NoSQL Data Architecture Patterns

Architecture Pattern is a logical way of categorizing data that will be stored on the Database. NoSQL is a type of database which helps to perform operations on big data and store it in a valid format. It is widely used because of its flexibility and a wide variety of services.
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

These are explained as following below.
1. Key-Value Store Database:
This model is one of the most basic models of NoSQL databases. As the name suggests, 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:

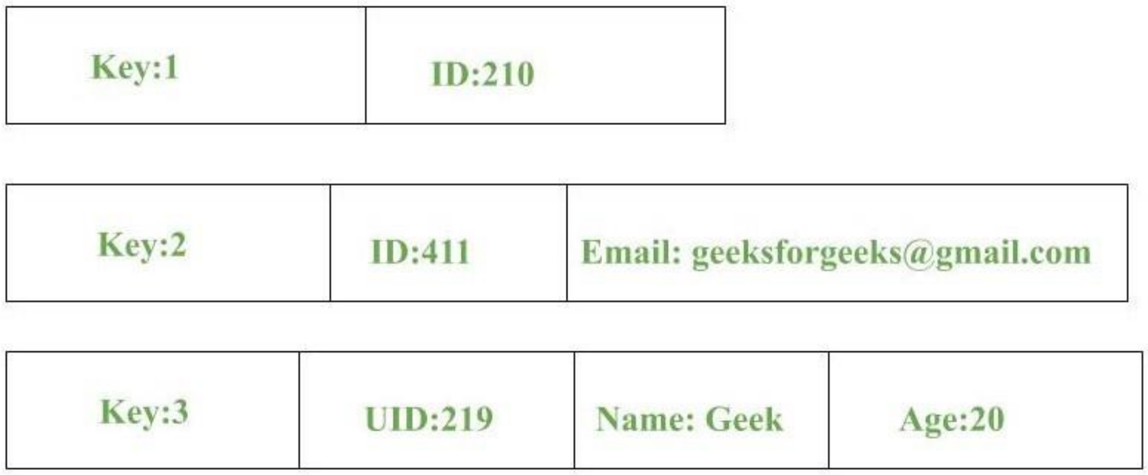
- Can handle large amounts of data and heavy load,
- Easy retrieval of data by keys.

Limitations:

- 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.

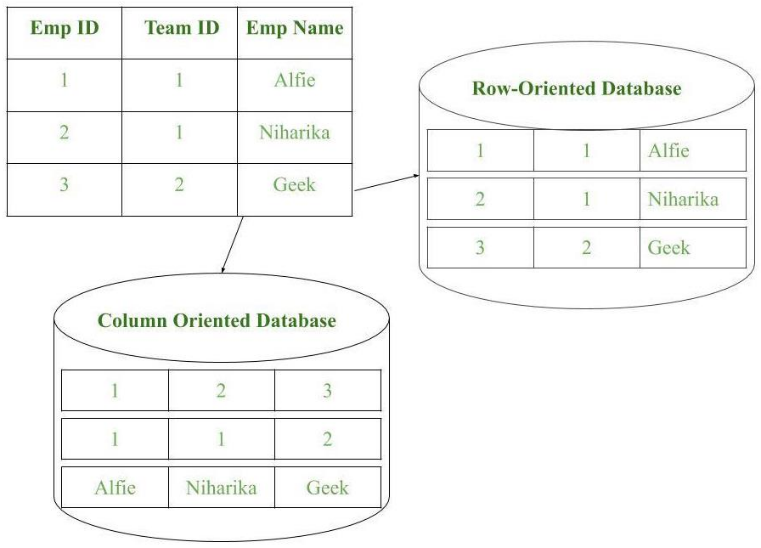
Basically, columns are mode of storage in this type.

<u>Advantages:</u>

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

<u>Examples:</u>

- 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. The use of nested documents is also very common. It is very effective as most of the data created is usually in form of JSONs and is unstructured.

<u>Advantages:</u>

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

<u>Limitations:</u>

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

<u>Examples:</u>

- MongoDB
- CouchDB

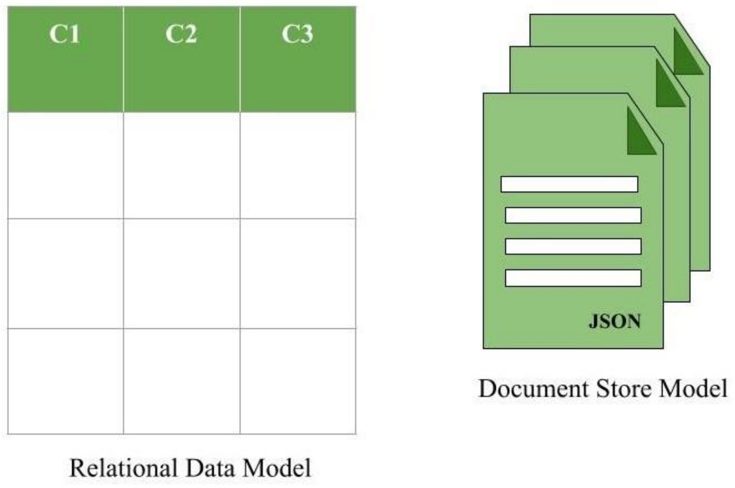


Figure – Document Store Model in form of JSON documents

4. Graph Databases:

Clearly, 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.

<u>Advantages:</u>

- 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)

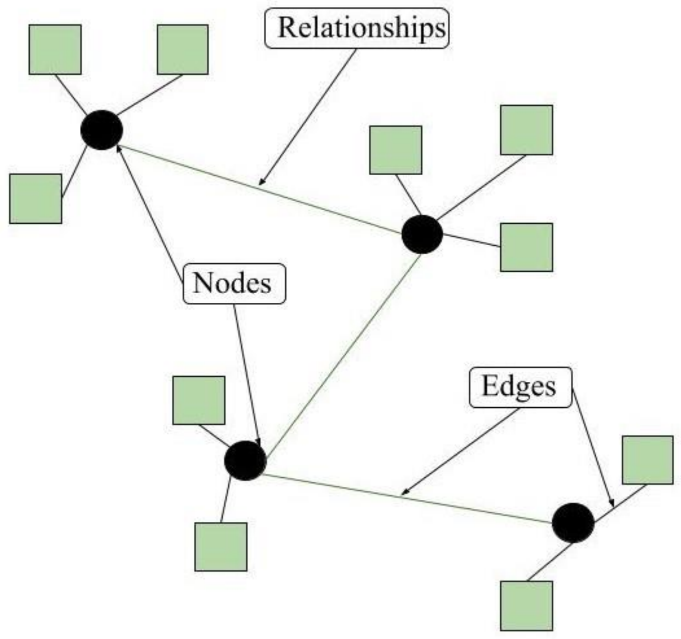


Figure – Graph model format of NoSQL Databases

NoSQL Data Architecture Patterns

An architecture pattern is a way of organizing data in a logical and structured manner. So, it can be stored and accessed efficiently. NoSQL databases are different from traditional relational databases. Because Tables are not used to store data in it. It uses document-oriented, key-value or graph formats. It makes them more flexible.

NoSQL databases can handle a wide variety of data types and sizes. It is suitable for big data applications. It can also provide faster performance than traditional relational databases. It is therefore ideal for applications that require real-time data processing.

Architecture Patterns of NoSQL

There are given architecture patterns used in NoSQL databases −

Key-Value Store

In this pattern, data is stored as a set of key-value pairs. This architecture is simple and efficient, allowing for fast read and write operations.

Advantages

- Simple and efficient architecture, allowing for fast read and write operations.
- Highly scalable and fault-tolerant.

Disadvantages

- Limited query capability.
- Not suitable for complex data structures.

Examples

- **Redis** a popular key-value store with in-memory storage and support for various data types.
- **Riak** a distributed key-value store with strong consistency and fault-tolerance.

Document Store

This pattern stores data as documents, typically in JSON or XML format. Each document contains all the necessary data, and the data can be indexed for easy retrieval.

Advantages

- Flexible schema allows for easy changes to data structure.
- High performance for read-heavy workloads.
- Good support for indexing and querying.

Disadvantages

- Lack of support for joins and transactions.
- Poor performance for write-heavy workloads.

Examples

- **MongoDB** a popular document store with support for dynamic schemas and horizontal scaling.
- **Couchbase** a distributed document store with in-memory caching and high availability.

Column-Family Store

This pattern stores data in column families, where each column family represents a set of related data. Within each column family, data is stored as a collection of key-value pairs.

Advantages

- Highly scalable architecture, ideal for big data applications.
- Efficient storage of large, sparse data sets.
- Good support for indexing and querying.

Disadvantages

- Limited support for ad-hoc queries and transactions.
- Complexity of data modeling.

Examples

- **Apache Cassandra** a distributed column-family store with high write throughput and fault-tolerance.
- **HBase** a column-family store that is part of the Hadoop ecosystem, with support for complex data structures and high scalability.

Graph Database

This pattern stores data as nodes and edges, allowing for the creation of complex relationships between data points. Graph databases are ideal for applications that require rich, interconnected data, such as social networks or recommendation engines.

Advantages

- Flexible data model that supports complex relationships.
- High performance for traversing large graphs.
- Good support for querying and indexing.

Disadvantages

- Limited support for ad-hoc queries and transactions.
- Poor performance for write-heavy workloads.

Examples

- **Neo4j** a popular graph database with high performance and good support for querying and indexing.
- **OrientDB** a multi-model graph database that supports graph, document, and key-value data models.

Object Database

This pattern stores data as objects, similar to how object-oriented programming languages like Java and Python work. Object databases are useful for applications that require complex data structures and relationships between objects.

Advantages

- Seamless integration with object-oriented programming languages.
- High performance for object-oriented workloads.
- Good support for complex data structures.

Disadvantages

- Limited support for ad-hoc queries and transactions.
- Limited adoption and community support.

Examples

- **db4o** an open-source object database with support for Java and .NET programming languages.
- **Perst** a lightweight object-oriented database with support for Java and C# programming languages.

Summary

We have discussed the architecture patterns used in NoSQL databases and their advantages and disadvantages. NoSQL databases use document-oriented, key-value, graph, or column-family formats to store data.

- Key-value store pattern is simple and efficient, suitable for fast read and write operations, and highly scalable and fault-tolerant.
- Document store pattern is flexible and suitable for indexing and querying.
- Column-family store pattern is ideal for big data applications and efficient storage of large, sparse data sets.
- Graph database pattern is suitable for complex, interconnected data sets.
- Object database pattern is useful for complex data structures and relationships between objects.

We have also provided examples of popular NoSQL databases for each pattern.