UNIT -1

INTRODUCTION TO DATABASES AND DATA MODELING

INTRODUCTION TO DATABASE

Database is collection of data which is related by some aspect. Data is collection of facts and figures which can be processed to produce information. Mostly data represents recordable facts. Data aids in producing information which is based on facts.

A database management system stores data, in such a way which is easier to retrieve, manipulate and helps to produce information . So a database is a collection of related data that we can use for

- **Defining** - specifying types of data
- **Constructing** - storing & populating.
- **Manipulating** - querying, updating, reporting.

A DBMS is a collection of software programs that allows a user to define datatypes, structures, constraints, store data permanently, modify and delete operations.

DBMS is basically a software used to add, modify, delete, select data from database.

In simpler words, DBMS is a collection of interrelated data and software programs to access those data.

PURPOSE OF DATABASE SYSTEM

The typical file processing system is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files. A file processing system has a number of major disadvantages.

- Data redundancy and inconsistency
- Difficulty in accessing data

- Data isolation – multiple files and formats
- Integrity problems
- Atomicity of updates
- Concurrent access by multiple users
- Security problems

1. Data redundancy and inconsistency: In file processing, every user group maintains its own files for handling its data processing applications.

2. Difficulty in accessing data:File processing environments do not allow needed data to be retrieved in a convenient and efficient manner.

3. Data isolation :Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.

4. Integrity problems:The data values stored in the database must satisfy certain types of consistency constraints. Example: The balance of certain types of bank accounts may never fall below a prescribed amount . Developers enforce these constraints in the system by addition appropriate code in the various application programs

5. Atomicity problems: Atomic means the transaction must happen in its entirety or not at all. It is difficult to ensure atomicity in a conventional file processing system. Example: Consider a program to transfer $50 from account A to account B.

6.Concurrent access anomalies: For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously. In such an environment, interaction of concurrent updates is possible and may result in inconsistent data. To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult to provide because data may be accessed by many different application programs that have not been coordinated previously.

Example: When several reservation clerks try to assign a seat on an airline flight, the system should ensure that each seat can be accessed by only one clerk at a time for assignment to a passenger.

7.**Security problems**: Enforcing security constraints to the file processing system is

difficult.

APPLICATION OF DATABASE

Database Applications

- Banking: all transactions
- Airlines: reservations, schedules
- Universities: registration, grades
- Sales: customers, products, purchases
- Manufacturing: production, inventory, orders, supply chain
- Human resources: employee records, salaries, tax deductions
- Telecommunication: Call History, Billing
- Credit card transactions: Purchase details, Statements

NEEDS OF DATABSES

A Database Management System or DBMS is a software that runs allows proper storing, organizing and managing large amounts of data. It ensures data consistency, integrity and security while allowing multiple users to access and manipulate data simultaneously.

Traditional File Systems

In earlier times, data was stored and retrieved using files in a typical file system. For example:

- A company might keep separate files for employees details, customer information and daily sales.
- These files could be stored as text documents, spreadsheets or printed records in cabinets.

This approach worked fine for small amounts of data but became challenging as the volume of data increased. File systems were the natural choice for several reasons:

- **Simplicity**: It was easy to create and manage files without requiring specialized software.
- **Low Cost**: There was no need to invest in additional tools or training to use file systems.
- **Direct Access**: Users could access files directly from storage devices.

Limitations of File-based Systems

1. Data Redundancy (Duplicate Data)

- The same data would often be stored in multiple files.
- **Example**: A customer's address might appear in both the "Orders" file and the "Customer Details" file, leading to unnecessary duplication.

2. Data Inconsistency

- When data changes in one file but not in others, it results in mismatched information.
- **Example**: If a customer updates their phone number, but it's updated only in the "Customer Details" file and not in the "Orders" file, the records become inconsistent.

3. Difficulty in Data Retrieval

- Retrieving specific information from a file required manual effort or complex programming.
- **Example:** Finding all orders made by a customer in the last year could take hours if the data was scattered across multiple files.

4. Limited Security

- File systems offered no advanced security features to control access.
- **Example:** Any employee with access to the storage could view sensitive information.

5. No Support for Relationships Between Data

- Relationships between data points (like linking customer details with orders) were hard to establish.
- **Example**: Connecting a customer to their purchase history would require manually cross-referencing multiple files.

Components of DBMS

- DBMS have several components, each performing very significant tasks in the database management system environment. Below is a list of components within the database and its environment.

- **Software**

  This is the set of programs used to control and manage the overall database. This includes the DBMS software itself, the Operating System, the network software being used to share the data among users, and the application programs used to access data in the DBMS.

- **Hardware**

  Consists of a set of physical electronic devices such as computers, I/O devices, storage devices, etc., this provides the interface between computers and the real world systems.

- **Data**

  DBMS exists to collect, store, process and access data, the most important component. The database contains both the actual or operational data and the metadata.

- **Procedures**

  These are the instructions and rules that assist on how to use the DBMS, and in designing and running the database, using documented procedures, to guide the users that operate and manage it.

- **Database Access Language**

  This is used to access the data to and from the database, to enter new data, update existing data, or retrieve required data from databases. The user writes a set of appropriate commands in a database access language, submits these to the DBMS, which then processes the data and generates and displays a set of results into a user readable form.

- **Query Processor**

  This transforms the user queries into a series of low level instructions. This reads the online user's query and translates it into an efficient series of operations in a form capable of being sent to the run time data manager for execution.

TYPES OF DATABASE USERS AND DATABASE SYSTEM ARCHITECTURE:

A DBMS architecture defines how users interact with the database to read, write, or update information. A well-designed architecture and schema (a blueprint detailing tables, fields and relationships) ensure data consistency, improve performance and keep data secure.

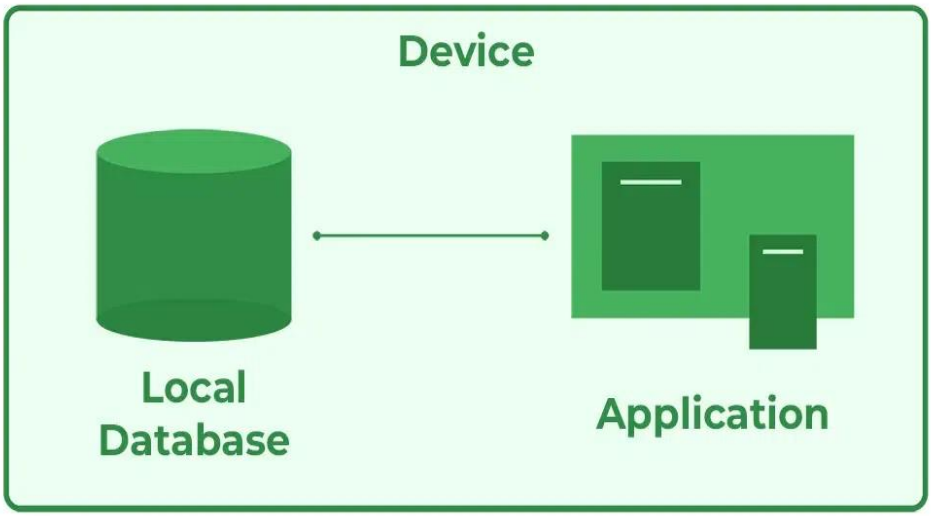
Types of DBMS Architecture

There are several types of DBMS Architecture that we use according to the usage requirements.

- 1-Tier Architecture
- 2-Tier Architecture
- 3-Tier Architecture

1-Tier Architecture

In 1-Tier Architecture, the user works directly with the database on the same system. This means the client, server and database are all in one application. The user can open the application, interact with the data and perform tasks without needing a separate server or network connection.



- A common example is Microsoft Excel. Everything from the user interface to the logic and data storage happens on the same device. The user enters data, performs calculations and saves files directly on their computer.
- This setup is simple and easy to use, making it ideal for personal or standalone applications. It does not require a network or complex setup, which is why it's often used in small-scale or individual use cases.
- This architecture is simple and works well for personal, standalone applications where no external server or network connection is needed.

Advantages of 1-Tier Architecture

Below mentioned are the advantages of 1-Tier Architecture.

- **Simple Architecture:** 1-Tier Architecture is the most simple architecture to set up, as only a single machine is required to maintain it.
- **Cost-Effective:** No additional hardware is required for implementing 1-Tier Architecture, which makes it cost-effective.
- **Easy to Implement:** 1-Tier Architecture can be easily deployed and hence it is mostly used in small projects.

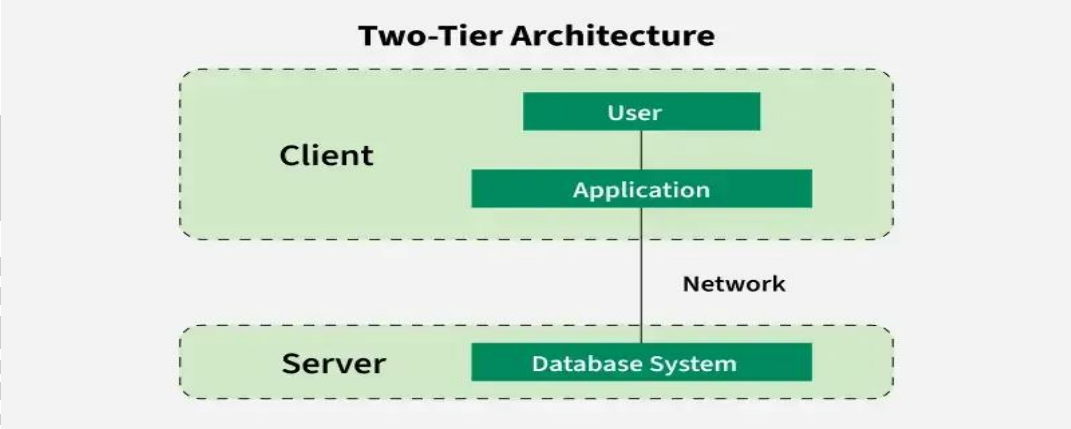
Disadvantages of 1-Tier Architecture

- **Limited to Single User:** Only one person can use the application at a time. It’s not designed for multiple users or teamwork.
- **Poor Security:** Since everything is on the same machine, if someone gets access to the system, they can access both the data and the application easily.
- **No Centralized Control:** Data is stored locally, so there's no central database. This makes it hard to manage or back up data across multiple devices.

- **Hard to Share Data:** Sharing data between users is difficult because everything is stored on one computer.

2-Tier Architecture

The 2-tier architecture is similar to a basic client-server model. The application at the client end directly communicates with the database on the server side. APIs like ODBC and JDBC are used for this interaction. The server side is responsible for providing query processing and transaction management functionalities.



- On the client side, the user interfaces and application programs are run. The application on the client side establishes a connection with the server side to communicate with the DBMS. For Example: A Library Management System used in schools or small organizations is a classic example of two-tier architecture.
- **Client Layer (Tier 1):** This is the user interface that library staff or users interact with. For example they might use a desktop application to search for books, issue them, or check due dates.
- **Database Layer (Tier 2):** The database server stores all the library records such as book details, user information and transaction logs.
- The client layer sends a request (like searching for a book) to the database layer which processes it and sends back the result. This separation allows the client to focus on the user interface, while the server handles data storage and retrieval.

Advantages of 2-Tier Architecture

- **Easy to Access:** 2-Tier Architecture makes easy access to the database, which makes fast retrieval.
- **Scalable:** We can scale the database easily, by adding clients or upgrading hardware.

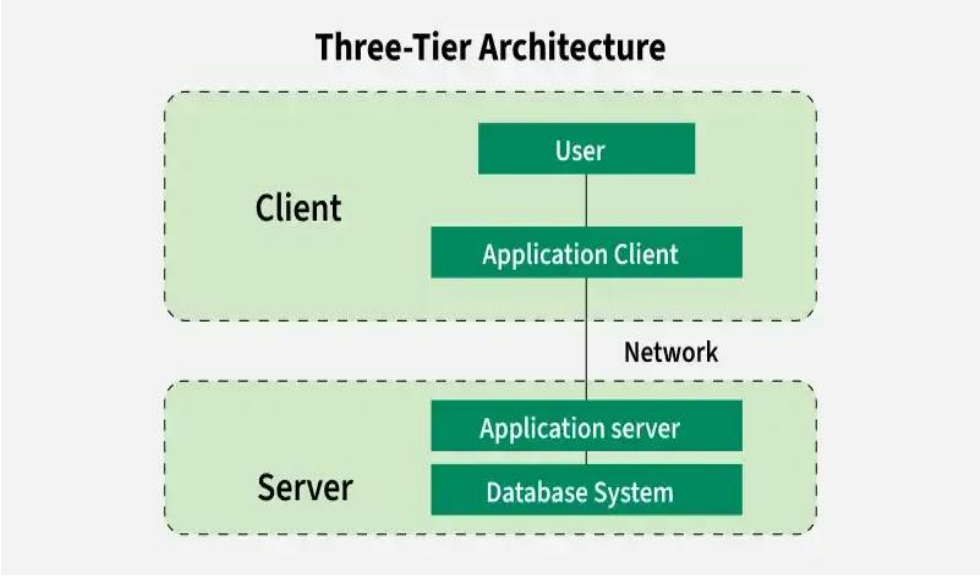
- **Low Cost:** 2-Tier Architecture is cheaper than 3-Tier Architecture and Multi-Tier Architecture.
- **Easy Deployment:** 2-Tier Architecture is easier to deploy than 3-Tier Architecture.
- **Simple:** 2-Tier Architecture is easily understandable as well as simple because of only two components.

Disadvantages of 2-Tier Architecture

- **Limited Scalability:** As the number of users increases, the system performance can slow down because the server gets overloaded with too many requests.
- **Security Issues:** Clients connect directly to the database, which can make the system more vulnerable to attacks or data leaks.
- **Tight Coupling:** The client and the server are closely linked. If the database changes, the client application often needs to be updated too.
- **Difficult Maintenance:** Managing updates, fixing bugs, or adding features becomes harder when the number of users or systems increases.

3-Tier Architecture

3 tier architecture, there is another layer between the client and the server. The client does not directly communicate with the server. Instead, it interacts with an application server which further communicates with the database system and then the query processing and transaction management takes place. This intermediate layer acts as a medium for the exchange of partially processed data between the server and the client. This type of architecture is used in the case of large web applications.



Example: E-commerce Store

- **User**: You visit an online store, search for a product and add it to your cart.
- **Processing**: The system checks if the product is in stock, calculates the total price and applies any discounts.

- **Database**: The product details, your cart and order history are stored in the database for future reference.

Advantages of 3-Tier Architecture

- **Enhanced scalability:** Scalability is enhanced due to the distributed deployment of application servers. Now, individual connections need not be made between the client and server.
- **Data Integrity:** 3-Tier Architecture maintains Data Integrity. Since there is a middle layer between the client and the server, data corruption can be avoided/removed.
- **Security:** 3-Tier Architecture Improves Security. This type of model prevents direct interaction of the client with the server thereby reducing access to unauthorized data.

Disadvantages of 3-Tier Architecture

- **More Complex:** 3-Tier Architecture is more complex in comparison to 2-Tier Architecture. Communication Points are also doubled in 3-Tier Architecture.
- **Difficult to Interact:** It becomes difficult for this sort of interaction to take place due to the presence of middle layers.
- **Slower Response Time:** Since the request passes through an extra layer (application server), it may take more time to get a response compared to 2-Tier systems.
- **Higher Cost:** Setting up and maintaining three separate layers (client, server and database) requires more hardware, software and skilled people. This makes it more expensive.

DATA MODELS:

A Data Model in Database Management System (DBMS) is the concept of tools that are developed to summarize the description of the database. Data Models provide us with a transparent picture of data which helps us in creating an actual database. It shows us from the design of the data to its proper implementation of data.

TYPES OF DATABASE:

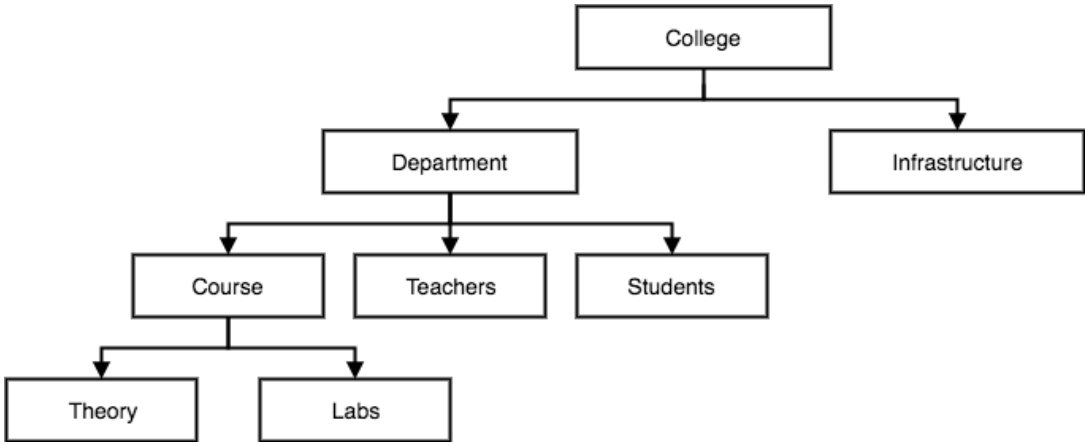
- Hierarchical Database
- Network Database
- Relational Database
- Object Oriented Database

1. Hierarchical Model

- The hierarchical Model is one of the oldest models in the data model which was developed by IBM. In a hierarchical model, data are viewed as a collection of tables, or we can say segments that form a hierarchical relation. the data is

organized into a tree-like structure where each record consists of one parent record and many children.

- Even if the segments are connected as a chain-like structure by logical associations, then the instant structure can be a fan structure with multiple branches. We call the illogical associations as directional associations.



Applications of Hierarchical Model

- **Content Management Systems (CMS):** Many CMSs use hierarchical databases to manage and organize content such as pages, articles, and media files in a structured manner. This allows for efficient retrieval and management of content that is nested within categories and subcategories.
- **Organizational Structures:** Businesses often utilize hierarchical databases to mirror their organizational charts. This helps in efficiently managing employee records, departmental data, and company resources that follow a clear hierarchical structure.
- **File Systems:** Operating systems often employ a hierarchical model to manage files and directories. This structure allows users to navigate through folders and find files easily, following a clear path from a root directory to subdirectories and files.
- **Information Systems (GIS):** Hierarchical databases are used in GIS to manage spatial data that is layered as in the case **Geographic** of maps where different layers (like terrain, roads, landmarks) are organized hierarchically

Advantages of the Hierarchical Model

- **Simplicity:** The tree-like structure of the hierarchical model is intuitive and easy to understand, which simplifies the design and navigation of databases. This structure clearly defines parent-child relationships, making the path to retrieve data straightforward.

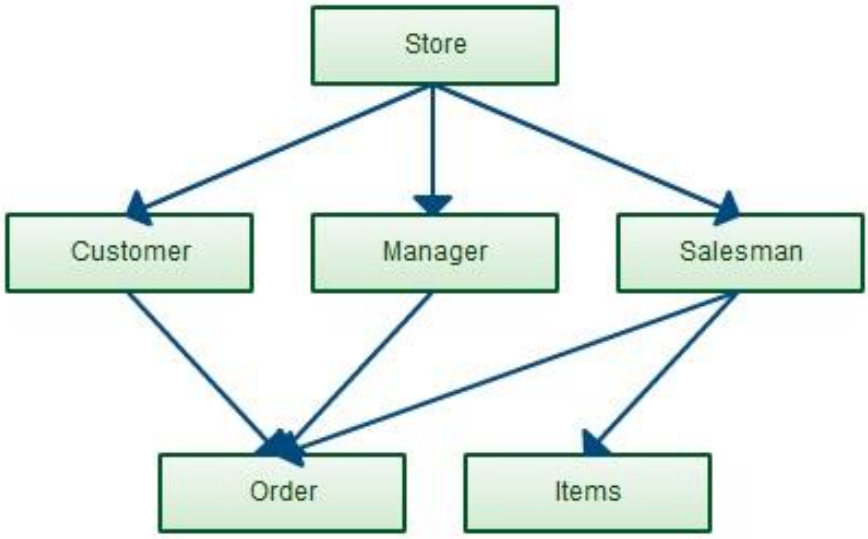
- **Data Integrity:** Due to its rigid structure, the hierarchical model inherently maintains data integrity. Each child record has only one parent, which helps prevent redundancy and preserves the consistency of data across the database.
- **Efficient Data Retrieval:** The model allows for fast and efficient retrieval of data. Since relationships are predefined, navigating through the database to find a specific record is quicker compared to more complex models where paths are not as clearly defined.
- **Security:** Hierarchical databases can provide enhanced security. Access to data can be controlled by restricting users to certain segments of the tree, which makes it easier to implement security policies based on data hierarchy.
- **Performance:** For applications where relationships are fixed and transactions require high throughput, such as in banking or airline reservation systems, the hierarchical model offers excellent performance and speed.

Limitations of the Hierarchical Model

- **Structural Rigidity:** The biggest limitation of the hierarchical model is its structural rigidity. Once the database is designed, making changes to its structure, such as adding or modifying the relationships between records, can be complex and labor-intensive.
- **Lack of Flexibility:** This model does not easily accommodate many-to-many relationships between records, which are common in more complex business applications. This limitation can lead to redundancy and inefficiencies when trying to model more intricate relationships.
- **Data Redundancy:** In cases where a child has more than one parent, the hierarchical model can lead to data duplication because each child-parent relationship needs to be stored separately. This redundancy can consume additional storage and complicate data management.
- **Complex Implementation:** Implementing a hierarchical database can be more complex than more modern relational databases, particularly when dealing with large sets of data that have diverse and intricate relationships.
- **Query Limitations:** The hierarchical model typically requires a specific type of query language that might not be as rich or flexible as SQL, used in relational databases. This can limit the types of queries that can be executed, affecting the ease and depth of data analysis.

Network Data Model

- Organizes the data in a Graph Structure
- Relationships among data are represented by links
- Data is represented by a collection of record types



ADVANTAGES

- **Handles Complex Relationships:**

The network model excels at representing many-to-many relationships, which are common in real-world scenarios, making it more flexible than the hierarchical model.

- **Data Access Flexibility:**

  It allows for multiple access paths to data, leading to faster data retrieval compared to the hierarchical model.

- **Data Integrity:**

  The model can enforce data integrity by ensuring that a member record cannot exist without a corresponding owner record.

- **Conceptual Simplicity:**

  Similar to the hierarchical model, it's conceptually simple and easy to design.

DISADVANTAGES

- **System Complexity:**

  The use of pointers to represent relationships can make the database structure complex to understand and manage.

- **Structural Changes:**

  Modifying the database structure can be difficult and may require changes to the application programs.

- **Operational Anomalies:**

Insertion, deletion, and updating operations can be inefficient due to the need for pointer adjustments.

- **Scalability Challenges:**

  Performance can degrade when dealing with large datasets or complex graphs.

- **Lack of Structural Independence:**

  Changes to the database structure can impact the application code, requiring significant rework.

Relational Data Model.

- The relational model uses a collection of tables to represent both data and the relationships among those data.
- name Each table has multiple columns, and each column has a unique.
- Tables are also known as **relations**.
- The relational model is an example of a record-based model.

- Record-based models are so named because the database is structured in fixed- format records of several types.
- Each table contains records of a particular type.
- Each record type defines a fixed number of fields, or attributes.
- The columns of the table correspond to the attributes of the record type.

Advantages of the Relational Model:

- **Simplicity and Ease of Use:** The table-based structure is intuitive and easy to understand, making it user-friendly.
- **Data Integrity:** Relationships between tables enforce data consistency and accuracy. Constraints like primary and foreign keys prevent invalid data.
- **Data Independence:** Changes to the physical storage of data do not necessarily affect how data is accessed or used.
- **Flexibility:** The model allows for easy addition, deletion, and modification of data.
- **Standardized Query Language (SQL):** SQL provides a powerful and consistent way to interact with relational databases.
- **Security:** Access control mechanisms (user privileges) help protect sensitive data.

Disadvantages of the Relational Model:

- Complexity with Complex Data: Representing complex, hierarchical, or graph-like data structures can be challenging.
- Performance Issues with Large Datasets: Complex queries, especially those involving many joins, can become slow on very large databases.
- Storage Overhead: Normalization (a process to reduce redundancy) can lead to more tables and potentially larger storage requirements.
- Potential for Inconsistency: Without proper design and implementation, data inconsistencies can occur.
- Learning Curve: SQL can have a steep learning curve for beginners.

Object-Oriented Data Model

Object-oriented data model that can be seen as extending the E-R model with notions of encapsulation, methods , and object identity.

The object-relational data model combines features of the object-oriented data model and relational data model.

Advantages of Object-Oriented Data Models:

- **Complex Data Handling:**

  OODMs excel at representing complex, real-world entities and their relationships, making them suitable for applications like multimedia databases or CAD systems.

- **Improved Performance:**

  By reducing the impedance mismatch between programming languages and the database, OODMs can offer performance benefits, especially for complex queries involving object navigation.

- **Enhanced Reusability:**

  Inheritance and encapsulation allow for code reuse and easiermaintenance, reducing development time and costs.

KEYS

A key allows us to identify a set of attributes and thus distinguishes entities from each other. Keys also help to uniquely identify relationships, and thus distinguish relationships from each other.

Different types of keys are:

- Super key
- Candidate key
- Primary key
- Foreign key

1.Super key: A **superkey** is a set of one or more attributes that, taken collectively, allow us to identify uniquely a tuple in the relation. For example, consider the student relation.

Student (Rollno, Name, Age) is a *Super key*.

2.Candidate key: A table which have more than one attribute that uniquely identify an instance of an entity set. These attributes are called *Candidate keys*.

For example, Consider the car relation,

car (license_no,engine_serial_no,make, model, year)

In this relation, we can find the two *Candidate keys*:

license_no and engine_serial_no.

3.Primary key: An attribute which is unique and not null, can identify an instance of the entity set is termed as *Primary key*.

For example, Consider the employee relation,

Employee (eno, ename, sex, doj, dob, sal, job, dno) in this eno is the *Primary key.*

4.Foreign key: An attribute in one relation whose value matches the primary key in some other relation is called a *Foreign key*.

For example, Consider the two relations dept and employee,

Dept (dno, dname, dloc)

Employee (eno, ename, sex, doj, dob, sal, job, dno)

In the above relations, for dept relation dno is the *primary key*, for employee relation eno is the primary key

primary key and here we can find that the employee relation - dno matches with the dept relation, so that employee relation dno is known as *Foreign key*.