A Framework for Relational Database Migration

Abdelsalam Maatuk, Akhtar Ali, and Nick Rossiter

Abstract—The dominance of traditional Relational DataBases (RDBs) and their limitation to support complex structure and user-defined data types provided by object-based/XML databases makes migrating an RDB into object-oriented databases, object-relational database and XML an active research area. The problem is how to effectively migrate existing RDBs, as a source, into OODB/ORDB/XML, as targets, and what is the best way to enrich and maintain RDBs’ semantics and constraints in order to meet the characteristics of the three targets? Existing work does not appear to provide a complete solution for more than one target database. We tackle this question by proposing a solution for migrating an RDB into the three targets based on available standards. The solution takes an existing RDB as input, enriches its meta data representation with as much semantics as possible, and constructs an enhanced Relational Schema Representation (RSR). Based on the RSR, a Canonical Data Model (CDM) is generated, which captures essential characteristics of target data models suitable for migration. A prototype has been implemented, which migrates a CDM/RDB into object-oriented (ODBG 3.0), object-relational (Oracle 10g) and XML databases.

I. 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. Instead of discarding the previous RDB or build non-relational applications on it, it is accepted to convert the old data and applications together 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 an integrated method that deals with DataBase Migration (DBM) from RDB to Object-Oriented DataBase (OODB)/Object-Relational DataBase (ORDB)/XML in order to provide an opportunity for exploration, experimentation and comparison among the alternative databases. 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 more than one database models and these are conceptually different. There is a lack of a canonical model that can be used as an intermediate stage for schema and data conversion from input (RDB) to various output targets.

Different researches are dedicated in RDB migrations focusing on different areas. Most existing proposals are restricted by a range of assumptions such as a source schema is required to be available for further normalisation to 3rd Normal Form (3NF) before the DBM process can begin [7]. Key-based inclusion dependency is assumed in many proposals with key attribute consistencies and another frequent assumption is that the initial schema is well designed [7], [10], [19]. Earlier models such as Entity Relationship Model (ERM), Extended ERM (EERM) and Object-Modeling Technique (OMT) are assumed in most works as an Intermediate Conceptual Representation (ICR) or target data models [12], [14] whereas others are restricted to a particular product, e.g., Oracle [11]. Several previous approaches fail to maintain all of the data semantics explicitly, e.g., integrity constraints and inheritance, whereas constraints are mapped into class methods [7] or into separate constraint classes [12]. For proof of concept, most proposals have been implemented in one way or another. Only a few attempt data conversion and even those that do have some drawbacks. However, the existing work does not provide a solution for more than one target database or 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 MIGrating an RDB into Object-based and XML 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 lose or unnecessary redundancies. The method is superior to the existing proposals as it can produce three different output databases as shown in Fig 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 data models, we believe that it is necessary to develop a Canonical Data Model (CDM) to bridge the semantic gap between them and to facilitate the migration process. The CDM should be able to preserve and enhance RDB’s integrity constraints and data semantics to fit in with target database characteristics. MIGROX has three phases: Semantic enrichment, schema translation and data conversion. In the 1st phase, the method produces a CDM, which is enriched with an RDB’s constraints and data semantics that may not have been explicitly expressed in it. The CDM so obtained is mapped into target schemas in the 2nd phase. The 3rd phase converts an RDB data into its equivalents in the new database environment. System architecture has been designed and a prototype implemented to demonstrate the process, which results successfully in target databases.

This paper is structured as follows. Section II provides an introduction to the semantic enrichment phase. An overview of the schema translation phase is introduced in Section III. Section IV presents the data conversion phase. Section V reviews some results of the MIGROX prototype. A general overview of the related work is presented in Sec-
A. Extracting Relational Schema Representation (RSR)

In order to produce an integrated source of database semantic information for the purpose of SE, implicit semantics have to be made explicit. Conflicts in naming have to be resolved, and attributes and interrelationships amongst data have to be deduced. In this section, we introduce an RSR, as a representation of an RDB’s metadata, to be used as a source of information for CDM generation. Basic information needed to proceed with the SE phase includes relation names and attributes’ properties that include attribute names, data types, max length, default values, and whether the attribute is nullable. Moreover, the most important information needed is about the keys including Unique Keys (UKs). We assume that data dependencies are represented by Primary Key (PKs) and Foreign Key (FKs) as for each FK value(s) there are an existing, matched PK value, which can be considered as a value reference. The inverse of an FK is called an Exported Key (EK). EKs play an important role as regards to OO/OR databases, which support bi-directional relationships. The user interaction might be necessary to provide any missing semantics.

Definition 1: In our approach an RDB schema is represented as a set of elements,

\[ RSR := \{ R | R := \langle n, A_{rsr}, PK, FK, EK, UK \rangle \} \]

where:

- \( r_n \) denotes the name of \( R \).
- \( A_{rsr} \) denotes the set of attributes of \( R \): \( A_{rsr} := \{ a | a := \langle a_n, t, l, n, d \rangle \} \), where \( a_n \) is an attribute name, \( t \) is its type, \( l \) is data length, \( n \) is nullable or not (‘\( y \)’ or ‘\( n \)’), and \( d \) is a default value if given.
- \( PK \) denotes \( R \)’s primary key: \( PK := \{ a | a := \langle pa, s \rangle \} \), where \( pa \) is an attribute name and \( s \) is a sequence number in the case of a composite key, however, \( s \) is assigned 1 in the case of a single valued key.
- \( FK \) denotes the set of foreign key(s) of \( R \): \( FK := \{ \beta | \beta := \langle er, \{ \langle fa, s \rangle \} \} \} \), where \( \beta \) represents one FK (whether it is single or composite), \( er \) is the name of an exporting (i.e., referenced) relation that contains the referenced FK, \( fa \) is a foreign key attribute name, and \( s \) is a sequence number.
- \( EK \) is a set of exported key(s) of \( R \): \( EK := \{ \gamma | \gamma := \langle ir, \{ \langle ea, s \rangle \} \} \} \), where \( \gamma \) represents one EK, \( ir \) is the name of an importing (i.e., referencing) relation that contains the exported attribute name \( ea \) (i.e., FK attribute).
- \( UK \) is a set of unique keys of \( R \): \( UK := \{ \delta | \delta := \langle ua, \} \).
The CDM definition, target attributes that represent relationships among classes are materialized into references or changed into other domain.

In this study the CDM is designed to upgrade the semantic level of RDB and to play the role of an intermediate stage for DBM from RDB to OODB/ORDB/XML acting on both levels: schema translation and data conversion. It represents explicit as well as implicit semantics of an existing RDB. Explicit semantic include relation and attribute names, keys, etc.; implicit semantic include classification of classes and attributes, and relationship names, types, cardinalities and inverse relationships. Its constructs are classified to facilitate the translation to complex target objects in reasonable way instead of flat one to one and complicated clustering conversions. Through the CDM, target databases can be obtained well-structured without proliferation of references and without unnecessary 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. However, 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 object oriented concepts with rich semantics that cater for OR and XML data models. However, the CDM is independent of an RDB from which it has taken semantics as well as any of the target databases to which it could be converted. It is enriched by semantics from an RDB model such as PKs, FKs, attributes max length, uniqueness, etc. In order to express as much semantics as possible, the model has taken into consideration features that are provided by object-based and XML models such as association, aggregation and inheritance. It provides non-OODB key concepts (i.e., FKs, null and UKs) and explicitly specifies that the attributes and cardinalities are optional or required. Relationships are defined in CDM in a way, which facilitates the extracting and transforming of data in the data conversion phase including defining and linking objects using user defined identifiers. Real world entities, multi-valued and composite attributes, and relationships are all represented as classes in CDM.

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

\[ CDM := \{ C \mid C := (cn, cls, abs, A_{cdm}, Rel, UK)_C \} \]

where:

- Each class \( C \) has a name \( cn \), given a classification \( cls \) and whether it is abstract \( abs \) 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, where:

\[ cls := \text{Regular Strong Class (RST)} \mid \text{Secondary (inherited) Strong Class (SST)} \mid \text{Subclass (SUB)} \mid \text{Secondary (inherited) Subclass (SSC)} \mid \text{Regular (M N relationship class without attributes)} \]

Relationship Class
### TABLE I
Result of RSR construction

<table>
<thead>
<tr>
<th>r</th>
<th>$A_{cdm}$</th>
<th>PK</th>
<th>FK</th>
<th>EK</th>
<th>UK</th>
</tr>
</thead>
<tbody>
<tr>
<td>emp</td>
<td>eno</td>
<td>int</td>
<td>25</td>
<td>n</td>
<td>eno</td>
</tr>
<tr>
<td>e-name</td>
<td>char</td>
<td>40</td>
<td>n</td>
<td>y</td>
<td>dept</td>
</tr>
<tr>
<td>address</td>
<td>char</td>
<td>40</td>
<td>y</td>
<td>dept</td>
<td>mgr</td>
</tr>
<tr>
<td>salario_emp</td>
<td>salario</td>
<td>int</td>
<td>25</td>
<td>y</td>
<td>de</td>
</tr>
<tr>
<td>dept</td>
<td>dno</td>
<td>int</td>
<td>25</td>
<td>n</td>
<td>dno</td>
</tr>
<tr>
<td>dname</td>
<td>char</td>
<td>40</td>
<td>n</td>
<td>dnum</td>
<td>1</td>
</tr>
<tr>
<td>mgr</td>
<td>int</td>
<td>25</td>
<td>n</td>
<td>dept_locations</td>
<td>dno</td>
</tr>
<tr>
<td>startd</td>
<td>date</td>
<td>y</td>
<td>proj</td>
<td>dno</td>
<td>1</td>
</tr>
<tr>
<td>works_on</td>
<td>eno</td>
<td>int</td>
<td>25</td>
<td>n</td>
<td>pino</td>
</tr>
<tr>
<td>pno</td>
<td>int</td>
<td>25</td>
<td>n</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(RRC) | Secondary Relationship Class (SRC), i.e., referenced RRC, M:N relationship with attributes or n-ary relationships, n>=2 | Multi-valued Attribute Class (MAC) | Composite Attribute Class (CAC) | Regular Component (in relationship with other classes rather than its whole) Class (RCC).

- **Abstraction (ab):** It is important for a superclass to check whether all of its objects are inherited or not. A superclass is abstract (i.e., $ab := \text{true}$) when all its objects are members of its subtype objects. Instances of an abstract type cannot appear in database extension but are subsumed into or by instances of its subtypes. A class is not abstract (i.e., $ab := \text{false}$) 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.

\[
A_{cdm} := \{a | a := \langle a_n, t, tag, l, n, d \rangle\},
\]

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 primary and foreign key ‘PF’ attribute. Each $a$ can have a maximum length $l$, may have a default $d$ value whereas $n$ indicates that $a$ is nullable or not (‘y’’|’n’).

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

\[
Rel := \{rel | rel := \langle RelType, dirC, dirAs, c, invAs\rangle\},
\]

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, invAs denotes a set of inverse attribute names representing the inverse relationship from $C$ side, and $c$ is a relationship cardinality constraint. $RelType$ 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 (or examining) data in a complete database is used to extract cardinality constraints. Querying data instances may not return the semantics of an existing RDB if data is incomplete as different database states give different cardinalities. Based on $c$, $C'$’s object(s) can be single-valued or set-valued.

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. That is a relation $R$ is a strong relation if its PK is not fully or partially composed of any FKs. Similarly, $R$ is a weak relation if its PK is partially composed of a PK of another strong relation. In addition, in EERM, an inheritance relationship is represented using generalization/specialization, which have different types of specification. However, such types of specialization can not be represented directly in relational data models. There are many alternative ways to model inheritance in relational data models [5]. The most common alternative represents inheritance as one relation for a superclass and one relation for every subclass. The superclass is represented by a relation $R$ with its key and attributes, where $R(pk, a_1, ..., a_n)$, $A(R) := \{pk, a_1, ..., a_n\}$ and $PK(R) := pk$. Each subclass $S$ with its attributes is represented by relation $S(pk, attributes of S)$ and $PK(S) := pk$. MIGROX assumes this alternative because it is based on PKs/FKs matching, and without user involvement it would not be possible to automatically identify other alternatives of inheritance. The main idea in inheritance is that a supertype is inherited by one or more other subtypes. A subtype can be inherited by other subtypes. Moreover, a subtype may inherit from more than one other supertypes, i.e., multiple-inheritance. However, as ODMG 3.0, SQL4 and XML Schema do not allow a concrete subtype to have more than one concrete supertype, multiple-inheritance is not supported in MIGROX. Therefore, each superclass in CDM can be inher-
ited by one or more subclasses, but a subclass can have only one superclass.

Consider the RSR shown in Table I, Fig 4 shows the resulting CDM, generated from the RSR and RDB in an easy to follow format hiding the deeply nested structure of CDM classes. The CDM shows only EMP and DEPT classes. For instance, the CDM’s class, EMP, which is SST, 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 HOUURLY_EMP classes. Cardinalities are determined for each class. Relationships defined in each class as: RetType (inv:As ←→ dirC(dirAs)), (←→ indicates bidirectional association and → indicates unidirectional aggradation).

**III. Schema Translation**

The Schema Translation phase aims at translating CDM, produced from the semantics enrichment phase, into its equivalent targets schemas as shown in Fig 5. Target schemas hold equivalent semantics to that of an existing RDB, which are enhanced and preserved in CDM. Three sets of translation rules are designed for mapping CDM into target schemas. Algorithms are developed for producing each target schema according to these rules. In this section, we define target schemas, which satisfy ODMG 3.0, SQL4 and XML Schema standards and introduce the schema translation phase.

**A. Target Models**

We first briefly define the output target models for the ST phase. Translating these models defined here to the actual schema definition languages is straightforward.

**Definition 3:** A target ODMG 3.0 schema is defined as a set of classes,

\[
\text{OODM} := \{ C_{oo} | C_{oo} := \langle c_n, \text{spr}, k, \text{A}_{oo}, \text{Rel}_{oo} \rangle \},
\]

where \(c_n\) is a name of a class \(C_{oo}\), \(\text{spr}\) is the name of its superclass, \(k\) is its primary key, \(\text{A}_{oo}\) is a set of its attributes of simple or complex data type and \(\text{Rel}_{oo}\) is a set of relationship types in which \(C_{oo}\) participates. The sets \(A_{oo}\) and \(\text{Rel}_{oo}\) are defined as follows:

- \(A_{oo} := \{a_{oo} | a_{oo} := \langle a_n, t, m \rangle\}, \) where \(a_n\) is a name of an attribute \(a_{oo}\), \(t\) is its data type, which can be primitive (e.g., string), user-defined constructed (e.g., struct) or object-based (e.g., class), and \(m\) denotes whether \(a_{oo}\) is single-valued (‘sv’) or collection-valued (‘cv’); and
- \(\text{Rel}_{oo} := \{\text{rel}_c | \text{rel}_c := \langle \text{rel}_n, \text{dirC}_n, m, \text{invRel}_n \rangle\}, \) where \(\text{rel}_n\) is the name of the relationship \(\text{rel}_c\), \(\text{dirC}_n\) is the name of the referenced class, and \(\text{invRel}_n\) is the name of the inverse relationship.

**Definition 4:** A target SQL4 ORDB schema is represented as \(\text{ORSchema} := \{ \text{UT}, \text{TT}, \text{UK}_t \}, \) where \(\text{UT}\) is a set of User Defined Types (UDTs), \(\text{TT}\) is a set of typed tables and \(\text{UK}_t\) is a set of UKs. The sets \(\text{UT}\) and \(\text{TT}\) are defined as follows:

- \(\text{UT} := \{ \text{udt} | \text{udt} := \langle \text{ut}_n, \text{sut}_t, \text{A}_{ut}, \text{uuid} \rangle\}, \) where \(\text{ut}_n\) is the name of the type \(\text{u dt}, \text{s ut}_t\) is the supertype name of \(\text{ud t}, \text{A}_{ut}\) is a set of attributes and \(\text{uuid}\) is a user defined identifier of \(\text{ud t};\)
- \(\text{A}_{ut} := \{ \text{a}_{ut} | \text{a}_{ut} := \langle \text{a}_n, \text{t}, \text{m}, \text{n}, \text{d} \rangle\}, \) where \(\text{a}_n\) is a name of an attribute \(\text{a}_{ut}\), \(\text{t}\) is its data type, which can have primitive, user-defined constructed or ref-based, \(m\) denotes whether \(\text{a}_{ut}\) is a single-valued (‘sv’) or a collection-valued (‘cv’), \(d\) is a default value in case of primitive and \(n\) denotes whether \(\text{a}_{ut}\) accepts nulls or not; and
- \(\text{TT} := \{ \text{tt} | \text{tt} := \langle \text{tt}_n, \text{ut}_n, \text{s ut}_t, \text{pk} \rangle\}, \) where \(\text{tt}_n\) is the name of a typed table \(\text{tt}, \text{ut}_n\) is a \(\text{tt}\)’s name that \(\text{tt}\) is defined based upon, \(\text{s ut}_t\) is its supertype’s name and \(\text{pk}\) is the PK of \(\text{tt}\).

**Definition 5:** A target XML Schema is represented as \(\text{XMLSchema} := \{ \text{Root},\text{GT} \}, \) where:

- \(\text{Root}\) is a global element declared under the schema with its direct local subelements and constraints representing the XML document tree, and \(\text{GT}\) is a set consisting of global complex types, which are defined to be referenced as types of subelements that are declared within the \(\text{Root}\) or by other defined global complex types. The \(\text{Root}\) and the set \(\text{GT}\) are defined as follows:
  - \(\text{Root} := \{ \text{root}_n, \text{RT}, \text{PK}_x, \text{FK}_x, \text{UK}_x \}, \) where \(\text{Root}\) has a name \(\text{root}_n\), a type \(\text{RT}\) and three sets of identity-constraints \(\text{PK}_x, \text{FK}_x\) and \(\text{UK}_x:\)
    - \(\text{RT}\) represents the \(\text{Root}\)’s local complex type that involves a set of local sub-element \(\text{le}\) declarations:
      \(\text{RT} := \{ \text{le} | \text{le} := \langle \text{e}_n, \text{et}, \text{min}, \text{max} \rangle\}, \) where \(\text{e}_n\) is the name of \(\text{le}, \text{et}\) is its type, and \(\text{min}\) and \(\text{max}\) are its minimum and maximum occurrences. The type of each subelement \(\text{et}\) is defined globally under the schema in the set \(\text{GT}\);
- PKx is a set of primary keys for subelements defined in the Root: PKx := \{pk | pk := \{pk_n, selector, PKfield\}\}. Each primary key has a constraint name pk_n, element set selector as scope for the key to be defined in, and a set of related sub-elements that are selected to be unique PKfield;

- FKx is a set of foreign keys, where FK := \{fk | fk := \{fk_n, ref, selector, FKfield\}\}. Each foreign key has a constraint name fk_n, an element set selector, a reference constrains name refer that refer to a matched PK constrain name, and a set of related subelements FKfield;

- UKx is a set of unique keys, where UK := \{uk | uk := \{uk_n, selector, UKfield\}\}. Each unique key has a constraint name uk_n, element set scope selector, and a set of related subelements selected to be unique UKfield; and

- GT := \{CT | CT := \{ct_n, base, LE\}\}, where ct_n is the name of a complex type CT, base is its a supertype’s name (if it is derived from other type), and LE is a set of elements that declared locally within CT. LE := \{le | le := \{en_n, et, nim, maze\}\} as defined as for Root’s type; however, data type of local elements le ∈ LE can be built-in data type (e.g., string) or predefined complex type (e.g., dependent complex type).

B. Translation Process

Given a CDM, the schema translation phase 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. Based on cls, each main CDM class C is translated into target type, where C.cls ≠ (“MAC” | “CAC” | “RRC”). The type is defined under its superclass if C.cls := “SUB” or “SSC”. However, MAC and CAC classes are mapped into multi-valued and composite (e.g., struct) attributes respectively, and RRC classes are mapped into an M:N relationship in which a pair of 1:M relationship is defined in each of the target types that participate in the relationship. Attributes C.Acdn 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 ∈ 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 Fig 6. 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 Fig 4 are shown in Fig 6 (ODMG 3.0 ODL) and Fig 7 (Oracle 10g), respectively. The XML Schema is provided in Fig 8.

\[
\text{class emp (extent emps, key eno)} \{ \\
\text{attribute string ename; attribute number eno; } \\
\text{attribute date bdate; attribute string address; } \\
\text{attribute set<struct kids(string kname, char sex;}) \text{ kids_eno; } \\
\text{relationship dept dept mgr inverse dept::emp eno; } \\
\text{relationship set<emp> emp_spreno inverse emp::emp eno; } \\
\text{relationship dept dept dno inverse dept::emp dno; } \\
\text{relationship emp emp eno inverse emp::emp_spreno } \\
\text{relationship set<proj> proj dnum inverse proj::emp eno; } \\
\text{class hourly_emp extends emp (extent hourly emps{attribute number pay scale;}); } \\
\text{class salaried_emp extends emp (extent salaried_emps{attribute number salary;}); } \\
\text{class dept (extent depts, key dno) { } \\
\text{attribute string dname; attribute number dno; } \\
\text{attribute date startd; attribute set<string> dept_locations dno; } \\
\text{relationship set<emp> emp dno inverse emp::dept dno; } \\
\text{relationship set<proj> proj dnum inverse proj::dept dno; } \\
\text{relationship emp emp dno inverse emp::dept mgr; } \\
\text{class proj (extent proj s, key ps) { } \\
\text{attribute string psname; attribute number psnum; attribute string location; } \\
\text{relationship set<emp> emp eno inverse emp::proj psnum; } \\
\text{relationship dept dept dno inverse dept::proj dnum;}; }
\]

Fig. 6. Sample Output OODB schema

IV. 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 in XML document. We propose using CDM to guide the conversion process. Data conversion is performed in three steps as shown in Fig 9. Firstly, the RDB relations’ tuples are extracted. Secondly, these data are transformed (converted) to match the target format. Finally, the transformed data are loaded into text files suitable for bulk loading in order to populate the schema generated earlier during the ST 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. Objects’ user-defined identifiers uoids are extracted by concatenating the class name with the PK data extracted from corresponding RDB table. Similarly, object-based relationships are established using uoids extracted from CDM relationship attributes, i.e., dirAs and invAs data. 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 customised SQL queries are embedded in these al-

\footnote{i.e., surrogate OID, which will be translated by the system into a physical OID during object loading}
V. Experimental Study

To demonstrate the effectiveness and validity of MIGROX, a prototype has been developed to realize the algorithms outlines in proceeding sections. The algorithms were implemented using Java 1.5 software development kit and Oracle 10g. The experiment was run on a PC with CPU Pentium IV 3.2 GHz and RAM 1024 MB operated under Windows XP Professional. We used the JDBC API to establish a connection with RDBMS that hold the source RDB. 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 Fig 3 and one equivalent target database generated by MIGROX (i.e., ORDB). For each query we give a description, an RDB Version (R-Q), and an ORDB Version (OR-Q) and the result of the query. The queries are run on Oracle 10g to check whether the results are the same or not.

1. Find the name of a department with dno = 4.
   R-Q/OR-Q: select s.ename from salaried s, dept d where s.deptno = d.dno and s.deptno = 4 and s.salary > 40000; Result: Finance

2. Find names of salaried employees in department 2 who make more than 40000 per year.
   R-Q: select e.ename from emp e, salaried_emp s where e.dno = 2 and e.eno = s.eno and s.salary > 40000; OR-Q: select e.ename from salaried_emp s where s.dept_dno.dno = 2 and s.salary > 40000; Result: Borg

Fig. 7. Sample Output ORDB schema

Fig. 8. Sample Output XML Schema
Fig. 9. Schematic view of converting relational data into targets

(a) definition of salaried_emp54321 object

(b) relationships among salaried_emp54321 and other objects

Fig. 10. Output OODB data

Wallace

3. Find all employees working in 'Accounts'.
   R-Q: select e.eno, e.ename from emp e, dept d where e.dno = d.dno and d.dname = 'Accounts';
   OR-Q: select st.column_value.eno, st.column_value.ename from dept d, table(d.emp_dno) st where d.dname = 'Accounts';
   Result:
   34534  Scott
   68844  Ali

4. Find all employees who have kids named “Alice” and “Michael”.
   R-Q: 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’;
   OR-Q: 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’;
   Result: Smith

5. Display a list of project names that involve an employee whose name is “Smith”.
   R-Q: select pname from proj p, works_on w, emp e where e.eno = w.eno and w.pno = p.pnum and e.ename = ‘Smith’;
   OR-Q: select pname from proj p, table(p.emp_eno) e where e.column_value.ename= ‘Smith’;
   Result:
   Way Station 1
   Way Station 2

After evaluating the results between the source and the target database, MIGROX is shown to be feasible and efficient 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 to the target databases. 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 databases. However, we cannot cover automatically referential integrity constraints on REFs that are in nested tables in ORDB because Oracle3 does not have a mechanism to do so. This integrity could be preserved manually once the schema is generated, e.g., using triggers.

VI. Related Work

In recent years, with the growing importance and benefits provided by object-based and XML databases, there has been a lot of effort on migrating RDBs into the relatively newer technologies [1], [15], [8], [10]. Migration of source RDB into object-based and XML databases is accomplished in the literature for only one target database (e.g., OODD, ORDB or XML). Existing work can be classified into two categories. The first category, which is called Source-to-Target (ST), 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), results in well-designed databases due to the amount of data semantics preserved in a conceptual intermediate stage, i.e., ICR.

Inferring conceptual schema from a logical RDB schema has been extensively studied [1], [13], [9]. Such conversions are usually specified by rules, which describe how to derive RDB’s constructs (e.g., relations, keys), classify them, and identify relationships among them. Semantic information is extracted by an in-depth analysis of schema, data and queries. Fonkam and Gray present an algorithm for converting RDB schemas into conceptual models [9]. Al-haj proposes semantics extraction by analysing data instances [1]. Petit et al. present an approach to extract EERM constructs from an RDB by analysing SQL queries in application programs [13].

Existing work for migrating RDBs into OODBs focus on schema translation using ST techniques [14], [15], [7]. Premelani and Blaha propose a procedure for mapping an RDB schema into an OMT schema [14]. Fahrner and Vossen propose a method, in which an RDB schema is normalised to 3NF, enriched by semantics using data dependencies and translated into an ODMG-93 schema [7]. Singh et al. propose an algorithm for mapping an RDB schema into a corresponding OODB schema based on common attributes factoring [15]. However, constraints are not considered in their approach. Behm et al. propose a model, called Semi Object Type (SOT), to facilitate restructuring of schemas during translating an RDB into an OODB [2]. An RDB schema is mapped into SOT schema, which is

3 http://download.oracle.com/docs/cd/B10501_01/appdev.920/a06594/title.htm
then redesigned and converted into an OODB schema. How to map UML models to ORDBs has been studied not long ago (e.g., [16], [11]), however, the focus has been on the design of ORDBs rather than on migration. Most of existing research on migrating RDBs to XML are following the SCT technique, focusing on generating a DTD schema and data [3], [4], [17]. Some existing work (e.g., [6], [4]) use data dictionaries and assume well-designed RDB (e.g., in 3NF) whereas some others consider legacy RDB (e.g., [18]) for migration into XML documents. Du et al. propose a method that employs a model called ORA-SS to support the translation of RDB schema into XML Schema [4].

Although known conceptual models, e.g., ERM and UML may be used as a CDM during DBM, we argue that they do not satisfy the characteristics and constructs of more than one target data model, and do not support data representation. Some important semantics have not been considered in these conceptual models. For instance, ERM does not support inheritance whereas UML should be extended by adding new stereotypes or other constructs to specify ORDB and XML models peculiarities [11], [17]. Several ICR models have been developed for specific applications. However, these models are incapable of capturing diverse characteristics of the three target data models. The SOT model [2] has been designed only for migrating RDBs into OODBs whereas the ORA-SS model [4] has been designed to support semi-structured data models.

VII. Conclusion

This paper contributes a solution to RDB migration, which is superior to the existing proposals as it can produce three different output databases. Besides, it exploits the ranges of powerful features that target data models provide such as ODMG 3.0, SQL4, and XML Schema. A system architecture is designed and a prototype has been 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 our prototype. We have analysed the results of queries obtained from both databases and found that both set of results were identical. Therefore, we concluded that the source and target databases are equivalents. 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

[19] Zhang, X., Zhang, Y., Fong, J. and Jia, X.: Translating RDB Schemas into Object-Oriented Schemas According to ODMG-93. Schemas into Object-Oriented Schemas According to ODMG-93.