IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN : 2278-0661, p-ISSN : 2278-8727
PP 18-22
www.iosrjournals.org

A Overview on OODB Technology

Bhushan M. Borhade[1], Dr. Abhijit Banubakode[2]
[1]Department of Computer technology, Rajarshi Shahu College of Engineering, Pune, India.
[2]Department of Information Technology, Rajarshi Shahu College of Engineering, Pune, India

Abstract: *OODB technology an acronym for the 'object oriented database' technology is a database management system that supports the modelling and creation of the data as objects. The evolution of the OODB concepts arise from the easy and efficient way of accessing the object using the popular concepts of OOPS i.e.'Encapsulation and Inheritance '.The objective of the paper is to have the overview of the OODB, its comparison with RDBMS such as schema, class hierachy, intergrity constraints, query processing ,etc and explore the areas for improvement in OODB for efficient performance.*

Keywords: *Object oriented database, Relational Database, Object oriented relational database*

I. Introduction

OODB began developing in the mid-80's out of a necessity to meet the requirements of applications beyond the data processing applications, which characterized relational database systems (fourth-generation database technology)[1]. The need to perform complex manipulations on existing databases and a new generation of database applications like CAD(computer aided design) and CAM(computer aided manufacturing),knowledge based systems generated a need that would be better satisfied by object-oriented databases (OODBs) [2]. The object oriented database have richer data model than the table oriented RDBMS.

If we integrate database capabilities with object programming language capabilities, the result is an object-oriented database management system or ODBMS. Object oriented concepts provide closer and direct manipulation of the real world problems and database is required for the concurrent and persistence sharing of the information in applications.[12]

There are a number of commercial OODBs. These include Gemstone from Servio Corporation, ONTOS from ONTO& ObjectStore from Object Design, Inc., Objectivity/DB from Objectivity, Inc., Versant from Versant Object Technology, Inc., Matisse from Intellitic International (France), Itasca (commercial version of MCC's ORION prototype) from Itasca Systems, Inc.,02 from 02 Technology (France)[3]. These products all support an object-oriented data model. Specifically, they allow the user to create a new class with attributes and methods have the class inherit attributes and methods from superclasses, create instances of the class each with a unique object identifier, and retrieve the instances either
individually or collectively, and load and run methods.

A few vendors are now offering database systems that combine relational and object-oriented capabilities in one database system such **ORDBMS** (object oriented relation database) which is encountered most commonly in available products.

II. Object Oriented Database Model (OODB)

In OODB, any real world entity is represented by one and only one concept that is 'Object'.

An **object** is a component of a program that knows how to perform certain actions and how to interact with other elements of the program. Objects are the basic units of object-oriented programming.
Each object is uniquely identified by a **system-defined** identifier (**OID**). A simple example of an object would be a person. Logically, you would expect a person to have a name. This would be considered a **property** of the person. You could also expect a person to be able to do something, such as walking or driving. This would be considered a **method** of the person.

The OODB paradigm is based on a number of basic concepts, namely object, identity, class,abstraction inheritance, overriding, and late binding [4]
A Class can be thought of as objects with the same properties and behavior are grouped together. An object can be an instance of only one class or an instance of several classes.

Abstraction is the process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics. In object-oriented programming, abstraction is one of the most central principles (along with encapsulation and inheritance). Through the process of abstraction, a programmer hides all but the relevant data about an object in order to reduce complexity and increase efficiency.
Polymorphism is the ability to appear in many forms. In object-oriented programming, *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. Polymorphism is considered to be a requirement of any true object-oriented programming language (OOPL)
In object oriented paradigms two main concepts are the Encapsulation and Inheritance.

Inheritance is roughly called reuse. Inheritance can be that new object may be created by extending an existing object. The term Inheritance can be understood as described next. An object has a data part and a program part. All objects that have the same attributes for the data part and same program part are called as class. Classes are arranged such that some class may inherit the attributes and program part from some other classes. Since inheritance makes it possible for different classes to share the same set of attributes and methods, the same program can be run against objects that belong to different classes.

Encapsulation is the packing of data and functions into a single component. The features of encapsulation are supported using classes in most object-oriented programming languages. It allows selective hiding of properties and methods in an object by building an impenetrable wall to protect the code from accidental corruption. This means that the users cannot see the inside of the object 'capsule' but can use the object by calling the program part of the object.

III. Architecture of OODB

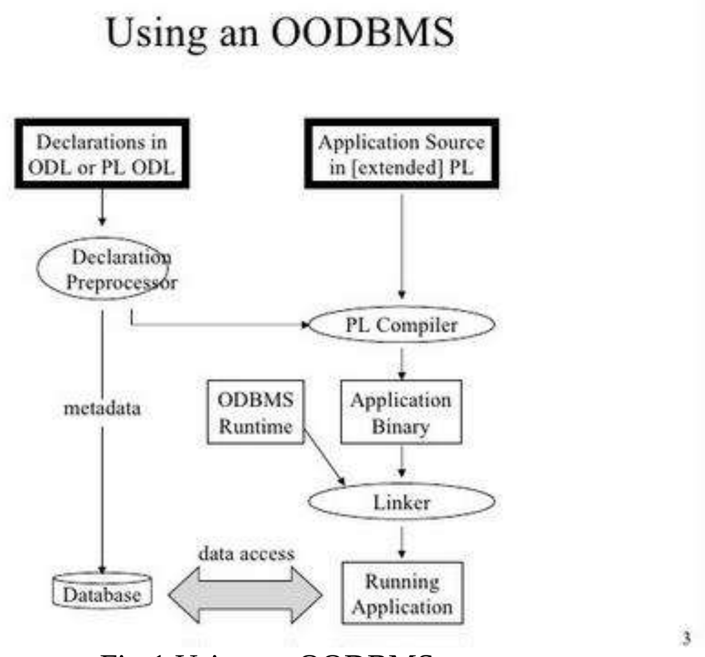

Fig:1 Using an OODBMS

Object-oriented technologies in use today include object-oriented programming languages such as C++ and Smalltalk, object-oriented database systems, object-oriented user interfaces (e.g., Macintosh and Microsoft window systems, Frame and Interleaf desktop publishing systems), etc. OODB working can be better explained from above figure 1.

Object Definition Language (ODL) is the specification language defining the interface to object types conforming to the ODMG (Object Data Management Group) Object Model. Often abbreviated by the acronym ODL. The flow of OODB is explained as above.

IV. Comparison Between OODB and RDBMS

There are concepts in the relational database model that are similar to those in the object database model. A relation or table in a relational database can be considered to be analogous to a class in an object database. A tuple is similar to an instance of a class but is different in that it has attributes but no behaviors. A column in a tuple is similar to a class attribute except that a column can hold only primitive data types while a class attribute can hold data of any type. Finally classes have methods which are computationally complete (meaning that general purpose control and computational structures are provided while relational databases typically do not have computationally complete programming capabilities although some stored procedure languages come close.

Few points are mentioned for the comparison between OODBMS and RDBMS

4.1 Advantages of using an OODBMS over an RDBMS

1. **Composite Objects and Relationships**: Objects in an OODBMS can store an arbitrary number of atomic types as well as other objects. It is thus possible to have a large class which holds many medium sized classes which themselves hold many smaller classes, ad infinitum. In a relational database this has to be done either by having one huge table with lots of null fields or via a number of smaller, normalized tables which are linked via foreign keys. Having lots of smaller tables is still a problem since a join has to be performed every time one wants to query data based on the "Has-a" relationship between the entities. Also an object is a better model of the real world entity than the relational tuple with regards to complex objects. The fact that an OODBMS is better suited to handling complex, interrelated data than an RDBMS means that an OODBMS can outperform an RDBMS by ten to a thousand times depending on the complexity of the data being handled.

2. **Class Hierarchy:** Data in the real world is usually having hierarchical characteristics. The ever popular Employee example used in most RDBMS texts is easier to describe in an OODBMS than in an RDBMS. An Employee can be a Manager or not, this is usually done in an RDBMS by having a type identifier field or creating another table which uses foreign keys to indicate the relationship between Managers and Employees. In an OODBMS, the Employee class is simply a parent class of the Manager class.

3. **Circumventing the Need for a Query Language**: A query language is not necessary for accessing data from an OODBMS unlike an RDBMS since interaction with the database is done by transparently accessing objects. It is still possible to use queries in an OODBMS however.

4. **No Impedance Mismatch:** In a typical application that uses an object oriented programming language and an RDBMS, a significant amount of time is usually spent mapping tables to objects and back. There are also various problems that can occur when the atomic types in the database do not map cleanly to the atomic types in the programming language and vice versa. This "impedance mismatch" is completely avoided when using an OODBMS.

5. **No Primary Keys:** The user of an RDBMS has to worry about uniquely identifying tuples by their values and making sure that no two tuples have the same primary key values to avoid error conditions. In an OODBMS, the unique identification of objects is done behind the scenes via OIDs and is completely invisible to the user. Thus there is no limitation on the values that can be stored in an object.

6. **One Data Model:** A data model typically should model entities and their relationships, constraints and operations that change the states of the data in the system. With an RDBMS it is not possible to model the dynamic operations or rules that change the state of the data in the system because this is beyond the scope of the database. Thus applications that use RDBMS systems usually have an Entity Relationship diagram to model the static parts of the system and a separate model for the operations and behaviors of entities in the application. With an OODBMS there is no disconnection between the database model and the application model because the entities are just other objects in the system. An entire application can thus be comprehensively modeled in one UML diagram.

4.2 Disadvantages of OODBMS over RDBMS

1. Source of immaturity of most of the current OODBs products is the lack of basic features that users of database systems have become accustomed to and therefore have come to expect. The features include a full nonprocedural query language (along with automatic query optimization and processing), [5] views, authorization, dynamic schema changes, and parameterized performance tuning. Besides these basic features, RDBs offer support for triggers, meta-data management, constraints such as UNIQUE and NULL - features that most OODBs do not support.

2. **Schema Changes**: In an RDBMS modifying the database schema either by creating, updating or deleting tables is typically independent of the actual application. In an OODBMS based application modifying the schema by creating, updating or modifying a persistent class typically means that changes have to be made to the other classes in the application that interact with instances of that class. This typically means that all schema changes in an OODBMS will involve a system wide recompile. Also updating all the instance objects within the database can take an extended period of time depending on the size of the database.

3. **Language Dependence**: An OODBMS is typically tied to a specific language via a specific API. This means that data in an OODBMS is typically only accessible from a specific language using a specific API, which is typically not the case with an RDBMS.

4. **Lack of Ad-Hoc Queries**: In an RDBMS, the relational nature of the data allows one to construct ad-hoc queries where new tables are created from joining existing tables then querying them. Since it is currently not possible to duplicate the semantics of joining two tables by "joining" two classes then there is a loss of flexibility with an OODBMS. Thus the queries that can be performed on the data in an OODBMS are highly dependent on the design of the system.

V. Future Scope

There is a lot of future work or scope for the oodb to become a full fledged database system.

Following are some of the major work area where OODB can improve the performance.

1. Need to support authorization i.e. allow the users lo grant and revoke privileges to read or change the tuples in the tables or views they created to other users, or to change the definition of the relations they created to other
2. Some of the current OODBs require the users to explicitly set and release locks whereas RDBs automatically set and release locks in processing query and update statements the users issue. This automatic set and release can be implemented in OODBS

VI. Conclusion

Along with the minimal query optimization [5] [9] for OODBs to make a major impact on the database market, following has to be done:

1. OODBs have to be made full-fledged database systems, sufficiently compatible with RDBs – a migration path is needed to allow the coexistence and the gradual migration from the current products to new products;
2. Application development tools and database access tools have to be developed for such database systems;
3. Architectures of the RDBs and OODBs have to be unified;
4. The data models of the RDBs and OODBs have to be unified.

References:

[1] Bertino, E., Negri, M., Pelagatti, G., and Sbattella, L., “Object-Oriented Query Languages: The Notion and the Issues,” IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No. 3, 1992.
[2] Khoshafian, S. Object-Oriented Databases, New York: John Wiley & Sons, 1993.
[3] Kim, W., “Object-Oriented Database Systems: Promises, Reality, and Future,” in Modern Database Systems: The Object Model, Interoperability and Beyond, pp. 255-280, Kim, W., ed., ACM Press, Addison Wesley, 1995.
[4] Brown, A.W., Object-Oriented Databases, Applications in Software Engineering. New York: McGraw-Hill, 1991.
[5] Abhijit Banubakode, “Optimization of Queries in the Object-Oriented Database Involving Path Expression”
[6] Dare Obasanjo “AN EXPLORATION OF OBJECT ORIENTED DATABASE MANAGEMENT SYSTEMS “
[7] Sikha Bagui, “Achievements and Weaknesses of Object-Oriented Databases”
[8] Won Kim, “Object-Oriented Database Systems: Promises, Reality, and Future”
[9] E.Bertino arid A.Quarati “An Approach to Support Method Invocations in Object-Oriented Queries “dipartimento di Mathematical - Univeresita di Genova
[10] Cruz, I.F., DOODLE: A visual language for object-oriented databases. In Proc. ACM SIGMOD Int. Conf. On Management of Data, pp. 71-80, 1992.
[11] Su, S. Y., Ranka, S., & He, X. (2000). Performance analysis of parallel query processing algorithms for object-oriented databases. Knowledge and Data Engineering, IEEE Transactions on, 12(6), 979-996.
[12] Abhijit Banubakode and Haridasa Acharya “Query Optimization In The Object-oriented Database Using Equi-join” Advances in Computational Sciences and Technology Print: ISSN 0973-6107, Online ISSN 0974-4738 Volume 4 Number 1 (2011) pp. 83–94 © Research India Publications.
[13] Ivan Bayross,SQL,PL/SQL The programming language of oracle,BPB Publication

ABOUT AUTHORS

Mr. Bhushan M. Borhade currently pursuing a ME degree in computer science technology from Pune university. Completed B.E degree in information Technology studies from Pune Institute of Computer Technology under the Savitribai Phule university,Pune Jun 2014.He had experience of one year in IT company.His area of interest is transaction integrityin OODB.

Dr. Abhijit Banubakode received Ph.D. degree in Computer Studies from Symbiosis Institute of Research and Innovation (SIRI), a constituent of Symbiosis International University (SIU), Pune, India in April 2014 and ME degree in Computer Engineering from Pune Institute of Computer Technology(PICT), University of Pune ,India in 2005 and BE degree in Computer Science and Engineering from Amravati University, India, in 1997. His current research area is Query Optimization in Compressed Object-Oriented Database Management Systems (OODBMS). Currently he is working as Professor and Head of Department (HOD) in Department of Information Technology, Rajarshi Shahu College of Engineering,Pune, India. He is having 16 years of teaching experience. He is a member of International Association of Computer Science and Information Technology (IACSIT), ISTE, CSI and presented 12 papers in International journal and conference.