Journal of Theoretical and Applied Information Technology
31st May 2016. Vol.87. No.3
© 2005 - 2016 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

MIGRATION OF A RELATIONAL DATABASE RDB TO AN OBJECT ORIENTED DATABASE OODB

[1]ALAE EL ALAMI, [2]MOHAMED BAHAJ,[3]ILIAS CHERTI
[1, 2,3]Hassan 1 University, FST Settat LAB LITEN
[1]elalamialae@gmail.com, [2]mohamedbahaj@gmail.com, [3]iliascherti5@gmail.com

ABSTRACT

The objective of this paper is to propose a solution to the migration of RDB to OODB which is based on metadata and the principle of semantic enrichment to extract different object principles including inheritance, aggregation, and composition. Then, to achieve the transformation of the physical schema and to establish the migration of data to the target database. This is done in an automatic way, a prototype has been implemented that takes a RDB and converts it to an OODB which is part of reverse engineering. Then, to realize the relational-object data mapping that respects the passage by reference and the different object principles.

Keywords: *RDB, ORDB, OODB, Semantic Enrichment, Schema Translation, Data Mapping, Metadata, DB4O*

1. INTRODUCTION

The relational model is a concept based on the principle of relation, all of these data is organized in a table, based on set theory which is known as a fundamental theory built around functions, relationships, integers, real numbers... The relational model is known for the simplicity of its design, ease of use, based on SQL which is a standardized language that follows the ISO/CEI 9075 :2011.

The relational model is limited as it uses a rigid structure, encounter much difficulty during its evolution, its simplicity also deprives him of several concepts, since it is based on set theory, and a set by definition refers to a collection objects, hence the birth of this data migration approach of a RDB to an OODB.

Several approaches have discussed the database migration problem, some are devoting to the migration of relational databases to the ORDB, an approach based on a data model called SOT Semi Object Type, which is based on the transformation of relational schema with the principle of reverse engineering, then the re-design of the SOT to an object-oriented schema through Frameworks, in the end achieving the migration of data[1].

An approach is based on the principle of modeling object using UML modeling language, which takes as starting element a class diagram, then the diagram extends stereotype for an advanced object-relational, the use of annotations helps developers to choose well which UDT must be compiled[2] [8].

Maatuk and Akhtar [3] discusses the problem of migration of a RDB to an ORDB, based on the RDB transformation to a Canonical Data Model, then complete its transformation into a physical schema based on a set of treatment.

Another approach makes the migration of RDB to ORDB in a Meta-model derived from the RDB,

then achieve the transformation to its target following a physical schema processing assembly and function, according to an intelligent design that supports different type of DBMS and in an automated manner [4].

A particular approach realize the migration of a RDB to an ORDB at the data mapping, based on a Meta-model and an object relational physical schema of the original RDB created before, data migration is done in an automatic manner which is based on intelligent detection methods of object concept [5].

A similar approach of data migration to the object-relational model which is based on a conceptual model of any type of language or method, based on a Meta-model that enriched the conceptual model with the necessary data and constraints, then proceed to the transformation of the Meta-model to the target physical schema [6].

Other concept of migration approach is used to show the transformation of a conceptual model based on a class diagram to an XML file based on

mathematical formulas to create graphs and XSD schemas, then define structure and end up by the validation and storage [7].

Xiuzhen Zhang [9] explain an approach of transformation of the RDB towards well-structured OODB schemas, which bases itself on the principle of keys, the inclusion dependence, and some constraint to establish the object oriented aspects as inheritance and association between the classes. The role of the approach is to eliminate data redundancy through a theorem on the multi-valued dependency MVD.

The approach of Andreas Behm [10] show the migration of the RDB towards the OODB bases itself on the definition of the user defined transformation rules, and generates the object-oriented physical schema and data migration that goes with it.

An approach uses the migration of RDB to OODB through a canonical model[13], which enriched the source relational database with object characteristics required, and cardinalities for key migration to establish the object schema physical target according to the ODMG standard[11] [12].

Certain approach shows the migration of RDB to OODB and promotes the ODBMS to the RDBMS by the fastness, direct manipulation of data persistence, increased security level in the data due to the three levels of access to data and their worn for instance variables and methods[14].

Our approach discusses the migration of relational databases to object-oriented databases, a prototype was created which proves the effectiveness of this approach, based on the principle of semantic enrichment which acts as the core of the application. the prototype extract and translate the physical schema of the relational database to an object-oriented physical schema and performs the data mapping of the RDB to an OODB, all in an automatic manner without the interference of the human factor.

2. SEMANTIC ENRICHMENT

The semantic enrichment is the first step of migration, which is an extended relational model, defined as a Meta-model that defines the different classes extracted from the RDB through metadata[15].

The Meta-model is defined as a collection of classes : = {C | C: = (Cn, Degree, Cls, A, Contributor)}

Cn =the name of the class.

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 that show the relation between classes.

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).

Journal of Theoretical and Applied Information Technology
31st May 2016. Vol.87. No.3
© 2005 - 2016 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Cn	Degre	Classification	Attribut
			`DatabaseMetaDatadmd = connection.getMetaData();`
			`ResultSet tables= dmd.getTables(catalog,schemaPattern,tableNamePattern,types);`
										Contributor

			tables	An	Type	Tag	l	N	D	
TABLE_NAME	Traitment	Traitment	Next()	COLUMN_NAME	DATA_TYPE	ResultSet clefs = dmd.getPrimaryKeys(catalog, schema, table) dmd.getExportedKeys(catalog, schema, table)	COLUMN_SIZE	IS_NULLABLE	COLUMN_DEF	Traitment

Figure 1: Structure of the Meta-model by metadata

The Meta-model creation process is carried out in two stages, the first stage is based on the extraction of information available outside these same data, information on the tables (table names) primary key, information on the columns of a table (name, type, size, defaults, and nullability attribute).

Cn	Degre	Classification	Attribut						Contributor
			An	Type	tag	l	N	D	
Person	1st	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	2nd	Inherts	Pno	Varchar	FK		N		Person
			Level	Varchar			N		
			Type	Varchar			N		
Employ	2nd	Inherts	Pno	Varchar	FK		N		Person
			Salary	Int			Y		
			Grade	Varchar			N		
Works_on	2nd	Association	Prno	Int	FK		N		Proj
			Pno	Varchar	FK		N		Person
Dept	1st	Simple	Dno	Int	PK		N		Person
			Dname	Varchar			N		
Proj	1st	Simple	Prno	Int	PK		N		Work_on
			Prname	Varchar			N		
			Description	Varchar		255	Y		
Kids	1st	Aggregation	Kno	Int	PK		N		
			Kname	Varchar			N		
			Sex	Char			N		
			Pno	Varchar	FK		N		Person

Figure 2: The Graphical Representation Of The Meta-Model

Using metadata to capture the structure and the information accessible through JDBC regarding the source RDB, the table below provides an overview of the main instructions to develop the Meta-model.

The second step is the detection of object principles from the relational database, our approach of database migration is done in an automatic way for every migration steps including detecting inheritance, also eliminates the generation of the parameter responsible for specifying the cardinality considered a redundant step due to the migration of the primary key in the transition from conceptual model to the physical model of the database. The issue that remains is whether there was a cardinality0 or 1, which results in the creation of at least one instance or no instance. The solution is just in the sub-parameter (n) of the parameter attribute of the Meta-model which specifies if the attribute can have a worthless value.

The degree of a class varies according to the identifier of the table if there is a primary key either not or if it is about a key composite. There are two values in the category degree, first degree for the classes which have a primary key and the second degree for the classes which not have a primary key or when it is a composite key.

The classification of classes is made dynamically according to a set of processing to perform and by the apprenticeship from the degree.

+For associations, the classification is made according to the degree, the absence of primary key and the interaction with two or more classes. For reflexive associations that interact in the same table, it is detected by the appearance of a foreign key that has no other table as a primary key resulting from a self-contribution.

+inheritance is detected according to a data dictionary that includes a set of names, each name indexes a set of synonym or list of matching words, which are dependent on the mother-daughter relationship according to standard naming rules, if we find a match we proceed with the verification of primary keys, if there's a match between the keys, which proves that there's an inheritance relationship daughter-mother, otherwise the treatment is continued (this check is used to address the problem of non-standard databases).

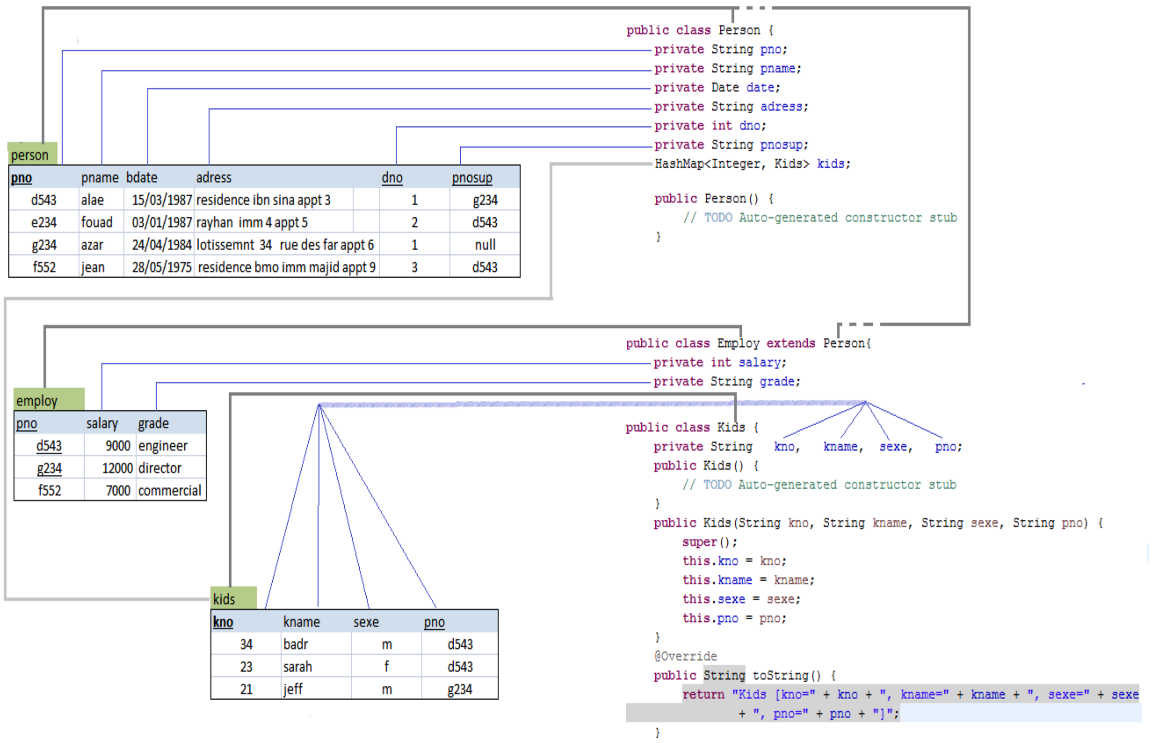
+Composition is detected when the class itself interacts with a single class, provided Cn.Classification≠association and there is no dependency between the two tables←(absence of RESTRICT or CASCADE).

+aggregation is detected when the class itself interacts with a single class, provided Cn.Classification≠Association and taking into account the dependency between the two tables←(existence of one of these keyword RESTRICT or CASCADE) in order to keep a history when deleting the object that comes into contact with it.

+Simple classes are the classes that remain without any of these classifications mentioned earlier and which are in the first degree.

3. SCHEMA TRANSLATION

The physical schema object-oriented database consists of a set of classes, class declares an attribute set that is the state of objects and their behavior, the object-oriented physical schema will be achieved in an automatic manner according to a set of processing based on the Meta-model.



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	residence bmo imm majid appt 9	3	d543

```
public class Person {
    private String pno;
    private String pname;
    private Date date;
    private String adress;
    private int dno;
    private String pnosup;
    HashMap<Integer, Kids> kids;

    public Person() {
        // TODO Auto-generated constructor stub
    }
```

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

```
public class Employ extends Person{
    private int salary;
    private String grade;
```

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

```
public class Kids {
    private String   kno,   kname,   sexe,   pno;
    public Kids() {
        // TODO Auto-generated constructor stub
    }
    public Kids(String kno, String kname, String sexe, String pno) {
        super();
        this.kno = kno;
        this.kname = kname;
        this.sexe = sexe;
        this.pno = pno;
    }
    @Override
    public String toString() {
        return "Kids [kno=" + kno + ", kname=" + kname + ", sexe=" + sexe
                + ", pno=" + pno + "]";
    }
```

Figure 3: The Approach Followed For The Creation Of The Physical Schema

The figure above shows how the relational tables are mapped to an object-oriented physical schema according to the Meta model with respect for the naming rules and characteristic of the principle(aggregation, composition, inheritance, polymorphism, encapsulation...).

The physical schema of the object-oriented database is realized from the meta-model, which begins by making the equivalence between types of extracts data from the RDB and types observed by the OO programming languages(eg VARCHAR→String).

Creating simple classes as Cn.classification=simple, begins by the declaration of variables with their scope then the two constructor(no parameters, with parameter) and methods that retrieves and sends the attribute values.

The creation of the classes association as Cn .classification=association, will be established of its attribute (if it exists) and both identifiers of the classes who enters in association with her with the principle of Hashtables, considered as a class that makes the referencing between two classes or more to eliminate joints problems. For reflexive associations is within the scope of the same class, the principle of the collections are used, the choice of collections changes depending on the parameter null of the parameter attribute.

The creation of the aggregation classes as cn.classification=aggregation, is achieved by the same way as the simple class, the creation of the aggregation classes whose cn.classification=aggregation, is achieved by the same way as the simple class, and it is referenced in the class entering in collaboration with it by adding a collection of the same object according to its nullability. The class entering in collaboration with the aggregation exchange skeleton with the appearance of a new object that will be in the constructor with parameter and the getters and setters of the class.

The creation of classes composition as Cn.classification=composition, requires an instance of the class that enters in collaboration with the composition, contain an instance of the class classified as composition, which render their instance physically related.

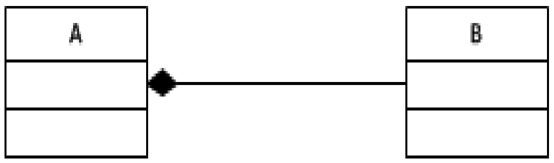


Figure 4: Graphic representation of the composition

```
Public class Compose {
                    protected Composite b;
                    public Compose(){
                              this.b = new Composite();
                              this.b.setCompose(this);
                    }
          }
Public class Composite {
                    Protected Compose a;
                    Public Compose getCompose() {
                              Return a;
                    }
                    Public void setCompose(Compose a) {
                              this.a = a;
                    }
          }
```
Figure 5: Implementation Of The Composition Concept

The creation of classes inheritance as Cn.classification=inherBY will be treated like other single classifications and the inheritance as Cn.classification=inherts Will inherit from classes which enters in collaboration with her.

Algorithm producing OODB schema :

```
Foreach class C ∈ meta-model do
if C.Classification ≠ 'composition' then
autoDeclarationClass();
autoGenerationAttribut();
autoGenerationConstructor();
autoGenerationConstructorAttribut();
autoGenerationGetters();
autoGenerationSetters();
autoGenerationToString();
```

```
end if
end for
#capitalFirstLetter(){
String className := C.Cn.replaceFirst(".",(table.charAt(0)+"").toUpperCase());
Return className;
}
#autoDeclarationClass(){
If C.Classification≠ ( 'inherts' && 'composition' ) then
    Print (" private class "+capitalFirstLetter(C.Cn)+" { ");
Else if C.Classification= 'inherts' then
    Print (" private class "+capitalFirstLetter(C.Cn)+" extends "+C.capitalFirstLetter (contributor)+" { ");
End if
}
#autoGenerationAttribut()
For C.Contributor.firstElement to C.Contributor.lastElement
    Foreach C1∈ meta-model do
        If C1.Contributor.element = 'agregation' then
            If C.Attribut.N = 'n' then

    print("privateHashMap<Integer, "+capitalFirstLetter (C1.Cn)+"> "+C1.Cn +" ;");
            else print("privateHashTable<"+capitalFirstLetter (C1.Cn)+ ",Integer) >"+C1.Cn +" ;");
            End if
        End if
    End for
//same treatment for the reflexive association
End for
For C.Attribut.firstElement to C.Attribut.lastElement
    If C.Attribut.type = 'Varchar ' then
        Print("private String "+C.Attribut.An+" ;");
    Else //same treatment for the other types
    End if
End for
#autoGenerationConstructorAttribut()
Print ("public "+capitalFirstLetter(C.Cn) +"(");
For C.Contributor.firstElement to C.Contributor.lastElement
    Foreach C1∈ meta-model do
        If C1.Contributor.element = 'agregation' then
            If C.Attribut.N = 'n' then
                print(" HashMap<Integer, "+capitalFirstLetter (C1.Cn)+"> "+C1.Cn +" ,");
            else print("privateHashTable<"+capitalFirstLetter (C1.Cn)+ ",Integer) >"+C1.Cn +" ,");
            End if
        End if
    End for
End for
For C.Attribut.firstElement to C.Attribut.lastElement
    If C.Attribut.type = 'Varchar ' then
        If C.Attribut.lastElelement = false
            Print("String "+C.Attribut.An+" ,");
        Else Print("String "+C.Attribut.An+" ){");
        End if
    Else //same treatment for the other types
    End if
End for
If C.Classification ≠ ('inherits' && 'composition') then
For C.Contributor.firstElement to C.Contributor.lastElement
    Foreach C1∈ meta-model do
```

```
                If C1.Contributor.element = 'agregation' then
                        print(" this."+C1.Cn+" = "+C1.Cn +" ;");
                End if
        End for
End for
For C.Attribut.firstElement to C.Attribut.lastElement
        If C.Attribut.type = 'Varchar ' then
                Print("this. "+C.Attribut.An+" = "+ C.Attribut.An);
        Else //same treatment for the other types
        End if
End for
Else If C.Classification = ( 'inherits' ) then

Print("super();");
For C.Contributor.firstElement to C.Contributor.lastElement
        Foreach C1∈ meta-model do
                If C1.Contributor.element = 'agregation' then
                        print(" this."+C1.Cn+" = "+C1.Cn +" ;");
                End if
        End for
End for
For C.Attribut.firstElement to C.Attribut.lastElement
        If C.Attribut.type = 'Varchar ' then
                Print("this. "+C.Attribut.An+" = "+ C.Attribut.An);

        Else //same treatment for the other types
        End if
End for
End if
Print("}");
………
```

4. DATA MAPPING

The part of data mapping from a RDB into an OODB, we opt for db4o which is a database open source embeddable object, which works on the operating system that supports Java or.NET language[16], which provides the ability to write the query in the programming language himself, queries in db4o is based on the Simple Object database Access SODA query API[17], which provides classes to perform complex queries on OODB.

To establish the migration of data we need two types of methods that retrieve data, a method for the complete recovery of a table, and another to make a personalized selection which aims to make the correspondence between the base class reference and DAO data Access Object that provides the ability to separate the object persistence and data access[18].

The custom select query is as follows:

```
public String[] select(String tableName, String key, String value) {
        String req = "SELECT * FROM " + tableName +
                " WHERE " + key + " = '" + value + "'";
        try {
                Statement sql = db.createStatement();
                ResultSet rs = sql.executeQuery(req);
                ResultSetMetaData rsm = rs.getMetaData();
                Int columns = rsm.getColumnCount();
                String data[];
                data = new String[columns];//Tuple
                if (rs.next()) {
                        for (int i=1; i<=columns; i++) {
                                data[i-1] = rs.getString(i);
                        }
                        Return data;
                }
                Else return null;
        }
        catch (Exception e) {
                e.printStackTrace();
                return null;
        }
}
```

Figure 6: Select query implementation

Data migration begins with the creation of the method that will be responsible for persistence, with which there will be no instances of the type of the class extracted from the Meta-model, then establish the storage of instances by using certain methods responsible for the storage DB4O according to the Object Container.

For simple classes and associations a simple method is required, for inheritance classes,

the method is applied on sub-classes, and the super-class's↔sub-class's dependence is managed through db4o which operates the physical schema of the object database generated by the constructor of the super-class. For the class aggregations, compositions, and reflexive associations, we call the responsible method for the specific selection to do the correspondence between objects via references and the class which enters into the contribution in the Meta-model will be instantiate first and then integrate into the instance of the class that collaborates with it.

Part of algorithm producing data migration from a RDB to an OODB :

```
Foreach class C ∈ meta-model do
String t[][] = db.selectAll(C.Cn);
If C.Classification ≠ ('inherits' && 'composition' && 'inherBy') then
print("public static void store"+capitalFirstLetter(C.Cn)+"(ObjectContainer db4o){");
for line=1 to t.length
For C.Contributor.firstElement to C.Contributor.lastElement
Foreach C1 ∈ meta-model do
If C1.Contributor.element = 'agregation' then
print(capitalFirstLetter(C1.Cn)+" "+C1.Cn +line+"=new HashMap<Integer, "+capitalFirstLetter(C1.Cn)+">(); ");
String tableAgregation[][] = db.selectAllByChamp(C1.Cn,C1.Attribut.tag(pk) , t[line][0]);
for(lineAgregation=1 to tableAgregation.length

        print(C1.Cn+lineAgregation+".put("+lineAgregation+", new "+capitalFirstLetter(C1.Cn))+"(");
for(column=0 to tableAgregation[line].length
if(c<tableAgregation[l].length-1) then
print("\""+tableAgregation [line][column]+"\" ,");
else
                print("\""+tableAgregation [l][c]+"\");");
End if
End for
End for
End if
End for
End for
End for

for(l=1 to l=t.length)
print (capitalFirstLetter(C.Cn)+" "+C.Cn+l+"=new "+ capitalFirstLetter(C.Cn)+"( ");
if(tableAgregation[l-1]=1)then
print("null, ");
else
        print(C1.Cn+l+", ");

End if
for(column=0 to column = t[l].length
if(column<t[l].length-1)then
            print(t[l][column]+" ,");
        Else print (t[l][ column]+");");
End if
print("db4o.store("+C.cn+l+");");
    End for
End for
End if
End for
```

The diagram below summarizes the whole approach of the migration of a RDB towards an OODB:

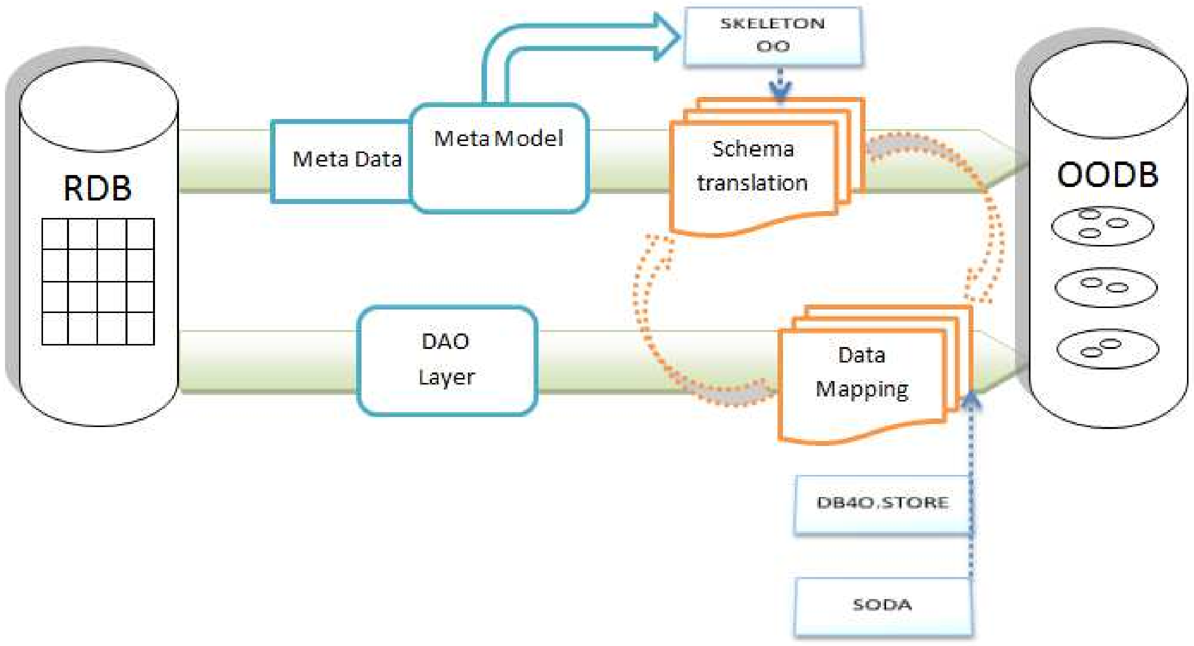


Figure 7: The Cycle Of Processing Of A RDB Towards An OODB

5. CONCLUSION

All in all, the article discusses the migration of a RDB to an OODB using metadata and meta-model that encapsulate all of the RDB and add the principles of object model through semantic enrichment, then establish the transformation of the relational physical schema to the object physical schema based on the Meta model and to realize the data mapping to the new physical schema, explaining the procedure to follow and proposing an algorithm to determine the selection.

Finally we realized a prototype which makes the migration in an automatic way without the interference of the human factor; the tests and the results prove the efficiency of our approach.

REFRENCES:

[1] A. Behm, A. Geppert, R. Dittrich, Algebraic Database Migration to Object Technology, *Proceeding ER'00 Proceedings of the 19th international conference on Conceptual modeling* ,Pages 440-453.

[2] E. Marcos, B. Vela, J.M. Cavero, A Methodological Approach for Object-Relational Database Design using UML, *Software and Systems Modeling Volume 2*, issue 1,pp 59-72.

[3] A. Maatuk, A. Ali, N. Rossiter, Converting Relational Databases into Object-relational Databases, *in JOT*, vol. 9, no. 2, pages 145-161, 2010.

[4] A. El Alami, M. Bahaj, The Road to a Full Migration of Relational Database (RDB) to Object Relational Database (ORDB): Semantic Enrichment, Target Schema, Data Mapping, *International Journal of Advanced Information Science and Technology (IJAIST)* Vol.30, No.30, October 2014 ISSN: 2319:2682

[5] M. bahaj, A. El Alami, The Migration Of Data From A Relationaldatabase (RDB) To An Object Relational (ORDB) Database, *Journal of Theoretical and Applied Information Technology,* 20th December 2013. Vol. 58 No.2 ISSN: 1992-8645.

[6] A. El Alami, M. Bahaj, The Migration of a Conceptual Object Model COM (Conceptual Data Model CDM, Unified Modeling Language UML class diagram ...) to the Object Relational Database ORDB, *MAGNT Research Report* (ISSN. 1444-8939) Vol.2 (4). PP: 318-327.

[7] M. Bahaj, N. Gherabi, Robust Representation for Conversion UML Class into XML Document using DOM, *International Journal of Computer Applications* (0975 – 8887) Volume 33– No.9, November 2011.

[8] T. Quatrani, Visual Modeling with Rational Rose 2000 and UML. Publisher: *Addison Wesley Second Edition,* October 19, 1999 ISBN: 0-201-69961-3, 288 pages.
[9] X. Zhang, Y. Zhang, J. Fong, X. Jia, Transforming RDB schema into well-structured OODB schema, *Information & Software Technology,* 41 275-281 (1999).
[10] A. Behm, A. Geppert, R. Klaus, On The Migration Of Relational Schemas And Data To Object-Oriented Database Systems, In: *Proceedings of the 5th International Conference on Re-Technologies for Information Systems*, pp. 13–33 (1997).
[11] A. Maatuk, M.A. Ali, N. Rossiter, Relational database migration: A perspective, *in DEXA*, vol. 5181, pp. 676–6832008.
[12] A. Maatuk, M.A. Ali, N. Rossiter, An integrated approach to relational database migration, *in IC-ICT2008*, 6 pp.
[13] A. Maatuk, M.A. Ali, N. Rossiter, Semantic Enrichment: The First Phase of Relational Database Migration, *In CIS2E '08*, 6pp, Bridgeport, USA, 2008.
[14] M. Alam, S.K. Wasan , Migration from relational database into object oriented database, *Journal of Computer Science*, vol. 2, issue. 10,pp 781-784.
[15] A. El Alami, M. Bahaj, Migration of the Relational Data Base (RDB) to the Object Relational Data Base (ORDB), *World Academy of Science, Engineering and Technology International Journal of Computer, Information Science and Engineering,* Vol:8 No:1, 2014.
[16] J. Paterson, S. Edlich, H. Hörning, and R. Hörning. *The Definitive Guide to db4o.* Apress, 2006.
[17] https://docs.oracle.com/cd/E12095_01/doc.10303/e12548/csoda.htm
[18] http://www.oracle.com/technetwork/java/dataaccessobject-138824.