**Types of Database Management Systems (DBMS)**

1. **Hierarchical DBMS**
   * Data is stored in a tree-like structure (parent-child).
   * Each child has only one parent, but a parent can have multiple children.
   * **Best for:** applications where relationships are predictable and don’t change often, like file systems or banking/telecom systems.
   * Example: IBM IMS.
2. **Network DBMS**
   * Similar to hierarchical, but allows many-to-many relationships.
   * Data is linked using pointers (records can have multiple parents and children).
   * **Best for:** complex relationships like supply chains or organizational structures.
   * Example: IDMS.
3. **Relational DBMS (RDBMS)**
   * Data is stored in tables (rows and columns).
   * Uses SQL (Structured Query Language) for querying.
   * **Best for:** applications needing structured data, transactions, and consistency (banking, HR systems, e-commerce).
   * Examples: MySQL, PostgreSQL, Oracle, SQL Server.
4. **Object-Oriented DBMS (OODBMS)**
   * Stores data as objects, similar to object-oriented programming.
   * Can handle complex data like multimedia, CAD, simulations.
   * **Best for:** applications where data is naturally object-based, like graphics, scientific research.
   * Example: db4o, ObjectDB.
5. **NoSQL DBMS** (Not Only SQL)
   * Non-relational, flexible schema.
   * Several sub-types:
     + **Document DBs** (store data as JSON-like documents) → MongoDB, CouchDB.
     + **Key-Value Stores** (data stored as key-value pairs, very fast) → Redis, DynamoDB.
     + **Column-based DBs** (optimized for large-scale analytics) → Cassandra, HBase.
     + **Graph DBs** (stores nodes and relationships) → Neo4j.
   * **Best for:** big data, high scalability, flexible structures, real-time apps.

**Relational DBMS vs NoSQL**

**Relational DBMS (RDBMS)**

* Structure: Tables with rows and columns.
* Schema: Fixed, structured.
* Query Language: SQL.
* Transactions: Strong support for **ACID** (Atomicity, Consistency, Isolation, Durability).
* Best for: banking, ERP, e-commerce — anything needing strict consistency.

**NoSQL DBMS**

* Structure: Can be documents, key-value, column, or graph.
* Schema: Flexible, can handle unstructured or semi-structured data.
* Query Language: Varies by DB type (not always SQL).
* Transactions: Many use **BASE** (Basically Available, Soft state, Eventually consistent).
* Best for: social networks, IoT, real-time analytics, large distributed systems.

**Key Difference in One Line**

* **RDBMS** = rigid but reliable (great for structured, transactional data).
* **NoSQL** = flexible and scalable (great for unstructured, fast-changing, massive data).