An Integrated Approach to Relational Database Migration

Abdelsalam Maatuk, Akhtar Ali, and Nick Rossiter
School of Computing, Engineering & Information Sciences
Northumbria University, Newcastle upon Tyne, UK
{abdelsalam.maatuk;akhtar.ali;nick.rossiter}@unn.ac.uk

Abstract

Relational DataBases (RDBs) are dominant in the market place yet they have limitations in the support of complex structure and user-defined data types provided by relatively recent database technologies (i.e., object-based and XML databases). Such a mismatch inspires work on migrating an RDB into these technologies. The problem is how to effectively migrate existing RDBs, as a source, into the recent database technologies, as targets, and what is the best way to enrich RDBs' semantics and constraints in order to meet the characteristics of these targets? Existing work does not appear to provide a solution for more than one target database. We tackle this question by proposing a solution for migrating an RDB into these targets based on available standards. The solution takes an existing RDB as input, enriches its metadata representation with as much semantics as possible, and constructs an enhanced Relational Schema Representation (RSR). Based on the RSR, a canonical data model is generated, which captures essential characteristics of the target data models that are suitable for migration. A prototype has been implemented, which successfully migrates RDBs into object-oriented, object-relational and XML databases using the canonical data model.

Keywords: Database, Database migration, Schema translation, Data conversion.

1. Introduction

Many organisations have stored their data in RDBs and aspire to take advantage of databases that have emerged more recently, e.g., object-based/XML databases. Instead of discarding existing RDBs or building non-relational applications on top of them, it is preferred to convert existing relational data into a new environment. However, the question is which of the new databases is most appropriate to move to? So there is a need for a method that deals with database migration from RDB to Object-Oriented DataBase (OODB)/Object-Relational DataBase (ORDB)/XML in order to provide an opportunity for exploration, experimentation and comparison among alternative database technologies. The method should assist in evaluating and choosing the most appropriate target database to adopt for non-relational applications to be developed according to required functionality, performance and suitability, and could help increase their acceptance among enterprises and practitioners. However, the difficulty facing this method is that it is targeting multiple database models that are conceptually different.

Research by others focuses on diverse areas of RDB migration. Most of existing proposals are restricted by a range of assumptions before the migration process can begin. A taxonomy on database migration proposals and techniques can be found in our work [9]. However, the existing work does not provide a solution for more than one target database, for either schema or data conversion. Besides, none of the previous proposals can be considered as a method for converting an RDB into an ORDB. In this paper, we propose an integrated method for **MIG**rating an **R**DB into **O**bject-based and **X**ml databases (MIGROX), which is able to preserve the structure and semantics of an existing RDB to generate OODB/ORDB/XML schemas, and to find an effective way to load data into target databases without loss or redundancies. The method is superior to the existing proposals as it can produce three different output databases as shown in Figure 1. In addition, the method exploits the range of powerful features that target data models provide such as ODMG 3.0, SQL4, and XML Schema. Due to the heterogeneity among the three target models, we believe that it is necessary to develop a Canonical Data Model (CDM) to bridge the semantic gap between them. The CDM is able to preserve and enhance RDB's integrity constraints and data semantics to fit in with target database's characteristics. MIGROX has three phases: 1) semantic enrichment, 2) schema translation and 3) data conversion. In the 1^{st} phase, the method produces a CDM, which is enriched with an RDB's data semantics that may not have been explicitly expressed in it. The CDM so obtained is mapped into target schemas in the 2^{nd} phase. The 3^{rd} phase con-

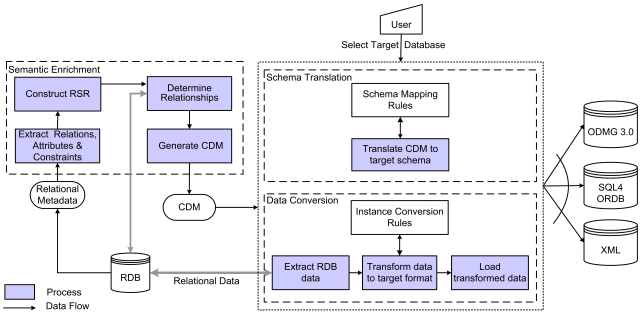

Figure 1. An Overview of MIGROX

verts an RDB data into its equivalents in the new database environment. A system architecture has been designed and a prototype implemented to demonstrate the process, which resulted successfully in target databases.

This paper is structured as follows. Section 2 provides an introduction to the semantic enrichment phase. An overview of the schema translation phase is introduced in Section 3. Section 4 presents the data conversion phase. Section 5 reviews some results of the MIGROX prototype. The related work is presented in Section 6, and Section 7 concludes the paper and points to future work.

2. Semantic Enrichment

The semantic enrichment phase involves the extraction of data semantics of an RDB to be represented in a Relational Schema Representation (RSR) followed by conversion into a much enriched CDM. This facilitates migration into new target databases without referring to the existing RDB repeatedly. The main benefit from using RSR and a CDM together is that an RDB is read and enriched once while the results can be used many times to serve different purposes. Further details on RSR and CDM definitions and their extraction from RDBs can be found in [8]. The semantic enrichment starts by extracting the basic metadata information about an existing RDB in order to construct RSR. To get the best results, it is preferable that the process is applied to a schema in 3rd Normal Form (3NF). The next steps are to identify the CDM constructs based on a classification of RSR constructs, including relationships and cardinalities, which are performed through data access. Lastly, the CDM structure is generated.

2.1. Extracting RSR

RSR provides an image of metadata obtained from an RDB. An efficient RSR construction overcomes the complications that occur during matching of keys in order to classify relations, attributes and relationships. Basic information needed to proceed with the semantic enrichment includes relation names and attributes' properties, e.g., attribute names, data types, length. Moreover, the most important information needed is about keys including Unique Keys (UKs). We assume that data dependencies are represented by Primary Keys (PKs) and Foreign Keys (FKs). The inverse of an FK is called an Exported Key (EK). EKs play an important role as regards to OODB/ORDB, which support bi-directional relationships.

Definition 1: An RDB schema is represented in our approach as a set of elements,

RSR := $\{R \mid R := \langle r_n, \mathbf{A}_{rsr}, \mathbf{PK}, \mathbf{FK}, \mathbf{EK}, \mathbf{UK}\rangle\}$, where r_n denotes the name of relation R, $\mathbf{PK}$, $\mathbf{FK}$, $\mathbf{EK}$ and $\mathbf{UK}$ denote primary key, set of foreign key(s), set of exported key(s) and set of unique keys of R, respectively.

2.2. Generating CDM

This subsection presents a formal definition of CDM. In this study the CDM is designed to upgrade the semantic level of RDB and to play the role of an intermediate stage for migrating RDB during schema translation and data conversion phases. The CDM represents explicit as well as implicit semantics of an existing RDB. Explicit semantics include relation and attribute names, keys, etc.; implicit semantics include classification of classes and attributes, and relationship names, types, cardinalities and inverse relationships. Through the CDM, well-structured target databases can be obtained without proliferation of references and redundancy. However, its richness may not be fully exploited due to the relatively limited expressiveness of the input RDB. For instance, object-based models encapsulate static, i.e., attributes and relationships and dynamic aspects, i.e., methods of objects. Dynamic aspects get less attention in CDM compared to static aspects because an RDB does not support methods attached to tables.

CDM has three concepts: class, attribute and relationship. The model can be seen as an independent model, which embraces OODB concepts with rich semantics that cater for ORDB and XML. In order to express as much semantics as possible, the model has taken into consideration features that are provided by the target models such as association, aggregation and inheritance. Real world entities, multi-valued and composite attributes, and relationship relations are all represented as classes in CDM.

Definition 2: CDM is defined as a set of classes,

$CDM := \{C \mid C := \langle cn, cls, abs, A_{cdm}, Rel, UK\rangle\}$, where each class C has a name cn, a classification cls and whether it is abstract or not. Each C has a set of attributes A_{cdm}, a set of relationship Rel and a set of UKs UK.

- *Classification* (cls): A class C is classified into different kinds of classes, which facilitate its translation into target schema, e.g., regular strong class (RST), subclass (SUB), composite attribute class (CAC).

- *Abstraction* (abs): A superclass is abstract (i.e., $abs :=$ `true`) when all its objects are members of its subtype objects. Instances of an abstract type cannot appear in database extension but are subsumed into instances of its subtypes. A class is not abstract (i.e., $abs :=$ `false`) when all (or some of) its corresponding RDB table rows are not members of other subtable rows.

- *Attributes* (A_{cdm}): A class C has a set of attributes of primitive data type.

  $A_{cdm} := \{a \mid a := \langle a_n, t, tag, l, n, d\rangle\}$, where each attribute a has a name a_n, data type t and a tag, which classifies attributes into a non-key 'NK', 'PK', 'FK' or both PK and FK attribute 'PF'. Each a can have a length l and may have a default value d whereas n indicates that a is nullable or not.

- *Relationships* (Rel): A class C has a set of relationships Rel. Each relationship $rel \in Rel$ between C and another class C' is defined in C to represent an association, aggregation or inheritance.

  $Rel := \{rel \mid rel := \langle RelType, dirC, dirAs, c, invAs\rangle\}$, where $RelType$ is a relationship type, $dirC$ is the name of C', $dirAs$ denotes a set containing the attribute names representing the relationship from C' side, and $invAs$ denotes a set of inverse attribute names representing the inverse relationship from C side. $RelType$ can have the followings values: '*associated with*' for association, '*aggregates*' for aggregation, and '*inherits*' or '*inherited by*' for inheritance. Cardinality c is defined by $min..max$ notation to indicate the occurrence of C' object(s) within C objects, where min is a minimum cardinality and max is a maximum cardinality. Querying data in a complete database is used to extract cardinality constraints.

Using key matching, RSR relations and their attributes are classified, relationships among relations are identified and their cardinalities are determined and translated into equivalents in the CDM. Abstraction of each class in CDM is checked. We assume that, in RDB, the kinds of relations are identified and relationships are represented by means of PKs/FKs. For example, a weak entity/relation is identified when a PK of a relation is a superset of FKs and the to-one relationship is determined when an FK refers to a PK. Other representations may lead to different target constructs. In addition, in Extended Entity Relationship Model (EERM), an inheritance is represented using generalization/specialzation, which have different types. However, such types of specialization can be represented (indirectly) in relational models by many alternative ways. The most common alternative is one relation for a superclass and one relation for every subclass. MIGROX assumes this alternative because it is based on PKs/FKs matching, and without user help it would not be possible to automatically identify the other alternatives. As in the target database standards, multiple-inheritance is not supported in MIGROX.

Consider the database shown in Figure 2. PKs are in *italics* and FKs are marked by "*". Figure 3 shows the resulting CDM for only `EMP` and `DEPT` classes. The CDM's class `EMP` has attributes: `ename`, `eno`, `bdate`, `address`, `spreno` and `dno`. Other properties (e.g., attributes' types, tags, default values) are not shown for the sake of space. The class `EMP` is '*associated with*' classes: `DEPT`, `WORKS_ON` and with itself. Moreover, it '*aggregates*' `KIDS` class and '*inherited by*' `SALARIED_EMP` and `HOURLY_EMP` classes. Relationships with cardinalities are defined in each class as: $RelType\ \{invAs \longleftrightarrow dirC(dirAs)_c\}$ ($\longleftrightarrow$ indicates bidirectional association and $\rightarrow$ indicates unidirectional aggradation).

EMP

ENAME	*ENO*	BDATE	ADDRESS	SPRENO*	DNO*
Smith	12345	09-Jan-55	31 Hampstead Rd, NE4 8AB	86655	3
Wong	34455	08-Dec-45	16 Hampstead Rd, NE1 7RU	86655	3
Scott	53453	31-Jul-62	4 Northcote St., NE5 5AL	34455	1
Ali	68844	15-Sep-52	49 Hill Street, RG1 2NU	34455	1
Borg	86655	10-Nov-27	162 London Road, OL1 4BX	null	2
Wallace	54321	20-Jun-31	91 St James Gate, NE1 4BB	86655	2

SALARIED_EMP

*ENO**	SALARY
86655	55000
54321	43000

HOURLY_EMP

*ENO**	PAY_SCALE
12345	3
34455	4
53453	2
68844	3

WORKS_ON

*ENO**	*PNO**
12345	1
34534	1
34455	1
12345	2
34534	2
34455	2
68844	3
34455	3
54321	4
54321	5
86655	5

PROJ

PNAME	*PNUM*	PLOCATION	DNUM*
Way Station 1	1	Newcastle	3
Way Station 2	2	London	3
Way Station 3	3	Reading	3
Salford House	4	Salford	2
4 Times Square	5	Reading	1

DEPT_LOCATIONS

*DNO**	*LOCATION*
1	Reading
2	Salford
2	Newcastle
3	London

KIDS

*ENO**	*KNAME*	SEX
12345	Alice	F
12345	Michael	M
34455	Alice	F
34455	Joy	F
54321	Scott	M

DEPT

DNAME	*DNO*	MGR*	STARTD
Accounts	1	86655	19-Jun-71
Administration	2	54321	01-Jan-85
Finance	3	12345	06-Oct-05

Figure 2. Sample company database

```
EMP[A_cdm := {ename, eno, bdate, address, spreno, dno},
Rel := {associated with{dno ⟷ DEPT(dnum)1..1,
eno ⟷ DEPT(mgr)0..1, spreno ⟷ EMP(eno)1..1,
eno ⟷ EMP(spreno)0..*, eno ⟷ WORKS_ON(eno)1..* },
aggregates{ eno → KIDS(eno)0..*},
inherited by {SALARIED_EMP, HOURLY_EMP}}]

DEPT[A_cdm := {dname, dnum, mgr, startd},
Rel := {associated with {mgr ⟷ EMP(eno)1..1,
dnum ⟷ EMP(dno)1..*, dnum ⟷ PROJ(dnum)1..*},
aggregates {dno → DEPT_LOCATIONS(dno)1..*}}]
```

Figure 3. Sample generated CDM schema

3. Schema Translation

The schema translation phase aims at translating CDM into its equivalent targets schemas. Target schemas hold equivalent semantics to those of an existing RDB, which are enhanced and preserved in the CDM. Three sets of translation rules are designed for mapping CDM into target schemas. Algorithms are developed for producing target schemas according to these rules. We provide here a skeleton of the target models, introduce the schema trans-

lation and provide fragments of schema generated by our prototype. Full details may be found in [8].

Definition 3: A target ODMG 3.0 schema is defined as a set of classes $OOschema := \{C_{oo} \mid C_{oo} := \langle c_n, spr, k, A_{oo}, Rel_{oo}\rangle\}$, where c_n is a name of a class C_{oo}, spr is the name of its superclass, k is its primary key, A_{oo} is a set of its attributes and Rel_{oo} is a set of relationship types in which C_{oo} participates.

Definition 4: A target SQL4 ORDB schema is represented as $ORschema := \{UT, TT, UK_{or}\}$, where UT is a set of User Defined Types (UDTs), TT is a set of typed tables and UK_{or} is a set of UKs.

Definition 5: A target XML Schema is represented as $XMLschema := \{Root, GT\}$, where $Root$ is a global element declared under the schema with its direct local subelements and constraints, and GT is a set consisting of global complex types, which are defined to be referenced as types of subelements that are declared within the $Root$ or by other defined global complex types.

Given a CDM, the schema translation starts by asking the user to determine which target is to be produced. Then, an appropriate set of rules is implemented to map the CDM into equivalent constructs in the target schema. Each rule maps a specific construct, e.g., class or attribute. By using CDM constructs classification, we can identify their equivalents in target schema definition language. For instance a CDM class that is classified as CAC is mapped into target database as composite attribute (e.g., **struct**). Attributes $C.A_{cdm}$ are translated into equivalents with the same names as that of CDM and their types are converted according to target data types. Keys are specified when attributes are tagged with 'PK'. The type of target relationship and its multiplicity are determined by the classification of a CDM class C' related to the class C being translated and the properties of each relationship rel defined in C, where $rel \in C.Rel$, e.g., $rel.RelType$, $rel.c$. Each rel is translated into an equivalent target association, aggregation or inheritance. Target relationship names are generated by concatenating $dirC$ with $dirAs$, and $C.cn$ with *invAs*, e.g., `dept_mgr` and `emp_eno` in Figure 4. Relationship cardinality $rel.c$ is mapped into single-valued when $rel.c$:= (0..1 | 1..1) or collection-valued otherwise. The OODB and ORDB schemas corresponding to the CDM in Figure 3 are shown in Figure 4 (ODMG 3.0 ODL) and Figure 5 (Oracle 10_g), respectively. The XML Schema is not provided due to limited space.

4. Data Conversion

The data conversion phase concerns converting existing RDB data to the format defined by the target schema. Data stored as tuples in an RDB are converted into complex objects/literals in object-based databases or elements

```
class emp (extent emps key eno) {
attribute string ename; attribute number eno;
attribute date bdate; attribute string address;
attribute set<struct kids{string kname, string
sex;}> kids_eno;
relationship dept dept_mgr inverse dept::emp_eno;
relationship set<emp> emp_spreno inverse
emp::emp_eno;
relationship dept dept_dno inverse dept::emp_dno;
relationship emp emp_eno inverse emp::emp_spreno
relationship set<proj> proj_pnum inverse
proj::emp_eno;};

class dept (extent depts key dno) {
attribute string dname; attribute number dno;
attribute date startd; attribute set<string>
dept_locations_dno;
relationship set<emp> emp_dno inverse emp::dept_dno;
relationship set<proj> proj_dnum inverse
proj::dept_dno;
relationship emp emp_eno inverse emp::dept_mgr;};
```

Figure 4. Sample Output OODB schema

```
create type kids_t as object(kname char(20), sex
char(1));
create type kids_ntt as table of kids_t;

create or replace type emp_t as object (
ename char(20), eno number, bdate date, address
char(30), dept_mgr ref dept_t, emp_spreno emp_ntt,
kids_eno kids_ntt, proj_pnum proj_ntt, dept_dno ref
dept_t, emp_eno ref emp_t) not final;

create or replace type dept_t as object (
dname char(20), dno number, startd date,
dept_locations_dno dept_locations_ntt, emp_dno emp_ntt,
proj_dnum proj_ntt, emp_eno ref emp_t) not final;

create table hourly_emp of hourly_emp_t
create table dept of dept_t
```

Figure 5. Sample Output ORDB schema

in XML document. Data conversion is performed in three steps: 1) RDB relations' tuples are extracted, 2) these tuples are transformed to match the target format, and 3) the transformed data are loaded into files suitable for bulk loading in order to populate the schema generated during the schema translation phase. Since relationships in object-based databases are reference-based, the process is accomplished in two separate passes. In the first pass, each RDB relations' tuples comprising of non-FK attributes are converted into equivalent target format in order to define objects. In the second pass, the initial object defined in the first pass are linked using FK values extracted from each RDB relation's tuples based on relationships defined in the target schema. User-defined identifers $uoid$s (i.e., surrogate OIDs, which will be translated by the system into physical OIDs during objects loading) for objects in the target database are defined by concatenating class names with the PK values extracted from corresponding RDB tables. Relationships are established using $uoid$s defined from values of CDM relationship attributes, i.e., $dirAs$ and $invAs$. However, relationships among XML elements are established by

key/keyref constraints specified in XML schema document. Each target database's data are generated using a set of data instance conversion rules. We have developed an algorithm for integrating the rules for each target database. The algorithm generates the target data in text files as initial objects files and relationships files. Sets of SQL queries are embedded in these algorithms to extract the desired data from an RDB. At last, a conversion program is generated to enact the schema file obtained from the schema transaction phase and the files generated during the data conversion phase.

Consider the CDM shown in Figure 3 and RDB data given in Figure 2 as input to the algorithm for generating OODB data. One tuple from the salaried_emp RDB table of an employee called "Wallace" is converted, along with related tuples in other tables, into target equivalents. The object definition (in LDB syntax [5]) that represents the RDB tuple is shown in Figure 6(a), whereas its relationships are defined in Figure 6(b).

```
%salaried_emp54321 := persistent hourly_emp (ename:
"Wallace", eno:54321, bdate:'1931-06-20', address:"91
St James Gate NE1 4BB", kids_eno:set(struct( kname:
"Scott", sex:  "M")), salary:43000);
```
(a) definition of salaried_emp54321 object

```
salaried_emp54321->update()->dept_dno.add(dept2);
salaried_emp54321->update()->emp_eno.add
(salaried_emp86655);
salaried_emp54321->update()->proj_pnum.add(proj4,
proj5);
```
(b) relationships among salaried_emp54321 and other objects

Figure 6. Output OODB data

5. Experimental Study

To demonstrate the effectiveness and validity of MIGROX, a prototype has been developed to realize the algorithms outlined in the preceding sections. The algorithms were implemented using Java 1.5 and Oracle 10_g. The experiment was run on a PC with Pentium IV 3.2 GHz CPU and 1024 MB RAM operated under Windows XP Professional. To evaluate scalability and performance of MIGROX, a set of queries have been designed to observe any differences between the source RDB and target databases. Due to limited space, this section presents only two sets of queries for the RDB shown in Figure 2 and one equivalent target database generated by MIGROX (i.e., ORDB). Table 1 shows the description, RDB version, ORDB version and result of each query. The queries are run on Oracle 10_g.

After evaluating the results between the source and the target databases, MIGROX is shown to be feasible, efficient and correct as the queries designed for retrieval operations return identical results. Target databases are generated without loss or redundancy of data. Moreover, many semantics can be converted for RDB into the targets, e.g., association, aggregation and inheritance with integrity constraints enforced on the target database. Some update operations (i.e., insert, delete and modify) are applied on the databases to show that integrity constraints in the RDB are preserved in the target database. However, we cannot cover automatically referential integrity on REFs that are in nested tables in ORDB because Oracle does not have a mechanism to do so. This integrity could be preserved once the schema is generated, e.g., using triggers.

6. Related Work

In recent years, with the growing importance and benefits provided by object-based and XML databases, there has been much effort on migrating RDBs into the relatively newer technologies [1, 13, 3, 10]. Migration of source RDB into object-based and XML databases is accomplished in the literature for only one target database. Existing work can be classified into two categories. The first category, which is called Source-to-Target (ST) technique, translates each construct in a source into an equivalent construct in a target database without using an Intermediate Conceptual Representation (ICR) for semantic enrichment. This technique usually results in ill-designed databases as some of the data semantics are ignored. The second category, which is called Source-to-Conceptual-to-Target (SCT) technique, results in well-designed databases due to the amount of data semantics preserved in a conceptual stage, i.e., ICR.

Inferring the conceptual schema from a logical RDB schema has been extensively studied [1, 11, 6]. Such conversions are usually specified by rules, which describe how to derive RDB's constructs, classify them, and identify relationships among them. Semantic information is extracted by an in-depth analysis of schema, data and queries.

Existing work for migrating RDBs into OODBs focus on schema translation using ST technique [12, 13, 4]. Premerlani and Blaha propose a procedure for mapping an RDB schema into an Object-Modeling Technique (OMT) schema [12]. Fahrner and Vossen propose a method, in which an RDB schema is translated into an ODMG-93 schema [4]. Singh *et al.* propose an algorithm for mapping an RDB schema into an OODB schema based on common attributes factoring [13]. How to map UML models to ORDBs has been studied recently [14, 10], however, the focus has been on the design rather than on migration. Most of research on migrating RDBs to XML are following the SCT technique, focusing on generating a DTD schema and data [3, 15]. Some work (e.g., [7, 3]) use catalogues and assume well-designed RDB whereas some others consider legacy RDBs (e.g., [16]) for migration into XML. Du *et al.* propose a method that employs a model called ORA-SS to support RDBs schema translation into XML Schema [3].

Although well-known conceptual models, e.g., ERM and

Table 1. Results of the Queries

Description	Relational query	Object-relational query	Result
Find the name of department 3	`select dname from dept where dno = 3;`	`select dname from dept where dno = 3;`	`Finance`
Find salaried employees in department 2 who make more than 50000 per year	`select e.ename from emp e, salaried_emp s where e.dno = 2 and e.eno = s.eno and s.salary >= 50000;`	`select s.ename from salaried_emp s where s.dept_dno.dno = 2 and s.salary >= 50000;`	`Borg`
Find all employees working in the Accounts department	`select e.eno, e.ename from emp e, dept d where e.dno = d.dno and d.dname = 'Accounts';`	`select s.column_value.eno, s.column_value.ename from dept d, table(d.emp_dno) s where d.dname = 'Accounts';`	`34534 Scott` `68844 Ali`
Find all employees who have kids named Alice and Michael	`select e.ename from emp e, kids d1, kids d2 where e.eno = d1.eno and e.eno = d2.eno and d1.kname = 'Alice' and d2.kname = 'Michael';`	`select h.ename from hourly_emp h, table(h.kids_eno) d1, table(h.kids_eno) d2 where d1.kname = 'Alice' and d2.kname = 'Michael';`	`Smith`
Display a list of project names that involve an employee called Smith	`select pname from proj p, works_on w, emp e where e.eno = w.eno and w.pno = p.pnum and e.ename = 'Smith';`	`select pname from proj p, table(p.emp_eno) e where e.column_value.ename= 'Smith';`	`Way Station 1` `Way Station 2`

UML may be used as a CDM during database migration, we argue that they do not satisfy the characteristics of more than one target data model, and do not support data representation. Some important semantics have not been considered in these models. For instance, ERM does not support inheritance whereas UML should be extended by adding new stereotypes to specify ORDB and XML model peculiarities [10, 15]. Several ICR models have been developed for specific applications. However, these models are incapable of capturing diverse characteristics of the three target models. The SOT model [2] has been designed only for migrating RDBs into OODBs whereas the ORA-SS model [3] has been designed to support semi-structured data models.

7. Conclusion

This paper contributes a solution to RDB migration, which is superior to the existing proposals as it can produce three different output databases. A system architecture has been designed and a prototype implemented, which generated successfully the target databases. The approach has been evaluated by comparing query results. We have designed several experiments that involve running queries on a source RDB and one target database, which is generated by the prototype. We have analysed the results of queries obtained from both databases and found that both set of results were identical. Therefore, we conclude that the source and target databases are equivalent. Moreover, the results obtained demonstrate that the MIGROX solution, conceptually and practically, is feasible, efficient and correct. Our future research focus is on data specific manipulation (e.g., update/query) translations and further prototyping to simplify relationship names that are automatically generated.

References

[1] R. Alhajj. Extracting the extended entity-relationship model from a legacy relational database. *Info. Syst.*, 28(6):597–618, 2003.

[2] A. Behm, A. Geppert, and K. R. Dittrich. Algebraic database migration to object technology. In *ER*, pages 440–453, 2000.

[3] W. Du, M.-L. Lee, and T. W. Ling. XML structures for relational data. In *WISE*, volume 1, 2001.

[4] C. Fahrner and G. Vossen. Transforming relational database schemas into object-oriented schemas according to ODMG-93. In *DOOD'95*, pages 429–446. Springer-Verlag, 1995.

[5] L. Fegaras, C. Srinivasan, A. Rajendran, and D. Maier. lambda-db: An ODMG-based object-oriented DBMS. In *SIGMOD Conference*, page 583, 2000.

[6] M. M. Fonkam and W. A. Gray. An approach to eliciting the semantics of relational databases. In *4th Int. Conf. Advanced Info. Syst. Eng.*, volume 593, pages 463–480, 1992.

[7] D. Lee, M. Mani, F. Chiu, and W. W. Chu. NeT and CoT: Translating relational schemas to XML schemas using semantic constraints. In *CIKM*, pages 282–291, 2002.

[8] A. Maatuk, A. Ali, and N. Rossiter. A framework for relational database migration. Technical report, http:// computing.unn.ac.uk/staff/cgma2/papers/ RDBM.pdf, 2008.

[9] A. Maatuk, A. Ali, and N. Rossiter. Relational database migration: A perspective. In *DEXA*, volume 5181, pages 676–683. Springer-Verlag, 2008.

[10] E. Marcos, B. Vela, and J. M. Cavero. A methodological approach for object-relational database design using UML. *Soft. and Syst. Model.*, 2(1):59–75, 2003.

[11] J.-M. Petit, J. Kouloumdjian, J.-F. Boulicaut, and F. Toumani. Using queries to improve database reverse engineering. In *ER '94*, pages 369–386. Springer-Verlag, 1994.

[12] W. J. Premerlani and M. R. Blaha. An approach for reverse engineering of relational databases. *Communications of the ACM*, 37(5):42–49, 1994.

[13] A. Singh, K. S. Kahlon, J. Singh, R. Singh, S. Sharma, and D. Kaur. Mapping relational database schema to object-oriented database schema. In *Int. Conf. on Computational Intelligence*, pages 153–155, 2004.

[14] S. D. Urban, S. W. Dietrich, and P. Tapia. *Succeeding with Object Databases*, chapter Mapping UML Diagrams to Object-Relational Schemas in Oracle 8, pages 29–51. John Wiley and Sons, Ltd, 2001.

[15] B. Vela and E. Marcos. Extending UML to represent XML schemas. In *CAiSE Short Paper Proceedings*, 2003.

[16] C. Wang, A. Lo, R. Alhajj, and K. Barker. Converting legacy relational database into XML database through reverse engineering. In *ICEIS*, volume 1, pages 216–221, 2004.