Introduction to NoSQL

NoSQL (Not Only SQL) databases are designed to handle large volumes of unstructured and semi-structured data. Unlike traditional relational databases that rely on fixed schemas and tables, NoSQL offers flexible data models and supports horizontal scaling. This makes them well-suited for modern applications that require high performance, scalability, and the ability to manage diverse data types efficiently.

Key Features of NoSQL Databases

- **Dynamic schema:** Allow flexible shaping of data to meet new requirements without the need to migrate or change schemas.
- **Horizontal scalability:** They scale horizontally for adding more nodes into the existing ones and acquire enough storage for even bigger datasets and much higher traffic by distributing the load on multiple servers.
- **Document-based:** Data are presented in flexible, semi-structured formats like JSON/BSON (e.g., MongoDB).
- **Key-value-based:** They possess a simple but fast access pattern (e.g., Redis) by storing data as pairs of keys and values.
- **Column-based:** Data are organized into columns instead of rows (e.g., CASSANDRA).
- **Distributed and high availability:** They are designed to be highly available and to automatically handle node failures and data replication across multiple nodes in a database cluster.
- **Flexibility:** Allow developers to store and retrieve data in a flexible and dynamic manner, with support for multiple data types and changing data structures.
- **Performance:** Perfect for big data and real-time analytics and high volume applications.

Challenges of NoSQL Databases

- **Lack of standardization:** NoSQL systems can be vastly different from one another, making it even harder to choose the right one for a specific use case.
- **Lack of ACID compliance:** NoSQL databases may not provide consistency, which is a disadvantage for applications that need strict data integrity.
- **Narrow focus:** Great for storage but lack functionalities as transaction management, in which relational databases are great.
- **Absence of Complex Query Support:** They are not designed to handle complex queries, which means that they are not a good fit for applications that require complex data analysis or reporting.
- **Lack of maturity:** Being relatively new, NoSQL may not have the reliability, security and feature set of traditional relational databases.
- **Management complexity:** For large datasets, maintaining a NoSQL database could be quite more complicated than managing a relational database.
- **Limited GUI Tools**: While some NoSQL databases, like MongoDB offer GUI tools like MongoDB Compass, not all NoSQL databases provide flexible or user-friendly GUI tools.

SQL vs. NoSQL:

Feature	SQL (Relational DB)	NoSQL (Non-Relational DB)
Data Model	Structured, Tabular	Flexible (Documents, Key-Value, Graphs)

Feature	SQL (Relational DB)	NoSQL (Non-Relational DB)
Scalability	Vertical Scaling	Horizontal Scaling
Schema	Predefined	Dynamic & Schema-less
ACID Support	Strong	Limited or Eventual Consistency
Best For	Transactional applications	Big data, real-time analytics
Examples	MySQL, PostgreSQL, Oracle	MongoDB, Cassandra, Redis

Popular NoSQL Databases & Their Use Cases

NoSQL Database	Type	Use Cases
MongoDB	Document-based	Content management, product catalogs
Redis	Key-Value Store	Caching, real-time analytics, session storage
Cassandra	Column-Family Store	Big data, high availability systems
Neo4j	Graph Database	Fraud detection, social networks

Use of NoSQL

- **Big Data Applications:** Efficiently stores and processes massive amounts of unstructured and semi-structured data.
- **Real-Time Analytics:** Supports fast queries and analysis for use cases like recommendation engines or fraud detection.
- **Scalable Web Applications:** Handles high traffic and large user bases by scaling horizontally across servers.
- **Flexible Data Storage:** Manages diverse data formats (JSON, key-value, documents, graphs) without rigid schemas.

Aggregate Data Model in NoSQL

We know, NoSQL are databases that store data in another format other than relational databases. NoSQL deals in nearly every industry nowadays. For the people who interact with data in databases, the Aggregate Data model will help in that interaction.

Features of NoSQL Databases:

- **Schema Agnostic:** NoSQL Databases do not require any specific schema or s storage structure than traditional RDBMS.
- **Scalability:** NoSQL databases scale horizontally as data grows rapidly certain commodity hardware could be added and scalability features could be preserved for NoSQL.
- **Performance:** To increase the performance of the NoSQL system one can add a different commodity server than reliable and fast access of database transfer with minimum overhead.
- **High Availability:** In traditional RDBMS it relies on primary and secondary nodes for fetching the data, Some NoSQL databases use master place architecture.
- **Global Availability:** As data is replicated among multiple servers and clouds the data is accessible to anyone, this minimizes the latency period.

Aggregate Data Models:

The term aggregate means a collection of objects that we use to treat as a unit. An aggregate is a collection of data that we interact with as a unit. These units of data or aggregates form the boundaries for ACID operation.

Example of Aggregate Data Model:

Here in the diagram have two Aggregate:

- Customer and Orders link between them represent an aggregate.
- The diamond shows how data fit into the aggregate structure.
- Customer contains a list of billing address
- Payment also contains the billing address
- The address appears three times and it is copied each time
- The domain is fit where we don't want to change shipping and billing address.

Consequences of Aggregate Orientation:

- Aggregation is not a logical data property It is all about how the data is being used by applications.
- An aggregate structure may be an obstacle for others but help with some data interactions.
- It has an important consequence for transactions.
- NoSQL databases don’t support ACID transactions thus sacrificing consistency.
- aggregate-oriented databases support the atomic manipulation of a single aggregate at a time.

Advantage:

- It can be used as a primary data source for online applications.
- Easy Replication.
- No single point Failure.
- It provides fast performance and horizontal Scalability.

- It can handle Structured semi-structured and unstructured data with equal effort.

Disadvantage:

- No standard rules.
- Limited query capabilities.
- Doesn't work well with relational data.
- Not so popular in the enterprise.
- When the value of data increases it is difficult to maintain unique values.