Advanced Database Concepts

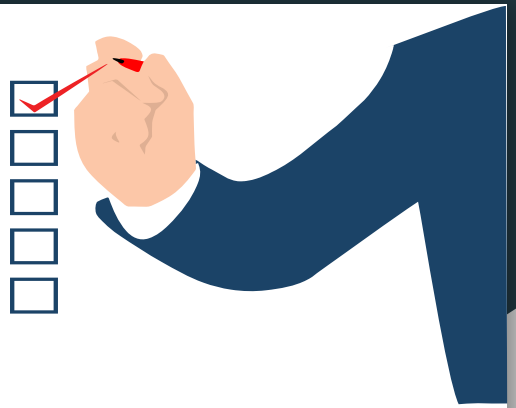


Dr Benabderrezak

Researcher in Cyber Security

Full-stack Web developer

Work Plan



1. Introduction to Object-Oriented DBMS
2. Object-Relational: SQL3 Language
3. Semi-Structured Data and XML
4. General Overview of Distributed Databases
5. Distributed Query Management
6. Distributed Transaction Management

2

Chapter 1

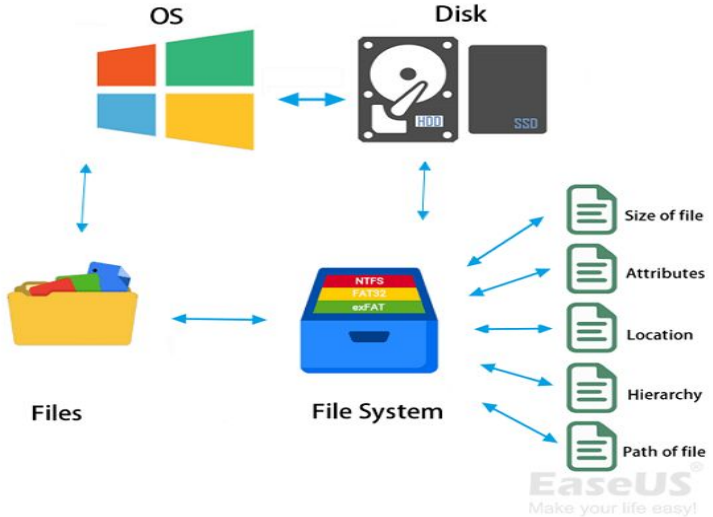
Introduction to Object-Oriented DBMS

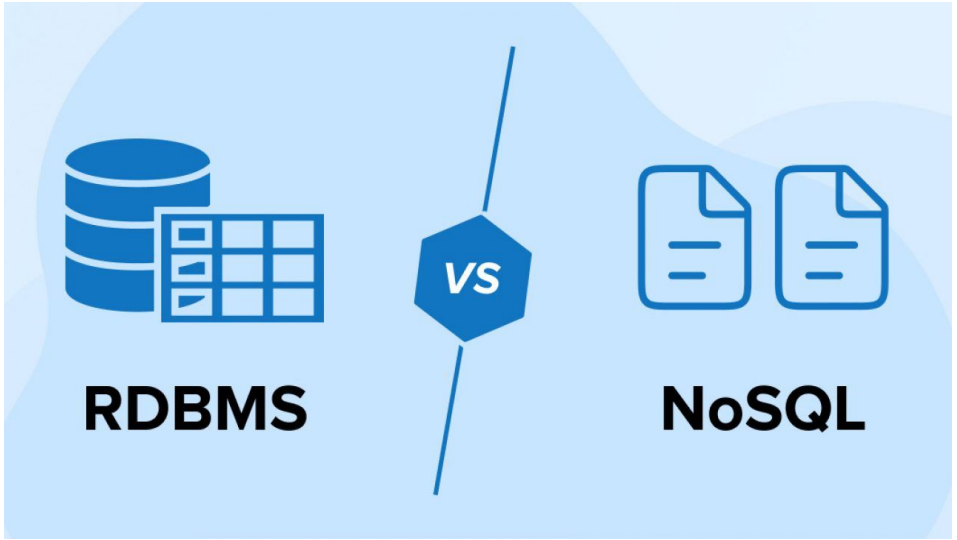




Evolution of DBMS

Database systems have evolved from **simple file systems** to relational databases and now to more complex object-oriented databases





Limitations of RDBMS

- Relational databases (RDBMS) are great for structured data but face challenges with complex, interrelated data like objects in software programming.
- They can't easily represent real-world entities and their relationships as objects do in object-oriented programming.

Object-Oriented DBMS - Overview & History

- Object-oriented databases (OODBMS) emerged in response to the need for databases to manage complex data more efficiently

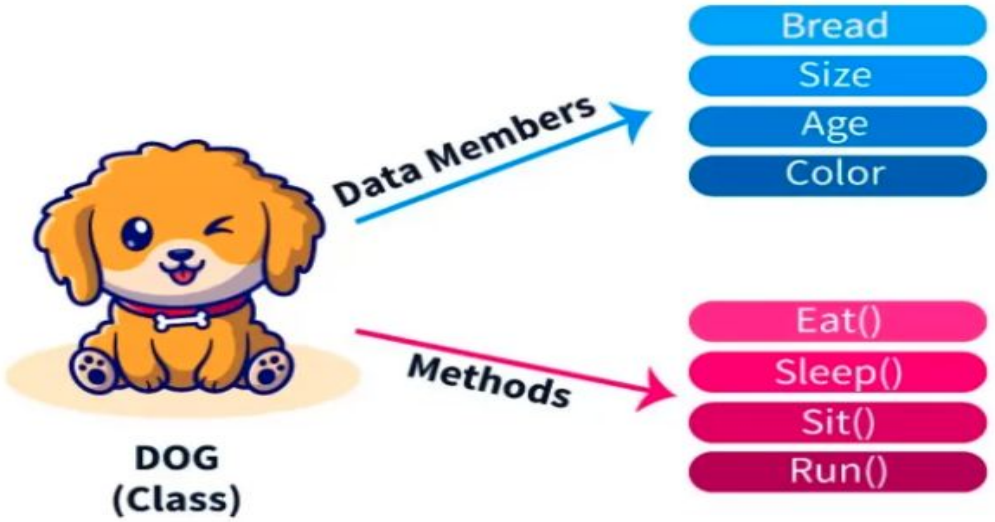
- combining the principles of databases and object-oriented programming, which became popular in the late 1980s

Fundamental Concepts of Object Models

- Object, Type, and Class
- Association
- Aggregation
- Composition
- Inheritance
- Dynamic Binding

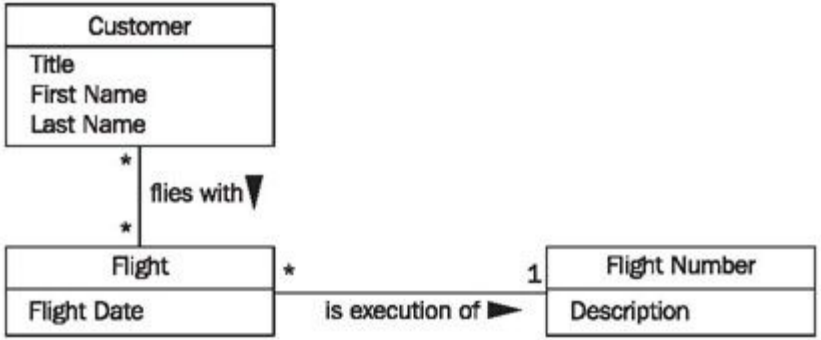
Object and Class

- In object-oriented systems, everything is treated as an object
- Each object belongs to a class that defines its structure (attributes) and behavior (methods)



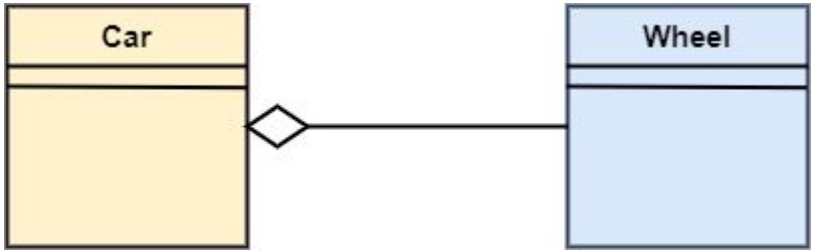
Association

- This represents relationships between objects
- like how a person object might be related to an address object



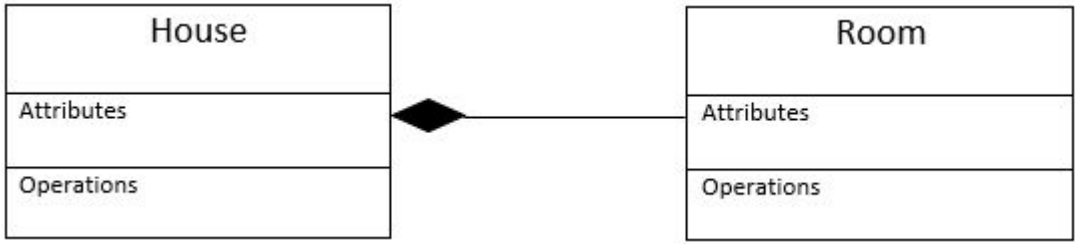
Aggregation

- Represents a "whole-part" relationship
- where one object is composed of other objects (e.g., a car is composed of wheels and an engine).



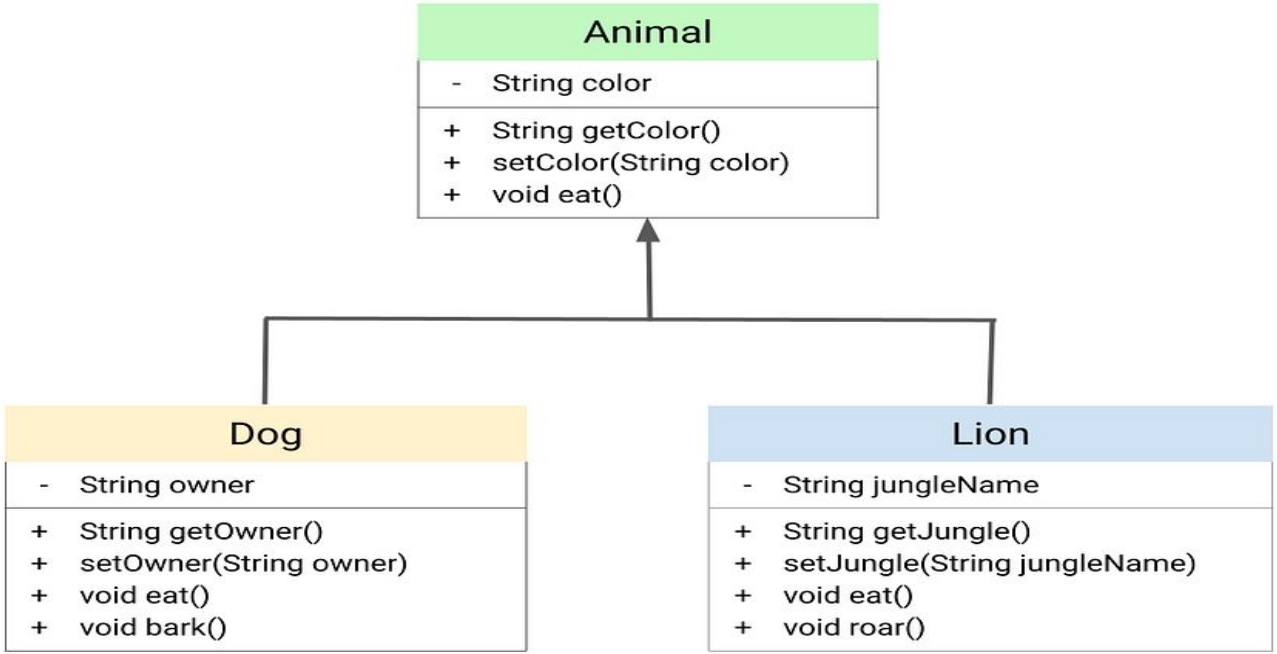
Composition

- A stronger form of aggregation
- where the parts cannot exist independently of the whole (e.g., a house and its rooms).



Inheritance

Allows new classes to take on properties and behaviors of existing classes, reducing redundancy



Dynamic Binding

The process of linking a procedure call to the code to be executed only at runtime, allowing more flexibility

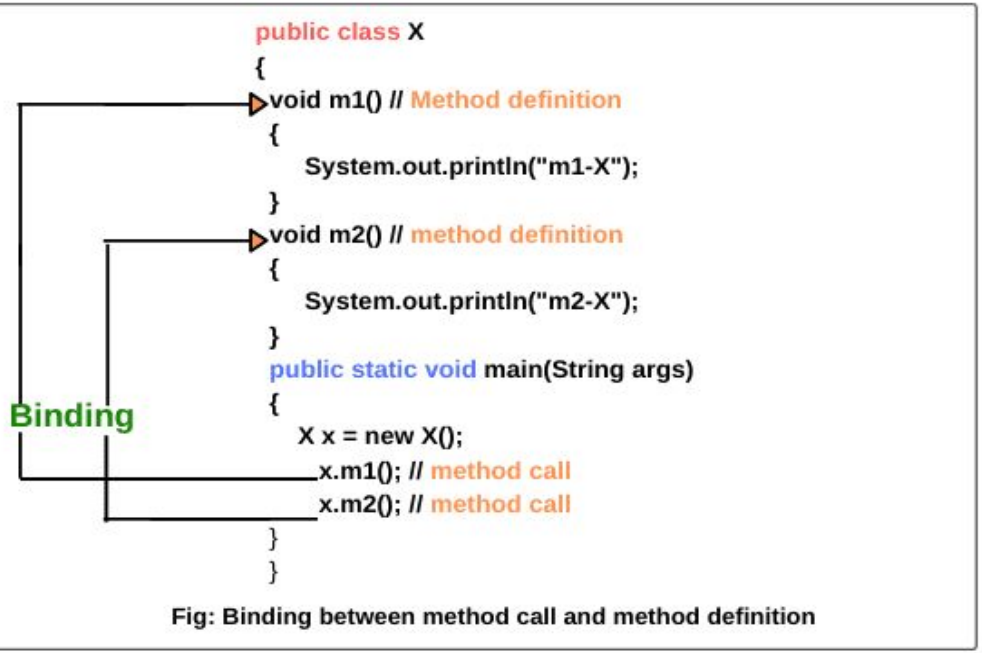


Fig: Binding between method call and method definition

Dynamic Binding :

- is a concept where the decision about which piece of code to run when calling a function or method is made while the program is running, not before

- This allows for more flexibility because the program can choose the correct code based on the specific type of object being used at that moment.

Dynamic Binding

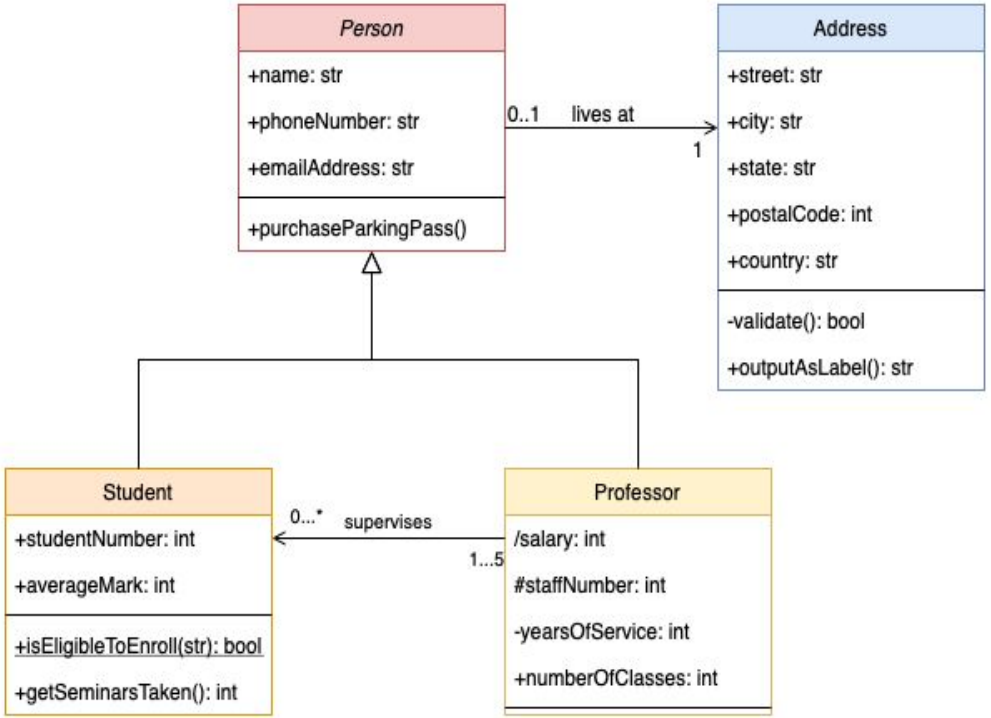
In simpler terms, it's like calling someone on the phone, but deciding which person to talk to only when they actually answer the call, instead of knowing beforehand

Object Modeling with UML (Unified Modeling Language)

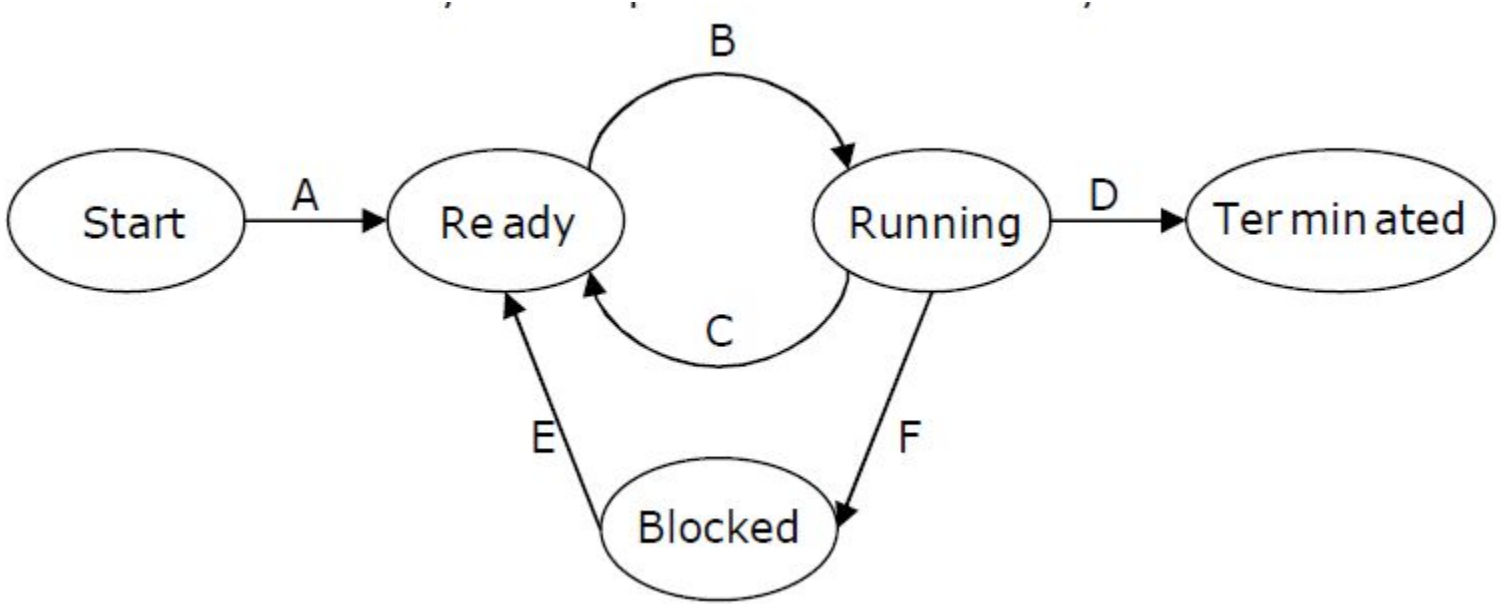
UML is a standardized visual language used to model and design software systems



Class Diagrams : Diagrams that show the structure of the system by illustrating the system's classes and their relationships



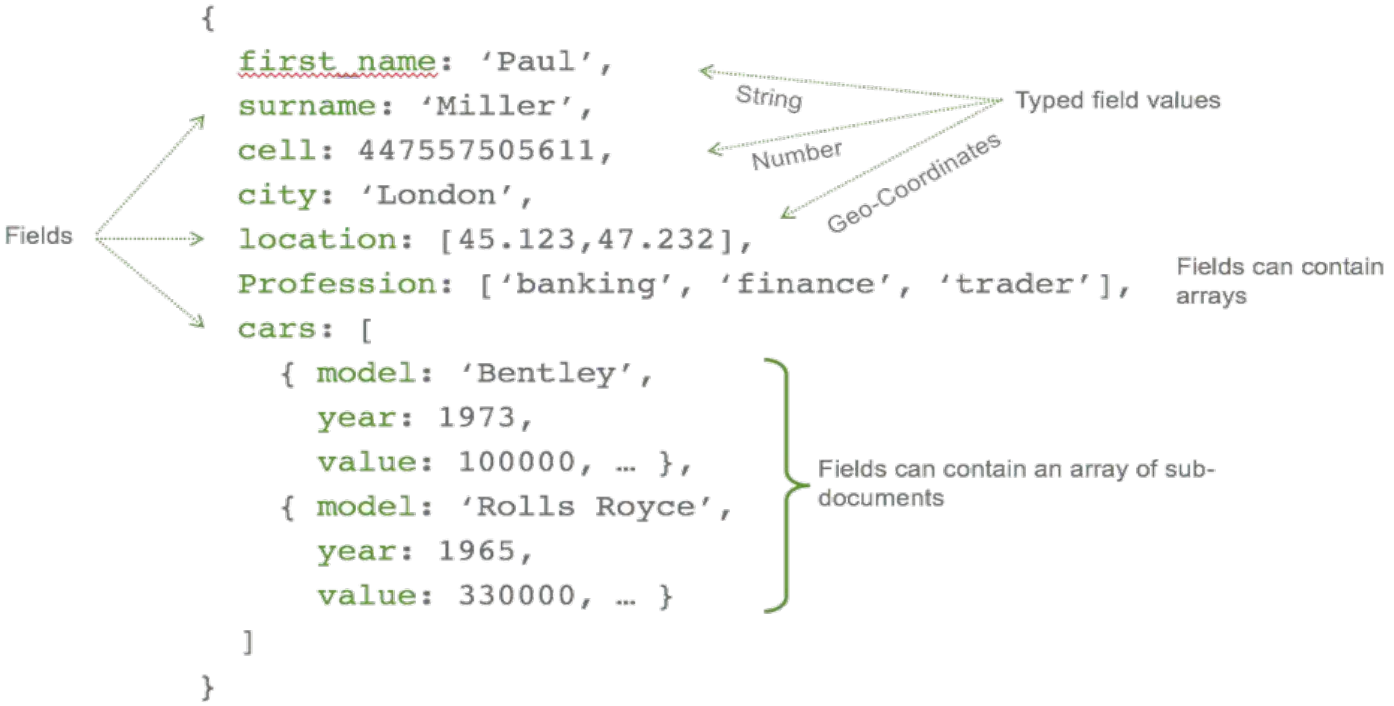
State/Transition Diagrams : Diagrams that depict how an object moves from one state to another, based on events that occur.



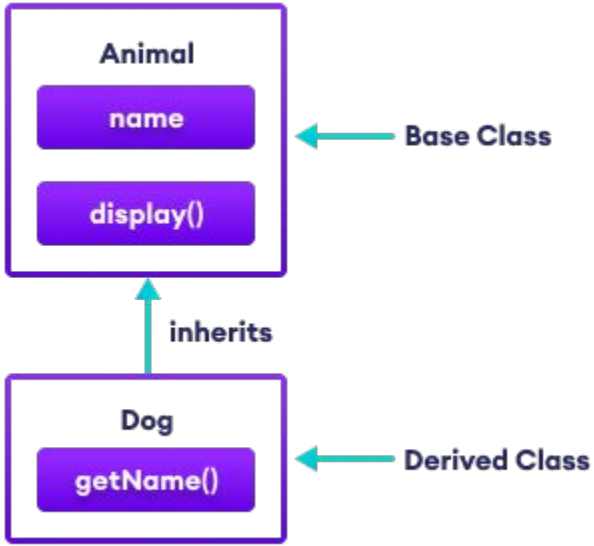
Characteristics of Object-Oriented DBMS

1. Object Support
2. Classes and Inheritance
3. Encapsulation
4. Polymorphism
5. Persistence
6. Complex Data Types

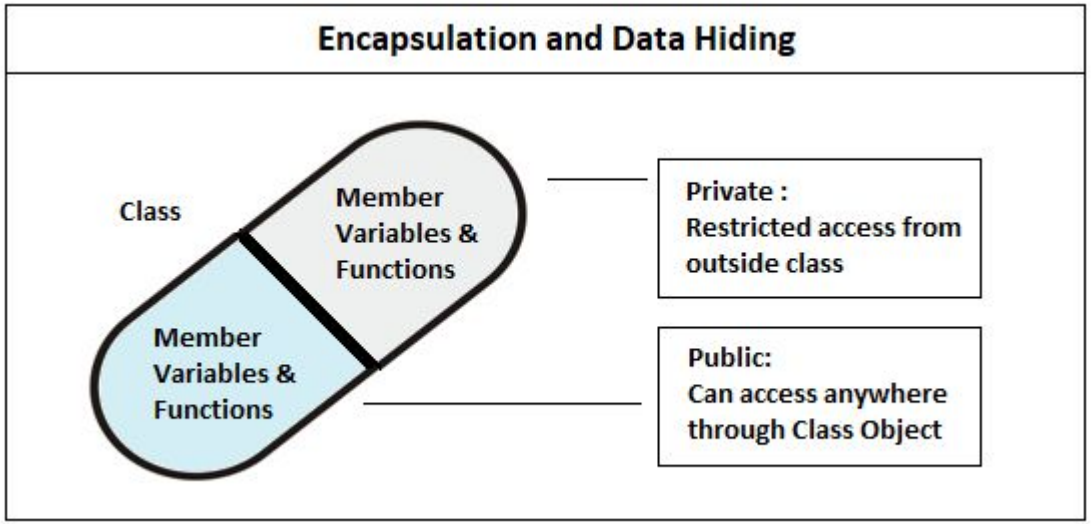
- **Object Support** : Data is stored as objects, similar to how it's managed in object-oriented programming.



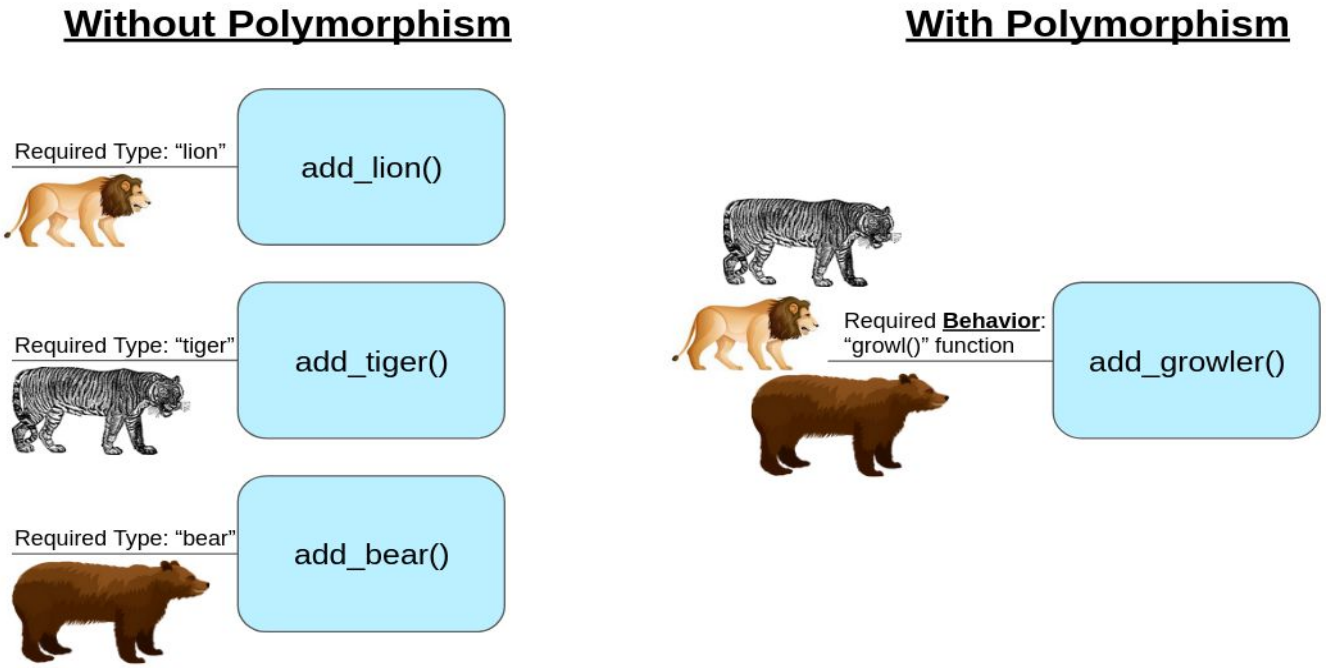
- **Classes and Inheritance** : Objects are grouped into classes, and these classes can inherit features from other classes, allowing reuse and better organization.



- **Encapsulation** : Objects bundle data (attributes) and the methods (functions) that operate on them, keeping them together



- **Polymorphism** : Different objects can respond to the same message or function call in their own way.



- **Persistence** : Objects remain stored in the database even after the program that created them ends.

- **Complex Data Types** : OODBMS can handle complex data types, such as multimedia, and relationships between data more easily than relational databases.

Chapter 2

Object-Relational : SQL3 Language



The SQL3 standard extends SQL to :

- support object-oriented features
- allowing databases to handle complex data types and relationships like objects in programming

Abstract Data Types and Object Tables

SQL3 allows custom data types and tables where rows are treated as objects with properties and methods.

Object Identification (OID)

Each object has a unique identifier (OID) that distinguishes it from others, like an ID number for rows in traditional databases.

Methods

Functions that define the behavior of objects within the database

Inheritance

SQL3 allows tables and types to inherit properties from other tables, similar to classes in object-oriented programming.

Large Object (LOB)

SQL3 supports handling large amounts of data, like images or videos, as objects in the database.

Object Views

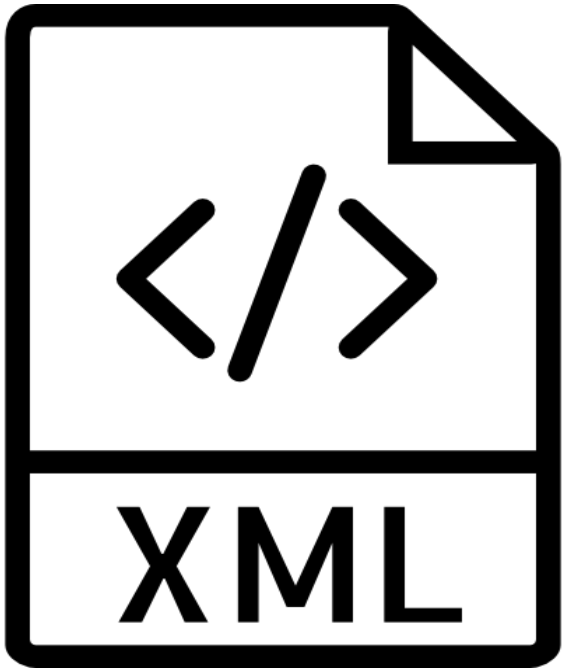
Views that treat data as objects, enabling more complex interactions with the data

Association in SQL3

SQL3 can define relationships between objects, like foreign keys in relational databases.

Chapter 3

Semi-Structured Data and XML

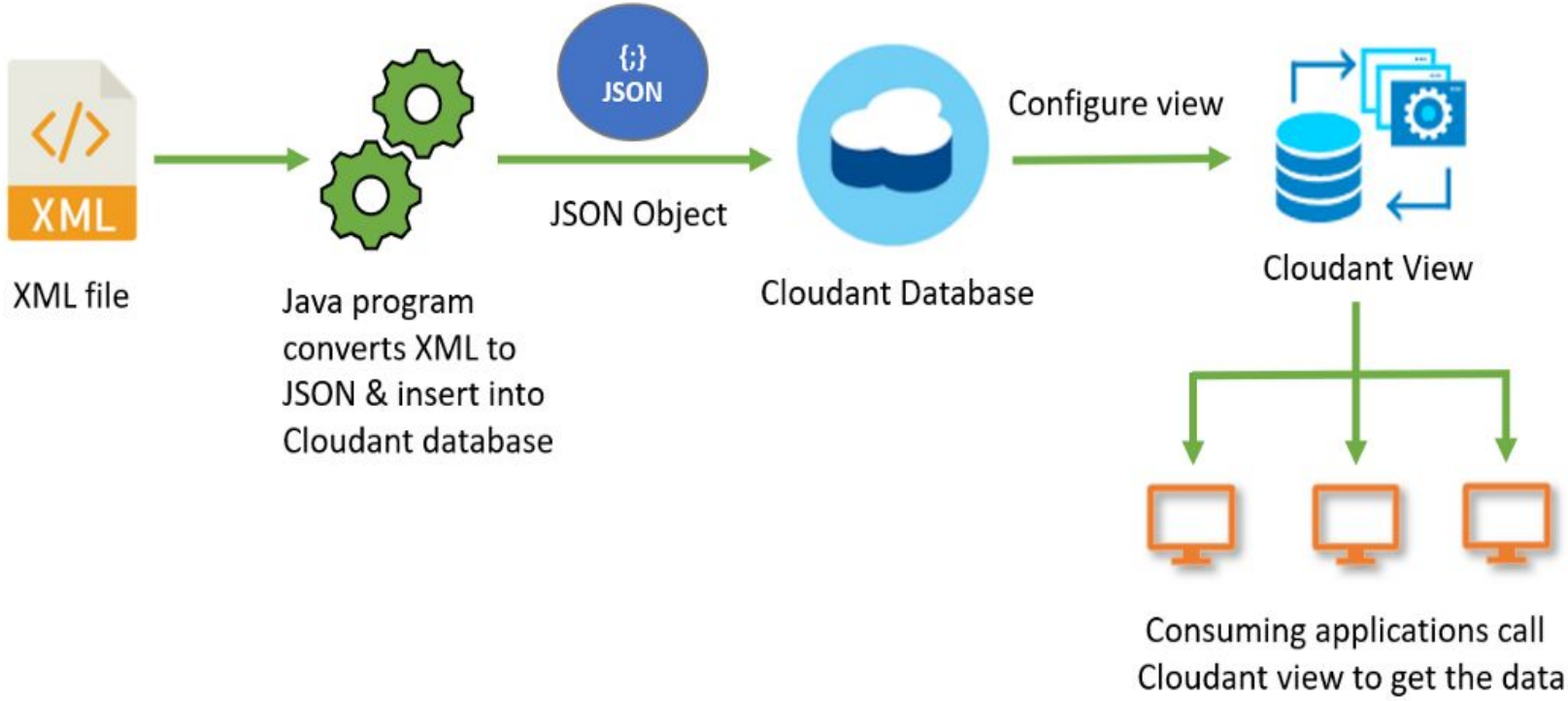


XML (Extensible Markup Language)

is used to describe data in a structured but flexible format, allowing users to define their own tags to represent data

```
<CATALOG>
   <SPRING>
      <TITLE>Garden Sales</TITLE>
      <LINE>Outdoor_Tools</LINE>
      <PAGE>
         <CAPTION>Goodbye, Winter!</CAPTION>
         <ITEM>Gardening Gloves</ITEM>
         <ITEM>Potting Soil</ITEM>
      </PAGE>
   </SPRING>
</CATALOG>
```

Databases can store and retrieve XML data, which is useful for handling documents or semi-structured data that doesn't fit neatly into rows and columns



- XML allows users to create their **own custom tags**
- making it a highly flexible language for data representation and sharing between systems

```
<objectPermissions>
    <allowCreate>true</allowCreate>
    <allowDelete>false</allowDelete>
    <allowEdit>true</allowEdit>
    <allowRead>true</allowRead>
    <modifyAllRecords>false</modifyAllRecords>
    <object>Awesom_O_4000__c</object>
    <viewAllRecords>true</viewAllRecords>
</objectPermissions>
```

XML Schema

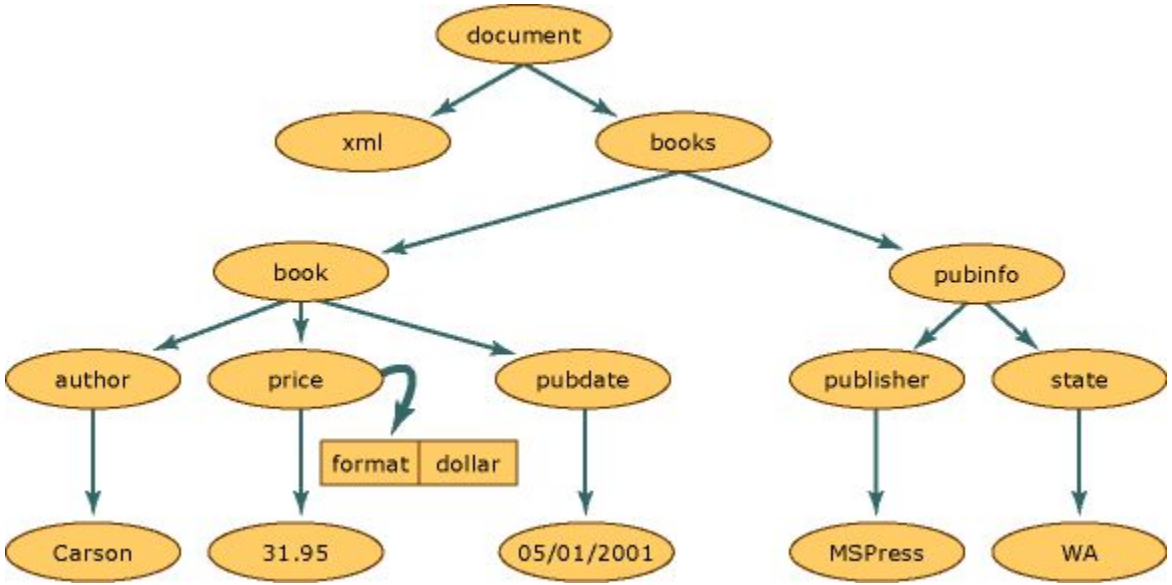
- defines the **structure** and **data types** within an XML document
- enforcing rules like which elements and attributes are required or what kind of data (text, number) they should contain

```
<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```



XML Data Model

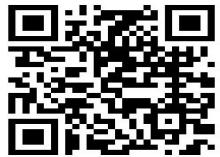
XDM is the underlying structure that describes how XML data is represented and manipulated



XQuery : A Query Language for XML Data

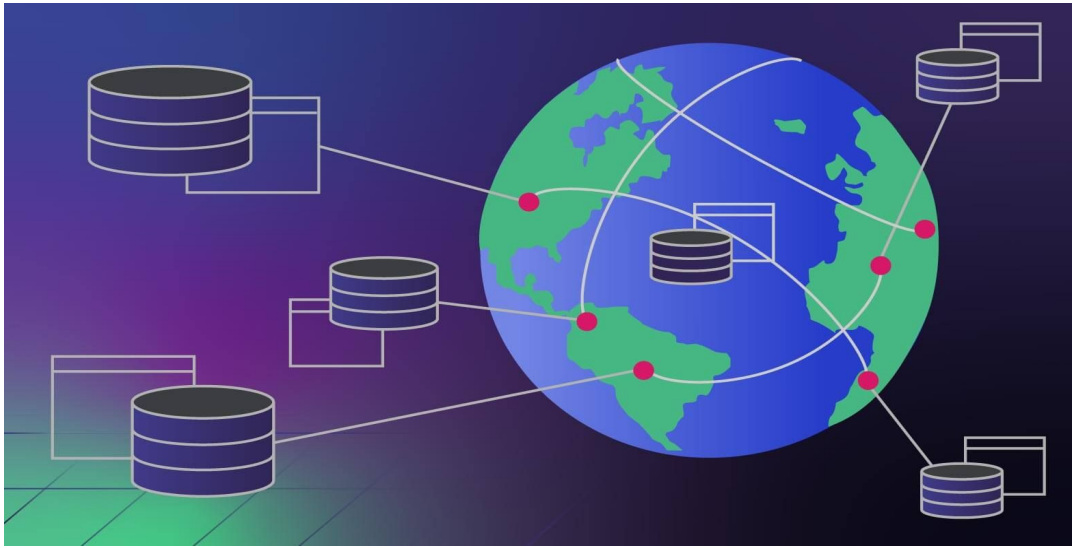
XQuery is a language designed to query XML data, similar to how SQL is used to query relational databases

```
for $x in doc("books.xml")/bookstore/book
where $x/price>30
order by $x/title
return $x/title
```



Chapter 4

General Overview of Distributed Databases



Distributed Databases

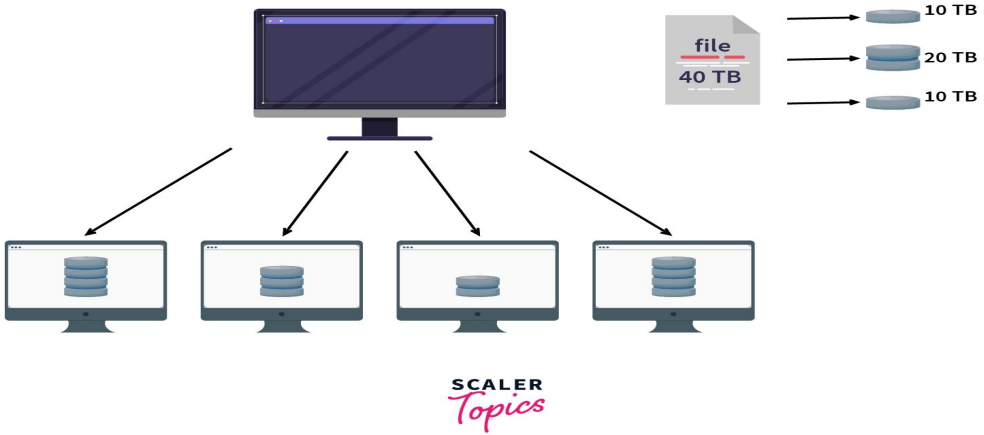
- A distributed database stores data across multiple locations, but it behaves as a single system to users.
- This increases efficiency, fault tolerance, and access speed

Distributed databases improve reliability, performance, and scalability by spreading data across several systems.



Distributed Database Management System (DDBMS)

A DDBMS manages the data in a distributed database, ensuring it functions as a cohesive system



Building a Distributed Database: Data Distribution Techniques

- **Fragmentation** : Splitting data into smaller pieces and distributing them across different locations
- **Horizontal** : Dividing data by rows
- **Vertical** : Dividing data by columns
- **Mixed** : Combining horizontal and vertical fragmentation

Architecture and Functions of a DDBMS

- The architecture defines how a DDBMS is structured, including its components and how they communicate.
- Its functions include data distribution, replication, and transaction management.

Transparencies in a DDBMS : 12 Rules by "DATE"

These rules aim to make the distributed nature of a database invisible to users, ensuring it behaves like a centralized system

Classification of Distributed Database Design Approaches:

Multi-Database Systems and Federated Systems

Multi-database systems involve managing multiple independent databases, while federated systems offer more integration, allowing them to work together as one.

Chapter 5

Distributed Query Management

Distributed Queries

Queries in a distributed system allow users to access and manipulate data stored in multiple locations as if it were in a single database.

Query Optimization

- **Global Optimization** : Ensures that a query is executed in the most efficient way across all distributed locations.

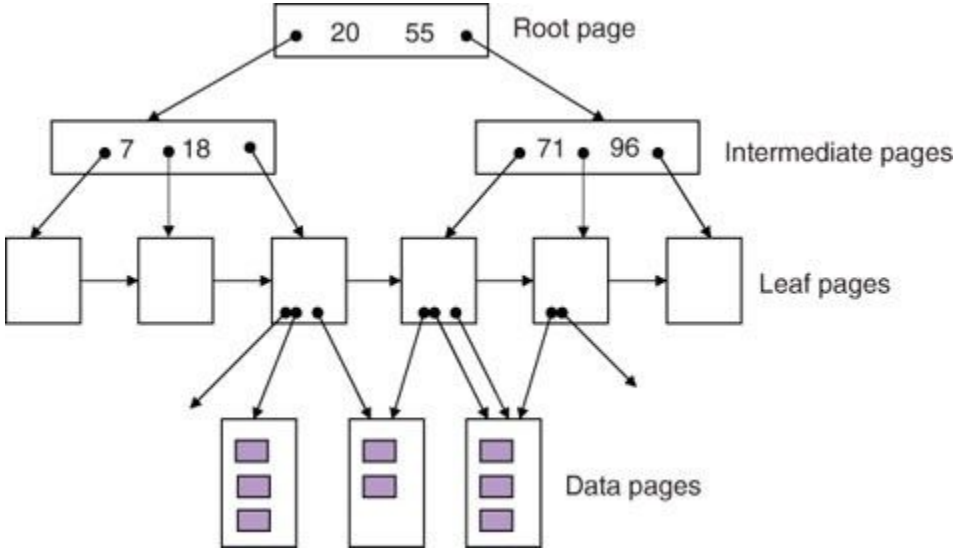
- **Local Optimization** : Focuses on optimizing individual parts of the query at each location.

Query Evaluation Strategies

- R* Optimization Algorithm (System R)

- Join Strategies.

R* Optimization Algorithm (System R) : A method for optimizing queries across distributed systems by considering data locations and costs

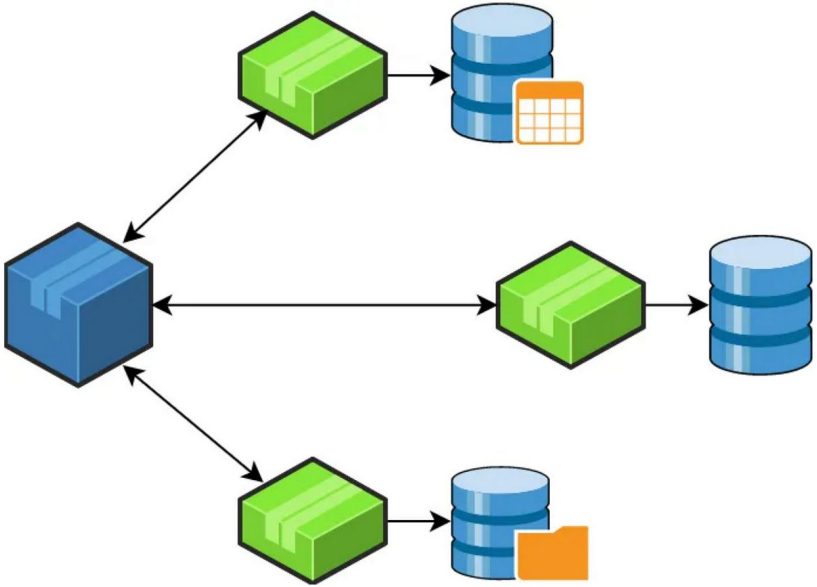


Join Strategies :

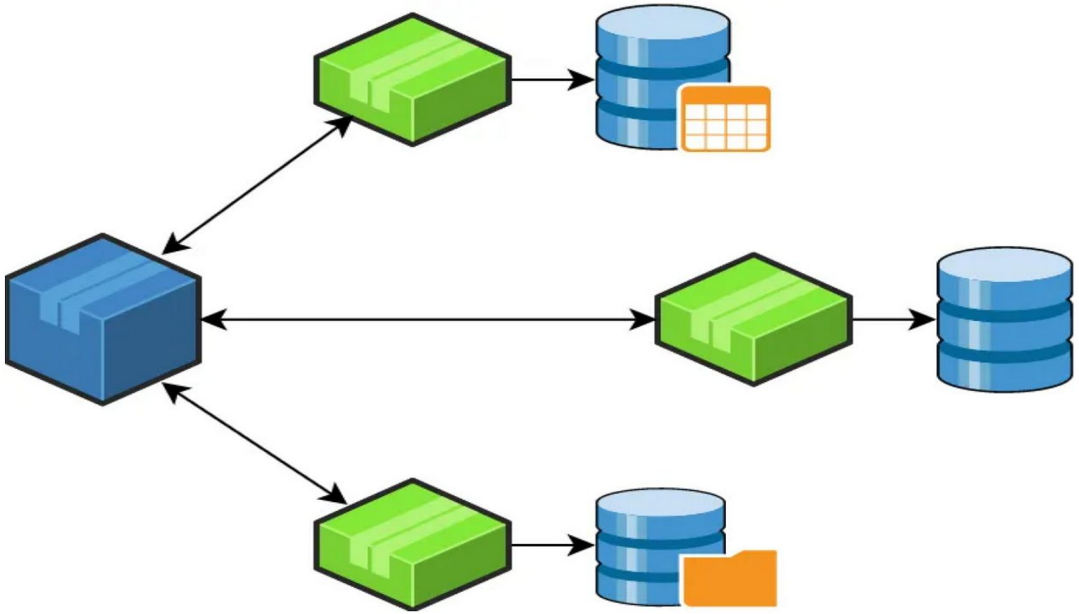
Techniques to efficiently combine data from different locations.

Chapter 6

Distributed Transaction Management



Distributed transactions involve changes to data in multiple locations, and the system ensures these changes happen reliably, even if parts of the system fail.



Concurrency Control

- Distributed systems must manage concurrent transactions without conflicts.
- Techniques like **distributed serialization**, **locking**, and timestamping ensure this.

Commit and Recovery

- **Two-Phase Commit (2PC)** : Ensures that either all parts of a transaction across multiple locations are committed or none of them are.

- **Three-Phase Commit (3PC)** : Adds an extra phase to improve fault tolerance in committing transactions.


THANK YOU
FOR YOUR
ATTENTION