Object Oriented Databases: Concepts and Utility

Omar Lahkim
School of Science and Engineering
Al Akhawayn University in Ifrane
Ifrane, Morocco
o.lahkim@aui.ma

Nasser Assem
School of Science and Engineering
Al Akhawayn University in Ifrane
Ifrane, Morocco
o.lahkim@aui.ma

Abstract— **Object-Oriented Database Systems have experienced rapid expansion in recent years, gaining a larger proportion of the database system market. This is due to the benefits and high performance of Object-Oriented Database Systems over Relational Database Systems. In this paper, we explain the main concepts of the Object-Oriented Database System along with its main components, and its advantages compared to relational Database Systems. Moreover, we mention the most popular Object-Oriented Database Systems along with a case study of the integration of Realm Database in Mobile Applications to illustrate its advantages and its simplicity of use.**

Keywords—Object-Oriented, Databases, OODBMS, concepts, utility, Realm

I. Introduction

File Systems have proven to be inefficient, and limited in terms of performance, usability, security, concurrency, data sharing, and redundancy. This led to researchers to come up with new technology to solve those issues which are Database Management Systems that is a collection of software or programs that maintain the data records. The most popular data model is the relational data model which stores the data as tables since it is based on the relational mathematics. But the Object-Oriented databases solve some limitations of the relational model and Relational Database Management Systems. This model stores data as objects that are uniquely identifiable and model real world entities that have a state and behavior which are stored as attributes and methods in this model as it is easier to implement with Object-Oriented Programming Languages. The most popular Object-Oriented Database Management Systems in 2021 are the GOOGLE Cloud Storage for Firebase, Apache OODT, Actian NoSQL, ObjectBox and Realm by MongoDB which we will be implementing in the case study.

In this paper, we explain the object-oriented databases, how they work, their concepts which are based on the object-oriented model, their components, why those databases are better than the relational databases, and a case study which explains and illustrate the implementation of those object-oriented databases on mobile applications.

II. Background

A. *Relational Databases*

A relational database is a form of database that stores and makes data points connected to one another. Relational databases are based on the relational model, which is itself based on the relational mathematics and is a simple and obvious manner of expressing data in tables. Each row in a relational database is a record with a unique ID called the key. The columns of the table carry data attributes, and each record typically includes a value for each attribute, making it simple to construct links between data points.

Relational Database Management Systems (RDBMS) make it easier to store and manage data. In a relational database management system (RDBMS), tables are linked together using various constraints. Tables are also known as entities. A single entry is represented by a row, whereas an attribute is represented by a column.

ORM which stands Object Relational Mapping is a technology that abstracts the SQL querying by providing methods and functions to manipulate data in the relational database programmatically using programming languages such as Python, Java, PHP using ORM frameworks such as Eloquent for PHP, Hibernate for Java, SQL Alchemy for

Python. This technology is widely used nowadays and is easier than writing plain SQL inside the code.

The four crucial properties of the relational database are referred to as ACID which are:

- Atomicity which
- Consistency
- Isolation
- Durability

The relational database model relies on relationships to link between data which are:

- One To One
- One to Many
- Many to Many

B. Object Oriented Programming Concepts

Object-oriented programming (OOP) is a programming paradigm that organizes software design around data, rather than functions and logic. An object is a data field with its own set of properties and behavior.

Object-oriented programming focuses on the objects that developers desire to handle rather than the logic that is required to manipulate them. This kind of programming is ideally suited to big, complicated, and frequently updated or maintained projects. This encompasses manufacturing and design software, as well as mobile applications; for example, OOP may be used to simulate manufacturing systems.

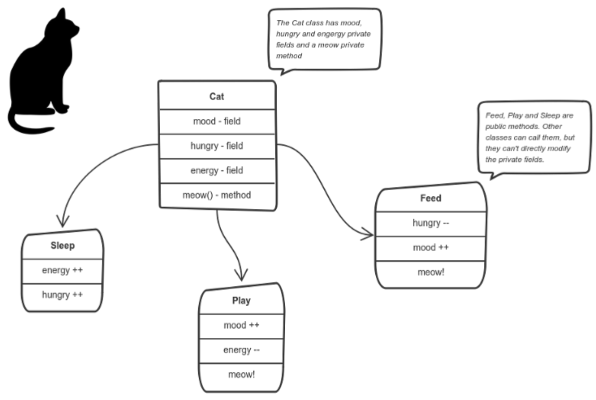
An object-oriented program's structure also makes it useful for collaborative development, where projects are organized into groups. Code reusability, scalability, and efficiency are also advantages of OOP.

The object-oriented programming's building blocks are:

- Classes which are data types that are created by the user and serve as the blueprint for objects, properties, and methods.
- Objects are instances of a class that have been declared and structured.
- State is represented by attributes, which are declared in the class
- Methods are functions specified within a class.

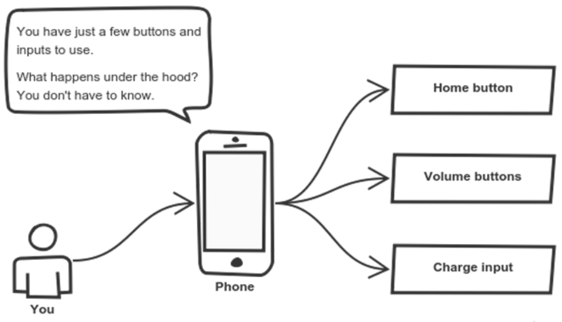
The main concepts of the Object-Oriented Programming are:

- **Encapsulation** which stands for the combination of properties and methods related to the same object.


The Cat class has mood, hungry and engergy private fields and a meow private method
Cat
mood - field
hungry - field
energy - field
meow() - method
Feed, Play and Sleep are public methods. Other classes can call them, but they can't directly modify the private fields.
Sleep
energy ++
hungry ++
Play
mood ++
energy --
meow!
Feed
hungry --
mood ++
meow!

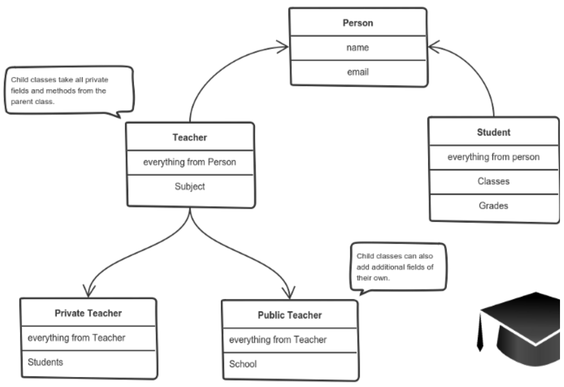
In the figure above, all the attributes and methods are encapsulated in each class.

- **Abstraction** which exposes only the necessary information to the outside world and hiding their background details.


You have just a few buttons and inputs to use.
What happens under the hood? You don't have to know.
Home button
Volume buttons
Charge input
Phone
You

For example, in the figure above, we can see that the person interacts with the phone using buttons, but he doesn't know or see how things are executed and implemented behind the scene.

- **Inheritance** which supports the code reuse between classes and improves the efficiency and development time. As it allows a class to inherit from other classes properties and methods.



In the following figure, a teacher is a person, so it inherits all its properties and methods, and a public teacher is a teacher, so it inherits the properties and methods from the teacher.

- **Polymorphism** which is mainly the ability to process the data in more than one form.

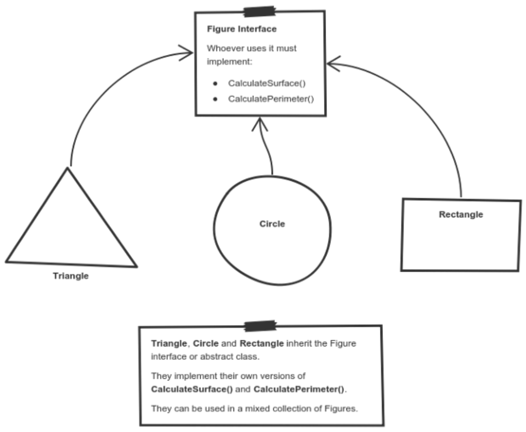


Figure Interface
Whoever uses it must implement:
• CalculateSurface()
• CalculatePerimeter()
Triangle
Circle
Rectangle
Triangle, Circle and Rectangle inherit the Figure interface or abstract class.
They implement their own versions of CalculateSurface() and CalculatePerimeter().
They can be used in a mixed collection of Figures.

For example, in the figure above, the structure of the classes triangle, circle, and rectangle is inherited from the interface Figure, but the implementation of the methods will not be the same and would be overwritten.

This paradigm has many advantages namely:

- Modularity
- Reusability
- Productivity
- Scalability
- Security
- Flexibility

III. Object oriented database

A. *Object-Oriented Database*

The object-oriented programming is a paradigm that allows the developer to model real world entities that have a state and behavior, but the object is only during the lifecycle of the application once it is stopped or killed all the data encapsulated in the objects are destructed. This paradigm is widely used nowadays as it makes the data manipulation easy and human understandable. The object-oriented databases are the solution to store and persist the data and the objects not only during the lifecycle of the application. This feature solves the problem of recovery and concurrency.

Most database systems generally contain textual and numerical data, but the object-oriented databases contain objects, which could hold large documents, pictures, videos, audio files… These database objects are identifiable by metadata.

B. *Components*

The object-oriented database management systems are based on four different components which are the Object Structure, Object Classes, Object Identity, and Object Containment.

- Object Structure means the properties that the object is composed of the properties of the object are known as attributes, Moreover, an object wraps data code into a single unit, providing data abstraction by concealing implementation details from the user. The object structure by itself is composed of three components which are: Messages, Methods, and Variables.
    - Messages act as an intermediary medium between an object and the outside world. We can distinguish two types of messages that are: Read-only messages, and Update Messages
        - Read-only Messages: The invoking message is said to be

read-only if the called method does not modify the value of a variable.

        - Update Messages: The invoking message is said to be an update message if the invoked method modifies the value of a variable.

    - Methods: the body of code that is run when a message is passed is known as a method. When a method is called, it produces an output value. There are two types of methods:

        - Read-only Methods: The read-only method refers to a method that has no effect on the value of a variable.

        - Update Methods: An update method is one that changes the value of a variable using a method.

    - Variables: It keeps track of an object's data. The variables' data allows the objects to be distinguished from one another.

- Object Classes in which a real-world item is referred to as an instance of a class. As a result, we must first declare a class before creating objects that differ in the values, they hold but share the same class definition. Messages and variables are kept in the objects, which relate to them.

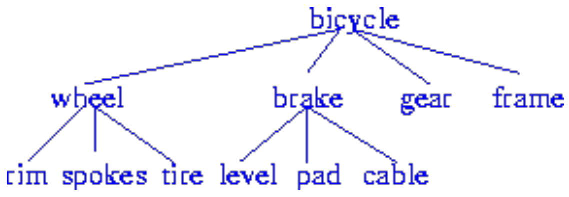
- Object Identity which means even if some or all the values of variables or method definitions change over time, an object keeps its identity. We can distinguish three forms of identity:

    - Value: Identity is determined by a data value (e.g., the primary key of a tuple in a relational database).

    - Name: For identification, a user-supplied name is utilized (e.g., file name in a file system).

    - Built-in: There is no need for a user-supplied identifier since identification is embedded into the data model or programming languages (e.g., in OO systems).

- Object Containment signifies that complex or composite items are objects that contain other objects. Multiple layers of confinement can exist, resulting in a containment hierarchy among items.



The figure above illustrates the containment hierarchy of a bicycle design database.

C. Concepts

- Objects and identity: Every real-world entity is represented as an object in the same way (associated with a unique id: used to identify an object to retrieve).

- Encapsulation: The idea of encapsulation in object-oriented databases is similar to object-oriented programming. The main distinction is that whether the object's data structure is part of the interface is not explicitly stated. The data structure is unquestionably a component of the implementation in computer languages.

  The behavior of an object is specified by the methods defined by the object. The goal of such a method may be to change the values of some attributes or to calculate a value based on the object's current state. Every object can specify an unlimited number of methods. The notion of encapsulation transforms the maintenance of critical procedures into a data-level activity.

- Classes and Instantiation: When looking at the notion of classes in object-oriented databases, it's important to understand the difference between the words class and type. A type is a term that refers to a group of things that have the same behavior. In this view, the type of an item is determined by the operations that may be performed on it. A class is a group of objects with identical internal structures. In this approach, a class specifies an object's implementation, while a type indicates how the object might be utilized.

  The term "instantiation" refers to the ability of a class specification to produce a group of objects with the same structure and behavior. A class defines a structure (a collection of characteristics), a set of actions, and a set of methods to carry out those activities.

  An object's ability to alter its class is a crucial element in the development of things. This means that an object's properties and actions can change while maintaining its identity. Enabling class modifications necessitates the development of a system for dealing with any potential semantic integrity issues. Applications must handle exceptions that may occur when an object is referred to as a different instance of a class than intended.

- Overloading, overriding, and late binding: It's common to use the same name for many, but related, procedures. Assume you want to show anything on your screen. Different viewers may be required for different products. You might want to use the method "view" to see all things. A photo viewer is initiated when you call "view" and give a reference to a picture. A media player is started when you call "view" and supply a reference to a video. You must first define the operation "view" in a common superclass "media" of the classes "image" and "video" to provide this capability. Each subclass redefines the "view" action to meet its own requirements. As a result, several methods with the same operation name are created. Using this feature has a significant benefit. The code is easier to maintain, and the addition of a new type does not necessitate any changes to current program components. By enabling this option, the system will no longer link names of operations to their respective methods at build time. "Late binding" refers to the run-time binding of operation names to their associated methods.

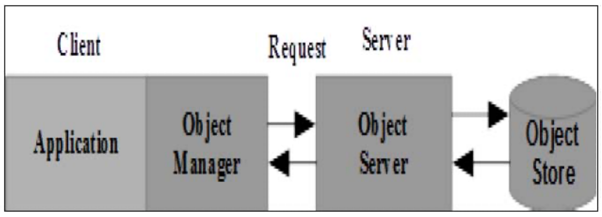
- Class Hierarchy and Inheritance: from an existing class, create a new class (subclass) (superclass). The subclass inherits all the existing class's properties and methods, as well as the ability to create new ones. Multiple inheritance (class lattice) vs. single inheritance (class hierarchy).

D. OODBMS architecture approaches

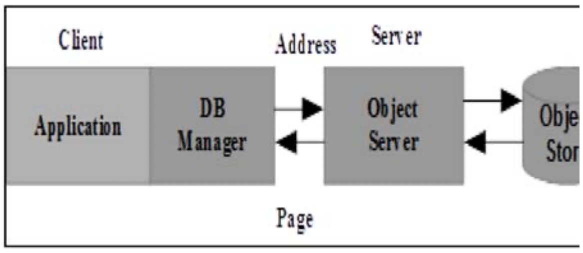
The core concept of an object-oriented database management system (OODBMS) is to give persistence to an object-oriented programming language (OOPL). The main distinction is that in this case, the database must hold both data and methods.

Architecture (Client/Server): There are three basic approaches to client/server architecture: Object Server, Page Server, and Database Server.

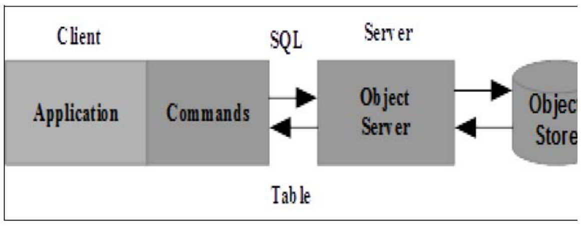
- Object Server: Between the client and server, there is a distributed processing environment. Other OODBMs functions are usually handled by the server. Client is in charge of transaction control and programming language interface.



- Page Server: In this client-server approach, the client is usually in charge of database processing. The server is in charge of secondary storage management and request response. A page can include many complicated or ordinary things.



- Database Server: In this technique, the client simply sends the request to the server, which processes it and returns the results to the application. Most of the database processing takes place on the server. RDBMSs are the most common users of this method.



E. *Why Object Oriented Databases*

- Support for User Defined Data Types: OODBs gives the ability to create and manage new user defined data types.
- OODBs allow for the creation of new types of relationships: inverse relationships are a new sort of relationship between objects that may be created using OODBs (a binary relationship)
- Identification does not necessitate the use of keys: Object data models, unlike relational models, employ object identity (OID) to identify objects in the system.
- Gains in performance over RDBMS: Gains in performance vary per application. Applications that leverage the object identity notion outperform RDBMSs in terms of performance.
- Object algebra development: While relational algebra is based on relational mathematics and is completely implemented, object algebra has not been properly described. Union, difference, select, create, and map are the five basic object-preserving operators.
- OODBMSs do not require joins: OODBs have the potential to decrease the number of joins required.
- Equality Predicates in OODBs: There are four types of equality predicates in OODBs:
  - Identity equality
  - Object value equality
  - Property value equality
  - Property identity equality

F. *Weaknesses*

- No Universal Standards
- Relational and Object Model Coherence: Relational databases are the foundation of any corporation. To beat relational databases, object databases must provide customers with consistent services that allow them to transition from relational to object databases. The architecture of the Relational model and the Object model must be consistent.
- Object database security issues include authentication, authorization, and accounting. Security rules such as Discretionary Access Control (DAC) and Mandatory Access Control (MAC) are used to safeguard data in object databases. Some systems use object-oriented ideas such as encapsulation to ensure security. Authorization facilities are available in an object database.
- For OODBMSs, there is no fixed query algebra: There is no standard query algebra for OODB due to the lack of standards in OODBMS. One of the causes for the challenge of query optimization is the lack of standard query algebra. For different object database, there exist multiple query languages.
- No Views: Views are transient tables in relational databases. Views are not supported in object databases. Due to the properties of the Object Model, such as object identity, an object-oriented

view capability is difficult to create. Views in object databases are challenging to build due to object-oriented programming concepts like inheritance and encapsulation.

- Query Expression Optimization: Query expression optimization is done to improve the system's performance. Queries must be optimized if performance advantages are to be realized. However, query optimization in object databases is problematic for the following reasons:
    - Data kinds that are defined by the user
    - Varieties of kinds that change
    - Encapsulation, complex objects, and procedures
    - The Object Query language has a nested structure
    - The Object Identity language has a layered structure
- Performance advantages over RDBs are limited. Performance benefits vary from application to application, resulting in a decrease in performance. Applications that leverage the object identity notion outperform RDBMSs in terms of performance. However, OODBMS performance is poor for applications that demand bulk database loading and do not employ OID.
- There are a few basic things that are missing: Triggers, meta data management, and restrictions such as UNIQUE and NULL are all missing from object databases.

G. Difference between OODBMS & RDBMS

Relational databases have been the industry standard for online and software development for a long time. Information is stored in this paradigm in linked tables. Links between complicated pieces of information with various components can also be saved and accessed here. However, with an object database, all the unit's components are immediately available. As a result, the data sets might be significantly more complicated. When using a relational database, we usually strive to handle basic data. The more complicated the data collection, the more links there are, crowding up the database.

Conclusion

Relational databases are undeniably popular, and they can be found almost anywhere. In the mid-1985s, the object-oriented database was introduced to overcome these restrictions and to enable sophisticated database applications such as CAD, CASE, and others. The popularity of object-based programming is another factor that encourages the development of object-based databases. As a result, database experts believe that combining object-oriented programming principles with database management systems will result in more powerful database management systems. Industry uses a variety of ways to create databases with object-oriented capabilities. Relational extensions and 18 pure object-oriented techniques are the most prevalent ways for developing object-oriented database systems. Many alternative methodologies are used to create OODBMSs by various manufacturers. OODBMSs eliminate the constraints of RDBMSs while also supporting sophisticated database applications with extra functionalities. However, they are not well-known in the sector due to a lack of standards. After some time, various limits in object-oriented database management systems are discovered.

References

[1] Damesha, H. S. (2015). Object Oriented Database Management Systems-Concepts, Advantages, Limitations and Comparative Study with Relational Database Management Systems. *Global Journal of Computer Science and Technology.*

[2] What is object-oriented programming? the four basic concepts of OOP. Indeed Career Guide. (n.d.). Retrieved November 15, 2021, from https://www.indeed.com/career-advice/career-development/what-is-object-oriented-programming

[3] What is an object-oriented database? MongoDB. (n.d.). Retrieved November 15, 2021, from https://www.mongodb.com/databases/what-is-an-object-oriented-database.

[4] Barry, D. K. (1996). The object database handbook: how to select, implement, and use object-oriented databases. John Wiley & Sons, Inc..

[5] Bertino, E., & Martino, L. (1993). *Object-oriented database systems: concepts and architectures*. Addison-Wesley Longman Publishing Co., Inc..

[6] Attoui, A., & Gourgand, M. Performance Evaluation for Clustering Algorithms in Object-Oriented Database Systems.

[7] Feuerlicht, G., Beranek, M., & Kovar, V. (2021). Design of Document Databases: What can we Learn from Object-Relational Databases?.

[8] MongoDB. (n.d.). MongoDB Realm. Retrieved November 17, 2021, from https://www.mongodb.com/realm

[9] Actian. (2020, December 28). Actian NoSQL Object Database. Retrieved November 17, 2021, from https://www.actian.com/data-management/nosql-object-database/

[10] Cloud Storage for Firebase | Firebase Documentation. (n.d.). Firebase. Retrieved November 17, 2021, from https://firebase.google.com/docs/storage

[11] Oodt, A. (n.d.). Apache OODT - Distributed Data Management. Apache OODT. Retrieved November 17, 2021, from https://oodt.apache.org/

[12] ObjectBox - Edge Database for Mobile, IoT, and Embedded Devices. (2021, November 5). ObjectBox. Retrieved November 17, 2021, from https://objectbox.io/

[13] Coronel, C., & Morris, S. (2018). Database Systems: Design, Implementation, & Management (13th ed.). Cengage Learning.

[14] Ishikawa, H. (2012). Object-Oriented Database System: Design and Implementation for Advanced Applications (Computer Science Workbench) (Softcover reprint of the original 1st ed. 1993 ed.). Springer.

[15] Freeman, E., & Robson, E. (2020). Head First Design Patterns: Building Extensible and Maintainable Object-Oriented Software 2nd Edition (2nd ed.). O'Reilly Media.

[16] Mach, E. (2019). Object Oriented Analysis & Design Cookbook: Introduction to Practical System Modeling. Independently published.

[17] Relational database. (n.d.). IBM. Retrieved November 17, 2021, from https://www.ibm.com/analytics/relational-database

[18] Education, I. C. (2021, November 3). Relational Databases. IBM Relational Databases. Retrieved November 17, 2021, from https://www.ibm.com/cloud/learn/relational-databases

[19] Curator, C. (n.d.). What Are Object-Oriented Databases And Their Advantages. Object Oriented Database Management Systems. Retrieved November 17, 2021, from https://www.c-sharpcorner.com/article/what-are-object-oriented-databases-and-their-advantages2/

[20] Wikipedia contributors. (2021, November 15). Object database. Wikipedia. Retrieved November 17, 2021, from https://en.wikipedia.org/wiki/Object_database

[21] freeCodeCamp.org. (2021, February 20). *How to explain object-oriented programming concepts to a 6-year-old*. Retrieved November 27, 2021, from https://www.freecodecamp.org/news/object-oriented-programming-concepts-21bb035f7260/

[22] Zaiane, O. (1998). *The Object-Oriented Data Model*. SIMON FRASER UNIVERSITY. Retrieved November 28, 2021, from https://www2.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter8/node3.html

[23] SchÃ¶nhart, S. A. M. (n.d.). *Object-Oriented Databases - Basic Concepts*. Universität Klagenfurt. Retrieved November 28, 2021, from http://cs-exhibitions.uni-klu.ac.at/index.php?id=391

[24] *Object-oriented databases: the insider tip in database models*. (2021, November 26). IONOS Digitalguide. Retrieved November 28, 2021, from https://www.ionos.com/digitalguide/hosting/technical-matters/object-oriented-databases/

[25] GeeksforGeeks. (2021, June 10). *Definition and Overview of ODBMS*. Retrieved November 28, 2021, from https://www.geeksforgeeks.org/definition-and-overview-of-odbms/