

Relational Database Migration to Object Oriented Environment: A Re-engineering Approach

Tanveen Singh Bharaj¹, Vishal Pandey²

Student, Department of Computer Engineering, Jaypee Institute of Information Technology, Noida Sector - 128,
Uttar Pradesh (U.P), India¹

Student, Department of Information Technology, Maharaja Agrasen Institute of Technology, Rohini Sector - 22,
New Delhi, India²

ABSTRACT: The traditional relational databases (RDBs) have found applications in number of areas and accepted as solution for storing and retrieving data. But RDBs cannot support data persistence, complex data structures and user defined data types required by object-oriented applications. On the other hand, object-oriented technology represents real world very well. Object-oriented databases (OODBs) aim to provide security, consistence, data independence and extensible data management services to support object-oriented models and extend support to handle complex and user defined data types which RDBs couldn't handle. It focuses on data rather than procedures and gives more security to data. OODBs support object oriented programming concepts of inheritance, polymorphism and encapsulation as well as database management concepts which are Atomicity, Consistency, Isolation and Durability (ACID). It is safe from unauthorized access as it provides three access specifiers which are private, public and protected which add strict security to data in database. The current study aims at designing a automated conversion tool that takes in relational database as input and converts it to object oriented schema by applying transformational rules and finally outputs an OODB. Then finally data migration takes place from RDBs to transformed OODB.

KEYWORDS: db4o, object oriented database, reengineering, schema transformation, data migration, object-oriented schema, relation databases.

I. INTRODUCTION

Huge volumes of data collected by organizations are often stored in relational databases systems (RDBs). But nowadays the volumes of data is increasing rapidly which can't be handled by RDBs. RDBs don't have capable storage structure to handle data such as digital, images, audio/video. There is absence of identity aspect and support towards user defined complex structures in RDBs [1]. They can't operate efficiently and effectively outside their language which is structured query language (SQL). Another drawback is that information is table form and relationships between entities are represented by values. In RDB exists relationship between data manipulation language (DML) and application programming language. Large scale commercial applications are developed in languages such as C++, java. In RDB, SQL is used as DML to insert, update and delete data. A major time gets invested in transforming data between two languages. This *impedance mismatch* adds complexities in RDBs [6]. In object-oriented databases (OODBs), programming language such as C++, Java itself acts as DML. Example, OODB may use java as DML and java itself can be used to build the complete application. Single language dependence helps in removing impendence mismatch and makes application development simple. As a response to overcome these shortcomings, and meet the increasing challenges in internet, OODBs have been proposed. Object-oriented database schema is considered to be conceptual description of the objects of real world. The object-oriented data model is considered to be the heart of the

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

coming generation of databases application [4, 11]. Users want to migrate their conventional databases by applying this new technology, without losing their data stored in legacy database systems. In OODBs data is represented as objects which are identified by object identifiers (OIDs). Objects reference each other by references/association. They allow object-oriented programmers to develop the product, store them as objects and replicate or modify existing objects to make new objects within the OODBs. There is requirement of OODBs in areas such as computer-aided design, software engineering and computer-integrated manufacturing [10, 13]. All these industries use OODBs to manage complex and highly interrelated data. Advanced office automation use OODB to handle hypermedia data. The efficiency of such database is greatly improved in areas where there is requirement of storing massive amount of data for example, a hospital database. The big O notation for such a database paradigm drops from $O(n)$ to $O(1)$ thereby increasing efficiency as one don't need to scan all tuples as opposed to RDBs. OODBs have many advantages over traditional database management systems as shown in Fig.1 :

- Support reliable interaction with object-oriented languages such as Java and C++.
- Better handling of user defined and complex data types.
- Takes into consideration relationships that exists between the different objects.
- Programming can be done with small procedural differences without affecting the entire system.
- The operations defined on these kinds of system are not dependent on database application running at that moment.

Various OODBMS are - *Db4o*, *Intersystem Cache*, *Objectivity/DB* etc [8]. Among these, *Db4o* is an open source database which the current study focuses on. The current study aims to present a methodology on automating the process of creation OODBs incrementally from RDBs. The proposed approach provides semantic solution in schema transformation and data migration preserving the semantics and constraints of RDBs. The proposed methodology provides approach towards systematic migration which aims to meet the following needs:

1. Providing methodical support in schema migration of OODBs from relational one.
2. Assisting transformation process of relational schemas and databases.
3. Allowing users to extend support towards method for schema transformation.
4. Exploiting features of OODBs to the fullest.

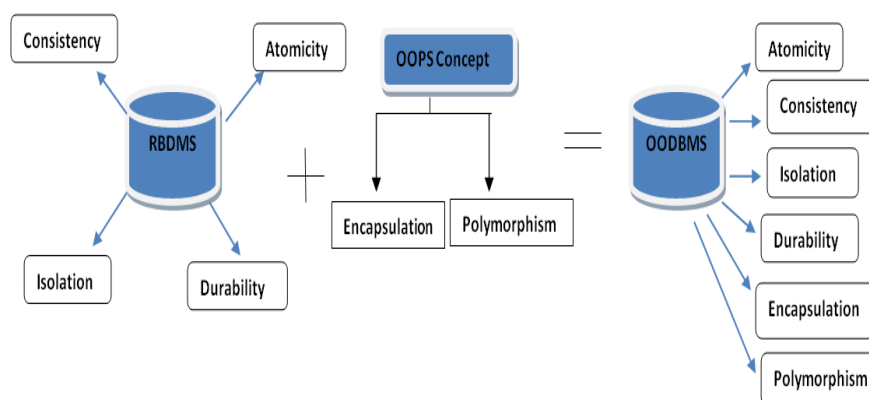


Fig. 1. Illustrates features of OODBMS

II. RELATED WORK

The transformation of RDBs to OODBs has been a topic of study over the past few decades and several approaches have been proposed so far. Most of the earlier proposed approaches doesn't transform relational schema to object one automatically. Solution proposed in Pegasus [14] simply maps each relation to a class and relation instance as the class

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

extension. The drawback of this solution is that many object oriented features are not exploited. Approaches proposed in [8] doesn't provide an automatic scheme transformation environment, the user is required to manually chose transformation rule to be applied on relational schema to outputs the corresponding object-oriented schema. This requires user needs to be well aware of database and it's constraints in order to choose a correct rule to apply for transformation. Existing approaches do not yet regard data migration to OODB to the extent it is required to be. Solution proposed in [5] rather migrates data from relational to OODB by writing records/tuples into a file. The records are written in form of SQL query, this again is tedious task as records of database of every table has to been written manually and knowledge on SQL query is required. Error in manually writing record into file may make converted database migration and integration inconsistent. Schema transformation needs knowledge on relational schema semantics, such as

- Classification of relations and it's attributes
- Referential integrity constraints or
- Primary and possible candidate keys.

This information may not be directly fetched from relational schema, therefore semantic enrichment has to be implemented [8]. Semantic enrichment is seen as an important area of *reverse engineering* which is incorporated in the current study. The current research takes into consideration entire conversion process from RDB to OODB completely involving semantic enrichment, schema transformation, database integration and migration.

Study has been carried out to investigate the approaches and techniques required in conversion of database. In solution proposed by [2] there are three significant approaches for database conversion: objects view over RDBs, data integration and migration. The technique of translation can be widely classified into major two categories:

- (i) Source-to-target technique: In this technique, physical source code is translated into an equivalent target with no Intermediate Conceptual Representation (ICR).
- (ii) Source-to-conceptual-to-target technique: In this technique, the schema is interpreted from logical to a conceptual one by recovering the domain semantics and making them explicit.

III. ENVIRONMENT FOR MIGRATING TO OBJECT ORIENTED DATABASE SYSTEMS

General approach of conversion to OODB comprise of two step process of transformation. The first one, transforms the relational schema to object-oriented schema. And second one, migrates data from RDB to OODB [7]. During the process of conversion to OODB, it is necessary to maintain complex dependencies in data which maybe present in the system. Therefore, the current study proposes an approach which has an advantage over other comparable approaches that it extends it's support to both phases whereas other related approaches emphasize more on first phase. The major advantage of proposed solution is it's feasibility as object-oriented data model is a superset of the relational model [8]. The current research is divided into three steps process as illustrated pictorially in Fig. 3:

1. *Schema Transformation and Extraction Phase* : Maps relational schema to equivalent object oriented schema based on proposed transformation rules.
2. *Instance Creation Process* : Objects are instantiated from transformed object oriented schema and stored in created OODB .
3. *Data Migration Phase* : The transformed schema and objects instantiated over it gives support to migrate data to OODB from RDB.

These three phases are implemented using proposed automation tool that uses Db4o as the object-oriented database management system (OODBMS) to store converted RDBs and java as object oriented programming language to design the tool.

The schema transformation phase, will reengineer the relational schema. The relational schema will undergo transformational to well designed and understandable target schema being object oriented schema, to adapt to new applications. The input is relational database out of which relational schema will be fetched and outputs corresponding object oriented schema and it's OODB using proposed predefined set of transformation rules. Incrementally, transformation rule will be applied (based on input relational schema) that will transform a part of the relational schema, into a so called *sub-pattern* that will corresponds to a part of the object-oriented schema as shown in Fig. 2. The output

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

object-oriented schema will combine all tables together along with their attributes. Schema transformation and extraction rules are subdivided into *class definition rule* and *attribute definition rule*. The output of applying class definition rule is creation of java class structure from relations (tables). A corresponding object-oriented java file will be created for every relational table in RDB. And attribute definition rule will define the instance variables for that particular class. Further investigation in proposed approach will take into consideration primary and foreign keys, referential integrity constraints and additional dependencies while transforming to object oriented schema. This process of extracting additional information from relational schema is known as semantic enrichment [1]. The second phase which is instance creation process, will create and instantiate objects from their object oriented schema by accessing their corresponding java file created in first phase as illustrated in Fig 4.

The third phase is, database migration. It is a process in which schema translation and data conversion of source database (that is RDBs as per current study) takes place to an equivalent database in the target environment (that is OODB). Migrating to object technology consists of reverse engineering of the application programs and migration of the database [12]. As the current study proposes to create objects out of data, database migration provides greater flexibility in terms of reengineering input relational schema to corresponding object-oriented schema. Additional, higher levels of performance is achieved as data is not required to be converted at the runtime [1].

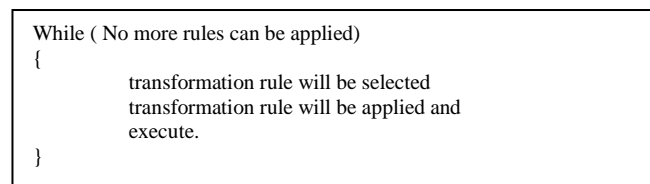


Fig. 2 Process of schema transformation

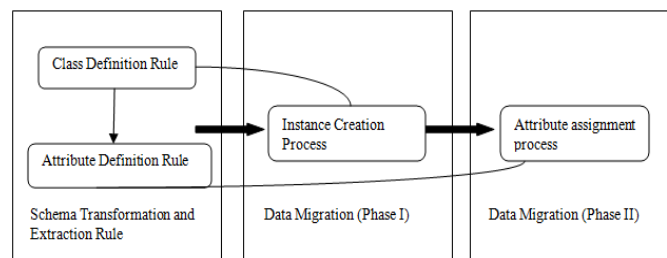


Fig. 3 Schema transformation and data migration

Data migration involves assigning values to the attributes of instances/objects created from instance creation process. These values are fetched from input RDB and migrated to objects created from second phase.

IV. THE MIGRATION ENVIRONMENT WITH AN EXAMPLE SCENARIO

The current study illustrates the proposed approach for conversion with the help of case study illustrated in Fig. 4. The proposed conversion algorithm is divided into three phases namely (i) Schema Transformation and Extraction phase (ii) Instance Creation process (iii) Data Migration phase.

Schema transformation is further divided into (a) instance definition rule and (b) attribute definition rule. The user will be queried to input it's relational schema, specifying table/relation name along with all attributes for each table in the database. The source schema is translated to semantically enriched target schema. The transformation of a source schema inputted by user to a target schema consists of two subparts.

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

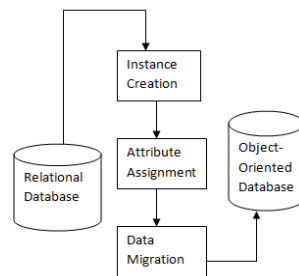


Fig. 4 Instance Creation and Data Migration

The first one, being Data-Base Reverse Engineering (DBRE), that aims to extract out the associated conceptual schema out of relational database, for example, Entity Relationship Model (ERM), expresses source schema's explicit and implicit semantics. Relation/table, attributes, keys and dependencies in data form explicit semantics. It is very important to extract out additional semantic information that are not explicitly specified in RDBs. The second one, being Data-Base Forward Engineering (DBFE), that aims to extract out the resultant physical schema from the conceptual one obtained from the first part [2]. The proposed approach translate input schema to target one without any ICR.

IV. (a) Assumption and Dependencies

The automated tool where the proposed algorithm is implemented is designed in Java using db4o as OODB. Java has become one of the important programming language because of it's portability and popularity with internet applications [3]. For consistency purposes, the RDB input by the user for conversion should be in third normal form. If two relations have primary key whose names are same, the user is required to choose one table as parent table, from which other table is derived. In Fig. 5, underlined attributes refer that they are primary key to that relation.

IV. (b)Phase I: Schema Mapping Rules

These are transformation rules that defines how a particular construct/pattern in SQL will be mapped and transformed to it's equivalent construct/pattern in OODB schema. The schema transformation is divided as:

Rule 1:Class Definition Rules

Every relation (table) of database is mapped to a new java class.

Student :	<u>ID</u>	name	dept_name	total_credits	Address	
Takes :	<u>ID</u>	<u>course ID</u>	<u>sec_ID</u>	semester	year	grade
Section :	<u>course ID</u>	<u>sec ID</u>	semester	year	room_no	
Classroom :	<u>building</u>	<u>room_no</u>	capacity			
Course :	<u>course ID</u>	title	dept_name	credits		
Department :	<u>dept ID</u>	dept_name	HOD	building		
Instructor:	<u>Instr ID</u>	name	dept_ID	salary		
Teaches:	<u>Instr ID</u>	<u>course ID</u>	<u>sec ID</u>	semester	year	
Exchange_Student :	<u>ID</u>	Country_from				

Fig. 5 University Database Schem

Pattern:

Student :	<u>ID</u>	name	dept_name	total_credits
-----------	-----------	------	-----------	---------------

Fig. 5(a) Student table

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

Referring to Fig. 5(a), student table will be mapped to Student class as given below:

Result:

```
Class Student
{ }
```

Rule 2:Class Inheritance Rule

If two tables in a database have same key (primary key), then class inheritance rule can be applied where one table will be chosen as parent table and other as child by the user. Here user intervention is required to choose the parent table out of two tables with same keys.

Pattern:

Student :	<u>ID</u>	name	dept_name	total_credits
Exchange_Student :	<u>ID</u>	Country_from		

Fig. 5(b) Student and Exchange_Student table

Referring to Fig. 5(b), if user chooses Student as the parent table then the result would be:

Result:

```
Class Student
{ }
Class Exchange_Student extends Student
{ }
```

Rule 3:Attribute Definition Rule

Attributes of relation are mapped as instance variables of that class (derived from rule 1 or rule 2).

Pattern:

Attribute Name	Datatype
ID	INT
Name	VARCHAR
Dept_name	VARCHAR
Total_credit	INT

Fig. 5(c) Student table

Referring to Fig. 5(c), student table will be mapped to Student class as given below:

Result:

```
Class Student
{
    int ID;
    String name;
    String dept_name;
    int total_credit;
}
```

Rule 4: Mapping foreign key relationship

Every foreign key relationship in relational schema is mapped to object identifier references or association attributes. If one table has a key referencing primary key of another class, then in OODB, the child class will have object of parent class as its instance variable to keep referential integrity constraint intact. The foreign key attribute is dropped from the class, leaving the class with semantically meaningful attributes and association attributes with other classes. The foreign key mapping rules have been described in rule 5 and rule 6.

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

Rule 5: Mapping many-many relationship

A relation whose attributes are a concatenation of primary keys of two other relations form many-to-many relationships or binary many-to-many relationship. If there is a table that is formed by combination of primary keys of two separate relation, the class thus formed by mapping, of such tables will contain reference to other two classes.

Pattern:

Student :	<u>ID</u>	name	dept_name	total_credits		
Takes :	<u>ID</u>	<u>course_ID</u>	sec_ID	semester	year	grade
Course :	<u>course_ID</u>	title	dept_name	credits		

Fig. 5(d) Student, Takes and Course table

Referring to Fig. 5(d), Student, Course and Takes table will be mapped as given below:
Result:

```
Class Student
{
  int ID;
  String name;
  String dept_name;
  int total_credits;
}
```

```
Class Course
{
  int course_ID;
  String title;
  String dept_name;
  int credits;
}
```

```
Class Takes
{
  int sec_ID;
  int semester;
  int year;
  String grade;
  set (Student S);
  set (Course C);
}
```

While setting objects of Student class and Course class during initialization of Takes class object, only those ID and course_ID will be accepted for insertion, which have already been set and inserted in previously initialized objects of Student and Course class.

Rule 6: Mapping one-one/ one-many relationship

A dependency between the keys of two table/relation in both directions leads to a one-to-one relationship. This is implemented by adding object identifier references of each other in both derived class.

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

Pattern:

Department :	<u>dept_ID</u>	dept_name	HOD	building
Instructor:	<u>Instr_ID</u>	name	dept_ID	salary

Fig. 5(e) Department and Instructor Table

Referring to Fig. 5(e), Department and Instructor table will be mapped class as given below, where dept_ID in Instructor table is foreign key to Department table:

Result:

```

Class Department
{
    int dept_ID;
    String dept_name;
    String HOD;
    String building;
    Instructor inst;
}

Class Instructor
{
    int Instr_ID;
    String name;
    int salary;
    Department dept;
}

```

Since every instructor belongs to a department and a department has set of instructors, therefore we need to set object reference of both classes in each other. During initialization of Department class object, while setting object of Instructor inside Department class object only those instr_ID will be accepted that have been previously assigned in objects of Instructor class. Similarly during initialization of Instructor class object, while setting object of Department class inside Instructor class object only those dept_ID will be accepted that have been previously assigned in objects of Department class.

Rule 7: Mapping composite multivalued attribute

Composite multivalued attribute tuple/row of atomic values will be transformed into a set of instance variables containing the composite attributes. Here user intervention and interaction is essential to specify datatype and name of composite attribute derived from multivalued attribute.

Pattern:

Student :	<u>ID</u>	name	dept_name	total_credits	Address
-----------	-----------	------	-----------	---------------	---------

Fig. 5(f) Student table containing address field

Referring to Fig. 5(f), student table will be mapped to Student class as given below with multivalued attribute:

Result:

```

Class Student
{
    int ID;
    String name;
    String dept_name;
    int total_credits;
    set address as (
        String street;
        String city;)
}

```


International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

While transforming Student relation to Student class, user shall be prompted about any multivalued attribute. If it exists, the user shall be required to specify which one is multivalued attribute and specify the subdivided attributes of multivalued attribute in transformed class.

E.g. In Fig. 5(f) user will be required to specify subdivided attributes of Address as String street and String city.

IV. (b) Phase II: Instance Creation Process

Every Java class file created from schema mapping rules will be compiled containing the structure of class along with the default and parameterized constructors. The files will be compiled at run time using reflection library in Java. This protects data from illegal access as only class objects can access other members of that class and only through constructors one can access and assign data values. Run time compilation of Java class is required for persisting objects of classes defined in java files. These objects are the one stored in created OODB in Db4o. This follows data migration into created OODB from RDB.

IV. (c) Phase III: Data Migration Phase

Data stored in tuples of RDB are transformed into complex objects in OODB. This involves loading relational data to the automated tool, and then inserting it into output database produced as a result of schema transformation and instance creation process. Once the object-oriented schema is ready and corresponding java files have been compiled. The user is prompted to transfer data from relational to OODB created in db4o. The user will specify the table for which he wants to migrate data, the tool will automatically generate required text fields that are equal to number of instance variables in that class. If the tool reads that input class has object reference to another class, it will automatically generate textfields for that reference class too for the user to input records. While entering the records the tool will take into consideration, no database constraint is violated. Example (considering foreign key constraint): If a class has object reference to another class, then reference object can take values that exists in the OODB for that reference class as illustrated in Fig.6, Fig. 7(a) and Fig. 7(b).

In Fig. 6. Structure of Car and Pilot class are as follows:

```
class Car
{
String model;
Pilot pilot;
}
```

```
class Pilot
{
String name;
int points;
}
```

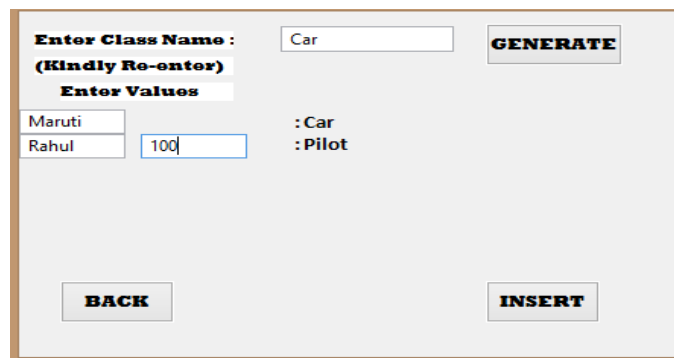


Fig. 6 Insertion of values in Car table

After insertion of values in Car class, Fig. 7(a) shows structure of Car table in OODB. Car table has reference to pilot class. Fig. 7(b) shows structure of each record inside Car table.

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

Row Id	model	pilot
1	Maruti	(G) com.db4o....
2	Honda	(G) com.db4o....

Fig. 7(a) Car table values in OODB

Field	Value	Type
com.db4o.Car	(G) com.db4o.Car	com.db4o.Car
model	Maruti	java.lang.String
pilot	(G) com.db4o.Pilot	com.db4o.Pilot
name	Rahul	java.lang.String
points	100	int

Fig. 7(b) Structure of Car table

Fig. 8 shows that values inside transformed Pilot class in OODB. From the figure it is observed that the record, entered inside Car class in Fig. 6 already had a reference in Pilot class table in Fig. 7(b). Had the record being entered in Car class inside Pilot reference was missing in Pilot table error would be generated as shown in Fig. 9. Fig. 9 generates error as reference to Pilot("Tanveen",101) is missing in Pilot table in Fig. 8. Therefore proposed data migration process not only migrates data from relational to transformed OODB, but also ensures same consistency is achieved as it was in RDB.

Row Id	name	points
1	Rahul	100
2	Brio	2000

Fig. 8 Pilot class inside OODB

V. THE ARCHITECTURE OF THE CONVERSION TOOL

The users will be provided with tool support that implements the current study. During conversion of database transformation rules are implemented depending on RDB patterns. Some migration rules require user interaction, but the current study have tried to minimize it to maximum. Therefore the user has to support conversion tool to simplify the migration process and vice-versa. The proposed conversion tool implements the following tasks:

1. Identify which rule to apply for a relational pattern
2. Applying transformation rule after identification.
3. Initialising objects for transformed pattern.
4. Visualizing the migration process.
5. Applying the migration process

These different tasks are reflected by the architecture sketched in Fig. 10. The lowest level is called *Transformation Definition* where the transformation rules have been defined. The function of the next level, *Knowledge Application*, is that it determines according to given relational pattern which transformation rule should be chosen and applied. After application of rules on entire input relational schema incrementally .java file of every relation will be created and compiled. Certain rules defined in this study also require user interaction which will be identified and prompted to user for his response in this level.

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

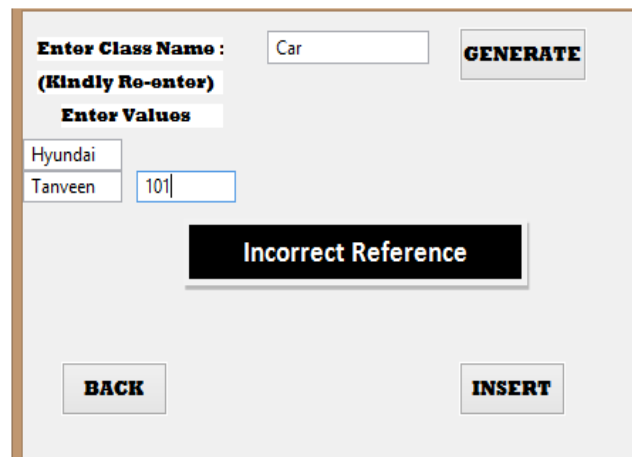


Fig. 9 Insertion of values in Car table

Migration Process Control instructs the user and records the progressing process of migration. Therefore user gets an outline of the migration technique, that is which section of the schema have been transformed or still have to be transformed. Furthermore, it enables the user to undo certain portions of migration process. Finally, the level *Object Representation* visualizes all items in newly created ODB in db4o after application of required transformation rules and completion of migration process.

VI. CONCLUSIONS AND RESULTS

The current study proposed an interactive environment for transforming relational into object-oriented database schemas with automated method for data migration to converted OODB. The user will input the source relational schema, depending on input relational pattern transformational rules are applied to get target enriched object-oriented schema. The transformation rules work on pattern to be fetched and matched from relational schema, a set of conditions and a sequence of operations to take place for transformation [8]. Such conversion can map database structure and its semantics from RDB to OODB without any information loss. The referential integrity constraints such as primary key and foreign key constraints have been rightly incorporated in object schema. The transformed schema is java class, which is compiled at run time to get .class file. Objects are initialized from .class file to construct OODB for the input RDB in db4o. Finally data is migrated to the transformed OODB. The current research is useful in incorporating semantics like generalization, association in transformed schema. Fig. 11 represents the comparative study between RDB and obtained OODB from RDB, thus the need to migrate to OODB from RDB. The contribution of current research is that:

1. Providing model for the adaptable definition of transformation rules,
2. Proposing transformation rules that utilise most of the object-oriented features
3. Ability to produce database migration programs from transformed object oriented schema.

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

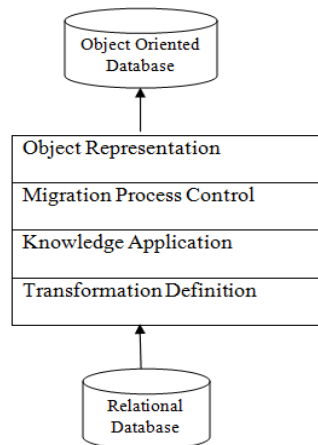


Fig. 10 Architecture of Proposed Conversion tool

Parameters	Relational Database	Created Object-Oriented Database
Support towards user datatypes	X	✓
Joins	✓	X
Handling long transactions	X	✓
Inheritance , Polymorphism & Encapsulation	X	✓
ACID (Atomicity, Consistency, Isolation & Durability)	✓	✓
Impedance Mismatch	X	✓
Speed	slower	faster

Fig. 11. Comparative study between RDB and transformed OODB

REFERENCES

- [1] Leelavathi Rajamanickam, "Empirical Study on Migrating Data From Relational Databases To Object-Oriented Technology", published International Journal Of Engineering And Computer Science ISSN:2319-7242 Volume 2 Issue 8 August, 2013 Page No.2403-2408.
- [2] Maatuk, Abdelsalam, Akhtar Ali, and Nick Rossiter. "Relational database migration: A perspective Database and Expert Systems Applications", Springer Berlin Heidelberg, 2008.
- [3] Morris Chang, Ophir Frieder and David Grossman, "The Object Behaviour of Java Object Oriented Database Management Systems", published in the Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC.02) IEEE 2002.
- [4] Reda Alhaji and Faruk Polat, "Reengineering Relational Databases to Object Oriented: Constructing the Class Hierarchy and Migrating Data", IEEE Journal,2001.pp. 1095-1350.
- [5] Joseph Fong, "Converting Relational to Object-oriented Databases", ACM Sigmod Record 26.1 (1997): 53-58.
- [6] Dr. Pushpa Suri, Meenakshi Sharma . "A Comparative study between performance of Relational and object oriented database in data warehousing" , International Journal of Database Management Systems (IJDMS), Vol.3, No.2, May 2011
- [7] Andreas Behm, Andreas Geppert, Klaus R. Dittrich, "On the migration of relational database schemas and data to object oriented database systems". published in the Proceedings 5th International Conference on Re-Technologies for Information Systems, 1997
- [8] Jens Jahnke, Wilhelm Schäfer, and Albert Zündorf. "A Design Environment for Migrating Relational to Object Oriented Database Systems." IEEE Xplore, published in the Proceedings International conference on Software Maintainance, 1996.



ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

- [9] Vinay Goyal, Amit Jain. "Reengineering of relational databases to object oriented database", International Journal of Research in Engineering and Technology, Volume: 03 Issue: 01 | Jan-2014.
- [10] Won Kim, "Research Direction in object Oriented Database systems", published in Proceeding PODS '90 Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, Pages 1-15.
- [11] William J Premerlani, Michael R Blaha, "An Approach for Reverse Engineering of Relational Databases", published in May 1994 Communications of the ACM.
- [12] Fahrner C. and G. Vossen, "A survey of database design transformations based on the entity-relationship model", Data Knowledge Engineering , 1995a, pp.21-35.
- [13] Xue Li, "A survey on Schema Evaluation in Object Oriented Databases", published in IEEE Conference on Technology of object oriented languages and systems,1999.
- [14] J. Albert, R. Ahmed, M. Ketabchi, W. Kent, and M. Shan. "Automatic importation of relational schemas in pegasus", In 3rd International workshop on research issues on data engineering: interoperability in multidatabase systems, 1993.