Non-Relational Databases and Their Types

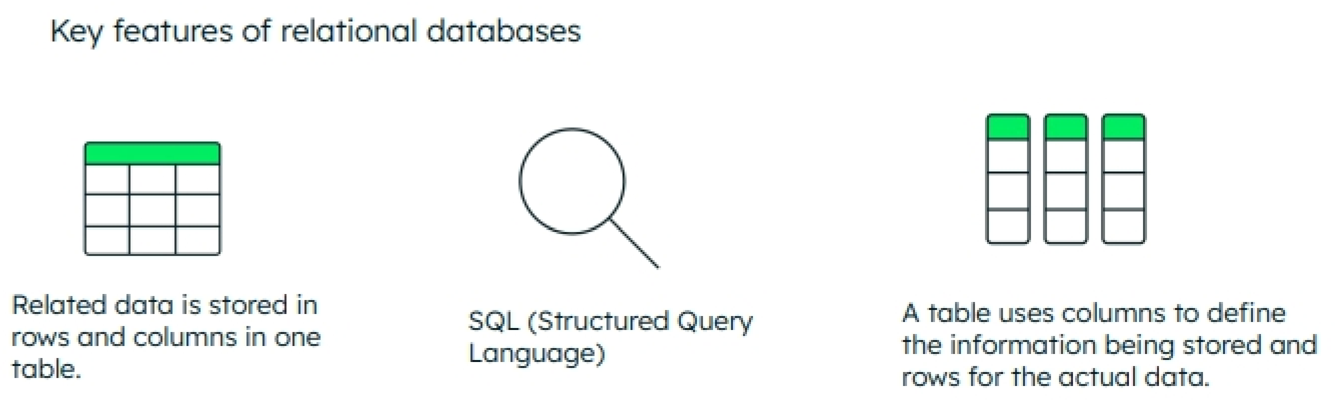
In the area of **database management**, the data is arranged in two ways which are **Relational Databases (SQL)** and **Non-Relational Databases (NoSQL)**. While relational databases organize data into **structured tables**, non-relational databases use various flexible data models like **key-value pairs**, **documents**, **graphs**, and **wide–column stores**.

Here, we will learn about non-relational database meaning and check non-relational database examples. But to understand non-relational databases, or "**NoSQL**" databases, we first need to look at relational databases.

Relational Database (SQL)

- A **relational database** stores data in a table composed of rows and columns. The **table** represents an object or entity, such as users, customers, orders, etc.
- The column represents the type of data that can be stored in the respective column.
- Relational Databases allow users to establish a connection between tables using **keys** for flexible data flow and querying.
- **SQL** was specifically designed to work with **tabular data**. These are often categorized as structured data.
- **This is because there can only be a single schema or structure for the data within a relational database.**
- SQL is a **declarative language**, which means that you describe in **SQL syntax** the desired result you wish from the query.

Key Features Of Relational Database


Key features of relational databases
Related data is stored in rows and columns in one table.
SQL (Structured Query Language)
A table uses columns to define the information being stored and rows for the actual data.

- Relational data models are similar to an **Excel** spreadsheet, with related data stored in rows and columns in one table.
- SQL (Structured Query Language) is the most common way of interacting with relational database systems. Developers can use SQL queries to execute **CRUD** (Create, Read, Update, Delete) operations.

Non-Relational Database (NoSQL)

- **Non-relational databases** different from relational databases because they do not store data in tabular form.
- Instead, non-relational databases are based on data structures like documents and graphs. **NoSQL** databases also come in a variety of types based on their data models.
- They offer scalability when dealing with **large volumes** of data and high load factors. They were designed when data was expected to be partitioned across multiple machines to scale, in contrast to relational databases, which assumed the data would stay on a single machine.

The Benefits Of A Non-Relational Database

- **Scalability**: Non-relational databases like **MongoDB** and **Cassandra** are designed to horizontally scale across **clusters** of **cheap** commodity hardware, offering seamless **scalability** as data volumes and user loads increase.
- **Flexibility in Data Models**: Unlike rigid table-based structures in relational databases, non-relational databases support flexible data models like document stores (e.g., **JSON** in MongoDB), **key-value pairs** (e.g., **Redis**), and wide-column stores (e.g., Cassandra), making it easier to store and manage unstructured or semi-structured data.
- **Performance**: Non-relational databases are optimized for specific use cases such as real-time data ingestion, high-speed **transactions**, and rapid access to large volumes of data. They often outperform relational databases in these scenarios due to their distributed architecture and optimized data storage formats.
- **Schemaless Design**: Non-relational databases typically do not enforce a rigid schema, allowing developers to evolve the data structure over time without downtime or complex migrations. This advantage is particularly beneficial in agile development environments and for handling diverse and **unpredictable** data types.
- **High Availability and Fault Tolerance**: Many non-relational databases are designed with built-in replication and automatic failover capabilities, ensuring high availability and data redundancy. This makes them suitable for mission-critical applications where continuous uptime is essential.
- **Cost-Effectiveness**: By using commodity hardware and open-source software, non-relational databases often provide a more cost-effective solution compared to traditional relational databases, especially at scale.

What Do NoSQL Databases Have In Common?

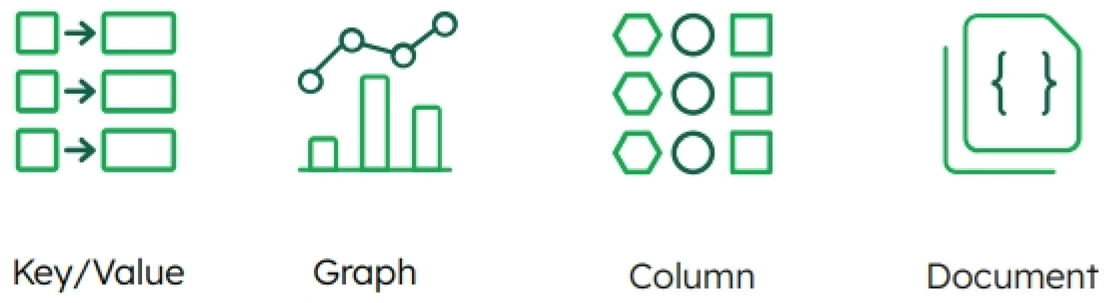
- **Non-Relational Structure**: NoSQL databases store data in flexible formats like key-value pairs, documents, or graphs, allowing for easier adaptation to changing data needs.
- **Scalability**: Designed for horizontal scaling across multiple servers, enabling efficient handling of large data volumes and high transaction rates.
- **High Performance**: Highly optimized for specific query types and workloads, prioritizing low latency and high throughput.
- **Flexibility in Data Models**: Supports various data structures (e.g., **documents**, columns, graphs) to fit diverse application requirements without rigid schemas.

- **Eventual Consistency**: Emphasizes availability and partition tolerance over strict immediate consistency across distributed nodes.
- **Horizontal Partitioning**: Uses sharding to distribute data across multiple **nodes**, improving performance and managing large datasets efficiently.

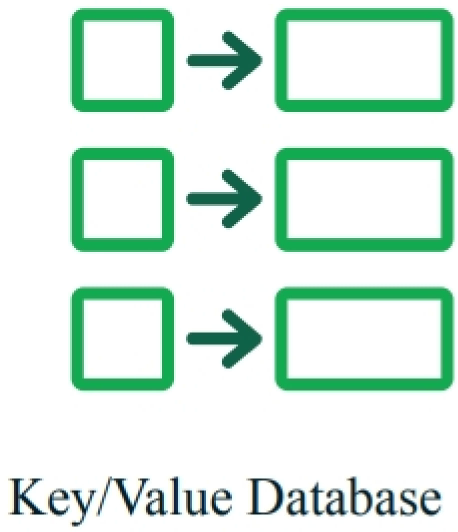
Non-Relational Database Types

There are four main types of non-relational databases:

- key/value
- graph
- column
- document



1. Key/Value Database



Key-value databases use a straightforward schema: a unique key is paired with a collection of values, where the values can be anything from a string to a large binary object. One of the benefits of using this structure in a database is that you don't have to worry about complex queries. Because the system knows where the data is stored, it only sends a request to that particular server.

Example of Key/Value Database

Key	**Value**
Name	**John Snow**
Age	**23**

2. Graph Database

Graph Database

<u>Graph database</u> is another type of non-relational database. A popular example of a graph database is **Neo4J**. This database stores information as a collection of nodes and edges, where the edges represent the relationships between the nodes.

3. Column Oriented Database

Column Oriented
or Wide Column

A column-oriented or wide-column non-relational database is primarily designed for analytics. **Cassandra** is a commonly used column-oriented database.

The advantage of column-oriented/row-oriented databases is that column-oriented databases return data in columns, making the query much more performant as it will not return many irrelevant fields that are not required for the query being serviced.

The primary key in a column-oriented database is the data or value, which is then mapped to row keys. This is the inverse, or opposite, of how the primary key works in a relational database.

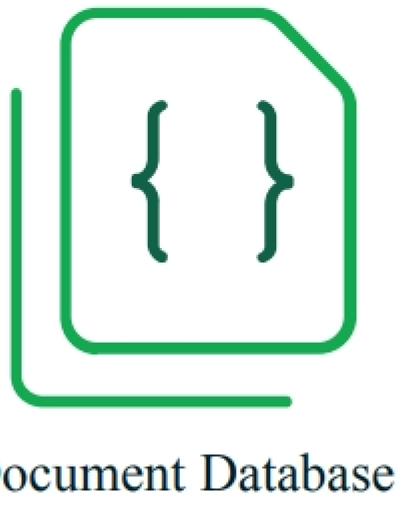
Example of Column Oriented Database

Name	ID
John	001
Sherlock	002

Height	ID
5’9	001
6’3	002

Age	ID
23	001
28	002

4. Document Database



Document databases, such as MongoDB, store data in a single document, which can have different shapes within the single collection or table that stores the documents. It provides a clear means of capturing relationships using sub-documents and embedded arrays within a single document.

Example of Document Database

```
{
  "_id":
ObjectId("5ef2d4b45b7f11b6d7a"),
  "user_id": "John Snow",
  "age": 23,
  "address":
    {
      "Country: "England"
      "City":"London",
      "Street": "Baker St."
    },
"Hobbies": [ violin ]
}
```

Non-Relational Database Management Systems

Some of the popular Non-Relational Database Management Systems are:

1. **MongoDB**
2. **Apache Cassandra**
3. **Redis**
4. **Couchbase**
5. **Apache HBase**
6. **Neo4j**
7. **Riak**
8. **Aerospike**
9. **OrientDB**
10. **ArangoDB**

These are some **Non-relational database names**, that you might hear in the market. Decide on which **Non-relational database software** is best for your work, and master that.

Relational And Non-Relational Database

Here's a comparison of Relational and Non-Relational Databases in tabular format:

Feature	Relational Database	Non-Relational Database
Data Structure	Tables with rows and columns	Various formats (document, key-value, columnar, graph)
Schema	Structured schema enforced by schemas	Flexible schema, often schema-less or dynamic
Query Language	SQL (Structured Query Language)	Query languages specific to the database type (e.g., JSON query languages, graph traversal languages)
ACID Compliance	ACID transactions	May vary; some offer ACID compliance, others eventual consistency

Feature	Relational Database	Non-Relational Database
Scalability	Vertical and horizontal scaling options	Horizontal scaling typically easier and more flexible
Flexibility	Less flexible with rigid schema definitions	Highly flexible due to schema-less or dynamic schema
Performance	Excellent for complex queries and joins	Optimal for hierarchical data storage and retrieval
Examples	MySQL, PostgreSQL, SQL Server	MongoDB, Cassandra, Redis, DynamoDB

Conclusion

In conclusion, the choice between relational and non-relational databases depends largely on the nature of the data and the requirements of the application. Relational databases excel in structured data environments where data integrity and complex querying are paramount. On the other hand, non-relational databases shine in scenarios demanding scalability, flexibility in data models, and high performance across distributed systems.

FAQs On Non-Relational Databases And Their Types

What are the main advantages of using a non-relational database?

Non-relational databases offer scalability, flexible data models, high performance, schemaless design, and cost-effectiveness compared to traditional relational databases.

When should I opt for a relational database instead of a non-relational one?

Opt for a relational database when dealing with structured data that requires strict data integrity, complex querying capabilities, and transactions adhering to ACID properties.

What are some popular examples of non-relational databases?

MongoDB, Cassandra, Redis, Couchbase, and Neo4j are among the popular non-relational databases used in various industries for their specific strengths in handling diverse data types and high-volume transactions.

How does horizontal scaling work in non-relational databases?

Non-relational databases achieve horizontal scaling by distributing data across multiple servers (nodes) in a cluster. This approach allows them to handle large volumes of data and high user loads efficiently.