VALUE ADDED COURSE

Course Title : NoSQL Database

Course Code : 24UCSVAC1

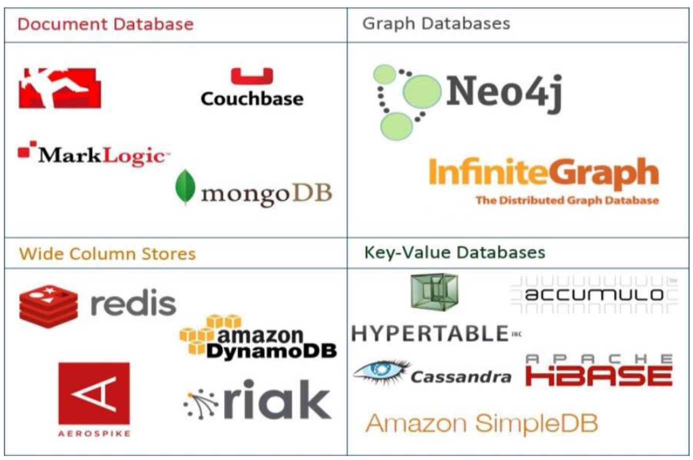
S.PRABHAVATHI
ASSISTANT PROFESSOR
DEPARTMENT OF COMPUTER SEIENCE & IT
JAMAL MOHAMED COLLEGE (A)
TRICHY – 620 020

<u>Unit-1</u>

Introduction to NoSQL – RDMBS Characteristics – ACID properties – NoSQL –Where does NoSQL comes from – Dynamo and BigTable – NoSQL and Bigdata– Why RDBMS not suitable for Bigdata – NoSQL Distinguishing Characteristics– NoSQL VS. SQL.

Introduction to NoSQL:

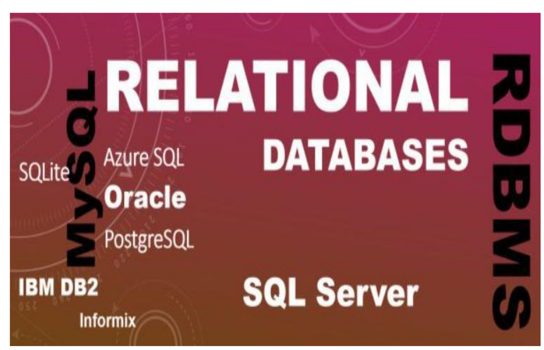
- NoSQL databases are currently a hot topic in some parts of computing, with over a hundred different NoSQL databases.
- **Definition**: NoSQL (Not Only SQL) databases are designed for distributed data stores with a need for large-scale data storage that traditional relational databases (RDBMS) can't handle efficiently.
- **Purpose**: To overcome the limitations of RDBMS by providing a more flexible and scalable data storage solution.



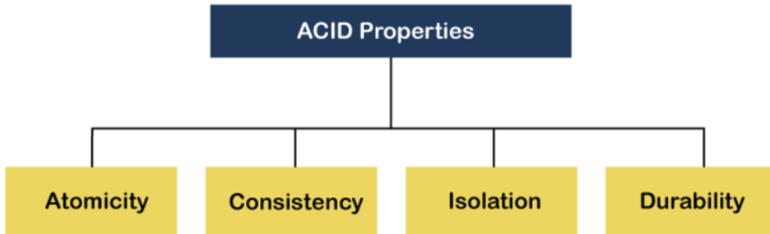
RDBMS Characteristics:

- **Structured Data**: Data is stored in tables with a predefined schema.

  - Data stored in columns and tables
  - Relationships represented by data
  - Data Manipulation Language
  - Data Definition Language
  - Transactions
  - Abstraction from physical layer
  - Applications specify what, not how
  - Physical layer can change without modifying applications
  - Create indexes to support queries
  - In Memory databases



ACID properties:



- **ACID Transactions**: Ensures reliability and consistency of database transactions.
  - **Atomicity**: Ensures that all operations within a transaction are completed successfully or none at all.
  - **Consistency**: Ensures that a transaction brings the database from one valid state to another.
  - **Isolation**: Ensures that the operations of a transaction are isolated from other transactions.
  - **Durability**: Ensures that once a transaction is committed, it remains so, even in the event of a system failure.

NoSQL:

- **Definition**: A broad class of database management systems that do not adhere to traditional RDBMS principles.

  NoSQL stands for:

  - No Relational
  - No RDBMS
  - Not Only SQL

  NoSQL is an umbrella term for all databases and data stores that don't follow the RDBMS principles
  - A class of products
  - A collection of several (related) concepts about data storage and manipulation
  - Often related to large data sets

  Key Features:

  - Schema-less design.
  - Horizontal scalability.
  - Distributed architecture.
  - High availability and fault tolerance.
  - Flexible data models.

Where Does NoSQL Come From?

- **Origins**: Emerged from the need to handle large-scale, unstructured, or semi-structured data generated by modern web applications.

- Non-relational DBMSs are not new
  - But NoSQL represents a new incarnation
  - Due to massively scalable Internet applications
  - Based on distributed and parallel computing
  - Development
  - Starts with Google
  - First research paper published in 2003
  - Continues also thanks to Lucene's developers/Apache (Hadoop) and Amazon(Dynamo)
  - Then a lot of products and interests came from Facebook, Netfix, Yahoo, eBay, Hulu, IBM, and many more

Dynamo and BigTable:

Three major papers were the seeds of the NoSQL movement

- **Amazon Dynamo**:
  - Key-value store designed for high availability and scalability.
  - Uses consistent hashing for data distribution.
  - Prioritizes availability over consistency (AP in CAP theorem).
- **Google BigTable**:
  - Column-family store designed for handling large-scale data across many servers.
  - Data is stored in a sparse, distributed, persistent multi-dimensional sorted map.
  - Influenced the design of Apache HBase and Cassandra.
- **CAP Theorem:**

  The CAP theorem says that a distributed system can deliver on **only two of three desired char acteristics: consistency, availability and partition tolerance**.

  At most two of the following three can be maximized at one time

  - Consistency
    - Each client has the same view of the data
  - Availability
    - Each client can always read and write
  - Partition tolerance
    - System works well across distributed physical networks

NoSQL and Big Data:

- **Big Data**: Refers to extremely large data sets that traditional RDBMS cannot handle effectively due to volume, velocity, and variety.
- **NoSQL**: Provides the scalability and flexibility required to store and process big data efficiently.

Challenges :

- Efficiently storing and accessing large amounts of data is difficult, even more considering fault tolerance and backups
- Manipulating large data sets involves running immensely parallel processes
- Managing continuously *evolving schema* and metadata for *semi-structured and un-structured* data is difficult

Why RDBMS is Not Suitable for Big Data:

- **Scalability Limitations**: Vertical scaling of RDBMS is limited and costly.
- **Rigid Schemas**: Predefined schemas are not flexible enough to accommodate diverse and rapidly changing data.
- **Performance**: RDBMS performance degrades with large-scale data operations.

The context is Internet
RDBMSs assume that data are

- Dense
- Largely uniform (structured data)

Data coming from Internet are

- Massive and sparse
- Semi-structured or unstructured

With massive sparse data sets, the typical storage mechanisms and access methods get stretched

NoSQL Distinguishing Characteristics:

- **Schema Flexibility**: Allows for dynamic data models.
- **Horizontal Scalability**: Scales out by adding more servers.
- **High Throughput**: Handles high read/write loads efficiently.
- **Eventual Consistency**: Prioritizes availability and partition tolerance, ensuring that the system will eventually become consistent.

NoSQL vs. SQL:

SQL DB	NoSQL DB
Examples: DB2, MySQL, Oracle, Postgress, SQL server	Examples: CouchDb MongoDB, RavenDb, Redis, Cassandra, Hbase, Neo4j,BigTable
These are called RDBMS.	These are called not only SQL database.
Based on ACID properties i.e. Atomicity, Consistency, Isolation and Durability	Based on CAP properties i.e. (Consistency, Availability and Partition tolerance)
These are table based database i.e. the data are stored in a table with rows and columns.	These databases are document based, key-value pairs or graph based etc.
These are standard schema based (predefined schema)	These are not standard schema based(dynamic schema)
These are scaled vertically. Load can be managed by increasing CPU, RAM etc in the same server.	These are scaled horizontally. A few servers can be added to manage large traffic.
Not preferred for large/big data sets.	Preferred for large/big data sets.
Preferred for complex query execution	Not preferred for complex query execution

Unit -2

NoSQL Datatypes - Sorted ordered Column Store – Document Databases – KeyValue Store – Graph Databases – Dealing with Bigdata and Scalability – NoSQL No ACID.

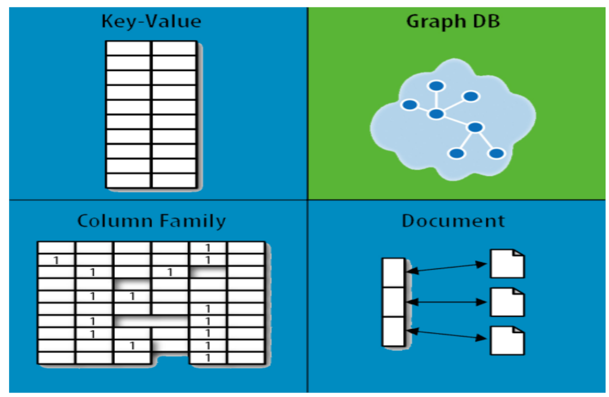
NoSQL Datatypes:

NoSQL databases use various data types tailored to the specific requirements of different data models. Key types include:

- **Primitive Types**: Strings, integers, floats, booleans.
- **Complex Types**: Arrays, lists, sets, maps, JSON/BSON documents.
- **Specialized Types**: Geospatial data, binary data, timestamps.

These types enable flexibility and efficient data storage across different NoSQL models.

Sorted Ordered Column Store:

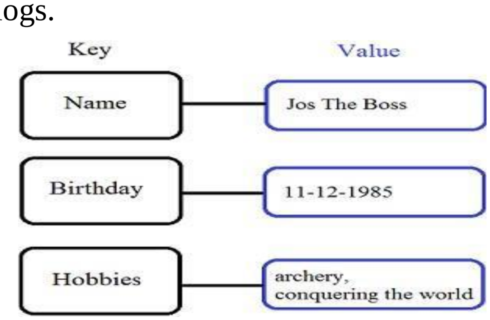


- **Definition**: Stores data in columns rather than rows, allowing for efficient querying and aggregation.
- **Characteristics**:
  - Data is stored in columns grouped into families.
  - Columns are sorted and can be indexed individually.
  - Supports sparse data efficiently.
- **Examples**: Apache Cassandra, HBase.
- **Use Cases**: Real-time analytics, time-series data, large-scale data warehousing.

Document Databases:

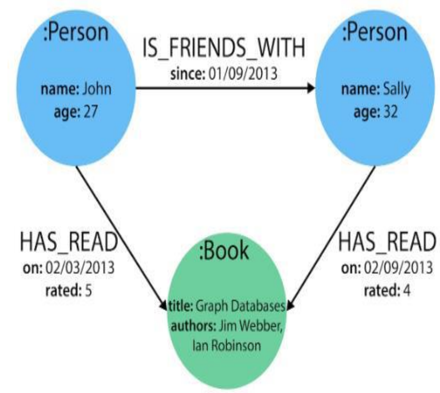
- **Definition**: Store data as documents, typically in JSON or BSON format.
- **Characteristics**:
  - Schema-less design allows for flexible and dynamic data structures.
  - Each document is a self-contained unit of data with its own schema.
  - Supports nested structures and complex queries.
- **Examples**: MongoDB, CouchDB.
- **Use Cases**: Content management systems, user profiles, catalogs.

Key-Value Store:



- **Definition**: Stores data as a collection of key-value pairs.
- **Characteristics**:
  - Simple data model with fast lookups and retrievals.
  - Suitable for caching and session storage.
  - Data can be strings, lists, sets, hashes, etc.
- **Examples**: Redis, Amazon DynamoDB.
- **Use Cases**: Caching, real-time data processing, session management.

Graph Databases:



- **Definition**: Store data in nodes, edges, and properties, representing entities and relationships.
- **Characteristics**:
  - Optimized for traversing and querying relationships.
  - Use graph structures to model complex interconnections.
  - Provides powerful querying capabilities for relationship-based data.
- **Examples**: Neo4j, OrientDB.
- **Use Cases**: Social networks, fraud detection, recommendation engines.

Dealing with Big Data and Scalability:

- **Challenges**:
  - Volume: Handling large amounts of data.
  - Velocity: Managing the speed of data generation and processing.
  - Variety: Processing diverse data types and structures.
- **NoSQL Solutions**:
  - **Horizontal Scalability**: Adding more nodes to handle increased load.
  - **Distributed Computing**: Distributing data across multiple machines for parallel processing.
  - **Partitioning and Sharding**: Dividing data into manageable segments to optimize performance and scalability.
  - **Replication**: Copying data across multiple nodes to ensure availability and fault tolerance.

NoSQL No ACID

- **ACID Properties**: Atomicity, Consistency, Isolation, Durability.
- **NoSQL Focus**: BASE Properties (Basically Available, Soft state, Eventually consistent).
  - **Basically Available**: Ensures system availability.
  - **Soft State**: System state may change over time.
  - **Eventually Consistent**: Data will eventually reach consistency.
- **Trade-offs**:
  - **ACID (RDBMS)**: Prioritizes consistency and reliability.
  - **BASE (NoSQL)**: Prioritizes availability and scalability, accepting eventual consistency for performance gains.