ISO/IEC JTC 1/SC 32 N 1214

Date: 2004-12-16

REPLACES: --

ISO/IEC JTC 1/SC 32

Data Management and Interchange

Secretariat: United States of America (ANSI)
Administered by Pacific Northwest National Laboratory on behalf of ANSI

DOCUMENT TYPE	Final Text Submitted for COR Publication
TITLE	Technical Corrigenda – ISO/IEC 9075-13 Information technology - Database Languages - SQL - Part 13: SQL/JRT (for SQL:200n)
SOURCE	SC 32 Secretariat
PROJECT NUMBER	1.32.03.05.99.00
STATUS	This is sent to ITTF for publication
REFERENCES	
ACTION ID.	ITTF
REQUESTED ACTION	
DUE DATE	
Number of Pages	6
LANGUAGE USED	English
DISTRIBUTION	P & L Members SC Chair WG Conveners and Secretaries

Douglas Mann, Secretariat, ISO/IEC JTC 1/SC 32

Pacific Northwest National Laboratory *, 13667 Legacy Circle Apt H, Herndon, VA, 20171, United States of America

Telephone: +1 202-566-2126; Facsimile; +1 202-566-1639; E-mail: MannD@battelle.org

available from the JTC 1/SC 32 WebSite http://www.jtc1sc32.org/

*Pacific Northwest National Laboratory (PNL) administers the ISO/IEC JTC 1/SC 32 Secretariat on behalf of ANSI

ISO/IEC JTC 1/SC 32

Date: 2004-12-10

COR ISO/IEC 9075-13:2004 (E)

ISO/IEC JTC 1/SC 32/WG 3

Nederlands Normalisatie Instituut (NNI)

Information technology — Database languages — SQL —

Part 13:
SQL Routines & Types Using the Java™ Programming Language (SQL/JRT)

TECHNICAL CORRIGENDUM 1

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

Partie 13: Routines et Types de SQL Utilisant le Langage de Programmation de Java™ (SQL/JRT)

RECTIFICATIE TECHNIQUE 1

Document type: Corridenga
Document subtype: Technical Corrigendum (COR)
Document stage: (5) IS Publication
Document language: English

Statement of purpose for rationale:

A statement indicating the rational 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 Subclause 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.

Contents

Page

Tables

Table

Page

Information technology — Database languages — SQL —

Part 13:
SQL Routines & Types Using the Java™ Programming Language (SQL/JRT)

TECHNICAL CORRIGENDUM 1

Foreword

1. *Rationale: Remove incorrect reference to obsolete part.*

   In the 7th paragraph, delete the 5th bullet.

2 Normative references

2.1 JTC1 standards

1. *Rationale: Correct references to IS rather than FCD documents.*

   Replace the references [Framework], [Foundation], [OLB], and [Schemata] with:

   [Framework] ISO/IEC 9075-1:2003, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework).*

   [Foundation] ISO/IEC 9075-2:2003, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)*

   [OLB] ISO/IEC 9075-10:2003, *Information technology — Database languages — SQL — Part 10: Object Language Bindings (SQL/OLB)*

   [Schemata] ISO/IEC 9075-11:2003, *Information technology — Database languages — SQL — Part 11: Information and Definition Schemas (SQL/Schemata)*

4 Concepts

4.3 Java class name resolution

1. *Rationale: Describe a JAR's SQL-Java path and its relationship to a JAR manifest's* `Class-Path` *attributes.*

   Replace the second paragraph with:

   When JARs appear in the CLASSPATH, an ability exists for further effective extension of that CLASSPATH. Additional JARs will be included in the class resolution process when a JAR in the CLASSPATH has a manifest specifying one or more `Class-Path` attributes. A `Class-Path` attribute provides *relative URLs* of additional JARs. These `Class-Path` attribute URLs are *relative* to the source, for example the directory containing the JAR whose manifest is then being processed. A full URL, for example a `file:/` or `http://` format URL, is not allowed in a `Class-Path` attribute. The JARs enumerated by `Class-Path` attributes extend the CLASSPATH.

   When a JVM is transitioned to being effectively within an SQL-environment, the problem of managing the JVM's class name resolution continues to exist, but with a change in emphasis. One important change is that an installed JAR manifest's `Class-Path` attributes cannot be honored. No relative URL has meaning when the source of the current JAR is given by a <catalog name>, <unqualified schema name>, and <jar id>. To allow the creators of Java applications a greater degree of control over class name resolution, and the added security associated with that control, a `Class-Path` attribute-like mechanism is defined to be a property of the JARs containing the Java applications, rather than as an environmental variable of the current session (such as, for example, CURRENT_PATH for dynamic statements). This mechanism, referred to as a JAR's SQL-Java path, provides a means for owners of installed JARs to control the class resolution process that the CLASSPATH and `Class-Path` attributes give users and creators of JARs outside an SQL-environment. But, note that these two mechanisms are only similar, they are not identical. If, while an external Java routine is being executed, a previously unseen class reference is encountered, that class is searched for in the JAR containing the definition of the currently executing class, and, if it is not found, the class will be sought in the manner specified by the SQL-Java path associated with that JAR (if any).

4.6 Unhandled Java exceptions

1. *Rationale: Remove the requirement that an SQLSTATE must be of class "38", external routine exception.*

   Replace the 2nd paragraph with the following:

   The message text may be specified in the Java exception specified in the Java `throw` statement. If the Java exception is an instance of `java.sql.SQLException`, or a subtype of that type, then it may also specify an SQLSTATE value. If the Java exception is not an instance of `java.sql.SQLException`, or if that exception does not specify an SQLSTATE value, then the default SQL exception condition for an uncaught Java exception is raised. See Subclause 15.1, "Class and subclass values for uncaught Java exceptions".

4.10 Privileges

1. *Rationale: Cannot have a privilege on a trigger, and there is no such thing as an SQL-invoked routine module.*

   Replace 2nd paragraph, 1st bullet

   — The identification of the base table, view, column, domain, character set, collation, transliteration, user-defined type, table/method pair, SQL-invoked routine, sequence generator, or JAR that the descriptor describes.

8 Additional common elements

8.2 <SQL Java path>

1. *Rationale: Make reference to a JAR's descriptor, and clarify the relevance of a JAR manifest's* `Class-Path` *attributes.*

   Replace General Rules 1) with:

   1) When a Java class *CJ* in a JAR *J* is executed in an SQL-implementation, let *P* be the SQL-Java path from *J*'s JAR descriptor.

      NOTE 8.1 — A JAR descriptor's SQL-Java path is set by an invocation of the `SQLJ.ALTER_JAVA_PATH` procedure.

2. *Rationale: Make reference to a JAR's descriptor, and clarify the relevance of a JAR manifest's* `Class-Path` *attributes.*

   Replace the lead text of General Rules 2) with:

   2) No `Class-Path` attribute in *J*'s manifest affects class resolution. Every static or dynamic reference in *CJ* to a class with the name *CN* that is not a system class and is not contained in *J* is resolved as follows.

      For each <path element> *PE* (if any) in *P*, in the order in which they were specified:

8.3 <routine invocation>

1. *Rationale: Clarify the invocation of the Syntax Rules of Subclause 8.6, "Java routine signature determination".*

   Replace Syntax Rule 1) b) with:

1) ...

b) If validation of the <Java parameter declaration list> has been implementation-defined to be performed by <routine invocation>, then the Syntax Rules of Subclause 8.6, "Java routine signature determination", are applied with <routine invocation>as the *syntactic element*, 0 (zero) as the *method specification index*, and *SR* as the *subject routine*.

2. *Rationale: Clarify the invocation of Subclause 9.2, "Store assignment".*

Replace General Rule 7) c) i) 4)with:

7) ...

c) ...

i) ...

4) Let *RESULT* be an arbitrary site of declared type *ERT*. The rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2 are applied with *SSFV* as *SOURCE* and *RESULT* as *TARGET*. The result of the <routine invocation> is the value of *RESULT*. No further General Rules of this Subclause are applied.

9 Schema definition and manipulation

9.4 <user-defined type definition>

1. *Rationale: Clarify the invocation of the Syntax Rules of Subclause 8.6, "Java routine signature determination".*

Replace Syntax Rule 14) with:

14) Insert after SR 9)b)xv) If *UDT* is an external Java data type and validation of the <Java parameter declaration list> has been implementation-defined to be performed by <user-defined type definition>, then the Syntax Rules of Subclause 8.6, "Java routine signature determination" are applied with <user-defined type definition>as the *syntactic element*, *i* as the *method specification index*, and no *subject routine*.

9.8 <SQL-invoked routine>

1. *Rationale: Clarify the invocation of the Syntax Rules of Subclause 8.6, "Java routine signature determination".*

Replace Syntax Rule 16) with:

16) Insert after SR 21) If *R* is an external Java routine, and validation of the <Java parameter declaration list> has been implementation-defined to be performed by <SQL-invoked routine>, then the Syntax Rules of Subclause 8.6, "Java routine signature determination", are applied with the <SQL-invoked routine> as the *syntactic element*, 0 (zero) as the *method specification index*, and no *subject routine*.

10 Access control

10.3 <revoke statement>

1. *Rationale: Adjust references to rules which have moved in Part 2.*

   Delete Syntax Rules 1), 2), 3), 4), and 5).

2. *Rationale: Adjust references to rules which have moved in Part 2.*

   Insert the following General Rules:

   0.1) Replace GR 0.5) b) i) 3) D) *P* and *D* are both usage privilege descriptors. The action and the identified domain, character set, collation, transliteration, user-defined type, sequence generator, or JAR of *P* are the same as the action and the identified domain, character set, collation, transliteration, user-defined type, sequence generator, or JAR of *D*, respectively.

   0.2) Insert after GR 0.24) b) *DT* is an external Java data type and the revoke destruction action would result in *A1* no longer having in its applicable privileges USAGE on the JAR whose <jar name> is contained in the <jar and class name> of the descriptor of *DT*.

   0.3) Insert after GR 0.28) Let *JR* be any JAR descriptor included in *S1*. *JR* is said to be *impacted* if the revoke destruction action would result in *A1* no longer having in its applicable privileges USAGE privilege on a JAR whose name is contained in a <resolution jar> contained in the SQL-Java path of *JR*.

   0.4) Insert after GR 0.29) r) If *RD* is an external Java routine, USAGE on the JAR whose <jar name> is contained in <external Java reference string> contained in the <external routine name> of the descriptor of *RD*.

   0.5) Insert after GR 0.31) If RESTRICT is specified, and there exists an impacted JAR, then an exception condition is raised: *dependent privilege descriptors still exist*.

11 Built-in procedures

11.1 SQLJ.INSTALL_JAR procedure

1. *Rationale: Create an installed JAR's JAR descriptor.*

   Insert the following General Rule:

   6.1) A JAR descriptor is created that describes the JAR being installed. The JAR descriptor includes the name of the JAR and an empty SQL-Java path.

11.3 SQLJ.REMOVE_JAR procedure

1. *Rationale: Remove the definition of the unused references old JAR and old classes.*

   Replace General Rule 4) with:

   4) If there is not an installed JAR> with <jar name> *JN*, then an exception condition is raised: *Java DDL — attempt to remove uninstalled JAR*. Equivalence of <jar name>s is determined by the rules for equivalence of identifiers as specified in Subclause 5.2, "<token> and <separator>", in ISO/IEC 9075-2.

2. *Rationale: Add a name, JR, for use when referencing the JAR being removed.*

   Insert the following General Rule:

   4.1) Let *JR* be the JAR identified by *JN*.

3. *Rationale: Use the reference, JR, for the JAR being removed.*

   Replace General Rule 5) with:

   5) If the value of the `undeploy` parameter is not 0 (zero), and if *JR* contains one or more deployment descriptor files, then the remove actions implied by those instances are performed in the reverse of the order in which the deployment descriptor files appear in the manifest.

4. *Rationale: Use the reference, JR, for the JAR being removed.*

   Replace General Rule 7) with:

   7) If there are any dependent SQL routines of *JR*, then an exception condition is raised: *Java DDL — invalid class deletion*.

5. *Rationale: Use the reference, JR, for the JAR being removed.*

   Replace General Rule 9) with:

   9) If there are any dependent SQL types of *JR*, then an exception condition is raised: *Java DDL — invalid class deletion*.

6. *Rationale: Use the reference, JR, for the JAR being removed.*

   Insert the following General Rules:

   9.1) Let the *dependent JARs* of a JAR be the JARs whose descriptors specify an SQL-Java path that immediately contains one or more <path element>s whose contained <jar name> is equivalent to the name of that JAR.

   9.2) If there are any dependent JARs of *JR*, then an exception condition is raised: *Java DDL — invalid JAR removal*.

7. *Rationale: Use the reference, JR, for the JAR being removed.*

   Replace General Rules 10) and 11) with:

   10) *JR* and all visible and non-visible classes contained in it are deleted.

   11) The USAGE privilege on *JR* is revoked from all users that have it.

8. *Rationale: Destroy the descriptor of the JAR being removed.*

   Insert the following General Rule:

   11.1) The descriptor of *JR* is destroyed.

11.4 SQLJ.ALTER_JAVA_PATH procedure

1. *Rationale: Add a name, JR, for use when referencing the JAR whose SQL-Java path is being altered, and a name, P, for use when referencing the value of the* `path` *parameter.*

   Insert the following General Rules:

   3.1) Let *JR* be the JAR identified by *JN*.

   3.2) Let *P* be the value of the `path` parameter.

2. *Rationale: Clarify the process of qualifying unqualified JAR names appearing in the path parameter.*

   Replace General Rules 4) and 5) with

4) If *P* does not conform to the format for <SQL Java path>, then an exception condition is raised: *Java DDL — invalid path*.

5) The current catalog and schema at the time of the call to the SQLJ.ALTER_JAVA_PATH procedure are the default, respectively, for each omitted <catalog name> and <schema name> in the <resolution jar>s contained in *P*. For each <path element> *PE* (if any) in *P*, *PE*'s <resolution jar> is updated to reflect the defaults for omitted <catalog name>s and <schema name>s.

3. *Rationale: Raise an exception condition when a JAR's SQL-Java path is self-referencing, and raise a completion condition if a JAR's SQL-Java path can't be represented in the Information Schema without truncation.*

Insert the following General Rules:

5.1) For each <path element> *PE* (if any) in *P*, let *RJ* be the <jar name> contained in the <resolution jar> contained in *PE*. If *RJ* is equivalent to *JN*, then an exception condition is raised: *Java DDL — self-referencing path*.

5.2) If *P* cannot be represented in the Information Schema without truncation, then a completion condition is raised: *warning — SQL-Java path too long for information schema*.

NOTE 71.1 — The Information Schema is defined in ISO/IEC 9075-11.

4. *Rationale: Raise an exception condition when a JAR's SQL-Java path is self-referencing, and raise a completion condition if a JAR's SQL-Java path can't be represented in the Information Schema without truncation.*

Replace General Rule 6) with:

6) The value of *P* is placed in the SQL-Java path of the JAR descriptor of *JR*, replacing the current SQL-Java path (if any) associated with *JR*.

13 Information Schema

13.2 JARS view

1. *Rationale: Add missing column to the definition of the JARS view.*

In the Definition, replace the view definition with:

```
CREATE VIEW JARS AS
  SELECT JAR_CATALOG, JAR_SCHEMA, JAR_NAME, JAVA_PATH
  FROM DEFINITION_SCHEMA.JARS
  WHERE ( JAR_CATALOG, JAR_SCHEMA, JAR_NAME, 'JAR' ) IN
          ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE
            FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES
```

```
            WHERE GRANTEE IN
                  ( 'PUBLIC', CURRENT_USER )
                OR
                  GRANTEE IN
                  ( SELECT ROLE_NAME
                    FROM ENABLED_ROLES ) )
        AND
           JAR_CATALOG =
           ( SELECT CATALOG_NAME
             FROM INFORMATION_SCHEMA_CATALOG_NAME ) ;
```

14 Definition Schema

14.2 JARS base table

1. *Rationale: Add missing column to the definition of the JARS base table.*

   In the Definition, replace the table definition with:

```
CREATE TABLE JARS (
    JAR_CATALOG            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    JAR_SCHEMA             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    JAR_NAME               INFORMATION_SCHEMA.SQL_IDENTIFIER,
    JAVA_PATH              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT JARS_PRIMARY_KEY
      PRIMARY KEY ( JAR_CATALOG, JAR_SCHEMA, JAR_NAME ),
    CONSTRAINT JAR_FOREIGN_KEY_SCHEMATA
      FOREIGN KEY ( JAR_CATALOG, JAR_SCHEMA )
        REFERENCES SCHEMATA
    )
```

2. *Rationale: Add missing column to the definition of the JARS base table.*

   In the Definition, replace the table definition with:

   2) Case:

      a) If the character representation of the SQL-Java path in the descriptor of the JAR being described can be represented without truncation, then the value of JAVA_PATH is that character representation.

      b) Otherwise, the value of JAVA_PATH is the null value.

15 Status codes

15.1 Class and subclass values for uncaught Java exceptions

1. *Rationale: Remove the requirement that an SQLSTATE must be of class "38", external routine exception.*

   Replace Rule 3) with:

   3) Case:

      a) If *E* is an instance of `java.sql.SQLException`, and the result *SS* of the Java method call `E.getSQLState()` is a five-character string, then let *C* be the first and second characters of *SS*, and let *SC* be the third, fourth, and fifth characters of *SS*.

      b) Otherwise, let *C* be '38' (corresponding to *external routine exception*) and *SC* be '000' (corresponding to *no subclass*).

15.2 SQLSTATE

1. *Rationale: Supply missing values Subcondition and Subclass values for three exception conditions and one completion condition.*

   Insert the following 4 rows, in the appropriate location for their respective Category, Condition, Class, and Subclass values, in Table 2.

Table 2 — SQLSTATE class and subclass values

Category	Condition	Class	Subcondition	Subclass
X	*Java DDL*	46	*invalid JAR removal*	00C
X	*Java DDL*	46	*invalid path*	00D
X	*Java DDL*	46	*self-referencing path*	00E
W	*warning*	01	*SQL-Java path too long for information schema*	011

Annex B

(informative)

Implementation-defined elements

1. *Rationale: Clarify the invocation of the Syntax Rules of Subclause 8.6, "Java routine signature determination".*

   Replace item 6) a) with:

   6) ...

      a) If validation of the <Java parameter declaration list> has been implementation-defined to be performed by <routine invocation>, then the Syntax Rules of Subclause 8.6, "Java routine signature determination", are applied with <routine invocation> *syntactic element*, 0 (zero) as the *method specification index*, and *SR* as the *subject routine*.

Annex E

(informative)

Routines tutorial

E.21 Exceptions

1. *Rationale: Remove the description of the restrictions imposed on SQLSTATE.*

   Replace the 7th paragraph with:

   The message text and SQLSTATE may be specified in the Java exception specified in the Java `throw` statement. If that exception does not specify an SQLSTATE, then the default SQL exception condition for an uncaught Java exception is raised. See Subclause 15.1, "Class and subclass values for uncaught Java exceptions".

