Red Database

Version 2.5

Extended stored procedures and functions

This document contents description of using external stored procedures (ESP) and functions (EUDF) on Java in DBMS Red Database 2.5. Document is intended for users knowing with DBMS Red Database, SQL, PSQL, and Java.

www.red-soft.biz

Table of contents

Introduction

Opportunity of creation External Stored Procedures (ESP) and External User Defined Functions (EUDF) using the Java has been added in Red Database 2.5. This feature essentially expands opportunities of PSQL for data process. For example, PSQL does not allow to work with objects of file system, but it is possible in Java ESP.

External Stored Procedures and External User Defined Functions have some advantages and before UDF:

- access to a context of a call. It allows to use ESP and EUDF in triggers;
- ESP can return data sets (ResultSet). Thus External Stored Procedures can be sources of data.
- ESP and EUDF allow to carry out operations between various databases.

EUDF always return one value any of types described in section «Conformity of types of data SQL and Java». External Stored Procedures have no output parameters, or return a data set (ResultSet.) EUDF cannot return a data set.

Described features demand JDK 1.6 or hight. Setup of parameters of interaction between server Red Database and virtual machine Java is carried out in configuration file jvm.conf. This file is located in the root directory of installation of a server.

It is used character «#» for a designation of comments in this file. The text following a character «#», up to the end of a line also is the comment. In a file it is necessary to set up following parameters:

- full path to the virtual machine
- path to to java-libraries

Full path to the virtual machine is specified in the first non comment line. You can specify either a path, or a name of a environment variable which stores it:

```
C:\Java1_6\jre\bin\client\jvm.dll #full path
<jvm_from_JAVA_HOME> #name of a environment variable
```

Path to java-libraries, that content body of ESP and EUDF, is specified by parameter -cp (abbr. class path). *.jar files , located in java_lib directory relatively of root directory of Red Database, are always loaded by server. Parameter -cp allows to specify additional *.jar files which will be loaded together with *.jar files from a directory java_lib. All names of jar-files are listed in one line. As a separator is used the character «;» for Windows operating systems, and a character «:» for Unix operating systems:

```
-cp java_lib/commons-beanutils-
1.6.1.jar;java_lib/commons-collections-2.1.jar
```

Presence of following Java-libraries is necessary for correct work External Stored procedures and external User Defined Functions:

- jaybird-esp-2.1.6.jar
- jaybird-full-2.1.6.jar

These libraries should be in subdirectory java-lib or pathes to them should be registered in parameter -cp of a configuration file jvm.conf. Also presence in the catalog plugins a server of library javaespudf.dll (javaespudf.so) is necessary.

If libraries with ESP and EUDF are located not in the catalog by default (java_lib) pathes to them also should be certain in parameter -cp a file jvm.conf.

1 Work with ESP and EUDF

1.1 Declaration ESP and EUDF on Java

Syntax of definition External Stored Procedures (ESP), written on Java, is follow:

```
CREATE PROCEDURE <PROCEDURE_NAME>
[(<PARAM> <DATATYPE> [,<PARAM> <DATATYPE> ...])]
[RETURNS (<PARAM> <DATATYPE> [,<PARAM> <DATATYPE>
...])]
LANGUAGE <LANGUAGE_NAME>
EXTERNAL NAME <'UNIQUE_PROCEDURE_IDENTIFIER'>;
```

- <PROCEDURE_NAME> – unique admissible name of External Stored Procedure
- <PARAM> <DATATYPE> – admissible names and types of input/output parameters
- <LANGUAGE_NAME> – language on which External Stored Procedure is written. This parameter should matter JAVA.
- UNIQUE_PROCEDURE_IDENTIFIER – expression containing a unique name of External Stored Procedure in apostrophes. At use of Java-procedures, it contains a name of a package, a classname and a name of the method, separated by a symbol «.».

Example of registration of External Stored Procedure:

```
CREATE PROCEDURE EXAMPLE_PROC(S CHAR(5))
LANGUAGE JAVA
EXTERNAL NAME 'esp.ExampleClass.exampleMethod';
```

In the given example External Stored Procedure is registered with name EXAMPLE_PROC and has one input parameter of type CHAR (5). In the expression containing a unique name of external procedure are presented: esp - a name of a package, ExampleClass - a classname, exampleMethod - a name of the method containing a body of External Stored Procedure.

Syntax of definition External User Defined Function (EUDF), written on Java, is follow:

```
DECLARE EXTERNAL FUNCTION <FUNCTION_NAME>
[<DATATYPE> [,<DATATYPE> ...]]
RETURNS <DATATYPE>
LANGUAGE <LANGUAGE_NAME>
EXTERNAL NAME <'UNIQUE_FUNCTION_IDENTIFIER'>;
```

- <FUNCTION_NAME> – unique admissible name of External User Defined

Function

- <DATATYPE> – admissible names and types of input/output parameters
- LANGUAGE_NAME – language on which External User Defined Function is written. This parameter should matter JAVA.
- UNIQUE_FUNCTION_IDENTIFIER – expression containing a unique name of External User Defined Function in apostrophes. At use of Java-functions, it contains a name of a package, a classname and a name of the method, separated by a symbol «.».

Example of registration of External User Defined Function:

```
DECLARE EXTERNAL FUNCTION EXAMPLE_FUN VARCHAR(20)
RETURNS VARCHAR(20)
LANGUAGE JAVA
EXTERNAL NAME 'esp.ExampleClass.exampleMethod';
```

Realized in «Ред the Database» 2.5 way of registration of ESP and EUDF written in language Java, differs from standard SQLJ, but it is closer to a current way of the description Stored Procedures.

1.2 Re-registration External Stored Procedures

For a re-registration of External Stored Procedures (change of their metadata) operator ALTER PROCEDURE is used. Its syntax is follow:

```
ALTER PROCEDURE PROCEDURE_NAME
[(<PARAM> <DATATYPE> [,<PARAM> <DATATYPE> ...])]
[RETURNS (<PARAM> <DATATYPE> [,PARAM <DATATYPE> ...])]
LANGUAGE LANGUAGE_NAME
EXTERNAL NAME 'UNIQUE_PROCEDURE_IDENTIFIER';
```

Example:

```
ALTER PROCEDURE EXAMPLE_PROC(
     S CHAR(100)
)
LANGUAGE JAVA
EXTERNAL NAME 'esp.ExampleClass.exampleMethod';
```

At a re-registration of External Stored Procedures you can change structure and types of data of input/output parameters, and also the expression containing a unique name of External Stored Procedure.

In Red Database 2.5 there also is an opportunity to register new External Stored Procedure, and if such procedure already exists to re-register it. For this purpose it is possible to use operator CREATE OR ALTER PROCEDURE or RECREATE PROCEDURE. Syntax of operator CREATE OR ALTER PROCEDURE:

```
CREATE OR ALTER PROCEDURE PROCEDURE_NAME
[(PARAM <DATATYPE> [,PARAM <DATATYPE> ...])]
[RETURNS (PARAM <DATATYPE> [,PARAM <DATATYPE> ...])]
LANGUAGE LANGUAGE_NAME
EXTERNAL NAME 'UNIQUE_PROCEDURE_IDENTIFIER';
```

Syntax of operator `RECREATE PROCEDURE`:

```
RECREATE PROCEDURE PROCEDURE_NAME
[(PARAM <DATATYPE> [,PARAM <DATATYPE> ...])]
[RETURNS (PARAM <DATATYPE> [,PARAM <DATATYPE> ...])]
LANGUAGE LANGUAGE_NAME
EXTERNAL NAME 'UNIQUE_PROCEDURE_IDENTIFIER';
```

Operator ALTER EXTERNAL FUNCTION allow to change name of EUDF. Its syntax:

```
ALTER EXTERNAL FUNCTION <FUNCTION_NAME>
EXTERNAL NAME 'NEW_UNIQUE_FUNCTION_IDENTIFIER';
```

Here NEW_UNIQUE_FUNCTION_IDENTIFIER is new unique name External User Defined Function.

1.3 Deleting ESP and EUDF

Deleting of External Stored Procedure is carried out by operator DROP PROCEDURE. Its syntax:

```
DROP PROCEDURE PROCEDURE_NAME
```

Example:

```
DROP PROCEDURE EXAMPLE_PROC
```

Attention! Unlike usual stored procedures, the server cannot check presence of calls of deleted procedure from other ESP or EUDF. Therefore deleting of such procedure will pass without errors, however by the subsequent call of ESP will be сгенерированно exception.

Deleting of External User Defined Function carry out operator DROP EXTERNAL FUNCTION. Its syntax:

```
DROP EXTERNAL FUNCTION FUNCTION_NAME
```

Example:

```
DROP EXTERNAL FUNCTION EXAMPLE_FUN
```

1.4 Call of ESP and EUDF on Java

The call of External Stored Procedures and External User Defined Functions is similar to a call usual Stored Procedures (SP) and the User Defined Functions (UDF). For example, the call of External Stored Procedure can be carried out by operator

EXECUTE PROCEDURE. ESP will be always carried out with the rights of the user calling its.

2 Rules of a writing of ESP and EUDF

2.1 Conformity of data types of SQL and Java languages

At a writing of External Stored Procedures and External User Defined Functions, it is necessary to use presented below the table of conformity of data types used in SQL of Red Database, and Java programming language.

Table 1 - Conformity of data types of SQL and Java languages

SQL data type	Java data types	
	Primitive type	Wrapper type
CHAR	-	String
VARCHAR	-	String
NUMERIC	-	java.math.BigDecimal
SMALLINT	short	Short
INTEGER	int	Integer
BIGINT	long	Long
FLOAT	float	Float
DOUBLE	double	Double
BLOB	-	org.firebirdsql.jdbc.FirebirdBlob
DATE	-	java.sql.Date
TIME	-	java.sql.Time
TIMESTAMP	-	java.sql.Timestamp

2.2 Rules of a writing of a body of ESP and EUDF on Java

1. As body ESP, written in language Java, any method declared as public static, and belonging to a public class (not to an internal inner class) can be used. At a creating of a body of External Stored Procedures on Java, you can only use java.sql. ResultSet and void data types as output parameters. As input parameters, you can use of any data types and their combinations described in section «Conformity of data types SQL and Java languages».
2. ResultSet can be received as result of internal query, query to an external database or in another way. For example, ResultSet can be calculated by means of any algorithm.
3. At a writing of External User Defined Functions on Java methods can return result of any admissible type, and also to contain up to 10 input parameters.

Example of the description of a body of ESP and EUDF on Java:

```
package example;
public class ExampleClass {
  //ESP, returning void
  public static void exampleProc1(String s) {
    //operators on Java
```

```
  }
  //ESP, returning ResultSet
  public static java.sql.ResultSet exampleProc2() {
    //operators on Java
  }
  //EUDF, returning String
  public static String exampleFun1(int i) {
    //operators on Java
  }
}
```

The overload of methods is resolved, but it should be avoided. Because, for example, methods with parameters int and Integer differ in Java, but they do not differ for ESP and EUDF in Red Database.

After a writing of the classes containing methods which will be used as a body of ESP and EUDF, it is necessary to create a jar-file. It is possible to make, using the utility jar, entered in JDK, or using of integrated development environments of java-programs (for example, IntelliJ IDEA, Eclipse, etc.).

The received jar-file, it is necessary to copy in the java_lib directory, being a folder where Red Database is installed. You can also specify a path to it in a file jvm.conf in parameter -cp.

2.3 Access to a call context from a method on Java

In conformity with standard SQLJ, in Red Database 2.5 it is available an opportunity of access to a call context ESP or EUDF. For this purpose it is necessary to use URL like " jdbc:default:connection: ", for example:

```
Connection connection =
DriverManager.getConnection("jdbc:default:connection:")
;
```

In this case all calls connection.commit() and connection.rollback() will be ignored.

Besides in addition to standard SQLJ, in Red Database it is possible to get access from a java-method to a database with separate transaction. For this purpose it is used URL like " jdbc:new:connection: ", for example:

```
Connection connection =
DriverManager.getConnection("jdbc:new:connection:");
```

It will allow to execute, for example, DDL or other operation demanding autonomy for performance.

2.4 Support of triggers

It is impossible to declare the trigger written in language Java. But it is possible to call ESP or EUDF on Java from the trigger written on PSQL. It is also possible to get access to a context of the trigger from ESP or EUDF by methods of object of a class

org.firebirdsql.javaudf:

- Getting of table name, for which the trigger is called (a method getTable());
- Getting of the operation which have called the trigger (a method getTriggerAction ()). Possible values:
  - insert of record (returned value 1);
  - change of record (returned value 2);
  - deleting of record (returned value 3);
- Getting of value "new.<name>" in the form of object (a method getObject_New(java.lang.String string), where string is a name of a field of the table);
- Getting of value "old.<name>" in the form of object (a method getObject_Old(java.lang.String string), where string is a name of a field of the table);
- Installation of value "new.<name>" in the form of object (a method setNewValue(java.lang. String string, java.lang.Object object), where string is a name of a field of the table, and object is new value of this field).

Example shows getting from the EUDF called from the trigger of table name for which the trigger is called:

```
public static String getTable() throws SQLException {
  Trigger trigger = new Trigger();
  return trigger.getTable();
}
```

3 Examples of ESP and EUDF

3.1 Calculation of a factorial of number

In following listing description EUDF, written on Java, is shown which makes recursive calculation of a factorial of number:

```
public static long factorial(int x) throws SQLException
{
  Connection con =
DriverManager.getConnection("jdbc:default:connection:")
;
  PreparedStatement pstmt =
con.prepareStatement("select factor(?) from
rdb$database");
  if ((x == 0) || (x == 1)) {
    return 1;
  }
  else {
    pstmt.setLong(1, x - 1);
    ResultSet rs = pstmt.executeQuery();
    rs.next();
    return x * rs.getLong(1);
  }
}
```

The code, necessary for registration of External User Defined Function of calculation of a factorial, is shown below:

```
DECLARE EXTERNAL FUNCTION FACTOR INTEGER
RETURNS BIGINT
LANGUAGE JAVA
EXTERNAL NAME 'esp.TestESP.factorial';
COMMIT;
```

Result of performance of following query will be number 120:

```
SELECT FACTOR(5) FROM RDB$DATABASE;
```

3.2 Example of work with file system

In following listing description ESP, written on Java, is shown which illustrates a work example with file system:

```
public static void loadFile(String path) throws
SQLException, FileNotFoundException, IOException {
  InputStream is = new FileInputStream(path);
  byte[] b = new byte[is.available()];
  is.read(b);
  Connection con =
DriverManager.getConnection("jdbc:default:connection:")
;
  PreparedStatement pstmt =
con.prepareStatement("INSERT INTO test_table(b) values
(?)");
  try {
    pstmt.setBytes(1, b);
    pstmt.execute();
  }
  finally {
    pstmt.close();
  }
}
```

The parameter of string type, setting a path to a file, is passed in ESP. Inside of procedure reading this file is carried out in a array of type byte[]. Then there is an insert in the table test_table containing a field of BLOB type.

Scripts of creation of the table test_table and registration ESP are shown below:

```
create table test_table(
     b blob
);
CREATE OR ALTER PROCEDURE TEST(S CHAR(100))
LANGUAGE JAVA
EXTERNAL NAME 'esp.TestESP.loadFile';
```

Example of SQL-query of running ESP:

```
execute procedure test('c:\image.jpg')
```

3.3 Example of ESP returning a data set

In following listing description ESP, written on Java, is shown which returns a data set:

```
public static ResultSet testRS() throws SQLException {
  Connection con =
```

```
DriverManager.getConnection("jdbc:new:connection:");
  PreparedStatement pstmt =
con.prepareStatement("SELECT * FROM test_table");
  return pstmt.executeQuery();
}
```

The External Stored Procedure described in an example makes selecting of data from the table test_table.

Scripts of creation of the table test_table and registration of ESP are shown below:

```
CREATE TABLE TEST_TABLE(
     F_INTEGER INTEGER,
     F_VARCHAR VARCHAR(10)
);
CREATE OR ALTER PROCEDURE TEST
RETURNS(
     F_INTEGER INTEGER,
     F_VARCHAR VARCHAR(10)
)
LANGUAGE JAVA
EXTERNAL NAME 'esp.TestESP.testRS';
```

Result of work of External Stored Procedure testRS is possible to see having executed query:

```
SELECT * FROM TEST;
```

3.4 Example of ESP working with other database

External Stored Procedure allows to connect to other databases means of JDBC. The example of body ESP, carrying out connection to other database is shown below:

```
public static void interConnect() throws SQLException {
  Connection con =
DriverManager.getConnection("jdbc:new:connection:");
  PreparedStatement pstmt =
con.prepareStatement("INSERT INTO test_table values
(?)");
  try {
    String url =
"jdbc:firebirdsql:localhost:D:/testbase.fdb";
    Connection extCon =
```

```
DriverManager.getConnection(url, "SYSDBA",
"masterkey");
    PreparedStatement extPstmt =
extCon.prepareStatement("SELECT * FROM test_table");
    try {
      ResultSet rs = extPstmt.executeQuery();
      while (rs.next()) {
        pstmt.setString(1, rs.getString(1));
        pstmt.execute();
      }
    }
    finally {
      extPstmt.close();
    }
  }
  finally {
    pstmt.close();
  }
}
```

Scripts of creation of the table test_table and registration of External Stored Procedure are shown below:

```
CREATE TABLE test_table(
F_VARCHAR VARCHAR(50)
)
CREATE OR ALTER PROCEDURE TEST
LANGUAGE JAVA
EXTERNAL NAME 'esp.TestESP.interConnect';
```

It is necessary to use operator EXECUTE PROCEDURE for a call of External Stored Procedure TEST:

```
EXECUTE PROCEDURE TEST;
```