Journal of Theoretical and Applied Information Technology
20th December 2013. Vol. 58 No.2
© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

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

MOHAMED BAHAJ & ALAE ELALAMI
Hassan 1 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 [].

Journal of Theoretical and Applied Information Technology
20th December 2013. Vol. 58 No.2
© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

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 | 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 | 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:

kids

kno	kname	sexe	pno
34	badr	m	d543
23	sarah	f	d543
21	jeff	m	g234

proj

prno	pname	description
1	Payment Management	integration of a module in an erp open source
2	tramway casa	realization of management complette Tramway casa

works_on

pno	prno
d543	1
f552	2
e234	1

employ

pno	salary	grade
d543	9000	engineer
g234	12000	director
f552	7000	commercial

trainee

pno	level	type
e234	master	hiring

dept

dno	dname
1	computer
2	commercial
3	after-sales service

person

pno	pname	bdate	adress	dno	pnosup
d543	alae	15/03/1987	residence ibn sina appt 3	1	g234
e234	fouad	03/01/1987	rayhan imm 4 appt 5	2	d543
g234	azar	24/04/1984	lotissemnt 34 rue des far appt 6	1	null
f552	jean	28/05/1975	rue la fayette residence bmo imm majid appt 9	3	d543

FIG 1: The Tables Representing The Relational Database

NDM obtained:

Table1: The Representation Of The NDM Obtained

Cn	**Degré**	**Classification**	**Attribut**						**Contributor**
			An	**Type**	**tag**	**l**	**N**	**D**	
Person	1er	inherBy	Pno	Varchar	PK		N		Kids
									Works_on
									Trainee
									Employ
			Pname	Varchar			N		
			Bdate	Date			N		
			Adress	Varchar		255	N		
			Dno	Int	FK		N		Dept
			PnoSup	Varchar	FK		Y		Person
Trainee	2eme	Inherts	Pno	Varchar	FK		N		Person
			Level	Varchar			N		
			Type	Varchar			N		
Employ	2eme	Inherts	Pno	Varchar	FK		N		Person
			Salary	Int			Y		
			Grade	Varchar			N		
Works_on	2eme	Association	Prno	Int	FK		N		Proj
			Pno	Varchar	FK		N		Person
Dept	1er	Simple	Dno	Int	PK		N		Person
			Dname	Varchar			N		
Proj	1er	Simple	Prno	Int	PK		N		Work_on
			Prname	Varchar			N		
			Description	Varchar		255	Y		
Kids	1er	Aggregation	Kno	Int	PK		N		
			Kname	Varchar			N		
			Sex	Char			N		
			Pno	Varchar	FK		n		Person

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 ≤i ≤ l} defined by, For all 0 0 ≤i≤ n, A_i :{ $a_{k,j}$ / 0 ≤k ≤ n / 0 ≤ j ≤ m} .

Table 2: Matrix Representation Of The Schema

A_i / $a_{k,j}$	$a_{k,j}$	0	…	…	m
A_0	0				
	…				
	…				
	N				

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:

```
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;
                }
                                                                }
```

Journal of Theoretical and Applied Information Technology
20th December 2013. Vol. 58 No.2
© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The selection result:
Kids table:

	kno	Kname	sex	pno
Kids	34	Badr	M	d543
	23	Sarah	F	d543
	21	Jeff	M	g234

Proj table :

	prno	Pname	description
Proj	1	Payment Management	integration of a module in an erp open source
	2	tramway casa	realization of management complete Tramway casa

Employ table:

	pno	Salary	Grade
Employ	d543	9000	Engineer
	g234	12000	Director
	f552	7000	Commercial

Works_on table:

	Pno	Prno
works_on	d543	1
	f552	2
	e234	1

Trainee table:

	Pno	Levell	Typee
Trainee	e234	Master	Hiring

Dept table:

	Dno	Dname
Dept	1	Computer
	2	Commercial
	3	after-sales service

Person table:

	Pno	pname	bdate	adress	dno
Person	d543	alae	15/03/1987	residence ibn sina appt 3	1
	e234	fouad	03/01/1987	rayhan imm 4 appt 5	2
	g234	azar	24/04/1984	lotissemnt 34 rue des far appt 6	1
	f552	jean	28/05/1975	rue la fayette residence bmo imm majid appt 9	3

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 ≥ 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 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_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_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_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_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:

```
	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:

```
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;
```

```
        /*
         * value must be formatted, eg:'v1','v2','v3'
         */
        public void addValue(String value){
                if(this.values==null)
                        this.values=new ArrayList<String>();
                this.values.add(value);
        }

}
```

On every table a check is made if there is a type date and it is formatted to be in accordance and compatible with oracle "dd / mm / yyyy".

The insertion begins following a process which respects the classification extracted from the NDM and which begins with the simple classes with or without aggregation, then the insertions concerning the inheritance will be realize, starting with the objects of the parent class, then it object that inherits from it with or without aggregation, and the final insertion is the associations, all the insertions have to respect the passage by reference which is extracted from the NDM (the tuple " contributor, tag = FK") to benefit from the strength of the object relational without passage by joints.

Exemple de requête d'insertion :

```
insert into employ values (
employ_type(
kids_t(
kids_type('34','badr','m','d543'),kids_type('23','sarah','f','d543')),
'd543','alae','15/01/1987','residence ibn sina imm d4 appt 3','1',
(select  REF (deptref) from dept deptref  where deptref.dno=1 ),
'g234','9000','engineer'))
```

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

[1] . Carlos Alberto Rombaldo Jr , Solange N. Alves Souza : O-ODM Framework for Object-Relational Databases. International Journal of Artificial Intelligence and Interactive Multimedia, Vol. 1, Nº 6.

[2]. Jinhyung Kim, Dongwon Jeong And Doo-Kwon Baik : A Translation Algorithm for Effective RDB-to-XML Schema Conversion Considering Referential Integrity Information . journal of information science and engineering 25, 137-166 (2009) .

[3]. Chitra Desai , Manisha Patil and Sahebrao Shinde :Representing object oriented database using XML-DTD. Chitra Desai *, Manisha Patil and Sahebrao Shinde.

[4]. F. Golobisky, A.Vecchietti, Mapping UML Class Diagrams into Object-Relational Schemas. Rosario, rgentina: Proceedings of ASSE, p 65-79, 2005

[5]. Stonebraker, Michael, Moore and Dorothy. Object-Relational DBMSs: The Next Great Wave (Morgan Kaufmann Series in Data Management Systems) ISBN: 1558603972.

[6]. J.W. Rahayu, E. Pardede and D. Taniar, On Using Collection for Aggregation and Association Relationships in XML Object-Relational Storage. *acm symposium on applied computing*, nicosia, cyprus, 2004.

[7]. M. Fotache, C. Strîmbei, "Object-Relational Databases: An Area with Some Theoretical Promises and Few Practical Achievements", Communications of the IBIMA, v. 9, ISSN 1943-7765, 2009.

[8]. Oracle Database 12c Documentation:http://www.oracle.com/technetwork/database/enterprise-edition/documentation/database-093888.html .

[9]. the API specification for version 6 of the Java™ Platform, Standard Edition: http://docs.oracle.com/javase/6/docs/api/ .

[10]. Allocating enough memory and solving OutOfMemoryErrors : http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.platform.doc.user%2Ftasks%2Frunning_eclipse.htm