *150*, *151*, *152*, *155*, *156*, *157*, *158*, *161*, *162*, *163*, *164*, *165*, *167*, *168*, *169*, *170*, *172*, *173*, *174*, *175*, *176*, *177*, *178*, *179*, *181*, *182*, *183*, *184*
SQL-agent • 55, 60, 188, 189
SQL argument • 97, 98, 105
<SQL argument> • 97, 98
<SQL argument list> • 98
SQL-client • 15, 16, 19, 20, 24, 25, 26, 28, 29, 33, 35, 55, 56, 100, 101, 116, 118, 119, 123, 124, 126, 127, 129, 137, 143, 144, 217, 222
SQL-client module • 15, 16, 19, 20, 24, 25, 26, 28, 33, 35, 55, 56, 100, 101, 116, 118, 119, 123, 124, 126, 127, 129, 137, 143, 144, 217, 222
<SQL-client module definition> • 15, 16, 19, 20, 24, 25, 26, 28, 35, 55, 56, 100, 101, 116, 118, 119, 123, 124, 126, 127, 129, 137, 143, 144, 217, 222
<SQL condition> • ***145***, 148, 208
SQL-connection • 19, 21, 22, 27
<SQL connection statement> • 29, *187*, 188
<SQL control statement> • *91*, 92
SQL-data • 16, 17, 19, 20, 21, 22, 45, 46, 49, 190
<SQL data statement> • 60
<SQL diagnostics statement> • 29
<SQL dynamic data statement> • 27, ***59***, 60, 217
<SQL dynamic statement> • 15, 35, 52, 53, ***59***, 60, 107, 111, 204
SQL-environment • 28
SQLEXCEPTION • *145*, 148, 207
<SQL executable statement> • ***59***
SQL-implementation • 27, 60, 190, 228
SQL-invoked method • 110
SQL-invoked procedure • 121, 122, 199
SQL-invoked routine • 15, 43, 53, 121, 143
<SQL-invoked routine> • 53, *133*, 143
<SQL language character> • 136
SQL parameter • 37, 78, 98, 106, 110, 116, 139, 141, 187
SQL parameter name • 141
<SQL parameter name> • 141
SQL parameter reference • 187
SQL parameters • 37, 116
SQL-path • 15, 28, 29, 68, 97, 218
<SQL-path characteristic> • ***68***, 97
<SQL prefix> • ***133***, *134*, 135
<SQL procedure statement> • 15, 19, 20, 26, 59, 60, 92, *133*, 139, 140, 141, 142, 143, 146, 147, 187, 204, 222
SQL routine • 53
<SQL routine body> • 53
SQL-schema • 19, 20, 22, 27, 60, 190, 217
<SQL schema statement> • 39, *91*, *187*, 189
SQL-server • 26, 29
SQL-session • 15, 16, 17, 19, 20, 22, 25, 26, 27, 28, 29, 33, 34, 35, 37, 38, 55, 60, 65, 66, 67, 68, 69, 88, 109, 114, 122, 188, 189, 217, 218
SQL-session module • 15, 25, 26, 28, 29, 33
<SQL session statement> • ***59***, *91*, *187*
SQL-session user identifier • 27, 188
SQLSTATE • *57*, *58*, 137, 139, 140, 143, *145*, 146, 147, 148, *149*, 153, 199, 208
<SQLSTATE char> • ***145***, 146
<SQLSTATE class value> • ***145***, 146, 147, 148
SQL-statement • 1, 5, 15, 16, 18, 19, 20, 21, 22, 24, 26, 27, 28, 52, 56, 59, 78, 100, 136, 137, 145, 188, 193, 194, 222
<SQL statement name> • 5, **34**, 36, *91*, 99, *100*, *101*, *116*, 204
<SQL statement variable> • 5, 18, ***91***, 92, *118*
<SQLSTATE subclass value> • ***145***, 146, 147
<SQL terminator> • ***133***, *134*, 135
SQL-transaction • 16, 19, 20, 22, 27, 60, 63, 99, 144, 189, 190, 217, 221
<SQL transaction statement> • *91*, *187*
<SQL variable name> • 141
SQL variable reference • 187
SQLWARNING • *145*, 148, 207
SQL_ERROR • 57
SQL_IDENTIFIER • 38, 75, 76
SQL_LANGUAGES • 197
SQL_LANGUAGE_BINDING_STYLE • 197
SQL_LANGUAGE_PROGRAMMING_LANGUAGE • 197
SQL_LANGUAGE_SOURCE • 197
SQL_LANGUAGE_YEAR • 197
<start transaction statement> • 189
STATE • *57*, *58*, 137, 139, 140, 143, *145*, 146, 147, 148, *149*, 153, 199, 208
STATEMENT • *57*
<statement cursor> • ***121***
<statement information item> • 193
<statement information item name> • ***193***
<statement name> • 25, ***34***, 35, 63, 99, 100, 101, 116, 118, *119*, 123, 127, 129, 221
<statement or declaration> • ***133***, 143, 144
STATIC • 15, 26, *55*, 56, *133*, 208
Store assignment • 46
strings • 23, 158
<string value expression> • 92
<string value function> • 93
STRUCTURE • *31*, *101*
structured type • 120, 153, 160, 166, 171, 175, 180, 185, 211, 212, 213
STYLE • 197
subfield • 92, 94, 95, 96, 97, 225
subject routine • 68, 97, 98, 105
subject table • 127, 129
subordinate descriptor areas • 71, 72, 88, 114
subordinate descriptors • 71, 102, 103
<subquery> • 191
SUBSTRING • 93
subtype • 69

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

successful completion • 147, 190
syntax error or access rule violation • 92, 97, 116, 118, 188
<system descriptor statement> • ***59***
SYSTEM_USER • 191

— T —

table • 5, 10, 11, 12, 20, 21, 22, 26, 28, 39, 41, 42, 50, 59, 61, 72, 95, 96, 102, 105, 111, 116, 122, 123, 125, 127, 129, 130, 131, 132, 133, 136, 139, 143, 146, 168, 187, 191, 192, 197, 199, 200, 225, 227, 228
table constraint • 197
<table expression> • 42
<table name> • 116, 127, 129, 136
<table or query name> • 191
<table reference> • 41, 127, 129, 131, 132
<table subquery> • 96, 225
<table value constructor> • 95
target • 25, 37, 38, 45, 46, 61, 73, 74, 83, 84, 85, 87, 94, 97, 98, 99, 111, 112, 113, 114, 116, 124, 127, 129, 131, 132, 143, 145, 146, 193, 199, 200, 221
<target specification> • 25, ***37***, 38, 61, 98, *111*, 112, 113, 114
<target table> • *127*, *129*, *131*, *132*
target table disagrees with cursor specification • 127, 129, 200
temporary • 20, 21, 22, 28, 116, 133, 136, 139, 143, 187
temporary table • 20, 21, 22, 28, 116, 136, 139, 143
<temporary table declaration> • 20, 21, 22, 28, *133*, 136, 139, 143, *187*
TIME • 23, 77, 78, 89, 96
<time precision> • 72, 105
TIMESTAMP • 23, 77, 78, 89, 96
<timestamp precision> • 72, 105
TM • 118, 147
TO • 78, 89, 94, 97, *145*, 146, 147, 148, *188*
<token> • 31, 135
TOP_LEVEL_COUNT • *31*, 75, *83*, 86, 102
to-sql function • 110, 144
transaction • 6, 16, 18, 19, 20, 22, 27, 60, 63, 91, 99, 144, 187, 188, 189, 190, 217, 219, 221
transaction-initiating • 18, 19, 27, 187, 188, 189
transaction-initiating statement • 18
transaction state • 18, 19, 20, 22, 60, 189, 190
transform functions • 28, 69, 92
<transform group characteristic> • ***69***, 97
<transform group specification> • *134*
transforms • 23
trigger • 8, 52
<trigger definition> • 52
triggered action • 52
<triggered action> • 52
<triggered SQL statement> • 52
TRIM • *65*, *66*, *67*, *68*, *69*, *80*, *99*, *121*
type • 11, 23, 25, 26, 28, 29, 35, 37, 38, 45, 47, 57, 62, 65, 66, 67, 68, 69, 71, 72, 73, 74, 75, 76, 77, 78, 80, 84, 86, 87, 88, 89, 91, 92, 93, 94, 95, 96, 97, 103, 104, 105, 108, 109, 110, 112, 113, 114, 118, 120, 136, 137, 138, 139, 140, 141, 144, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 193, 199, 208, 209, 210, 211, 212, 213, 214, 215, 218, 219, 221, 225
Type • 75, 77, 78, 84, 86, 193
TYPE • *69*, 71, 72, 73, 74, 76, 77, *84*, 85, 87, 88, 89, 103, 104, 105, 108, 114, 115, *150*, *151*, *152*, *155*, *156*, *157*, *158*, *161*, *162*, *163*, *164*, *165*, *167*, *168*, *169*, *170*, *172*, *173*, *174*, *175*, *176*, *177*, *178*, *179*, *181*, *182*, *183*, *184*
typed table • 95
types • 11, 23, 25, 29, 38, 47, 69, 71, 74, 75, 76, 77, 78, 84, 86, 88, 92, 94, 95, 96, 97, 103, 109, 113, 120, 141, 152, 153, 160, 166, 171, 175, 180, 185, 210, 211, 212
TYPE_NAME • 72, 74, 76, 84, 87, 90, 105

— U —

undefined • 47, 80, 85, 87, 92, 94, 97, 199, 221
undefined DATA value • 85, 199
underlying table • 127, 129, 131, 132
UNNAMED • 76, *84*, 86, 103, 104, 221
<unqualified schema name> • 34, 66
<unsigned integer> • *145*, 146, 147, 148
<updatability clause> • 120, 130, 205, 208, 211
updatable • 123, 127, 129, 130, 199
updatable column • 130, 199
updatable cursor • 123, 127, 129
UPDATE • 120, *129*, *132*, 194, 205, 208, 211
<update source> • 97
<update statement: positioned> • 20, 21, 23, 130, 132, 144
<update statement: searched> • 19, 21, 22, *91*, *187*
<update target> • 97
USAGE • *161*, *162*, *163*, *164*, *165*, 219
USER • 72, 74, 76, 77, 84, 87, 89, 90, 105, 191
user-defined • 23, 28, 29, 37, 38, 62, 69, 72, 74, 92, 105, 109, 113, 114, 136, 139, 140, 141, 144, 149, 150, 151, 152, 153, 155, 156, 157, 158, 160, 161, 162, 164, 165, 166, 167, 169, 171, 172, 173, 174, 175, 176, 178, 180, 181, 183, 185, 212, 213, 214
user-defined type • 28, 29, 38, 62, 69, 72, 74, 92, 105, 109, 113, 114, 136, 139, 140, 141, 144, 151, 152, 153, 157, 158, 160, 164, 165, 166, 169, 171, 174, 175, 178, 180, 183, 185, 212, 213, 214
<user-defined type> • 29, *37*, 38, *69*, 136, *150*, 151, 152, *155*, *156*, 158, *162*, 164, 165, *167*, 169, *172*, 174, *176*, 178, *181*, 183
user-defined type locator variable • 62, 139, 141, 152, 153, 158, 160, 165, 166, 169, 171, 174, 175, 178, 180, 183, 185, 212, 213

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

<user-defined type name> • 74, 105, 136, 139, 141, *150*, 153, 160, *162*, 166, *167*, 171, *173*, 175, *176*, 180, *181*, 185, 212, 213
user-defined types • 141
user identifier • 27, 188
USER_DEFINED_TYPE_CATALOG • 72, 74, 76, *84*, 87, 89, 105
USER_DEFINED_TYPE_NAME • 72, 74, 76, *84*, 87, 90, 105
USER_DEFINED_TYPE_SCHEMA • 72, 74, 76, *84*, *87*, *89*, 105
USING • *57*, *101*, *107*
<using argument> • ***107***, 109
<using arguments> • ***107***, 109
using clause does not match dynamic parameter specifications • 107, 108, 199
using clause does not match target specifications • 111, 112, 199
using clause required for dynamic parameters • 116, 123, 199
using clause required for result fields • 116, 199
<using descriptor> • ***101***, *107*
<using input descriptor> • ***107***, 108

— V —

valid • 3, 25, 46, 60, 63, 65, 66, 67, 68, 69, 71, 72, 73, 80, 82, 84, 87, 88, 89, 98, 99, 100, 101, 102, 107, 108, 111, 112, 116, 121, 122, 123, 127, 129, 137, 143, 144, 146, 150, 152, 156, 158, 163, 165, 168, 170, 173, 175, 179, 182, 184, 189, 190, 199, 200, 221
VALUE • *83*, *86*, 165, 219
<value expression> • 92, 93, 94, 95, 97, 106, 120, 191, 211
<value expression primary> • *40*, 94
<value specification> • 37, *65*, *66*, *67*, *68*, *69*, 97, 98, 187, 191, 219
VARCHAR • *154*, *155*, 156, 159, *172*, 174, *175*
variable-length • 159, 173
VARYING • 23, 72, 73, 77, 88, 92, 93, 97, 115, *154*, *155*, 156, *157*, 159, 173, *181*, 182, *184*, 185, 210
view • 8, 39, 50, 79, 123
view definition • 50
<view definition> • 50
viewed table • 50

— W —

warning • 27, 103, 190, 200
WHENEVER • *145*
<when operand> • 95
WHERE • *127*, *129*, *131*, *132*
WITH • 77, 78, *80*, 89, 96, *101*, 102, 103, *121*, 218
WITHOUT • 77, 78, *101*, 119, 120, 216
without an intervening • 34

— Y —

YEAR • 78, 89, 94, 197

— Z —

ZONE • 77, 78, 89, 96

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ICS 35.060

Price based on 228 pages

© ISO/IEC 1999 – All rights reserved

--`,,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale