



## Perspectives of Object Orientation in a Database Management System

**Bunakiye R. J. & Acheme D. I.**  
Department of Computer Science  
Edo University, Iyamho, Nigeria  
E-mails: [jbunakiye@gmail.com](mailto:jbunakiye@gmail.com), [ijeggs@gmail.com](mailto:ijeggs@gmail.com),  
Phones: 08061324564, 08062889197;

### ABSTRACT

An object-oriented database management system (OODBMS) is a database system which supports an object-oriented data model. Just as any traditional database system, it must provide disk management, data sharing, data integrity, security, and a query language. Further, in support of an object-oriented data model, it must manage complex objects with object identity, support objects that encapsulate data and behaviour, structure objects in classes, and organize classes in a hierarchy. Having attracted reasonable maturity in the research community and the business community, this paper looks at object oriented database concepts and how far it has improved upon the relational database management system (RDBMS, and also examines data modeled in the object oriented database environments using the unified modeling language (UML).

**Keywords:** Object-Oriented Data Model, Application Semantics, Complex Data Structures, Embedded Properties

---

### ISTEAMS Cross-Border Conference Proceedings Paper Citation Format

Bunakiye, R.J & Acheme, D.I. (2017): Perspectives of Object Orientation in a Database Management System. Proceedings of the 9th iSTEAMS Multidisciplinary Conference, University of Ghana, Legon, Accra Ghana. Pp 431-436

---

### 1. INTRODUCTION

A database system is a collection of stored data together with their description (the database) and a hardware/software system for their reliable and secure management, modification and retrieval. A database is supposed to represent the interesting semantics of an application (the mini world) as completely and accurately as possible. The relational model is the basis of many commercial relational DBMS products (e.g., DB2, Informix, Oracle, and Sybase) and the structured query language (SQL) as widely accepted standard for both retrieving and updating data. The basic relational model is simple and mainly views data as tables of rows and columns. The types of data that can be stored in a table are basic types such as integer, string, and decimal. Relational DBMSs have been extremely successful in the market. However, the traditional RDBMSs are not suitable for applications with complex data structures or new data types for large, unstructured objects, such as computer aided design/manufacturing (CAD/CAM), Geographic information systems, multimedia databases, imaging and graphics. The RDBMSs typically do not allow users to extend the type system by adding new data types.

They also only support first-normal-form relations in which every column must be atomic, i.e., no sets, lists, or tables are allowed inside a column. Due to the new needs in database systems, a number of researches for object oriented database management systems (OODBMS) have begun in the early 80. An OODBMS is the result of combining object oriented programming principles with database management principles. The interest in this paper is centred on the meeting point between relational database management systems and the object oriented database management system and the additional functionalities provided by the OODBMS in the processes of database management and control. In the context of OODBMS, objects are user defined complex data types, they are abstract representation of some real world entity that has a unique identity, embedded properties, and the ability to interact with other objects and act upon itself' (text).



It is a software construct that encapsulates state and behaviour. An object, which has structure or state (variables) and methods (behaviour/operations) can interact with other objects to create a system. An object state is the set of values that the object's attributes have at any given time. To change the objects state, you must change the values of the object's attributes. To change the value of the object's attributes, send a message to the object. The message will invoke a method; the code that performs a specific operation on the object's data.

## 2. MOTIVATION AND OPEN PROBLEMS

Object-oriented database technology is a synergy of object-oriented programming and database technologies. Both programming and database concepts have come together to provide object-oriented databases. The most significant characteristic of object-oriented database technology is that it combines object-oriented programming with database technology to provide an integrated application development system. There are many advantages to including the definition of operations with the definition of data: The defined operations apply ubiquitously and are not dependent on the particular database application running at the moment. The data types can be extended to support complex data such as multimedia by defining new object classes that have operations to support the new kinds of information. Inheritance allows one to develop solutions to complex problems incrementally by defining new objects in terms of previously defined objects. Polymorphism and dynamic binding allow one to define operations for one object and then to share the specification of the operation with other objects.

Some advantages of OODBMS include support for more semantic information, support for complex objects, extensibility of data types, versioning, reusability, and speed of application development and application. However the challenge is that the properties of the synergy are only approximate equivalences. The properties in OODBS are actually not applicable in RDBMS and vice versa. Although there are great advantages of using an OODBMS over an RDBMS, some disadvantages do exist. Some identified disadvantages are lack of theoretical foundation, lack of business data design and management tools, steep learning curve, low market presence, and lack of compatibility between different OODBMSs

## 3. OBJECT STRUCTURE AND ORIENTATION

The object, being an abstract representation of a real world entity with unique identity and embedded properties has the ability to interact with other objects and self. It contains, attributes involving called instance variables, and domain instances, it also contains object state involving object values at any given time. The state (current value) of a complex object may be constructed from other objects (or other values) by using certain type constructors. The object structure is such that the constructors contain basic atom, and tuple types; other set collection type are list, bag and arrays. Object orientation is a set of design and development principles, based on autonomous computer structures known as objects. The object oriented contribution areas are; programming languages, graphical user interfaces, and database design operating systems. A consideration of the features of both object-oriented systems and database management systems has led to a definition of an object-oriented database, which has distinguishable mandatory, optional and open features. The features, which must be present to ensure accurate and well-articulated object orientation in a database management system are described in figure 1.



Features of OODBMS from General databases	Features of OODBMS from Object Oriented databases
<p>Orthogonal Persistence of data</p> <p>Able to handle large databases</p> <p>Controlled Concurrency</p> <p>Restoring or data Recovery</p> <p>Query facility on adhoc basis</p>	<p>Construction of complex Objects</p> <p>identity of an object</p> <p>Feature of Classes and types</p> <p>Property of encapsulation</p> <p>Property of Inheritance</p> <p>Property of overriding combined with late binding</p> <p>Property of Extensibility</p> <p>Property of Computational completeness</p>

**Figure 1 OODB Features**  
(Source – ecomputernotes.com)

### 3.1 Complex Objects Mechanism

Object oriented databases supports complex objects mechanism that allows an object to contain attributes that can themselves be objects. In other words, the schema of an object is not in first normal-form, every instance in the database has a unique identifier, which is a property of an object that distinguishes it from all other objects and remains for the lifetime of the object. In object-oriented systems, an object has an existence (identity) independent of its value that enforces encapsulation and information hiding. This means, the state of objects can be manipulated and read only by invoking operations that are specified within the type definition and made visible through the public clause. In an object-oriented database system encapsulation is achieved if only the operations are visible to the programmer and both the data and the implementation are hidden in the class. The concept of class is associated with run time execution that refers to a collection of all objects with the same internal structure (attributes) and methods. Most relational database applications involve the use of SQL embedded within a conventional programming language. The problem with this approach is that whilst SQL deals with sets of records, programming languages tend to work on a record at a time basis. This difficulty is known as the impedance mismatch. Object-oriented databases attempt to provide a seamless join between program and database and hence overcome the impedance mismatch. For this reason most data manipulation language of object-oriented database are computationally complete.

### 3.2 Supporting the Object Oriented Features

Basically, OODBMSs are object databases that provides DBMS capabilities to objects that have been created using an object-oriented programming language (OOPL). The basic principle is to add persistence to objects and to make objects persistent. Consequently application programmers who use OODBMSs typically write programs in a native OOPL such as Java, C++ or Smalltalk, and the language has some kind of Persistent class, Database class, Database Interface, or Database API that provides DBMS functionality as, effectively, an extension of the OOPL. Object-oriented DBMSs, therefore, support advanced object-oriented database applications with features like support for persistent objects from more than one programming language, distribution of data, advanced transaction models, versions, schema evolution, and dynamic generation of new types.

## 4. MODELING THE OBJECT DATA WITH UML

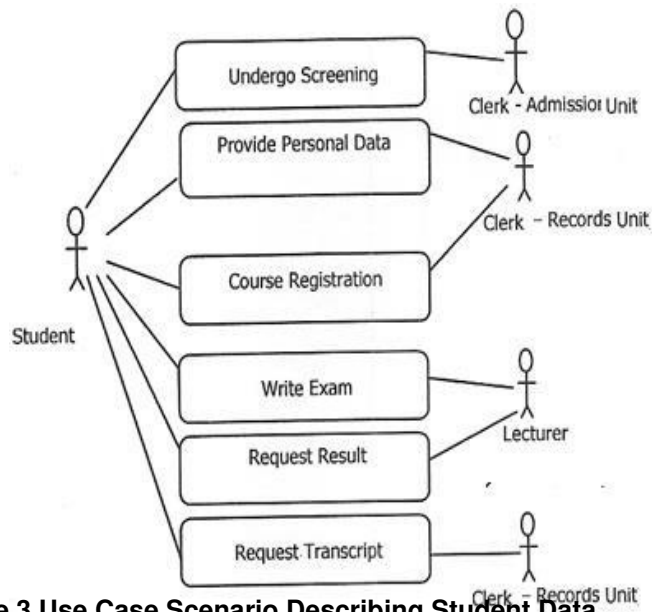
This section describes object data modeling and the persistency concept in OODB using the unified modeling language (UML). Unified modeling language is a methodology for modeling object oriented structures. Figure 2 is the structure of a class represented in UML notation.



Class Name
Attributes
Operations (Method)

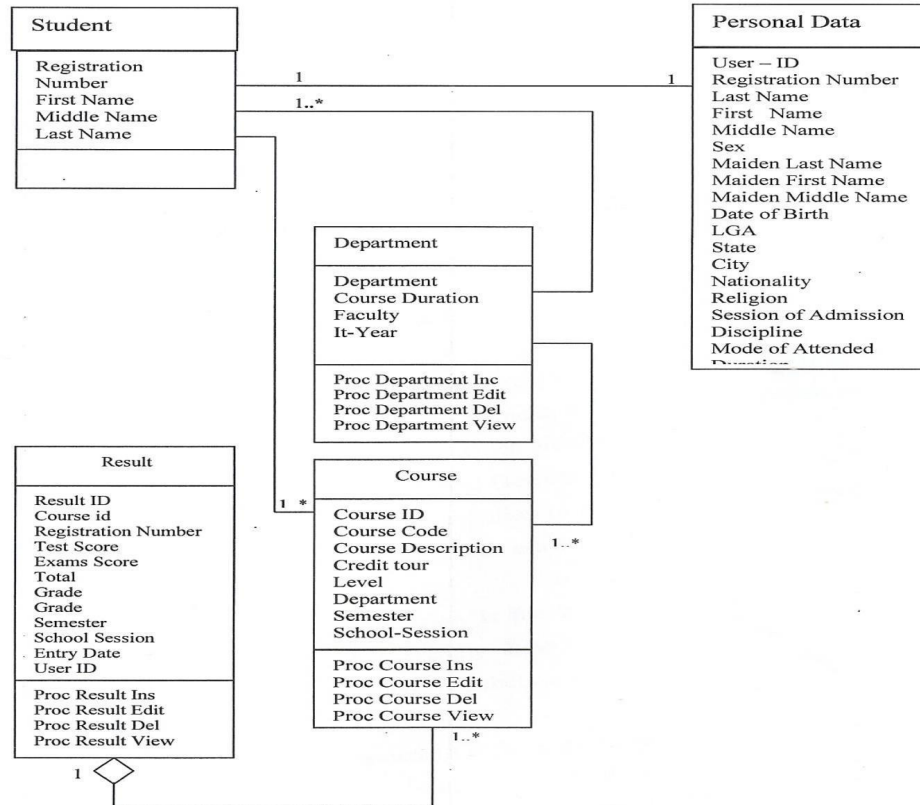
**Figure 2 UML Notation.**

Taking the example of a student relationship with a university, where the student goes through a screening exercise, get admitted, write the necessary examinations and request for results and the transcript. Use Case is used to represent the process. Use Case diagrams are among the various CASE tools employed in UML for modeling systems. A use case represents a process with a system. Use case descriptions describe what a use case does. Scenarios describe different paths through a use case. From the use case diagram in figure 3, the following objects to be modeled are identified: student, personal data, course, results and transcripts.



**Figure 3 Use Case Scenario Describing Student Data**

Classes are identified from Use Case. An object consists of three parts: structure (attribute, and relationship to other objects like aggregation, and association), behaviour (a set of operations) and characteristic of types (generalization/serialization). An object is similar to an entity in E-R model; therefore we begin with an example to demonstrate the structure and relationship. The class diagram in figure 4, models the student information system.




**Figure 4: Class Diagram Modeling Student Information**

The central class is the student, associated with it are personal data of the student, department the student belong, courses offered by the students and the results class respectively. The UML class notation is applied with a rectangular box which is divided into three parts; name, attributes, and operation

The class diagram of the unified modeling language show an object-oriented database schema. It contains five classes; "student", "Personal Data", "Department", "Course", and Result". "Student" is the central class, "Department" is a subclass of "Student" and an association relationship is established from Department to "Student". There may exist a 'bidirectional relationship between student" and "Department". This shows that a "Department" instance has many "Student" instances and the other indicates that a "Student" belongs to one "Department". Attributes are like the fields in a relational model. However in the Student example we have, for attributes Name and Registration number, and the complex types: Personal data and Department, which are also objects. Personal data and Department are associations with 1: 1 and 1: N relationship; Result is an aggregation (a Result for different courses). The 1: N relationship is usually realized as attributes through complex types and at the behavioural level. Table 1 explains these multiplicities of association for the class diagram



**Table 1: Multiplicities of association for the class diagram**

Multiplicities	Meaning
0..1	Zero or one instance.
0..* or *	No limit on the number of instances (including none)
1	Exactly one instance
1..*	At least one instance
	 Aggregation

### 5. CONCLUSION

The OODBMS contains extensive concepts, which makes it a lot more complicated comparing with the RDBMS. This paper gives OODBMS concepts. It also talks about the basics of object modeling. In the future, if the identified limitations of the OODBMS are overcome, then, the anticipation is that the OODBMS will be more widely used.

### REFERENCES

1. Alfons Kemper and Guido Moerkotte, *Object-Oriented Database Management*, 1994
2. Atkinson, Malcolm et al, The Object-Oriented Database Manifesto. In *Proceeding of the First International Conference on Deductive and Object-oriented Databases*, pages 223-240, Kyoto, Japan, December 1989 D. Maier and J. Stein, Development and implementation of an object-oriented DBMS. In *Research Directions in Object-Oriented Programming*, B. D. Shriver, P. Wegner, editors, MIT Press, 1987. Also in *Readings in Object-Oriented Database Systems*, S. Zdonik and D. Maier, editors, Morgan Kaufmann, 1990.
3. D. Maier and J. Stein, Indexing in an object-oriented DBMS. In *Proceeding of the International workshop on Object-Oriented Databases*, pages 171-182, Pacific Grove, CA, September 1986.
4. David Maier's home page: [www.cse.ogi.edu/~maier](http://www.cse.ogi.edu/~maier)
5. Neubauer, Bruce J. (2002), "Data Modeling in the Undergraduate Database Course: Adding UML and XML Modeling to the Traditional Course Content." *Journal of Computing in Small Colleges*, Vol. 17, No. 5, pp. 147-153
6. Orfali, R H. Edwards, J., *The Essential Client/Server Survival Guide*, John Wiley & Sons, 1996. Pressman, R. S., *Software Engineering a Practitioner's Approach*, McGraw Hill, fifth edition, 2001. Ramakrishnan, R, Gehrke, J., *Database Management System*, McGraw Hill, second edition, 2000.
7. Siau, Keng and Qing Cao (2001), "Unified Modeling Language (UML) - a Complexity Analysis." *Journal of Database Management*, Vol. 12, No. 1, pp. 26-34.
8. Urban, Susan D. and Suzanne W. Dietrich (2003), "Using UML Class Diagrams for a comparative Analysis of Relational, Object-Oriented, and Object-Relational Database Mappings." *ACM SIGCSE Bulletin*, Vol. 35, No. 1, pp. 21-25.
9. Wagner, Paul. "Teaching Data Modeling: Process and Patterns." Paper presented at the Proceedings of the 10<sup>th</sup> annual SIGCSE conference on Innovation and technology in computer science education, Capacrica, Portugal, June 2005.