Migration of relational databases RDB to database for objects db4o for schema and data.

Youness Khourdifi
Ph.D.: Department of Mathematics and Computer Sciences
Faculty of Science and Technology, University Hassan I
Settat, Morocco
E-mail: ykhourdifi@gmail.com

Mohamed Bahaj
Professor: Department of Mathematics and Computer Sciences
Faculty of Science and Technology, University Hassan I
Settat, Morocco
E-mail: mohamedbahaj@gmail.com

Abstract— In This paper, we present an approach that takes an existing relational database as input, obtains a copy of its metadata and the principle of semantic enrichment to extract different object principles including aggregation, inheritance, and composition. The first phase of the migration process is the CDM is generating an automatic, in the second phase we move towards the creation of schema object from the relational schemas, and the last step is the mapping of data with Query by Example in db4o.

Keywords— Relational databases, database for objects db4o, Migration, Canonical Data Model CDM, Data Mapping, Object Oriented Database, Semantic Enrichment, Schema Translation, Query by Example.

I. INTRODUCTION

Most people who write software for a living have at least some familiarity with databases. Many of them are also familiar with object-oriented programming and have probably needed to use a database to provide persistence for software objects. db4o is an object database, which means that unlike the more common relational databases, it looks at data the same way that programs do. Something that is not possible with relational databases, because the relational model is a concept based on the principle of the relationship, all these data is organized in a table [1]. We tackle this problem by proposing a solution to the migration of RDB to DB4O for schemas and data mapping with Query by Example.

Traditional relational databases are dominant in the market as most data today still stored and maintained in systems relational databases. However, relational databases have their place in supporting complex structures and data types, user-supplied based on technologies defined objects and World Wide Web. Therefore, a new generation of database management systems began to emerge in the market, offering more features and flexibility. They have been proposed databases, in a relatively new oriented object database (OODB) [2], which support several different concepts to meet the requirements of complex applications (e.g., multimedia, computer aided design, etc.) that require rich data types. Therefore, it is expected that the need to convert the RDB in technologies that have emerged recently increased significantly such that the object database OODB [3].

1.1. OODBMS and db4o:

1.1.1. Instances of utilization and functionality:

When there is only one application at a time running on a database and if it's an application dedicated to mobility as found in systems based on location or in embedded software, this can worth trying a second generation OODBMS. Because of their ability to manage the relationships between objects, OODBMS are intrinsically well suited when complex object models, flat object or large object tree structures are in order - which is often the case in the Systems Geographical Information (SGI). Here are some of the technical features of db4o:

- Integrated mode and client-server mode.
- No server administration at runtime.
- The database properties are checked out of the host application.
- Need little disk space for the libraries of the program, as considerably as in memory during execution.
- The simplicity of use: db4o uses application programming interfaces (APIs) from reflective Java and.NET. There is therefore no additional annotation, no pre or post processing (byte code engineering), any subclass or interface implementation to perform.
- Lazy loading of Control Methods relationships embedded objects.
- Replication tools as supplements.

As one might expect from a database, db4o implements ACID (atomicity, consistency, isolation and durability) that provide safe transactions: a transaction starts when opening or queries the database, and ends with the methods commit () or rollback (). Three approaches are implemented for queries: Query by Example, the SODA queries (Simple Object Database Access, or Simple Object Access database) and 'Native Queries'.

1.1.2. Advantages:

Talking ORM environment leads us to compare OODBMS with this approach. Db4o is very fast if one thinks the test beds [4].

A conceptual argument is that there is no mismatch object / relational: no adjustment of the types or sizes of data (within separate relational, no SQL and no textual SQL) to manage. It is important to mention that db4o provides a convenient management of agile development techniques: it is because the requests are written in the language of the application (Java, .NET) and is, therefore, safe in terms the data typing. There are also friendly features schema evolution, not SQL. Software engineers have life easier than before because they reside in the object world, unlike the professional databases.

1.1.3. Restrictions

Db4o is probably not suited for employment in large data warehouses or data mining. It is especially recommended when multiple applications access the database with many views. The fact that different languages are available for queries and shows no mature standard language is available in comparison to the predominant SQL relational model. Constraints such as referential integrity are not (yet) part of any language except Native Queries that simply implement callback functions. Finally, the current lack of standardization has been recognized. The actions of the Management Subject Group (OMG) are under way to move to a new release of the Version 4 ODMG standard.

II. RELATED WORK

In engineering, migration relational databases there is significant research to solve the problem of setting up relational databases already in production in applications that runs on the object paradigm among this research there are certain assumptions migration with a single target (ORDB | OODB | XML) which could be a point of a disadvantage or limitation on the other they chose to work with an intermediary to migrate a relational database to several models.

The migration starts with the extraction of relationships and dependencies in the conceptual schema model as Entity-Relationship and its varieties; this is called the semantic enrichment [5] [6].

Maatuk and al. [7] propose an approach that takes an existing RDB as an input, expanding its representation of the metadata with the necessary semantics, and built a canonical data model (CDM) to generate three different targets (ORDB | OODB | XML). CDM capture the essential characteristics of target data models, and product is enriched with constraints and data semantics of RDB who may not have been explicitly expressed in the source metadata. The CDM is then mapped into target patterns according to internationally recognized standards such as ODMG 3.0 [8], SQL4 and XML.

Other researchers such as R Alhajj and al. [9] they implement a system that builds an understanding of a conventional base given by taking these characteristics as input and produces the object-oriented database corresponding output. The system draws a chart summarizing the conceptual model. Links in the graph are classified inheritance links and aggregation links. This classification resulted in the class hierarchy.

A. Behm and al.[10] they work in the first phase, with the use of transformation rules to build an object-oriented schema that is semantically equivalent to the relational schema. In the second phase, schema transformation information is used to generate programs which migrate the data in a relational oriented database object. The two concepts, the schema transformation and data migration are implemented using O2 as OODBMS.

A. El alami and al. [11] 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, this solution is based on the New Canonical data model (NDM).

Maatuk and al. suggest a solution takes an existing RDB input, expanding its representation of the metadata with the necessary semantics, and produce an improved model canonical data, which captures the essential features of the ORDB target, and is suitable for migration. A prototype has been developed, which migrates successfully BRD in ORDBs (Oracle 11g) based on the canonical model [12].

C. Fahrner and al. [13] propose a three-step process that first goal is to complete a given relational schema, namely, to make the semantic information also carries explicit as possible using a variety of data dependencies. A complete schematic is then converted into a ODMG schema in a simple way, by generating class relational schemas. The result is generally not optimal object-oriented point of view; so the original object-oriented scheme is finally improved to better exploit the available options in the object-oriented paradigm.

Another research focus also on the schema transformation like X. Zhang and al. [14] they discuss class structures and define well structured classes. Based on MVDs, a theorem is given the transformation of a relationship diagram in a well-structured class. With the aim of transforming the RDB schema diagram OODB, a composition process simplifying RDB entry scheme and an algorithm transforming the RDB simplified diagram in OODB well-structured classes are developed.

A new approach discusses the transition from relational databases to a document-oriented model of NoSQL. The method is based on the routing of a data model which establishes a migration of the physical schema and the data mapping [16].

In Our approach we discusses an approach that takes an existing relational database as input, obtains an object-oriented database in the output with its metadata and the principle of semantic enrichment to extract different object principles including aggregation, inheritance, and composition. The first phase of the migration process is the Canonical Data Model which is invented by Maatuk and al.[15] , in the second phase we move towards the creation of schema object from the
relational schemas with a schema transformation algorithm, and the last step we produce a data migration algorithm we always keep association and key in the mapping of data with Query by Example for db4o.

III. SEMANTIC ENRICHMENT

Semantic enrichment is a process of analyzing and examining a database to capture its structure and definitions at a higher level of meaning. This is done by enhancing a representation of an existing database's structure in order to make hidden semantics explicit.

Fundamentally, we mean enriching the content/context of data by tagging, categorizing, and/or classifying data in relationship to each other, to dictionaries, and/or other base reference sources. At its simplest, this means adding additional contextual information to some existing data set (think of adding traffic data to road maps where the traffic data provides the context of road conditions, the probability of delay, the length of projected obstructions, the condition of the road, etc.).

In the phase of the semantic enrichment, we operate the CDM to keep the semantic database and for creating OODB schema and data migration with transformation algorithms. Taking the example of a relational database in Fig.1:

![Fig.1 Relational databases for University](image)

<table>
<thead>
<tr>
<th>Student</th>
<th>stu_id*</th>
<th>stu_fname</th>
<th>stu_lname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Department</td>
<td>dep_code</td>
<td>dep_name</td>
<td></td>
</tr>
<tr>
<td>Instructor</td>
<td>ins_id*</td>
<td>ins_fname</td>
<td>ins_lname</td>
</tr>
<tr>
<td>Location</td>
<td>loc_code</td>
<td>loc_name</td>
<td>loc_country</td>
</tr>
<tr>
<td>Course</td>
<td>crs_code</td>
<td>crs_utile</td>
<td>crs_credits</td>
</tr>
<tr>
<td>Section</td>
<td>sec_id*</td>
<td>sec_term</td>
<td>sec_bldg</td>
</tr>
<tr>
<td>Enrollment</td>
<td>stu_id*</td>
<td>sec_id*</td>
<td>grade_code</td>
</tr>
<tr>
<td>Prerequisite</td>
<td>crs_code*</td>
<td>crs_requires*</td>
<td></td>
</tr>
<tr>
<td>Qualified</td>
<td>ins_id*</td>
<td>crs_code*</td>
<td></td>
</tr>
</tbody>
</table>

We show the result of a database table with the concept of university CDM in Table 1.

<table>
<thead>
<tr>
<th>Cn</th>
<th>Cls</th>
<th>Abs</th>
<th>A_db4o</th>
<th>Rel</th>
<th>UK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>RRC</td>
<td>false</td>
<td>stu_id</td>
<td>assso</td>
<td>stu_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>stu_fname</td>
<td>Enrollment</td>
<td>stu_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>stu_lname</td>
<td>Enrollment</td>
<td>stu_id</td>
</tr>
<tr>
<td>Department</td>
<td>RST</td>
<td>false</td>
<td>dep_code</td>
<td>assso</td>
<td>dep_code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dep_name</td>
<td>Course</td>
<td>dep_code</td>
</tr>
<tr>
<td>Instructor</td>
<td>SST</td>
<td>false</td>
<td>ins_id*</td>
<td>assso</td>
<td>Ins_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ins_fname</td>
<td>Section</td>
<td>Ins_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ins_lname</td>
<td>Ins_id</td>
<td>Instr_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dep_code</td>
<td>Instr_id</td>
<td>Instr_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Dept_name</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1: Results of CDM generation

IV. SCHEMA TRANSLATION

In this section we explain how the CDM translates to a target schema equivalent DB4O. A set of rules was designed to bring the classes and relationships.

A target DB4O schema is defined as a set of class:

\[ \text{OOSchemaDb4o} := \{ \text{C}_{db4o} \mid \text{C}_{db4o} = \langle \text{name}, \text{Att}_{db4o}, \text{Rel}_{db4o} \rangle \} \]

Where Att its attributes, and Rel, the relationship with class.

\[ \text{Att}_{db4o} := \{ \text{A}_{db4o} \mid \text{A}_{db4o} = \langle \text{An}, \text{t}, \text{sv} \mid \text{cv} \rangle \} \]

\[ \text{Rel}_{db4o} := \{ \text{Rel} \mid \text{Rel} = \langle \text{reln}, \text{dirCn}, m, \text{invReln} \rangle \} \]

The processing algorithm based on the CDM to simplify the migration task to OODB schema for all classes that belong to CDM if the classification of the class C is an SST | SUR | SSC and the relationship between class is composition, we create a variable that contains the Var compo element in determining the location of a class C with respect to the class C', either within the class or to outside, in the case of a composition within an inheritance, in the case of assignment to Class compo value Cn we create a collection for compo. For the relations between class there are two types with either a single value or multiple values if the relationship is between two classes (0…* 1…*) so the relationship class is a relationship with more value.

For treatment of the inheritance if the classification of a class C is SST then we create a new class and a variable n to set the location of the class C in the CDM within a loop is incremented to browse the following class.

If the classification of a class C is SSC we create a class C extend the class C that accepts the inheritance, if not we will verify the classification is that it is a SUB is we create another class C " extend the class C is class final or not accepted the inheritance.
If the classification of a class C and a RST and the type of relationship is a composition then we create a simple collection if not a single class. For relations in cases of RRC and CAC if the relationship is (m, n) then we create a class with its own owner.

The ProduceDB4Oschema algorithm shown in Figure 2 implements these translation rules.

1. Algorithm ProduceDB4Oschema (cdm: CDM) return OOschema
2. Target Schema: OOSchema := Ø
3. Foreach class C ∈ cdm do // Composition inside inheritance
4. If (Cn.classification = (SST | SUR | SSC) && RelType = composition) then
5. Var compo = getDirC
6. Foreach C ∈ CDM do
7. If Cn = compo then
8. Create collection of compo
9. end if
10. end for
11. Foreach relationship ∈ C.relation do
12. If c.relation = (0...*,1...*) then
13. Multiplicity := 'Collection value'
14. Else multiplicity := 'Single value'
15. ElseIf C.classification = SST then // Treatment of Inheritance
16. Create class Cn
17. Specify dirC (element n) Else if c.classification = SSC then
18. Create class Cn extend element n
19. Specify dirC (element m)
20. Else if C.classification = SUB then
21. Create class Cn extend element m final
22. End if
23. If (Cn.classification = (RST) && RelType = composition) then
24. Var compo = getDirC
25. Foreach C ∈ CDM do
26. If Cn = compo then
27. Create collection of compo
28. end if
29. end for
30. Else
31. Create class Cn
32. End if
33. Foreach rel ∈ RELdb4o do
34. If C.classification = (RRC | CAC) && rel.relType = "associate with" then
35. If rel.c = (m,n) then
36. Create class with its own propriety
37. End if
38. End if

V. DATA CONVERSION WITH QBE

In this section we describe the GenerateDb4oData algorithm given in Figure 4, for processing and filling OODB schema generated from the schema translation phase, we choose db4o for several reasons is cited in the introduction queries in db4o is based on the query by Example, and for the extraction of data we have operated a JDBC programming interface given in Figure 3, that allows Java applications to access through a common interface to databases relational why there are JDBC drivers.

public class Connect {
public static void main(String[] args) {
String tabname;
try {
Class.forName("org.postgresql.Driver");
String url = "jdbc:postgresql://localhost:5432/University";
String user = "admin";
String passwd = "admin";
Connection condb = DriverManager.getConnection(url, user, passwd);
Statement sql4o = condb.createStatement();
ResultSet res = sql4o.executeQuery("SELECT *
FROM tabname");
ResultSetMetaData resMeta = res.getMetaData();
For (int i = 1; i <= resMeta.getColumnCount(); i++)
System.out.print("\t" + resMeta.getColumnName(i).toUpperCase() + " \* |");
while(res.next()){
for(int i = 1; i <=
resMeta.getColumnCount(); i++)
System.out.println("\t" + res.getObject(i).toString() + " | ");
}
res.close();
sql4o.close();
} catch (Exception e) {
PrintStackTrace();
}
}

Fig.3 The implementation of selection queries

After extracting the data we have injected into the OODB schema generated from the schema translation phase, it always keeps the relationships between classes and references is marked with first and second keys in the RDB database.

1. Algorithm Generate Db4oData (cdm: CDM) return Db4oData
2. Foreach C ∈ CDM do
3. T is an array of type string,
VI. Conclusion

This article provides a solution to the problem of translation of RDB schemas and data objects based on db4o. The solution generates a basic type of most recent data object and can easily integrate into Java or .Net projects also queries based Query by Example which is optimal for querying a OODB basis in all stages of migration we proposed algorithms in its main objectives is the semantic enrichment of maintenance to extract the different characteristics of the object, including aggregation, inheritance and composition.

REFERENCES