Re-engineering Relational Databases: The Way Forward

Abdelsalam Maatuk
Faculty of Sciences,
Department of Computer,
Omar Al-Mukhtar University,
Libya
ammaatuk@yahoo.com

M. Akhtar Ali
School of Computing,
Engineering & Information
Sciences, Northumbria
University, UK
akhtar.ali@unn.ac.uk

Nick Rossiter
School of Computing,
Engineering & Information
Sciences, Northumbria
University, UK
nick.rossiter@unn.ac.uk

ABSTRACT

This paper surveys the recent literature about various research trends relevant to Relational DataBase (RDB) re-engineering. The paper presents an analysis of approaches and techniques used in this context, including construction of object views on top of RDBs, database integration and database migration. A categorisation is presented of the selected work, concentrating on migrating an RDB as a source into object-based and XML databases as targets. Database migration from the source into each of the targets is discussed and critically evaluated, including the semantic enrichment, schema translation and data conversion. Based on a detailed analysis of the existing literature, it seems that the existing work does not provide a complete solution for more than one target database for either schema or data conversion. Besides, none of the existing proposals can be considered as a method for migrating an RDB into an object-relational database. We propose such a method based on an intermediate canonical data model, which enriches the semantics of the source RDB and captures characteristics of the target databases.

Keywords

Re-engineering databases; database migration; semantic enrichment; schema translation; data conversion

1. INTRODUCTION

The increasing popularity of new object-based and World Wide Web (WWW) technologies and non-traditional applications (e.g., multimedia, computer aided design) are considered to be among the most significant recent changes in information technology. These novel technologies have been dominant in the area of information systems due to their productivity, flexibility and extensibility. Object-Oriented DataBase (OODB), Object-Relational DataBase (ORDB) and eXtensible Markup Language (XML), all which support various diverse concepts, have been proposed in order to fulfil the demands of newer and more complex applications. However, as the majority of today's data are still stored in Relational DataBases (RDBs), therefore it is expected that the need to convert such RDBs into the technologies that have emerged recently will grow substantially [18, 25]. Numerous methods have been proposed in the past for re-engineering RDBs into the newer databases. This paper provides a survey of this literature.

We focus on the case where the input is an RDB and the outputs are OODB, ORDB and XML. Hence, we do not cover the inverse of the process (e.g., from OODB into RDB). Three aspects of re-engineering have been discussed: semantics enrichment, schema translation and data conversion. There are three approaches used to accomplish database re-engineering: 1) viewing objects on top of RDBs where data is processed in object/XML form and stored in relational form; 2) database integration where a gateway is used on top of multiple heterogeneous databases to support a single view; and 3) database migration where an RDB is migrated into its equivalents. On the other hand, translation techniques are divided into two categories: i) source-to-target translation, in which a source database is translated directly into a target database, and ii) source-to-conceptual-to-target translation, in which a source schema is enriched by semantics or recovered to a conceptual schema before being translated into a target schema. The source-to-target translation includes flat, clustering and nesting translation. The proposals for RDB re-engineering in the literature have been discussed in separate categories according to the different target databases. Within each category, existing proposals have been compared in terms of translation techniques, prerequisites, and specific features. The aims have been to provide a comprehensive view of the problem of RDB migration, to review various techniques and proposals, to identify their commonalities and differences, to assess the impact of pervious research, and to show how it has shaped current and future research in this area.

The remainder of this paper is organised as follows. Section 2 provides a brief introduction to the database re-engineering process. Section 3 reviews current approaches and techniques related to database re-engineering. Section 4 gives an overview of proposals for RDB migration. Section 5 presents a review of existing proposals for migrating RDBs into OODBs, and work on mapping RDBs into ORDBs is reviewed in Section 6. Section 7 then provides a review of work on migrating RDBs into XML. Section 8 concludes the paper.

2. DATABASE RE-ENGINEERING

Database application re-engineering is a process in which all components (i.e., schema, data, application programs, queries and update operations) of a source database application are converted into their equivalents in a target database environment. However, application programs and queries conversion is a software engineering job and is, therefore, out of the scope of this paper, i.e., we assume that database re-engineering (or migration) includes schema translation and data conversion.

Schema Translation: A schema of an existing data model can be translated into an equivalent target schema expressed in the target data model through applying a set of mapping rules [41]. The generation of a well-designed target schema depends on the flexibility of these rules. Each rule maps a specific construct, e.g., attribute or relationship. Both schemas should hold equivalent semantics. The translation of a source schema to a target schema consists of two steps. The first step, called DataBase Reverse Engineering (DBRE), aims to recover the conceptual schema, e.g., an Entity Relationship (ER) model, which expresses the explicit and implicit data semantics of the source schema. Explicit semantics involve relations, attributes, keys and data dependencies. It is necessary to extract extra semantics that are not expressed explicitly in RDBs (e.g., inheritance relationship, cardinality constraints, relationship names). The second step, called DataBase Forward Engineering (DBFE), aims to obtain the target physical schema from the conceptual schema obtained in the first step. The first step is generally known as the semantic enrichment process, which is essential for database migration and database integration [29]. However, the source schema can be translated directly to a target one without intermediate representation [19]. An expert user or a tool might be required to provide the missing semantics or to refine the results to exploit the concepts of the target database [39, 19].

Data Conversion: This is a process for converting data instances from the source database into the target database. Data stored as tuples in an RDB are converted into complex objects/literals in object-based databases or elements in XML documents. This involves extracting and restructuring RDB data, and then reloading the converted data into a target database in order to populate the schema generated earlier during the schema translation process [22].

3. APPROACHES AND TECHNIQUES

This section introduces approaches and techniques related to database re-engineering. Section 3.1 discusses approaches to database conversion whereas Section 3.2 discusses existing translation techniques.

3.1 Conversion Approaches

There are three approaches related to database re-engineering. The first approach is for handling data stored in RDBs through Object-Oriented (OO)/XML interfaces. Connecting an existing RDB to a conceptually different database system is the basis of the second approach, and the third approach is to migrate an RDB into a target database. The first and second approaches deal with schema translation, whereas in the third approach both schema and data are completely migrated into a target database. Due to substantial investments in many traditional RDBs, part of their data may need to be formatted and implemented in a new and different platform. Hence, constructing a gateway interface between the two databases might be preferred. Migrating to a new database system (DBMS) might be a good decision to make if the existing system is too expensive to maintain.

3.1.1 Approach 1: Non-relational applications on top of RDBs

Data may be required to be processed in object/XML form and stored in relational form based on the concept of object for programs and RDB for persistence. This process requires object-to/from-Relational and XML-to/from-relational mapping techniques, which link RDBs to non-relational applications. Such mapping is bi-directional on demand of updating an RDB using OO/XML interfaces. This is the reverse direction from where object-based/XML schemas are translated into an RDB schema.

Viewing objects on top of RDBs: While OO objects are associated via references, data in RDB tables are linked through the values of primary keys and foreign keys. A single object might be represented by several tuples in several tables, and therefore, joining these tables is required for queries. The problem lies in converting these objects to tabular forms in order for them to be stored in and retrieved from RDB systems when needed. This constant conversion leads to a semantic gap between the two different paradigms, which is known as the object-relational impedance mismatch [29]. To avoid this, developers have to write huge amount of code to map objects in programs into tuples in an RDB, which can be very time-consuming to write and execute. Another solution would be to use mapping query system/middleware. Query systems, e.g., Penguin [30] support object views for RDBs, which enable non-traditional applications to share data with their object schema. Penguin is an object-based DBMS that relies on RDBs for persistent [30]. Middleware is a software that links OO Programming Language (OOPL) concepts to data stored in RDBs through ODBC/JDBC, thus creating a virtual object database. Such middlewares provide mapping tools for binding tuples in RDBs, making them appear as objects for OOPLs. However, mapping using middleware requires time for schema mapping, on each occasion that stored data are accessed.

Publishing RDB data as XML documents: RDB data can be published as XML documents, using special declarative languages, to be exchanged over the Web. Various proposals, which make RDB data accessible to XML have been described [9, 26]. Through converting an RDB into XML, users see views that can be queried using XML query languages. However, data in such applications is not fully materialized in XML form, whereas the results are. Furthermore, adapting the object view for representing XML data in an RDB faces restrictions, such as data collection representations and tag naming. XPERANTO [9] and XTABLES [26] are among the systems taking this approach. The XPERANTO system translates XML-based queries into SQL over (object-)RDBs [9]. The system receives and deconstructs SQL queries and returns XML documents. However, users have to specify the queries and define more complex views using an appropriate query language, once the system publishes a default XML view. In addition, mismatches exist between XML and SQL query syntax, and more advanced object features and integrity constraints are not considered precisely. XTABLES provides the user with

a single query language which can be used to query seamlessly over relational data and metadata [26]. In addition, XTABLES can query and store XML documents in RDBs. XML documents can be stored in relational DBMSs [25]. A database that allows XML data to be stored in it is called an XML-enabled database. A whole document can be stored in a large single column in a table. The column can be a binary large object (BLOB) or a character large object (CLOB). XML-enabled databases are useful for retrieving and storing data which conform to XML form. However, they cannot effectively store a complete document with its identity, order and comments.

3.1.2 Approach 2: Database Integration

A connection can be established between RDBs and other databases which allows the applications built on top of a new DBMS to access both relational and object/XML DBMSs, giving the impression that all data are stored in one database. This represents a simple level of database integration between systems [40]. This is achieved using a special type of software called *gateways*, which support connectivity between DBMSs and do not involve the user in SQL and RDB schema. Hence, queries and operations are converted into SQL and the results are translated into target objects. Many applications use two or more underlying databases. On retrieving data from both systems, the unification of their two schemas is necessary by providing two-way mapping. During integration, systems cooperate autonomously by creating a unified and consistent data view for several databases, hiding heterogeneities and query languages [29]. Most commercial DBMSs such as Objectivity and ObjectStore provide flexibility of mapping and gateways construction among heterogeneous databases. The difference between gateways and object-relational mapping tools is that, in the former, objects are persistently stored in the new developed database system; whereas in mapping or publishing data, objects are created and handled in the normal way but are stored in an RDB. However, in both approaches old data stored in an RDB are retained.

3.1.3 Approach 3: Database Migration

Migration of an RDB into its equivalents is usually accomplished between two databases according the literature. The first database is an RDB, called the *source*, and the second, called the *target*, which represents the result of the migration process. In addition, the process is performed with or without the help of an intermediate conceptual representation, e.g., an ER model as a stage of enrichment. The input source schema is enriched semantically and translated into a target schema. Data stored in the source database are converted into the target database based on the target schema. Generally, relations and attributes are translated into equivalent target objects. Foreign keys may be replaced by another domain or relationship attributes. Weak entity relations may be mapped into component classes, multi-valued or composite attributes inside their parent class/entity. Other relationships, such as associations and inheritance, can also be extracted by analysing data dependencies or database instances. In data conversion, attributes that are not foreign keys become literal attribute values of objects, elements or sets of elements. Foreign keys realise relationships among tuples, which are converted into value-based or object references in a target database. The challenge in this process is that the data of one relation may be converted into a collection of literal/references rather than into one corresponding type. This is because of the heterogeneity of concepts and structures in the source and target data models.

3.2 Translation Techniques

Existing techniques used for RDB schema translation can be classified into two types: (i) Source-to-Target (S2T), including flat, clustering and nesting translation techniques, and (ii) Source-to-Conceptual-to-Target (SCT) translation. In some of these techniques, data might be converted based on the resulting target schema.

3.2.1 Source-to-Target (S2T) technique

This type of technique translates a physical schema source code directly into an equivalent target. However, as the target schema is generated using one-step mapping with no intermediate stage for enrichment, this technique usually results in an ill-designed database because some of the data semantics (e.g., integrity constraints) are not considered. This approach can take the following three forms:

Flat technique: This technique converts each relation into an object class/XML element in the target database [39, 22, 46]. Foreign keys are mapped into references to connect objects. However, due to the one-to-one mapping, the flattened form of RDBs is preserved in the generated database, so that object-based model features and the hierarchical form of the XML model are not exploited. This means that the target database is semantically weaker and of a poorer quality than the source. Moreover, creating too many references causes degraded performance during data retrieval.

Clustering technique: This technique is performed recursively by grouping entities and relationships together starting from atomic entities to construct more complex entities until the desired level of abstraction (e.g., aggregation) is achieved [47, 34]. A strong entity is wrapped with all of its direct weak entities, forming a complex cluster labelled with the strong entity name. This technique works well when the aim is to produce hierarchical forms with one root. This technique may reduce search time by avoiding join operations, and thus speeding up query processing, however, it may lead to complex structures and is prone to errors in translation. In addition, materializing component entities within their parent/whole entities may cause data redundancy, the loss of semantics and the breaking of relationships among objects.

Nesting technique: This technique uses the iterated mechanism of a *nest* operator to generate a nested target structure from tuples of an input relation [18]. The target type is extracted from the best possible nesting outcome. For a table T with a set of columns X, nesting on a non-empty column(s) $Y \in X$ collects all tuples that agree on the remaining columns $X - Y$ into a set [18]. However, the technique has various limitations, e.g., mapping each table separately and ignoring integrity constraints. Besides, the process is quite expensive, since it needs all tuples of a table to be scanned repeatedly in order to achieve the best possible nesting.

3.2.2 Source-to-Conceptual-to-Target (SCT) technique

This type of technique enriches a source schema by data semantics that might not have been clearly expressed. The schema is translated from a logical into a conceptual schema

through recovering the domain semantics (e.g., primary keys, foreign keys, cardinalities, etc.) and making them explicit. The results are represented as a conceptual schema using DBRE process [10]. The resulting conceptual schema can be translated into the target logical schema effectively using DBFE process. In this way, the technique results in a well-designed target database.

3.2.3 DataBase Reverse Engineering (DBRE)

DBRE is a process for enriching a source schema using semantics that might have not been clearly expressed by acquiring as much information as possible about objects and the relationships that exist among them [8]. Inferring conceptual schema from a logical RDB schema via DBRE has been extensively studied [28, 5, 10, 2]. Such conversions are usually specified by rules, which describe how to derive RDB constructs (e.g., relations, attributes, data dependencies, keys), classify them, and identify relationships among them. Semantic information is extracted by an in-depth analysis of relations in an RDB schema together with their data dependencies into a conceptual schema such as ER, UML, OO and XML data models. Data and query statements have also been used in some studies to extract data semantics. In addition, data dictionary and expert users are considered in some proposals, whereas others are based on schema design. However, some of these techniques could be combined together to form a more comprehensive solution.

Schema-based proposals: Most of the existing DBRE studies fall into this category, where the inputs are RDB schemas and the outputs are data semantics from analysing relations and attributes [12, 27, 20, 10]. The extraction of data semantics by converting an RDB schema into an Extended ER (EER) model has been studied in the early nineties [20, 10]. Two algorithms are proposed to extract a conceptual ER from an existing RDB based of the classification of relations and attributes [12, 27]. However, all those algorithms do not consider inheritance relationships. Fonkam and Gray presented a more general algorithm that is based on these algorithms, where the original contribution of this algorithm was to establish generalisation hierarchies [20]. Chiang et al. proposed a method that focuses on deriving an EER from a 3NF RDB [10]. This type of method uses a variety of heuristics to recover domain semantics through the classification of relations, attributes and key-based inclusion dependencies using the schema. However, expert involvement is required to distinguish between similar EER constructs, i.e., weak entities and specific relationship types [10]. In addition, the consistency of key naming and a well-formed schema is assumed.

Data content-based proposals: Several studies have proposed the extraction of semantics by analysing data instances and possibly schemas [10, 42, 2]. Soutou proposed a process for extracting the cardinalities of n-ary relations representing relationships by generating a set of SQL queries [42]. Data instances are used for relation classifications with respect to their keys [10]. Alhajj developed algorithms that utilise data to derive all possible candidate keys for identifying the foreign keys of each given relation in a legacy RDB [2]. This information is then used to derive a graph called RID, which includes all possible relationships among RDB relations. The RID graph works as a conceptual schema [2].

Query-based proposals: Inferring a conceptual schema based on the analysis of DDL and SQL queries embedded in applications has been suggested by several authors [5, 37, 4]. Petit et al. presented a method to extract EER model constructs from an RDB by analysing SQL queries in application programs [37]. In common with Andersson, Petit et al. extracted a conceptual schema by investigating equi-join statements [5, 37]. The method uses a join condition and the *distinct* keyword for attribute elimination during key identification. Akoka et al. focused on extracting generalisation hierarchies in an RDB using DDL, DML and data analysis [4].

Other proposals: Soutou presented an algorithm for inferring n-ary relationships from RDBs through a combination of a data dictionary, and the analysis of schema and data [43]. Alhajj and Polat re-engineered an RDB into an OODB using an expert user and the data dictionary as primary sources of information [3]. Since an RDB does not enable a natural way of representing inheritances, several heuristic and algorithmic methods have been proposed to elicit inheritance relationships hidden in RDBs [20, 4]. Data instances, schemas, DDL and DML specifications, along with understanding null value semantics, are used to detect inheritance.

Design-based proposals: Some works that have design characteristics can be used for DBRE [28, 33]. A method based on a generic schema specification model and DBRE techniques has been proposed to deal with design and re-engineering database applications [28]. Marcos et al. presented rules to translate a UML class diagram into an ORDB schema in SQL3 and Oracle 8_i [33].

The problems of DBRE process arise from processing badly-designed and poorly documented applications [28]. Many RDBs might have been specified without definition of constraints, such as keys and integrity constraints [7]. These semantics specified into conceptual schema might not be presented explicitly in data dictionaries. For example, foreign keys are not possible in Oracle 5. Moreover, many RDBs do not contain semantic constraints for optimization reasons, and not all databases are built by experienced developers, who may produce poor or inadequate structures [28].

3.2.4 DataBase Forward Engineering (DBFE)

This process is known as schema translation. A conceptual schema generated from the DBRE process can be translated into a high level data model through the application of a set of rules, called schema mapping rules. Several proposals have been made for transforming conceptual schemas, e.g., ER, EER, UML or other specific models into object-based/XML schemas [35, 28, 14, 31, 33, 45]. These proposals and many others have been used as a basis for middlewares, gateways and CASE tools.

4. RBD MIGRATION PROPOSALS

Before we embark on a detailed review on proposals used in an RDB migration, this section describes a set of properties, which can be used to compare and evaluate existing proposals. Indeed each proposal has its properties, e.g., prerequisites and data model used. These properties lead to different mapping rules for the migration process, which in turn affect the results and quality of the process. Table 1 provides a comparison and classification of some of these proposals showing the input and target generated databases, the preservation of data semantics, and technique used and prerequisites of each proposal. However, detailed descrip-

tions on these proposals as works for migrating RDBs into OODBs, ORDBs and XML according to these properties are given in Sections 5, 6 and 7, respectively.

4.1 Migration prerequisites

Existing work of database migration enforces different prerequisites on the source databases being migrated. These include the consistency of naming attributes, the availability of all keys and schema, inclusion and functional dependencies, and database instances. Most existing proposals are limited by the assumptions that they make. For instance, a source schema is required to be available for further normalisation to Third Normal Form (3NF) [10, 19] or even to 4NF [48] before the migration process can begin. However, this is not a practical choice for existing RDBs. Data dependency, which is most often represented by key constraints, plays the most important role in this process. Evaluation of functional, inclusion and key-based dependency is assumed in many proposals. Other kinds of data dependency may also be required, e.g., Multi-valued Dependency (MVD) [48, 24] and Exclusion Dependency (ED) [8, 22]. The problem of synonyms and homonyms may be assumed to has been resolved [39]. Also, the classification of relations with respect to their keys, e.g., to know whether the primary keys and foreign keys are constructed from each other may be required [10]. Other frequent assumptions are that the initial schema is well-designed and that all basic relevant constraints are given in the descriptions of the schema or provided by the user [7, 2, 46].

4.2 Input and output models

In existing work, the RDB migration process usually takes one RDB as input and aims to generate one target database. A source schema is translated into another equivalent schema and data are converted in accordance with schema translation. However, most work to date has focused on translating RDB schemas directly into other non-standard target schemas, in the context of database integration [8, 22, 48]. Few attempts have been made to generate target data models based on their conceptual schemas or other representations, as an intermediate stage for enrichment. Numerous methods have been proposed for DBRE by transforming logical data models into ER, EER, UML models. A large body of literature exists on DBFE (or database design) aiming to transform such conceptual models to logical data models. In addition, only few works consider current standards, i.e., ODMG 3.0, SQL4 and XML Schema as target models [19, 46].

4.3 Conceptual models used

Earlier models such as ER, EER and Object-Modeling Technique (OMT) are assumed in most studies as a conceptual model or target data models, whereas other works are restricted to a particular product, e.g., Oracle [45]. To enrich a source RDB structurally and semantically, graphs and models are proposed as an intermediate stage [1, 13, 14, 2, 23]. A graph called an RID, developed by Alhajj [2], has been used to translate an RDB into an OODB [3] or into an XML [46]. This graph, similar to an ER diagram is used for identifying relationships and cardinalities. A model, called BarceLona Object Oriented Model (BLOOM), has also been developed to act like a canonical model for federated DBMSs [1]. Its main goal is to upgrade the semantic level of the local schemas of different databases and to facilitate their integration. Behm et al. proposed a model, called Semi Object Type (SOT), to facilitate the restructuring of schemas during the translation of an RDB into an OODB [7]. Another model, called ORA-SS, has been proposed to support the design of non-redundant storage of semi-structured data models [13]. The ORA-SS is used as an intermediate model to map an RDB into an XML Schema [13]. The model has its own diagrammatic notations for expressing class attributes and relationships, similar to those of ER and OO data models. The model represents data as directed graphs, and focuses on modelling n-ary relationships as well as distinguishing between the attributes of relationships and those of objects. However, it uses the technique of nesting and referencing in representing relationships among objects.

4.4 Semantic preservation

RDBs typically contain implicit and explicit data semantics, concerning integrity constraints and relationships among relations. Target databases should hold equivalents to these semantics. Several previous proposals have failed to explicitly maintain all of the data semantics (e.g., integrity constraints and inheritance). Constraints are instead mapped into class methods [19] or into separate constraint classes [35]. Relationships are translated in most of the work, however, inheritance relationships have not been fully addressed. Few studies address database optimization issues, e.g., horizontal and vertical partitioning [35, 41]. Object-based data models consist of static properties (attributes and relationships) and dynamic properties (methods or functions), which make them richer than relational data models. Most existing proposals focus on constructing a static rather than dynamic target schema.

4.5 User involvement

A common observation in the different proposals is that user interaction is necessary at some point to provide additional information to achieve the desired results. User intervention might be required for the classification and understanding of keys in an RDB [8], choosing the appropriate transformation rule [6], or identifying complex relationship structures [19]. User involvement is also required for resolving optimization issues such as naming conflicts and vertically or horizontally partitioned relations [10, 39, 41], and for selecting XML documents' roots and directing the conversion process [18, 24].

5. MIGRATING RBD INTO OODB

In this section, existing proposals for the migration of RDBs into OODBs are discussed in further detail.

ER-to-OODB: Narasimhan et al. proposed a procedure that deals with an RDB abstraction through mapping its related ER model into an OO schema to exploit the ER model features, e.g., multi-valued attributes [35]. This work suggests creating a separate constraint class with methods as a sub-class for each of the OODB classes. The translation of EER models into OO models by a set of transforation rules has been illustrated [21]. EER strong entities are mapped into classes with corresponding attributes [21]. Weak entities and aggregations are mapped into component and composite object-classes, respectively. Relationships among entities are mapped into associations, generalisation/specialisation into inheritance, and categorizations

Table 1: RDB migration (prerequisites, input and output databases)

Proposal	S2T	DC	Tec	Input	Prerequisites	Data Semantics					Output			Target model
						AS	AG	IN	RI	OP	OODB	ORDB	XML	
[22]	√	√	S2T	RDB	FD,ID,ED	√	×	√	×	×	√	×	×	NS
[47]	√	×	S2T	RDB	keys, ID	√	√	√	×	×	√	×	×	NS
[41]	√	×	S2T	RDB	FD, PKs, FKs, 2NF	√	√	√	×	√	√	×	×	OMT
[48]	√	×	S2T	RDB	FD, ID, 4NF, MVD	√	√	√	×	×	√	×	×	NS
[19]	√	×	S2T	RDB	keys, FD, ID, 3NF	√	√	√	√	√	√	×	×	ODMG-93
[7]	√	√	SCT	RDB	keys, DD, Ins	√	×	√	√	×	√	×	×	ODMG-93
[3]	√	√	SCT	RDB	keys, DD, Ins	√	√	√	×	×	√	×	×	NS
[35]	√	×	S2T	ER	ER	√	√	×	×	×	√	×	×	NS
[39]	√	×	S2T	RDB	keys, non-3NF	√	√	√	×	√	√	×	×	OMT
[8]	√	×	S2T	RDB	FD, ID, ED, non-3NF	√	√	√	×	×	√	×	×	BLOOM
[44]	√	×	S2T	UML	UML class diagram	√	√	×	√	√	×	√	×	Oracle 8_i
[33]	√	×	S2T	UML	UML class diagram	√	√	√	√	√	×	√	×	SQL3
[45]	√	×	S2T	UML	UML class diagram	√	√	√	√	×	×	√	×	Oracle 8_i
[23]	√	√	SCT	RDB	PKs, FKs	√	√	√	√	×	×	×	√	XML Schema
[31]	√	√	S2T	EER	FD, ID	√	×	×	×	×	×	×	√	DTD
[14]	√	×	SCT	RDB	3NF	√	√	√	√	×	×	×	√	XML Schema
[24]	√	√	SCT	RDB	FD, MVD, JD, TD	√	√	×	×	√	×	×	√	DTD
[18]	√	×	S2T	RDB	PKs, FKs	√	√	×	×	√	×	×	√	DTD
[46]	√	√	SCT	RDB	PKs, FKs, DD	√	×	×	√	×	×	×	√	XML Schema
[16]	√	×	S2T	RDB	PKs, FKs	√	√	×	×	×	×	×	√	DTD
[25]	√	√	SCT	RDB	keys, FD, IN, MVD	√	√	√	×	×	×	×	√	DTD

S2T: Schema Translation DC: Data Conversion Tec: Technique FD: Functional ID: Inclusion Dependency TD: Transitive Dependency Ins: Data instances Dependency JD: Join Dependency PK: primary key FK: foreign key AS: Association AG: Aggregation IN: Inheritance RI: Referential Integrity OP: Optimization NS: Non-standard

into multi-inheritance.

RDB-to-OODB: Several methods have been proposed for migrating RDBs into OODBs directly, i.e., without using an intermediate conceptual representation [8, 39, 19, 22, 41, 48]. However, all these proposals, except [22], concern only schema translation. Premerlani and Blaha proposed a procedure for mapping an RDB schema into an OMT schema [39]. An initial schema is produced by representing each table and its attributes as a tentative class, and primary keys and foreign keys are determined by resolving synonyms and homonyms. Then, horizontally partitioned classes are refined into single classes, and associations and generalisations are identified using the evaluation of keys. Finally, OO classes are refined through eliminating redundant associations. Fahrner and Vossen described a method, in which an RDB schema is normalised to 3NF, enriched by semantics using data dependencies, and translated into an ODMG-93 ODL schema [19]. This method makes extensive use of inclusion and exclusion dependencies. Moreover, the resulting schema is then restructured by the user with respect to OO paradigm options, e.g., binary relationship relations are eliminated and integrity constraints are mapped into class methods. Castellanos et al. presented a method that generates the BLOOM [1] schema from an RDB [8]. The method consists of two phases. An RDB schema is improved semantically based on a knowledge acquisition process to discover implicit semantics by analysing the schema and data instances. Then the enriched RDB schema is converted into a BLOOM schema. The knowledge acquisition phase involves the determination of keys and their types, of data dependencies such as functional, inclusion and exclusion dependencies, and of the normalization of the schema to 3NF. However, unlike in Premerlani and Blaha method [39] optimization structures, e.g., horizontal decomposition or different representation possibilities of complex attribute structures are not considered. Fong suggested a sound theoretical method for converting RDBs data into OODBs [22]. Relation tuples are converted, downloaded into sequential files, and then reloaded into the OODB. However, weak entities and multi-valued and composite attributes are not clearly tackled in this work. Ramanathan and Hodges presented a method for mapping an RDB schema that is at least in 2NF into an OODB schema without the explicit use of inclusion dependencies, and without changing the existing schema [41]. All of the information required during the process comes from information on primary keys and foreign keys. However, the method also addresses database optimisation issues such as BLOBs, horizontal and vertical partitioning, which cannot be mapped into object schema without using data dependencies. Zhang et al. described a method based on MVD to remove data redundancy and update anomalies [48]. A composition process is proposed to reduce the input RDB schema. Then, the simplified relations are mapped into equivalent OO classes.

Clustering RDB relations-to-OODB: Yan and Ling presented a method that produces an OODB schema from an RDB using a clustering technique, in which clusters of relations that represent object classes, aggregation, association and inheritance relationships are identified [47]. A strong entity is wrapped with all of its direct weak entities, forming a complex cluster which holds the strong entity name. In the case of deep levels of clustering, a dominant entity may aggregate its component entities if they have no relationships with other entities. The method proposes generating OIDs for identified objects by concatenating the key values of each tuple with the relation name. Missaoui et al. adapted the clustering technique to produce a clustered EER diagram [34]. In this method, related entities are identified and defined as one unit. The diagram produced is then translated into an OO schema.

RDB-*SOT*-OODB: Behm et al. proposed a model called SOT to facilitate RDBs migration [7]. An RDB schema is mapped into the SOT schema, which is then translated into an OO schema. An SOT schema consists of a set of SOTs where each has a set of attributes of basic type, collection and reference. The references represent the relationship between SOTs. Every SOT and attribute is identified by a unique identifier to avoid naming conflicts. Transformation rules consist of five parts, namely, definitions, patterns, preconditions, schema and data operations [6]. The data migration process is accomplished automatically.

6. MIGRATING RDB INTO ORDB

Transforming conceptual models (e.g., EER, UML class diagrams) into ORDB have been studied extensively over the past ten years [44, 33, 36, 15]. A common finding from these

studies is that the logical structure of an ORDB schema is achieved by creating object-types from UML diagrams. Tables are created based on the pre-defined object-types. An association relationship is mapped using **ref** or a collection of **ref**s depending on the multiplicity of the association. Multi-valued attributes are defined using arrays/nested tables. Inheritance is defined using foreign keys or **ref** types in Oracle 8_i and the **under** clause in Oracle 9_i/SQL3 [33].

A method of mapping and preserving collection semantics into an ORDB has recently been proposed [36]. The method transforms UML conceptual aggregation and association relationships into ORDB using **row** and **multiset** provided by SQL4. More recent work has focused on mapping UML aggregation/composition relationships into ORDBs [33, 15]. Urban et al. described essential rules for converting UML class diagrams into ORDB schemas, using triggers to preserve inverse relationships between objects for bi-directional relationships [44]. Marcos et al. proposed new UML stereotype extensions for an ORDB design, focusing on aggregation and composition relationships [33]. Although most ORDB concepts are present in these proposals, their focus has been on the design of ORDBs rather than on migration. However, if a migration process uses a conceptual model as an intermediate stage, then these proposals could be useful in schema translation.

7. MIGRATING RDB INTO XML

Some proposals for migrating RDBs into XML use data dictionaries and assume well-designed RDB [14, 3, 18] whereas others consider legacy RDB for migration into XML documents [46]. Besides, the resulting XML schemas might be a DTD [18], XML Schema [46] or independent XML language [13]. However, several researchers have proposed ways to transform UML class diagrams to XML [11, 45].

RDB-to-*ER*-to-XML: Wang et al. proposed a method focusing on legacy RDBs [46]. Firstly, the ER model is extracted from the RDB by applying the DBRE technique described in [2], which results in an RID graph. Then, the RID graph is mapped into an XML Schema. The structure of the generated XML document is based on user specification into a flat or nested structure. After the schema is translated, an XML document is generated from RDB data. However, inheritance and aggregation relationships are not considered properly in this study. Fong and Cheung introduced a method in which data semantics are extracted from an RDB schema into an EER model, which is then mapped into an XSD graph [23]. The XSD graph captures relationships and constraints and is mapped in turn into an XML Schema. However, the authors suggested mapping foreign keys into a hierarchy of element/sub-elements, which may cause redundancy when an element has a relationship with more than one element. Fong et al. used the EER model to enrich an existing RDB semantically and translating it into a corresponding DTD schema. The RDB data are converted and loaded into an XML document according to the translated DTD schema [25].

RDB-to-DTD: Laforest and Boumediene described two algorithms to extract data-centric and paragraph-centric DTD from RDBs automatically [16]. One table is determined to be the main root element, and then columns of that table, which are neither primary key nor foreign keys are mapped as its sub-elements. The primary key is added to its root elements as an attribute. Other tables that hold the primary key of the root table as foreign keys are translated as sub-elements with cardinality "*". For each foreign key included in the primary key, a new sub-element with PCDATA type is generated, holding the same name as its reference table. Foreign keys that are not included in the primary key are converted into sub-elements in the root. Their composition then has to be defined, and their cardinalities defined as "1" or "?" depending on integrity constraints.

ER-to-XML: Kleiner and Lipeck translated a well-known EER model to DTD [31]. However, some data semantics cannot be represented, e.g., the limitations of DTD in specifying composite keys. Moreover, some relationships, i.e., inheritance and aggregation are not considered in this work. The work has been extended considering inheritance relationships, and generate an XML Schema from an EER model [38]. However, the algorithm tries to create a hierarchal structure that is deeper rather than larger. This may cause redundancy or disconnected elements in the resulting XML document.

UML-to-XML: Conrad et al. proposed a method for transforming UML into DTD in the context of OO software design [11]. Vela and Marcos proposed a method for extending UML to represent an XML Schema in graphical notation, which has a unique equivalence with an XML Schema [45]. However, although UML can model data semantics such as aggregation and inheritance, it is still weak and unsuitable in handling the hierarchal structure of the XML data model [23].

RDB-to-*ORA-SS*-to-XML Schema: Du et al. developed a method that employs an ORA-SS model to support the translation of an RDB schema into an XML Schema [14]. They proposed a variety of translation rules for mapping a semantically enriched RDB schema into an ORA-SS model [13], which in turn is then translated into the XML Schema. However, they adopted an exceptionally deep clustering technique, which is prone to errors.

RDB-to-*DOMs*-to-DTD document: Fong et al. proposed a procedure to translate RDBs into XML documents [24]. Based on data dependency constraints, this work de-normalises an RDB into joined tables, which are then translated to Document Object Models (DOMs). These DOMs are integrated into one DOM, which is then mapped into a DTD schema. Based on the DTD schema generated and data dependencies, each tuple of the joined tables is loaded into an object instance in DOM and then transformed into a DTD document.

NeT and CoT algorithms: Lee et al. presented the Flat Translation (FT) algorithm that maps RDB tables into DTD elements [17]. However, the algorithm neither utilises features provided by the XML model nor considers integrity constraints. Another algorithm known as Nesting-based Translation (NeT) has been proposed to remedy the drawbacks of FT using an iterated mechanism of the *nest* operator to generate nested structures of DTD schema from relational inputs [18]. However, this algorithm has some limitations, e.g., the mapping of each table separately and the nesting operations are too time consuming, as all tuples in a table need to be scanned repeatedly to achieve the best nesting outcome. Together with NeT, Lee et al. presented a Constraints-based Translation (CoT) algorithm that considers the preservation of integrity constraints.

8. DISCUSSION

The investigation into the problem of RDB re-engineering shows that proposals made so far have had different focuses. Each proposal has made certain assumptions to facilitate the process, which might be a point of limitations or a drawback. While existing works for migrating into OODBs focus on schema translation using source-to-target techniques, we have noted that most works for migrating to XML have used source-to-conceptual-to-target techniques, focusing on generating a DTD schema and data. Moreover, all research on the generation of ORDBs has focused on design rather than migration. It could be concluded, based on our analysis of the literature, that research into the migration of RDBs into object-based/XML databases is still immature, and that therefore several areas are in need of further attention.

Due to their focus on schema rather than data, the proposals reviewed above either ignore data conversion or assume working on virtual target databases (using mapping and gateways middleware) and data retain stored in RDBs. Moreover, there are still shortcomings in the implementation of RDB data conversion in a more effective manner into more than one environment. Using middleware may lead to slow performance, making the process expensive at run-time because of the dynamic mapping of tuples to complex objects. However, using object-based DBMSs and native-XML, objects can be stored and retrieved directly without any need for translation layers, hence saving development time and increasing performance.

Some semantics (e.g., inheritance, aggregation) are not considered in some work. This is mainly due to their lack of support for such semantics either in source or target data models, e.g., ER model and DTD lack support for inheritance. Despite the ability of UML to model data semantics such as aggregations and inheritances, UML is still weak and unsuitable for handling the hierarchical structure of the XML data model [23]. Although inheritance relationships could be indirectly realized in an RDB, they have been either ignored or only briefly considered. Different types of inheritance have not been tackled, such as unions, mutual exclusion, partition and multiple inheritance; and neither have their constraints, e.g., total/partial, disjoint and overlapping. Translating inheritance relationships from RDBs to object-based/XML databases and capturing their data semantics, needs more attention.

There has been less efforts to use standards such as the ODMG 3.0, SQL4 and XML Schema as target models. The adoption of standards is essential for better semantic preservations, portability and flexibility. In the ODMG 3.0 model, referential integrity is maintained automatically via inverse references. SQL4 has the ability to address complex objects in ORDBs. Compared to DTD, the XML Schema offers a much more extensive set of data types, and provides powerful referencing, nesting and inheritance mechanisms of attributes and elements.

The majority of work so far has generated databases that are either like flat relational, in which object-based model features and the hierarchical form of the XML model are usually missed, or have deep levels of clustering/nesting, which may cause data redundancy. It would be desirable to avoid the flattened form and to reduce the levels of clustering object structures as much as possible in order to increase the utilisations of the target models and to avoid undesirable redundancy. This requires the preservation of the semantics of the source database into a conceptual model, which takes into account the relatively richer data model of the target database environment. The success of the migration process depends on the extent to which data semantics are retained in the conceptual model and how they are translated into a target database.

Although known conceptual models, e.g., ER, EER and UML may be used as an intermediate representation during RDB migration, it has been argued here that they are not appropriate for the characteristics and constructs of more than one target data model, and are not supporting data representation. UML should be extended by adding new stereotypes or other constructs to specify the peculiarities of ORDB and XML models [33, 45]. In addition, several dependent models have been developed for specific applications, but these are inappropriate to be applied to generate three different data models. The SOT model [7] has been designed only for migrating RDBs into OODBs. The BLOOM model [1] was defined for different local schemas to be integrated in federated systems, whereas the ORA-SS model [14] has been designed to support semi-structured data models.

The evaluation of the different techniques and proposals has shown that very few of the existing studies provide solutions to the problems mentioned above. Viewing objects on top of existing RDBs and establishing gateways to access existing data for only data retrieval purposes cannot solve the problem of mismatch between different paradigms or preserve RDB data semantics. In addition, the existing work on database migration does not provide a complete solution for more than one target database for either schema or data conversion. Besides, none of the existing proposals can be considered as a method for migrating an RDB into an ORDB. An integrated method, which deals with migration from RDB to OODB/ORDB/XML, which covers both schema and data does not yet exist.

We propose a complete method called MIGROX [32], which takes an existing RDB as an input, enriches its metadata representation with required semantics, and constructs a Canonical Data Model (CDM). The CDM captures the essential characteristics of the target data models, for the purpose of migration. MIGROX is superior to existing proposals as it produces three different output databases, and exploits the range of powerful features provided by target data models, i.e., ODMG 3.0 and SQL4. MIGROX has three phases: 1) semantic enrichment, 2) schema translation, and 3) data conversion. In the first phase, the CDM is produced which is enriched with the RDB's constraints and data semantics that may not have been explicitly expressed in its metadata. The CDM so obtained is mapped into target schemas in the second phase. The third phase converts RDB data into its equivalents in the new database environment.

9. ADDITIONAL AUTHORS

10. REFERENCES

[1] Abelló, A., Oliva, M., Rodríguez, M. E. and Saltor, F., The Syntax of BLOOM99 Schemas. TR LSI-99-34-R Dept LSI, Jul., 1999.

[2] Alhajj, R., Extracting the Extended Entity-Relationship Model from a Legacy Relationals Database, *Information Systems*, vol. 28, pp. 597–618, 2003.

[3] Alhajj, R. and Polat, F., Reengineering Relational Databases to Object-Oriented, Constructing the Class Hierarchy and Migrating the Data, *WCRE'01*, pp. 335–344, 2001.

[4] Akoka, J., Comyn-Wattiau, I. and Lammari, N.: Relational Database Reverse Engineering, Elicitation of Generalization Hierarchies, in *ER '99*, pp. 173–185, 1999.

[5] Andersson, M., Extracting an Entity Relationship Schema from a Relational Database through Reverse Engineering, in *13th Int. Conf. on the ER Approach*, pp. 403–419, 1994.

[6] Behm, A., Geppert, A. and Dittrich, K. R., On the Migration of Relational Schemas and Data to Object-Oriented Database Systems, in *5th Int. Conf. on Re-Technologies for Info. Syst.*, pp. 13–33, 1997.

[7] Behm, A., Geppert, A. and Dittrich, K. R., Algebraic Database Migration to Object Technology, in *ER*, pp. 440–453, 2000.

[8] Castellanos, M., Saltor, F. and García-Solaco, M., Semantically Enriching Relational Databases into an Object Oriented Semantic Model, *DEXA*, pp. 125–134, 1994.

[9] Carey, M., Florescu, D., Ives, Z., Lu, Y., Shanmugasundaram, J., Shekita, E. and Subramanian, S., XPERANTO, Publishing Object-Relational Data as XML, in *WebDB (Informal Proceedings)*, pp. 105–110, 2000.

[10] Chiang, R. H., Barron, T. M. and Storey, V. C., Reverse Engineering of Relational Databases, Extraction of an EER Model from a Relational Database, *Data Knowl. Eng.*, vol. 12, pp. 107–142, 1994.

[11] Conrad, R., Scheffner, D. and Freitag J. C., XML Conceptual Modeling Using UML, in *19th Int. Conf. on Conceptual Modeling*, vol. 1920, pp. 558–571, 2000.

[12] Davis, K. H. and Arora, A. K., Converting a Relational Database Model into an Entity-Relationship Model, in *6th Int. Conf. on ER Approach*, pp. 271–285, 1988.

[13] Dobbie, G., Wu, X., Ling, T. and Lee, M., ORA-SS: Object-Relationship-Attribute Model for Semistructured Data. TR 21/00 National University of Singapore, 2001.

[14] Du, W., Lee, M. and Ling, T. W., XML Structures for Relational Data, in *WISE* (1), pp. 151–160, 2001.

[15] Eessaar, E., Whole-Part Relationships in the Object-Relational Databases, in *WSEAS*, pp. 1263-1268, 2006.

[16] Laforest, F. and Boumediene, M., Study of the Automatic Construction of XML Documents Models from a Relational Data Model, in *DEXA Workshops*, pp. 566–570, 2003.

[17] Lee, D., Mani, M., Chiu, F. and Chu, W. W., Nesting-Based Relational-to-XML Schema Translation, in *WebDB*, pp. 61–66, 2001.

[18] Lee, D., Mani, M., Chiu, F. and Chu, W. W., NeT and CoT: Translating Relational Schemas to XML Schemas using Semantic Constraints, in *CIKM*, pp. 282–291, 2002.

[19] Fahrner, C. and Vossen, G., Transforming Relational Database Schemas into Object-Oriented Schemas according to ODMG-93, in *DOOD '95*, pp. 429–446, 1995.

[20] Fonkam, M. M. and Gray, W. A., An Approach to Eliciting the Semantics of Relational Databases, in *4th Int. Conf. on Advanced Info. Syst. Eng.*, vol. 593, pp. 463–480, 1992.

[21] Fong, J., Mapping Extended Entity Relationship Model to Object Modeling Technique, *SIGMOD Record*, vol. 24(3), pp. 18–22, 1995.

[22] Fong, J., Converting Relational to Object-Oriented Databases, *SIGMOD Record*, vol. 26, pp. 53–58, 1997.

[23] Fong, J. and Cheung, S. K., Translating Relational Schema into XML Schema Definition with Data Semantic Preservation and XSD Graph, *Information & Software Technology*, vol. 47, pp. 437–462, 2005.

[24] Fong, J., Wong, H. K. and Cheng, Z., Converting Relational Database into XML Documents with DOM, *Information & Software Technology*, vol. 45, pp. 335–355, 2003.

[25] Fong, J., Fong, A., Wong, H. and Yu, P., Translating Relational Schema with Constraints into XML Schema, *Int. Journal of Software Engineering and Knowledge Engineering*, vol. 16(2), pp. 201–244, 2006.

[26] Funderburk, J., Kiernan, G., Shanmugasundaram, J., Shekita, E. and Wei, C., XTABLES: Bridging Relational Technology and XML, *IBM Systems Journal*, vol. 41(4), pp. 616–641, 2002.

[27] Johannesson, P and Kalman, K., A Method for Translating Relational Schemas into Conceptual Schemas, in *8ht Int. Conf. on Enity-Relationship Approach*, pp. 271–285, 1989.

[28] Hainaut, J., Tonneau, C., Joris, M. and Chandelon, M., Schema Transformation Techniques for Database Reverse Engineering, in *12th Int. Conf. on the Entity-Relationship Approach*, vol. 823, pp. 364–375, 1994.

[29] Hohenstein, U. and Plesser, V., Semantic Enrichment, A First Step to provide Database Interoperability, *Workshop Föderierte Datenbanken*, pp. 3–17, 1996.

[30] Keller, A. and Wiederhold, G., Penguin: Objects for programs, relations for persistence, Succeeding with Object Databases, John Wiley & Sons, 2001.

[31] Kleiner, C. and Lipeck, U. W., Automatic Generation of XML DTDs from Conceptual Database Schemas, *GI Jahrestagung* (1), pp. 396–405, 2001.

[32] Maatuk, A., Ali, A. and Rossiter, N., An Integrated Approach to Relational Database Migration, in *Int. Conf. on Information and Communication Technologies*, Bannu, Pakistan, pp. 6pp, 2008 (In Press).

[33] Marcos, E., Vela, B. and Cavero, J. M., A Methodological Approach for Object-Relational Database Design using UML, *Soft. and Syst. Modeling*, vol. 2, pp. 59–75, 2003.

[34] Missaoui, R., Gagnon, J. and Godin, R., Mapping an Extended Entity-Relationship Schema into a Schema of Complex Objects, in *14th Int. Conf. on Object-Oriented and Entity-Relationship Modelling*, vol. 1021, pp. 204–215, 1995.

[35] Narasimhan, B., Navathe, S. B. and Jayaraman, S.,

On Mapping ER Models into OO Schemas, in *12th Int. Conf. on the Entity-Relationship Approach*, vol. 823, pp. 402–413, 1993.

[36] Pardede, E., Rahayu, J. and Taniar, D., Mapping Methods and Query for Aggregation and Association in Object-Relational Database using Collection, In *ITCC* (1), pp.539-, 2004.

[37] Petit, J., Kouloumdjian, J., Boulicaut, J. and Toumani, F., Using Queries to Improve Database Reverse Engineering, in *13th Int. Conf. on the Entity-Relationship Approach*, vol. 881, pp. 369–386, 1994.

[38] Pigozzo, P. and Quintarelli, E., An Algorithm for Generating XML Schemas from ER Schemas, in *SEBD*, pp. 192–199, 2005.

[39] Premerlani, W. J. and Blaha, M. R., An Approach for Reverse Engineering of Relational Databases, *Communications of the ACM*, vol. 37, pp. 42–49, 1994.

[40] Parent, C. and Spaccapietra, S., Database Integration: The Key to Data Interoperability, *Advances in Object-Oriented Data Modeling*, pp. 221–253, 2000.

[41] Ramanathan, S. and Hodges, J., Extraction of Object-Oriented Structures from Existing Relational Databases, *SIGMOD Record*, vol. 26(1), pp. 59–64, 1997.

[42] Soutou, C., Extracting N-ary Relationships Through Database Reverse Engineering, in *15th Int. Conf. on Conceptual Modeling*, vol. 1157, pp. 392–405, 1996.

[43] Soutou, C., Inference of Aggregate Relationships through Database Reverse Engineering, in *17th Int. Conf. on Conceptual Modeling*, vol. 1507, pp. 135–149, 1998.

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

[45] Vela, B. and Marcos, E., Extending UML to Represent XML Schemas, in *CAiSE Short Paper Proceedings*, 2003.

[46] Wang, C., Lo, A., Alhajj, R. and Barker, K., Novel Approach for Reengineering Relational Databases into XML, in *ICDEW '05 Workshops*, pp. 1284, 2005.

[47] Yan, L. and Ling, T. W., Translating Relational Schema with Constraints into OODB Schema, in *the IFIP WG 2.6 Database Semantics Conf. on Interoperable Database Systems*, vol. A-25, pp. 69–85, 1993.

[48] Zhang, X., Zhang, Y., Fong, J. and Jia, X., Transforming RDB Schema to Well-structured OODB Schema, *Information & Software Technology*, vol. 41, pp. 275–281, 1999.