USE OF OBJECT-ORIENTED CONCEPTS IN DATABASES FOR EFFECTIVE MINING

[1]***Ajita Satheesh,***
[1]*UIT, RGPV, Bhopal, MP,India*
[1]*ajitargpv@yahoo.co.in*

[2]***Dr. Ravindra Patel,***
[2]*UIT,RGPV, Bhopal, MP,India*
[2]*ravindra@rgtu.net*

ABSTRACT-Data mining is a process that uses a variety of data analysis tools to discover knowledge, patterns and relationships in data that may be used to make valid predictions. With the popularity of object-oriented database systems in database applications, it is important to study the data mining methods for object-oriented databases. The traditional Database Management Systems (DBMSs) have limitations when handling complex information and user defined data types which could be addressed by incorporating Object-oriented programming concepts into the existing databases. Classification is a well-established data mining task that has been extensively studied in the fields of statistics, decision theory and machine learning literature. This paper focuses on the design of an object-oriented database, through incorporation of object-oriented programming concepts into existing relational databases. In the design of the database, the object-oriented programming concepts namely inheritance and polymorphism are employed. The design of the object-oriented database is done in such a well equipped manner that the design itself aids in efficient data mining. Our main objective is to reduce the implementation overhead and the memory space required for storage when compared to the traditional databases.

Keywords:* *Data mining, Classification, Relational Database Management System (RDBMS), Object-Oriented Database (OODB), Object-Oriented Programming (OOP).concepts, Inheritance, Polymorphism, Generalization.

I. 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 [1]. 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 [2]. 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 [8]. 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 predictions. The predictive mining techniques consist of a series of tasks namely Classification, Regression and Deviation detection [3], [4].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 [13], [26]. For example, in the transportation network, all highways with the same structural and behavioral properties can be classified as a class highway [10]. From the application point of view, Classification helps in credit approval, product marketing, and medical diagnosis [27]. So many techniques such as decision trees, neural networks, nearest neighbor methods and rough set-based methods enable the creation of classification models [11]. Regardless of the potential effectiveness of data mining to appreciably enhance data analysis, this technology still to be a niche technology unless an effort is taken to integrate data mining technology with traditional database system [39]. Database systems offer a uniform framework for data mining by proficiently administering large datasets, integrating different data-types and storing the discovered knowledge.

For over a decade, Relational databases (RDB) has been the accepted solution for efficient storage and retrieval of huge volumes of data [37]. The RDBs are based on tables which are static components of organizational information. In addition to this, RDB can handle only simple predefined data types and faces problems when dealt with complex data types, user defined data types and multimedia [23]. Thus, the RDB technology fails to handle the needs of complex information systems [25]. Often the semantics of relational databases are left unexplored within many relationships which cannot be extracted without users' help. Object-Oriented databases (OODB) solve many of these problems. Based on the concept of abstraction and generalization, object oriented models capture the semantics and complexity of the data. [37]. Therefore, many research organizations are employing object-oriented database (OODB) to solve their problems of data storage, retrieval and processing [18].

OODB is a database in which the concepts of object-oriented languages are utilized. The principal strength of OODB is its ability to handle applications involving complex and interrelated information [25]. But in the current scenario, the existing object-oriented database management system (OODBMS) technologies are not efficient enough to compete in the market with their relational counterparts. Apart from that, there are numerous applications built on existing relational database management systems (RDBMS). It's difficult, if not impossible, to move off those RDBs. Hence, we intend to incorporate the object-oriented concepts into the existing RDBMSs, thereby exploiting the features of RDBMSs and OO concepts. Undoubtedly, one of the significant characteristic of object-oriented programming is inheritance [5]. Inheritance is the concept by which the variables and methods defined in the parent class (super class) are automatically inherited by its child class (sub class) [38]. There are two ways to represent class relationships in object-oriented programming and they are "is a" and "has a" relationships. . In an "is-a" relationship, an object of a sub class can also be thought of as an object of its super class [36], [17]. For instance, a class named Car exhibits an "is a" relationship with a base class named Vehicle, since a car is a vehicle.

In a "has-a" relationship which is also known as composition [40], a class object holds one or more object references as data members. For example, a bicycle has a steering wheel and, in the same way, a wheel has spokes [36]. Inheritance can also be stated as generalization, because the "is-a" relationship represents a hierarchy between the classes of objects [38]. In generalization hierarchies, the data members and methods of the super class are inherited by the subclasses and the objects of the subclass can use up those common properties of the super class without redefinition [10]. For example, a "fruit" is a generalization of "apple", "orange", "mango" and many others. Similarly, one can consider fruit to be an abstraction of apple, orange, etc. Conversely, since apples are fruits (i.e., an apple **is-a** fruit), apples are bound to contain all the attributes common for a fruit. This concept of generalization is very powerful, because it reduces redundancy and maintains integrity [10].

Polymorphism is another important Object oriented programming concept. It is a general term which stands for 'many forms'. Polymorphism in brief can be defined as "one interface, many implementations". It is a property of being able to assign a different meaning or usage to something in different contexts – in particular, to allow an entity such as a variable, a function, or an object to take more than one form [12]. Polymorphism is different from method overloading or method overriding [40]. In literature (e.g., [41]), polymorphism can be classified into three different kinds namely: pure, static, and dynamic. Pure polymorphism refers to a function which can take parameters of several data types. Static polymorphism can be stated as functions and operators overloading. Dynamic polymorphism is achieved by employing inheritance and virtual functions. Dynamic binding or runtime binding allows one to substitute polymorphic objects for each other at run-time. Polymorphism has a number of advantages. Its chief advantage is that it simplifies the definition of clients, as it allows the client to substitute at run-time, an instance of one class for another instance of a class that has the same polymorphic interface [12].

It is becoming increasingly important to extend the domain of study from relational database systems to object-oriented database systems and probe the knowledge discovery mechanisms in object-oriented databases, because object-oriented database systems have emerged as a popular and influential setting in advanced database applications. The fact that standards, are still not defined for OODBMSs as those for relational DBMSs and as most organizations have their information systems based on a relational DBMS technology, 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.

A novel and innovative approach for the design of an object-oriented database is presented in this paper. The design of the OODB is carried out in a proficient mode with the intention of achieving efficient classification on the database. In our proposed approach, we utilize the object-oriented programming concepts: inheritance and polymorphism to achieve the above stated goals. Chiefly, we extend the existing relational databases by incorporating the object-oriented programming concepts, to attain an object-oriented database. The OODB is structured mainly by employing the class hierarchies of inheritance. The inheritance or the class relationships namely "is-a" and "has-a" are used to represent a class hierarchy in the proposed OODB. Another object- oriented programming concept, Polymorphism is utilized to achieve better classification. Polymorphism enables the usage of simple SQL queries to classify the designed OODB. The experimental results stated, portrays the efficiency of the proposed approach. The designed OODB demands less implementation overhead and saves considerable memory space compared to RDBs while exploiting its essential features.

The rest of the paper is organized as follows; Section 2 gives a concise description about OODB. In Section 3, the proposed approach to the design of OODB for effective classification is presented in detail. Section 4 summarizes the results of our experiments and the conclusions are summed up in Section 5.

II. OBJECT-ORIENTED DATABASE (OODB)

The chief advantage of OODB is its ability to represent real world concepts as data models in an effective and presentable manner [15]. 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 [18]. From the user's perception, OODB is just a collection of objects and inter-relationships among objects [14]. Those objects that resemble in properties and behavior are organized into classes [16]. 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 [9]. The method describes the behavior of the objects associated with the class [20]. A class/subclass hierarchy is used to represent complex objects where attributes of an object itself contains complex objects [26].

The most important object-oriented concept employed in an OODB model includes the inheritance mechanism and composite object modeling [24]. In order to cope with the increased complexity of the object-oriented model, one can divide class features as follows: simple attributes - attributes with scalar types; complex attributes - attributes with complex types, simple methods - methods accessing only local class simple attributes; complex methods - methods that return or refer instances of other classes [19]. The object-oriented approach uses two important abstraction principles for structuring designs: classification and generalization. Classification is defined as an abstraction principle by which objects with similar properties are grouped into classes defining the structure and behavior of their instances. Generalization is an abstraction principle by which all the common properties shared by several classes are organized into a single super class to form a class hierarchy [7].

From the very outset of the first OODBMS Gemstone [21] in the mid-eighties, a dozen other commercial OODBMSs have joined the fierce competition in the market [22]. Regarding the applications of OODB, its vendors have laid their focus on Computer Aided Design (CAD), Computer Aided Manufacturing (CAM) and Computer Aided Software Engineering (CASE). All these user applications are meant to handle complex information and the OODB systems promises to propose efficient solutions to these problems. Factory and office automation are other application areas of object-oriented database technology [25].

III. NEW APPROACH TO THE DESIGN OF OBJECT ORIENTED DATABASE

In general literature, defines three approaches to build an ODBMS: extending an object-oriented programming language (OOPL), extending a relational DBMS, and starting from scratch .The first approach develops an ODBMS by encompassing to an OOPL persistent storage to achieve multiple concurrent accesses with transaction support. The second is an extended relational approach; an ODBMS is built by incorporating an existing relational DBMS with object-oriented features such as classes and inheritances, methods and encapsulations, polymorphism and complex objects. The third approach aims to revolutionize the database technology in the sense that an ODBMS is designed from the ground up, as represented by UniSQL and OpenODB . In our design, we employ the second approach which extends the relational databases by utilizing the OOP concepts.

The proposed approach makes use of the OOP concepts namely, inheritance and polymorphism to design an OODB and perform classification in it respectively. Normally, database is a collection of tables. Hence when we consider a database, it is bound to contain a number of tables with common fields. In our approach, we group together such common set of fields to form a single generalized table. The newly created table resembles the base class in the inheritance hierarchy. This ability to represent classes in hierarchy is one of the eminent OOP concepts. Next we employ another important object-oriented characteristic dynamic polymorphism, where different classes 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. The use of these object-oriented concepts for the design of OODB ensures that even complex queries can be answered more efficiently. Particularly the data mining task, classification can be achieved in an effective manner.

Let T denote a set of all tables on a database D and $t \subset T$, where 't' represents the set of tables in which some fields are in common. Now we create a generalized table composing of all those common fields from the table set '*t*'. To portray the efficiency of our proposed approach, we consider a traditional ERP package. An ERP database for large business organizations will have a number of tables but to best illustrate the OOP concepts employed in our approach, we concentrate on three tables namely, Employees, Suppliers and Customers. The tables are represented in Figure 1.

Employees

Employee ID	Contact Name	Age	Gender	Marital Status	Title	Of Court	Birth Date	Hire Date	Place ID	City	Region	Postal Code	Country	Home Phone	Extension
1	Priya	25	Female	Married	Sales Representative	Ms.	08-Dec-1984	01-May-2006	Shahpura	Bhopal	MP	462016	INDIA	0755-2560941	5467
2	Sudeep	37	Male	Married	Vice President, Sales	Dr.	19-Feb-1972	14-Aug-1996	Piplani	Bhopal	MP	462021	INDIA	0755-2725539	3457

(a)

Customers

Customer ID	Company Name	Contact Name	Age	Gender	Marital Status	Birth Date	Contact Title	Place ID	City	Region	Postal Code	Country	Home Phone	Fax
1	Cadbury's	Seema	22	Female	Married	02-Sep-1987	Sales Representative	Shahpura	Bhopal	MP	462016	INDIA	0755-2725461	0755-5286754
2	Top'n Town	Alka	24	Female	Married	12-Aug-1985	Owner	Piplani	Bhopal	MP	462021	INDIA	0755-5234567	0755-5234568

(b)

Suppliers

Supplier ID	Company Name	Contact Name	Age	Gender	Marital Status	Birth Date	Contact Title	Place ID	City	Region	Postal Code	Country	Phone	Fax	Home Page
3	Cadbury's	Sunita	33	Female	Married	12-Oct-1976	Sales Representative	Piplani	Bhopal	MP	48104	INDIA	0755-2546709		
28	Top'n Town	Sujata	43	Female	Married	21-May-1967	Sales Representative	Shahpura	Bhopal	MP	74000	INDIA	-755-5234567		

(c)

Figure 1: Sample Tables in ERP database (a) Employees Table (b) Customers Table (c) Suppliers Table

The above set of tables namely Employees, Suppliers and Customers can be represented equivalently as classes. The class structure may look like as in Figure 2.

Class Employees	**Class Suppliers**	**Class Customer**	
Char Employee ID;	Char SupplierID;	Char CustomerID;	
Char ContactName;	Char CompanyName;	Char CompanyName;	
Int Age;	Char Name;	Char Name;	
Char Gender;	Int Age;	Int Age;	
Char Marital Status;	Char Gender;	Char Gender;	
Char Title;	Char MaritalStatus;	Char MaritalStatus;	
Char Title of Courtesy;	Char BirthDate;	Char BirthDate;	
Char Birth Date;	Char ContactTitle;	Char ContactTitle;	Common Attributes
Char Hire Date;	Char PlaceId;	Char PlaceId;	
Char Place ID;	Char City;	Char City;	
Char City;	Char Region;	Char Region;	
Char Region;	Char PostalCode;	Char PostalCode;	
Char Postal Code;	Char Country;	Char Country;	
Char Country;	Char Phone;	Char Phone;	
Char HomePhone;	Char Fax;	Char Fax;	
Char Extension;	Char HomePage;		

Figure 2: Class Structure of Employees, Suppliers and Customers Table

From the above class structure, it is understood that every table has a set of general or common fields (highlighted ones) and table-specific fields. On considering the Employee table, it has general fields like Name, Age, Gender etc. and table-specific fields like Title, HireDate 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 records 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. To perform better classification, we design an OODB by incorporating the inheritance concept of OOP.

A. DESIGN OF THE OODB

First in our proposed approach, we design an OODB by utilizing the inheritance concept of OOP by which we eliminate the problem of redundancy. First, we locate all the general or common fields from the table set 't'. Then, all these general or common fields are fetched and stored in a single table and all the related tables can inherit it. Thus the generalized table resembles the base class of the OOP paradigm. In our approach, we create a new table called 'Person', which contains all those common fields and the other tables like Employees, Customers inherit the Person table without redefining it.

Here, we have used two important mechanisms namely generalization and composition. Generalization depicts an "is-a" relation and composition represents an "has-a" relation. Both these relationships can be best illustrated as below: The generalized table "Person" contains all the common fields and the tables "Employees, Suppliers and Customers" inheriting the table "Person" is said to have an "is-a" relationship with the table Person i.e., an Employee 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 an "has-a" relationship with the table Places i.e., a Person has a place and similarly, A Place has a Postal Code.

Figure 3 represents the inheritance class hierarchy of the proposed OODB design. In the following pictured design, the small triangle (—▷) represents "is-a" relationship and the arrow (→) represents "has-a" relationship.

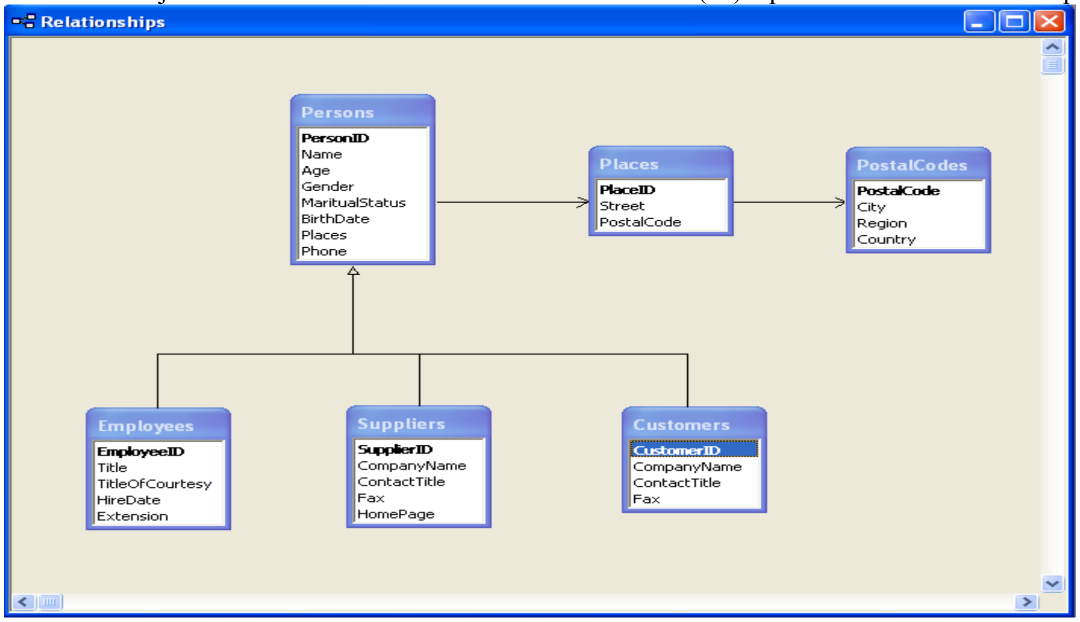

Figure 3: Inheritance hierarchy of classes in the proposed OODB design

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 Employees, Suppliers and Customers, which contain only the specialized attributes. Moreover, inheritance allows us to define the 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, we can define a method getContactNumebrs() 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 in Figure 4.

Persons

PersonID	Contact Name	Age	Gender	MaritalStatus	Birth Date	PlaceId	Phone
1	Aradhna	42	Female	Married	11-Mar-1967	Assam	0234-2725461
2	Susheela	24	Female	Married	12-Aug-1985	Bombay	0423-5234567
3	Mitesh	23	Male	Married	29-Apr-1986	Manipur	0688-2768879
4	Mohnish	41	Male	Married	16-May-1968	Gwalior	0675-2546890

(a)

Employees

Employee ID	Title	Title Of Courtesy	Hire Date	Extension
37	Sales Representative	Ms.	01-May-2006	5467
38	Vice President, Sales	Dr.	14-Aug-1996	3457

Employees				
Employee ID	**Title**	**Title Of Courtesy**	**Hire Date**	**Extension**
39	Sales Representative	Ms.	01-Apr-2006	3355

(b)

Suppliers				
Supplier ID	**Company Name**	**Contact Title**	**Fax**	**Home Page**
46	Toyota	Purchasing Manager	0245-2678543	
47	Top'n Town	Order Administrator	0755-2765489	
48	Cadbury's	Sales Representative	0675-5234786	
49	Optel	Marketing Manager	0755-2456034	

(c)

Customers			
Customer ID	**Company Name**	**Contact Title**	**Fax**
1	Ashok Leyland	Sales Associate	
2	Top'n Town	Owner	
3	Hindustan Uniliver Ltd.	Marketing Manager	

(d)

Places		
PlaceID	**Street**	**PostalCode**
Assam	Dibrugarh	567012
Bombay	Thane	230001
Bhopal	Bairagarh	462016
Gwalior	Mata Mandir	560015

(e)

PostalCodes			
PostalCode	**City**	**Region**	**Country**
462001	Bhopal	SE	INDIA
429014	Indore	CEN	INDIA
234016	Agra	NE	INDIA
456234	Gwalior	CEN	INDIA

(f)

Figure 4: Structure of tables in the proposed OODB design (a) Persons (b) Employees (C) Suppliers (d) Customers (e) Places (f) Postal Codes

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

B. DATA MINING IN THE DESIGNED OODB

Dynamic polymorphism or late binding 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. This OOP concept and simple SQL queries can be used to perform classification in the designed OODB. Here, a single method can do the classification process for all the tables. The uniqueness of our concept is that the classification process can be performed by using simple SQL query while the existing classification approaches for OODB employ complex techniques such as decision trees, neural networks, nearest neighbor methods and more. We can also access the method, specifically for individual entities namely Employees, Suppliers and Customers. By integrating the polymorphism concept, the code is simpler to write and easier to manage. As a result of the designed OODB, the task of classification can be carried out effectively by using simple SQL queries. Thus in our approach by incorporating the OOP concepts for designing the OODB, we exploit the maximum advantages of OOP and also the task of classification is performed more effectively.

IV. IMPLEMENTATION AND RESULTS

In this section, we have presented the experimental results of our approach. The proposed approach for the design of OODB and classification has been programmed in JAVA with MS Access as database. We have considered only three tables for experimentation. But in general, an organization may have a number of tables to manage. Specifically, the number of records is enormous in each table. The incorporation of the OOP concepts to such databases greatly reduced the implementation overhead incurred. Moreover, the memory space occupied is reduced to a great extent as the size of the table increases. These are some of the eminent benefits of the proposed approach. We have performed a comparative analysis of the space utilized before and after generalization of tables and thus we have computed the saved memory space. The comparison is performed with varying number of records in the tables such as 1000, 2000, 3000, 4000 and 5000 and the results are stated below in Figure 5.

Normalized						**Un Normalized**		
	Tables	**Fields**	**Records**	**Total Records of Table**	**Memory size of the table**	**Fields**	**Total Records of the table**	**Memory size of the table**
1	Customers	4	1000	4000	40000	15	15000	150000
2	Employees	5	1000	5000	50000	16	16000	160000
3	Suppliers	5	1000	5000	50000	16	16000	160000
4	Persons	8	3000	24000	240000			
5	Places	3	500	1500	15000			
6	Postalcodes	4	250	1000	10000			
Total				40500	405000		47000	470000

Saved Memory (KB): 63.47656

Normalized						**Un Normalized**		
	Tables	**Fields**	**Records**	**Total Records of Table**	**Memory size of the table**	**Fields**	**Total Records of the table**	**Memory size of the table**
1	Customers	4	2000	8000	80000	15	30000	300000
2	Employees	5	2000	10000	100000	16	32000	320000
3	Suppliers	5	2000	10000	100000	16	32000	320000
4	Persons	8	6000	48000	480000			
5	Places	3	1000	3000	30000			
6	Postal codes	4	500	2000	20000			
Total				81000	810000		94000	940000

Saved Memory (KB): 126.9531

	Tables	**Fields**	**Records**	**Total Records of Table**	**Memory size of the table**	**Fields**	**Total Records of the table**	**Memory size of the table**
1	Customers	4	3000	12000	120000	15	45000	450000
2	Employees	5	3000	15000	150000	16	48000	480000
3	Suppliers	5	3000	15000	150000	16	48000	480000
4	Persons	8	9000	72000	720000			
5	Places	3	1500	4500	45000			
6	Postal codes	4	750	3000	30000			
Total				121500	1215000		141000	1410000

Saved Memory (KB):190.4297

	Tables	**Fields**	**Records**	**Total Records of Table**	**Memory size of the table**	**Fields**	**Total Records of the table**	**Memory size of the table**
1	Customers	4	4000	16000	160000	15	60000	600000

2	Employees	5	4000	20000	200000	16	64000	640000
3	Suppliers	5	4000	20000	200000	16	64000	640000
4	Persons	8	12000	96000	960000			
5	Places	3	2000	6000	60000			
6	Postal codes	4	1000	4000	40000			
Total				162000	1620000		188000	1880000

Saved Memory (KB):253.9063

	Tables	**Fields**	**Records**	**Total Records of Table**	**Memory size of the table**	**Fields**	**Total Records of the table**	**Memory size of the table**
1	Customers	4	5000	20000	200000	15	75000	750000
2	Employees	5	5000	25000	250000	16	80000	800000
3	Suppliers	5	5000	25000	250000	16	80000	800000
4	Persons	8	15000	120000	1200000			
5	Places	3	2500	7500	75000			
6	Postal codes	4	1250	5000	50000			
Total				202500	2025000		235000	2350000

Saved Memory (KB):317.3828

Figure 5: The results of comparative analysis

The graphical representation of the results is illustrated in Figure 6. From the graph, it is clear that the saved memory space increases, as the number of records in each table increases.

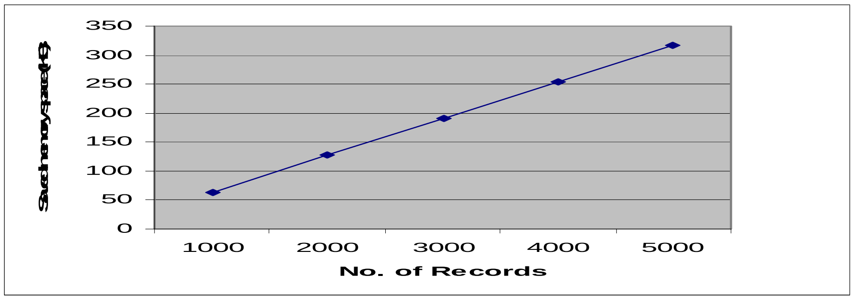

Figure 6: Graph demonstrating the above evaluation results

Moreover in the proposed approach, we have placed the common methods in the generalized class and entity-specific methods in the subclasses. Because of this design, we have saved a considerable memory space. For instance, in case of a traditional database if a method getContactNumbers() is defined to get the contact numbers, the method has to be defined in all the classes and all those results are to be combined to obtain the final result. But in the proposed approach, we have generalized all the classes, so the redefinition of methods for all the related classes is not needed. If there are 'n' classes, placing the common methods in the base class can save a memory space of

$$(n-1)\sum_{i=1}^{m} memory\ \ size\ \ of\ \ i^{th}\ \ method$$

Where 'm' is the number of common methods in the super class.

V. CONCLUSION

Data mining has been gaining tremendous interest and hence research on data mining has mushroomed within the last few decades. A promising approach for managing complex information and user defined data types is by incorporating object-orientation concepts into relational database management systems. In this paper, we have presented an approach for the design of an object-oriented database and performing classification effectively in it. The

Object oriented programming concepts such as inheritance and polymorphism has been utilized in the presented approach. Owing to this design of OODB, an efficient classification task has been achieved by utilizing simple SQL queries. The experimental results have demonstrated the effectiveness of the presented approach. Our approach has successfully reduced the implementation overhead incurred in the design of an OODB. Our approach has also reduced the amount of memory space inquired for storing databases that grow in size.

REFERENCES

[1]. Satchidananda Dehuri, "Genetic Algorithms For Multi-Criterion Classification And Clustering In Data Mining", International journal of computing and information sciences, Vol. 4, No. 3, pp. 143-154, 2006.

[2]. Jeffrey W. Seifert, "Data Mining: An Overview", CRS Report for Congress, 2004.

[3]. J. Han, M. Kamber, "Data Mining: Concepts and Techniques", Morgan Kaufmann Publishers, 2001.

[4]. F. Coenen, Leng, P., Goulbourne, G., "Tree Structures for Mining Association Rules," Journal of Data Mining and Knowledge Discovery, Vol. 15, pp. 391-398, 2004.

[5]. Ulrich Frank, "Delegation: An Important Concept for the Appropriate Design of Object Models," Journal of Object-Oriented Programming, Vol. 13, No. 3, pp. 13-18, 2000.

[6]. J. Robin. "Inductive Object-Oriented Logic Programming", In Proceedings of the second workshop on Implementation Technology for Computational Logic Systems of the European Network of Excellence in Computational Logic, pp.56-68, 2003

[7]. Georg Gottlob, Michael Schrefl and Brigitte Rock, "Extending Object-Oriented Systems with Roles", ACM Transactions on Information Systems, Vol. 14, No. 3, pp. 268–296, July 1996.

[8]. Hsinchun Chen, Sherrilynne S. Fuller, Carol Friedman, and William Hersh, "Knowledge Management, Data Mining, and Text Mining In Medical Informatics", Chapter 1, pages: 3-34, Springer,2005.

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

[10]. Babak Ameri Shahrabil, Wolfgang Kainz, "An Implementation Approach for Object-oriented Topographic Databases using Standard Tools," In proceedings of Eleventh International Symposium on Computer-Assisted Cartography, pp. 103-112, 30 October-1 November, Tehran, Iran, 1993.

[11]. Ahmed Sultan Al-Hegami, "Classical and Incremental Classification in Data Mining Process," International Journal of Computer Science and Network Security, Vol. 7 No.12, 2007.

[12]. Claudia Pon, Luis Olsina, Máximo Prieto, "A Formal Approach to Building a Polymorphism Metric in Object-Oriented Systems," In proceedings of 4th International ECOOP Workshop on Quantitative Approaches, 2000.

[13]. Jing Zhong, Yan Fu and Jun-lin Zhou, "A Classification Approach Based on Evolutionary Neural Networks," International Journal of Computational Intelligence Research, Vol.2, No. 1, pp. 72-75, 2006.

[14]. L. Yaolin, M. Molenaar and Ai Tinghua, "Frameworks for Generalization Constraints and Operations Based on Object-Oriented Data Structure in Database Generalization," In Proceedings of the 20th International Cartographic Conference, Vol. 3, pp. 2000-2012, Beijing, China, 2001.

[15]. Rajan John and Dr. V. Saravanan, "Vertical Partitioning in Object Oriented Databases Using Intelligent Agents," International Journal of Computer Science and Network Security, Vol.8, No.10, 2008.

[16]. Sikha Bagui, "Achievements and Weaknesses of Object-Oriented Databases," Journal of Object Technology, Vol. 2, No. 4, pp. 29-41, 2003.

[17]. Woochun Jun and Le Gruenwald, "A Class Hierarchy Concurrency Control Technique in Object-Oriented Database Systems," In proceedings of the 3rd Joint Conference of Information Sciences, pp. 293-296, March 1997.

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

[19]. A.S. Darabant, "A New Approach In Fragmentation Of Distributed Object Oriented Databases Using Clustering Techniques," Studia Univ. babes, Vol. L, No. 2, 2005.

[20]. M.B.Thuraisingham, "Mandatory Security in Object-Oriented Database Systems," In Proceedings of the 4th International Conference on Object-Oriented Programming Systems, Languages, and Applications, pp. 203–210, October 1989.

[21]. "Gemstone Database Management System" from http://www.gemstone.com/.

[22]. J. Blakeley, "Object-oriented database management systems," Tutorial at SIGMOD, Minneapolis, MN, May 1994.

[23]. Neal Leavitt, "Whatever Happened to Object-Oriented Databases?" IEEE Computer Society, Vol. 33, No. 8, pp. 16-19, 2000.

[24]. Shermann Sze-Man Chan, and Qing Li, "Supporting Spatio-Temporal Reasoning in an object-Oriented Video Database System", 1999.

[25]. Mansaf Alam, Siri Krishan Wasan, "Migration from Relational Database into Object Oriented Database," Journal of Computer Science, Vol. 2, No. 10, pp. 781-784, 2006.

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

[27]. Micheline Kamber, Lara Winstone, Wan Gong, Shan Cheng and Jiawei Han, "Generalization and Decision Tree Induction: Efficient Classification in Data Mining," In Proceedings of International Workshop Research Issues on Data Engineering, pp. 111-120, 7-8 April, Birmingham, UK, 1997.

[28]. Linna Li, Bingru Yang, Faguo Zhou, "A Framework for Object-Oriented Data Mining", In proceedings of fifth International Conference on Fuzzy Systems and Knowledge Discovery, Vol: 2, pp: 60-64, 2008.

[29]. Al-Jadir. L., "Encapsulating classification in an OODBMS for data mining applications", in Proceedings of the Seventh International Conference on Database Systems for Advanced Applications, pp:100-106, China, 2001.

[30]. Vladimir Novacek, "Data Mining Query Language for Object-Oriented Database", Lecture Notes in Computer Science, Vol: 1475, February 19 1998.

[31]. Jiawei Han, Shojiro Nishio, Hiroyuki Kawano, Wei Wang, "Generalization-based data mining in object-

oriented databases using an object-cube model", Data and Knowledge Engineering, vol:25, pp:1-2, 1998.

[32]. Lina Al-Jadir, "Integrating Association Rule Mining Algorithms with the F2 OODBMS", Lecture Notes in Computer Science, vol: 2736, pp: 724-736, 2003.

[33]. Beat Wuthrich, Kamalakar Karlapalem, "Data Mining Opportunities in Very Large Object Oriented Databases," ACM-SIGMOD workshop on Research Issues on Data Mining, 1996.

[34]. [Dai, H. "An object-oriented approach to schema integration and data mining in multiple databases", Technology of Object-Oriented Languages, pp: 294 - 303, 1997.

[35]. Erik Odberg, "Category Classes: Flexible Classification and Evolution in Object-Oriented Databases", Lecture Notes in Computer Science, Vol: 811, pp: 406-420, 1994.

[36]. Harvey Deitel and Paul Deitel, "Java™ How to Program", Prentice Hall, Fifth Edition, 2003.

[37]. Joseph Fong, "Converting Relational to Object-Oriented Databases," SIGMOD Record 1997, Vol. 26, No. 1, 1997.

[38]. B.G. Geetha, V. Palanisamy, K. Duraiswamy and G. Singaravel, "A Tool for Testing of Inheritance Related Bugs in Object Oriented Software," Journal of Computer Science, Vol. 4, No. 1, pp. 59-65, 2008.

[39]. Amir Netz, Surajit Chaudhuri, Jeff Bernhardt, Usama Fayyad, "Integration of Data Mining and Relational Databases," In Proceedings of the 26th International Conference on Very Large Databases, pp. 719-722, Cairo, Egypt, 2000.

[40]. Sierra, Kathy; Bert Bates (2005). Head First Java, 2nd Ed.. O'Reilly Media, Inc.. ISBN 0596009208.

[41]. Benlarbi, S. and Melo, W. Polymorphism measures for early risk prediction, In International Conference of Software Engineering (ICSE'99), Los Angeles, CA,