SQL Databases:

SQL databases, also known as **Relational Database Management Systems (RDBMS)**, use structured tables to store data. They rely on a **predefined schema** that determines the organization of data within tables, making them suitable for applications that require a fixed, consistent structure.

- **Structured Data**: Data is organized in tables with rows and columns, making it easy to relate different types of information.
- **ACID Compliance**: SQL databases follow the ACID properties (Atomicity, Consistency, Isolation, Durability) to ensure reliable transactions and data integrity.
- **Examples**: Popular SQL databases include **MySQL**, **PostgreSQL**, **Oracle**, and **MS SQL Server**.

NoSQL Databases:

NoSQL (Not only Structured Query Language) databases, are designed to handle **unstructured or semi-structured data**. Unlike SQL databases, NoSQL offers **dynamic schemas** that allow for more flexible data storage, making them ideal for handling massive volumes of data from various sources. The reason it is called NoSQL is to emphasize that these databases can handle non-tabular, non-relational data models as well as support SQL-like query languages.

- **Flexible Schema**: NoSQL databases allow the storage of data without a predefined structure, making them more adaptable to changing data requirements.
- **CAP Theorem**: NoSQL databases are designed based on the **CAP theorem** (Consistency, Availability, Partition Tolerance), which prioritizes availability and partition tolerance over strict consistency.
- **Examples**: Well-known NoSQL databases include **MongoDB**, **Cassandra**, **CouchDB**, and **HBase**.

Difference between SQL and NoSQL:

1. **Database storage model:**
   SQL databases store data in tables containing rows and columns whereas NoSQL systems store data using various methods depending on the type

of unstructured data being ingested (e.g., JSON documents, key-value pairing, family grouping, graph nodes/edges).

2. **Data type:**
   NoSQL databases store, and retrieve unstructured data whereas SQL databases store and retrieve structured data.

3. **Schemas:**
   SQL databases rely on a strict, predefined data schema. However, NoSQL databases use flexible schemas which enable them to store data in its various native formats.

4. **Scalability:**
   SQL databases supports vertical scalability and NoSQL databases supports horizontal scalability.

5. **Data Integrity:**
   SQL databases are ACID-compliant (strong consistency) whereas NoSQL databases are BASE-compliant (more available, less consistent).

   Atomicity, Consistency, Isolation, and Durability (ACID) compliance: ACID compliance refers to a set of properties that guarantee the reliability, consistency, and data integrity of database transactions.

   Basically Available, Soft State, Eventually Consistent(BASE) compliance : BASE compliance refers to a set of principles used in distributed systems (especially NoSQL databases) that prioritize availability and partition tolerance over strict consistency.

6. **Query Language:**
   SQL databases uses SQL (Structured Query Language) to communicate with DB whereas NoSQL databases uses various query languages according to the type of database used.(e.g., MongoDB uses its own query language)

7. **Performance:**
   SQL databases are efficient for complex queries and transactions whereas NoSQL databases are better for large-scale data and fast read/write operations.

8. **Use Cases:**
   SQL databases are best for transactional systems (banking, ERP, etc.) whereas NoSQL databases are ideal for big data, real-time web apps, and data lakes

9. **Examples:**
   SQL databases: MySQL, PostgreSQL, Oracle, MS SQL Server.
   NoSQL databases: MongoDB, Cassandra, CouchDB, Neo4j.

<u>MONGODB FEATURES:</u>

MongoDB is a flexible, document-oriented database platform that is designed to be the cloud database of choice for enterprise applications. MongoDB provides a number of features that make it a great choice for a wide variety of applications.

Document Model:

MongoDB's document model provides a highly flexible and intuitive way to store and retrieve data. It uses BSON (Binary JSON) format to store documents, which supports rich data types and faster performance. The schema-less nature allows documents within a collection to have different fields and structures, enabling rapid development and schema evolution without downtime. Validation rules can be added later to enforce structure if needed.

Sharding:

Sharding in MongoDB enables horizontal scalability by distributing large datasets across multiple shards or servers. This allows queries to run more efficiently by targeting specific shards based on a shard key. The mongos router handles query routing, ensuring balanced loads across shards and seamless scaling for applications handling massive volumes of data or high user traffic.

Replication:

Replication increases data availability and fault tolerance in MongoDB. It involves maintaining a set of servers called a replica set where one node acts as the primary (handling writes), and others act as secondaries (replicating data). In case of failure, a secondary can automatically take over as the new primary, ensuring high availability without manual intervention.

Authentication:

MongoDB supports robust authentication mechanisms to secure database access. The default is SCRAM (Salted Challenge Response Authentication Mechanism), which requires a username, password, and authentication database. It also supports other mechanisms such as LDAP, x.509 certificates, and Kerberos for enterprise-level security and integration.

Database Triggers:

Triggers in MongoDB Atlas allow you to execute custom logic in response to database events such as inserts, updates, or deletions. They are useful for real-time workflows, auditing, alerting, and data validation. Triggers can also be scheduled, and they are easy to manage through the Atlas UI, enabling seamless event-driven programming.

Time Series Data:

MongoDB supports native time series collections optimized for storing and querying time-based data like sensor readings or logs. These collections offer efficient storage, automatic data expiration, and granularity controls. Developers can manage vast sequences of time-stamped data with high performance and minimal effort.

Ad-Hoc Queries:

MongoDB excels at handling ad-hoc queries, which are dynamic and vary based on user needs. Its flexible schema, rich query language (MQL), and powerful indexing allow developers to retrieve data using filters, ranges, regular expressions, and geospatial queries in real time—making it ideal for applications requiring real-time analytics.

Indexing:

Indexing in MongoDB is crucial for performance optimization. It supports single field, compound, geospatial, text, and hashed indexes, among others. Indexes significantly speed up query performance by allowing the database to avoid full collection scans. Developers can create and drop indexes on demand, even on nested fields or array elements.

Load Balancing:

MongoDB inherently supports load balancing through its architecture. By using sharding and replication, MongoDB distributes read and write operations across multiple nodes, ensuring efficient resource utilization and high throughput. It avoids the need for an external load balancer while maintaining consistent performance across concurrent user requests.