PART 1: Types of Databases

1.1 Relational Databases (RDBMS)

Definition: Data is stored in tables (rows and columns), and relationships are maintained using keys.

Examples: MySQL, PostgreSQL, SQLite, Microsoft SQL Server, Oracle Database

Use cases: Structured data, ACID compliance required, Banking, enterprise applications, CRM, ERP

1.2 Non-Relational Databases (NoSQL)

Broad category. Doesn't use fixed schemas. Includes:

a) Document Stores - MongoDB, CouchDB

b) Key-Value Stores - Redis, DynamoDB

c) Columnar Stores - Cassandra, HBase

d) Graph Databases - Neo4j, ArangoDB, Amazon Neptune

Use cases: Semi-structured/unstructured data, High scalability/performance, Real-time analytics, etc.

PART 2: Relational Databases Internals

2.1 How Relational Databases Work Internally

- Tables store data in row format.
- Relationships through Primary keys and Foreign keys.

2.2 Data Storage

- Stored on disk in blocks/pages.

- Metadata in system catalogs.

- Frequently accessed pages cached in buffer pool.

2.3 Database Engine

- Manages storage, indexes, queries, logging.

- MySQL: InnoDB (transactions), MyISAM (faster reads)

- PostgreSQL: Single engine with extensions.

2.4 Indexes

- Improve query performance.

- Types: B-Tree, Hash, GIN/GiST

- Stored in separate structures.

2.5 Scaling RDBMS

Vertical Scaling: Add CPU/RAM/SSD.

Horizontal Scaling: Sharding - splitting data across servers.

Tools: Vitess, Citus

2.6 Database Clusters

- Group of DB nodes for HA.

- Master-slave, multi-master, read replicas.

- Tools: Patroni (PostgreSQL), MySQL Group Replication

2.7 Data Types

- INT, BIGINT, FLOAT, VARCHAR, CHAR, TEXT, BOOLEAN, DATE, TIMESTAMP

- CHAR: Fixed length, padded

- VARCHAR: Variable length

- TEXT: Large text fields

PART 3: SQL Queries

SELECT * FROM users;

INSERT INTO users (name, email) VALUES ('Shubham', 's@example.com');

UPDATE users SET name = 'Shubh' WHERE id = 1;

DELETE FROM users WHERE id = 1;

PART 4: Joins in SQL

INNER JOIN: Only matching records

LEFT JOIN: All from left + matched from right

RIGHT JOIN: All from right + matched from left

FULL OUTER JOIN: All records from both

CROSS JOIN: Cartesian product

Example:

SELECT orders.id, customers.name

FROM orders

INNER JOIN customers ON orders.customer_id = customers.id;

PART 5: Non-Relational Databases

5.1 Purpose of NoSQL

- Handle semi/unstructured data

- Schema flexibility, scale-out

- Great for microservices, analytics

5.2 CAP Theorem

- Consistency, Availability, Partition Tolerance

- Pick any two:

  - CP: MongoDB

  - AP: Cassandra

5.3 Internals

MongoDB: BSON docs, B-tree indexes, auto-sharding.

Redis: In-memory, fast, RDB/AOF persistence.

Elasticsearch: Lucene-based, inverted index, distributed.

5.4 Scaling NoSQL

- Built-in horizontal scaling, replication.