# DERBY系列8-DERBY服务端编码



# 介绍

本章介绍如何使用derby进行编码，为derby 写入方法以及触发方法

# 网络连接

许多数据库会通过规则来进行分享代码，使代码变得可复用，但是在编码的过程中需要注意避免程序封闭和死锁并确保任何操作都能更新到原有的代码中

为了使用相同的语句，规则下的代码必须执行同一个连接和SQL语句。

在使用JDBC连接中，需要起到可重复利用的效果

|  |
| --- |
| Connection conn = DriverManager.getConnection("jdbc:default:connection"); |

在连接的URL代码中，不支持带用户和密码的连接方式，在derby中不符合java.sql.DriverManager.getConnection()方法中的规则，在使用网络连接时存在的以下规则限制

1. 只可以用存储方法去提交回滚
2. 不能改变自动提交连接属性
3. 不能将属性定义为ROUTINE，因为他是关键字

# 本地连接

在derby中，也提供了本地连接的方式，可以通过存储过程的方式去调用

|  |
| --- |
| CALL MYPROC() |

你可以在不进行提交的情况下进行回滚

# SQL与JDBC的规则

在JDBC下，需要对SQL语句进行抓获异常操作，保证语句能够在JDBC下进行异常信息处理并提出错误异常

|  |
| --- |
| try {  preparedStatement.execute();  } catch (SQLException se ) {  String SQLState = se.getSQLState();  if ( SQLState.equals( "23505" ) ) {  correctDuplicateKey();  } else if ( SQLState.equals( "22003" ) ) {  correctArithmeticOverflow();  } else {  throw se;  }  } |

当出现SQL异常信息时，他的状态会从38000开始提示，在SQL状态范围是38001-38999

(当出现SQL99时是可接受范围的，更多的异常状态吗详细信息会在以后介绍)

# 触发动作方法

下面是derby 触发例子

|  |
| --- |
| CREATE TRIGGER . . .  DELETE FROM FlightAvailability  WHERE flight\_id IN (SELECT flight\_id FROM FlightAvailability  WHERE YEAR(flight\_date) < 2005); |

再触发动作方法中，存在着一些限制  
 1.他不能包含动态参数

2.不能再已经声明触发的表中进行关联

下面以提供了详细的触发例子

|  |
| --- |
| CREATE TRIGGER trig1  AFTER UPDATE ON flights  REFERENCING OLD AS UPDATEDROW  FOR EACH ROW  INSERT INTO flights\_history  VALUES (UPDATEDROW.FLIGHT\_ID, UPDATEDROW.SEGMENT\_NUMBER,  UPDATEDROW.ORIG\_AIRPORT, UPDATEDROW.DEPART\_TIME,  UPDATED ROW.DEST\_AIRPORT, UPDATEDROW.ARRIVE\_TIME,  UPDATEDROW.MEAL, UPDATEDROW.FLYING\_TIME, UPDATEDROW.MILES,  UPDATEDROW.AIRCRAFT,'inserted from trig1'); |

这个触发方法是更新FlightAvailability表并向flights\_history插入

flight\_id的数据

|  |
| --- |
| CREATE TRIGGER FLIGHTSUPDATE  AFTER UPDATE ON flights  REFERENCING OLD AS OLD NEW AS NEW  FOR EACH ROW  WHEN (OLD.FLIGHT\_ID <> NEW.FLIGHT\_ID)  UPDATE FlightAvailability  SET FLIGHT\_ID = NEW.FLIGHT\_ID  WHERE FLIGHT\_ID = OLD.FLIGHT\_ID; |

# Derby函数

Derby函数通过返回JDBC中的RESULTSET实现，通过以下方法去实现结果集

1. NEXT()
2. CLOSE()
3. WASNULL()
4. GETXXX();

下面通过静态方法来实现函数调用

|  |
| --- |
| public static ResultSet read() {...} |

通过静态公有方法去规范derby函数名字

|  |
| --- |
| CREATE FUNCTION externalEmployees  ()  RETURNS TABLE  (  employeeId INT,  lastName VARCHAR( 50 ),  firstName VARCHAR( 50 ),  birthday DATE  )  LANGUAGE JAVA  PARAMETER STYLE DERBY\_JDBC\_RESULT\_SET  READS SQL DATA  EXTERNAL NAME 'com.example.hrSchema.EmployeeTable.read' |

调用一个表函数，封装在一个表的构造函数中，通过集合进行查询

|  |
| --- |
| INSERT INTO employees  SELECT s.\*  FROM TABLE (externalEmployees() ) s; |

在之后，提供过GETXXX()类似的方法去调用每个derby函数

在声明DERBY SQL方式中有以下GETXXX()方法

|  |  |
| --- | --- |
| 创建函数声明的方式 | 通过GETXXX()方法去调用 |
| BIGHT | GETLONG() |
| BLOB | GETBLOB |
| CHAR | GETSTRING() |
| CHAR FOR BIT DATA | GETBYTES() |
| CLOB | GETCLOB() |
| DATE | GETDATE() |
| DECIMAL | GETBIGDECIMAL() |
| DOUBLE | GETDOUBLE() |
| DOUBLE PRECISION | GETDOUBLE() |
| FLOAT | GETDOUBLE() |
| INTEGER | GETINT() |
| LONG VARCHAR | GETSTRING() |
| LONG VARCHAR FOR BIT DATA | GETBYTES() |
| NUMERIC | GETBIGDECIMAL() |
| REAL | GETFLOAT() |
| SMALLINT | GETSHORT() |
| TIME | GETTIME() |
| TIMESTAMP | GETTIMESTAMP() |
| VARCHAR | GETSTRING() |
| VARCHAR FOR BIT DATA | GETBYTES() |
| XML | 不支持 |

下面是derby表中函数的例子

|  |
| --- |
| package com.example.hrSchema;  import java.sql.\*;  /\*\*  \* Sample Table Function for reading the employee table in an  \* external database.  \*/  public class EmployeeTable  {  public static ResultSet read()  throws SQLException  {  Connection conn = getConnection();  PreparedStatement ps = conn.prepareStatement(  "select \* from hrSchema.EmployeeTable" );  return ps.executeQuery();  }  protected static Connection getConnection()  throws SQLException  {  String EXTERNAL\_DRIVER = "com.mysql.jdbc.Driver";  try {  Class.forName( EXTERNAL\_DRIVER );  }  catch (ClassNotFoundException e) {  throw new SQLException( "Could not find class "  + EXTERNAL\_DRIVER );  }  Connection conn = DriverManager.getConnection(  "jdbc:mysql://localhost/hr?user=root&password=mysql-passwd"  );  return conn;  }  } |