International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:8, No:1, 2014

Migration of the Relational Data Base (RDB) to the Object Relational Data Base (ORDB)

Alae El Alami, Mohamed Bahaj

***Abstract*—**This paper proposes an approach for translating an existing relational database (RDB) schema into ORDB. The transition is done with methods that can extract various functions from a RDB which is based on aggregations, associations between the various tables, and the reflexive relationships. These methods can extract even the inheritance knowing that no process of reverse engineering can know that it is an Inheritance; therefore, our approach exceeded all of the previous studies made for the transition from RDB to ORDB. In summation, the creation of the New Data Model (NDM) that stocks the RDB in a form of a structured table, and from the NDM we create our navigational model in order to simplify the implementation object from which we develop our different types. Through these types we precede to the last step, the creation of tables.

The step mentioned above does not require any human interference. All this is done automatically, and a prototype has already been created which proves the effectiveness of this approach.

***Keywords*—**Relational databases, Object-relational databases, Semantic enrichment.

I. Introduction

MANY problems have emerged with the RDB [9]. We recall from them the reconstruction of complex objects split across relational tables is costly because it causes many joins. For this a solution has appeared, it is the ORDB [7]. Who has addressed most of these problems, we recall the use of reference facilitates the use of very large multimedia data by allowing them to be easily shared and costs less. Yet, the question that arises is how to achieve a migration from a RDB to an ORDB.

Several approaches have addressed the topic of migration which shows the transformation of the aggregation and associations from the conceptual model to the object relational model [1]. Based on the notion of collections of the Unified Modeling Language [8]. Other approaches are based on the creation of an ORDB from the UML [4]. In those authors have proposed the use of cardinalities to preserve and store the aggregations and compositions [10].

An approach takes all of the relational database and stores it in a structured table [2], [5], [6] that contains several parameters, tables, attributes, classifications, class types (abstract / concrete), the names of the relationships, class that interacts with the relationship and cardinalities in order to realize the schema translation.

Yet all approaches require the interference of the human factor in order to achieve migration, either on all the work, or on one or several parts. Therefore, in this paper we propose a method that requires no human interference.

Our approach is based on capturing metadata from a RDB that treats and stores it as a structured table that holds the information necessary for the migration.

This paper deals with the steps of the migration that are composed of 3 parts: the first is the implementation of the structured table to the NDM in Section II; the second part is the conversion from the NDM to the navigational model [3] in Section III; and the third part deals with the transformation of the Navigational Model (NavM) to the ORDB.

II. Semantic Enrichment of Relational Database: New Data Model

A. Definition and Identification of the New Data Model

The NDM is a type of table describing the different classes extracted from a RDB with the data necessary for the realization of an ORDB.

The NDM is defined as a collection of classes

NDM: = {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.

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)

*Observation: treating cardinalities cannot help us since the transformation of the Conceptual Data Model (CDM) to the Logical Data Model (LDM) in the RDB has been treated for the migration of attributes.

1. Classification (cls)

For the classification, classes are composed of four parts:

- Aggregation is when the class interacts with a single class (the class itself may be the (first degree | 2nd degree)), not included in the classification and inheritance as the class has a FK;

Alae El Alami is a Phd in the Faculty of Science and Technology / Department of Mathematics and Computer Sciences, University Hassan I Settat, Morocco (e-mail: elalamialae@gmail.com).

Mohamed Bahaj is Professor the Faculty of Science and Technology / Department of Mathematics and Computer Sciences, University Hassan I Settat, Morocco (e-mail: mohamedbahaj@gmail.com).

- Associations: a class of 2nd degree which interacts with two or more classes; Special case: for reflexive relationships including their cardinalities in the CDM were « 1-n» «0-n » becomes an association with two FK with the name of the attribute of FK not found in any other table in the RDB, which includes a special treatment of the entities to know the referencing.
- Inheritance: When a class inherits from another class (no reverse approach can identify the inheritance).

For this we have developed a technique that is based on creating a table that will contain a maximum of possible probabilities of inheritance in accordance with the naming standards. Our table will consist of two columns: the first containing the mother class; and the second, the subclasses.

In order to know the inheritance we compare the names of the first-degree classes of the NDM with the first column of our created table. If a correspondence is found, we compare all the discovered classes that interact with our class within the NDM with the subclasses from the corresponding line of the table that contain possible inheritances. If an inheritance is found in this step, we precede to the last step of verification. We verify whether the PK value of the subclasses is equivalent to the PK value of the mother class in order to avoid a breach of the rules for naming tables.

So if there is a match we will extract an Inheritance, if not we will continue our treatment of classification knowing that the treatment of inheritance is the first step of the classification.

Here is an example that shows the most famous case of use of heritage shown in Table I.

TABLE I
THE CASE OF INHERITANCE THE MOST COMMON

Inherited By	Inherits
Person	Student
	Teacher
Animal	Cat
	Dog
	Horse
Document	Book
	Newspaper
Account	Savings
	Current
	…
…	…
	…

- The simple classes: the classes of the first degree, which are not within the classification of aggregation, inheritance and association.

2. Contributor

It defines the list of classes that the class starts interacting with themselves. The purpose of this part is to know whether it is an aggregation, an association, a single class or a class that inherits from the class starting, also for the creation of reference during the transition to the ORDB.

B. Generation of the NDM from a RDB

The translation of the RDB to the NDM is the first step of the migration into the ORDB

Consider that the RDB includes in Fig. 1. Example of NDM show in Table II generated from a RDB.

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 (the pks are underlined in bold **ex**, the fks are underlined ex)

TABLE II
RESULT OF THE GENERATION OF A NDM

Cn	Degre	Classification	Attribute						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

Observation: VARCHAR is synonymous with VARCHAR2 but this usage may change in future versions (provided for backward compatibility only for Oracle datatypes [11]).

III. NAVM (NAVIGATIONAL MODEL)

After obtaining the NDM, we create the navigational model.

A. Definition and Objectives

- A model that plots the object implementation of a database while drawing up the navigation path between relations with the principle of referencing.
- Facilitates the transition towards the object by a set of rules for transposition.
- Promotes the Visualization of complex structures and possible navigation paths.

Why navigational?

The model introduces the logical links of the type REF (REF implementation is undetermined in the conceptual level).

The references (ref or REF) facilitate the navigation between objects.

TABLE III
NAVIGATION SYMBOL OF THE NAVM

Simple	Multiple
Single link	Multiple link
Single link can be null	Multiple link can be null
Single link necessarily valued	Multiple link necessarily valued

The absence of the circle means necessarily valued

The classes will be divided into two parts, the external classes and internal classes:

+ Internal classes are the classes classified as aggregation in the NDM.
+ External classes are the other classifications in the NDM.

Example

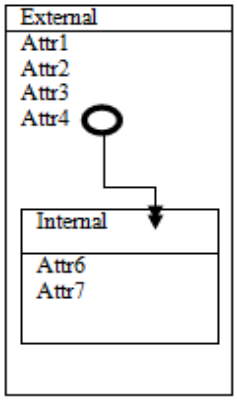

Fig. 2 The composition of the external and internal classes

Example of the NavM extracted from the UML:

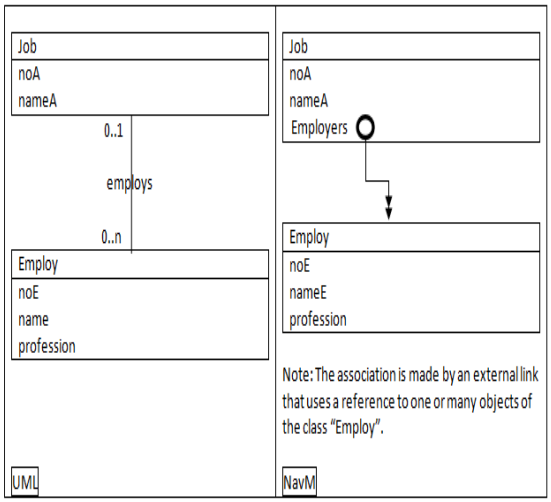

Fig. 3 From the conceptual model to the navigational model

B. Transformation Rules

Fig. 4 Modeling inheritance in navigational

Inheritance: It follows the same principle of the class diagram in the UML, either for the parent class or subclass.

Association: The navigation link is simple and necessarily valued keeping attributes if there is an association with the class attributes.

The navigation links starts from the association class to the class that interacts with it (universal solution).

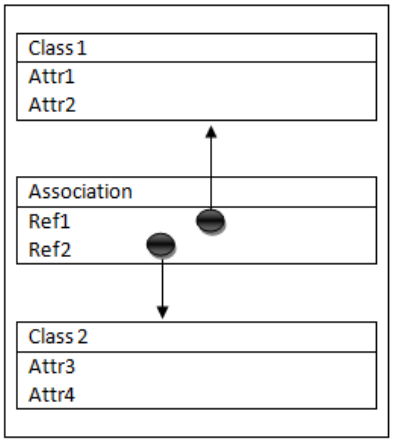

Fig. 5 How to model the association link

Aggregation: It becomes an internal class type object referenced by an attribute of assembly with a multiple link that can be valueless.

Simple class: For the simple class we must see the classification of the class that interacts with it in the NDM.

+ If the simple class interacts with an association, we will not need to trace the path of navigation because it is already done.
+ If a simple class interacts with another simple class, we have two navigation links:
- -The first link starts of the class that contains a foreign key, which is a primary key in the other class, and is simple and cannot be valueless.
- -The second link starts from the other class, which is multiple and can be valueless.

C. The NDM Transformed into the NAVM

This stage of the migration is a part of reverse engineering to show the transition by reference and the elimination of joints, and plays a pivot role between the conceptual and implementation object.

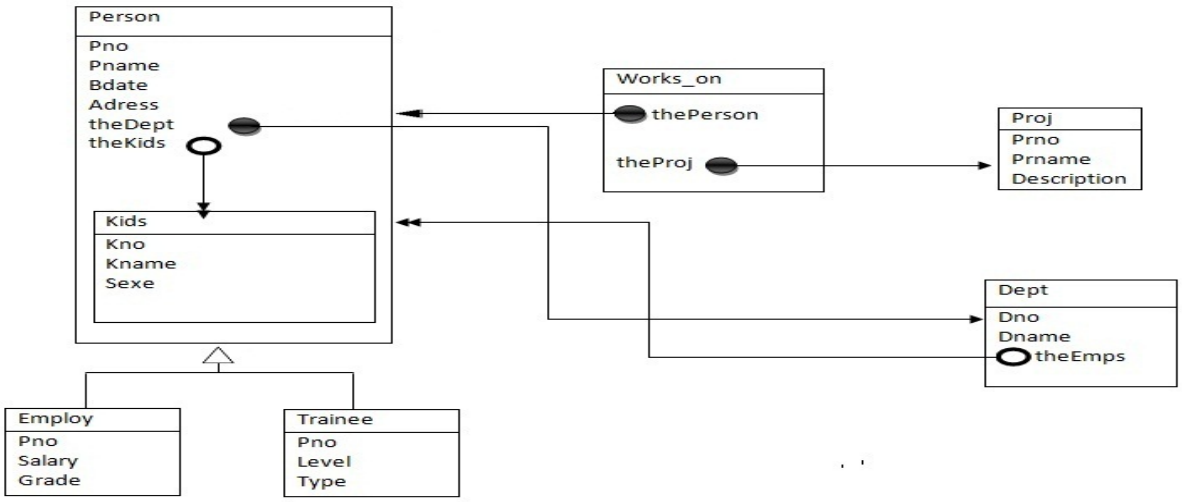

Fig. 6 Transformation of the relational model to the navigational model

IV. Translation of the NDM to an ORDB

A. Approach for the Translation of the NDM into an ORDB

1. The Creation of Types

+ Creation of the types defined in the NDM as aggregation.
+ Creation of the types defined in the NDM as association.

To create these types we keep the same name listed in the RDB and we add **_type** (concatenation).

+ Creation of composite types, those classes entering in collaboration with the aggregation taking into account their classification, and other types whose classification in the NDM is simple.
+ The creation of the types defined in the NDM as an inheritance starting with the parent class and ending with the subclasses.

2. Creating Tables

The creation of tables is made by the typed classes and is classified in the NDM as inheritance (parent, subclasses), association, and simple class. The aggregations are included in the first-degree class that interacts with itself. All tables are created with the necessary constraints.

B. Method of Creating and Naming Rule

To create the types we keep the same name that appears in the RDB and we add _type (concatenation).

- Syntax

CREATE [OR REPLACE] TYPE nameRDB_Type AS OBJECT
(column1 type1, column2 type2,...)

To create types that contain other types that represent aggregations, the type name that represents the aggregation remains the same and we add **_t** .

- Syntax

CREATE [OR REPLACE] TYPE nameRDB1_Type AS OBJECT
(column1 type1, column2 type2,...)
/
CREATE [OR REPLACE] TYPE nameRDB2_Type AS OBJECT
(column1 type1, column2 type2, nameRDB1_t set(nameRDB1_type),...)

For the creation of types with references, we add a **ref_** next to the name of the RDB with the keyword **REF** and the referenced type.

Observation: For reflexive relationships near many recordings [1-n] we concatenate the PK with the FK, and the side of a single record [1-1] we concatenate the FK with the PK.

- Syntax

CREATE [OR REPLACE] TYPE nameRDB1_Type AS OBJECT
(column1 type1, column2 type2,...)
/
CREATE [OR REPLACE] TYPE nameRDB2_Type AS OBJECT
(column1 type1, column2 type2,nameRDB1_t nameRDB1_type,...)
/
CREATE [OR REPLACE] TYPE nameRDB3_Type AS OBJECT
(column1 type1, column2 type2,ref_nameRDB2 REF nameRDB2_type,...)

For the creation of types that represent the inheritance, we add **Under** for the sub class and the keyword **not final** if the type has subtypes, and **final** if the type has no subtypes.

- Syntax

CREATE [OR REPLACE] TYPE nameRDB1_Type AS OBJECT
(column1 type1, column2 type2,...)
NOT FINAL
/

CREATE [OR REPLACE] TYPE nameRDB2_Type under nameRDB_type
(column1 type1, column2 type2,nameRDB1_t nameRDB1_type,...)
FINAL

Creating tables starts from typed classes. The table keeps the same name that appears in the RDB, and then we add the keyword **OF** and the type corresponding with the constraints captured in the NDM (PK constraint, reference constraint, not null constraint ...).

- Syntax:

CREATE TABLE [schema.]nameTable OF [schema.] nameType
[(column [DEFAULT expression]
[constraintOnLine [constraintOnLine]...
| constraintREFOnLine]
| { constraintOffline | constraintREFOffline }
[,column...])]
;

V. Conclusion

The aforementioned work shows the steps of migrating from a RDB to an ORDB with a simple and practical method to capture the relationships between different classes, associations, aggregations and as well as the inheritance. Currently no approach has proposed such a solution to extract the inheritance from a RDB. We can trace the navigational model to better see how the navigation is made between classes and respect the navigation links for best listings.

This method is done with a normalized database to exploit the power of the object-relational as our solution of inheritance is based on a normalized database. If the normalized database is not used, we will have a simple class with an aggregation in place of the mother and sub class.

This solution exceeds the existing works as it generates an ORDB without the interference of the human factor. This approach also allows the possibility to make changes in the physical schema of the database obtained in cases where the user wants to manually update the database. Since the work is done in console mode, a prototype was created to prove the effectiveness of this approach.

A forthcoming article will present a prototype that examines the subsequent stage of the migration that affects the passage of the data from a RDB to an ODB.

TABLE IV
Final Result of the Migration

```
CREATE TYPE kids_type AS OBJECT
(kno int, kname varchar(20),sex char(1),pno varchar(20))
/

CREATE TYPE dept_type AS OBJECT
(dno int, dname varchar(20))
/

CREATE TYPE proj_type AS OBJECT
(prno int, prname varchar(20),description varchar(255))
/

CREATE TYPE person_type AS OBJECT
(pno varchar(10),pname varchar(20),
bdate date,address varchar(255),
dno int , pnosup varchar(20),
kids_t set(kids_type),
ref_dept ref (dept_type)scope dept,
ref_pno_pnosup set(ref(person_type)),
ref_pnosup_pno ref(person_type) scope person)
 NOT FINAL
/

CREATE TYPE trainee_type UNDER person_type
(pno varchar(10), level varchar(20),type varchar(20))
FINAL
/

CREATE TYPE employe_type UNDER person_type
(pno varchar(10),salary int,grade varchar(20))
 FINAL
/

CREATE TYPE work_on_type AS OBJECT
(prno int, pno varchar(20),
ref_proj set (ref(proj_type)),
ref_person set(ref(person_type)))
/

CREATE TABLE dept  OF dept_type(
constraint pk_dept primary key(dno));


CREATE TABLE proj  OF proj_type(
constraint pk_proj primary key(prno));

CREATE TABLE work_on  OF work_on_type(
constraint refer_work_on_person ref_person references person,
constraint refer_work_on_proj ref_proj references proj);

CREATE TABLE person OF person_type(
constraint pk_person primary key(pno),
constraint refer_person ref_dept references dept);

CREATE TABLE trainee OF trainee_type UNDER person;
CREATE TABLE employe OF employe_type UNDER person;
```

REFERENCES

[1] G. O. Young, "Synthetic Structure of Industrial Plastics (Book Style with Paper Title and Editor)," in Plastics, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.
[2] Abdelsalam Maatuk, Akhtar Ali, Nick Rossiter: Semantic Enrichment: The First Phase of Relational Database Migration. In CIS2E '08, 6pp, Bridgeport, USA, 2008.
[3] Nora Koch, Hubert Baumeister, Rolf Hennicker and Luis Mandel. : Extending UML to Model Navigation and Presentation in Web Applications. In Proc. of the Workshop Modeling Web Applications in the UML, UML'00, 2000.
[4] Ming Wang. : Using UML for Object-Relational Database Systems Development: A Framework. Issues in Information Systems, VOL 9, No. 2, 2008.
[5] Maatuk, A., Ali, M. A. and Rossiter, N.: An Integrated Approach to Relational Database Migration. In IC-ICT '08, pp. 16, Bannu, Pakistan, 2008.
[6] Abdelsalam Maatuk,M. Akhtar Ali,Nick Rossiter . : Converting Relational Databases into Object-relational Databases. in JOT, vol. 9, no. 2, pages 145-161, 2010.
[7] Stonebraker, Michael, Moore and Dorothy. Object-Relational DBMSs: The Next Great Wave (Morgan Kaufmann Series in Data Management Systems) ISBN: 1558603972.
[8] K. Barclay and J. Savage-Object-Oriented Design with UML and Java,2004, ISBN 0 7506 6098 8.
[9] S. Sumathi, S. Esakkirajan —Fundamentals of Relational Database Management Systems, 2007, ISBN 978-3-540-48397-7.
[10] 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.
[11] http://www.oracle.com/technetwork/database/enterprise-edition/documentation/index.html.