Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 65 (2013) 20192-20195

ISSN: 2229-712X

A novel approach for converting relational database to an object oriented database: data migration and performance analysis

Sonali P. Dhawak*, Amit Sinhal and Akansha Jain

CS &E Department, Technocrats Institute of Technology, Bhopal, India.

ARTICLE INFO

Article history:

Received: 20 November 2013;
Received in revised form:
15 December 2013;
Accepted: 24 December 2013;

Keywords

Algorithm,
Data Migration,
Object-Oriented Databases,
Reengineering of legacy Databases,
Relational Databases.

ABSTRACT

The object-oriented data model is predicted to be the heart of the next generation of database systems. Users want to move from old legacy databases into applying this new technology that provides extensibility and exhibility in maintenance. However, a major limitation on the wide acceptance of object-oriented databases is the amount of time and money invested on existing database applications, which are based on conventional legacy systems. Users do not want to lose the huge amounts of data present in conventional databases. This paper presents a novel approach to transform a given conventional database into an object-oriented database. It is assumed that the necessary characteristics of the conventional database to be re-engineered are known and available. The source of these characteristics might be the data dictionary and/or an expert in the given conventional database. We implemented a system that builds an understanding of a given conventional database by taking these characteristics as input and produces the corresponding object-oriented database as output. Finally, we handle the migration of data from the conventional database to the constructed object-oriented database.

Introduction

During the last two decades, Relational Database Management System (RDBM) has been established as the technology, handling databases up to terabytes. Relational DBMSs have been extremely successful in the market; however RDBMS lack the mechanisms to deal with complex structured data. Their tabular approach does not allow a suitable modeling of complex hierarchical objects. Most of the applications such as Geographical Information System, CAD, Multimedia, and Engineering etc. are characterized by having to manage complex, highly interrelated information, which was difficult to manage in RDBMS. To combat the limitations of RDBMS and meet the challenge of the increasing rise of the internet and the Web, programmers developed object-oriented databases in 1980 [7]. In recent years, database research has concentrated on object-oriented data models, which allow to store highly structured data. With regard to the data structuring concepts offered, an object-oriented data model can be looked upon as an extension of the nested relational model, [5] which allows to store relations as attribute values. However, the relational model only permits the alphanumeric data management.

Overview of OODBMS

An OODBMS is the result of combining object oriented programming principles with database management principles. Object oriented programming concepts such as encapsulation, polymorphism and inheritance are enforced along with regular database management concepts such as the Atomicity, Consistency, Isolation and Durability (ACID properties) which lead to system integrity, support for an *ad hoc* query language and secondary storage management systems which allow for managing very large amounts of data.. OODB [6] is a system while supporting all the functionality of a relational database system (including queries, transactions, backup and recovery mechanisms), also offers an Object oriented programming language interface, user defined data types, object identifiers and the ability to manage objects persistently. Features that are common in the RDBMS world such as transactions, the ability to handle large amounts of data, indexes, deadlock detection, backup and restoration features and data recovery mechanisms also exist in the OODBMS world. Following figure shows the features of Object oriented Database [8]. A primary feature of an OODBMS is that accessing objects in the database is done in a transparent manner such that interaction with persistent objects is no different from interacting with in-memory objects. Database operations typically involve obtaining a database root from the OODBMS which is usually a data structure like a graph, vector, hash table, or set and traversing it to obtain objects to create, update or delete from the database. When a client requests an object from the database, the object is transferred from the database into the application's cache where it can be used either as a transient value that is disconnected from its representation in the database or it can be used as a mirror of the version in the database in that updates to the object are reflected in the database and changes to object in the database require that the object is refetched from the OODBMS.

Analysis of Problem

This Paper presents an approach to transform a given relational database into an object-oriented database by mapping the relational schema to object oriented schema and migrating data from relational database to object oriented database. A System for relational to object oriented database (SRTOOD) is a interactive application that implements an interactive static schema mapping and data mapping process. It produces a machine-readable schema mapping database and generates source code for the class declarations and for automatic creation of object instances based on the mapping database. The user can then intervene and accept, modify, or reject the suggestion. Very few commercial systems try to provide object-oriented access to existing relational data. In order to achieve the objective of an application, the schema mapping process should generate a C#

Tele:
E-mail addresses: sonalidhawak@gmail.com

class that contains the class declarations corresponding to the re-engineered relational schema. Application programs can then use the C# classes to construct object oriented database and run-time data mapping system to migrate the relational database.

Proposed Work

The work consists of practical approach to map a relational database schema to object oriented database schema. After mapping relational schema to object oriented schema, the construction of database schema class creation is carried out. And finally the data from relational databases is migrated to the object oriented database system. Data mapping is concerned with the process of migrating the data from a relational database (in the form of tuples) into complex objects in an object oriented database. It provides a persistent view of the relational database by doing a one-time port of the data into a data store managed by a full-edged object-oriented database management system. It has the advantage that once the data has been ported, no dynamic data mapping is required between tuples and objects, thus removing the run-time overhead. However, unlike relational database management systems, most object-oriented database management systems do not have any loading routines. The main reason is the complexity of the data caused by embedded objects, relationships, inheritance, and so on. Therefore, efficient routines are needed for bulk loading data from a relational database into an object-oriented database.

Experimental Model

The system architecture has two major components. The first one is concerned with the mapping of the existing relational schema to an object-oriented schema. Schema mapping is a one-time static operation. The result of carrying out this schema mapping is a mapping database that contains information about the new object classes that have been generated and the manner in which these could be associated with existing relational tables. The information includes definition of object classes, associations, inheritance relationships, and aggregation relationships. Moreover, it contains additional information in order to allow creation of objects from a combination of one or moretuples from the relational database. Most of the information required for the schema mapping comes from the catalog of the relational database. An important constraint satisfied by the schema mapping presented in this dissertation is that the original relational schema is not modified in any manner.

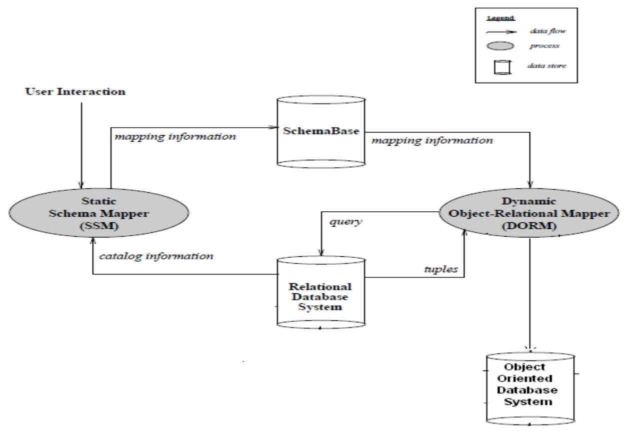

Figure 1: System Architecture

This ensures that existing relational database applications are not affected by the mapping process and thus satisfy the goals of using an object wrapper, as described in the previous section. While the first component is concerned with the schema mapping, the second component is concerned with the data mapping. Unlike schema mapping, data mapping is a dynamic process in that the data in the realational database is migrated to object oriented database in the form of an object.

Performance Analysis

The mapping between the relational tuples and the objects corresponding to the generated object schema is described.. Since there are two phases in the schema mapping process, two data mapping strategies have been specified. After the class hierarchy is constructed, it is time to migrate the current relational database contents into the corresponding object-oriented database. This is acceptedas the most important and vital step in the whole re-engineering process. Once system is developed then we will got the result. From the result, we can analyse that the requirements gets fullfilled or not. the implementation of propose experimental model consist of Elimination of 2NF relations, Elimination of widow relations, Elimination of orphan relations, Elimination of BLOB attributes, Identification of classes, Identification of associations, dentification inheritance, dentification of aggregation and Data Mapping is carried out or not.

A.Grid Data Relational Schema

For testing the computational models with our experimental model, we are going to use Grid Data Relational Schema. The Naval Environmental Oceanographic Now-vcasting System (NEONS) is a geophysical relational database designed by the Naval Oceanographic and Atmospheric Research Laboratory (Jurkevics 1992). NEONS is a large database containing a variety of data such as grid model output, satellite images, and so on. The portion of the database to be considered here corresponds to grid data. Grid data contains atmospheric and oceanographic numerical model outputs, user-defined parameters, and gridded climatology data. The collection of data values corresponding to the grid points in a rectangular grid comprises a grid eld.

B. Computational Analysis

Here we have considered three data migration models for computational analysis

1. Conceptual Model
2. Logical Model
3. Physical Model

we are going to test these models with Grid Data Relational Schema one by one.

Conceptual Model

The conceptual model has to be one of the most under utilised yet most beneficial resources in any data migration. A conceptual model works by recording very high level business objects and defining their relationships. A conceptual data model identifies the highest-level relationships between the different entities.

Logical data model

Logical data models should be based on the structures identified in a preceding conceptual data model, since this describes the semantics of the information context, which the logical model should also reect. Even so, since the logical data model anticipates implementation on a specific computing system, the content of the logical data model is adjusted to achieve certain efficiencies. The term 'Logical Data Model' is sometimes used as a synonym of 'Domain Model' or as an alternative to the domain model.

Physical Model

Physical data model represents how the model will be built in the database. A physical database model shows all table structures, including column name, column data type, column constraints, primary key, foreign key, and relationships between tables.

Data Migration Algorithm

```
Steps:
/*First pass.*/
Consider classes in order, according to their position
in the inheritance hierarchy, starting with leaves;
For every class R and its corresponding relation R do
  For every tuple r in relation R do
    If tuple r is not marked then
      Locate and mark the tuple that joins with r
      in each of the super-relations of R, if any;Construct an object
      oid in Linstances(R) to represent tuple r such that, related to oid:
      Attributes with atomic domains in
      Wattributes(R) get their values from
      the corresponding tuple r
      and the already located related tuples
      in the super- relations of R ;
      Attributes with non-atomic domains in
      Wattributes(R) are assigned the value nil;
    End If
  End For
End For
/*Second pass.*/
Consider classes in order, according to their position in the
inheritance hierarchy, starting with the direct subclasses of the
root;
For every class R and its corresponding relation R do
  For every attribute x in Lattributes(R)
  such that x has a non-atomic domain do
    If the name of x consists of the two sub-strings,
    "New..." and a relation name S then
      Let spk be the primary key of S and fk be the
      corresponding foreign key in relation R;
      For every tuple r in relation R do
        Let oid be the object that corresponds to tuple r;
        Locate tuple s, in S, that joins with r based on
        the equality of fk and spk;
        Let oid'2Winstances(R) be the object that
        corresponds to tuple s;
        Assign oid' to the value of attribute New S in oid;
      End For
    ElseIf  x expects its values to be a set of objects
    from an existing class T then
      Let U be the relation that has the same name as attribute x;
      Let rpk be the primary key of relation R and ur fk
      be the corresponding foreign key in U;
      Let tpk be the primary key of T and ut fk
      be the corresponding foreign key in U;
      For every tuple r in relation R do
        Let oid be the object that corresponds to r;
        Locate tuple(s) u, in relation U, that join with
        r based on the equality of urfk and rpk; Locate tuple(s) t,
        in T, that join with tuple(s) u, found in the previous
        step, based on the equality of ut f k and tpk;
        Assign the set of objects that correspond to the
        tuples found in the previous step to the value of attribute x in
        oid;
      End For
    Else if x expects its values to be a set of tuples of the
    form (R1, ..., Rn,a1,....,am), where n  1 ,
    m  0 and (m + n)  2 then
      Let U be the relation that has the same name as attribute x;
      Let rpk be the primary key of relation R and
      ur fk be the corresponding foreign key in U;
      Let tpki be the primary key of relation Ri
      and v,tf ki be the corresponding foreign key in U;
      For every tuple r in relation R do
        Let oid be the object that corresponds to r;
        Locate tuple(s) u, in U, that join with
        tuple r based on the equality of urfk and rpk;
        For every u located in the previous step do
          For i=1 to n do
            Locate tuple ti, in relation Ri, that joins with
            tuple u, based on the equality of ut fki and tpki;
          End For
          Construct a tuple (t1,...tn,v1,...,vm), where v1 to vm
          are the values in tuple for the non-key attributes al to am in U;
        End For
        Assign the set of tuples constructed in the previous
        step to the value of x in oid;
      End For
    End If
  End For
End For
End Algorithm.
```

Comparison between Experimental Analysis and Computational Analysis

Here proposed experimental model compared with computational model through the following table.

Table 1: Comparison At Different Schema Inputs to Proposed Method

Operations	Conceptual Model	Logical Model	Physical Model	Proposed Model
Elimination of 2NF relations	√	√	×	√
Elimination of widow relations	√	√	×	√
Elimination of orphan relations	√	√	×	√
Elimination of BLOB attributes	×	×	×	√
Identifying classes	√	√	√	√
Identifying associations	×	×	√	√
Identifying inheritance	×	×	√	√
Identifying aggregation	×	×	×	√
Data Mapping	√	√	√	√

From the above table we can justify that our proposed method based on experimental model is perform well as compared to the computational model because proposed method having the passes all the parameters in comparision with computational model model.

Conclusion

This paper implements an approach to transform a given conventional relational database into an object-oriented database. We implemented a system that builds an understanding of a given conventional database by taking these characteristics as input and produces the corresponding object-oriented databaseas output. Finally, we handle the migration of data from the conventional database to the constructed object-oriented database.

References

[1] Ramanathan, C. Providing object-oriented access to a relational database. In Proceedings of the 32nd ACM annual

southeast conference held in Tuscaloosa, Alabama, March, 1994.

[2]Andersson M. Extracting an entity-relationship schema from a relational database through reverse engineering. In Proceedings of the13th international conference on entity relationship approach held in Manchester, UK, December, 1991.

[3] Reda Alhaji and Faruk Polat, \Reengineering Relational Databases to Object Oriented : Constructing the Class Hierarchy and Migrating Data", IEEE Journal,2001.pp. 1095-1350.

[4] Abdelsalam Maatuk, Akhtar Ali, and Nick Rossiter , A Framework for Relational Database Migration published in School of Computing, Engineering & Information Sciences Northumbria University, Newcastle upon Tyne, UK in 2001

[5] Andreas Behm, Andreas Geppert, Klaus R. Dittrich2 ,On the migration of relational schemas and data to object-oriented database published in Proceedings of the 5th International Conference on Re-Technologies in Information Systems, Klagenfurt, Austria, December 1997

[6] William J Premerlani, Michael R Blaha, An Approach for Reverse Engineering of Relational Databases published in May 1994 Communications of the ACM

[7]Chia-Tien Dan Lo, Morris Chang, Ophir Frieder and David Grossman, The Object Behavior 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

[8]John Grant, Jarek Gryz, Jack Minker, and Louiqa Raschid, Logic-Based Query Optimization for Object Databases published in IEEE Transactions on knowledge and data engineering, vol. 12, no. 4, July/August 2000.

[9] Fahrner C. and G. Vossen, \A survey of database design transformations based on the entity-relationship model", Data Knowledge Engineering(15),1995a,pp.21350.

[10] Cattell R., Object data management: Object-oriented and extended relational-database systems", Addison-Wesley, 1994.

[11] Chiang,R. H. L.,T. M. Barron and V. C. Storey,\ Performance evaluation of reverse engineering relational databases into extended entity-relationship mod-els",In Proceedings of the 12th international conference on entity,1993.

[12] Johannesson P. and K. Kalman, A method for translating relational schemas into conceptual schemas", In Proceedings of the eighth international conference on entity relationship approach held in Toronto Canada,1989,pp. 27185.

[13] Andersson M., ”Extracting an entity- relationship schema from a relational database through reverse engineering”, In Proceedings of the 13th international conference on entity relationship approach held in Manchester, Volume 28, ACM Press,1994.

[14] Jeusfeld M. A. and U. A. Johnen, An executable meta model for re-engineering of database schemas", Technical Report 9419, Technical University of Aachen,1994.

[15] Fong J., “Mapping extended entity relationship model to object modelling tech-nique", SIGMOD Record 24,1995,pp. 18-22.

[16] Herzig R. and M. Gogolla, “Transforming conceptual data models into an object model", In Proceedings of the 11th international conference on entity relationship approach held in Karlsruhe, Germany,Springer,1992,pp. 28098.

[17] Blaha, W. J. Premerlani and J. E. Rumbaugh,\Relational database design using an object-oriented methodology", Communications of the ACM, 1988, pp. 41427.

[18] Keller W. and J. Coldewey, Relational database access layers A pattern language. In Pattern languages of programming design", Addison-Wesley, 1996.