BRITISH STANDARD

BS ISO/IEC 9075-13:2002

Information technology — Database languages — SQL —

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

ICS 35.060

NO COPYING WITHOUT BSI PERMISSION EXCEPT AS PERMITTED BY COPYRIGHT LAW

BSI Documents
Document Sharing Hub
British Standards

Full document details: BS ISO/IEC 9075-13:2002

National foreword

This British Standard reproduces verbatim ISO/IEC 9075-13:2002 and implements it as the UK national standard.

The UK participation in its preparation was entrusted to Technical Committee IST/40, Data management and interchange, which has the responsibility to:

— aid enquirers to understand the text;

— present to the responsible international/European committee any enquiries on the interpretation, or proposals for change, and keep the UK interests informed;

— monitor related international and European developments and promulgate them in the UK.

A list of organizations represented on this committee can be obtained on request to its secretary.

Cross-references

The British Standards which implement international publications referred to in this document may be found in the *BSI Catalogue* under the section entitled "International Standards Correspondence Index", or by using the "Search" facility of the *BSI Electronic Catalogue* or of British Standards Online.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

This British Standard, having been prepared under the direction of the Disc Board, was published under the authority of the Standards Policy and Strategy Committee on 12 August 2002

Summary of pages

This document comprises a front cover, an inside front cover, the ISO/IEC title page, pages ii to x, pages 1 to 187, and a back cover.

The BSI copyright date displayed in this document indicates when the document was last issued.

Amendments issued since publication

Amd. No.	Date	Comments

© BSI 12 August 2002

ISBN 0 580 40238 X

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-13:2002

INTERNATIONAL STANDARD

ISO/IEC 9075-13

First edition
2002-07-01

Information technology — Database languages — SQL —

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

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

Partie 13: Routines et types utilisant le langage de programmation Java™ (SQL/JRT)

Reference number
ISO/IEC 9075-13:2002(E)

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-13:2002

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

Contents

Page

Foreword . . . viii

Introduction . . . ix

1 Scope . . . 1

2 Normative references . . . 3
2.1 JTC1 standards . . . 3
2.2 Publicly-available specifications . . . 3

3 Definitions, notations, and conventions . . . 5
3.1 Definitions . . . 5
3.1.1 Definitions provided in Part 13 . . . 5
3.1.2 Definitions taken from Java . . . 6
3.2 Notations . . . 6
3.3 Conventions . . . 7
3.3.1 Specification of built-in procedures . . . 7
3.3.2 Specification of deployment descriptor files . . . 7
3.3.3 Relationships to other parts of ISO/IEC 9075 . . . 8
3.3.3.1 Clause, Subclause, and Table relationships . . . 8
3.4 Object identifier for Database Language SQL . . . 12

4 Concepts . . . 17
4.1 The Java programming language . . . 17
4.2 SQL-invoked routines . . . 18
4.3 Java class name resolution . . . 20
4.4 SQL result sets . . . 20
4.5 Parameter mapping . . . 21
4.6 Unhandled Java exceptions . . . 22
4.7 Data types . . . 23
4.8 User-defined types . . . 23
4.8.1 User-defined type comparison and assignment . . . 26
4.8.2 Accessing static fields . . . 26
4.8.3 Converting objects between SQL and Java . . . 27
4.8.3.1 SERIALIZABLE . . . 27
4.8.3.2 SQLDATA . . . 27
4.8.3.3 Developing for portability . . . 28

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

4.9 Built-in procedures . . . 28
4.10 Privileges . . . 29
4.11 JARs . . . 29
4.11.1 Deployment descriptor files . . . 30

5 Lexical elements . . . 31
5.1 <token> and <separator> . . . 31
5.2 Names and identifiers . . . 32

6 Scalar expressions . . . 35
6.1 <method invocation> . . . 35
6.2 <new specification> . . . 36

7 Predicates . . . 37
7.1 <comparison predicate> . . . 37

8 Additional common elements . . . 39
8.1 <Java parameter declaration list> . . . 39
8.2 <SQL Java path> . . . 40
8.3 <routine invocation> . . . 42
8.4 <language clause> . . . 51
8.5 Java routine signature determination . . . 52

9 Schema definition and manipulation . . . 59
9.1 <table definition> . . . 59
9.2 <view definition> . . . 60
9.3 <user-defined type definition> . . . 61
9.4 <attribute definition> . . . 65
9.5 <alter type statement> . . . 69
9.6 <drop data type statement> . . . 70
9.7 <SQL-invoked routine> . . . 71
9.8 <alter routine statement> . . . 74
9.9 <drop routine statement> . . . 75
9.10 <user-defined ordering definition> . . . 76
9.11 <drop user-defined ordering statement> . . . 78

10 Access control . . . 79
10.1 <grant privilege statement> . . . 79
10.2 <privileges> . . . 80
10.3 <revoke statement> . . . 81

11 Built-in procedures . . . 83
11.1 SQLJ.INSTALL_JAR procedure . . . 83
11.2 SQLJ.REPLACE_JAR procedure . . . 85
11.3 SQLJ.REMOVE_JAR procedure . . . 87
11.4 SQLJ.ALTER_JAVA_PATH procedure . . . 89

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

12 Java topics . . . 91
12.1 Java facilities supported by this part of ISO/IEC 9075 . . . 91
12.1.1 Package java.sql . . . 91
12.1.2 System properties . . . 91
12.2 Deployment descriptor files . . . 92

13 Information Schema . . . 95
13.1 JAR_JAR_USAGE view . . . 95
13.2 JARS view . . . 96
13.3 METHOD_SPECIFICATIONS view . . . 97
13.4 ROUTINE_JAR_USAGE view . . . 98
13.5 TYPE_JAR_USAGE view . . . 99
13.6 USER_DEFINED_TYPES view . . . 100
13.7 Short name views . . . 101

14 Definition Schema . . . 103
14.1 JAR_JAR_USAGE base table . . . 103
14.2 JARS base table . . . 104
14.3 METHOD_SPECIFICATIONS base table . . . 105
14.4 ROUTINE_JAR_USAGE base table . . . 106
14.5 ROUTINES base table . . . 107
14.6 TYPE_JAR_USAGE base table . . . 108
14.7 USAGE_PRIVILEGES base table . . . 109
14.8 USER_DEFINED_TYPES base table . . . 110

15 Status codes . . . 113
15.1 Class and subclass values for uncaught Java exceptions . . . 113
15.2 SQLSTATE . . . 114

16 Conformance . . . 115
16.1 Claims of conformance . . . 115

Annex A SQL Conformance Summary . . . 117

Annex B Implementation-defined elements . . . 123

Annex C Implementation-dependent elements . . . 127

Annex D SQL Feature Taxonomy . . . 129

Annex E Routines tutorial . . . 131
E.1 Technical components . . . 131
E.2 Overview . . . 132
E.3 Example Java methods: region and correctStates . . . 133
E.4 Installing region and correctStates in SQL . . . 133
E.5 Defining SQL names for region and correctStates . . . 135
E.6 A Java method with output parameters: bestTwoEmps . . . 136
E.7 A CREATE PROCEDURE best2 for bestTwoEmps . . . 137

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

E.8 Calling the best2 procedure . . . 138
E.9 A Java method returning a result set: orderedEmps . . . 138
E.10 A CREATE PROCEDURE rankedEmps for orderedEmps . . . 140
E.11 Calling the rankedEmps procedure . . . 141
E.12 Overloading Java method names and SQL names . . . 141
E.13 Java main methods . . . 143
E.14 Java method signatures in the CREATE statements . . . 144
E.15 Null argument values and the RETURNS NULL clause . . . 145
E.16 Static variables . . . 147
E.17 Dropping SQL names of Java methods . . . 148
E.18 Removing Java classes from SQL . . . 148
E.19 Replacing Java classes in SQL . . . 149
E.20 Visibility . . . 150
E.21 Exceptions . . . 150
E.22 Deployment descriptors . . . 151
E.23 Paths . . . 154
E.24 Privileges . . . 156
E.25 Information Schema . . . 156

Annex F Types tutorial . . . 157
F.1 Overview . . . 157
F.2 Example Java classes . . . 157
F.3 Installing Address and Address2Line in an SQL system . . . 159
F.4 CREATE TYPE for Address and Address2Line . . . 160
F.5 Multiple SQL types for a single Java class . . . 162
F.6 Collapsing subclasses . . . 162
F.7 GRANT and REVOKE statements for data types . . . 164
F.8 Deployment descriptors for classes . . . 164
F.9 Using Java classes as data types . . . 166
F.10 SELECT, INSERT, and UPDATE . . . 166
F.11 Referencing Java fields and methods in SQL . . . 167
F.12 Extended visibility rules . . . 168
F.13 Logical representation of Java instances in SQL . . . 168
F.14 Static methods . . . 170
F.15 Static fields . . . 170
F.16 Instance-update methods . . . 171
F.17 Subtypes in SQL/JRT data . . . 173
F.18 References to fields and methods of null instances . . . 174
F.19 Ordering of SQL/JRT data . . . 175

Annex G Incompatibilities with ANSI NCITS 331 . . . 177
G.1 References . . . 177
G.2 Incompatibilities . . . 177

Index . . . 181

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

TABLES

Tables **Page**

1 Clause, Subclause, and Table relationships 8
2 System properties 91
3 SQLSTATE class and subclass values 114
4 Feature taxonomy for features outside Core SQL 129

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-13:2002

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.

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

The main task of the joint technical committee is to prepare International Standards. 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 part of ISO/IEC 9075 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 9075-13 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)*

— *Part 9: Management of External Data (SQL/MED)*

— *Part 10: Object Language Bindings (SQL/OLB)*

— *Part 11: Information and definition schemas (SQL/Schemata)*

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

Annexes A, B, C, D, E, F and G of this part of ISO/IEC 9075 are for information only.

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

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 Java routines and types.

5) Clause 5, “Lexical elements”, defines a number of lexical elements used in the definition of Java routines and types.

6) Clause 6, “Scalar expressions”, defines the elements of the language that produce scalar values.

7) Clause 7, “Predicates”, defines the predicates of the language.

8) Clause 8, “Additional common elements”, defines additional language elements that are used in various parts of the language.

9) Clause 9, “Schema definition and manipulation”, defines the schema definition and manipulation statements associated with the definition of Java routines and types.

10) Clause 10, “Access control”, defines facilities for controlling access to SQL-data.

11) Clause 11, “Built-in procedures”, defines new built-in procedures used in the definition of Java routines and types.

12) Clause 12, “Java topics”, defines the facilities supported by implementations of this part of ISO/IEC 9075 and the conventions used in deployment descriptor files.

13) Clause 13, “Information Schema”, defines viewed tables that contain schema information.

14) Clause 14, “Definition Schema”, defines base tables on which the viewed tables containing schema information depend.

15) Clause 15, “Status codes”, defines SQLSTATE values related to Java routines and types.

16) Clause 16, “Conformance”, defines the criteria for conformance to this part of ISO/IEC 9075.

17) Annex A, “SQL Conformance Summary”, is an informative Annex. It summarizes the conformance requirements of the SQL language.

18) 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.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-13:2002

19) 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.

20) Annex D, “SQL Feature 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.

21) Annex E, “Routines tutorial”, is an informative Annex. It provides a tutorial on using the features defined in this part of ISO/IEC 9075 for defining and using SQL-invoked routines based on Java static methods.

22) Annex F, “Types tutorial”, is an informative Annex. It provides a tutorial on using the features defined in this part of ISO/IEC 9075 for defining and using SQL structured types based on Java classes.

23) Annex G, “Incompatibilities with ANSI NCITS 331”, is an informative Annex. It lists the incompatibilities between this edition of this part of ISO/IEC 9075 and NCITS 331.1 and NCITS 331.2.

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 16, “Conformance”, Subclauses begin a new page. Any resulting blank space is not significant.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-13:2002

Information technology — Database languages — SQL —

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

1 Scope

This part of International Standard ISO/IEC 9075 specifies the ability to invoke static methods written in the Java™ programming language as SQL-invoked routines and to use classes defined in the Java programming language as SQL structured user-defined types. (Java is a registered trademark of Sun Microsystems, Inc.)

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

(Blank page)

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-13:2002

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9075. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, 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 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.

2.1 JTC 1 standards

ISO 8824-1:1998, *Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation*

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 9075-10:2000, *Information technology — Database languages — SQL — Part 10: Object Language Bindings (SQL/OLB)*

2.2 Publicly-available specifications

The Java Language Specification, Second Edition, Bill Joy (Editor), Guy Steele, James Gosling, and Gilad Bracha, Addison-Wesley, 2000, ISBN 0-201-31008-2.

The Java Virtual Machine Specification, Second Edition, Tim Lindholm and Frank Yellin, Addison-Wesley, 1999, ISBN 0-201-43294-3.

Java 2 Platform, Standard Edition, v1.2.2, API Specification,
`http://web2.java.sun.com/products/jdk/1.2/docs/api/`.

Java Object Serialization Specification,
`http://web2.java.sun.com/products/jdk/1.2/docs/guide/serialization/spec/serialTOC.doc.html`.

The JavaBeans™ 1.01 Specification,
`http://java.sun.com/products/javabeans/docs/spec.html`.

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

2.2 Publicly-available specifications

JDBC™ 2.0 API, Version 1.0, Seth White & Mark Hapner, Sun Microsystems, Inc., 30 May, 1999.

JDBC 2.0 Standard Extension API, Version 1.0, Seth White & Mark Hapner, Sun Microsystems, Inc., 7 December, 1998.

JDBC API Tutorial and Reference, Second Edition: Universal Data Access for the Java 2 Platform, Seth White, Maydene Fisher, Rick Cattell, Graham Hamilton, and Mark Hapner, Addison Wesley, Reading MA, 1999, ISBN 0-201-43328-1.

W3C Architecture domain: Naming and Addressing (URLs), `http://www.w3.org/Addressing/Activity.html`.

RFC 1738, *Uniform Resource Locators (URL)*, T. Berners-Lee, L. Maxinter, M. McCahill, December, 1994.

RFC 1808, *Relative Uniform Resource Locators*, R. Fielding, June, 1995.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-13:2002

3 Definitions, notations, and conventions

3.1 Definitions

3.1.1 Definitions provided in Part 13

Insert this paragraph For the purposes of this part of ISO/IEC 9075, the definitions given in ISO/IEC 9075-1, ISO/IEC 9075-2, ISO/IEC 9075-5, and ISO/IEC 9075-10, and the following definitions, apply.

a) **class file**: A file containing the compiled byte code for a Java class.

b) **default connection**: a JDBC connection to the current SQL-implementation, SQL-session, and SQL-transaction established with the data source URL `'jdbc:default:connection'`.

c) **deployment descriptor**: one or more SQL-statements that specify <install actions> and <remove actions> to be taken, respectively, by the `SQLJ.INSTALL_JAR` and `SQLJ.REMOVE_JAR` procedures and that are contained in a *deplyment descriptor file*. For example, when a JAR is installed, one or more <SQL-invoked routine>s that specify LANGUAGE JAVA and either PROCEDURE or FUNCTION and the associated <grant privilege statement>s can be specified in the deployment descriptor and executed as part of the install process.

d) **deployment descriptor file**: a text file containing deployment descriptors that is contained in a JAR, for which the JAR's manifest entry, as described by the `java.util.jar` section of *Java 2 Platform, Standard Edition, v1.2.2, API Specification*, specifies `SQLJDeploymentDescriptor: TRUE`.

e) **external Java data type**: an SQL user-defined type defined with a <user-defined type definition> that specifies an <external Java type clause>.

f) **external Java routine**: an external routine defined with an <SQL-invoked routine> that specifies LANGUAGE JAVA and either PROCEDURE or FUNCTION, or defined with a <user-defined type definition> that specifies an <external Java type clause>.

g) **installed JAR**: a JAR whose existence has been registered with the SQL-environment and whose contents have been copied into that SQL-environment due to execution of one of the procedures `SQLJ.INSTALL_JAR` and `SQLJ.REPLACE_JAR`.

h) **Java Archive (JAR)**: a zip formatted file, as described by the `java.util.zip` section of *Java 2 Platform, Standard Edition, v1.2.2, API Specification*, containing zero or more Java `class` and `ser` files, and zero or more deployment descriptor files. JARs are a normal vehicle for distributing Java programs and the mechanism specified by this International Standard to provide the implementation of external Java routines and external Java data types to an SQL-environment.

i) **JVM**: A Java Virtual Machine, as defined by *The Java Virtual Machine Specification, Second Edition*.

j) **ser file**: A file containing representations of Java objects in the form defined in *Java Object Serialization Specification*.

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

k) **subject Java class**: the Java class uniquely identified by the combination of the class's *subject Java class name* and its containing JAR.

l) **subject Java class name**: the fully-qualified package and class name of a Java class.

m) **system class**: any Java class provided by a conforming implementation of this part of ISO/IEC 9075 that can be referenced by an external Java routine or an external Java data type without that class having been included in an installed JAR.

3.1.2 Definitions taken from Java

This part of ISO/IEC 9075 makes use of the following terms defined in *The Java Language Specification, Second Edition*:

a) **block**

b) **class declaration**

c) **class instance**

d) **class variable**

e) **field**

f) **instance initializer**

g) **instance variable**

h) **interface**

i) **local variable**

j) **nested class**

k) **package**

l) **static initializer**

m) **subpackage**

This part of ISO/IEC 9075 makes use of the following terms defined in *The Java Virtual Machine Specification, Second Edition*:

a) **class file**

b) **Java Virtual Machine**

3.2 Notations

Insert this paragraph The syntax notation used in this part of ISO/IEC 9075 is an extended version of BNF ("Backus Normal Form" or "Backus Naur Form"). This version of BNF is fully described in Subclause 6.1, "Notation", of ISO/IEC 9075-1.

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

3.3 Conventions

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

3.3.1 Specification of built-in procedures

Built-in procedures are specified in terms of:

— **Function**: A short statement of the purpose of the procedure.

— **Signature**: A specification, in SQL, of the signature of the procedure. The only purpose of the signature is to specify the procedure name, parameter names, and parameter types. The manner in which these built-in procedures are defined is implementation-dependent.

— **Access Rules**: A specification in English of rules governing the accessibility of schema objects that must hold before the General Rules may be successfully applied.

— **General Rules**: A specification in English of the run-time effect of invocation of the procedure. Where more than one General Rule is used to specify the effect of an element, the required effect is that which would be obtained by beginning with the first General Rule and applying the Rules in numeric sequence unless a Rule is applied that specifies or implies a change in sequence or termination of the application of the Rules. Unless otherwise specified or implied by a specific Rule that is applied, application of General Rules terminates when the last in the sequence has been applied.

— **Conformance Rules**: A specification of how the element must be supported for conformance to SQL.

The scope of notational symbols is the Subclause in which those symbols are defined. Within a Subclause, the symbols defined in the Signature, Access Rules, or General Rules can be referenced in other rules provided that they are defined before being referenced.

3.3.2 Specification of deployment descriptor files

Deployment descriptor files are specified in terms of:

— **Function**: A short statement of the purpose of the deployment descriptor file.

— **Model**: A brief description of the manner in which a deployment descriptor file is identified within its containing JAR.

— **Properties**: A BNF specification of the syntax of the contents of a deployment descriptor file.

— **Description**: A specification of the requirements and restrictions on the contents of a deployment descriptor file.

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

3.3.3 Relationships to other parts of ISO/IEC 9075

3.3.3.1 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 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 3.1.1, “Definitions provided in Part 13”	*(none)*	*(none)*
Subclause 3.1.2, “Definitions taken from Java”	*(none)*	*(none)*
Subclause 3.2, “Notations”	Subclause 3.2, "Notation"	ISO/IEC 9075-2
Subclause 3.3, “Conventions”	Subclause 3.3, "Conventions"	ISO/IEC 9075-2
Subclause 3.3.1, “Specification of built-in procedures”	*(none)*	*(none)*
Subclause 3.3.2, “Specification of deployment descriptor files”	*(none)*	*(none)*
Subclause 3.3.3, “Relationships to other parts of ISO/IEC 9075”	*(none)*	*(none)*
Subclause 3.3.3.1, “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, “The Java programming language”	*(None)*	*(None)*
Subclause 4.2, “SQL-invoked routines”	Subclause 4.23, "SQL-invoked routines"	ISO/IEC 9075-2
Subclause 4.3, “Java class name resolution”	*(None)*	*(None)*
Subclause 4.4, “SQL result sets”	*(None)*	*(None)*
Subclause 4.5, “Parameter mapping”	*(None)*	*(None)*
Subclause 4.6, “Unhandled Java exceptions”	*(None)*	*(None)*
Subclause 4.7, “Data types”	Subclause 4.1, "Data types"	ISO/IEC 9075-2
Subclause 4.8, “User-defined types”	Subclause 4.8, "User-defined types"	ISO/IEC 9075-2

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

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 4.8.1, “User-defined type comparison and assignment”	Subclause 4.8.4, "User-defined type comparison and assignment"	ISO/IEC 9075-2
Subclause 4.8.2, “Accessing static fields”	*(None)*	*(None)*
Subclause 4.8.3, “Converting objects between SQL and Java”	*(None)*	*(None)*
Subclause 4.8.3.1, “SERIALIZABLE”	*(None)*	*(None)*
Subclause 4.8.3.2, “SQLDATA”	*(None)*	*(None)*
Subclause 4.8.3.3, “Developing for portability”	*(None)*	*(None)*
Subclause 4.9, “Built-in procedures”	*(None)*	*(None)*
Subclause 4.10, “Privileges”	Subclause 4.31.2, "Privileges"	ISO/IEC 9075-2
Subclause 4.11, “JARs”	*(None)*	*(None)*
Subclause 4.11.1, “Deployment descriptor files”	*(None)*	*(None)*
Clause 5, “Lexical elements”	Clause 5, "Lexical elements"	ISO/IEC 9075-2
Subclause 5.1, “<token> and <separator>”	Subclause 5.1, "<token> and <separator>"	ISO/IEC 9075-2
Subclause 5.2, “Names and identifiers”	Subclause 5.1, "<token> and <separator>"	ISO/IEC 9075-2
Clause 6, “Scalar expressions”	Clause 6, "Scalar Expressions"	ISO/IEC 9075-2
Subclause 6.1, “<method invocation>”	Subclause 6.11, "<method invocation>"	ISO/IEC 9075-2
Subclause 6.2, “<new specification>”	Subclause 6.24, "<new specification>"	ISO/IEC 9075-2
Clause 7, “Predicates”	Clause 8, "Predicates"	ISO/IEC 9075-2
Subclause 7.1, “<comparison predicate>”	Subclause 8.2, "<comparison predicate>"	ISO/IEC 9075-2
Clause 8, “Additional common elements”	Clause 10, "Additional common elements"	ISO/IEC 9075-2
Subclause 8.1, “<Java parameter declaration list>”	*(None)*	*(None)*
Subclause 8.2, “<SQL Java path>”	*(None)*	*(None)*
Subclause 8.3, “<routine invocation>”	Subclause 10.4, "<routine invocation>"	ISO/IEC 9075-2
Subclause 8.4, “<language clause>”	Subclause 10.2, "<language clause>"	ISO/IEC 9075-2

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

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 8.5, “Java routine signature determination”	*(None)*	*(None)*
Clause 9, “Schema definition and manipulation”	Clause 11, "Schema definition and manipulation"	ISO/IEC 9075-2
Subclause 9.1, “<table definition>”	Subclause 11.3, "<table definition>"	ISO/IEC 9075-2
Subclause 9.2, “<view definition>”	Subclause 11.21, "<view definition>"	ISO/IEC 9075-2
Subclause 9.3, “<user-defined type definition>”	Subclause 11.40, "<user-defined type definition>"	ISO/IEC 9075-2
Subclause 9.4, “<attribute definition>”	Subclause 11.41, "<attribute definition>"	ISO/IEC 9075-2
Subclause 9.5, “<alter type statement>”	Subclause 11.42, "<alter type statement>"	ISO/IEC 9075-2
Subclause 9.6, “<drop data type statement>”	Subclause 11.48, "<drop data type statement>"	ISO/IEC 9075-2
Subclause 9.7, “<SQL-invoked routine>”	Subclause 11.49, "<SQL-invoked routine>"	ISO/IEC 9075-2
Subclause 9.8, “<alter routine statement>”	Subclause 11.50, "<alter routine statement>"	ISO/IEC 9075-2
Subclause 9.9, “<drop routine statement>”	Subclause 11.51, "<drop routine statement>"	ISO/IEC 9075-2
Subclause 9.10, “<user-defined ordering definition>”	Subclause 11.54, "<user-defined ordering definition>"	ISO/IEC 9075-2
Subclause 9.11, “<drop user-defined ordering statement>”	Subclause 11.55, "<drop user-defined ordering statement>"	ISO/IEC 9075-2
Clause 10, “Access control”	Clause 12, "Access control"	ISO/IEC 9075-2
Subclause 10.1, “<grant privilege statement>”	Subclause 12.2, "<grant privilege statement>"	ISO/IEC 9075-2
Subclause 10.2, “<privileges>”	Subclause 10.5, "<privileges>"	ISO/IEC 9075-2
Subclause 10.3, “<revoke statement>”	Subclause 12.6, "<revoke statement>"	ISO/IEC 9075-2
Clause 11, “Built-in procedures”	*(None)*	*(None)*
Subclause 11.1, “SQLJ.INSTALL_JAR procedure”	*(None)*	*(None)*
Subclause 11.2, “SQLJ.REPLACE_JAR procedure”	*(None)*	*(None)*
Subclause 11.3, “SQLJ.REMOVE_JAR procedure”	*(None)*	*(None)*

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

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 11.4, “SQLJ.ALTER_JAVA_PATH procedure”	*(None)*	*(None)*
Clause 12, “Java topics”	*(None)*	*(None)*
Subclause 12.1, “Java facilities supported by this part of ISO/IEC 9075”	*(None)*	*(None)*
Subclause 12.1.1, “Package java.sql”	*(None)*	*(None)*
Subclause 12.1.2, “System properties”	*(None)*	*(None)*
Subclause 12.2, “Deployment descriptor files”	*(None)*	*(None)*
Clause 13, “Information Schema”	Clause 20, "Information Schema"	ISO/IEC 9075-2
Subclause 13.1, “JAR_JAR_USAGE view”	*(None)*	*(None)*
Subclause 13.2, “JARS view”	*(None)*	*(None)*
Subclause 13.3, “METHOD_SPECIFICATIONS view”	Subclause 20.32, "METHOD_SPECIFICATIONS view"	ISO/IEC 9075-2
Subclause 13.4, “ROUTINE_JAR_USAGE view”	*(None)*	*(None)*
Subclause 13.5, “TYPE_JAR_USAGE view”	*(None)*	*(None)*
Subclause 13.6, “USER_DEFINED_TYPES view”	Subclause 20.65, "USER_DEFINED_TYPES view"	ISO/IEC 9075-2
Subclause 13.7, “Short name views”	Subclause 20.69, "Short name views"	ISO/IEC 9075-2
Clause 14, “Definition Schema”	Clause 21, "Definition Schema"	ISO/IEC 9075-2
Subclause 14.1, “JAR_JAR_USAGE base table”	*(None)*	*(None)*
Subclause 14.2, “JARS base table”	*(None)*	*(None)*
Subclause 14.3, “METHOD_SPECIFICATIONS base table”	Subclause 21.24, "METHOD_SPECIFICATIONS base table"	ISO/IEC 9075-2
Subclause 14.4, “ROUTINE_JAR_USAGE base table”	*(None)*	*(None)*
Subclause 14.5, “ROUTINES base table”	Subclause 21.33, "ROUTINES base table"	ISO/IEC 9075-2
Subclause 14.6, “TYPE_JAR_USAGE base table”	*(None)*	*(None)*

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

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 14.7, “USAGE_ PRIVILEGES base table”	Subclause 21.50, "USAGE_ PRIVILEGES base table"	ISO/IEC 9075-2
Subclause 14.8, “USER_DEFINED_ TYPES base table”	Subclause 21.52, "USER_ DEFINED_TYPES base table"	ISO/IEC 9075-2
Clause 15, “Status codes”	Clause 22, "Status codes"	ISO/IEC 9075-2
Subclause 15.1, “Class and subclass values for uncaught Java exceptions”	*(None)*	*(None)*
Subclause 15.2, “SQLSTATE”	Subclause 22.1, "SQLSTATE"	ISO/IEC 9075-2
Clause 16, “Conformance”	Clause 8, "Conformance"	ISO/IEC 9075-1
Subclause 16.1, “Claims of conformance”	Subclause 8.2.3, "Claims of conformance"	ISO/IEC 9075-1
Annex A, “SQL Conformance Summary”	Annex A, "SQL Conformance Summary"	ISO/IEC 9075-2
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 Taxonomy”	Annex F, "SQL feature and package taxonomy"	ISO/IEC 9075-2
Annex E, “Routines tutorial”	*(None)*	*(None)*
Annex F, “Types tutorial”	*(None)*	*(None)*
Annex G, “Incompatibilities with ANSI NCITS 331”	*(None)*	*(None)*
Table 1, “Clause, Subclause, and Table relationships”	*(none)*	*(none)*
Table 2, “System properties”	*(None)*	*(None)*
Table 3, “SQLSTATE class and subclass values”	Table 27, "SQLSTATE class and subclass values"	ISO/IEC 9075-2
Table 4, “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 Subclause 6.3, "Object identifier for Database Language SQL", of ISO/IEC 9075-1 with the following additions:

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

Format

```
<invoked routine languages> ::=
         !! All alternatives from ISO/IEC 9075-1
       | <invoked Java>

<invoked Java> ::=
       8 | invokedJava <left paren> 8 <right paren>

<Part 13 yes> ::=
       <Part 13 conformance>
       <Part 13 routines>
       <Part 13 types>
       <Part 13 JAR privileges>
       <Part 13 interfaces>

<Part 13 conformance> ::=
         13
       | sqljrt2002 <left paren> 13 <right paren>

<Part 13 routines> ::=
         <Part 13 routines no>
       | <Part 13 routines yes> <routines support>

<Part 13 routines no> ::=
       0 | routinesno <left paren> 0 <right paren>

<Part 13 routines yes> ::=
       1 | routinesyes <left paren> 1 <right paren>

<Part 13 types> ::=
         <Part 13 types no>
       | <Part 13 types yes> <types support>

<Part 13 types no> ::=
       0 | typesno <left paren> 0 <right paren>

<Part 13 types yes> ::=
       1 | typesyes <left paren> 1 <right paren>

<Part 13 JAR privileges> ::=
         <Part 13 JAR privileges no>
       | <Part 13 JAR privileges yes> <JAR privileges support>

<Part 13 JAR privileges no> ::=
       0 | JARprivilegesno <left paren> 0 <right paren>

<Part 13 JAR privileges yes> ::=
       1 | JARprivilegesyes <left paren> 1 <right paren>

<Part 13 interfaces> ::=
       <Part 13 serializable> <Part 13 sqldata>

<Part 13 serializable> ::=
         <Part 13 serializable no>
       | <Part 13 serializable yes>

<Part 13 serializable no> ::=
       0 | serializableno <left paren> 0 <right paren>
```


Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

3.4 Object identifier for Database Language SQL

```
<Part 13 serializable yes> ::=
      1 | serializableyes <left paren> 1 <right paren>

<Part 13 sqldata> ::=
        <Part 13 sqldata no>
      | <Part 13 sqldata yes>

<Part 13 sqldata no> ::=
      0 | sqldatano <left paren> 0 <right paren>

<Part 13 sqldata yes> ::=
      1 | sqldatayes <left paren> 1 <right paren>

<routines support> ::=
      <routines commands> <routines deployment>

<routines commands> ::=
        <routines commands no>
      | <routines commands yes>

<routines commands no> ::=
      0 | routinescommandsno <left paren> 0 <right paren>

<routines commands yes> ::=
      1 | routinescommandsyes <left paren> 1 <right paren>

<routines deployment> ::=
        <routines deployment no>
      | <routines deployment yes>

<routines deployment no> ::=
      0 | routinesdeploymentno <left paren> 0 <right paren>

<routines deployment yes> ::=
      1 | routinesdeploymentyes <left paren> 1 <right paren>

<types support> ::=
      <types commands> <types deployment>

<types commands> ::=
        <types commands no>
      | <types commands yes>

<types commands no> ::=
      0 | typescommandsno <left paren> 0 <right paren>

<types commands yes> ::=
      1 | typescommandsyes <left paren> 1 <right paren>

<types deployment> ::=
        <types deployment no>
      | <types deployment yes>

<types deployment no> ::=
      0 | typesdeploymentno <left paren> 0 <right paren>

<types deployment yes> ::=
      1 | typesdeploymentyes <left paren> 1 <right paren>

<JAR privileges support> ::=
```


Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-13:2002

```
      <JAR privileges commands> <JAR privileges deployment>

<JAR privileges commands> ::=
        <JAR privileges commands no>
      | <JAR privileges commands yes>

<JAR privileges commands no> ::=
      0 | JARprivilegescommandsno <left paren> 0 <right paren>

<JAR privileges commands yes> ::=
      1 | JARprivilegescommandsyes <left paren> 1 <right paren>

<JAR privileges deployment> ::=
        <JAR privileges deployment no>
      | <JAR privileges deployment yes>

<JAR privileges deployment no> ::=
      0 | JARprivilegesdeploymentno <left paren> 0 <right paren>

<JAR privileges deployment yes> ::=
      1 | JARprivilegesdeploymentyes <left paren> 1 <right paren>
```

Syntax Rules

1) Insert this SR Specification of <Part 13 yes> implies that conformance to ISO/IEC 9075-13 is claimed.

2) Insert this SR If <Part 13 yes> is not specified then <invoked Java> shall not be specified.

3) Insert this SR If <Part 13 conformance> specifies `sqljrt2002`, then <SQL edition> shall specify <1999>.

4) Insert this SR Specification of <Part 13 routines no> implies that conformance to Feature J621, “external Java routines”, is not claimed.

5) Insert this SR Specification of <Part 13 routines yes> implies that conformance to Feature J621, “external Java routines”, is claimed.

6) Insert this SR Specification of <Part 13 types no> implies that conformance to Feature J622, “external Java types”, is not claimed.

7) Insert this SR Specification of <Part 13 types yes> implies that conformance to Feature J622, “external Java types”, is claimed.

8) Insert this SR Specification of <Part 13 JAR privileges no> implies that conformance to Feature J561, “JAR privileges”, is not claimed.

9) Insert this SR Specification of <Part 13 JAR privileges yes> implies that conformance to Feature J561, “JAR privileges”, is claimed.

10) Insert this SR Specification of <Part 13 serializable no> implies that conformance to Feature J541, “Serializable”, is not claimed.

11) Insert this SR Specification of <Part 13 serializable yes> implies that conformance to Feature J541, “Serializable”, is claimed.

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

3.4 Object identifier for Database Language SQL

12) Insert this SR Specification of <routines commands no> implies that conformance to Feature J511, “Commands”, is not claimed for Feature J621, “external Java routines”.

13) Insert this SR Specification of <routines commands yes> implies that conformance to Feature J511, “Commands”, is claimed for Feature J621, “external Java routines”.

14) Insert this SR Specification of <types commands no> implies that conformance to Feature J511, “Commands”, is not claimed for Feature J622, “external Java types”.

15) Insert this SR Specification of <types commands yes> implies that conformance to Feature J511, “Commands”, is claimed for Feature J622, “external Java types”.

16) Insert this SR Specification of <JAR privileges commands no> implies that conformance to Feature J511, “Commands”, is not claimed for Feature J561, “JAR privileges”.

17) Insert this SR Specification of <JAR privileges commands yes> implies that conformance to Feature J511, “Commands”, is claimed for Feature J561, “JAR privileges”.

18) Insert this SR Specification of <routines deployment no> implies that conformance to Feature J531, “Deployment”, is not claimed for Feature J621, “external Java routines”.

19) Insert this SR Specification of <routines deployment yes> implies that conformance to Feature J531, “Deployment”, is claimed for Feature J621, “external Java routines”.

20) Insert this SR Specification of <types deployment no> implies that conformance to Feature J531, “Deployment”, is not claimed for Feature J622, “external Java types”.

21) Insert this SR Specification of <types deployment yes> implies that conformance to Feature J531, “Deployment”, is claimed for Feature J622, “external Java types”.

22) Insert this SR Specification of <JAR privileges deployment no> implies that conformance to Feature J531, “Deployment”, is not claimed for Feature J561, “JAR privileges”.

23) Insert this SR Specification of <JAR privileges deployment yes> implies that conformance to Feature J531, “Deployment”, is claimed for Feature J561, “JAR privileges”.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-13:2002

4 Concepts

4.1 The Java programming language

The Java programming language is a class-based, object-oriented language. This part of ISO/IEC 9075 uses the following Java concepts and terminology.

A *class* is the basic construct of Java programs, in that all executable Java code is contained in a Java class definition. A class is declared by a *class declaration* that specifies a possibly empty set consisting of zero or more fields, zero or more methods, zero or more nested classes, zero or more interfaces, zero or more instance initializers, zero or more static initializers, and zero or more constructors.

The scope of a variable is a class, an instance of the class, or a method of the class. The scope of a variable that is declared *static* is the class, and the variable is called a *class variable*. The scope of each other variable declared in the class is instances of the class, and such a variable is called an *instance variable*. Class variables and instance variables of a class are called *fields* of that class. The scope of a variable declared in a method is the *block* or Java `for` statement in which it is declared in that method, and the variable is called a *local variable*.

A *class instance* consists of an instance of each instance variable declared in the class and each instance variable declared in the superclasses of the class. Class instances are strongly typed by the class name. The operations available on a class instance are those defined for its class.

With the exception of the class `java.lang.Object`, each class is declared to *extend* (at most) one other class; a class not explicitly declared to extend another class implicitly extends `java.lang.Object`. The declared class is a *direct subclass* of the class that it extends; the class that it extends is the *direct superclass* of the declared class.

Class B is a *subclass* of class A if B is a direct subclass of A, or if there exists some class C such that B is a direct subclass of C and C is a subclass of A. Likewise, class B is a *superclass* of class A if B is a direct superclass of A, or if there exists some class C such that B is a direct superclass of C and C is a superclass of A. A subclass has all of the fields and methods of its superclasses and an instance of class B may be used wherever an instance of a superclass of B is permitted.

A *method* is an executable routine. A method can be declared *static*, in which case it is called a *class method*; otherwise, it is called an *instance method*. A class method can be referenced by qualifying the method name with either the class name or the name of an instance of the class. An instance method is referenced by qualifying the method name with a Java expression that results in an instance of the class or, in the case of a constructor, with "`new`". The method body of an instance method can reference its class variables, instance variables, and local variables.

The *Java method signature* of a method consists of the method name and the number of parameters and their data types.

A *package* consists of zero or more classes, zero or more interfaces, and zero or more *subpackages* (a subpackage is a package within a package); each package provides its own name space and classes within a package are able to refer to other classes in the same package, including classes not referenceable from outside the package. Every class belongs to exactly one package, either an explicitly specified named package or the anonymous default package. A class can specify Java `import` statements to refer to other named packages whose classes can then be referenced within the class without package qualification.

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

4.1 The Java programming language

A class, field, or methods can be declared as *public*, *private*, or *protected*. A public variable or method can be accessed by any method. A private variable or method can only be referenced by methods in the same class. A protected variable or method can only be referenced by methods of the same class or subclasses thereof. A method that is not declared as public, private, or protected can only be called by methods declared by classes in the same package.

An *interface* is a Java construct consisting of a set of method signatures. An interface can be implemented by zero or more classes, a class can be declared to implement zero or more interfaces, and a class is required to have methods with the signatures specified by all of its declared interfaces.

The Java *Serializable* interface, `java.io.Serializable`, as described in *Java 2 Platform, Standard Edition, v1.2.2, API Specification*, defines a transformation between a Java instance and a `java.io.OutputStream` or `java.io.InputStream`, as defined by the `java.io.OutputStream` and `java.io.InputStream` sections of *Java 2 Platform, Standard Edition, v1.2.2, API Specification*, respectively, writing a persistent representation of an instance of a Java object and reading a persistent representation of an instance of a Java object. This transformation retains sufficient information to identify the most specific class of the instance and to reconstruct the instance.

The Java *SQLData* interface, `java.sql.SQLData`, as described in *Java 2 Platform, Standard Edition, v1.2.2, API Specification*, defines a transformation between a Java instance and an SQL user-defined data type.

The source for a Java class is normally stored in a *Java file* with the file-type "java", *e.g.*, `myclass.java`. Java is normally compiled to a byte coded instruction set that is portable to any platform supporting Java. A file containing such byte code is normally stored in a *class file* with the file-type "class", *e.g.*, `myclass.class`.

A set of class files can be assembled into a *Java archive* file, or *JAR* (usually with a file extension of ".jar". A JAR is a zip formatted file containing a set of Java class files. JARs are the normal vehicle for distributing Java programs.

4.2 SQL-invoked routines

Insert after 1st paragraph An SQL-invoked routine can be an SQL routine or an external routine. An SQL routine is an SQL-invoked routine whose <language clause> specifies SQL. The <routine body> of an SQL routine is an <SQL procedure statement>; the <SQL procedure statement> forming the <routine body> can be any SQL-statement, including an <SQL control statement>, but excluding an <SQL schema statement>, <SQL connection statement>, or <SQL transaction statement>.

Insert after 1st paragraph An external routine is one whose <language clause> does not specify SQL. The <routine body> of an external routine is an <external body reference> whose <external routine name> identifies a program written in some standard programming language other than SQL.

Insert after 1st paragraph External routines appear in two seemingly similar, but fundamentally differing, forms, where the key differentiator is whether or not the external routine's routine descriptor specifies that the body of the SQL-invoked routine is written in Java. When the body of the SQL-invoked routine is written in Java, the external routine is an *external Java routine*; some differences from other external routines include:

— For any other external routine, the *executable form* (such as a dynamic link library or some run-time interpreted form) of that routine exists externally to the SQL-environment's catalogs; for an external Java routine, the executable form is provided by a specified subject Java routine that exists in the SQL-environment's catalogs in an installed JAR.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-13:2002

— Because an installed JAR is not required to be completely self-contained (*i.e.*, to have no references to Java classes outside of itself), a mechanism is provided to allow a subject Java class to reference classes defined by class files contained in its installed JAR or in other installed JARs. See Subclause 8.2, “<SQL Java path>”.

NOTE 1 – Once an external Java routine has been created, its use in SQL statements executed by the containing SQL-environment is similar to that of other external routines.

Replace the lead-in paragrph of the 10th paragraph A static SQL-invoked method, whether or not it is an external Java routine, satisfies the following conditions:

Delete the 13th and 14th paragraphs

Replace the 16th paragraph 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 that is not an external Java routine, 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>", in ISO/IEC 9075-2.) 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>", in ISO/IEC 9075-2.) If the instance SQL-invoked method is an external Java routine, the term “set of overriding methods” is not applicable; for such methods, the capabilities provided by overriding methods duplicate Java’s own mechanisms and the subject routine executed is the one that would be invoked when no overriding methods are specified. 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 Subclause 13.5, "<SQL procedure statement>, in ISO/IEC 9075-2. 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 programming language.

Replace the 18th paragraph If the SQL-invoked routine is an external routine, then an effective SQL parameter list is constructed before the execution of the <routine body>. The effective SQL parameter list has different entries depending on the parameter passing style of the SQL-invoked routine. The value of each entry in the effective SQL parameter list is set according to the General Rules of Subclause 8.3, “<routine invocation>”. When the SQL-invoked routine is not an external Java routine, the values in the effective SQL parameter list are passed to the program identified by the <routine body> according to the rules of Subclause 13.6, "Data type correspondences", in ISO/IEC 9075-2; when the SQL-invoked routine is an external Java routine, values in the effective SQL parameter list are passed to the program identified by <routine body> according to the rules of Subclause 4.5, “Parameter mapping”. After the execution of that program, if the parameter passing style of the SQL-invoked routine is SQL, then the SQL-implementation obtains the values for output parameters (if any), the value (if any) returned from the program, the value of the exception

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

data item, and the value of the message text (if any) from the values assigned by the program to the effective SQL parameter list. If the parameter passing style of the SQL-invoked routine is JAVA, then such values are obtained from the values assigned by the program to the effective SQL parameter list and the uncaught Java exception (if any). If the parameter passing style of the SQL-invoked routine is GENERAL, then such values are obtained in an implementation-defined manner.

4.3 Java class name resolution

Typical JVMs provide a *class name resolution*, or search path, mechanism based on an environmental variable called CLASSPATH. When a JVM encounters a previously unseen reference to a class, the members of the list of directories and JARs appearing in the classpath are examined in order until either the class is found or the end of the list is reached. Failure to locate a referenced class is a runtime error that will often cause the application that experiences it to terminate.

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. 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 classpath-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). Therefore, if, while an external Java routine is being executed, a previously unseen class reference is encountered, that class is first 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).

An SQL-Java path specifies how a JVM resolves a class name when a class within a JAR references a class that is not a system class or not in the same JAR. `SQLJ.ALTER_JAVA_PATH` is used to associated an SQL-Java path with a JAR. An SQL-Java path is a list of (referenced item, referenced JAR) pairs. A referenced item can be either a class, a package, or '*' to specify the entire JAR. The SQL-Java path list is searched in the order the pairs are specified. For each (referenced item, referenced JAR) pair (*RI*, *RJ*):

— If *RI* is the class name, then the class must be defined in *RJ*. If it is not, an exception condition is raised.

— If *RI* is the package of the class being resolved, then the class must be defined in *RJ*. If it is not, an exception condition is raised.

— If *RI* is '*' and the class is defined in *RJ*, then that resolution is used; otherwise, subsequent pairs are tested.

4.4 SQL result sets

Cursors, or SQL result sets, appear to Java applications in two forms; the first, as an object of a class that implements the interface `java.sql.ResultSet` as defined by JDBC in *JDBC API Tutorial and Reference, Second Edition: Universal Data Access for the Java 2 Platform*, referred to as a *JDBC ResultSet*; the second, as an object of a class that implements the interface `sqlj.runtime.ResultSetIterator` as defined by ISO/IEC 9075-10, referred to as an *SQLJ Iterator*.

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

In ISO/IEC 9075-2, SQL-invoked procedures are declared to be able to return zero or more dynamic result sets, referred to as *result set cursors*. To be a returned result set cursor, a cursor's declaration must specify WITH RETURN, and the cursor must be open at the point that the SQL-invoked procedure exits. While external Java routines that are SQL-invoked procedures can likewise be declared to return zero or more dynamic result sets, in some other respects, this part of ISO/IEC 9075's treatment of result set cursors differs from that of ISO/IEC 9075-2.

In a Java application, all JDBC ResultSets and SQLJ Iterators are implicitly result set cursors, that is, their underlying cursor declarations implicitly specify WITH RETURN. So, in this part of ISO/IEC 9075, to actually be a returned result set cursor it is not sufficient that the corresponding JDBC ResultSet's or SQLJ Iterator's underlying cursor be open when the SQL-invoked procedure exits; the JDBC ResultSet or SQLJ Iterator must also have been explicitly assigned to a parameter of the subject Java routine that represents an output parameter. As discussed in Subclause 4.5, "Parameter mapping", and Subclause 8.3, "<routine invocation>", output parameters are represented to a subject Java routine as the first element of a one dimensional array of a Java data type that can be mapped to an SQL data type. For dynamic result sets, the array must be of a class that implements the interface `java.sql.ResultSet` or the interface `sqlj.runtime.ResultSetIterator`, the JDBC ResultSet or SQLJ Iterator must have been explicitly assigned to the first element of that array, and that JDBC ResultSet or SQLJ Iterator must not have been closed.

It is important to note that this difference in how a result set cursor becomes a returned result set cursor is invisible to the calling application. As described in Subclause 8.3, "<routine invocation>", the calling application will be returned zero or more dynamic result sets in the order in which the cursors were opened, just as in ISO/IEC 9075-2; the order of the parameters in the subject Java routine does not impact the order in which the calling application accesses the returned result sets.

4.5 Parameter mapping

Let *ST* be some SQL data type and let *JT* be some Java data type.

ST and *JT* are *simply mappable* if and only if *ST* and *JT* are specified respectively in the first and second columns of some row of the *JDBC data type mapping table*, Table 47.1, entitled "JDBC Types mapped to Java Types", in *JDBC API Tutorial and Reference, Second Edition: Universal Data Access for the Java 2 Platform*. The Java data type *JT* is the *corresponding Java data type* of *ST*.

ST and *JT* are *object mappable* if and only if *ST* and *JT* are specified respectively in the first and second columns of some row of the *JDBC object type mapping table*, Table 47.3, entitled "Mapping from JDBC Types to Java Object Types", in *JDBC API Tutorial and Reference, Second Edition: Universal Data Access for the Java 2 Platform*, or if the descriptor of *ST* specifies that it is an external Java data type and the descriptor specifies *JT* as the <Java class name> in the <jar and class name>.

ST and *JT* are *output mappable* if and only if *JT* is a one dimensional array type with an element data type *JT2* (that is, *JT* is "*JT2*[]") and *ST* is either simply mappable to *JT2* or object mappable to *JT2*.

An SQL array type with an element data type *ST* and *JT* are *array mappable* if and only if *JT* is a one dimensional array type with an element data type *JT2* and *ST* is either simply mappable to *JT2* or object mappable to *JT2*.

ST and *JT* are *mappable* if and only if *ST* and *JT* are simply mappable, object mappable, output mappable, or array mappable.

A Java data type is *mappable* if and only if it is mappable to some SQL data type.

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

4.5 Parameter mapping

A Java class is *result set oriented* if and only if it is either:

— A class that implements the Java interface `java.sql.ResultSet`.

— A class that implements the Java interface `sqlj.runtime.ResultSetIterator`.

NOTE 2 – These classes are generated by iterator declarations (`#sql iterator`) as specified in ISO/IEC 9075-10.

A Java data type is *result set mappable* if and only if it is a one-dimensional array type with an element type that is a result set oriented class.

A Java method with M parameters is *mappable* (to SQL) if and only if, for some N, 0 (zero) $\leq N \leq M$, the data types of the first N parameters are mappable, the last $M-N$ parameters are result set mappable, and the result type is either simply mappable, object mappable, or `void`.

A Java method is *visible* in SQL if and only if it is public and mappable. In addition, to be visible, a Java method must be static if used as the external Java routine of an SQL-invoked procedure or an SQL-invoked regular function.

A Java class is *visible* in SQL if and only if it is public and mappable.

The book *JDBC API Tutorial and Reference, Second Edition: Universal Data Access for the Java 2 Platform* contains JDBC's SQL to Java data type mappings defined in the JDBC type mapping tables. If *ST* is an external Java data type that appears in the INFORMATION_SCHEMA . USER_DEFINED_TYPES view, then JDBC's data type mapping tables are effectively extended. A row (*ST*, *JT*) is considered to be an additional row in Table 47.3, *Mapping from JDBC Types to Java Object Types*, and a row (*JT*, *ST*) is considered to be an additional row in Table 47.4, *Mapping from Java Object Types to JDBC Types*.

4.6 Unhandled Java exceptions

Java exceptions that are thrown during execution of a Java method in SQL can be caught, or handled, within Java; if this is done, then those exceptions do not affect SQL processing. All Java exceptions that are uncaught when a Java method called from SQL completes appear in the SQL-environment as SQL exception conditions.

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 that exception specifies an SQLSTATE value, the first two characters of that SQLSTATE shall be '38'. 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".

When a Java method executes an SQL statement, any exception condition raised in the SQL statement will be raised in the Java method as a Java exception that is specifically the `java.sql.SQLException` subclass of the Java class `java.lang.Exception`. For portability, a Java method called from SQL, that itself executes an SQL statement and that catches an SQLException from that inner SQL statement, should re-throw that SQLException.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-13:2002

4.7 Data types

Replace 16th paragraph Each host language has its own data types, which are separate and distinct from SQL data types, even though similar names may be used to describe the data types. Mappings of SQL data types to data types in host languages are described in Subclause 11.49, "<SQL-invoked routine>", in ISO/IEC 9075-2, in Clause 16, "Embedded SQL", in ISO/IEC 9075-5, and in Subclause 8.1, "<embedded SQL host program>", in ISO/IEC 9075-10. Not every SQL data type has a corresponding data type in every host language.

4.8 User-defined types

Insert after 1st paragraph User-defined types appear in two seemingly similar, but fundamentally differing, forms in which the key differentiator is whether or not the create type statement for the user-defined type specifies an external language of "JAVA". When an external language of JAVA is specified, the user-defined type is an *external Java data type* and the create type statement defines a mapping of the user-defined type's attributes and methods directly to the public attributes and methods of a *subject Java class*. This is different from user-defined types that are not external Java data types. The differences include:

— For every other user-defined type, there is no requirement for an association with an underlying class; each method of a user-defined type that is not an external Java data type can be written in a different language (for example, one method could be written in SQL and another written in Fortran). Such user-defined types cannot have methods written in Java. By contrast, all methods of an external Java data type must be written in Java, (implicitly) have a parameter style of JAVA, and be defined in the associated Java class or one of its superclasses.

— For every other user-defined type, there is no explicit association between a user-defined type's attributes and any external representation of their content. In addition, the mapping between a user-defined type's methods and external methods is made over time by subsequent CREATE METHOD statements. By contrast, for external Java data types, the association between the user-defined type's attributes and methods and the public attributes and methods of a subject Java class is specified by the create type statement.

— For external Java data types, the mechanism used to convert the SQL-environment's representation of an instance of a user-defined type into an instance of a Java class is specified in the USING <interface specification> clause. Such conversions are performed, for example, when an external Java data type is specified as a (subject) parameter in a method or function invocation, or when a Java object returned from a method or function invocation is stored in a column declared to be an external Java data type. <interface specification> can be either SERIALIZABLE, specifying the Java-defined interface `java.io.Serializable` (not to be confused with the isolation level of SERIALIZABLE), or SQLDATA, specifying the JDBC-defined interface `java.sql.SQLData`. See Subclause 9.3, "<user-defined type definition>".

— For every other user-defined type, there is no explicit support of static attributes. For external Java data types, the <user-defined type definition> is allowed to include <static field method spec>s that define observer methods against specified static attributes of the subject Java class.

The scope and persistence of any modifications to static attributes made during the execution of a Java method is implementation-dependent.

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

— For every other user-defined type, the implementation of every method that isn't an SQL routine exists externally to the SQL-environment. For external Java data types, the implementation of the methods is provided by a specified subject Java class that exists within the SQL-environment in an *installed JAR*.

— External Java data types may only be structured types, not distinct types.

— Support for the specification of overriding methods is not provided for methods that are external Java routines.

NOTE 3 – Once an external Java data type has been created, its use in SQL statements executed by the containing SQL-implementation is similar to that of other user-defined types.

Replace 8th paragraph A user-defined type is described by a user-defined type descriptor. A user-defined type descriptor includes:

— The name of the user-defined type (<user-defined type name>). This is the type designator of that type, used in type precedence lists (see Subclause 9.5, "Type precedence list determination", in ISO/IEC 9075-2).

— An indication of whether the user-defined type is ordered.

— The ordering form for the user-defined type (EQUALS, FULL, or NONE).

— The ordering category for the user-defined type (RELATIVE, COMPARABLE, MAP, or STATE).

— A <specific routine designator> identifying the ordering function, depending on the ordering category.

— If the user-defined type is a direct subtype of another user-defined type, then the name of that user-defined type.

— An indication of whether the user-defined type is instantiable or not instantiable.

— An indication of whether the user-defined type is final or not final.

— The transform descriptor of the user-defined type.

— 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.

- If user-defined representation is specified, then the type descriptor of the representation type of the referencing type of the structured type; otherwise, if derived representation is specified, then the list of attributes.

NOTE 4 – "user-defined representation", "derived representation", and "system-defined representation" of a reference type are defined in Subclause 4.10, "Reference types", in ISO/IEC 9075-2.

— An indication of whether the user-defined type is an external Java data type.

Insert following the 8th paragraph If the user-defined type is an external Java data type, then the user-defined type descriptor also includes:

— The <jar and class name> of the user-defined type.

— The <interface specification> of SERIALIZABLE or SQLDATA.

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

— The attribute descriptor of every originally-defined attribute and every inherited attribute of the user-defined type.

— If <method specification list> is specified, then, for each <method specification> contained in <method specification list>, a *method spec descriptor* that includes:

- The <method name>.
- The <specific method name>.
- The <SQL parameter declaration list>.
- The <returns data type>, and indication of SELF AS RESULT.
- The <result cast from type>, if any.
- The package, class, and name of the Java routine corresponding to this method and, if specified, its signature.
- An indication of whether STATIC or CONSTRUCTOR is specified.
- If STATIC is specified, then an indication of whether this is a static field method.
- If this is a static field method, then the <Java field name> of the static field and the <Java class name> of the class that declares that static field.
- An indication of whether the method is deterministic.
- An indication of whether the method possibly writes SQL data, possibly reads SQL data, possibly contains SQL, or does not possibly contain SQL.
- An indication of whether the method should not be invoked if any argument is the null value, in which case the value of the method is the null value.

If the user-defined type is not an external Java data type, then the user-defined type descriptor also includes:

— An indication of whether the user-defined type is a structured type or a distinct type.

— If the representation is a predefined data type, then the descriptor of that type; otherwise, the attribute descriptor of every originally-defined attribute and every inherited attribute of the user-defined type.

— If the <method specification list> is specified, then, for each <method specification> contained in <method specification list>, a *method specification descriptor* that includes:

- The <method name>.
- The <specific method name>.
- The <SQL parameter declaration list> augmented to include the implicit first parameter with parameter name SELF.
- The <language name>.
- If the <language name> is not SQL, then the <parameter style>.
- The <returns data type>.

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

- The <result cast from type>, if any.
- An indication as to whether the <method specification> is an <original method specification> or an <overriding method specification>.
- If the <method specification> is an <original method specification>, then an indication of whether STATIC or CONSTRUCTOR is specified.
- An indication whether the method is deterministic.
- An indication whether the method possibly writes SQL data, possibly reads SQL data, possibly contains SQL, or does not possibly contain SQL.
- An indication whether the method should not be invoked if any argument is the null value, in which case the value of the method is the null value.

NOTE 5 – The characteristics of an <overriding method specification> other than the <method name>, <SQL parameter declaration list>, and <returns data type> are the same as the characteristics for the corresponding <original method specification>.

4.8.1 User-defined type comparison and assignment

Replace 5th paragraph Let *comparison function* of a user-defined type T_a be the ordering function included in the user-defined type descriptor of the comparison type of T_a, if any.

Replace 6th paragraph 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 and to be the same as those of all their supertypes. If the comparison category is COMPARABLE, then no comparison functions shall be specified for *T1* and *T2*. 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.

4.8.2 Accessing static fields

The fields of a Java class can be defined to be either *static* or *non-static*. Static fields of a Java class can additionally be specified to be *final*, which makes them read-only. In Java, non-final fields are allowed to be updated.

SQL's <user-defined type definition> does not include a facility for specifying attributes to be STATIC. This is, in part, because of the difficulty in specifying the scope, persistence, and transactional properties of static attributes in a database environment. An external Java data type's <user-defined type definition> does, however, provide a mechanism for read-only access to the values of Java static fields. The <static field method spec> clause defines a method name for a method with no parameters; its <external variable name clause> specifies the name of a static field of the subject Java class or a superclass of the subject Java class. A static field method is invoked in the normal manner for STATIC methods and returns the value of the specified Java static field. Whether final or non-final, SQL provides no mechanism for updating the values of Java static fields.

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

4.8.3 Converting objects between SQL and Java

While application programmers or end users manipulating Java objects in the database through SQL statements need not be aware of the specific mechanism used to achieve that conversion, the developer of the Java class itself needs to prepare for it in the form of implementing special Java interfaces (*i.e.*, `java.io.Serializable` or `java.sql.SQLData`). <user-defined type definition> introduces a clause for specifying the interface for converting object state information between the SQL database and Java in the scope of SQL statements. As mentioned above, a conversion from SQL to Java can potentially take place when an object that has been persistently stored in the SQL database is accessed from inside an SQL statement to retrieve attribute (or field) values, or to invoke a method on the object, or when the object is used as an input argument in the invocation of a method. A conversion in the opposite direction, from Java to SQL, may be required when a newly created or modified object, or an object that is the return value of a method invocation, needs to be persistently stored in the database.

This International Standard supports these options to specify object state conversion in the <external Java type clause>:

— If the <user-defined type definition> specifies an <interface specification> of SERIALIZABLE, then the Java interface `java.io.Serializable` is used for object state conversion.

— If the <user-defined type definition> specifies an <interface specification> of SQLDATA, then the Java interface `java.sql.SQLData` defined in JDBC 2.0 is used for object state conversion.

— If the <user-defined type definition> does not specify an <interface specification>, then it is implementation-defined whether the Java interface `java.io.Serializable` or the Java interface `java.sql.SQLData` will be used for object state conversion.

4.8.3.1 SERIALIZABLE

If the <interface specification> of a <user-defined type definition> specifies SERIALIZABLE, then object state communication is based on the Java interface `java.io.Serializable`. The Java class referenced in the <external Java class clause> of the <user-defined type definition> must specify "`implements java.io.Serializable`" and must provide a niladic constructor.

In this case, the SQL object state that is stored persistently and made available to methods of the SQL type is defined entirely by the Java serialized object state. The attributes defined for the SQL type must correspond to public fields of the corresponding Java class, which must be listed in the <external Java attribute clause> of each attribute. Consequently, the SQL attributes define access to those portions of the object state that are intended to become visible inside SQL statements, but might not comprise the complete state of the object (which may include additional fields in the Java class).

4.8.3.2 SQLDATA

If the <interface specification> of a <user-defined type definition> specifies SQLDATA, then object state communication is based on the Java interface `java.sql.SQLData` defined in JDBC 2.0. The Java class referenced in the <external Java class clause> of the <user-defined type definition> must specify "`implements java.sql.SQLData`" and must provide a niladic constructor.

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

In this case, only the attributes defined in the statement comprise the complete state of the SQL object type. Additional public or private attributes defined in the Java class do not become part of the object state defined by this part of ISO/IEC 9075. The Java object representation may be entirely different from the SQL object attributes, if desired. For example, an SQL Point type may define a geometric point in terms of cartesian coordinates, while the corresponding Java class defines it using polar coordinates. The only requirement to be met by the implementor of the Java class is that the implementations of the `java.sql.SQLData` methods `readSQL` and `writeSQL` read and write the attributes in the same order in which they are defined in the <user-defined type definition>.

To improve portability, it is possible to also specify <external Java attribute clause>s for SQL attributes, even if an <interface specification> of SQLDATA is specified. However, the <external Java attribute clause>s are ignored in this case, because they are not needed for implementing attribute access in SQL or for converting objects between SQL and Java.

4.8.3.3 Developing for portability

The following guidelines provide maximum portability of Java classes across different implementations of this part of ISO/IEC 9075 that may not support both the SERIALIZABLE and the SQLDATA options:

— The Java class used for implementing the SQL type should implement both `java.io.Serializable` and `java.sql.SQLData`.

— The Java class should define the complete object state that needs to become persistent or has to be preserved across invocations as public Java fields.

— The EXTERNAL NAMEs of the SQL attributes should be specified.

The <interface using clause> should be omitted in the <user-defined type definition>, so that an implementation that does not support both interfaces can default to the interface that it supports.

4.9 Built-in procedures

This part of ISO/IEC 9075 differs slightly from other parts of ISO/IEC 9075 in its treatment of the schema object introduced to install the external Java routines and external Java data types in an SQL-environment — that is, in its treatment of JARs. Rather than define new SQL-schema statements that (for example) add or drop JARs using optional clauses to cause execution of their contained deployment descriptors, this International Standard introduces a set of four built-in procedures and a new schema in which those procedures are defined.

The new schema — named SQLJ — is, like the schema named INFORMATION_SCHEMA, defined to exist in all catalogs of an SQL system that implements this part of ISO/IEC 9075, and to contain all of the built-in procedures defined in this part of ISO/IEC 9075.

Built-in procedures defined in this part of ISO/IEC 9075 are:

— `SQLJ.INSTALL_JAR` — to load a set of Java classes in an SQL system.

— `SQLJ.REPLACE_JAR` — to supersede a set of Java classes in an SQL system.

— `SQLJ.REMOVE_JAR` — to delete a previously installed set of Java classes.

— `SQLJ.ALTER_JAVA_PATH` — to specify a path for name resolution within Java classes.

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

4.10 Privileges

Replace 1st paragraph A privilege authorizes a given category of <action> to be performed on a specified base table, view, column, domain, character set, collation, transliteration, user-defined type, trigger, SQL-invoked routine, or JAR by a specified <authorization identifier>.

Replace 2nd paragraph, 1st bullet

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

Replace 8th paragraph A privilege descriptor with an <action> of USAGE is called a *usage privilege descriptor* and identifies the existence of a privilege on the domain, user-defined type, character set, collation, transliteration, or JAR identified by the privilege descriptor.

Insert after 8th paragraph The privileges for facilities defined in this part of ISO/IEC 9075 are as follows:

— The privileges required to invoke the `SQLJ.INSTALL_JAR`, `SQLJ.REPLACE_JAR`, and `SQLJ.REMOVE_JAR` procedures are implementation-defined.

NOTE 6 – This is similar to the implementation-defined privileges required for creating a schema.

— Only the owner of the JAR is permitted to invoke the `SQLJ.ALTER_JAVA_PATH` procedure and the owner must also have the USAGE privilege on each JAR referenced in the path argument.

— Invocations of <SQL-invoked routine> and <drop routine statement> to define and drop external Java routines are governed by the normal Access Rules for SQL-schema statements.

— Invocations of Java methods referenced by SQL names are governed by the normal EXECUTE privilege on SQL routine names.

It is implementation-defined whether a Java method called by an SQL name executes with "definer's rights" or "invoker's rights" — that is, whether it executes with the user-name of the user who performed the <SQL-invoked routine> or the user-name of the current user.

4.11 JARs

A JAR is a zip-formatted file containing a set of Java `class` and `ser` files and optionally a deployment descriptor file. Installed JARs provide the implementation of external Java routines and external Java data types to an SQL-environment.

JARs are created outside the SQL-environment. They are copied into the SQL-environment by the `SQLJ.INSTALL_JAR` procedure. No subsequent SQL statement or procedure modifies an installed JAR in any way, other than to remove it from the SQL-environment, to replace it in its entirety, or to alter its SQL-Java path. In particular, no SQL operation adds classes to a JAR, removes classes from a JAR, or replaces classes in a JAR. The reason for this "no modification" principle for installed JAR is that JARs are often signed, and often contain *manifest* data that might be invalidated by modification of JARs by the SQL-environment.

Each installed JAR is represented by a *JAR descriptor*. A JAR descriptor contains:

— The catalog name, schema name, and JAR identifier of the JAR.

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

— The SQL-Java path of the JAR.

4.11.1 Deployment descriptor files

When a JAR is installed, one or more <SQL-invoked routine>s that define external Java routines must be executed before the static methods of its contained Java classes can be used as SQL-invoked routines, and one or more <user-defined type definition>s must be executed before its contained classes can be used as user-defined types. In addition, <grant privilege statement>s may be required to define privileges for newly created SQL-invoked routines and user-defined types. Later, when a JAR is removed, corresponding <drop routine statement>s, <drop data type statement>s, and <revoke statement>s must be executed.

If a JAR is to be installed in several SQL implementations, the <SQL-invoked routine>s, <user-defined type definition>s, <user-defined ordering definition>s, <grant privilege statement>s, <drop routine statement>s, <drop data type statement>s, <drop user-defined ordering statement>s, and <revoke statement>s will often be the same for each implementation. To assist the automation of repeated installations, deployment descriptor files contain the variants of SQL-schema statements defined in this part of ISO/IEC 9075. These statements are grouped into multi-statement *install actions* and *remove actions* respectively executed by `SQLJ.INSTALL_JAR` and `SQLJ.REMOVE_JAR` procedures when deployment is requested. In addition, an implementation-defined *implementor block* is provided to allow specification of custom install and remove actions. Since the SQL-schema statements refer to their containing JAR in the <SQL-invoked routine>s and <user-defined type definition>s, within a deployment descriptor file, the JAR name "`thisjar`" is used as a place holder JAR name for the containing JAR.

This part of ISO/IEC 9075 provides a new mechanism to execute its variants of SQL-schema statements, namely by requesting deployment during invocation of `SQLJ.INSTALL_JAR` and `SQLJ.REMOVE_JAR` procedures. A conforming SQL-implementation is required to support either deployment descriptor based execution of its SQL-schema statements (Feature J531, "Deployment") or another standard statement execution mechanism such as direct invocation or embedded SQL (Feature J511, "Commands"); a conforming SQL-implementation is not required to support both mechanisms.

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

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

       | COMPARABLE

       | INTERFACE

       | JAVA

       | SQLDATA

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

       | JAR
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

5.2 Names and identifiers

Function

Specify names.

Format

```
<jar name> ::=
      [ <schema name> <period> ] <jar id>

<jar id> ::= <identifier>

<Java class name> ::=
      [ <packages> <period> ] <class identifier>

<jar and class name> ::=
      <jar id> <colon> <Java class name>

<qualified Java field name> ::=
      [ <Java class name> <period> ] <Java field name>

<packages> ::=
      <package identifier> [ <period> <package identifier> ]...

<package identifier> ::= <Java identifier>

<class identifier> ::= <Java identifier>

<Java field name> ::= <Java identifier>

<Java method name> ::= <Java identifier>

<Java identifier> ::= !! See the Syntax Rules
```

Syntax Rules

1) Insert this SR <Java identifier> shall be a valid identifier according to the rules of Java parsing and lexical analysis.

   NOTE 7 – The rules of Java parsing and lexical analysis are specified in *The Java Language Specification, Second Edition*.

2) Insert this SR The character set supported, and the maximum lengths of the <package identifier>, <class identifier>, <Java field name>, and <Java method name> are implementation-defined.

3) Insert after SR14) Two <jar name>s are equivalent if and only if they have equivalent <jar id>s and equivalent implicit or explicit <schema name>s.

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

Access Rules

No additional Access Rules.

General Rules

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

2) Insert this GR A <jar id> represents an unqualified JAR name.

3) Insert this GR A <Java class name> identifies a fully qualified Java class.

4) Insert this GR A <packages> identifies a fully qualified Java package.

5) Insert this GR A <package identifier> represents an unqualified Java package name.

6) Insert this GR A <class identifier> represents an unqualified Java class name.

7) Insert this GR A <Java field name> represents the name of a field within a Java class.

8) Insert this GR A <Java method name> represents the name of a method within a Java class.

Conformance Rules

No additional Conformance Rules.

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

(Blank page)

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-13:2002

6 Scalar expressions

6.1 <method invocation>

Function

Reference an SQL-invoked method of a user-defined type value.

Format

No additional Format items.

Syntax Rules

1) Insert after SR2) If *UDT* is an external Java data type, then <method invocation> shall immediately contain <direct invocation>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

6.2 <new specification>

Function

Invoke a method on a newly-constructed value of a structured type.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert after CR1) Without Feature J571, "NEW operator", the schema identified by the implicit or explicit <schema name> of <routine name> *RN* immediately contained in <routine invocation> immediately contained in <new specification> shall not contain a user-defined type whose user-defined type name is *RN* that is an external Java data type. If Feature J571, "NEW operator", is not supported, then the mechanism used to invoke a constructor of an external Java data type is implementation-defined.

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

7 Predicates

7.1 <comparison predicate>

Function

Specify a comparison of two row values.

Format

No additional Format items.

Syntax Rules

NOTE 8 – Replace 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 — they must be the same throughout a type family. If the comparison category is COMPARABLE, then no comparison functions shall be specified for *T1* and *T2*; 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.

NOTE 9 – Replace Note 127 If the comparison form is FULL, then the comparison category is constrained to be COMPARABLE, RELATIVE, or MAP; if the comparison form is EQUALS, then the comparison category is also permitted to be STATE.

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR 1)b)iii) and its subrules If the comparison category of UDT_x is COMPARABLE, then:

   a) The subject SQL data type must be an external Java data type. Let *JC* be the subject Java class of that external Java data type.

      NOTE 10 – Syntax Rules in Subclause 9.10, "<user-defined ordering definition>", require that *JC* implement the Java interface `java.lang.Comparable`. The interface `java.lang.Comparable` requires an implementing Java class to have a method named `compareTo`, whose result data type is Java `int`.

   b) Let *XJV* be the value of *X* in the associated JVM. Let *YJV* be the value of *Y* in that associated JVM.

   c) `X = Y`

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002

has the same result as if the JVM executed the Java boolean expression

```
XJV.compareTo(YJV) == 0
```

d) `X < Y`

has the same result as if the JVM executed the Java boolean expression

```
XJV.compareTo(YJV) < 0
```

e) `X <> Y`

has the same result as if the JVM executed the Java boolean expression

```
XJV.compareTo(YJV) != 0
```

f) `X > Y`

has the same result as if the JVM executed the Java boolean expression

```
XJV.compareTo(YJV) > 0
```

g) `X <= Y`

has the same result as if the JVM executed the Java boolean expression

```
XJV.compareTo(YJV) <= 0
```

h) `X >= Y`

has the same result as if the JVM executed the Java boolean expression

```
XJV.compareTo(YJV) >= 0
```

Conformance Rules

No additional Conformance Rules.

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-13:2002