BS ISO/IEC 19075-4:2021

BSI Standards Publication

Information technology — Guidance for the use of database language SQL

Part 4: Routines and types using the Java™ programming language

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 19075-4:2021

National foreword

This British Standard is the UK implementation of ISO/IEC 19075-4:2021. It supersedes PD ISO/IEC TR 19075-4:2015, which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee IST/40, Data management and interchange.

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

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is, and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

© The British Standards Institution 2021
Published by BSI Standards Limited 2021

ISBN 978 0 539 13081 2

ICS 35.060

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 September 2021.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 19075-4:2021

INTERNATIONAL STANDARD

ISO/IEC 19075-4

First edition
2021-08

Information technology — Guidance for the use of database language SQL —

Part 4: Routines and types using the Java™ programming language

Technologies de l'information — Recommandations pour l'utilisation du langage de base de données SQL —

Partie 4: Routines et types utilisant le langage de programmation Java™

Reference number
ISO/IEC 19075-4:2021(E)

Documents
Document Sharing Hub

© ISO/IEC 2021

Full document details: BS ISO/IEC 19075-4:2021

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2021

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

© ISO/IEC 2021 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 19075-4:2021

Contents

Page

Foreword . . . v
Introduction . . . vii
1 Scope . . . 1
2 Normative references . . . 2
3 Terms and definitions . . . 3
4 Routines tutorial . . . 4
4.1 Context of routines . . . 4
4.2 Technical components . . . 4
4.3 Overview . . . 5
4.4 Example Java methods: region and correctStates . . . 5
4.5 Installing region and correctStates in SQL . . . 6
4.6 Defining SQL names for region and correctStates . . . 7
4.7 A Java method with output parameters: bestTwoEmps . . . 8
4.8 A CREATE PROCEDURE best2 for bestTwoEmps . . . 10
4.9 Calling the best2 procedure . . . 10
4.10 A Java method returning a result set: orderedEmps . . . 11
4.11 A CREATE PROCEDURE rankedEmps for orderedEmps . . . 12
4.12 Calling the rankedEmps procedure . . . 13
4.13 Overloading Java method names and SQL names . . . 14
4.14 Java main methods . . . 15
4.15 Java method signatures in the CREATE statements . . . 16
4.16 Null argument values and the RETURNS NULL clause . . . 17
4.17 Static variables . . . 19
4.18 Dropping SQL names of Java methods . . . 20
4.19 Removing Java classes from SQL . . . 20
4.20 Replacing Java classes in SQL . . . 21
4.21 Visibility . . . 22
4.22 Exceptions . . . 22
4.23 Deployment descriptors . . . 23
4.24 Paths . . . 25
4.25 Privileges . . . 27
4.26 Information Schema . . . 28
5 Types tutorial . . . 29
5.1 Overview . . . 29
5.2 Example Java classes . . . 29
5.3 Installing Address and Address2Line in an SQL system . . . 31
5.4 CREATE TYPE for Address and Address2Line . . . 31
5.5 Multiple SQL types for a single Java class . . . 33

© ISO/IEC 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 19075-4:2021

5.6 Collapsing subclasses. 33
5.7 GRANT and REVOKE statements for data types. 35
5.8 Deployment descriptors for classes. 35
5.9 Using Java classes as data types. 36
5.10 SELECT, INSERT, and UPDATE. 37
5.11 Referencing Java fields and methods in SQL. 38
5.12 Extended visibility rules. 38
5.13 Logical representation of Java instances in SQL. 39
5.14 Static methods. 40
5.15 Static fields. 41
5.16 Instance-update methods. 41
5.17 Subtypes in SQL/JRT data. 43
5.18 References to fields and methods of null instances. 44
5.19 Ordering of SQL/JRT data. 45

Bibliography. 48

Index. 49

© ISO/IEC 2021 – All rights reserved
Documents
Document Sharing Hub

Full document details: BS ISO/IEC 19075-4:2021

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents), or the IEC list of patent declarations received (see patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

This first edition of ISO/IEC 19075-4 cancels and replaces ISO/IEC TR 19075-4:2015.

This document is intended to be used in conjunction with the following editions of the parts of the ISO/IEC 9075 series:

— ISO/IEC 9075-1, sixth edition or later;

— ISO/IEC 9075-2, sixth edition or later;

— ISO/IEC 9075-3, sixth edition or later;

— ISO/IEC 9075-4, seventh edition or later;

— ISO/IEC 9075-9, fifth edition or later;

— ISO/IEC 9075-10, fifth edition or later;

— ISO/IEC 9075-11, fifth edition or later;

— ISO/IEC 9075-13, fifth edition or later;

— ISO/IEC 9075-14, sixth edition or later;

— ISO/IEC 9075-15, second edition or later;

— ISO/IEC 9075-16, first edition or later.

© ISO/IEC 2021 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 19075-4:2021

A list of all parts in the ISO/IEC 19075 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/-national-committees.

© ISO/IEC 2021 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 19075-4:2021

Introduction

The organization of this document is as follows:

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

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

3) Clause 3, "Terms and definitions", defines the terms and definitions used in this document.

4) Clause 4, "Routines tutorial", provides a tutorial on the use of routines written in the Java™[1] programming language within SQL expressions and statements.

5) Clause 5, "Types tutorial", provides a tutorial on the use of user-defined types written in the Java programming language within SQL expressions and statements.

Java™ is the trademark of a product supplied by Oracle. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named.

© ISO/IEC 2021 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 19075-4:2021

© ISO/IEC 2021 – All rights reserved
Documents
Document Sharing Hub

Full document details: BS ISO/IEC 19075-4:2021

Information technology — Guidance for the use of database language SQL —

Part 4:
Routines and types using the Java™ programming language

1 Scope

This document provides a tutorial of SQL routines and types using the Java™ programming language.

This document discusses the following features of the SQL Language:

— The use of routines written in the Java programming language within SQL expressions and statements.

— The use of user-defined types written in the Java programming language within SQL expressions and statements.

© ISO/IEC 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 19075-4:2021

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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.

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

ISO/IEC 9075-13, *Information technology — Database languages — SQL — Part 13: SQL Routines and Types Using the Java™ Programming Language (SQL/JRT)*

Java Community Process. *The Java™ Language Specification* [online]. Java SE 13 Edition. Redwood Shores, California, USA: Oracle, Available at `https://docs.oracle.com/javase/specs/jls/-se13/jls13.pdf`

Java Community Process. *JDBC™ 4.3 Specification* [online]. Edition 4.3. Redwood Shores, California, USA: Oracle, Available at `https://download.oracle.com/otn-pub/jcp/-jdbc-4_3-mrel3-eval-spec/jdbc4.3-fr-spec.pdf`

© ISO/IEC 2021 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 19075-4:2021

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 9075-1 apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at http://www.iso.org/obp

© ISO/IEC 2021 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 19075-4:2021

4 Routines tutorial

4.1 Context of routines

The requirements for the material discussed in this document shall be as specified in ISO/IEC 9075-1, ISO/IEC 9075-13, Java, and JDBC.

4.2 Technical components

ISO/IEC 9075-13 includes the following:

— New built-in procedures.

- `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.

— New built-in schema.

The built-in schema named `SQLJ` is assumed to be in all catalogs of an SQL system that implements the SQL/JRT facility, and to contain all of the built-in procedures of the SQL/JRT facility.

— Extensions of the following SQL statements:

- CREATE PROCEDURE/FUNCTION — to specify an SQL name for a Java method.
- DROP PROCEDURE/FUNCTION — to delete the SQL name of a Java method.
- CREATE TYPE — to specify an SQL name for a Java class.
- DROP TYPE — to delete the SQL name of a Java class.
- GRANT — to grant the USAGE privilege on Java JARs.
- REVOKE — to revoke the USAGE privilege on Java JARs.

— Conventions for returning values of OUT and INOUT parameters, and for returning SQL result sets.

— New forms of reference: Qualified references to the fields and methods of columns whose data types are defined on Java classes.

— Additional views and columns in the Information Schema.

© ISO/IEC 2021 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 19075-4:2021

4.3 Overview

This tutorial shows a series of example Java classes, indicates how they can be installed, and shows how their static, public methods can be referenced with SQL/JRT facilities in an SQL-environment.

The example Java methods assume an SQL table named EMPS, with the following columns:

— NAME — the employee's name.

— ID — the employee's identification.

— STATE — the state in which the employee is located.

— SALES — the amount of the employee's sales.

— JOBCODE — the job code of the employee.

The table definition is:

```
CREATE TABLE emps (
    name    VARCHAR(50),
    id      CHARACTER(5),
    state   CHARACTER(20),
    sales   DECIMAL (6,2),
    jobcode INTEGER );
```

The example classes and methods are:

— `Routines1.region` — A Java method that maps a US state code to a region number. This method does not use SQL internally.

— `Routines1.correctStates` — A Java method that performs an SQL UPDATE statement to correct the spelling of *state* codes. The old and new spellings are specified by input-mode parameters.

— `Routines2.bestTwoEmps` — A Java method that determines the top two employees by their sales, and returns the columns of those two employee rows as output-mode parameter values. This method creates an SQL result set and processes it internally.

— `Routines3.orderedEmps` — A Java method that creates an SQL result set consisting of selected employee rows ordered by the sales column, and returns that result set to the client.

— `Over1.isOdd` and `Over2.isOdd` — Contrived Java methods to illustrate overloading rules.

— `Routines4.job1` and `Routines5.job2` — Java methods that return a string value corresponding to an integer jobcode value. These methods illustrate the treatment of null arguments.

— `Routines6.job3` — Another Java method that returns a string value corresponding to an integer jobcode value. This method illustrates the behavior of static Java variables.

Unless otherwise noted, the methods that invoke SQL use JDBC. One of the methods is shown in both aversion using JDBC and a version using SQL/OLB. The others could also be coded with SQL/OLB.

It is assumed that the import statements `import java.sql.*;` and `java.math.*;` have been included in all classes.

4.4 Example Java methods: region and correctStates

This clause shows an example Java class, `Routines1`, with two simple methods.

© ISO/IEC 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 19075-4:2021

— The `int`-valued static method `region` categorizes 9 states into 3 geographic regions, returning an integer indicating the region associated with a valid state or throwing an exception for invalid states. This method will be called as a function in SQL.

— The `void` method `correctStates` updates the EMPS table to correct spelling errors in the state column. This method will be called as a procedure in SQL.

```
public class Routines1 {
  //An int method that will be called as a function
  public static int region(String s) throws SQLException {
    if (s.equals("MN") || s.equals("VT") || s.equals("NH")) return 1;
      else if (s.equals("FL") || s.equals("GA") || s.equals("AL")) return 2;
      else if (s.equals("CA") || s.equals("AZ") || s.equals("NV")) return 3;
      else throw new SQLException("Invalid state code", "38001");
  }
  //A void method that will be called as a stored procedure
  public static void correctStates (String oldSpelling, String newSpelling)
      throws SQLException {
    Connection conn = DriverManager.getConnection ("jdbc:default:connection");
    PreparedStatement stmt = conn.prepareStatement
        ("UPDATE emps SET state = ? WHERE state = ?");
    stmt.setString(1, newSpelling);
    stmt.setString(2, oldSpelling);
    stmt.executeUpdate();
    stmt.close();
    conn.close();
    return;
  }
}
```

4.5 Installing region and correctStates in SQL

The source code for Java classes such as `Routines1` will normally be in one or more Java files (i.e.,, files with file-type “java”). When they are compiled (using the `javac` compile command), the resulting code will be in one or more class files (i.e.,, files with file-type “class”). A set of class files is then collected into a Java JAR, which is a ZIP-coded collection of files.

To use Java classes in SQL, a JAR containing them is loaded into the SQL system by calling the SQL `SQLJ.INSTALL_JAR` procedure. The `SQLJ.INSTALL_JAR` procedure is a new built-in SQL procedure that makes the collection ofJava classes contained in a specified JAR available for use in the current SQL catalog. For example, assume that the above `Routines1` class has been assembled into a JAR with local file name “`~/classes/Routines1.jar`”:

```
SQLJ.INSTALL_JAR('file:~/classes/Routines1.jar', 'routines1_jar', 0)
```

— The first parameter of the `SQLJ.INSTALL_JAR` procedure is a character string specifying the URL of the given JAR. This parameter is never folded to upper case.

— The second parameter of the `SQLJ.INSTALL_JAR` procedure is a character string that will be used as the name of the JAR in the SQL system. The JAR name is an SQL qualified name, and follows SQL conventions for qualified names.

  The JAR name specified as the second parameter of the `SQLJ.INSTALL_JAR` procedure identifies the JAR within the SQL system. That is, the JAR name specified is used only in SQL, and has nothing to do with the contents of the JAR itself. The JAR name is used in the following contexts, which are described in later clauses:

  - As a parameter of the `SQLJ.REMOVE_JAR` and `SQLJ.REPLACE_JAR` procedures.

© ISO/IEC 2021 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 19075-4:2021

- As a qualifier of Java class names in SQL CREATE PROCEDURE/FUNCTION statements.
- As an operand of the extended SQL GRANT and REVOKE statements.
- As a qualifier of Java class names in SQL CREATE TYPE statements.

The JAR name may also be used in follow-on facilities for downloading JARs from the SQL system.

— JARs can also contain *deployment descriptors*, which specify implicit actions to be taken by the `SQLJ.INSTALL_JAR` and `SQLJ.REMOVE_JAR` procedures. The third parameter of the `SQLJ.INSTALL_JAR` procedure is an integer that specifies whether or not (indicated by non-zero or zero values, respectively) the `SQLJ.INSTALL_JAR` procedure is expected to execute the actions specified by a deployment descriptor in the JAR.

The name of the INSTALL_JAR procedure is qualified with the schema name SQLJ. All built-in procedures of the SQL/JRT facility are defined to be contained in that built-in schema. The SQLJ schema is assumed to be present in each catalog of an SQL system that implements the SQL/JRT facility.

The first two parameters of `SQLJ.INSTALL_JAR` are character strings, so if they are specified as literals, single quotes are used, not the double quotes used for SQL delimited identifiers.

The actions of the `SQLJ.INSTALL_JAR` procedure are as follows:

— Obtain the JAR designated by the first parameter.

— Extract the class files that it contains and install them into the current SQL schema.

— Retain a copy of the JAR itself, and associate it with the value of the second parameter.

— If the third parameter is non-zero, then perform the actions specified by the deployment descriptor of the JAR.

After a JAR has been installed with the `SQLJ.INSTALL_JAR` procedure, the static methods of the classes contained in that JAR can be referenced in the CREATE PROCEDURE/FUNCTION statement, as described in the next Subclause.

4.6 Defining SQL names for region and correctStates

Before a Java method can be called in SQL, that method must have an SQL name. This is done with new options on the SQL CREATE PROCEDURE/FUNCTION statement. For example:

```
CREATE PROCEDURE correct_states(old CHARACTER(20), new CHARACTER(20))
  MODIFIES SQL DATA
  LANGUAGE JAVA PARAMETER STYLE JAVA
  EXTERNAL NAME 'routines1_jar:Routines1.correctStates';
CREATE FUNCTION region_of(state CHARACTER(20)) RETURNS INTEGER
  NO SQL
  LANGUAGE JAVA PARAMETER STYLE JAVA
  EXTERNAL NAME 'routines1_jar:Routines1.region';
```

The CREATE PROCEDURE and CREATE FUNCTION statements specify SQL names and Java method signatures for the Java methods specified in the EXTERNAL NAME clauses. The format of the method names in the external name clause consists of the JAR name that was specified in the `SQLJ.INSTALL_JAR` procedure followed by the Java method name, fully qualified with the package name(s) (if any) and class name.

The CREATE PROCEDURE for `correct_states` specifies the clause MODIFIES SQL DATA. This indicates that the specified Java method modifies (via INSERT, UPDATE, or DELETE) data in SQL tables. The CREATE FUNCTION for `region_of` specifies NO SQL. This indicates that the specified Java method performs no SQL operations.

© ISO/IEC 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 19075-4:2021

Other clauses that can be specified are READS SQL DATA, which indicates that the specified Java method reads (through SELECT) data in SQL tables, but does not modify SQL data, and CONTAINS SQL, which indicates that the specified method invokes SQL operations, but neither reads nor modifies SQL data. The alternative CONTAINS SQL is the default.

The SQL procedure and function names that are define with such CREATE PROCEDURE/FUNCTION statements are used as normal SQL procedure and function names:

```
SELECT name, region_of(state) AS region
FROM emps
WHERE region_of(state) = 3;
CALL correct_states ('GEO', 'GA');
```

Multiple SQL names for the same Java method can be defined:

```
CREATE PROCEDURE state_correction(old CHARACTER(20), new CHARACTER(20))
  MODIFIES SQL DATA
  LANGUAGE JAVA PARAMETER STYLE JAVA
  EXTERNAL NAME 'routines1_jar:Routines1.correctStates';
CREATE FUNCTION state_region(state CHARACTER(20)) RETURNS INTEGER
  LANGUAGE JAVA PARAMETER STYLE JAVA
  EXTERNAL NAME 'routines1_jar:Routines1.region';
```

The various SQL function and procedure names for a Java method can be used equivalently:

```
SELECT name, state_region(state) AS region
FROM emps
WHERE region_of(state) = 2;
CALL state_correction ('ORE', 'OR');
```

The SQL names are normal 3-part SQL names, and the first two parts of the 3-part names are defaulted as defined in SQL for CREATE PROCEDURE and CREATE FUNCTION statements.

There are other considerations for the CREATE PROCEDURE/FUNCTION statement, dealing with parameter data types, overloaded names, and privileges, which is discussed in later Subclauses.

4.7 A Java method with output parameters: bestTwoEmps

The parameters of the `region` and `correctStates` methods are all input-only parameters. This is the normal Java parameter convention.

SQL procedures also support parameters with mode OUT and INOUT. The Java language does not directly have a notion of output parameters. SQL/JRT therefore uses arrays to return output values for parameters of Java methods. That is, if an `Integer` parameter is to be used to return a value to the caller, the type of that parameter is specified to be `Integer[ ]`, i.e., an array of `Integer`. Such an array will contain only one element: the input value of the parameter is contained in that element when the method is called, and the method sets the value of that element to the desired output value.

As seen in the following clauses, this use of arrays for output parameters in the Java methods is visible only to the Java method. When such a method is called as an SQL procedure, normal scalar data items are supplied as parameters. The SQL system performs the mapping between those scalar data items and Java arrays implicitly.

The following Java method illustrates the way that output parameters are coded in Java. This method, `bestTwoEmps`, returns the `name`, `id`, `region`, and `sales` of the two employees that have the highest `sales` in the regions with numbers higher than a parameter value. That is, each of the first 8 parameters is an OUT parameter, and is therefore declared to be an array of the given type.

© ISO/IEC 2021 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 19075-4:2021

The following version of the `bestTwoEmps` method uses SQL/OLB for statements that access SQL:

```
public class Routines2 {
  public static void bestTwoEmps (
      String[ ] n1, String[ ] id1, int[ ] r1, BigDecimal[ ] s1,
      String[ ] n2, String[ ] id2, int[ ] r2, BigDecimal[ ] s2,
      int regionParm) throws SQLException {
    #sql iterator ByNames (String name, String id, int region, BigDecimal sales);
    n1[0]= "****"; n2[0]= "****"; id1[0]= ""; id2[0]= "";
    r1[0]=0; r2[0]=0; s1[0]= new BigDecimal(0); s2[0]= new BigDecimal(0);
    ByNames r = null;
    try {
      #sql r = {SELECT name, id, region_of(state) AS region, sales
                FROM emp
                WHERE region_of(state) > :regionParm
                  AND sales IS NOT NULL
                ORDER BY sales DESC};
      if (r.next()) {
        n1[0] = r.name();
        id1[0] = r.id();
        r1[0] = r.region();
        s1[0] = r.sales();
      }
      else return;
      if (r.next()) {
        n2[0] = r.name();
        id2[0] = r.id();
        r2[0] = r.region();
        s2[0] = r.sales();
      }
      else return;
    } finally {
        if (r != null) r.close();
      }
  }
}
```

Note that since the above Java method uses SQL/OLB for SQL operations, it does not have to explicitly obtain a connection to the SQL system. By default, SQL/OLB executes any SQL contained in a routine in the context of the SQL statement invoking that routine.

For comparison, here's a version of the `bestTwoEmps` method using JDBC instead of SQL/OLB:

```
public class Routines2 {
  public static void bestTwoEmps (
      String[ ] n1, String[ ] id1, int[ ] r1, BigDecimal[ ] s1,
      String[ ] n2, String[ ] id2, int[ ] r2, BigDecimal[ ] s2,
      int regionParm) throws SQLException {
    n1[0]= "****"; n2[0]= "****"; id1[0]= ""; id2[0]= "";
    r1[0]=0; r2[0]=0; s1[0]= new BigDecimal(0); s2[0]= new BigDecimal(0);
    java.sql.PreparedStatement stmt = null;
    try {
      Connection conn = DriverManager.getConnection
          ("jdbc:default:connection");
      stmt.conn.prepareStatement
          ("SELECT name, id, region_of(state) AS region, sales
           FROM emp
           WHERE region_of(state) > ?
             AND sales IS NOT NULL
           ORDER BY sales DESC");
      stmt.setInt(1, regionParm)
      ResultSet r = stmt.executeQuery();
```

© ISO/IEC 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 19075-4:2021