TECHNICAL REPORT

ISO/IEC TR 19075-3

First edition
2015-07-01

Information technology — Database languages — SQL Technical Reports —

Part 3: SQL Embedded in Programs using the Java™ programming language

Technologies de l'information — Langages de base de données — SQL rapports techniques —
Partie 3: SQL intégrées dans des programmes utilisant le langage de programmation de Java™

Reference number
ISO/IEC TR 19075-3:2015(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword . . . v

Introduction . . . vi

1 Scope . . . 1

2 Normative references . . . 3

2.1 ISO and IEC standards . . . 3

2.2 Other international standards . . . 3

3 Use of SQL in programs written in Java . . . 5

3.1 Design goals . . . 5

3.2 Advantages of SQL/OLB over JDBC . . . 5

3.3 Consistency with existing embedded SQL languages . . . 6

3.4 Profile customization overview . . . 6

3.4.1 Profile customization process . . . 7

3.4.2 Profile customization utilities . . . 8

3.5 Examples . . . 8

3.5.1 Example of Profile generation and naming . . . 8

3.5.2 Example of a JAR manifest file . . . 8

3.5.3 Host variables . . . 9

3.5.4 Host expressions . . . 9

3.5.5 SQL/OLB clauses . . . 10

3.5.6 Connection contexts . . . 10

3.5.7 Default connection context . . . 10

3.5.8 Iterators . . . 11

3.5.8.1 Positional bindings to columns . . . 11

3.5.8.2 Named bindings to columns . . . 12

3.5.8.3 Providing names for columns of queries . . . 13

3.5.9 Invoking SQL-invoked routines . . . 13

3.5.10 Using multiple SQL/OLB contexts and connections . . . 14

3.5.11 SQL execution control and status . . . 15

3.5.12 Multiple `java.sql.ResultSet` objects from SQL-invoked procedure calls . . . 16

3.5.13 Creating an SQL/OLB iterator object from a `java.sql.ResultSet` object . . . 17

3.5.14 Obtaining a `java.sql.ResultSet` object from an iterator object . . . 17

3.5.15 Working with user-defined types . . . 18

3.5.16 Batching . . . 19

3.5.17 Example program . . . 19

3.5.18 Host variable definition . . . 20

Index 23

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 2.

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.

In exceptional circumstances, when the joint technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example), it may decide to publish a Technical Report. A Technical Report is entirely informative in nature and shall be subject to review every five years in the same manner as an International Standard.

Attention is drawn to the possibility that some of the elements of this document 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 TR 19075-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

ISO/IEC TR 19075 consists of the following parts, under the general title *Information technology — Database languages — SQL Technical Reports*:

— Part 1: XQuery Regular Expression Support in SQL

— Part 2: SQL Support for Time-Related Information

— Part 3: SQL Embedded in Programs Using the Java™ Programming Language

— Part 4: SQL With Routines and Types Using the Java™ Programming Language

— Part 5: Row Pattern Recognition in SQL

NOTE 1 — The individual parts of multi-part technical reports are not necessarily published together. New editions of one or more parts may be published without publication of new editions of other parts.

Introduction

The organization of this part of ISO/IEC 19075 is as follows:

1) Clause 1, “Scope”, specifies the scope of this part of ISO/IEC 19075.

2) Clause 2, “Normative references”, identifies additional standards that, through reference in this part of ISO/IEC 19075, constitute provisions of this part of ISO/IEC 19075.

3) Clause 3, “Use of SQL in programs written in Java”, provides a tutorial on the embedding of SQL expressions and statements in programs written in the Java programming language.

TECHNICAL REPORT ISO/IEC TR 19075-3:2014

Information technology — Database languages — SQL Technical Reports —

Part 3:
SQL Embedded in Programs Using the Java™ Programming Language

1 Scope

This Technical Report describes the support for the use of SQL within programs written in Java.

The Report discusses the following features of the SQL Language:

— The embedding of SQL expressions and statements in programs written in the Java programming language

(Blank page)

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

2.1 ISO and IEC standards

[ISO9075-1] ISO/IEC 9075-1:2011, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*.

[ISO9075-2] ISO/IEC 9075-2:2011, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)*.

[ISO9075-10] ISO/IEC 9075-10:2008, *Information technology — Database languages — SQL — Part 10: Object Language Bindings (SQL/OLB)*.

2.2 Other international standards

[Unicode] The Unicode Consortium, *The Unicode Standard*. (Information about the latest version of the Unicode standard can be found by using the "Latest Unicode Version" link on the "Enumerated Versions of The Unicode Standard" page.)
`http://www.unicode.org/versions/enumeratedversions.html`

[Java] *The Java™ Language Specification, Third Edition*, James Gosling, Bill Joy, Guy Steele, and Gilad Bracha, Prentice Hall, June 14, 2005, ISBN 0-321-24678-0.

[JDBC] *JDBC™ 4.0 Specification*, Final v1.0, Lance Andersen, Sun Microsystems, Inc., November 7, 2006.

[JNDI] *Java Naming and Directory Interface™*, Sun Microsystems, Inc. `http://java.sun.com/-j2se/1.5.0/docs/guide/jndi/index.html`.

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

(Blank page)

3 Use of SQL in programs written in Java

3.1 Design goals

The following items represent the major design features of [ISO9075-10].

— Provide a concise, legible mechanism for embedding SQL-statements in a program that otherwise conforms to [Java].

— Syntactic and semantic check of SQL-statements prior to program execution.

SQL/OLB can use an implementation-defined mechanism at translate time to check embedded SQL-statements to make sure that they are syntactically and semantically correct.

— Allow the syntax and semantics of SQL-statements to be location-independent.

The syntax and semantics of SQL-statements in an SQL/OLB program do not depend on the configuration under which SQL/OLB is running. This makes it possible to implement SQL/OLB programs that run on the client, in the SQL-server, or in a middle tier.

— Provide facilities that enable the programmer to move between the SQL/OLB and JDBC environments by sharing a single SQL-connection in both environments.

— Provide for binary portability of translated and compiled Java SQL-client applications such that they can be used transparently with multiple SQL-servers. In addition, binary portability profiles allow for customization and optimization of SQL-statements within an SQL/OLB application.

3.2 Advantages of SQL/OLB over JDBC

JDBC provides a complete, low-level SQL interface from Java to SQL-implementations. SQL/OLB is designed to fill a complementary role by providing a higher-level programming interface to SQL-implementations in such a manner as to free the programmer from the tedious and complex programming interfaces found in lower-level APIs.

The following are some major differences between the two:

— SQL/OLB source programs are smaller than equivalent JDBC programs since the translator can implicitly handle many of the tedious programming chores that dynamic interfaces require.

— SQL/OLB programs can type-check SQL code at translate time using an implementation-dependent mechanism. JDBC, being a completely dynamic API, can not.

— SQL/OLB programs allow direct embedding of Java host expressions within SQL-statements. JDBC requires a separate call statement for each bind variable and specifies the binding by position number.

— SQL/OLB enforces strong typing of query outputs and values returned and allows type checking on calls. JDBC passes values to and from SQL without compile time type checking.

— SQL/OLB provides simplified rules for invoking SQL-invoked routines. [JDBC] requires a generic call to an SQL-invoked routine, *fun*, to have the following syntax:

```
prepStmt.prepareCall("{call fun(...)}");       // For SQL-invoked procedures
prepStmt.prepareCall("{? = call fun(...)}");   // For SQL-invoked functions
```

SQL/OLB provides simplified notations:

```
#sql { CALL fun(...) };              // SQL-invoked procedure
// Declare x
...
#sql x = { VALUES(fun(...)) };       // SQL-invoked function
                                     // VALUES is an SQL construct
```

3.3 Consistency with existing embedded SQL languages

Programming languages containing embedded SQL are called *host languages*. Java differs from the traditional host languages (Ada, C, COBOL, Fortran, MUMPS (M), Pascal, PL/I) in ways that significantly affect its embedding of SQL.

— Java has automatic storage management (also known as "garbage collection") that simplifies the management of storage for data retrieved from SQL-implementations.

— All Java types representing composite data, and data of varying sizes, have a distinguished value **`null`**, which can be used to represent the SQL **`NULL`** value. This gives Java programs an alternative to the indicator variables that are part of the interfaces to other host languages.

— Java is designed to support programs that are *automatically heterogeneously portable* (also called "super portable" or simply "downloadable"). That, along with Java's type system of classes and interfaces, enables *component software*. In particular, an SQL/OLB translator, written in Java, can call components that are specialized by SQL-implementations, in order to leverage the existing authorization, schema checking, type checking, transactional, and recovery capabilities that are traditional of SQL-implementations, and to generate code optimized for particular SQL-implementations.

— Java is designed for binary portability in heterogeneous networks, which promises to enable binary portability for applications that use SQL.

— SQL/OLB extends the traditional concept of embedded host variables by allowing generalized host expressions.

3.4 Profile customization overview

This Subclause describes how implementation-specific "customized" SQL execution control can be added to SQL/OLB applications. The SQL/OLB runtime framework uses the following interfaces:

— **SQLJ.runtime.profile.RTStatement** to execute SQL-statements.

— **SQLJ.runtime.profile.RTResultSet** to describe query results.

