ISSN:
Print - 2277 - 0593
Online - 2315 - 7461

Journal of Natural Sciences, Engineering and Technology

A DESIGN OF AN OBJECT-ORIENTED DATABASE FOR EFFECTIVE DATA MINING

***A.P. ADEWOLE AND R.O. SULAIMON**

Department of Computer Science, Federal University of Agriculture, Abeokuta, Nigeria
***Corresponding author**: philipwole@yahoo.com **Tel:** +2348034404207

ABSTRACT

This paper focuses on the design of an object-oriented database (OODB), through incorporation of object-oriented programming (OOP) concepts into existing relational databases. The proposed approach makes use of the OOP concepts namely, inheritance, encapsulation and polymorphism to design an OODB and perform classification in it respectively. Usually, database is a collection of tables with common fields. In this study, those common fields are grouped together to form a single generalized table. The newly created table resembles the base class in the inheritance hierarchy. Polymorphism allows different classes to have methods of the same name and structure, performing different operations based on the calling object. The polymorphism is specifically employed to achieve classification in a simple and effective manner while Encapsulation ensures the hiding of the data and behavior of an object behind a limited and well-described interface. In Java terms, the limited and well-described interface is the set of public methods and attributes. The use of these object oriented concepts for the design of OODB ensures that even complex queries can be answered more efficiently. Particularly, data mining task and classification can be achieved in an effective manner.

Keywords: Classification, Encapsulation, Inheritance, Object Oriented database (OODB), Object-oriented programming concepts, Polymorphism.

INTRODUCTION

In the modern computing world, the amount of data generated and stored in databases of organizations is vast and continuing to grow at a rapid pace. The data stored in these databases possess valuable hidden knowledge. The discovery of such knowledge can be very fruitful for taking effective decisions. Thus the need for developing methods for extracting knowledge from data is quite evident. Data mining, a promising approach to knowledge discovery, is the use of pattern recognition technologies with statistical and mathematical techniques for discovering meaningful new correlations, patterns and trends by analyzing large amounts of data stored in repositories.

Data mining has made its impact on many applications such as marketing, customer relationship management, engineering, medicine, crime analysis, expert prediction, web mining, and mobile computing, among others. In general, data mining tasks can be classified into two categories: Descriptive mining and Predictive mining. Descriptive mining is the process of extracting vital characteristics of data from databases. Some of descriptive mining techniques are clustering, association rule mining and sequential mining. Predictive mining is the process of deriving hidden patterns and trends from data to make predic-

tions. The predictive mining techniques consist of a series of tasks namely; classification, regression and deviation detection. One of the important tasks of data mining is Data Classification which is the process of finding a valuable set of models that are self-descriptive and distinguishable data classes or concepts, to predict the set of classes with an unknown class label. For example, in the transportation network, all highways with the same structural and behavioral properties can be classified as a class highway. From the application point of view, Classification helps in credit approval, product marketing, and medical diagnosis. So many techniques such as decision trees, neural networks, nearest neighbor methods and rough set-based methods enable the creation of classification models. Regardless of the potential effectiveness of data mining to appreciably enhance data analysis, this technology is yet to be a niche technology unless an effort is taken to integrate data mining technology with traditional database system.

Database systems offer a uniform framework for data mining by proficiently administering large datasets, integrating different data-types and storing the discovered knowledge. Over the years, relational database (RDB) has been the accepted model for efficient storage and retrieval of large amount of data. The relational database (RDB) is a type of database or database management system that stores information in tables rows and columns of data and conducts searches by using data in specified columns of one table to find additional data in another table. In a relational database, the rows of a table represent records (collections of information about separate items) and the columns represent fields (particular attributes of a record). In conducting searches, a relational database matches information from a field (attributes) in one table with information in a corresponding field of another table to produce a third table that combines requested data from both tables.

The RDB is a model that is based on tables with static components of organizational information. Additionally, RDB can handle only simple predefined data types and faces problems when dealt with complex data types, user defined data types and multimedia. These factors make RDB less efficient in handling complex information systems. Object oriented database (OODB) solve many of these problems. OODB is a database that utilizes object-oriented language concepts. Object database management system (ODBMS) provide an efficient way to the integration of database capabilities with an object oriented programming language. ODBMS makes database objects appear as programming language objects. An ODBMS extends the language with transparently persistent data, concurrency control, data recovery, associative queries, and other capabilities. ODBMS are used when there is a need for high performance on complex data. ODBMSs provide the lowest cost for development and best performance combination when using objects because they store objects on disk and have the transparent program integration with object-oriented programming languages.

Object-oriented programming (OOP) is a way of designing and coding programs. OOP is significantly different from traditional programming; it has a different way of programming structures. Instead of programs having sequences of instructions to be processed, OOP views programs as sets of data structures that have both data elements

and program instructions. Another way to understand the difference between traditional programming and OOP is that traditional programming is organized around logic first and data second, whereas OOP is organized around data first and logic second. To design a traditional program to create an order, for example, flowchart or pseudocode of the logic of the ordering process would first be developed. The data to be processed would be documented as a part of the logic.

The fact that RDB has been the accepted model and vast amount of applications have been built on the model, it's highly difficult to neglect those RDBs. Hence, we intend to incorporate the object-oriented concepts into the existing RDBMs. This effort will be better prepared to address the limitations of RDB and explore the benefits of OODB over RDB more effectively. Inheritance is one of the most significant OOP concepts. Inheritance is the ability for one entity/ object to take on the behavior (method) and characteristics (attributes) of another. Java supports inheritances through extending classes. Polymorphism is another significant object-oriented programming concepts, polymorphism refers to a programming language's ability to process objects differently depending on their data type or class. More specifically, it is the ability to redefine methods for derived *classes.* For example, given a base class shape, polymorphism enables the programmer to define different area methods for any number of derived classes, such as circles, rectangles and triangles. No matter what shape an object is, applying the area method to it will return the correct results. Encapsulation ensures the hiding of the data and behavior of an object behind a limited and well-described interface. In Java terms, the limited and well-described interface is the set of public methods and attributes. The incorporation of the object oriented programming concepts into the existing RDBMSs will be an ideal choice to design a database that best suit the advanced database applications.

The primary objective of this study is to present an innovative approach for the design of an object-oriented database system. The design of the OODB is carried out in a proficient mode with the intention of achieving efficient classification in the database by utilizing the object-oriented programming concepts: inheritance, encapsulation and polymorphism. It also reduces the implementation overhead when compared to the traditional databases.

LITERATURE REVIEW

Research papers have introduced related works such as schema translation, data type mapping, query translation and meta-data. In the schema translation, Blaha(1994) discussed converting object oriented Models into RDBMS schema. Shim *et al.*(1999) designed a schema mapping gateway for accessing RDB through an OODB interface.

In the data type mapping, DeFazio and Srinivasan (1996) extended RDBMS for handling complex domains. The approach required applications to integrate layered products from independent software vendors (ISVs) to deliver specific functionality for a domain. In the query translation, Yu (1995) discussed the translation from object-oriented queries to relational queries. He proposed a method that used the extended OODB predicate graphs and relational predicate graphs to facilitate the query translation.

In the meta-data, Lim and Shin(1999) used

meta-data and mapping libraries to extend ODBMSs by using metaclasses. It could turn traditional classes into normal objects, which could receive messages containing attributes and methods that described the database's structure and behavior.

The main advantage of OODB is its ability to represent real world concepts as data models in an effective and presentable manner Rajan and Saravanan (2008). OODB is optimized to support object-oriented applications, different types of structures including trees, composite objects and complex data relationships. The OODB system handles complex databases efficiently and it allows the users to define a database, with features for creating, altering, and dropping tables and establishing constraints (Kelly Nunn-Clark *et al.,* 2003).

From the user's perception, OODB is just a collection of objects and inter-relationships among objects. Those objects that resemble in properties and behavior are organized into classes. Every class is a container of a set of common attributes and methods shared by similar objects. The attributes or instance variables define the properties of a class while the method describes the behavior of the objects associated with the class.

A class/subclass hierarchy is used to represent complex objects where attributes of an object itself contains complex objects (Kitsana *et al.,* 2004). The most important object-oriented concept employed in an OODB model includes the inheritance mechanism and composite object modeling Xue Li (1999).

RESEARCH METHODOLOGY

Database is bound to contain a number of tables with common fields with each of the table represented as classes in OOP. In the proposed approach, such common set of fields are grouped together to form a single generalized table. The newly created table resembles the base class in the inheritance hierarchy. The presentation of classes in hierarchy is one of the eminent OOP concepts. By employing polymorphism, different classes have methods of the same name and structure, performing different operations based on the calling object. The polymorphism and encapsulation are specifically used to achieve classification in a simple and effective manner.

The object oriented programming concepts namely inheritance, polymorphism and encapsulation used to design the OODB ensure that even complex queries can be answered more effectively, particularly the data mining tasks (classification) can be achieved effectively.

Suppose T denote a set of all tables on a database D and *t* contained in T, where 't' represents the set of tables in which some fields are common. Now we create a generalized table composing of all those common fields from the table set 't'. In order to show the efficiency of the proposed approach, a traditional database (relational) sample that consist of three tables was created using MySql wamp server. The tables consist of common fields and table specific fields. These three tables are used to best illustrate the OOP concepts employed in this study. The three tables are namely: Shop owner, Customers and Suppliers.

Table 1: Shop owner

Owner ID	Contact name	Age	Gender	Post Held	Place ID	City	Region	Postal Code	Country	Phone Number
23	Mariam	36	Female	Sales representative	Ikeja	Lagos	South south	01	Nigeria	08054672345
24	Oluwatosin	30	Male	Assignment manager	Bodija	Ibadan	South west	02	Nigeria	08134567678
21	Olawale	31	Male	CEO	Beere	Ibadan	South west	02	Nigeria	08096543421

Table 2: Customers

Customer ID	Company Name	Contact Name	Age	Gender	Marital Status	Post Held	Place ID	City	Region	Postal Code	Country	Phone Number
1	Mastercom computers	Raheem	24	Male	Unmarried	Sales representative	Ikeja	Lagos	South south	1	Nigeria	080
2	Princeton technology	Raheem	37	Male	Married	Manager	Gariki	Abuja	North east	9	Nigeria	080
3	IT world	Sulaimon	35	Male	Married	Sales representative	Ikeja	Lagos	South south	1	Nigeria	080
4	Gateway computers	Adebayo	40	Male	Married	CEO	Guguhalada	Abuja	North east	9	Nigeria	080

Table 3: Suppliers

Supplier ID	Company Name	Age	Gender	Marital Status	Post Held	Place ID	City	Region	Postal Code	Country	Phone Number
2	Princeton technology	34	Male	Married	Sales representative	Ikeja	Lagos	South west	02	Nigeria	08056167
5	Mastercom computers	34	Male	Married	Manager	Ikeja	Lagos	South west	02	Nigeria	08034678

The above set of tables can be represented equivalently in classes.

From the above table structure, it is understood that every table has a set of general or common fields and table-specific fields. On considering the customers table, it has general fields like name, age, gender, place ID etc. and table-specific fields like contact title, company name etc. These general fields occur repeatedly in most tables. This causes redundancy and thereby increases space complexity. Moreover, if a query is given to retrieve a set of information for the whole organization satisfying a particular rule, there may be a need to search all the tables separately. So, this replication of general fields in the table leads to a poor design which affects effective data classification.

In this study, an OODB is designed by utilizing the inheritance concept of OOP by which the problem of redundancy is eliminated. Firstly, all the general or common fields from the table set 't' are located. Then, all these general or common fields are fetched and stored in a single table so that all the related tables can inherit it. Thus the generalized table resembles the base class of the OOP paradigm. For instance, a new table is created called 'Person', which contains all the common fields and the other tables like Shop owners, Customers, Suppliers that contain only table specific fields then inherit the Person table without redefining it. Generalization depicts an "is-a" relation and composition represents a "has-a" relation. The generalized table "Person" contains all the common fields and the tables "Shop owners, Suppliers and Customers" inheriting the table "Person" is said to have an "is-a" relationship with the table Person i.e., a Shop owner is a Person, A Supplier is a Person and A Customer is a Person. Similarly to exemplify the composition relation, the table Person contains an object reference of the "Places" table as its field. Then the table Person is said to have a "has-a" relationship with the table Places i.e., a Person has a place and similarly, A Place has a postal code.

The generalized table 'Person' is considered as the base class 'Person' and the fields are considered as the attributes of the base class 'Person'. Therefore, the base class 'Person', which contains all the common attributes, is inherited by the other classes namely Shop owners, Suppliers and Customers, which contain only the specialized attributes. Moreover, inheritance allows the creation of a generalized methods in the base class and specialized methods in the sub classes. For example, if there is a need to get the contact numbers of all the people associated with the organization, a method getShopownerNumebrs() can be defined in the base class 'Person' and it can be shared by its subclasses. In addition, the generalized class 'Person' exhibits composition relationship with another two classes 'Places' and 'PostalCodes'. The class 'Person' uses instance variables, which are object references of the classes 'Places' and 'PostalCodes'. The tables in the proposed OODB design are shown below;

Table 4: Persons

Person ID	Contact Name	Age	Gender	Marital Status	Place ID	Phone Number
1	Raheem	34	Male	Married	Ikeja	08056788945
3	Oluwatosin	24	Male	Unmarried	Bodija	08067894532
4	Mohammed	37	Male	Married	Cape town	09123456789
8	Juliet	28	Female	Unmarried	Gariki	08033125678

Table 5: Customers

Customer ID	Computer Name	Post Held
23	Mastercom computer	Manager
25	IT World	Assistant Manager
12	Info Web	Sales Representative
26	Genius computer	CEO

Table 6: Suppliers

Supplier ID	Company Name	Contact Title
20	Mastercom computer	Manager
23	IT World	Sales Representative
10	Princeton technology	CEO
11	Info Web	Assistant Manager

Table 7: Shop owner

Owner ID	Post Held
1	Sales Representative
2	CEO
4	Assistant Manager
5	Manager

Table 8: Places

Place ID	Street	Postal Code
Ikeja	Awolowo way	01
Gariki	Gariki road	09
Bodija	Ajibola	02
Cape town	Mandela way	55780

Table 9: Postal codes

Postal Code	City	Country
01	Lagos	Nigeria
02	Ibadan	Nigeria
09	Abuja	Nigeria
55780	Cape town	South Africa

Owing to the incorporation of inheritance concept in the proposed design, the database can be extended by effortlessly adding new tables, by merely inheriting the common fields from the generalized table.

Dynamic polymorphism allows us to define methods with the same name in different classes and the method to be called is decided at runtime based on the calling object. Here, a single method can do the classification process for all the tables. We can also access the method, specifically for individual entities namely Shop owner, Customers and Suppliers.

IMPLEMENTATION AND RESULT

In this section, the experimental result of the proposed approach is presented. The proposed approach for the design of OODB and classification has been programmed in JAVA using Netbean java application interface with MySql (wamp server) as database.

The comparative result of the proposed approach and the traditional approach is shown below:

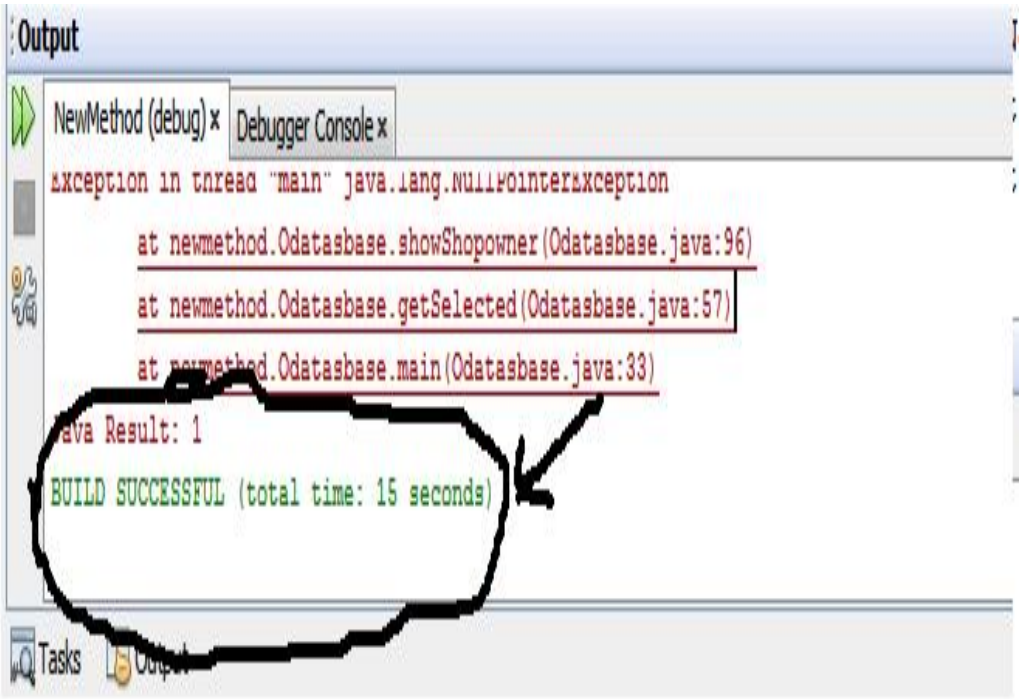

Figure 1: Implementation time for OODB

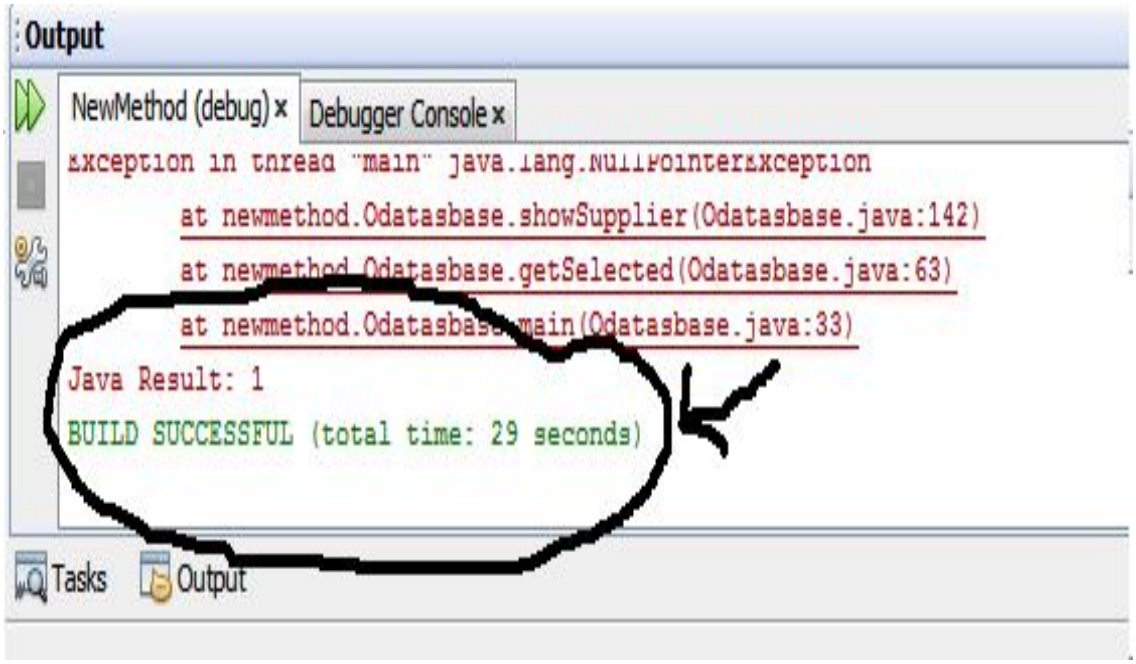

Figure 2: Implementation time for Existing RDB

Figures 1 and 2 show the time taken to carry out a particular data classification (execute query) in the proposed approach as compared to when performed in an existing relational database. While OODB takes 15 seconds to execute RDB takes 29 seconds.

This depicts a decrease of 35% in the implementation overhead .

CONCLUSION

The primary objective of this study is to present an innovative approach for the design of an object-oriented database system. This study has shown that relational database model concepts can be incorporated into OODB models. The design of the OODB is carried out in a proficient mode with the intention of achieving efficient classification in the database by utilizing the object-oriented programming concepts: inheritance, encapsulation and polymorphism. The approach reduces the implementation overhead when compared to the traditional databases.

REFERENCES

Addison-Wesley 2000. An Introduction to Database Systems, Seventh Edition, Boston, MA.

Blaha M. 1994. Converting OO Models into RDBMS Schema, IEEE Software, pp. 28-39.

Codd E.F. 1970. A relational model of data for large shared data banks. *Commun. ACM,* 13(6): 377–387.

Darabant A.S. 2005. A New Approach In Fragmentation Of Distributed Object Oriented Databases Using Clustering Techniques, Studia Univ. babes, L(2).

DeFazio S., Srinivasan, J. 1996. Database Extensions for Complex Domains, Proc. Of IEEE Data Engineering, pp. 200-202.

Kelly Nunn-Clark, Lachlan Hunt, Teo Meng Hooi and Balachandran Gnanasekaraiyer 2003. Problems of Storing Advanced Data Abstraction in Databases, In Proceedings of the First Australian Undergraduate Students' Computing Conference, pp. 59-64.

Kitsana Waiyamai, Chidchanok Songsiri and Thanawin Rakthanmanon 2004. Object-Oriented Database Mining: Use of Object Oriented Concepts for Improving Data Classification Technique, Lecture Notes in Computer Science, 3036: 303-309.

Lim J., Shin D. 1999. A Methodology of Constructing Canonical Form Database Schemas in a Multiple Heterogeneous Database Environment, *Journal of Database Management,* 9(4).

Rajan, J., Saravanan, V. 2008. "Vertical Partitioning in Object Oriented Databases Using Intelligent Agents, International *Journal of Computer Science and Network Security,* 8(10).

Shim J., Scheuermann, P., Vingralek, R. 1999. Dynamic Caching of Query Results for Decision Support Systems, In Proceedings of the 11th International Conference on Scientific and Statistical Database Manage-

ment, Cleveland, Ohio, USA.

Xue Li. 1999. A Survey of Schema Evolution in Object- Oriented Databases, Technology of Object-Oriented Languages and Systems, pp. 362-371, Nanjing, China.

Yu C. 1995. Translation of Object-Oriented Queries to Relational Queries, Proc. of IEEE on Data Engineering, pp. 90-97.

(Manuscript received: 29th November, 2010; accepted: 22nd December, 2010).