Использование Blob в интерфейсе SQLData для объектов Oracle

Вадим Земляной

Несколько странный способ, который описан в документации к Oracle, относительно работы с типами данных LOB, подтолкнул меня найти другой путь, который позволяет более красиво использовать Blob с JDBC.

Для того, чтобы записать содержимое в поле Blob, вам необходимо сначала получить ссылку на это поле — локатор.

В документации к Oracle для этого сначала вызывают инструкцию `SELECT`.

```
stmt.execute ("insert into my_blob_table
               values ('row1', empty_blob()");
BLOB blob;
   String cmd = "SELECT * FROM my_blob_table WHERE X='row1'";
   ResultSet rest = stmt.executeQuery(cmd);
   BLOB blob = ((OracleResultSet)rset).getBLOB(2);
   OutputStream outstream = blob.getBinaryOutputStream();
   outstream.write(….
```

Это не очень логично, что для того что бы вставить что-то в таблицу, нужно сначала из неё что то взять. Возможно инженеры из Oracle так шутят по поводу третьего закона Ньютона. Богам тоже нужен отдых. Но какая бы на то не была причина, этот способ не очень подходит для использования в объектах, реализующих интерфейс SQLData и содержащих BLOB. Обычно, когда Вы пишете такие объекты, Вы стремитесь к тому, что бы они были максимально независимы, и привязывать их к какой-либо таблице крайне нежелательно. Решением здесь может быть использование временных LOB (Temporary LOB).

Так как все мы любим примеры и не любим тратить время на копание в документации, я приведу пример реализации этого способа:

```
public class SQLObject implements SQLData{
    private String sql_type;
    private oracle.sql.BLOB content;
    public Object obj;
    private OracleCallableStatement close_st;

  public SQLObject() {
  }

  public SQLObject(String sql_type, Object obj,Connection conn)
                                                  throws Exception {
    this.sql_type = sql_type;
    OracleCallableStatement open_st =
       (OracleCallableStatement) conn.prepareCall(
          "BEGIN
           DBMS_LOB.CREATETEMPORARY(:1,TRUE,DBMS_LOB.SESSION);
           DBMS_LOB.OPEN(:1,DBMS_LOB.LOB_READWRITE);
           END;");
    open_st.registerOutParameter(1,Types.BLOB);
    open_st.execute();
    close_st =
     (OracleCallableStatement)conn.prepareCall
               ("BEGIN DBMS_LOB.CLOSE(?); END;");
    content = open_st.getBLOB(1);
    java.io.OutputStream os = content.getBinaryOutputStream();
    java.io.ObjectOutputStream objs =
                           new java.io.ObjectOutputStream(os);
    objs.writeObject(obj);
    objs.flush();
    objs.close();
  }

  // define a get method to return the SQL type of the object
  public String getSQLTypeName() throws SQLException {
    return sql_type;
  }

  // define the required readSQL() method
  public void readSQL(SQLInput stream, String typeName)
                                       throws SQLException {
```

```
    sql_type = typeName;
    content = (BLOB)stream.readBlob();
    try{
      java.io.ObjectInputStream is =
        new java.io.ObjectInputStream(content.getBinaryStream());
      obj = is.readObject();
      is.close();
    }catch ( Exception e){
      throw new java.sql.SQLException(e.getMessage());
    }
  }

  // define the required writeSQL() method
  public void writeSQL(SQLOutput stream)  throws SQLException {
    try{
      stream.writeBlob(content);
    }catch ( Exception e){ }
    close_st.setBLOB(1, content);
    close_st.execute();
    close_st.close();
  }
  protected void  finalize() throws java.lang.Throwable{
    if ((close_st != null )&& !close_st.closed)
    try{
      close_st.setBLOB(1, content);
      close_st.execute();
      close_st.close();
    }catch ( Exception e){ }
    super.finalize();
  }
}
Класс для работы с таким объектом выглядит так :
public class OraObject {
  private PreparedStatement get_st;
  private PreparedStatement set_st;
  private final String get_str =
                    "SELECT cv FROM docs  WHERE ID=(0)" ;
  private final String set_str =
                    "INSERT into docs(cv,id) VALUES(?,0)" ;
  protected String url         =
```

```
                  "jdbc:oracle:thin:@your_host:1521:your_db" ;
protected String user        ="you" ;
protected String pass        ="your_password" ;


public OraObject() {
  Connection conn = null;
  try{
     Class.forName("oracle.jdbc.driver.OracleDriver") ;
     Driver driver = DriverManager.getDriver( url ) ;
     conn = DriverManager.getConnection(url, user, pass);
     Map map = ((OracleConnection)conn).getTypeMap();
     map.put("CONTENT", Class.forName("test.db.SQLObject"));
     set_st = conn.prepareStatement(set_str);
     SQLObject sql_cv =
         new SQLObject("CONTENT",new String("1"),conn);
     SQLObject sql_cv1 =
         new SQLObject("CONTENT",new String("2"),conn);
     SQLObject[]  sql_vts =  {sql_cv,sql_cv1};
     ArrayDescriptor arraydesc =
         ArrayDescriptor.createDescriptor("CONTENTS", conn);
     Array array = new ARRAY(arraydesc, conn, sql_vts);
     set_st.setArray(1,array);
     set_st.execute();
     set_st.close();
     get_st = conn.prepareStatement(get_str);
     ResultSet rs = get_st.executeQuery();
     rs.next();
     ARRAY ar = ((OracleResultSet)rs).getARRAY(1);
     Object[] cv = (Object[]) ar.getArray();
     for (int i=0; i<cv.length; i++) {
       SQLObject variant = (SQLObject) cv[i];
       System.out.println(variant.obj.toString());
     }
     rs.close();
     get_st.close();
  }catch (Exception e){
    e.printStackTrace() ;
  }finally{
     try {
```

```
          if (conn != null) conn.close();
       }catch (Exception e){}
       System.exit(0);
    }
  }
   public static void main(String[] args) {
    OraObject oraObject = new OraObject();
  }
}
```

Здесь объект используется в массиве, но так же можно и обычным способом, используя `setObject()`. Скрипт для этого примера такой:

```
CREATE OR REPLACE TYPE content AS OBJECT(content BLOB);
CREATE OR REPLACE TYPE contents AS TABLE OF content;

DROP TABLE  docs;
CREATE TABLE docs
  (id NUMBER(11),
  cv contents DEFAULT NULL )
NESTED TABLE cv STORE AS cvs_table
TABLESPACE users
  STORAGE (INITIAL    32K
        NEXT          32K
        MINEXTENTS    1
        MAXEXTENTS    999
        PCTINCREASE 0);

ALTER TABLE docs ADD CONSTRAINT pk_documents_id PRIMARY KEY (id);
```