NoSQL

Architecture patterns of NoSql

The data is stored in NoSQL in any of the following four data architecture patterns.

1. Key-Value Store Database
2. Column Store Database
3. Document Database
4. Graph Database

1. Key-Value Store Database

- This model is one of the most basic models of NoSQL databases.
- The data is stored in form of Key-Value Pairs.
- The key is usually a sequence of strings, integers or characters but can also be a more advanced data type.
- The value is typically linked or co-related to the key. The key-value pair storage databases generally store data as a hash table where each key is unique. The value can be of any type (JSON, BLOB(Binary Large Object), strings, etc).
- This type of pattern is usually used in shopping websites or e-commerce applications.

Advantages:

- Can handle large amounts of data and heavy load,
- Easy retrieval of data by keys.

Disadvantages:

- Complex queries may attempt to involve multiple key-value pairs which may delay performance.
- Data can be involving many-to-many relationships which may collide.

Examples:

- DynamoDB
- Berkeley DB

2. Column Store Database

- Rather than storing data in relational tuples, the data is stored in individual cells which are further grouped into columns.
- Column-oriented databases work only on columns. They store large amounts of data into columns together.
- . Format and titles of the columns can diverge from one row to other.
- Every column is treated separately. But still, each individual column may contain multiple other columns like traditional databases.
- columns are mode of storage in this type.

Advantages:

- Data is readily available
- Queries like SUM, AVERAGE, COUNT can be easily performed on columns.

Examples:

- HBase
- Bigtable by Google
- Cassandra

3. Document Database

- The document database fetches and accumulates data in form of key-value pairs but here, the values are called as Documents.
- Document can be stated as a complex data structure. Document here can be a form of text, arrays, strings, JSON, XML or any such format.
- It is very effective as most of the data created is usually in form of JSONs and is unstructured.

Advantages:

This type of format is very useful and apt for semi-structured data.
Storage retrieval and managing of documents is easy.

Limitations:

Handling multiple documents is challenging
Aggregation operations may not work accurately.

Examples:

MongoDB
CouchDB

4. Graph Databases

- This architecture pattern deals with the storage and management of data in graphs.
- Graphs are basically structures that depict connections between two or more objects in some data.
- The objects or entities are called as nodes and are joined together by relationships called Edges. Each edge has a unique identifier. Each node serves as a point of contact for the graph.
- This pattern is very commonly used in social networks where there are a large number of entities and each entity has one or many characteristics which are connected by edges.
- The relational database pattern has tables that are loosely connected, whereas graphs are often very strong and rigid in nature.

Advantages:

Fastest traversal because of connections.
Spatial data can be easily handled.

Limitations:

Wrong connections may lead to infinite loops.

Examples:

Neo4J
FlockDB(Used by Twitter)

CAP theorem for NoSQL

- The CAP theorem, originally introduced as the CAP principle, can be used to explain some of the competing requirements in a distributed system with replication. It is a tool used to make system designers aware of the trade-offs while designing networked shared-data systems.
- The three letters in CAP refer to three desirable properties of distributed systems with replicated data. **CAP Theorem: satisfying all three at the same time is impossible**

1.Consistency (among replicated copies).

- All replicas contain the same version of data
- Client always has the same view of the data (no matter what node)

2.Availability (of the system for read and write operations)

- System remains operational on failing nodes
- All clients can always read and write

3.Partition tolerance (in the face of the nodes in the system being partitioned by a network fault)

- multiple entry points
- System remains operational on system split (communication malfunction)
- System works well across physical network partitions

Advantages of NoSQL

(i) Flexible Data Model:

NoSQL databases are highly flexible as they can store and combine any type of data, both structured and unstructured, unlike relational databases that can store data in a structured way only.

(ii) Evolving Data Model :

NoSQL databases allow you to dynamically update the schema to evolve with changing requirements while ensuring that it would cause no interruption or downtime to your application.

(iii) Elastic Scalability:
NoSQL databases can scale to accommodate any type of data growth while maintaining low cost.

(iv) High Performance:
NoSQL databases are built for great performance, measured in terms of both throughput (it is a measure of overall performance) and latency (it is the delay between request and actual response).

(v) Open-source:

NoSQL databases don’t require expensive licensing fees and can run on inexpensive hardware, rendering their deployment cost-effective.