Unit - III NoSQL Databases and Big Data Management

3.1 Introduction NoSQL in Big Data

As the volume, velocity, and variety of data continue to grow, traditional relational databases (SQL) often struggle to store and process large-scale, unstructured, and rapidly changing data. This is where **NoSQL databases** become essential.

NoSQL:

NoSQL (Not Only SQL) refers to a broad category of database systems designed to handle:

- Massive volumes of data
- Flexible or unstructured data models
- High-speed read/write operations
- Distributed and scalable architectures

NoSQL for Big Data:

Scalable: Easily handles huge data by adding more servers.

Flexible: Supports unstructured and semi-structured data (JSON, documents, etc.).

High Performance: Fast read/write for large, real-time data.

Distributed: Built for fault tolerance and large-scale distributed systems.

Types of NoSQL databases:

1. **Key–Value Stores** – Simple key/value pairs (e.g., Redis).
2. **Document Stores** – JSON-like documents (e.g., MongoDB).
3. **Column-Family Stores** – Data in columns for large-scale analytics (e.g., Cassandra).

4. **Graph Databases** – Focus on relationships and networks (e.g., Neo4j).

3.2 NoSQL Data Store : NoSQL, CAP theorem, Schema-less Models

CAP Theorem

In a distributed system, you can only guarantee **two** of the following three at the same time:

- **C – Consistency:** All nodes see the same data.
- **A – Availability:** System always responds to requests.
- **P – Partition Tolerance:** System continues working even if network failures occur.

NoSQL systems choose different trade-offs based on use cases (e.g., Cassandra: AP, MongoDB: CP).

Schema-less Models

NoSQL databases are **schema-less**, meaning:

- Data doesn't require a fixed schema.
- Each record can have different fields.
- Supports flexible and evolving data structures (e.g., JSON documents).

3.3 NoSQL Data Architecture Patterns : Key-Value Store, Document Store, Tabular Data, Object Data Store.Graph Database

NoSQL Data Architecture Patterns

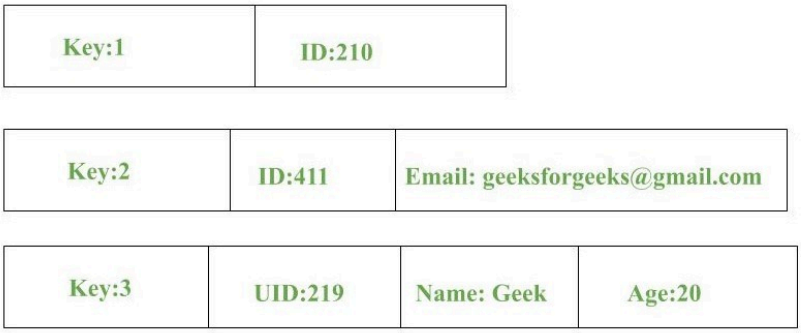
NoSQL databases are designed to handle large volumes of structured, semi-structured, and unstructured data with high scalability, distributed architecture, and flexible schemas.

Architectural patterns:

1. Key–Value Store

Stores data as a collection of **key–value pairs**.

The key is a unique identifier; the value can be any blob (string, JSON, binary).



Used mostly for:

- Caching (e.g., Redis)
- User session management
- Real-time recommendations

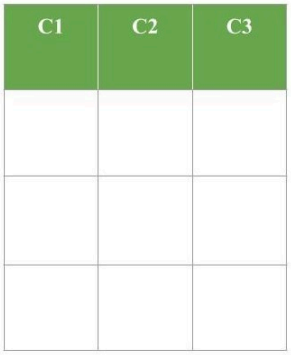
Example:

- Redis, Amazon DynamoDB (key-value mode)

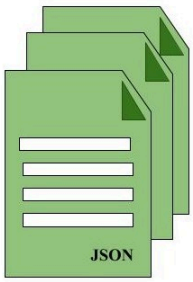
2. Document Store

Data stored as **documents**, usually JSON, BSON, or XML.

Flexible schema; each document can have different fields.



Relational Data Model



Document Store Model

Used mostly for:

- Content management systems
- User profiles, product catalogs
- Event logging

Example:

- MongoDB, CouchDB, Firebase Firestore

3. Tabular Data

Organizes data into **tables, rows, and dynamic columns**.

Unlike relational tables, each row can have different columns.

Optimized for distributed storage across clusters.

Use Cases

- IoT data ingestion
- Real-time analytics
- Time-series applications

Example:

- Apache Cassandra, HBase, Google Bigtable

4. Object Data Store

Stores data as **objects**, similar to OOP objects, containing both data and metadata.

Often used for storing large binary objects (BLOBs).

Use Cases

- Media storage (images, videos)
- Backup and archive systems

- Cloud-native applications

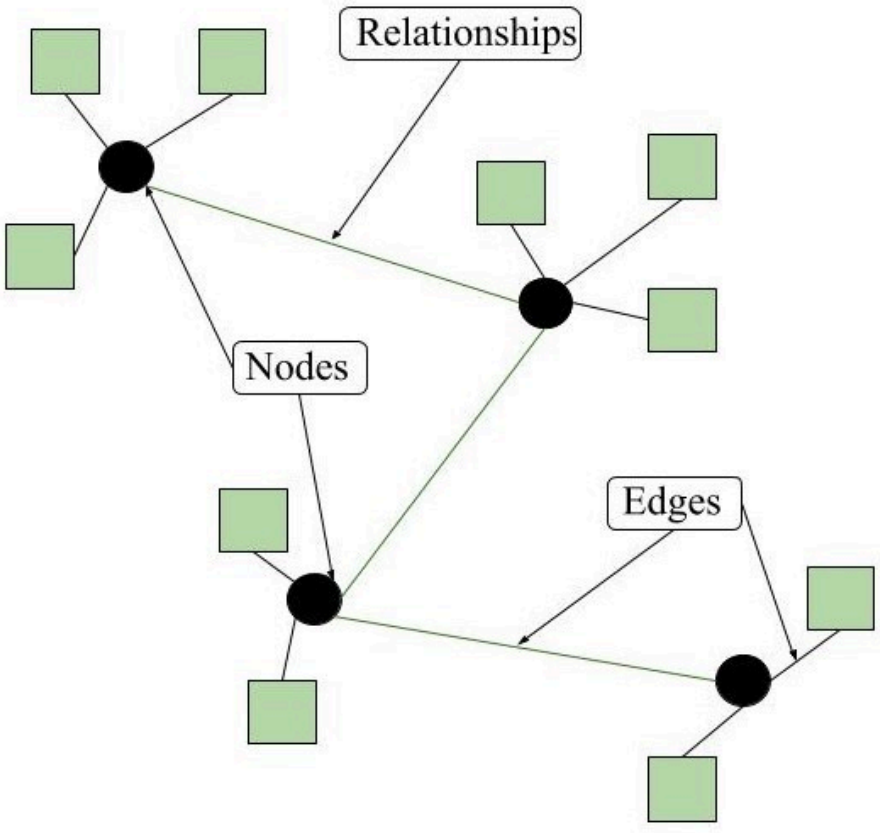
Example:

- Amazon S3, Azure Blob Storage, Google Cloud Storage

5. Graph Database

Stores data as **nodes (entities)** and **edges (relationships)**.

Optimized for traversing complex relationships.



Use Cases

- Social networks
- Fraud detection
- Recommendation engines
- Network and IT operations

Examples

- Neo4j, Amazon Neptune, ArangoDB, JanusGraph

3.4 NoSQL to Manage Big Data

NoSQL databases are designed to handle Big Data challenges: high volume, high velocity, and high variety. They offer flexible schemas, horizontal scalability, and distributed architecture.

Why NoSQL is Used for Big Data

1. **Scalability:** Supports horizontal scaling across distributed servers.
2. **Flexibility:** Handles structured, semi-structured, and unstructured data without strict schemas.
3. **High Performance:** Optimized for fast read/write operations.
4. **Distributed Architecture:** Built-in sharding and replication for fault tolerance.

How NoSQL Helps Manage Big Data

- Handles data variety (JSON, logs, media, sensor data).
- Supports real-time ingestion and processing.
- Enables distributed Big Data systems (integrates with Hadoop, Spark, Kafka).
- Provides high availability.
- Scales to petabytes across large clusters.

3.5 MongoDB Database

Stores data in JSON-like BSON documents

Flexible schema and scalable

Features

- **Document Model:** Stores data as documents
- **Schema Flexibility:** Fields can vary
- **High Performance:** Fast read/write
- **Scalability:** Sharding support
- **Replication:** Replica sets for reliability
- **Indexes:** Text, geospatial, compound
- **Aggregation:** Advanced data processing

Data Model Structure

- Database → Collection → Document → Fields

Commonly Used for

- Web/mobile apps
- User profiles
- Product catalogs
- Content management
- IoT and logs

Questions:

1. Difference between RDBMS versus Hadoop.
2. What is NoSQL? Explain why NoSQL is important in handling Big Data.
3. Explain the key characteristics of NoSQL databases.
4. Explain the CAP Theorem.
5. What are schema-less models in NoSQL databases?
6. Explain the Key–Value Store and Document Store architectural patterns with their use cases.
7. How does NoSQL help in managing Big Data?
8. Explain MongoDB as a NoSQL database. Describe its features and common use cases.