

1. Overview of Object-Oriented Concepts

Object-Oriented Databases (OODB) integrate **object-oriented programming concepts** with **database capabilities**.

They store data as **objects** rather than tables (as in RDBMS).

Key Concepts

Concept	Description
Object	Real-world entity that combines data and behavior.
Class	Blueprint or template for creating objects.
Inheritance	Enables a subclass to acquire attributes and behaviors of a superclass.
Encapsulation	Bundling of data and methods together.
Polymorphism	Same operation behaves differently on different classes.
Persistence	Objects outlive the process that created them (stored permanently).
Identity (OID)	Unique identifier assigned to each object, independent of data values.

Example (in pseudo-code):

```
class Student {  
    String name;  
    int rollNo;  
    void display() { ... }  
}
```

2. Object Model of ODMG (Object Data Management Group)

ODMG (now part of the OMG) defined **standards for object databases** to ensure portability and interoperability.

Main Components of ODMG Model

1. **Object Model** – Defines basic building blocks (objects, literals, collections, relationships).
2. **Object Definition Language (ODL)** – Defines structure of object schemas.
3. **Object Query Language (OQL)** – Querying and manipulating object data.
4. **Bindings** – Language bindings for Java, C++, Smalltalk.

Basic Constructs in ODMG Object Model

- **Object Identifier (OID)**
- **Attributes**
- **Methods**
- **Relationships**
- **Collections** (Set, Bag, List, Array, Dictionary)

3. Object Definition Language (ODL)

- ODL is similar to an object-oriented data definition language.
- Used to define classes, attributes, and relationships in object databases.

Example:

```
interface Student {  
    attribute string name;  
    attribute int rollNo;  
    relationship Course enrolls inverse Course::students;  
};  
interface Course {  
    attribute string title;  
    relationship Set<Student> students inverse Student::enrolls;  
};
```

4. Object Query Language (OQL)

- OQL is similar to SQL but operates on objects instead of tables.
- It supports **navigation through object relationships** and **method invocations**.

Example:

```
SELECT s.name  
FROM Student s  
WHERE s.rollNo > 100;
```

With relationships:

```
SELECT c.title  
FROM s IN students, c IN s.enrolls  
WHERE s.name = "Rahul";
```

5. Object Database Conceptual Design

Steps:

1. **Requirement Analysis:** Identify objects and relationships.
2. **Class Definition:** Define classes, attributes, and operations.
3. **Inheritance Hierarchy:** Define superclass and subclass relationships.
4. **Association and Aggregation:** Represent real-world relationships.
5. **Mapping:** Map objects to the OODBMS schema.

Goal: To ensure object structure reflects real-world entities and behavior.

Distributed Database Concepts

6. Distributed Database Concepts

A **Distributed Database System (DDBS)** consists of a collection of **logically interrelated databases** distributed over a computer network.

Key Features

- Data stored across multiple sites.
- Sites communicate via a network.
- Users perceive data as a single integrated database.

Advantages

- Improved reliability and availability
- Faster local processing
- Scalability
- Modular growth

Disadvantages

- Complex design and management
- Costly communication
- Distributed transaction management difficulty

7. Data Fragmentation, Replication, and Allocation Techniques

(a) Data Fragmentation

Dividing a database into smaller pieces (fragments) that can be stored across multiple sites.

Type	Description
Horizontal Fragmentation	Splitting a relation by rows (tuples).
Vertical Fragmentation	Splitting a relation by columns (attributes).
Hybrid Fragmentation	Combination of both horizontal and vertical.

Example:

Example:

Employee table:

Site 1 → Employees from Assam
 Site 2 → Employees from Delhi

(b) Replication

- Keeping **copies of data** at multiple sites.
- Improves availability and fault tolerance.
- Types:
 - **Full Replication** – entire database replicated at all sites.
 - **Partial Replication** – selected fragments replicated.
 - **No Replication** – each fragment stored once.

(c) Data Allocation

- Deciding **where to store each fragment or replica**.
- **Objectives:** Minimize communication cost, maximize performance, ensure reliability.

8. Types of Distributed Database Systems

Type	Description
Homogeneous DDBS	All sites use same DBMS software and schema.

Heterogeneous DDBS	Sites may use different DBMS products and data models.
Federated DDBS	Each site maintains its autonomy; integrated logically.
Client-Server DDBS	Clients send queries; servers manage data.

9. Query Processing in Distributed Databases

Phases:

1. **Query Decomposition** – Parsing and translating the query.
2. **Data Localization** – Determine fragments needed.
3. **Optimization** – Choose best execution plan (based on cost).
4. **Execution** – Retrieve and combine data from multiple sites.

Challenges:

- Data location transparency
- Communication cost minimization
- Join operations across sites

10. Overview of Concurrency Control in Distributed Databases

Goal: Ensure correct execution of concurrent transactions across multiple sites.

Techniques:

1. **Two-Phase Locking (2PL):**

- Ensures serializability.

- Two phases: **Growing** (acquire locks), **Shrinking** (release locks).

2. **Distributed Timestamp Ordering:**

- Assigns timestamps to transactions.
- Ensures global order of execution.

3. **Optimistic Concurrency Control:**

- Transactions execute without locking.
- Validation performed before commit.

Problems:

- Deadlocks
- Communication delays
- Replication conflicts

11. Recovery Techniques in Distributed Databases

Objectives:

- Ensure atomicity and durability in distributed transactions.

Techniques:

1. **Two-Phase Commit Protocol (2PC):**

- Ensures all sites either commit or rollback.
- Steps:
 - **Phase 1:** Coordinator asks all sites to prepare.
 - **Phase 2:** If all agree → commit; else → abort.

2. Three-Phase Commit Protocol (3PC):

- Reduces the risk of blocking in case of site failure.

3. Checkpointing:

- Saves database state periodically to speed up recovery.