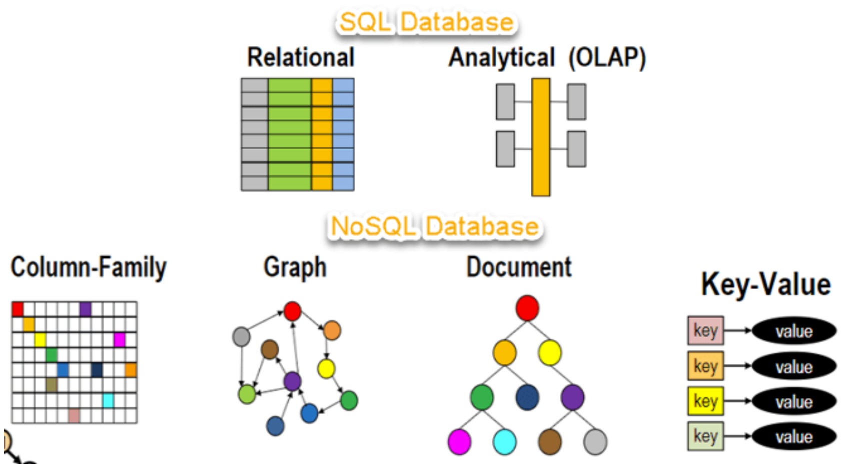
# NoSQL

## What is NoSQL?

NoSQL is a non-relational DBMS, that does not require a fixed schema, avoids joins, and is easy to scale. NoSQL database is used for distributed data stores with massive data storage needs. NoSQL is used for Big data and real-time web apps. For example, companies like Twitter, Facebook, Google that collect terabytes of user data every single day.

NoSQL database stands for "Not Only SQL" or "Not SQL" Traditional RDBMS uses SQL syntax to store and retrieve data for further insights. Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data.

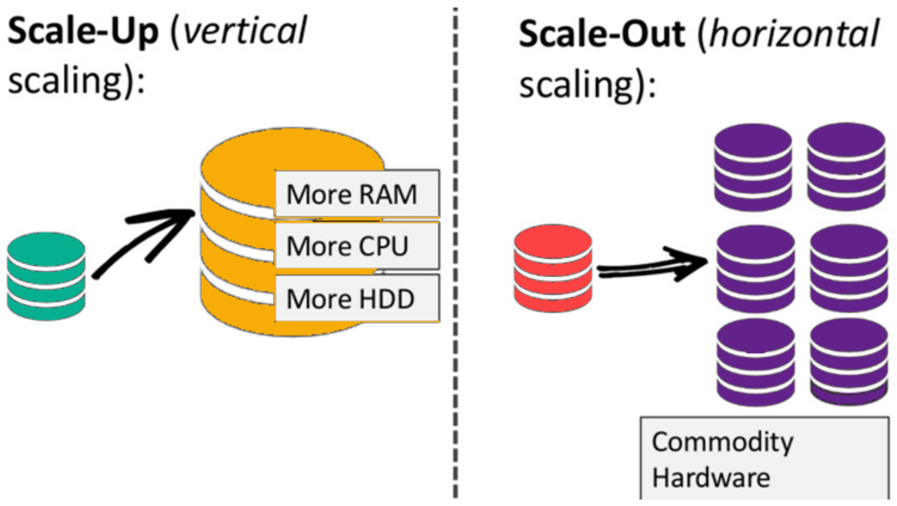


## Why NoSQL?

The concept of NoSQL databases became popular with Internet giants like Google, Facebook, Amazon, etc. who deal with huge volumes of data. The system response time becomes slow when you use RDBMS for massive volumes of data.

To resolve this problem, we could "scale up" our systems by upgrading our existing hardware. This process is expensive. The alternative for this issue is to distribute database load on multiple hosts whenever the load increases. This method is known as "scaling out."

NoSQL database is non-relational, so it scales out better than relational databases as they are designed with web applications in mind.


Scale-Up (*vertical* scaling):
More RAM
More CPU
More HDD
Scale-Out (*horizontal* scaling):
Commodity Hardware

## Features / Characteristics of NoSQL:

- It’s more than rows in tables—NoSQL systems store and retrieve data from many formats: key-value stores, graph databases, column-family (Bigtable) stores, document stores, and even rows in tables.
- It’s free of joins—NoSQL systems allow to extract data using simple interfaces without joins.
- It’s schema-free—NoSQL systems allow to drag-and-drop data into a folder and then query it without creating an entity-relational model i.e. offers heterogeneous structures of data in the same domain.
- It works on many processors—NoSQL systems allow to store database on multiple processors and maintain high-speed performance.
- Multiple NoSQL databases can be executed in a distributed fashion. It uses shared-nothing commodity computers—Most (but not all) NoSQL systems leverage low-cost commodity processors that have separate RAM and disk.
- It supports linear scalability—When you add more processors, you get a consistent increase in performance.
- It’s innovative—One can use several ways of storing, retrieving, and manipulating data. NoSQL supporters (also known as NoSQLers) have an inclusive.
- The final but certainly no less important key feature to seek in a NoSQL database is zero downtime. This is made possible by a masterless architecture, which allows for multiple copies of data to be maintained across different nodes. If a node goes down, no problem: another node has a copy of the data for easy, fast access.

## Difference between RDBMS and NoSQL Databases:

| RDBMS | NoSQL |
|---|---|
| Data is stored in a relational model, with rows and columns. A row contains information about an item while columns contain specific information, such as ‘Model’, ‘Date of Manufacture’, ‘Color’. | Data is stored in a host of different databases, with different data storage models. |
| Follows fixed schema. Meaning, the columns are defined and locked before data entry. In addition, each row contains data for each column. | Follows dynamic schemas. Meaning, you can add columns anytime. |
| Supports vertical scaling. Scaling an RDBMS across multiple servers is a challenging and time-consuming process. | Supports horizontal scaling. One can scale across multiple servers. Multiple servers are cheap commodity hardware or cloud instances, which make scaling cost-effective compared to vertical scaling. |
| Atomicity, Consistency, Isolation and Durability (ACID) Compliant | Not ACID Compliant. |

## Types of NoSQL Database:

There are mainly four categories of NoSQL databases. Each of these categories has its unique attributes and limitations. No specific database is better to solve all problems. One should select a database or a combination of multiple database based on his product needs.

1. Key-Value database
2. Document-based database
3. Column-based database
4. Graph-based database

## Key-Value database:

A key-value database, is a database that uses a simple key-value method to store data. Data is stored in key/value pairs. It is designed in such a way to handle lots of data and heavy load. Key-value pair storage databases store data as a hash table where each key is unique, and the value can be a JSON, BLOB(Binary Large Objects), string, etc.

Here's an example of a key-value store:

| Key | Value |
|---|---|
| Bob | (123) 456-7890 |
| Jane | (234) 567-8901 |

This type of database is very quick to query, due to its simplicity. The key-value model is well suited to storing things like user profiles and session info on a website, blog comments, telecom directories, IP forwarding tables, shopping cart contents on e-commerce sites, and more.

**Example:** Voldemorte, Aerospike, Oracle Berkeley DB, etc.

## Document-based database:

Document-based NoSQL database stores and retrieves data as a key value pair but the value part is stored as a document. The document is stored in JSON or XML formats. The value is understood by the DB and can be queried. Here's an example of a document written in JSON:

```
{
   '_id' : 1,
   'writerName' : { 'Christopher Manning' },
   'books' : [
      {
         'bookname' : 'Introduction to Information Retrieval',
         'datereleased' : 2008,
      }
   ]
}
```

In a document store, this document can be retrieved by referring to the _id. The document type is mostly used for CMS systems, blogging platforms, real-time analytics & e-commerce applications. It should not use for complex transactions which require multiple operations or queries against varying aggregate structures.

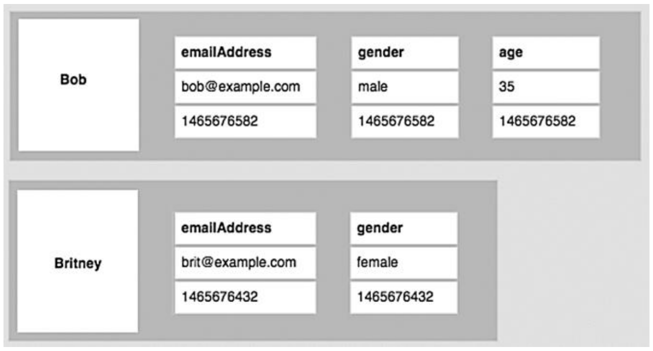
**Example:** MongoDB, DocumentDB, OrientDB, etc.

## Column-based database:

A column store database is a type of database that stores data using a column oriented model.

Column stores work in a similar fashion to relational databases in that they have rows, columns, and tables (also known as column families). However, these work differently in column store databases.

In a column store database, the columns in each row are contained within that row. Each row can have different columns to the other rows. They can be in a different order, then can even have different data types, etc.

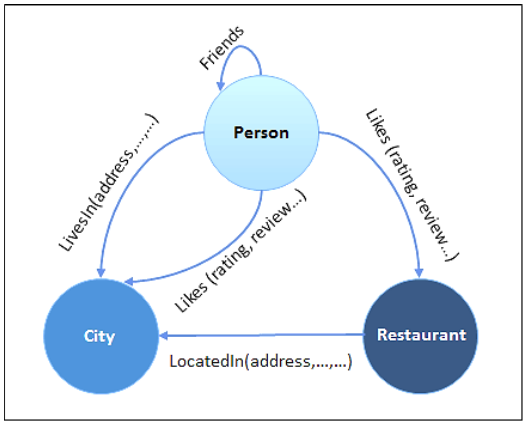


This method of storing data can be extremely quick to load and query. Columnar databases can also store massive amounts of data, and they can be scaled easily using massively parallel processing (MPP), which involves having data spread across a large cluster of machines.

**Example:** Bigtable, Cassandra, Hbase, etc.

## Graph-based database:

A graph type database stores entities as well the relations amongst those entities. The entity is stored as a node with the relationship as edges. An edge gives a relationship between nodes. Every node and edge has a unique identifier.


Friends
Person
LivesIn(address,.....)
Likes (rating, review...)
Likes (rating, review...)
City
Restaurant
LocatedIn(address,...,...)

Graph databases are very well suited to applications like social networks, realtime product recommendations, network diagrams, fraud detection, access management, etc.

**Example:** Neo4j, Blazegraph, etc.

## Querying Mechanism of NoSQL Databases:

The Structured Query Language (SQL) used by traditional databases provides a uniform way to communicate with the server when storing and retrieving data. SQL syntax is highly standardized, so while individual databases may handle certain operations differently (e.g., window functions), the basics remain the same.

By contrast, each NoSQL database tends to have its own syntax for querying and managing the data. CouchDB, for instance, uses requests in the form of JSON, sent via HTTP, to create or retrieve documents from its database. MongoDB sends JSON objects over a binary protocol, by way of a command-line interface or a language library.

Some NoSQL products can use SQL-like syntax to work with data, but only to a limited extent. For example, Apache Cassandra, a column store database, has its own SQL-like language, the Cassandra Query Language or CQL. Some of the CQL syntax is straight out of the SQL playbook, like the SELECT or INSERT keywords. But there is no way to perform a JOIN or subquery in Cassandra, and thus the related keywords don’t exist in CQL.

## Dynamic Schemas:

Relational databases require that schemas be defined before you can add data. For example, you might want to store data about your customers such as phone numbers, first and last name, address, city and state – a SQL database needs to know what you are storing in advance.

This fits poorly with agile development approaches, because each time you complete new features, the schema of your database often needs to change. So if you decide, a few iterations into development, that you'd like to store customers' favorite items in addition to their addresses and phone numbers, you'll need to add that column to the database, and then migrate the entire database to the new schema.

If the database is large, this is a very slow process that involves significant downtime. If you are frequently changing the data your application stores – because you are iterating rapidly – this downtime may also be frequent. There's also no way, using a relational database, to effectively address data that's completely unstructured or unknown in advance.

NoSQL databases are built to allow the insertion of data without a predefined schema. That makes it easy to make significant application changes in real-time, without worrying about service interruptions – which means development is faster, code integration is more reliable, and less database administrator time is needed. Developers have typically had to add application-side code to enforce data quality controls, such as mandating the presence of specific fields, data types or permissible values. More sophisticated NoSQL databases allow validation rules to be applied within the database, allowing users to enforce governance across data, while maintaining the agility benefits of a dynamic schema.

## CAP (also called Brewer's) Theorem:

CAP theorem tries to demonstrate the properties expected by a NoSQL database. Most of the databases are designed to achieve two of these properties at the cost of another property.

C — Consistency
This demonstrates the guarantee on the execution of updates and the availability of the updates as soon as they are acknowledged to the updater. In simpler terms if a database is consistent, updates are available as soon as they are completed, which is not a guarantee in a distributed environment.

A — Availability
This demonstrates the property of a database where it is capable to serve a request. Most of the SQL databases drop queries if the load/execution times are greater. Availability is expected to be very high and response times are expected to be very low in NoSQL databases by elimination of transactional properties that are present in SQL databases.

P — Partition tolerance
The property of databases being able to function with failures among nodes due to network issues. For an example a database may contain several nodes (MongoDB nodes) that work together (By a mechanism such as Mapreduce). The property is preserved if the database as a whole can operate even one or more nodes are unreachable in a distributed environment.

## Advantages of NoSQL Databases:

Some advantages of NoSQL databases for data science are articulated below:

- ❖ Store data from structured, semistructured and unstructured sources.
- ❖ Store and transmit all types of data in JavaScript Object Notation (JSON), which is a standard key-and-value pair format that is easy for both humans and applications to read and write. JSON is especially optimal for dynamic web pages and is natively supported in most standard programming languages.

- ❖ Use SQL to enable access, query and manipulation of both relational and nonrelational data types.
- ❖ Store data in a schema-on-read model that is agnostic to data sources and downstream uses, while offering the flexibility and extensibility that data science projects require.
- ❖ Implement distributed architectures that can scale horizontally to big data volumes, varieties and velocities, while also enabling the massively parallel processing (MPP) required of the most complex, resource-intensive data science pipeline functions.
- ❖ Simplify the complex problem of managing application state at large scale, enabling application developers to store application state locally on devices—tablets, smartphones and wearable devices—while synchronizing that state with the nearest NoSQL cloud database instance.
- ❖ Streamline complex data delivery scenarios by using cloud data services rather than on-premises software deployment to manage distributed database services.

## Disadvantages of NoSQL Databases:

Some disadvantages of NoSQL databases for data science are articulated below:

- ❖ Narrow focus – NoSQL databases have very narrow focus as it is mainly designed for storage but it provides very little functionality. Relational databases are a better choice in the field of Transaction Management than NoSQL.
- ❖ Open-source – NoSQL is open-source database. There is no reliable standard for NoSQL yet. In other words two database systems are likely to be unequal.
- ❖ Management challenge – The purpose of big data tools is to make management of a large amount of data as simple as possible. But it is not so easy. Data management in NoSQL is much more complex than a relational database. NoSQL, in particular, has a reputation for being challenging to install and even more hectic to manage on a daily basis.
- ❖ GUI is not available – GUI mode tools to access the database is not flexibly available in the market.
- ❖ Backup – Backup is a great weak point for some NoSQL databases like MongoDB. MongoDB has no approach for the backup of data in a consistent manner.
- ❖ Large document size – Some database systems like MongoDB and CouchDB store data in JSON format. Which means that documents are quite large (BigData, network bandwidth, speed), and having descriptive key names actually hurts, since they increase the document size

## How to Choose the Right NoSQL Database?

NoSQL databases provide high operational speed and increased flexibility for software developers and other users when compared to traditional tabular (or SQL) databases. NoSQL databases can be scaled across thousands of servers, though sometimes with loss of data consistency. But what makes NoSQL databases especially relevant today is that they are particularly well suited for working with large sets of distributed data, which makes them a good choice for big data and analytics projects.

There are more than two dozens of open source and commercial NoSQL databases are available in the market which are vary in architecture and function. So one need to pick the right database type that is best suited for his desired task.

- ❖ In general, key-value stores are best for the persistent sharing of data by multiple processes in an application. The key-value model is well suited to storing things like user profiles and session info on a website, blog comments, telecom directories, IP forwarding tables, shopping cart contents on e-commerce sites, and more.
- ❖ The document type is mostly used for CMS systems, blogging platforms, real-time analytics & e-commerce applications. It should not use for complex transactions which require multiple operations or queries against varying aggregate structures.
- ❖ If you plan to do deep relationship analysis for proximity calculation, social networks analysis, realtime product recommendations, fraud detection, etc. a graph database might be the better choice.
- ❖ If you need to collect data very rapidly and at high volumes for analytics, look at a column-based database. Moreover, it can be scaled easily using massively parallel processing (MPP), which involves having data spread across a large cluster of machines. Such NoSQL databases tend also to offer document and graph support as well.