ČESKÁ TECHNICKÁ NORMA

ICS 35.060 **Duben 2005**

Informační technologie – Databázové jazyky – SQL – Část 5: Vazby hostitelského jazyka (SQL/Vazby)

ČSN ISO/IEC 9075-5 OPRAVA 2

36 9178

idt ISO/IEC 9075-5:1999/Cor. 2:2003

Technical corrigendum 2

Tato oprava přejímá anglickou verzi opravy ISO/IEC 9075-5:1999/Cor.: 2:2003

This Corrigendum implements English version of Corrigendum ISO/IEC 9075-5:1999/Cor.: 2:2003

Opravu 2 této mezinárodní normy připravila společná technická komise ISO/IEC JTC 1, *Informační technologie*, subkomise SC 32, *Správa dat a výměna*. Tato oprava ISO/IEC 9075-5:1999/Cor. 2:2003 ruší a nahrazuje ISO/IEC 9075-5:1999/Cor. 1:2000.

ČSN ISO/IEC 9075-5 (36 9178) Informační technologie – Databázové jazyky – SQL – Část 5: Vazby hostitelského jazyka (SQL/Vazby) z listopadu 2001 se opravuje takto:

Oznámení o účelu zdůvodnění:

Oznámení obsahuje označení zdůvodňující každou změnu ISO/IEC 9075.

Objasňuje uživateli této normy, proč bylo rozhodnuto změnit původní formulaci.

V mnoha případech jsou to ediční důvody nebo vysvětlení formulace, v některých případech se jedná o opravu chyby nebo opomenutí v původní formulaci.

Poznámky k číslování

Kde tato oprava zavádí novou syntaxi, přístup, všeobecná pravidla a pravidla shody, nová pravidla byla očíslována následovně:

Pravidla vložená mezi, například, Pravidla 7) a 8) jsou číslována 7.1), 7.2), atd. [nebo 7) a.1), 7) a.2), atd.].

Pravidla vložená před Pravidlo 1 jsou číslována 0.1), 0.2), atd.

Kde oprava zavádí nové pododstavce, nové pododstavce byly očíslovány následovně:

Pododstavce vložené mezi, například, Pododstavec 4.3.2 a 4.3.3 jsou číslovány 4.3.2a, 4.3.3b, atd.

Pododstavce vložené před, například, 4.3.1 jsou číslovány 4.3.0, 4.3.0a, atd.

Oprava 2

Contents

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>

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

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”.

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>
      | <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) $\boxed{\text{Add after SR 6) b) iii)}}$ 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) $\boxed{\text{Add after GR 7) b) ii)}}$ 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) $\boxed{\text{Insert after SR4)}}$ 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) $\boxed{\text{Insert after GR5)}}$ 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>.

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 forthe 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 <scopeoption> immediately contained in <descriptor name>.

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*.

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) $\leq k \leq n$:

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

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 F8 31, “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.*

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).

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) ≤ *j* ≤ *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) ≤ / ≤ *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) ≤ *k* ≤ *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) ≤ *l* ≤ 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) ≤ *j* ≤ *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) ≤ *k* ≤ *b*, be the <groupname>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) ≤ *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 SVIO_c TUIO_c ;
        SET SVI_1 = TSIN_1 ( CAST ( PNI_1 AS TTI_1 ))
        .
        .
        .
        SET SVI_a = TSIN_a ( CAST ( PNI_a AS TTI_a )) ;
        SET SVIO_1 = TSION_1 ( CAST ( PNIO_1 AS TTIO_1 )) ;
        .
        .
        .
        SET SVIO_c = TSION_c (CAST ( PNIO_c AS TTIO_c )) ;
        NES;
        SET PNO_1 = CAST ( FSON_1 ( SVO_1 ) AS TSO_1 ) ;
        .
        .
        .
        SET PNO_b = CAST ( FSON_b ( SVO_b ) AS TSO_b ) ;
        SET PNIO_1 = CAST ( FSION_1 ( SVIO1 ) AS TSIO_1 ) ;
        .
        .
        .
        SET PNIO_c = CAST ( FSION_c ( SVIO_c ) AS TSIO_c ) ;
    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 F4 61, “Named character sets”, an <embedded SQL declare section> shall not contain 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.

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 datatype 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.

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 hostidentifier> *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.

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 theBNF 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:

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 ofthe <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:

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.

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).

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

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>.

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 F4 61, “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 F4 61, “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.

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 Cprogram>":

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

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.

U p o z o r n ě n í : Změny a doplňky, jakož i zprávy o nově vydaných normách jsou uveřejňovány ve Věstníku Úřadu pro technickou normalizaci, metrologii a státní zkušebnictví.

ČSN ISO/IEC 9075-5 OPRAVA 2

Vydal: ČESKÝ NORMALIZAČNÍ INSTITUT, Praha
Vytiskl: XEROX CR, s.r.o.
Rok vydání 2005, 24 stran
Distribuce: Český normalizační institut, Hornoměcholupská 40, 102 04 Praha 10

72008 Cenová skupina 412

8 590963 720081