SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING-IOT Including CS&BCT

COURSE NAME : 19SB602 FULL STACK DEVELOPMENT FOR NEXT GENERATION IOT

III YEAR / VI SEMESTER

Unit V - NG-IoT-Next Generation Internet of Things

Topic :NO SQL

NO SQL

- The term NoSQL, short for “not only SQL,” refers to **non-relational databases** that store data in a non-tabular format, rather than in rule-based, relational tables like relational databases do. NoSQL databases use a flexible schema model that supports a wide variety of unstructured data such as documents, key-value, wide columns, and graphs.

- Organizations choose NoSQL databases for their flexibility, high performance, horizontal scalability, and ease of development.

5 TYPES OF NOSQL DATABASES

- **Document databases:** MongoDB, CouchDB
- **Key-value databases:** Redis, Amazon DynamoDB
- **Column-oriented databases:** Apache Cassandra, HBase
- **Graph databases:** Neo4j, Amazon Neptune
- **In-memory databases:** Redis, Memcached

Document databases

- Document databases, also called document-oriented databases or a document store, are used to store and query semi-structured data.
- Data is stored in a JSON-like document similar to the data objects that developers use in application code, making it easier to create and update applications without referencing a primary schema.
- Document databases are most commonly used for blogging platforms, ecommerce, real-time analytics, and content management systems.

Key-value databases

- Key-value databases, also referred to as key-value stores, are the simplest type of NoSQL databases. Data is stored in a “key-value” structure, where a unique key is paired with a value such as a string, number, boolean, or complex objects.
- You can use the key to store or retrieve its associated value. Key-value stores are most commonly used for user preferences, shopping carts, and user profiles in web applications.

Column-oriented databases

- Column-oriented databases, or wide-column stores, store and read data in rows and are organized as a set of columns.
- While similar to the tabular format of relational databases, column names and formatting in wide-column stores can vary from row to row in a single table. They are optimal for analytics use cases, where you may need to query across specific columns in a database and aggregate the value of a given column quickly. Wide-column stores are most commonly used for catalogs, fraud detection, and recommendation engines.

Graph databases

- Graph databases organize data as nodes in a graph, focusing on the relationships between data elements. The connections between nodes (edges) are stored as first-class elements, enabling richer representations of data relationships while offering more simplified storage and navigation.
- Graph databases are most commonly used in systems that map relationships, including social media platforms, reservation systems, fraud detection systems, and logistics applications.

Graph databases

- Graph databases organize data as nodes in a graph, focusing on the relationships between data elements. The connections between nodes (edges) are stored as first-class elements, enabling richer representations of data relationships while offering more simplified storage and navigation.
- Graph databases are most commonly used in systems that map relationships, including social media platforms, reservation systems, fraud detection systems, and logistics applications.

In-memory databases

- In-memory databases store data in memory in order to provide ultra-low latency for real-time applications. Redis and Valkey are examples of in-memory NoSQL databases.
- In-memory databases are most commonly used for caching, messaging, streaming, and real-time analytics.

How does NoSQL work?

NoSQL features are unique to the database you choose. However, they typically share several similar high-level qualities:

•They follow flexible schemas that don’t require you to determine or declare a fixed schema for your data, - semi-structured and unstructured data

•They scale horizontally, using range or hash distributions

•They’re optimized for specific data models and workload patterns, such as key-value, wide-column, or in-memory

- They often exhibit consistency at some later point (for example, eventual consistency model) rather than following the stricter ACID (atomicity, consistency, isolation, durability) properties of relational and SQL databases

- They usually don't support cross-shard transactions or flexible isolation modes

- These features make non-relational databases ideal for applications that require large scale, reliability, high availability, and frequent data changes.

SQL versus NoSQL: Use cases and examples

Feature	SQL (Relational DB)	NoSQL (Non-Relational DB)
Schema	Fixed and predefined	Flexible or schema-less
Scalability	Vertical (scale-up)	Horizontal (scale-out)
Data Model	Structured tables with rows/columns	Key-Value, Document, Column, Graph
Query Language	Standardized (SQL)	Varies by database, no universal standard
Transactions	Strong ACID compliance	Often eventual consistency, limited ACID

SQL versus NoSQL: Use cases and examples

SQL use cases:	NoSQL uses include:
•Financial transactions •Healthcare data analysis •Customer and transaction information	•Mobile, web, and IoT applications •Real-time web applications •Personalization, recommendations, and real-time customer experiences •Inventory and catalog management •Fraud detection and identity authentication •Ad tech

Advantages of NOSQL

- Flexible data model and schema
- Agile development
- Scalability
- Massive data storage
- High availability
- Faster queries

Disadvantages of NOSQL

- Relatively new and less mature than relational databases
- Limited developer expertise and community support
- Fewer tools and third-party integrations
- No standard query language (unlike SQL)
- Weaker support for ACID transactions in many databases
- Not suitable for complex queries and joins
- Managing indexes across distributed nodes can be inefficient
- Eventual consistency may not be ideal for all applications

Any Query????

Thank you……