THE MIGRATION OF DATA FROM A RELATIONAL DATABASE (RDB) TO AN OBJECT RELATIONAL (ORDB) DATABASE

MOHAMED BAHAJ & ALAE ELALAMI
Hassan I University, FST Settat LAB LITEN
Department of Mathematics and Computer Science

ABSTRACT

The present article deals with the passage of migrating data from a relational database to an object relational database, by developing methods of selection and insertion which is based on optimizing the extraction of information in a predefined data model, which deals with the transition from relational to object relational, which fits to any situation processing . The migration handles the majority of the object concept including inheritance.

The transition of migration is done in an automatic way, without the interference of the human factor, a prototype is already created that proves the effectiveness of this approach. Keywords: Data Migration, RDB, ORDB, Prototype, Automation

1. INTRODUCTION

The object relational model includes both the object model and the relational model [5], which offers several advantages, the use of different object mechanism, inheritance, overloading, encapsulation of data, new types [6], the elimination of joints with passing by reference, at the same time benefiting from the simplicity of the relational concept and ease of use.

Some developers use the object-relational mapping but do not exploit the power of relational object [7], due to the passage related to transform a class to a table, some Framework provides the possibility of persistence without a solid background in the DBMSs [1].

Approaches shows the transition from relational to XML, ways differ but causes still the same, is to receive the benefit that offers XML and perpetuate the database [2]. This article discusses the transition from relational to XML and XML to ORDB [3] [4] to perform the migration of data from RDB to ORDB.

The migration of a RDB to an ORDB it is done by two steps, the first step is the transition from the physical schema of a RDB to the physical schema of an ORDB, and the second step is the extraction of data from the RDB and his injection into the new ORDB.

The insertion of a relational database data to the new object-relational database [8] will be extracted in an automated manner without the interference of the human factor, following the same approach for the migration of physical model of relational data to the object relational model which will be in three stages. The first step in selecting a RDB data, the second step is a passage alternating with which there will be a choice between two methods, made to keep the data in memory or store them in an external file, and the third stage role, is to insert the data into the new ORDB.

2. THE DATA SELECTION

The migration of data from a RDB towards ORDB begins extracting metadata of the RDB, from which a set of processing is realized to conceive the NEW DATA MODEL NDM [1].
NDM is a kind of table that defines the set of classes taken from the RDB with the necessary data for the realization of migration, including the classification of each class (inheritance, aggregation, association).

$$NDM: = \{C \mid C: = (cn, degree, cls, A, contributor)\}$$

Cn: the class name, Degree: first degree (the tables that contain PK) | 2nd degree (the tables that contain FK without PK), Cls: aggregation, association, inheritance, simple class (the class that does not belong to the other classifications), Contributor=class list. A=attribute:=\{a \mid a := (an, t, tag, l, n, d)\} (An :name of the attribute, T:type of the attribute, Tag: primary key(PK) | foreign key(FK),L: length of the attribute, N:if the attribute takes the parameter null, D:the default value of the attribute).

Example of relational database:

FIG 1: The Tables Representing The Relational Database
Table 1: The Representation Of The NDM Obtained

<table>
<thead>
<tr>
<th>Cn</th>
<th>Degré</th>
<th>Classification</th>
<th>Attribut</th>
<th>Contributor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>An</td>
<td>Type</td>
</tr>
<tr>
<td>Person</td>
<td>1er</td>
<td>inherBy</td>
<td>Pno</td>
<td>Varchar</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pname</td>
<td>Varchar</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bdate</td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Adress</td>
<td>Varchar</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Dno</td>
<td>Int</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PnoSup</td>
<td>Varchar</td>
</tr>
<tr>
<td>Trainee</td>
<td>2eme</td>
<td>Inherts</td>
<td>Pno</td>
<td>Varchar</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Level</td>
<td>Varchar</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Type</td>
<td>Varchar</td>
</tr>
<tr>
<td>Employ</td>
<td>2eme</td>
<td>Inherts</td>
<td>Pno</td>
<td>Varchar</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Salary</td>
<td>Int</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Grade</td>
<td>Varchar</td>
</tr>
<tr>
<td>Works_on</td>
<td>2eme</td>
<td>Association</td>
<td>Prno</td>
<td>Int</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pno</td>
<td>Varchar</td>
</tr>
<tr>
<td>Dept</td>
<td>1er</td>
<td>Simple</td>
<td>Dno</td>
<td>Int</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Dname</td>
<td>Varchar</td>
</tr>
<tr>
<td>Proj</td>
<td>1er</td>
<td>Simple</td>
<td>Prno</td>
<td>Int</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Prname</td>
<td>Varchar</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Description</td>
<td>Varchar</td>
</tr>
<tr>
<td>Kids</td>
<td>1er</td>
<td>Aggregation</td>
<td>Kno</td>
<td>Int</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Kname</td>
<td>Varchar</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sex</td>
<td>Char</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pno</td>
<td>Varchar</td>
</tr>
</tbody>
</table>

Data migration of RDB towards ORDB continues with the recovery of data, a method will be created to extract data, the data will be stored in dynamic arrays.
3. CREATION METHOD:

Data is recovered from the class names that we will extract from the NDM with a select query, with the data subtracted from the RDB and the information stored in the NDM, the schema is obtained of the table.

Ai is a rectangular array of m*n numbers stored line by line. There are m rows and n numbers in each row.

The tables: \{A_i / 0 \leq i \leq l\} defined by, For all 0 \leq i \leq n, A_i :\{ a_{k,j} / 0 \leq k \leq n / 0 \leq j \leq m \}.

Table 2: Matrix Representation Of The Schema

<table>
<thead>
<tr>
<th>Ai / a_{k,j}</th>
<th>a_{k,j}</th>
<th>0</th>
<th>…</th>
<th>…</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_0</td>
<td>0</td>
<td>0</td>
<td>…</td>
<td>…</td>
<td>m</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>N</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

Syntax:

Extraction of all the data of a database with a selection request, based on the pattern Data Access Object DAO, which makes the link between the data access layer and the business layer of the application, which gives a fluidity to operate on the storage system, thus achieving a migration of a system to another.

The selection query in combination with the JAVA language is as follows:

```java
public String[][] selectAll(String tableName) {
    String req = "SELECT * FROM " + tableName;
    try {
        Statement sql = db.createStatement();
        ResultSet rs = sql.executeQuery(req);
        ResultSetMetaData rsm = rs.getMetaData();
        int columns = rsm.getColumnCount();
        String data[][];
        rs.last();
        int rows = rs.getRow() + 1;
        data = new String[rows][columns];
        for (int i = 1; i <= columns; i++) {
            data[0][i-1] = rsm.getColumnName(i);
        }
        int row = 1;
        rs.beforeFirst();
        while (rs.next()) {
            for (int i=1; i<=columns; i++) {
                data[row][i-1] = rs.getString(i);
            }
            row++;
        }
        return data;
    }
    catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
```
The selection result:

**Kids table:**

<table>
<thead>
<tr>
<th>kno</th>
<th>Kname</th>
<th>sex</th>
<th>pno</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>Badr</td>
<td>M</td>
<td>d543</td>
</tr>
<tr>
<td>23</td>
<td>Sarah</td>
<td>F</td>
<td>d543</td>
</tr>
<tr>
<td>21</td>
<td>Jeff</td>
<td>M</td>
<td>g234</td>
</tr>
</tbody>
</table>

**Proj table:**

<table>
<thead>
<tr>
<th>prno</th>
<th>Pname</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Payment Management</td>
<td>integration of a module in an erp open source</td>
</tr>
<tr>
<td>2</td>
<td>tramway casa</td>
<td>realization of management complete Tramway casa</td>
</tr>
</tbody>
</table>

**Employ table:**

<table>
<thead>
<tr>
<th>pno</th>
<th>Salary</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>d543</td>
<td>9000</td>
<td>Engineer</td>
</tr>
<tr>
<td>g234</td>
<td>12000</td>
<td>Director</td>
</tr>
<tr>
<td>f552</td>
<td>7000</td>
<td>Commercial</td>
</tr>
</tbody>
</table>

**Works_on table:**

<table>
<thead>
<tr>
<th>Pno</th>
<th>Prno</th>
</tr>
</thead>
<tbody>
<tr>
<td>d543</td>
<td>1</td>
</tr>
<tr>
<td>f552</td>
<td>2</td>
</tr>
<tr>
<td>e234</td>
<td>1</td>
</tr>
</tbody>
</table>

**Trainee table:**

<table>
<thead>
<tr>
<th>Pno</th>
<th>Levell</th>
<th>Typee</th>
</tr>
</thead>
<tbody>
<tr>
<td>e234</td>
<td>Master</td>
<td>Hiring</td>
</tr>
</tbody>
</table>

**Dept table:**

<table>
<thead>
<tr>
<th>Dno</th>
<th>Dname</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Computer</td>
</tr>
<tr>
<td>2</td>
<td>Commercial</td>
</tr>
<tr>
<td>3</td>
<td>after-sales service</td>
</tr>
</tbody>
</table>

**Person table:**

<table>
<thead>
<tr>
<th>Pno</th>
<th>pname</th>
<th>bdate</th>
<th>adress</th>
<th>dno</th>
</tr>
</thead>
<tbody>
<tr>
<td>d543</td>
<td>alae</td>
<td>15/03/1987</td>
<td>residence ibn sina appt 3</td>
<td>1</td>
</tr>
<tr>
<td>e234</td>
<td>fouad</td>
<td>03/01/1987</td>
<td>rayhan imm 4 appt 5</td>
<td>2</td>
</tr>
<tr>
<td>g234</td>
<td>azar</td>
<td>24/04/1984</td>
<td>lotissement 34 rue des far appt 6</td>
<td>1</td>
</tr>
<tr>
<td>f552</td>
<td>jean</td>
<td>28/05/1975</td>
<td>rue la fayette residence bmo imm majid appt 9</td>
<td>3</td>
</tr>
</tbody>
</table>
4. **PASSAGE BY ALTERNATIVE**

During the passage of the data of a RDB towards an ORDB, we are going to be confronted to the size of the database, for it we are going to develop two methods.

The first one when the RDB is not huge, otherwise when its size is not superior to the size of the quantity of memory needed of the integrated development environment IDE which it needs for its execution, since our prototype is created by an IDE based on JAVA, thus inherits the memory allocation parameters of a java application, which means the XMX and XMS, these two parameters can be set; in our case eclipse.ini; on condition that $XMX \geq XMS$ [10].

So we just need to disconnect from the relational database management system RDBMS and connect to the object relational database management system ORDBMS and make our integration process.

The second method when there is a huge database which means that the size of the database exceeds the amount of memory size of IDE, in this case we store what we got in a semi-structured way in eXtensible Markup Lanuage XML file.

The XML file is as follows:

```xml
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bdr>
  <name_of_relationnel_table1>
    <attribute_name_with_tag_pk_in_ndm id="value of the attribute">
      <attribute_name>value of the attribute</attribute_name>
      <attribute_name>value of the attribute</attribute_name>
      <attribute_name>value of the attribute</attribute_name>
    </attribute_name_with_tag_pk_in_ndm>
    <attribute_name_with_tag_pk_in_ndm id="value of the attribute">
      <attribute_name>value of the attribute</attribute_name>
      <attribute_name>value of the attribute</attribute_name>
      <attribute_name>value of the attribute</attribute_name>
    </attribute_name_with_tag_pk_in_ndm>
  </name_of_relationnel_table1>
  <name_of_relationnel_table2>
    <attribute_name_with_tag_pk_in_ndm id="value of the attribute">
      <attribute_name>value of the attribute</attribute_name>
      <attribute_name>value of the attribute</attribute_name>
      <attribute_name>value of the attribute</attribute_name>
    </attribute_name_with_tag_pk_in_ndm>
    <attribute_name_with_tag_pk_in_ndm id="value of the attribute">
      <attribute_name>value of the attribute</attribute_name>
      <attribute_name>value of the attribute</attribute_name>
      <attribute_name>value of the attribute</attribute_name>
    </attribute_name_with_tag_pk_in_ndm>
  </name_of_relationnel_table2>
</bdr>
```

5. **INSERTING DATA INTO THE ORDB**

The insertion of the data in an ORDBMS begins with the disconnection of the RDBMS, and according to the size of the DB a choice arises, if we are going to parse the XML file or to use the memory of our IDE.

The storage of the data towards an ORDB begins by storing our objects in relations (tables). Instances will be created with the SQL statement (insert):

```
Insert into <table> values (<<constructor>(<value>, <value>,...)));
```

A request of insertion will be created which will adapt itself to any situation of insertion.
That it is an inheritance or an aggregation in compliance with the standards of object relational databases [9].

The insert query is as follows:

```java
public void insertO(TableR table) throws SQLException {
    Statement sql=null;
    try {
        int type = ResultSet.TYPE_SCROLL_INSENSITIVE;
        int mode = ResultSet.CONCUR_UPDATABLE;
        sql=db.createStatement(type,mode);
    }
    catch (Exception e) {
        e.printStackTrace();
        //return -1;
    }
    for (String v:table.getValues()) {
        StringBuffer req = new StringBuffer("INSERT INTO 
            .append(table.getNomTable()).append(" VALUES(");
        req.append(v).append(";
            System.out.println(req);
        sql.executeUpdate(req.toString());
    }
}
```

The TableR object is as follows:

```java
public class TableR{
    public TableR() {
        // TODO Auto-generated constructor stub
    }
    public TableR(String nomTable, ArrayList<String> values) {
        super();
        this.nomTable = nomTable;
        this.values = values;
    }
    String nomTable;
    public String getNomTable() {
        return nomTable;
    }
    public void setNomTable(String nomTable) {
        this.nomTable = nomTable;
    }
    public ArrayList<String> getValues() {
        return values;
    }
    public void setValues(ArrayList<String> values) {
        this.values = values;
    }
    ArrayList<String> values;
```
6. CONCLUSION

This paper presents an approach on the migration of data of a RDB towards an ORDB, based on a migration of the relational schema model to the object relational schema model using the metadata capture and treatment of different characteristic of the object including inheritance.

The approach begins with the extraction of data from the RDB, then a passage by choice to find the method the most appropriate according to the size of the RDB, the transition may be following a transformation of the relational towards the XML to use the disconnect mode or directly from relational to object relational limited just to the memory of the IDE, and in the end the insertion of the data in the ORDMS respecting the physical model of the ORDB, everything is done in an automatic way without the interference of the human factor knowing that no approach has provide an automatic data migration from RDB to ORDB.

This migration is made with a database normalize to guarantee the exploitation of the principle object relational, a prototype was realized which proves the efficiency of this approach.

REFERENCE


