Ред База Данных

Версия 2.5

Внешние хранимые процедуры и функции

Данный документ содержит описание использования синтаксиса внешних хранимых процедур и функций на языке Java в СУБД «Ред База Данных» 2.5. Документ рассчитан на пользователей, знакомых с принципами организации баз данных СУБД «Ред База Данных», языком SQL и PSQL, а также языком Java

www.red-soft.biz

Содержание

Введение

В «Ред База Данных» 2.5 добавлена возможность создания внешних процедур (External Stored Procedure – ESP) и внешних функций (External User Defined Function -- EUDF) с использованием языка программирования Java, что существенно расширяет возможности языка PSQL по обработке данных. Например, с помощью PSQL невозможно работать с объектами файловой системы, а язык Java позволяет это.

Кроме того, внешние процедуры и функции имеют ряд преимуществ и перед UDF:

- возможность получить доступ к контексту вызова. Это позволяет использовать внешние хранимые процедуры в триггерах;
- внешние процедуры могут возвращать наборы данных — ResultSet. Таким образом внешние хранимые процедуры могут выступать как источники данных;
- с помощью внешних процедур и функций можно выполнять операции между различными базами данных.

В отличие от внешних процедур, внешние функции всегда возвращают одно значение какого-либо из типов описанных в разделе «Соответствие типов данных SQL и Java». Внешние процедуры либо не имеют выходных параметров, либо возвращают набор данных (ResultSet). EUDF не могут возвращать набор данных.

Для использования описываемого в этом руководстве функционала необходимо установить JDK не ниже 1.6. Настройка параметров взаимодействия сервера «Ред База Данных» с виртуальной машиной Java осуществляется с помощью конфигурационного файла jvm.conf, который расположен в корневом каталоге установки сервера.

Для обозначения комментариев в этом файле используется символ «#». Текст, следующий за символом «#», до конца строки также является комментарием. В файле необходимо настроить следующие параметры:

- полный путь к виртуальной машине
- путь к java-библиотекам

Полный путь к виртуальной машине указывается в первой незакомментированной строке. При этом может быть указан как путь, так и имя переменной окружения, которая его хранит:

```
C:\Java1_6\jre\bin\client\jvm.dll #полный путь

<jvm_from_JAVA_HOME> #имя переменной окружения
```

Путь к Java-библиотекам, которые содержат тело внешних процедур и функций задается с помощью параметра -cp (сокр. от class path). По умолчанию сервер подгружает *.jar файлы, расположенные только в директории java_lib, относительно корневой директории сервера. С помощью параметра –cp можно указать дополнительные *.jar фалы, которые будут подгружаться вместе с *.jar файлами из директории java_lib. При этом все имена jar-файлов перечисляются в одну строку, в качестве разделителя используется символ «;» для Windows систем, и символ «:» для Unix систем:

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

Для корректной работы внешних хранимых процедур и функций необходимо наличие следующих Java-библиотек:

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

Эти библиотеки должны располагаться либо в подкаталоге java-lib либо пути к ним должны быть прописаны в параметре -cp в конфигурационном файле jvm.conf.

Также необходимо наличие в каталоге plugins сервера библиотеки javaespudf.dll (javaespudf.so)

Если библиотеки с внешними хранимыми процедурами и функциями расположены не в каталоге по умолчанию (java_lib), то пути к ним также должны быть определены в параметре -cp файла jvm.conf.

1 Работа с внешними хранимыми процедурами

1.1 Объявление внешних процедур и функций на Java

Синтаксис определения внешней хранимой процедуры (External Storage Procedures, ESP), написанной на Java, приведен в листинге 1:

Листинг 1.1 Объявление внешней хранимой функции (ESP)

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

Здесь:

- <PROCEDURE_NAME> – уникальное допустимое имя внешней процедуры;
- <PARAM> <DATATYPE> – допустимые имена и типы входных и возвращаемых параметров.
- <LANGUAGE_NAME> – язык, на котором написана внешняя процедура. Этот параметр должен иметь значение JAVA.
- UNIQUE_PROCEDURE_IDENTIFIER – выражение, содержащее уникальное имя внешней процедуры в апострофах. При использовании Java-процедур, оно содержит имя пакета, имя класса и имя метода, разделённых символом «.».

Пример регистрации внешней процедуры:

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

В данном примере внешняя процедура регистрируется с именем EXAMPLE_PROC и имеет один входной параметр типа CHAR(5). В выражении, содержащем уникальное имя внешней процедуры: esp – имя пакета, ExampleClass – имя класса, exampleMethod – имя метода, содержащего тело внешней процедуры.

Листинг 1.2 Объявление внешней функции (EUDF)

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

- <FUNCTION_NAME> – уникальное допустимое имя внешней функции
- <DATATYPE> – допустимые типы входных и возвращаемых параметров.
- LANGUAGE_NAME – язык, на котором написана внешняя функция. Этот параметр должен иметь значение JAVA.
- UNIQUE_FUNCTION_IDENTIFIER – выражение, содержащее уникальное имя внешней функции в апострофах. При использовании Java-функций, оно содержит имя пакета, имя класса и имя метода, разделённых символом «.».

Пример регистрации EUDF:

```
DECLARE EXTERNAL FUNCTION EXAMPLE_FUN VARCHAR(20)
   RETURNS VARCHAR(20)
   LANGUAGE JAVA
```

```
   EXTERNAL NAME 'esp.ExampleClass.exampleMethod';
```

Реализованный в «Ред База Данных» 2.5 вариант регистрации внешних процедур и функций, написанных на языке Java, отличается от стандарта SQLJ, но он более близок к текущему варианту описания хранимых процедур.

1.2 Изменение внешних хранимых процедур

Для перерегистрации внешних процедур (изменения их метаданных) используется оператор ALTER PROCEDURE. Его синтаксис представлен на листинге 2.5:

Листинг 1.3 – Синтаксис оператора ALTER PROCEDURE

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

Пример изменения процедуры

```
ALTER PROCEDURE EXAMPLE_PROC(S CHAR(100))
   LANGUAGE JAVA
   EXTERNAL NAME 'esp.ExampleClass.exampleMethod';
   При перерегистрации внешних процедур возможно изменение состава и
   типов данных входных и выходных параметров, а также выражения,
   содержащего уникальное имя внешней процедуры.
```

В «Ред База Данных» 2.5 также есть возможность зарегистрировать новую внешнюю процедуру, а если такая процедура уже существует, то перерегистрировать ее. Для этого можно использовать оператор CREATE OR ALTER PROCEDURE или RECREATE PROCEDURE:

Листинг 1.4 – Синтаксис оператора 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';
```

Листинг 1.5 - Синтаксис оператора RECREATE PROCEDURE:

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

Оператор ALTER EXTERNAL FUNCTION позволяет изменить имя внешней функции:

Листинг 1.6 – Синтаксис оператора ALTER EXTERNAL FUNCTION:

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

Здесь NEW_UNIQUE_FUNCTION_IDENTIFIER – новое уникальное имя внешней функции.

1.3 Удаление внешних процедур и функций

Удаление внешней процедуры осуществляется оператором DROP PROCEDURE.

Листинг 1.7 Синтаксис оператора DROP PROCEDURE:

```
DROP PROCEDURE <PROCEDURE_NAME>;
```

Пример:
```
DROP PROCEDURE EXAMPLE_PROC;
```

Внимание! В отличие от обычных хранимых процедур, сервер не может проконтролировать наличие вызовов удаляемой процедуры из других внешних процедур или функций. Поэтому удаление такой процедуры пройдет без ошибок, однако при последующем вызове внешней процедуры будет сгенерированно исключение.

Удаление внешней функции осуществляется оператором DROP EXTERNAL FUNCTION. Его синтаксис:

Листинг 1.8 Синтаксис оператора DROP EXTERNAL FUNCTION

```
DROP EXTERNAL FUNCTION <FUNCTION_NAME>;
```

Пример:
```
DROP EXTERNAL FUNCTION EXAMPLE_FUN;
```

1.4 Вызов внешних процедур и функций на Java

Вызов внешних процедур и функций аналогичен вызову обычных хранимых процедур (SP) и функций, определенных пользователем (UDF). Например, вызов внешней процедуры можно осуществить оператором EXECUTE PROCEDURE. При этом внешняя процедура всегда будет выполняться с правами вызывающего её пользователя.

2 Правила написания внешних хранимых процедур

2.1 Соответствие типов данных SQL и Java

При написании внешних процедур и функций, следует пользоваться представленной ниже таблицей соответствия типов данных SQL, использующихся в «Ред База Данных», и языка программирования Java.

Таблица 1 - Соответствие типов SQL и Java

Тип SQL	Типы Java	
	Примитивный тип	Объектный тип
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 Правила написания тела внешних процедур и функций на Java

1. В качестве тела ESP, написанной на языке Java, может использоваться любой метод, объявленный как public static, и принадлежащий к public-классу (не внутреннему inner-классу). При написании тела внешних процедур на Java, допускается использование в качестве выходных параметров только типы данных java.sql.ResultSet и void. В качестве входных параметров возможно использование любых типов данных и их комбинаций, описанных в разделе «Соответствие типов данных SQL и Java».

2. ResultSet может быть получен как результат внутреннего запроса, запроса к внешней базе данных или другим способом. Например, ResultSet может быть вычислен при помощи какого-либо алгоритма.

3. При написании внешних функций на Java методы могут возвращать результат любого допустимого типа, а также содержать до 10 входных параметров.

Листинг 2.1 - Пример описания тела внешних процедур и функций на Java

```
package example;
  public class ExampleClass {
    //ESP, возвращающая void
    public static void exampleProc1(String s) {
      //операторы на Java
    }
    //ESP, возвращающая ResultSet
    public static java.sql.ResultSet exampleProc2() {
      //операторы на Java
    }
    //EUDF, возвращающая String
```

```
    public static String exampleFun1(int i) {
      //операторы на Java
    }
}
```

Перегрузка методов разрешена, но её следует избегать, потому что, например, методы с параметрами int и Integer различаются с точки зрения Java, но не с точки зрения внешних процедур и функций «Ред База Данных».

После написания классов, содержащих методы, которые будут использоваться в качестве тела внешних процедур и функций, необходимо создать jar-файл. Сделать это можно, используя утилиту jar, входящую в JDK, или средствами среды разработки java-программ (например, IntelliJ IDEA, Eclipse и др.).

Полученный jar-файл, необходимо скопировать в каталог java_lib, находящийся в папке, где установлена «Ред База Данных» или в файле jvm.conf в параметре -cp указать путь к нему.

2.3 Доступ к контексту вызова из метода на Java

В соответствии со стандартом SQLJ, в «Ред База Данных» 2.5 имеется возможность получения доступа к контексту выполнения, откуда была вызвана внешняя процедура или функция. Для этого необходимо использовать URL вида "jdbc:default:connection:", например:

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

В этом случае все вызовы connection.commit() и connection.rollback() будут проигнорированы.

Кроме того, в дополнении к стандарту SQLJ, в «Ред База Данных» можно получить доступ из java-метода к базе данных с отдельной транзакцией. Для этого используется URL вида "jdbc:new:connection:", например:

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

Это позволит, например, выполнить DDL или другую операцию, требующую автономности для выполнения.

2.4 Поддержка триггеров

Объявить триггер, написанный на языке Java, нельзя. Но можно вызвать внешнюю процедуру или функцию на Java из триггера, написанного на PSQL. При этом с помощью методов объекта класса org.firebirdsql.javaudf.Trigger из внешней процедуры или функции можно получить доступ к контексту триггера:

- получение имени таблицы, для которой вызван триггер (метод getTable());
- получение операции, вызвавшей триггер (метод getTriggerAction()), возможные значения:
  - вставка записи (возвращаемое значение 1);
  - изменение записи (возвращаемое значение 2);
  - удаление записи (возвращаемое значение 3);
  - соединение с БД (возвращаемое значение 4);
  - отсоединение от БД (возвращаемое значение 5);
  - старт транзакции (возвращаемое значение 6);
  - commit транзакции (возвращаемое значение 7);
  - rollback транзакции (возвращаемое значение 8);
- получение значения "new.<name>" в виде объекта (метод getObject_New(java.lang.String string), где string – имя поля таблицы);

- получение значения "old.<name>" в виде объекта (метод getObject_Old(java.lang.String string), где string – имя поля таблицы);
- установка значения "new.<name>" в виде объекта (метод setNewValue(java.lang.String string, java.lang.Object object), где string – имя поля таблицы, a object – новое значение этого поля).

Внимание! При использовании триггеров БД на событие connect из ESP или EUDF нельзя получить текущий attachment.

Пример получения из внешней функции, вызванной из триггера, имени таблицы, для которой вызван триггер

Листинг 2.2 - Пример получения контекста вызова из триггера

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

3 Примеры внешних процедур и функций

3.1 Вычисление факториала числа

В листинге 4.1 приведено описание EUDF, написанной на Java, которая производит рекурсивное вычисление факториала числа:

Листинг 3.1 - Пример вычисления факториала с помощью Java EUDF

```
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);
  }
  }
```

В листинге 4.2 приведен код, необходимый для регистрации внешней функции вычисления факториала.

Листинг 3.2 - Регистрация в базе данных функции из листинга 4.1

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

Результатом выполнения запроса

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

будет число 120.

3.2 Пример работы с файловой системой

В следующем листинге приведено описание внешней процедуры, написанной на Java, иллюстрирующей пример работы с файловой системой:

Листинг 3.3 - Работа с файлами с помощью ESP

```
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();
    }
  }
```

В ESP передаётся параметр строкового типа, задающий путь к файлу. Внутри процедуры осуществляется считывание этого файла в массив типа byte[], с последующей вставкой в таблицу test_table, содержащую поле типа BLOB.

Скрипты создания таблицы TEST_TABLE и регистрации ESP для примера из листинга 4.3приведены ниже:

Листинг 3.4 - Регистрация в базе данных метаданных для процедуры из листинга 4.3

```
CREATE TABLE TEST_TABLE(B BLOB);
  CREATE OR ALTER PROCEDURE TEST(
    S CHAR(100)
  )
  LANGUAGE JAVA
  EXTERNAL NAME 'esp.TestESP.loadFile';
```

Пример SQL-запроса выполнения ESP:

```
EXECUTE PROCEDURE TEST('c:\image.jpg')
```

3.3 Пример внешней процедуры, возвращающей набор данных

В следующем листинге приведено описание внешней процедуры, написанной на Java, возвращающей набор данных:

Листинг 3.5 - Работа с набором данных

```
public static ResultSet testRS() throws SQLException {
  Connection con =
  DriverManager.getConnection("jdbc:new:connection:");
  PreparedStatement pstmt = con.prepareStatement("SELECT * FROM
  test_table");
  return pstmt.executeQuery();
  }
```

Описанная в примере внешняя процедура производит выборку данных из таблицы test_table.

Скрипты создания таблицы TEST_TABLE и регистрации внешней процедуры приведены ниже:

Листинг 3.6 - Регистрация в базе данных метаданных для процедуры из листинга 4.5

```
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';
```

Просмотреть результат работы внешней процедуры testRS можно выполнив запрос

```
SELECT * FROM TEST;
```

3.4 Пример внешней процедуры, осуществляющей работу с другой базой данных

С помощью внешних процедур возможны подключения к другим базам данных посредством JDBC. Пример тела ESP, осуществляющей подключение к другой БД, делающей в ней выборку данных из таблицы test_table и вставляющей выбранные данные в таблицу test_table базы данных, из которой была вызвана внешняя процедура, приведен ниже:

Листинг 3.7 - Работа с несколькими базами данных из ESP

```
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();
   }
   }
```

Скрипты создания таблицы test_table и регистрации внешней процедуры приведены ниже:

Листинг 3.8 - Регистрация в базе данных метаданных для процедуры из листинга 4.7

```
CREATE TABLE test_table(
     F_VARCHAR VARCHAR(50)
   )

CREATE OR ALTER PROCEDURE TEST
   LANGUAGE JAVA
   EXTERNAL NAME 'esp.TestESP.interConnect';
```

Для вызова внешней процедуры TEST следует использовать оператор EXECUTE PROCEDURE:

```
EXECUTE PROCEDURE TEST;
```