JDBC - UPDATING A RESULT SET EXAMPLE

Following is the example, which makes use of the **ResultSet.CONCUR_UPDATABLE** and **ResultSet.TYPE_SCROLL_INSENSITIVE** described in the Result Set tutorial. This example would explain INSERT, UPDATE and DELETE operation on a table.

It should be noted that tables you are working on should have Primary Key set properly.

This sample code has been written based on the environment and database setup done in the previous chapters.

Copy and past the following example in JDBCExample.java, compile and run as follows −

```
//STEP 1. Import required packages
import java.sql.*;

public class JDBCExample {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost/EMP";

   //  Database credentials
   static final String USER = "username";
   static final String PASS = "password";

 public static void main(String[] args) {
   Connection conn = null;
   try{
      //STEP 2: Register JDBC driver
      Class.forName("com.mysql.jdbc.Driver");

      //STEP 3: Open a connection
      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL,USER,PASS);

      //STEP 4: Execute a query to create statment with
      // required arguments for RS example.
      System.out.println("Creating statement...");
      Statement stmt = conn.createStatement(
                           ResultSet.TYPE_SCROLL_INSENSITIVE,
                           ResultSet.CONCUR_UPDATABLE);
     //STEP 5: Execute a query
      String sql = "SELECT id, first, last, age FROM Employees";
      ResultSet rs = stmt.executeQuery(sql);

      System.out.println("List result set for reference....");
      printRs(rs);

      //STEP 6: Loop through result set and add 5 in age
      //Move to BFR postion so while-loop works properly
      rs.beforeFirst();
      //STEP 7: Extract data from result set
      while(rs.next()){
         //Retrieve by column name
         int newAge = rs.getInt("age") + 5;
         rs.updateDouble( "age", newAge );
         rs.updateRow();
      }
      System.out.println("List result set showing new ages...");
      printRs(rs);
      // Insert a record into the table.
      //Move to insert row and add column data with updateXXX()
      System.out.println("Inserting a new record...");
      rs.moveToInsertRow();
      rs.updateInt("id",104);
```

```
      rs.updateString("first","John");
      rs.updateString("last","Paul");
      rs.updateInt("age",40);
      //Commit row
      rs.insertRow();

      System.out.println("List result set showing new set...");
      printRs(rs);

      // Delete second record from the table.
      // Set position to second record first
      rs.absolute( 2 );
      System.out.println("List the record before deleting...");
      //Retrieve by column name
      int id  = rs.getInt("id");
      int age = rs.getInt("age");
      String first = rs.getString("first");
      String last = rs.getString("last");

      //Display values
      System.out.print("ID: " + id);
      System.out.print(", Age: " + age);
      System.out.print(", First: " + first);
      System.out.println(", Last: " + last);

     //Delete row
      rs.deleteRow();
      System.out.println("List result set after \
                                deleting one records...");
      printRs(rs);

      //STEP 8: Clean-up environment
      rs.close();
      stmt.close();
      conn.close();
   }catch(SQLException se){
      //Handle errors for JDBC
      se.printStackTrace();
   }catch(Exception e){
      //Handle errors for Class.forName
      e.printStackTrace();
   }finally{
      //finally block used to close resources
      try{
         if(conn!=null)
            conn.close();
      }catch(SQLException se){
         se.printStackTrace();
      }//end finally try
   }//end try
   System.out.println("Goodbye!");
}//end main

   public static void printRs(ResultSet rs) throws SQLException{
      //Ensure we start with first row
      rs.beforeFirst();
      while(rs.next()){
         //Retrieve by column name
         int id  = rs.getInt("id");
         int age = rs.getInt("age");
         String first = rs.getString("first");
         String last = rs.getString("last");

         //Display values
         System.out.print("ID: " + id);
         System.out.print(", Age: " + age);
         System.out.print(", First: " + first);
         System.out.println(", Last: " + last);
     }
```

```
      System.out.println();
   }//end printRs()
}//end JDBCExample
```

Now let us compile the above example as follows −

```
C:\>javac JDBCExample.java
C:\>
```

When you run **JDBCExample**, it produces the following result −

```
C:\>java JDBCExample
Connecting to database...
Creating statement...
List result set for reference....
ID: 100, Age: 33, First: Zara, Last: Ali
ID: 101, Age: 40, First: Mahnaz, Last: Fatma
ID: 102, Age: 50, First: Zaid, Last: Khan
ID: 103, Age: 45, First: Sumit, Last: Mittal

List result set showing new ages...
ID: 100, Age: 38, First: Zara, Last: Ali
ID: 101, Age: 45, First: Mahnaz, Last: Fatma
ID: 102, Age: 55, First: Zaid, Last: Khan
ID: 103, Age: 50, First: Sumit, Last: Mittal

Inserting a new record...
List result set showing new set...
ID: 100, Age: 38, First: Zara, Last: Ali
ID: 101, Age: 45, First: Mahnaz, Last: Fatma
ID: 102, Age: 55, First: Zaid, Last: Khan
ID: 103, Age: 50, First: Sumit, Last: Mittal
ID: 104, Age: 40, First: John, Last: Paul

List the record before deleting...
ID: 101, Age: 45, First: Mahnaz, Last: Fatma
List result set after deleting one records...
ID: 100, Age: 38, First: Zara, Last: Ali
ID: 102, Age: 55, First: Zaid, Last: Khan
ID: 103, Age: 50, First: Sumit, Last: Mittal
ID: 104, Age: 40, First: John, Last: Paul

Goodbye!
C:\>
```