Types of Database

1. Relational Databases

What are Relational Databases?

Relational databases are based on the **Relational Model**, which was introduced in the **1970s**. Even though they are old, they are still **widely used today**.

They are also called **RDBMS (Relational Database Management Systems)**.

How data is stored?

- Data is stored in **tables (rows and columns)**
- Tables are connected using **keys**
  - **Primary Key** → uniquely identifies a row
  - **Foreign Key** → connects one table to another

Tables can be **JOINed** using foreign keys.

Example (Real World)

Imagine an **online shopping website**:

- **User Table** → user_id, name, email
- **Purchases Table** → purchase_id, user_id, product, price

Using **user_id**, we can join both tables and know
which user bought which product

Operations in Relational Databases

They use **SQL (Structured Query Language)** for:

- CREATE (insert data)
- READ (select data)
- UPDATE (modify data)
- DELETE (remove data)

Popular Relational Databases

- **MySQL**
- **Microsoft SQL Server**
- **Oracle**

Advantages of Relational Databases

1.Very popular & widely used

- Used continuously since the **1970s**
- Strong and trusted systems

2.Best for structured data

- Data in fixed format (rows & columns)

3.Strong data normalization

- Avoids data duplication
- Maintains data consistency

4.Powerful & standard query language (SQL)

- Easy to learn
- Same SQL works almost everywhere

5.Highly optimized

- Fast queries
- Efficient indexing

Disadvantages / Limitations

1.Scalability issues

- Difficult to scale **horizontally** (adding more servers)

2.Complexity increases with big data

- When data becomes huge:
  - Joins become heavy
  - System becomes complex
  - Performance may decrease

Real-world example

Think of a **school record system**:

- Student table
- Marks table
- Fees table

Everything is well-structured and connected.
But if the school grows to **millions of students**, managing and scaling becomes difficult.

2. Object-Oriented Databases

What are Object-Oriented Databases?

Object-Oriented Databases are based on the **Object-Oriented Programming (OOP) model**, which is widely used today.

In this type of database, **data is stored as objects**, not as rows and tables.

Key OOP Concepts used

Object-oriented databases use core OOP ideas like:

- **Encapsulation** → data + methods are bundled together
- **Object Identity** → every object has a unique identity
- **Inheritance** → child objects can reuse parent properties
- **Methods** → functions operate directly on data

📌 These concepts clearly **differentiate OODB** from the E-R and relational models.

Why Object-Oriented Databases?

Sometimes databases become **very complex**, with many relations and joins.
Maintaining relationships between many tables becomes **difficult and confusing**.

Object-oriented databases solve this by keeping **everything related inside one object**.

How data is stored?

- Data is stored as **objects**
- An object contains:
  - Data (attributes)
  - Functions (methods)
- No need to break data into multiple tables

Real-world example

Think of a **Car object**:

- Data → engine, color, model
- Methods → start(), stop(), accelerate()

In an object-oriented database, the **entire car** is stored as **one object**,
not spread across multiple tables.

Advantages of Object-Oriented Databases

1.Easy & fast data storage and retrieval

- No heavy joins
- Direct access to objects

2.Handles complex data easily

- Images, videos, audio, nested data
- Supports many data types

3.Best for real-world problem modeling

- Natural mapping with real-life objects

4.Works smoothly with OOP languages

- Java, C++, Python
- No object-relational mismatch

Disadvantages of Object-Oriented Databases

1.High complexity

- Read, write, update, delete operations can be slower

2.Less community support

- Not as popular as relational databases
- Fewer tools and developers

3.No view support

- Unlike relational databases, views are not supported

Examples of Object-Oriented Databases

- **ObjectDB**
- **GemStone**

3. NoSQL Databases

What are NoSQL Databases?

NoSQL means **“Not Only SQL”**.

NoSQL databases do **not store data in tables** like relational databases.
Instead, they store data in **different formats**, depending on the use case.

They are designed to handle:

- **Huge data**
- **High traffic**
- **Fast growth**

Types of NoSQL Databases

NoSQL databases are mainly of **four types**:

1.Key–Value

- Data stored as key : value
- Example: userID : userData

2.Document-based

- Data stored in JSON-like documents
- Very flexible

3.Wide-Column

- Data stored in columns instead of rows

4.Graph

- Data stored as nodes and edges
- Used for relationships

How data is stored?

- No rows and columns
- Data is stored in **flexible structures**
- Format can change anytime

This is why NoSQL databases are called **schema-free**.

Real-world example

Think of **social media app data**:

- User profiles
- Posts
- Likes
- Comments

Every user can have **different data fields**.
Storing this in tables is hard, but NoSQL handles it easily.

Features of NoSQL Databases

1.Schema-free

- No fixed structure
- Easy to change data format

2.Non-tabular data storage

- JSON, key-value, graph, columns

3.Highly flexible

- Data structure adjusts dynamically

4.Handles Big Data

- Works well with massive data and traffic

5.Horizontal scalability

- Add more servers instead of upgrading one

6.Mostly open-source

- Cost-effective
- Community-driven

Limitation

1.Not suitable for complex joins like RDBMS
2.Less strict data consistency (in many cases)

4. Hierarchical Databases

What is a Hierarchical Database?

As the name suggests, a **hierarchical database** stores data in a **hierarchy** (levels).

It follows a **parent–child relationship**, just like a **tree structure**.

Where is it useful?

Best suited when data naturally follows a **one-to-many** relationship.

Real-world example

In a **company**:

- One **Department**
  - Many **Employees**

Each employee reports to **only one department**.

Structure of Hierarchical Database

- Data is organised like a **tree**
- There is one **root (parent)**
- Parent can have **many children**
- A child can have **only one parent**

This is a strict rule.

How data is stored?

- Data is stored as **records**
- Each record has **fields**
- Each field stores **only one value**

To fetch data, the system:

- Starts from the **root**
- Moves **top to bottom**
- Traverses the entire tree

Real-world example

Think of **computer folders (Windows OS)**:

- C: Drive
  - Program Files
    - Application folders
      - Files

This is exactly how hierarchical databases work.

Why hierarchical databases exist?

Disk storage itself is **hierarchical**,
so this model fits well as a **physical data model**.

Advantages of Hierarchical Databases

1.Easy to understand & use

- Simple parent–child structure

2.Fast data access

- Traversing is quick for one-to-many data

3.Perfect for menus & folders

- Website drop-down menus
- File systems (Windows, Linux)

4.Safe data modification

- Adding or deleting data does not disturb entire database

5.Language support available

- Most programming languages support tree traversal

Disadvantages of Hierarchical Databases

1.Very inflexible

- A child cannot have multiple parents

2.Not good for complex relationships

- Many-to-many relationships are impossible

3.Sequential searching

- Must search from top to bottom
- Time-consuming for large trees

4.Data redundancy

- Same data may be repeated in different branches

Example of Hierarchical Database

- **IBM IMS**

5. Network Databases

What is a Network Database?

A Network Database is an extension of the Hierarchical Database model.

It removes the biggest limitation of hierarchical databases.

Key Idea

A child record can have multiple parent records

So, data is no longer limited to one-to-many.
It supports many-to-many (M:N) relationships.

Structure of Network Database

- **Data is organised in a graph structure**
- **Records are connected using links**
- **One record can be connected to many records**

Real-world example

Think of a college system:

- One Student can enroll in many Courses
- One Course can have many Students

This relationship cannot be represented properly in hierarchical databases, but network databases handle this easily.

Advantages of Network Databases

1.Handles complex relationships

- Supports many-to-many relations

2.More flexible than hierarchical databases

- No strict single-parent rule

3.Efficient data access

- Direct record-to-record links

Disadvantages of Network Databases

1.Very difficult to maintain

- Complex structure
- Hard to modify or update

2.Slow retrieval in some cases

- M:N links increase traversal time

3.Low community & web support

- Not widely used today
- Fewer tools and developers

Examples of Network Databases

- Integrated Data Store (IDS)
- IDMS (Integrated Database Management System)
- Raima Database Manager
- TurboIMAGE