Semantic Enrichment: The First Phase of

Relational Database Migration

Abdelsalam Maatuk, M. Akhtar Ali, Nick Rossiter

School of Computing, Engineering & Information Sciences, Northumbria University, Newcastle upon Tyne, UK.

Email: {abdelsalam.maatuk;akhtar.ali;nick.rossiter}@unn.ac.uk

*Abstract*—Semantic enrichment is a process of analyzing and examining a database to capture its structure and definitions at a higher level of meaning. This is done by enhancing a representation of an existing database's structure in order to make hidden semantics explicit. In contrast to other approaches, we present an approach that takes an existing relational database as input, obtains a copy of its meta data and enriches it with as much semantics as possible, and constructs an enhanced Relational Schema Representation (RSR). Based on RSR, a Canonical Data Model (CDM) is generated, which captures essential characteristics of target databases (i.e., object-based and XML) suitable for migration. We have developed an algorithm for generating CDM from an enhanced relational representation of an input relational database. A prototype has been implemented and experimental results are reported.

*Keywords*—Semantic enrichment, Relational databases, Object-based databases, XML Schema

# Introduction

 Object-Oriented DataBases (OODBs), Object-Relational DataBases (ORDBs) and eXtensible Markup Language (XML) have become mainstream because they offer more functionality and flexibility than traditional Relational DataBases (RDBs). The advantages provided by these relatively newer technologies and the dominance of RDBs and their weaknesses in handling complex data have motivated a growing trend for migrating RDBs into OODBs, ORDBs and XML instead of designing them from scratch [3,5,11]. This can be accomplished through reverse engineering an RDB into a conceptual schema that is enriched with its semantics and constraints. The result can be converted into another database according to a target platform. However, the question is: which of the new databases is most appropriate to use? So there is a need for an integrated method that deals with database migration from RDB to OODB/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. Such techniques could help increase the acceptance of the newer approaches among enterprises and practitioners. However, the difficulty facing this method is that it is targeting three databases that are conceptually different. Due to the heterogeneity among the three target data models, a canonical model is needed to bridge the semantic gap among them. We believe that it is necessary to develop a Canonical Data Model (CDM) to facilitate our integrated approach to database migration [15]. The CDM should be able to preserve and enhance RDB's integrity constraints and data semantics to fit in with target databases' characteristics. Consequently, additional domain semantics need to be investigated, e.g., relation classifications and relationship identification.

Our aim in this paper is to present an approach in which necessary (explicit) semantics (e.g., relation and attribute names, keys, etc.) about a given RDB could be inferred, leading to the construction of an enriched structure called Relational Schema Representation (RSR). The RSR constructs are then classified to produce a CDM, which is enhanced by additional (implicit) data semantics (e.g., classes and attributes classification, and relationship names, types, cardinalities, inverse relationships). More specifically, our aim is to construct an RSR from the extracted logical relational schema as part of a process called Semantic Enrichment (SE), which results in generating a further enriched structure (i.e., CDM) that can be used for migrating RDBs into object-based/XML databases. Canonical models designed for database integration should have semantics at least equal to any of the local schemas to be integrated [19]. Similarly, our CDM is designed to upgrade the semantics level of RDB and to play the role of an intermediate stage for migrating RDBs to OODB/ORDB/XML acting on both levels: schema translation and data conversion. Its constructs are classified to facilitate the migration into complex target objects avoiding the flat one-to-one and complicated nesting conversions. Through the CDM, well-structured target databases can be obtained without proliferation of references and redundancy. However, its richness may not be fully exploited due to the relatively limited expressiveness of the input RDB. Consequently, some object concepts provided by target database, e.g., behavioural aspects, get less attention in our CDM.

The rest of the paper is organised as follows. Section II provides an introduction to the SE process. Section III describes how to construct an RSR based on meta data extracted from an existing RDB. Section IV presents the CDM definition and how to generate it from an RSR and an existing RDB. We evaluate our method in Section V. The related work is presented in Section VI, and Section VII provides conclusions.

# Overview of Semantic Enrichment

 Semantic Enrichment (SE) is a process of analyzing databases to understand their structure and meaning, and to make hidden semantics explicit. Conflicts in naming have to be resolved, and attributes and interrelationships amongst data have to be deduced. In our approach, the SE process involves the extraction of data semantics of an RDB to be represented in RSR followed by conversion into a much enriched CDM. This facilitates the migration into new target databases without referring repeatedly to the existing RDB. The main benefit from using RSR and CDM together is that an RDB is read and enriched once while the results can be used many times to serve different purposes (e.g., schema translation, data conversion). Fig. 1 shows the schematic view of the SE process. The process starts by extracting the basic metadata information about an existing RDB in order to construct RSR, which is designed in such a way to ease key matching for its constructs classification. To get the best results, it is preferred that the SE process is applied to the schema in 3rd Normal Form (3NF). A relation that is not in 3NF may have redundant data, update anomalies or no clear semantics of whether it represents one real world entity or a relationship type. The next major step is to identify the CDM constructs based on classification of RSR constructs, including relationships and cardinalities, which are performed through data access. Lastly, the CDM structure is generated.

# Extracting RSR from an RDB

 In this section, we define 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 process includes relation names and attribute properties (i.e., attribute names, data types, length, default values, and whether the attribute is nullable). Moreover, the most important information needed is about 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 there is 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:** An RDB schema is represented in our approach as a set of elements $RSR$, where:

|  |
| --- |
| **Emp** (eno, ename, bdate, address, spreno\*,dno\*)**Salaried\_emp** (eno*\**, salary)**Hourly\_emp** (eno*\**, pay\_scale)**Proj** (pnum, pname, plocation, dnum\*)**Dept** (dno, dname, mgr\*, startd)**Dept\_locations** (dno\*, location)**Works\_on** (eno*\*,* pno*\**) Kids (eno\*, name, sex) |

Fig. 2. Sample input RDB schema

$RSR := \{R | R := 〈rn, Arsr, PK, FK, EK, UK〉\}$,

* $rn$denotes the name of relation $R, $
* $Arsr$ denotes the set of attributes of $R$: $Arsr := \{ a | a := 〈an , t, l, n, d〉\}$, where $an$ is an attribute name, $t$ its type, $l$ data length, $n$ nullable or not (‘y’|‘n’) and $d$a default value if given. The function $A(R)$ returns a set containing only $an $component of attributes.
* $PK$ denotes R’s PK: $PK := \left\{α \right| α := 〈pa, s〉\}$, where $α$ represents one PK attribute, $pa$ 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. The function $P(R) $returns a set containing only *pa* component of the PK of $R$.
* $FK$ denotes a set of FK(s) of $R: FK :=\left\{β \right| β :=〈er,\{〈fa, s〉\}〉\}$, where $β$ represents one FK (whether it is single or composite), $er$ is the name of an exporting (i.e., referenced) relation that contains the referenced PK, $fa$ is a FK attribute name, and $ s$ is a sequence number. The function $F(R) $returns a set containing only $fa $component of all FKs of $R$. The function $fk(β) $returns a set containing $fa$ component of one FK of $R$.
* $EK$ is a set of EK(s) of $R: EK :=\left\{γ \right| γ :=〈ir, \{〈ea, s〉\}〉\},$ where $γ$ 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). The function $E(R)$ returns a set containing only *ea* component of all EKs of$ R$. The function $ek(γ)$ returns a set containing $ea $component of one EK of $R$.



Fig. 1. Schematic view of the Semantic Enrichment process

* $UK$ is a set of UK(s) of $R: UK :=\left\{δ \right| δ :=\{〈ua, s〉\}\}$, where δ represents one UK, $ua$ is an attribute name and $s$ is a sequence number.
* The function *getRSRrelation*($r$) returns the $RSR$ relation that corresponds to the relation name$ r$. To get an element of a composite construct, we use “.” notation, e.g., $R.rn$.

 The main purpose behind constructing an RSR is to read essential metadata into memory outside the database’s secondary storage. An efficient RSR construction overcomes the complications that occur during matching keys in order to classify relations (e.g., strong or weak), attributes, e.g., non-key attribute (NK) and relationships, e.g., M:N, inheritance, etc. Each relation $ R$ is constructed with its semantics as one element, which is easily identifiable and upon which set theoretic operations can be applied for matching keys. Each of $R$’s elements describes a specific part of $R$ (e.g., $Arsr$ describes $R$’s attributes). An important advantage of RSR is that it identifies the set $ EK$ therefore adding more semantics to an RDB’s metadata. The $EK$ holds keys that are exported from $ R$ to other relations.

 Consider the database shown in Fig. 2. PKs are underlined and FKs are marked by “\*”. Table I gives the RSR constructed from the database showing only Emp, Salaried\_emp, Dept and Works\_on relations.

# Generation of CDM from RSR

 This section presents the CDM definition and its algorithm. We concentrate on how to identify CDM constructs using information provided by RSR and how to generate relationships and cardinalities among classes using data instances. RSR constructs are classified, enriched and translated into CDM. CDM specifications are based on the similarities among object-based and XML data models. Similarities produce natural correspondences that can be exploited to bridge the semantic gap among diverse data models. This provides a basis for the CDM that can be used as an intermediate representation to convert an RDB into more than one target database. The definition of CDM is provided in Subsection A. The algorithm to generate CDM from an RSR and an RDB is presented in Subsection B.

## Definition of CDM

 CDM has three concepts: class, attribute and relationship. Attributes define class structure, whereas relationships define a set of relationship types. The model is enriched by semantics from an RDB such as PKs, FKs, attributes length, etc. Besides, the model has taken into consideration features that are provided by object-based and XML databases such as association, aggregation and inheritance. However, the CDM is independent of an RDB, from which it has taken semantics as well as any target databases to which it might be converted. Real world entities, multi-valued and composite attributes, and relationship relations are all represented as classes in CDM. Object-based databases encapsulate static (i.e., properties) and dynamic aspects (i.e., methods) of objects. However, dynamic aspects will get less attention in CDM compared to static aspects because an RDB does not support methods attached to relations. Static aspects involve a definition of class, and its attributes and relationships. CDM classes are connected through relationships.

 **Definition 2:** The CDM is defined as a set of classes, $CDM:= \left\{C \right| C:= 〈cn, cls, abs, Acdm, Rel, UK〉\},$ where each class $C$has a name$ cn$, a classification $cls$and whether it is abstract or not *abs*. Each $C$has a set of attributes$ Acdm$, a set of relationships $Rel$and a set of UKs$ UK$.

|  |
| --- |
| TABLE IRESULT OF RSR CONSTRACTION |
| *r n* | *A rsr* | *PK* | *FK* | *EK* | *UK* |
| *an* | *t* | *l* | *n* | *d* | *pa* | *s* | *er* | *fa* | *s* | *ir* | *ea* | *s* | *ua* | *s* |
| Emp | enoenamebdateaddresssprenodno | intchardatecharintint | 25404025 | nnyyyn |  | eno | 1 | EmpDept | sprenodno | 11 | Salaried\_empHourly\_empWorks\_onDeptKidsEmp | enoenoenomgrenospreno | 111111 |  |  |
| Salaried\_emp | enosalary | intint | 25 | ny |  | eno | 1 | Emp | eno | 1 |  |  |  |  |  |
| Dept | dnodnamemgrstartd | intcharintdate | 4025 | nnny |  | dno | 1 | Emp | mgr | 1 | EmpProjDept\_locations | dnodnumdno | 111 | mgr | 1 |
| Works\_on | enopno | intint | 25 | nn |  | enopno | 12 | EmpProj | enopno | 11 |  |  |  |  |  |
|  |

* Classification (*cls*): A class $C$is classified into nine different kinds of classes (according to relationship participations), which facilitate its translation into target schemas:
1. Regular Strong Class (RST): main class,
2. Secondary Strong Class (SST): super-class,
3. Subclass (SUB): subclass,
4. Secondary Subclass (SSC): inherited subclass,
5. Regular Relationship Class (RRC): M:N relationship class without attributes,
6. Secondary Relationship Class (SRC): referenced RRC, M:N relationship with attributes, or n-ary relationships, where n>2,
7. Multi-valued Attribute Class (MAC): class represents multi-valued attributes,
8. Composite Attribute Class (CAC): class represents composite attributes, and
9. Regular Component Class (RCC): component class in a relationship rather than its whole class.
* Abstraction (*abs*): A superclass is abstract (i.e., *abs* := true) when all its objects are members of its subtypes. Instances of an abstract type cannot appear in database extension but are subsumed into instances of its subtypes. A class is not abstract (i.e., *abs* := false) when all (or some of) its corresponding RDB table rows are not members of other subtables.
* Attributes (*Acdm*): A class $C$has a set of attributes of primitive data type,

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

* Relationships (*Rel*): Each class $C$ has a set of relationships $Rel:=\left\{rel \right| rel:= 〈RelType, dirC, dirAs, c, invAs〉\}. $ Each relationship $rel$ is defined in $C$ with another class $C’$, through a set of attributes $dirAs$ using relationship type $RelType$ and cardinalities $c.$ $RelType$can have the following values: ‘*associated with*’ for association, ‘*aggregates*’ for aggregation, and ‘*inherits*’ or ‘*inherited by*’ for inheritance. CDM does not support multiple inheritances, as target database standards do not allow a concrete subtype to have more than one concrete super-type; hence, a subclass inherits only from one superclass. The $dirC$ is the name of the related class $C’ $participating in the relationship, and $dirAs$ is a set containing the attribute names representing the relationship in $C’$, whereas the inverse relationship attribute names in $C$ are contained in the set $invAs$. Cardinality $c$is defined by *min..max* notation to indicate the minimum and maximum occurrence of $C'$object(s) within $C$objects.

## Algorithm for Generation of CDM

**algorithm** **GenerateCDM**($rsr$: RSR) **return** CDM

 $cdm$ : CDM := $∅$

1. **foreach** relation $ R$ ∈ $rsr$ **do**

 $r $: relation name := $R.rn$

 $A’$: set[nonKey attribute] := $A\left(R\right)-(P\left(R\right)∪F(R)$)

2. **if** ($P(R)∩F(R)=∅$) **then** $ cls $:= RST // classify classes

 **else if** ($DFK(R$*)>*1) **then**

 **if** ($DFK(R)$ = 2 **and** $A’ = ∅ $**and** $E(R) = ∅)$ **then**

 $cls $:= RRC **else** $cls $ := SRC

 **else if** $(P(R)⊆F(R))$ **then** $ cls $:= SUB

 **else if** $(F\left(R\right)-P(R) = ∅$ **and** $ E(R) = ∅ $) **then**

 **if** (*DangKs*($R$) = 1 **and** $ A’ = ∅$ ) **then** $cls $ := MAC

 **else** $cls $:= CAC

 **else** $cls $:= RCC

 **end if**

3. $ Acdm$ := *classifyAttributes*($R.Arsr$)

4. **foreach** foreign key $β \in R.FK$ **do**

 // identify and classify relationships

 $Re$ := *getRSRrelation* ($β.er$)

 $re$ : relation name := $Re.rn$

 **if** ($fk\left(β\right) ⊈P(R)$ **or**  $ DFK(R)\geq 2$) **then**

 $RelType$ := ‘*associated with*’

 **else if** $(P \left(R\right)⊆ fk\left(β\right) $) **then** $ RelType$ := ‘*inherits*’

 **end if**

 $c$ := *deterCard*($r, fk(β)$) // determine cardinality

 $Rel :=Rel∪\{〈RelType, re$,$ P(Re ), c, fk(β)〉\}$ // add $rel $

 **end for**

5. **foreach** exported key $γ\in R.EK $**do**

 // identify and classify inverse relationships

 $Ri$ := *getRSRrelation* ($γ.ir$)

 $ ri $: relation name := $Ri.rn$

  **if** $(ek(γ) ⊈ P (Ri)$ **or** *DFK* $\left(Ri \right)\geq 2$) **then** $ $

 $RelType$ := ‘*associated with*’

 **else if** ( $ek(γ) ⊆ P (Ri)$ ) **then**

 **if** ($ek(γ)\ne P(Ri$)) **then** $ RelType$ := ‘*aggregates*’

 **else** $RelType$ := ‘*inherited by*’

 **if** ($cls$ = RST) **then** $ cls $:= SST

 **else** $ cls $:= SSC

 **end if**

 **end if**

 $c$ := *deterInverCard*($r, ri, ek\left(γ\right) $) // inverse cardinality

 $Rel := Rel ∪\{〈RelType, ri, ek(γ), c, P(R)〉\}$

 **end for**

$abs$ **:=** *checkAbstraction*($R$)

 $cdm:=cdm∪\{〈r, cls, abs,Acdm, Rel, R.UK〉\}$ // add a class

 **end for**

 **return** $cdm$

**end algorithm**

Fig. 3. The **GenerateCDM** Algorithm

 This subsection presents the **GenerateCDM** algorithm shown in Fig. 3. Given an RSR and RDB data as input, the algorithm goes through a main loop to classify RSR constructs and generates their equivalents in CDM (ref point 1). Using key matching the algorithm classifies each relation$ R$ of the input RSR, its attributes and relationships. Abstraction of each class in CDM is checked using the *checkAbstraction* function. The set of unique keys $R.UK$ remains unchanged. Each relation $ R $is classified and mapped into class$ C $ (ref point 2). We assume that relation kinds/relationships participation are represented in RDB by means of PKs/FKs matching, i.e., keys composition of each other. Other representations may lead to different target constructs. For instance, $R$ is a main class if its PK is not fully or partially composite of any FKs, i.e., $P\left(R\right)∩F\left(R\right)=∅; R$ is a subclass, if its PK is entirely composite of the primary key of a superclass relation (i.e., we assume one relation for each superclass and one for each subclass in inheritance representations). Similarly $R$ is a weak relation if its PK is a partial composite of the primary key of a strong relation. Several functions are used to facilitate the CDM construct classifications, e.g., *DFK*(*R*) function returns the number of disjoint FKs, if $R$ is a relationship relation, and *DangKs*(*R*) returns the number of dangling key attributes in the case that $R$ is a weak entity relation. Using the *classifyAttributes* function, attributes of $R$ are identified and mapped (with other properties, e.g., data type) into attributes of$ C $ (ref point 3). Attributes are classified into non-key attribute, PK attributes or FK attributes using$ tag$. Using $PK,$ $FK$ and $EK$ sets of $R,$ all relationships are identified, classified and their cardinalities determined, and then mapped into CDM as association, inheritance or aggregation. $FK$ set (ref point 4) shows relationships (i.e., ‘*associated with’*, ‘*inherits’*) which are established from other relations side, when $ R$ contains FKs, whereas $EK$ set (ref point 5), helps to identify relationships (i.e., ‘*associated with’*, ‘*aggregates’* and ‘*inherited by’*) when $R$ is a dominant (referenced) relation. Cardinality *c* of each relationship is determined by querying data in a complete database. The function *deterCard* determines *c* when $R$ contains FKs, and the *deterInverCard* function returns the inverse *c* when $R$ is referenced by other relations.

After generating CDM, we translate it into object-based/XML schemas, details of which can be found in our technical report [16].

 Consider the RSR shown in Table I to the input to the **GenerateCDM** algorithm. Fig. 4 shows the resulting CDM for only **Emp** and **Dept** classes. The CDM's class **Emp** has attributes: *ename, eno, bdate, address, spreno* and *dno*. Other properties (e.g., attributes' types, tags, default values) are not shown due to lack of space. The class **Emp** is *'associated with'* classes: **Dept**, **Works\_on** and with itself. Moreover, it *'aggregates'* **Kids** class and *'inherited by'* **Salaried\_emp** and **Hourly\_emp** classes. Relationships with cardinalities are defined in CDM classes as: $RelType\{invAs\leftrightarrow dirC\left(dirAs\right)c$($\leftrightarrow $ indicates bidirectional association and $ \leftarrow $indicates aggregation).

|  |
| --- |
| **Emp**[*Acdm*:= {ename, eno, bdate, address, spreno, dno}, Rel:= {*associated with*{ dno $\leftrightarrow $ **Dept**(dno)1..1, eno $\leftrightarrow $ **Dept**(mgr)0..1, spreno $\leftrightarrow $ **Emp**(eno)1..1, eno $\leftrightarrow $ **Emp**(spreno)0..\*, eno $\leftrightarrow $ **Works\_on**(eno)1..\*}, *aggregates* {eno $\leftarrow $**Kids**(eno)0..\*}, *inherited by*{**Salaried\_emp**, **Hourly\_emp**}}] **Dept**[*Acdm*:= {dname, dno, mgr, startd},Rel:= {*associated with* {mgr $\leftrightarrow $ **Emp**(eno)1..1, dno $\leftrightarrow $ **Emp**(dno)1..\*, dno $\leftrightarrow $ **Proj**(dnum)1..\*}, *aggregates* {dno$ \leftarrow $**Dept\_locations**(dno)1..\*}}]  |

Fig. 4. Sample generated CDM schema

# Experimental study

 To demonstrate the effectiveness and validity of the CDM, a prototype has been developed using Java 1.5, realizing the algorithm presented in this paper. We setup two experiments to evaluate our approach by examining the differences between source RDB and target databases generated by the prototype. The experiments were run on a PC with Pentium IV 3.2 GHz CPU and 1024 MB RAM operating under Windows XP Professional. We measured database equivalences, including semantics preservation, loss of data and redundancy, and integrity constraints. Full details about the experiments can be found in [14,15].

 In the first experiment, we test schema information preservation by comparing the target schemas resulting from our prototype and those generated from other manual-based mapping techniques. A schema is correct if all concepts of underlying model are used correctly with respect to syntax and semantics [10]. In general, the results from the database engineering process could be validated against the results that are obtained manually by a knowledgeable person [7]. So we claim that the CDM generated from an RDB is correct when target schemas generated based on it are equivalent to the schemas mapped from the same RDB by other approaches. The CDM is then validated as a representation of an existing RDB. The second experiment was a query-based experiment based on the BUCKY benchmark [4]. We have translated the benchmark queries into equivalent versions in OODB and XML and run them on their native systems, observing any differences in results regarding data content and integrity constraints equivalence.

 After evaluating the results, our approach is shown to be feasible, efficient and correct. Given that all approaches that have been compared to our approach, in the first experiment, are manual techniques, which give the user an opportunity to use all features of target models to result in well-designed physical schemas, we found that our approach, which is fully-automatic has the ability to generate a more accurate and intuitive target schemas. The CDM, which preserves an enhanced structure of an existing RDB, is translatable into any of the three target schemas and the queries return identical results. Therefore, target databases are generated without loss or redundancy of data. Moreover, many semantics can be converted from RDB into the targets, e.g., association, aggregation and inheritance with integrity constraints enforced on the target database. Some update operations are applied on the databases to show that integrity constraints in the RDB are preserved in the target database. However, we cannot cover automatically referential integrity on REFs that are in nested tables in ORDB because Oracle does not have a mechanism to do so; this integrity could be preserved once the schema is generated, e.g., using triggers. In addition, the keys of XML elements may not be valid for other element(s), which would substitute them in instance document. This is because Xpath 2.0 is not schema-aware.

# Related Work

Inferring a conceptual schema from a logical RDB schema has been extensively studied by many researchers in the context of database reverse engineering [5, 2, 3, 6, 18], semantic enrichment [19, 12] and schema translation [9, 20, 11]. Such conversions are usually specified by rules, which describe how to derive RDBs constructs (e.g., relations, attributes, 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 model such as Entity-Relationship Model (ERM), UML, object oriented and XML data models. Data and query statements are also used in some work to extract some semantics. However, most of the work has been focused on schema translation rather than data conversion with an aim to generate one target data model based on its conceptual schema or other representations as an intermediate stage for enrichment. In addition, the existing work does not provide a complete solution for more than one target database, for either schema or data conversion. A classification on database migration techniques can be found in our work [13].

An approach that focuses on deriving an Extended ERM (EERM) from an existing RDB is presented in [6]. The process recovers domain semantics through classification of relations, attributes and key-based inclusion dependencies using the schema. However, expert involvement is required to distinguish between similar EERM constructs. The approach discussed in [3] extracts a conceptual schema by analysing equi-join statements. The approach uses a join condition and a distinct keyword for elimination of attributes during key location. Ref [2] developed algorithms that utilise data to derive all possible candidate keys for identifying foreign keys of each given relation in a legacy RDB. This information is then used to construct what is termed as RID graph, which includes all possible relationships among RDB relations. Ref [11] introduces a method in which data semantics are extracted from an RDB into an EERM, which is then mapped into a conceptual XML Schema Definition language (XSD) graph that captures relationships and constraints among entities in an EERM. Finally, the XML logical schema is extracted from the XSD. A model, called BLOOM, is developed, which acts like a CDM for federated database management systems [1]. Its goal is to upgrade the semantic level of local schemas of different databases and facilitate their integration. A method, which improves an RDB schema semantically and translates it into a BLOOM schema, is described in [5]. Ref [18] proposes a procedure for mapping an RDB schema into an Object-Modelling Technique (OMT) schema [18]. Ref [9] develops a method for converting an RDB schema into a model, called ORA-SS [8], which is then translated into XML Schema. However, they adopt an exceptionally deep clustering technique, which is prone to error.

 Although current conceptual models, e.g., ERM or UML may be used as a CDM in database migration, 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 (e.g., inheritance, aggregation) have not been considered in some work, mainly due their lack of support either in source or target models, e.g., ERM and DTD lack support for inheritance. UML should be extended by adding new stereotypes or other constructs to specify ORDB and XML models peculiarities [17, 20] and it is still weak and not suitable to handle the hierarchical structure of the XML data model [11]. Several dependent models were developed at specific applications, and they are inappropriate to be applied to generate three different data models; e.g., BLOOM was defined for different schemas to be integrated in federated systems and ORA-SS has been designed to support semi-structured data models.

In contrast, the CDM described in this paper can be seen as an independent model, which embraces object oriented concepts with rich semantics that cater for object-relational and XML data models. It preserves a variety of classification for classes, attributes and relationships, which enable us to represent the target complex structures in an abstract way. Classes are distinguished as abstract classes and concrete classes. Relationships are defined in the CDM in a way that facilitates extracting and loading of data during data conversion including defining and linking objects using user-defined object identifiers. Moreover, it provides non-OODB key concepts (i.e., FKs, null and UKs) and explicitly specifies whether attributes are optional or required using null values. Because of these characteristics, our CDM can facilitate the migration of an existing RDB into OODB/ORDB/XML during both schema translation and data conversion phases.

# Conclusion and Future Works

 This paper presents an approach to the semantic enrichment, in which necessary data semantics about a given RDB are inferred and enhanced to produce an RSR. The RSR constructs are then classified to generate a CDM, which provides a description of the existing RDB's implicit and explicit semantics. The generated CDM is a sound source of semantics and is a well organised data model, which forms the starting point for the remaining phases of database migration. In addition to considering most important characteristics of target models, the CDM preserves all data semantics that can possibly be extracted from an RDB, e.g., integrity constraints, associations, aggregations and inheritance. Moreover, the CDM represents a key mediator for converting an existing RDB data into target databases. It facilitates reallocation of attributes in an RDB to the appropriate values in a target database. A prototype has been implemented based on the algorithm proposed in this paper for generating OODB, ORDB and XML schemas. Our approach has been evaluated by comparing the prototype’s outputs with the results of existing methods. We found that the results were comparable. Therefore, we conclude that the source and target databases were equivalent. Moreover, the results obtained demonstrate that our approach, conceptually and practically, is feasible, efficient and correct. Our future research focus is on data specific manipulation (e.g., update/query) translations and further prototyping to simplify relationship names that are automatically generated.

##### References

1. A. Abelló, M. Oliva, M. E. Rodríguez and F. Saltor, *The Syntax of BLOOM99 Schemas*, TR LSI-99-34-R Dept LSI, 1999.
2. R. Alhajj, “Extracting the extended entity-relationship model from a legacy relational database,” *Inf. Syst.* vol. 28, pp. 597-618, 2003.
3. M. Andersson, “Extracting an entity relationship schema from a relational database through reverse engineering,” in *13th int. conf. on the ER approach*, pp. 403-419, 1994.
4. M. Carey, et al., “The BUCKY object-relational benchmark,” *SIGMOD Rec*, vol. 26, pp. 135-146, 1997.
5. M. Castellanos, F. Saltor and M. García-Solaco, “Semantically enriching relational databases into an object oriented semantic model,” in *DEXA,* pp. 125-134, 1994.
6. R. Chiang, T. Barron and V. C. Storey, “Reverse engineering of relational databases: Extraction of an EER model from a relational database,” *Data Knowl. Eng*, vol. 12, pp. 107-142, 1994.
7. R. Chiang, T. Barron and V. C. Storey, “A framework for the design and evaluation of reverse engineering methods for relational databases,” *Data Knowl. Eng*., vol. 21, pp. 57-77, 1996.
8. G. Dobbie, X. Wu, T. Ling and M. Lee, *ORA-SS: Object-Relationship-Attribute Model for Semistructured Data,* TR 21/00 National University of Singapore, 2001.
9. W. Du, M. Li, L. Tok and W. Ling, “XML structures for relational data,” *WISE*, vol. 1, pp. 151-160, 2001.
10. C. Fahrner and G. Vossen, “A survey of database design transformations based on the Entity-Relationship model,” *Data Knowl. Eng.,* vol. 15, pp. 213-250, 1995.
11. J. Fong and S. K. Cheung, “Translating relational schema into XML schema definition with data semantic preservation and XSD graph,” *Inf. & Soft. Tech.*, vol. 47, pp. 437-462, 2005.
12. U. Hohenstein and V. Plesser, “Semantic enrichment: A first step to provide database interoperability,” *Workshop Fderierte Datenbanken*, pp. 3-17, 1996.
13. A. Maatuk, M. A. Ali, and N. Rossiter, “Relational database migration: A perspective,” in *DEXA*, vol. 5181, pp. 676-683,2008.
14. A. Maatuk, M. A. Ali, and N. Rossiter, *Relational database migration: An Evaluation,* Technical report, <http://computing.unn.ac.uk/staff/cgma2/papers/Migper.pdf>, 2008.
15. A. Maatuk, M. A. Ali, and N. Rossiter, “An integrated approach to relational database migration,” in *IC-ICT2008*, 6pp, in press, <http://computing.unn.ac.uk/staff/cgma2/papers/icict08.pdf>, 2008.
16. A. Maatuk, M. A. Ali, and N. Rossiter, *A framework for relational database migration,* Technical report, <http://computing.unn.ac.uk/staff/cgma2/papers/RDBM.pdf>, 2008.
17. E. Marcos, B. Vela and J. M. Cavero, “Extending UML for object-relational database design,” in *4th int. conf. on the unified modeling language*, vol. 2185, pp. 225-239, 2001.
18. W. J. Premerlani and M. R. Blaha, “An approach for reverse engineering of relational databases,” *Commun. ACM*, vol. 37, pp. 42-49, 1994.
19. F. Saltor, M. Castellanos and M. García-Solaco “Suitability of datamodels as canonical models for federated databases,” *SIGMOD,* vol*.* 20, pp. 44-48, 1991.
20. B. Vela and E. Marcos, “Extending UML to represent XML schemas,” *CAiSE* short paper proceedings, 2003.