# Discussion-7 Object-Oriented Databases

**Discussion Topic:**

What are the advantages and limitations of different storage mechanisms for objects?
How can an object be mapped to a relational storage?
Provide an example in your post.

**My Post:**

Hello Class,

**Overview of the Object-Oriented Concept**

In Software Engineering (SE), the Object-Oriented concept (OO) is a paradigm where systems and real-world entities (e.g. users) are defined as "objects." These objects are abstractions that, in the context of a software application, encapsulate both data (attributes) and operations that can be performed on that data (methods). The data are the attributes that define the characteristics of the objects and the operation on the data are the methods (procedures or functions) that define the behavior of an object (Colorado State University Global, n.d.). In other words, objects contain both executable code (methods) and data (attributes). OO is composed of six main concepts:

- Classification ─ type of object ─
- Abstraction ─ Hiding the object's internal implementation details, including some of its attributes and the functionalities of its methods, while representing those attributes and methods internally ─
- Encapsulation ─ Defining the boundary of an object (modularization), defining an object's attributes and methods ─
- Association ─ Relationships defining objects' interactions with each other ─
- Inheritance ─ Relationships defining objects as a generalization (parents) of other objects (children) ─
- Polymorphism ─ How objects respond differently to the same message (same method call) ─

**The Differences Between OO, OOP, and OODB (ODBMS)**

Object-Oriented Programming (OOP) embodies the principles of the OO concept. OOP is a style of programming that encapsulates data (attributes) and code (methods) within classes which are blueprints for creating (instantiate) objects. In other words, the OO approach is a general concept, OOP is a specific way of implementing it. On the other hand, Object-Oriented Databases (OODBs) also called Object Database Management Systems (ODBMS) "store data as objects, similar to how certain programming languages manage data. Instead of tables with rows and columns like traditional databases, object databases use complex data structures to represent data" (MongoDB, n.d.).

**Overview of Object-Oriented Database**

OODBs are databases based on the OO paradigm and store data as objects. They are "able to store objects together with their attribute values, operations, and relationships" (Unhelkar, 2018, p.219). The databases store objects “as they are” preserving the object's structure. In SE, this facilitates the design of OO-based software systems, as objects can be loaded directly from the database into memory and executed "as they are." This is useful in software design because, in addition to storing object-based data, OODBs can directly store (“as they are”) other complex data types like Binary Large Objects (BLOBs) and unstructured data (e.g., video and audio). Unlike relational databases that use a tabular (rows and columns) format, which is not suited for storing these two data types. OODBs have several other advantages besides storing complex data, as well as disadvantages, the table below lists these advantages and disadvantages.

**Table 1**
*ODBMS Advantages and Disadvantages*

| Advantages | Disadvantages |
|---|---|
| Can directly store object storage:<br>- Integration with OOP<br>- Objects (data, methods, relationships) are stored "as is," eliminating assembly/disassembly.<br>- Supports BLOBs and complex unstructured data without format conversion. | Has performance limitations:<br>- Lower efficiency for simple data/relationships.<br>- Possible slower access due to late binding.<br>- Poor performance for simple queries compared to Relational Databases Management Systems (RDBMS).<br>- Not well-suited for dynamic data that is constantly changing, as changes to an object will require to rewrite the entire object. |
| Can handle complex data:<br>- Supports inheritance, polymorphism, encapsulation, and complex data types (e.g., spatial, multimedia).<br>- Easier navigation (search) through object hierarchies. | It is difficult to adopt and support:<br>- Limited adoption and developer expertise.<br>- Fewer user tools, libraries, and community support compared to RDBMS. |
| Fast development and implementation**:**<br>- No need to map objects to tables (simpler mapping).<br>- Less code is required for object-oriented applications. | No standardization:<br>- Lack of standardization across vendors.<br>- Less stable standards than RDBMS (risk of changes). |
| Can model the real world:<br>- Data model aligns with real-world entities and OOP principles.<br>- Relationships (inheritance, association) are stored directly. | Scalability challenges:<br>- Complex data models may hinder partitioning data across nodes.<br>- Vertical/horizontal scaling requires specialized infrastructure. |

| | |
|---|---|
| Well-suited for concurrency control encapsulation:<br><br>- Better concurrency control (hierarchy locking).<br>- Encapsulation improves data security and integrity. | High cost and difficulty to integrate with other systems:<br><br>- Higher costs (specialized software/hardware).<br>- Difficult integration with BI/reporting tools. |
| Suited for distributed architecture:<br><br>- well-suited for applications that run across multiple computer systems and systems connected over a network (i.e., a distributed system) | Can be complex:<br><br>- Object-oriented paradigms can be more complex than relational models.<br>- Steeper learning curve for developers accustomed to relational models. |

*Note:* The table lists the advantages and disadvantages of Object Database Management Systems (ODBMS). From several sources (Colorado State University Global, n.d.; Unhelkar, 2018; Worldlovely, 2024; Akshitakumawat, 2023)

As shown by Table 1, OODBs are ideal for storing complex data such as complex unstructured data (AD/CAM, multimedia, scientific databases). However, they lag the standardization, the simplicity, and the relatively low cost found in Relational Databases Management Systems.

**Overview of Relational Database**

Relational Databases (RBs) also called Relational Databases Management Systems (RDBMS) are databases that store and organize predefined relationships in the form of tables that store data points in pre-defined categories through rows and columns (Microsoft, n.d.; Google n.d.). Relationships are established through logical connections between tables, typically using primary keys and foreign keys. RDBMS structure data based on these relationships, making it easier to query and manage data using Structured Query Language (SQL). RBMS uses the concept of primary key and foreign key. A primary key is a unique identifier for each record in a table (each row) (InterSystems, n.d.). A foreign key establishes a link between two tables, allowing related data to be accessed using the keys. RBMS have several advantages and disadvantages, the table below lists some of them.

**Table 2**
*RDBMS Advantages and Disadvantages*

| Advantages | Disadvantages |
|---|---|
| Simplicity:<br><br>- Simplicity of the relational model.<br>- Easier to understand, implement, and manage. | Scalability:<br><br>- Performance can degrade with extremely large datasets and high transaction volumes.<br>- Features like transactions and joins become less efficient across multiple servers. |
| Accuracy and data integrity:<br><br>- Primary and foreign keys prevent duplicate data. | Schema: |

| | |
|---|---|
| - Enforces data accuracy<br>- Data typing and validity checks ensure correct data input.<br>- Warns when data is missing. | - Structure is defined beforehand (fixed schema), making changes difficult and potentially leading to downtime.<br>- Adding new columns or altering the schema often requires modifying existing applications.<br>- Inflexible for frequently changing data requirements. |
| Security:<br><br>- Role-based security limits data access. | Cost:<br><br>- Software for creating and configuring a relational database can be expensive.<br>- Managing the database often requires specialized expertise. |
| Accessibility and flexibility:<br><br>- Multiple users can access data simultaneously; Easy to navigate and query tables using SQL.<br>- Easy to add, update, or delete tables, relationships, and data without impacting the overall database or applications. | Performance:<br><br>- Can be slower than other database types, especially with large datasets or complex queries. |
| ACID compliance:<br><br>- Supports Atomicity, Consistency, Isolation, and Durability for reliable transactions.<br>- Atomicity: Defines all the elements that make up a complete database transaction.<br>- Consistency: Defines the rules for maintaining data points in a correct state after a transaction.<br>- Isolation: Keeps the effect of a transaction invisible to others until it is committed.<br>- Durability: Ensures that data changes become permanent once the transaction is committed. | Memory:<br><br>- Can occupy a significant amount of physical memory due to its tabular structure. |
| Ease of use and collaboration:<br><br>- Complex queries can be easily run using SQL.<br><br>- Even non-technical users can learn to interact with the database<br><br>- Multiple users can operate and access data concurrently.<br><br>- Built-in locking prevents simultaneous access during updates. | Object-Orientation:<br><br>- Does not inherently support object-oriented concepts like classes and inheritance<br><br>- Challenging to represent certain types of data or relationships that are better suited for object-oriented models. |
| Database design: | Database design: |

| | |
|---|---|
| - Reduces data redundancy and improves data integrity (Normalization).<br>- Supports one-to-one, one-to-many, and many-to-many relationships. | - Designing a schema can be complex and time-consuming, requiring significant expertise, especially for large applications<br>- Proper normalization and establishing relationships can be time-consuming and require expertise. |

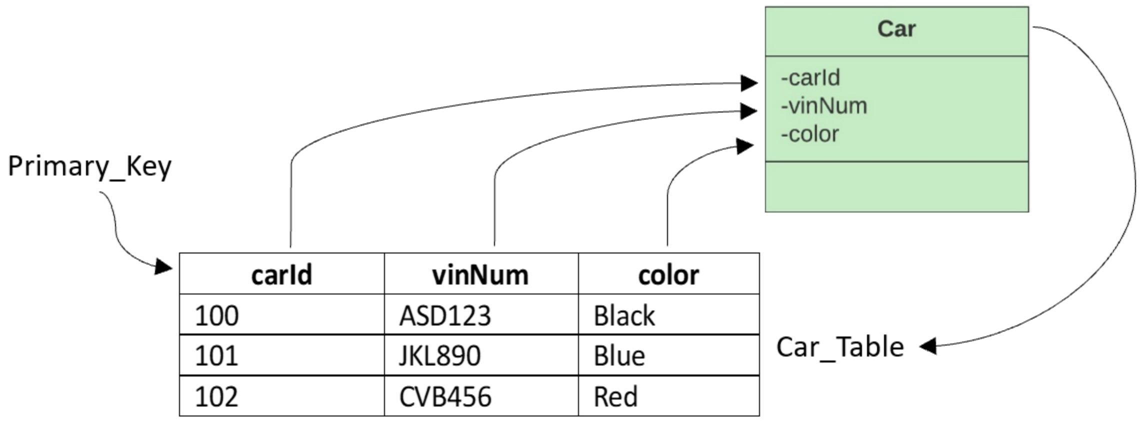
*Note*: The table lists the advantages and disadvantages of Relational Databases Management Systems (InterSystems, n.d.; Pedamkar, 2023;  EASA, 2022; Google, n.d.; OCI, 2021; Singh; 2024, Pedamkar, 2023; Goray, 2012)

RDBMS can be used to store object-based data; however, they store data in tables that are structurally different from objects, requiring objects to be translated from classes to tables (Unhelkar, 2018). Additionally, the data and behavior represented by a class cannot be directly transferred to a table because DBs can only store data. However, it is possible to map object-oriented designs into RDBMS, but it is a difficult task.

**Mapping Objects to RDBMS**

Object-Relational Mapping (ORM) is a technique that bridges the gap between OOP and RDBMS (Rajputtzdb, 2024). One of the simplest ORM forms of mapping is one-to-one mapping, it where classes to tables, class attributes to the table columns, and the object to rows See Figure 1

**Figure 1**
RDBMS Object Mapping



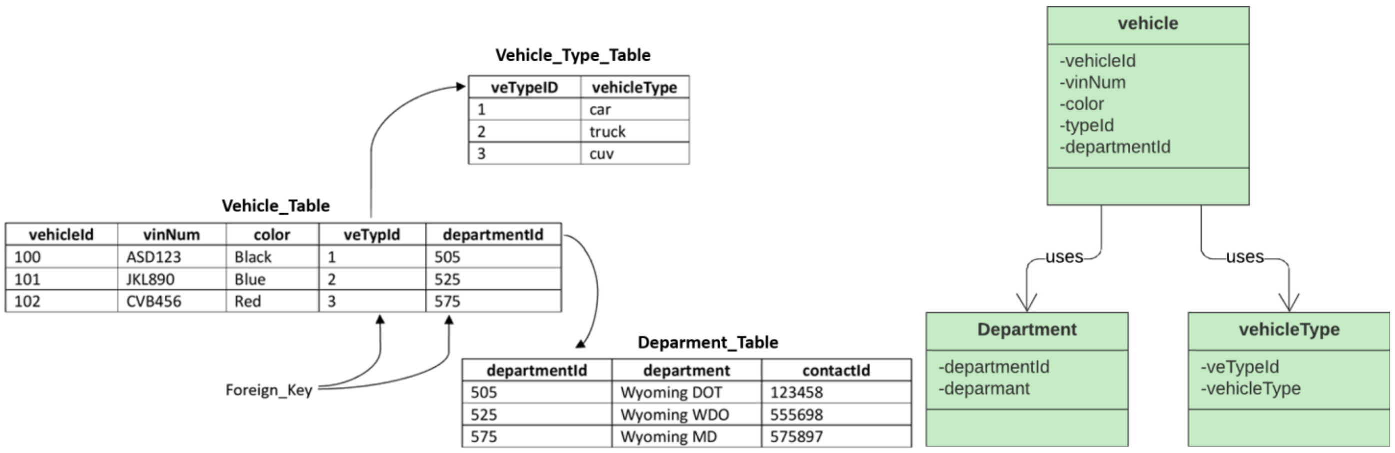
| carId | vinNum | color |
|---|---|---|
| 100 | ASD123 | Black |
| 101 | JKL890 | Blue |
| 102 | CVB456 | Red |

*Note:* The figure depicts how objects are mapped to RDBMS table using the ORM One-To-One mapping form.

To map and simulate object behavior with RDBMS ORM combined with the CRUD functions can be implemented. ORM the association relationships between classes using the foreign key in tables. The figure below depicts how foreign keys emulate the association relationship between classes

**Figure 2**

*Foreign Keys and OO Association Relationship*



Vehicle_Type_Table

| veTypeID | vehicleType |
|---|---|
| 1 | car |
| 2 | truck |
| 3 | cuv |

Vehicle_Table

| vehicleId | vinNum | color | veTypId | departmentId |
|---|---|---|---|---|
| 100 | ASD123 | Black | 1 | 505 |
| 101 | JKL890 | Blue | 2 | 525 |
| 102 | CVB456 | Red | 3 | 575 |

Deparment_Table

| departmentId | department | contactId |
|---|---|---|
| 505 | Wyoming DOT | 123458 |
| 525 | Wyoming WDO | 555698 |
| 575 | Wyoming MD | 575897 |

*Note:* the figure depicts how foreign keys emulate OO association relationships.

To emulate the aggregation and other association relationships within RDBM, CRUD functions which stand for

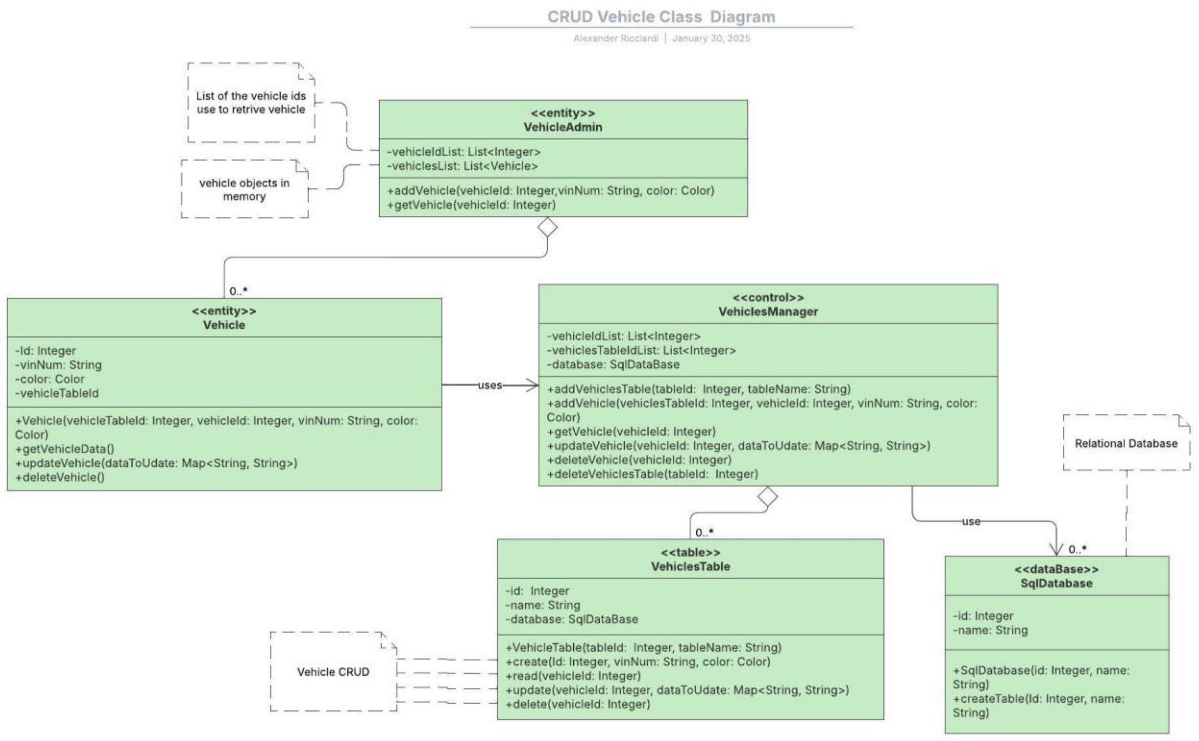
- Create—creates an object.
- Read—search for an object (record) from storage based on a criterion (key).
- Update—search and update objects (records).
- Delete—locates and remove a persistent object

(Unhelkar, 2018)

These operations are executed using languages such as SOL. The figure below depicts a class diagram illustrating how CRUD operations and association relationships are linked to RDBMS.

**Figure 3**

*CRUD and RDBMS Class Diagram*


CRUD Vehicle Class Diagram
Alexander Ricciardi | January 30, 2025
List of the vehicle ids use to retrive vehicle
vehicle objects in memory
<<entity>> VehicleAdmin
-vehicleIdList: List<Integer>
-vehiclesList: List<Vehicle>
+addVehicle(vehicleId: Integer,vinNum: String, color: Color)
+getVehicle(vehicleId: Integer)
0..*
<<entity>> Vehicle
-Id: Integer
-vinNum: String
-color: Color
-vehicleTableId
+Vehicle(vehicleTableId: Integer, vehicleId: Integer, vinNum: String, color: Color)
+getVehicleData()
+updateVehicle(dataToUdate: Map<String, String>)
+deleteVehicle()
uses
<<control>> VehiclesManager
-vehicleIdList: List<Integer>
-vehiclesTableIdList: List<Integer>
-database: SqlDataBase
+addVehiclesTable(tableId: Integer, tableName: String)
+addVehicle(vehiclesTableId: Integer, vehicleId: Integer, vinNum: String, color: Color)
+getVehicle(vehicleId: Integer)
+updateVehicle(vehicleId: Integer, dataToUdate: Map<String, String>)
+deleteVehicle(vehicleId: Integer)
+deleteVehiclesTable(tableId: Integer)
Relational Database
use
0..*
0..*
<<table>> VehiclesTable
-id: Integer
-name: String
-database: SqlDataBase
+VehicleTable(tableId: Integer, tableName: String)
+create(Id: Integer, vinNum: String, color: Color)
+read(vehicleId: Integer)
+update(vehicleId: Integer, dataToUdate: Map<String, String>)
+delete(vehicleId: Integer)
Vehicle CRUD
<<dataBase>> SqlDatabase
-id: Integer
-name: String
+SqlDatabase(id: Integer, name: String)
+createTable(Id: Integer, name: String)

*Note:* The figure depicts a class diagram illustrating how CRUD operations are emulated within objects and how association relationships are linked to RDBMS using CRUD and key attributes.

To summarize, Object-Oriented Databases (ODBMS) directly support object data types; on the other hand, Relational Databases (RDBMS) are prevalent due to their widely adopted standards and robustness; however, they store data in tables that are structurally different from objects, requiring objects to be translated from classes to tables. Nonetheless, Object-relational mapping (ORM) allows RDBMS to effectively manage object-oriented data by bridging the gap between objects and tables.

-Alex

**References:**

Akshitakumawat (2023, May 1). Definition and Overview of ODBMS. GeeksForGeeks. https://www.geeksforgeeks.org/definition-and-overview-of-odbms/

Colorado State University Global (n.d.). Module 7.1 *Data Storage Mechanisms.* [Interactive lecture]. CSC470 Software Engineering, CSU Global, Departement of Computer Science. Canvas. Retrieved January 29, 2025, from: https://csuglobal.instructure.com/courses/104036/pages/7-dot-1-data-storage-mechanisms?module_item_id=5372300

EASA (2022, January 25). *8 advantages of a Relational Database*. EASA Blog. https://www.easasoftware.com/insights/8-advantages-of-a-relational-database/

Google (n.d.). *What is a relational database?* Google Cloud. https://cloud.google.com/learn/what-is-a-relational-database

InterSystems (n.d.) *What is a relational database and why do you need one?* InterSystems. https://www.intersystems.com/resources/what-is-a-relational-database/

Microsoft (n.d.). *What is a relational database?* Microsoft Azure. https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-a-relational-database

MongoDB (n.d.). *The basics of object-oriented databases*. MongoDB. https://www.mongodb.com/resources/basics/databases/what-is-an-object-oriented-database#:~:text=An%20object%2Doriented%20database%20stores,data%20structures%20to%20represent%20data.

OCI (2021, June 18). *What is a relational database? (RDBMS)?* Oracle. https://www.oracle.com/database/what-is-a-relational-database/

Pedamkar, P. (2023, July 6). *Relational database advantages*. EDUCBA. https://www.educba.com/relational-database-advantages/

Rajputtzdb (2024, February 28) *What is Object-Relational Mapping (ORM) in DBMS?* GeeksForGeeks. https://www.geeksforgeeks.org/what-is-object-relational-mapping-orm-in-dbms/

Singh, P. (2024, April 24). *15 Advantages and Disadvantages of RDBMS*. Internshala Trainings Blog. https://trainings.internshala.com/blog/advantages-and-disadvantages-of-rdbms/

Goray, S. (2021, December 10). *Top 10+ Advantages and Disadvantages of Using RDBMS*. WAC. https://webandcrafts.com/blog/advantages-disadvantages-rdbms

Unhelkar, B. (2018). Chapter 13 — Database modeling with class and sequence diagrams. *Software engineering with UML.* CRC Press. ISBN 9781138297432

Worldlovely (2024, December). *Database management system* [Discussion Post]. Hack The Forum. https://www.hacktheforum.com/database-management-system/object-oriented-dbms/#google_vignette