**SQL vs NoSQL: What's the difference?**

<https://www.guru99.com/sql-vs-nosql.html>

**What is SQL?**

Structured Query language (SQL) **pronounced as "S-Q-L" or sometimes as "See-Quel**" is the standard language for dealing with Relational Databases. A relational database defines relationships in the form of tables.

SQL programming can be effectively used to insert, search, update, delete database records.

That doesn't mean SQL cannot do things beyond that. It can do a lot of things including, but not limited to, optimizing and maintenance of databases.

Relational databases like MySQL Database, Oracle, Ms SQL Server, Sybase, etc. use SQL.

**What is NoSQL?**

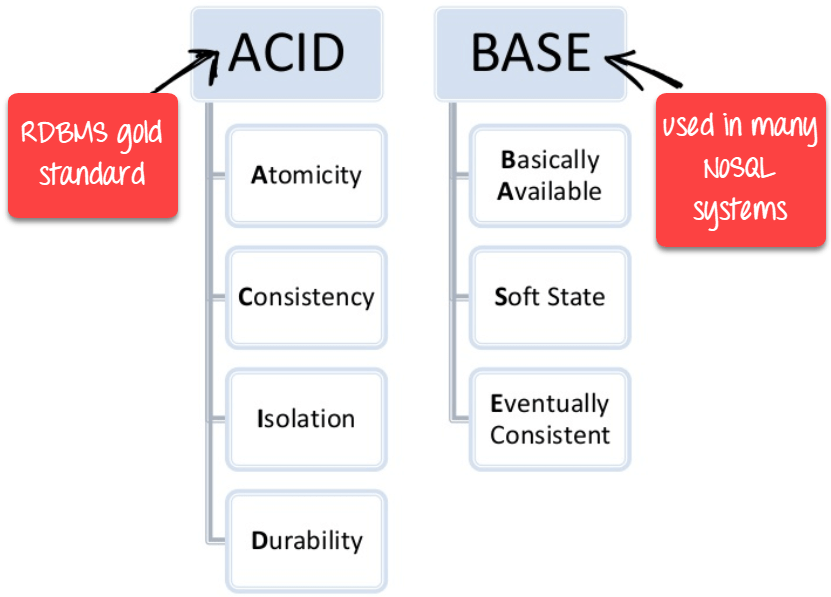
NoSQL is a non-relational DMS, that does not require a fixed schema, avoids joins, and is easy to scale. NoSQL database is used for distributed data stores with humongous data storage needs. NoSQL is used for Big data and real-time web apps. For example companies like Twitter, Facebook, Google that collect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

NoSQL database stands for "Not Only SQL" or "Not SQL." Though a better term would NoREL NoSQL caught on. Carl Strozz introduced the NoSQL concept in 1998.

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.

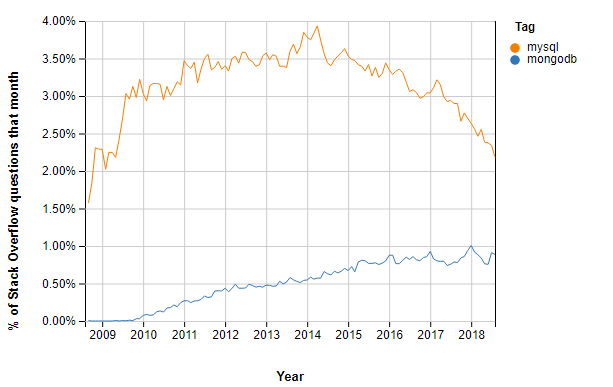
**Difference between SQL and NoSQL**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **SQL** | **NOSQL** |
| Definition | SQL databases are primarily called RDBMS or Relational Databases | NoSQL databases are primarily called as Non-relational or distributed database |
| Design for | Traditional RDBMS uses SQL syntax and queries to analyze and get the data for further insights. They are used for OLAP systems. | NoSQL database system consists of various kind of database technologies. These databases were developed in response to the demands presented for the development of the modern application. |
| Query Language | Structured query language (SQL) | No declarative query language |
| Type | SQL databases are table based databases | NoSQL databases can be document based, key-value pairs, graph databases |
| Schema | SQL databases have a predefined schema | NoSQL databases use dynamic schema for unstructured data. |
| Ability to scale | SQL databases are vertically scalable | NoSQL databases are horizontally scalable |
| Examples | Oracle, Postgres, and MS-SQL. | MongoDB, Redis, , Neo4j, Cassandra, Hbase. |
| Best suited for | An ideal choice for the complex query intensive environment. | It is not good fit complex queries. |
| Hierarchical data storage | SQL databases are not suitable for hierarchical data storage. | More suitable for the hierarchical data store as it supports key-value pair method. |
| Variations | One type with minor variations. | Many different types which include key-value stores, document databases, and graph databases. |
| Development Year | It was developed in the 1970s to deal with issues with flat file storage | Developed in the late 2000s to overcome issues and limitations of SQL databases. |
| Open-source | A mix of open-source like Postgres & MySQL, and commercial like Oracle Database. | Open-source |
| Consistency | It should be configured for strong consistency. | It depends on DBMS as some offers strong consistency like MongoDB, whereas others offer only offers eventual consistency, like Cassandra. |
| Best Used for | RDBMS database is the right option for solving ACID problems. | NoSQL is a best used for solving data availability problems |
| Importance | It should be used when data validity is super important | Use when it's more important to have fast data than correct data |
| Best option | When you need to support dynamic queries | Use when you need to scale based on changing requirements |
| Hardware | Specialized DB hardware (Oracle Exadata, etc.) | Commodity hardware |
| Network | Highly available network (Infiniband, Fabric Path, etc.) | Commodity network (Ethernet, etc.) |
| Storage Type | Highly Available Storage (SAN, RAID, etc.) | Commodity drives storage (standard HDDs, JBOD) |
| Best features | Cross-platform support, Secure and free | Easy to use, High performance, and Flexible tool. |
| Top Companies Using | Hootsuite, CircleCI, Gauges | Airbnb, Uber, Kickstarter |
| Average salary | The average salary for any professional SQL Developer is $84,328 per year in the U.S.A. | The average salary for "NoSQL developer" ranges from approximately $72,174 per year |
| ACID vs. BASE Model | ACID( Atomicity, Consistency, Isolation, and Durability) is a standard for RDBMS | Base ( Basically Available, Soft state, Eventually Consistent) is a model of many NoSQL systems |

[](https://www.guru99.com/images/1/101818_0550_Differenceb2.png)

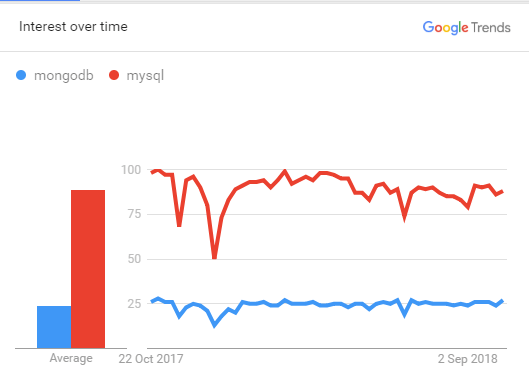
ACID vs BASE

**When use SQL?**

[](https://www.guru99.com/images/1/101818_0550_Differenceb3.png)NoSQL DB (Mongo) Vs RDBMS DB (MySQL) Stackoverflow Questions

* SQL is the easiest language used to communicate with the RDBMS
* Analyzing behavioral related and customized sessions
* Building custom dashboards
* It allows you to store and gets data from the database quickly
* Preferred when you want to use joins and execute complex queries

**When use NoSQL?**

[](https://www.guru99.com/images/1/101818_0550_Differenceb1.png)

NoSQL DB (mongo) Vs RDBMS DB (mysql) Google Trend

* When ACID support is not needed
* When Traditional RDBMS model is not enough
* Data which need a flexible schema
* Constraints and validations logic not required to be implemented in

database

* Logging data from distributed sources
* It should be used to store temporary data like shopping carts, wish list and session data

**KEY DIFFERENCE**

* **SQL** pronounced as "S-Q-L" or as "See-Quel" is primarily called RDBMS or Relational Databases whereas **NoSQL** is a Non-relational or Distributed Database.
* SQL databases are table based databases whereas NoSQL databases can be document based, key-value pairs, graph databases.
* SQL databases are vertically scalable while NoSQL databases are horizontally scalable.
* SQL databases have a predefined schema whereas NoSQL databases use dynamic schema for unstructured data.
* SQL requires specialized DB hardware for better performance while NoSQL uses commodity hardware.

# NoSQL Tutorial: Learn NoSQL Features, Types, What is, Advantages

# <https://www.guru99.com/nosql-tutorial.html>

## What is NoSQL?

NoSQL is a non-relational DMS, that does not require a fixed schema, avoids joins, and is easy to scale. NoSQL database is used for distributed data stores with humongous 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." Though a better term would NoREL NoSQL caught on. Carl Strozz introduced the NoSQL concept in 1998.

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.

# Introduction to NoSQL databases Tutorial

# <https://www.simplilearn.com/introduction-to-nosql-databases-tutorial-video>

Welcome to the first chapter of the MongoDB tutorial (part of the [MongoDB Developer and Administrator Course](https://www.simplilearn.com/big-data-and-analytics/mongodb-certification-training)). This tutorial provides an introduction to the NoSQL and NoSQL database. Let us explore the objectives of this lesson in the next section.

## Objectives

After completing this NoSQL tutorial, you will be able to:

* Explain what NoSQL databases are.
* Explain the purpose of NoSQL databases.
* List the benefits of NoSQL database over traditional RDBMS database.
* Identify various types of NoSQL databases.
* List the differences between NoSQL and RDBMS.
* Explain MongoDB in relation to the CAP theorem

Let us begin with understating what NoSQL is in the next section of this tutorial.

## What is NoSQL?

Traditionally, the software industries use relational databases to store and manage data persistently. Not only SQL or NoSQL is a new set of a database that has emerged in the recent past as an alternative solution to relational databases.

Carl Strozzi (Pronounce as Stro-jee ) introduced the term NoSQL to name his file-based database in 1998.

NoSQL refers to all databases and data stores that are not based on the Relational Database Management Systems or RDBMS principles. It relates to large data sets accessed and manipulated on a Web scale. NoSQL does not represent single product or technology. It represents a group of products and a various related data concepts for storage and management. NoSQL was a hashtag that was chosen for a tech meetup to discuss the new databases.

We will learn about various database features of NoSQL in the next section of the tutorial.

**Want to check the course preview of our MongoDB Developer and Administrator Course?**[Click here to watch](https://www.simplilearn.com/big-data-and-analytics/mongodb-certification-training?source=GhPreviewCTAText#/course-preview)

## Database Features of NoSQL

NoSQL Databases can have a common set of features such as:

* Non-relational data model.
* Runs well on clusters.
* Mostly open-source.
* Built for the new generation Web applications.
* Is schema-less.

We will discuss the purpose of using NoSQL in the next section of this tutorial.

## Why NoSQL?

With the explosion of social media, user-driven content has grown rapidly and has increased the volume and type of data that is produced, managed, analyzed, and archived. In addition, new sources of data, such as sensors, Global Positioning Systems or GPS, automated trackers, and other monitoring systems generate huge volumes of data on a regular basis.

These large volumes of data sets, also called big data, have introduced new challenges and opportunities for data storage, management, analysis, and archival. In addition, data is becoming increasingly semi-structured and sparse. This means that RDBMS databases which require upfront schema definition and relational references are examined.

To resolve the problems related to large-volume and semi-structured data, a class of new database products has emerged. These new classes of database products consist of column-based data stores, key/value pair databases, and document databases. Together, these are called NoSQL. The NoSQL database consists of diverse products with each product having unique sets of features and value propositions.

In the next section of this NoSQL tutorial, we will identify the differences between NoSQL and RDBMS.

## Difference Between RDBMS and NoSQL Databases

Given below is a table that explains the difference between RDBMS (Relationship Database Management Systems) and NoSQL.

|  |  |
| --- | --- |
| 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’. * Follows fixed schema. Meaning, the columns are defined and locked before data entry. In addition, each row contains data for each column. * Supports vertical scaling. Scaling an RDBMS across multiple servers is a challenging and time-consuming process. * Atomicity, Consistency, Isolation & Durability(ACID) Compliant | * Data is stored in a host of different databases, with different data storage models. * Follows dynamic schemas. Meaning, you can add columns anytime. * Supports horizontal scaling. You can scale across multiple servers. Multiple servers are cheap commodity hardware or cloud instances, which make scaling cost-effective compared to vertical scaling. * Not ACID Compliant. |

In the next section, we will discuss the various benefits of NoSQL.

## Benefits of NoSQL

The desired technical characteristics of an enterprise-class NoSQL solution are as follows.

**Primary and Analytic Data Source Capability**

The first criterion of an enterprise-class NoSQL solution is it must serve as a primary or active data source that receives data from different business applications. It also must act as a secondary data source or analytic database that enhances business intelligence applications. From business perspective, the NoSQL database must be capable of quickly integrating all types of data viz structured, semi-structured, or unstructured. In addition, it must be able to execute high-performance queries.   
Once all the required data is collected in the database, data administrators may want to perform an analysis in real time and in map-reduce form. An enterprise-class NoSQL database can easily handle such requests using the same database. It does not require loading the data into a separate analytic database for analysis.

**Big Data Capability**

NoSQL databases are not restricted to working with big data. However, an enterprise-class NoSQL solution can scale to manage large volumes of data from terabytes to petabytes. In addition to storing large volumes of data, it delivers high performance for data velocity, variety, and complexity.

**Continuous Availability or No Single Point of Failure**

To be considered enterprise-class, a NoSQL database must offer continuous availability, with no single point of failure. Moreover, rather than providing the continuous availability feature outside the software, the NoSQL solution delivers inherent continuous availability.

The NoSQL databases must include the following key features:

* All nodes in a cluster must be able to serve read request even if some machines are down.
* Must be capable of easily replicating and segregating data between different physical shelves in a data center. This helps avoid hardware outages.
* Must be able to support data distribution designs that are multi-data centers, on-premises or in the cloud.

**Multi-Data Center Capability**

Typically, business enterprises own highly distributed databases that are spread across multiple data centers and geographic locales. Data replication is a feature offered by all legacy RDBMS. However, none can offer a simple model of data distribution between various data centers without any performance issue.

A simple method includes the ability to handle multiple data centers without concerning the occurrences of the read and write operations. A good NoSQL enterprise solution must support multi data-center deployment and must provide configurable option to maintain a balance between performance and consistency.

**Easy Replication for Distributed Location-Independent Capabilities**

To avoid data loss affecting an application, a good NoSQL solution provides strong replication abilities. These include a read-anywhere and write-anywhere capability with full location-independence support.

This means you can write data to any node in a cluster, have it replicated on other nodes, and make it available to all users irrespective of their location. In addition, the write capability on any node must ensure that the data is safe in the event of a power failure or any other incident.

**No Need for Separate Caching Layer**

A good NoSQL solution is capable of using multiple nodes and distributing data among all the participating nodes.

Thus, it does not require a specific caching layer to store data. The memory caches of all participating nodes can store data for quick input and output or I/O access. NoSQL database eliminates the problem of synchronizing cache data with the persistent database. Thus, it supports simple scalability with fewer management issues.

**Cloud-Ready**

As an adaptation of cloud infrastructure is increasing day by day, an enterprise-class NoSQL solution must be cloud-ready.

A NoSQL database cluster must be able to function in a cloud setting, such as Amazon EC2, and also must be able to expand and contract a cluster when necessary. It also must support a hybrid solution where part of the database is hosted within the enterprise premise and another part is hosted in a cloud setting.

**High Performance with Linear Scalability**

An enterprise-class NoSQL database can enhance performance by adding nodes to a cluster. Typically, the performance of database systems may go down when additional nodes are added to a cluster. However, a good NoSQL solution increase performance for both read and writes operations when additional nodes are added. These performance gains are linear in nature.

**Flexible Schema Support**

An enterprise-class NoSQL database offers a flexible or dynamic schema design to manage all types of data—structured, semi-structured, and non-structured. Therefore, the need to have different vendors to support the different data types does not arise.

NoSQL databases may support various schema formats, such as columnar/Bigtable and document.

Therefore, choosing an appropriate database based on application requirement is a key design decision.

The flexible or dynamic schema support ensures that you can make schema changes to a structure without making the structure offline. This support is critical considering the near-zero downtime and round-the-clock availability for business applications.

**Support Key Developer Languages and Platforms**

Ideally, an enterprise-class NoSQL solution must support all major operating systems. In addition, it must run on a product hardware that does not require any tweaks or other proprietary add-ons.

The NoSQL database must provide client interfaces and drivers for all common developer languages. It must offer a structured query language or SQL or a similar language that helps store and access data in a NoSQL database.

**Easy to Implement, Maintain, and Grow**

A NoSQL database must be simple but robust. In other words, it must be easy to implement and use and must offer sturdy functionality to handle various enterprise applications.

In addition, the NoSQL vendor must supply good management tools that assist data professionals to perform various administrative tasks, such as adding capacity to a cluster, running utility tasks, and so on.

The NoSQL database must allow easy growth without making any change to the front-end of the business application.

**Thriving Open Source Community**

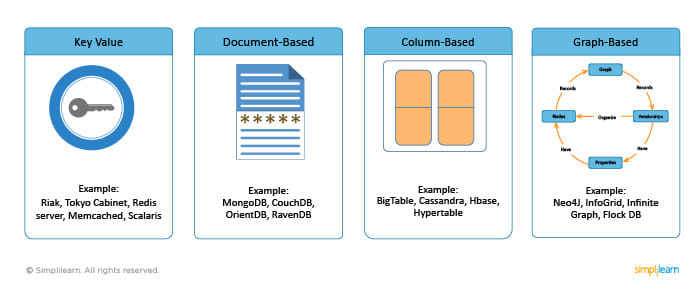
For an open source NoSQL database, having a vibrant community is essential to make a regular contribution to enhance the core software.

Moreover, open source communities generally provide excellent quality assurance or QA testing. This sometimes eliminates the need for software companies to hire, train, and retain a QA team.

To encourage a thriving open source community, including activities on mailing lists and technical forums, initiate technical discussions, and participate in conferences.

In the next section of this tutorial, we will discuss the various types of NoSQL databases.

## Types of NoSQL

There are four basic types of NoSQL databases.  


**Key-Value database**

Key-Value database has a big hash table of keys and values. Riak (Pronounce as REE-awk), Tokyo Cabinet, Redis server, Memcached ((Pronounce as mem-cached), and Scalaris are examples of a key-value store.

**Document-based database**

Document-based database stores documents made up of tagged elements. Examples: MongoDB, CouchDB, OrientDB, and RavenDB .

**Column-based database**

Each storage block contains data from only one column, Examples: BigTable, Cassandra, Hbase, and Hypertable.

**Graph-based database**

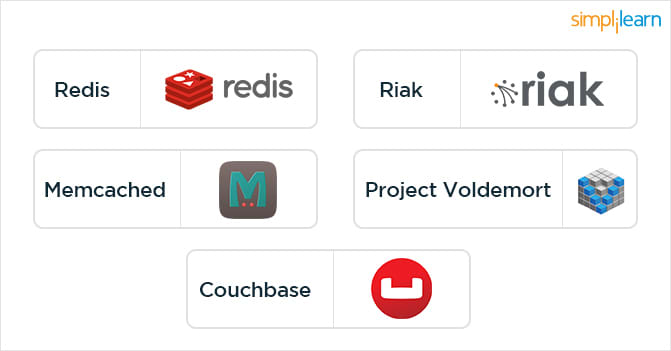
A graph-based database is a network database that uses nodes to represent and store data. Examples are Neo4J, InfoGrid, Infinite Graph, and FlockDB. The availability of choices in NoSQL databases has its own advantages and disadvantages.

The advantage is, it allows you to choose a design according to your system requirements. However, because you have to make a choice based on requirements, there is always a chance that the same database product may not be used properly.

We will learn about the key-value database in the next section of this tutorial.

### 

## Key-Value Database

From an Application Program Interface or API perspective, a key-value database is the simplest NoSQL database.   


This database stores every single item as a key with a value. You can get the value for a key, add a value for a key, or delete a key. The value is a blob that the database stores without knowing its content. The responsibility lies with the application to understand what is stored. Typically, key-value databases use primary-key access.

Therefore, they generally offer enhanced performance and scalability. All key-value databases may not have the same features.

For example, data is not persistent in Memcached while it is in Riak.

These features are important when implementing certain solutions.

For example, you need to implement caching of user preferences.

If you implement them in Memcached, you may lose all the data when the node goes down and may need to get them from the source database. If you store the same data in Riak, you may not lose data but must consider how to update the stale data. It is important to select a key-value database based on your requirements.

We will continue with the discussion on a key-value database in the next section.

## Advantages & Disadvantages of Key-Value Database

The key value store does not have a defined schema. It contains client defined semantics for understanding what the values are. A key value store is simple to build and easy to scale. It also tends to have great performance because the access pattern can be optimized to suit your requirement.

The advantages & disadvantages of the key-value store include the following.

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| * Queries: You can perform a query by using the key. Even range queries on the key are usually not possible. * Schema: Key value databases have the following schema - the key is a string, the value is a blob. The client determines how to parse data. * Usages: Key value databases can access data using a key. Key-value type database suffers from major weaknesses. | * It does not provide any traditional database capabilities, such as consistency when multiple transactions are executed simultaneously. * As the volume of data increases, maintaining unique values as keys become difficult. |

In the next section of the NoSQL tutorial, we will discuss Document database.

## Document Database

This NoSQL database type is based on the concept of documents.

* It stores and retrieves various documents in formats, such as XML, JavaScript Object Notation or JSON (Pronounce as JAY- Sahn), Binary JSON or BSON.
* These documents are self-descriptive, hierarchical tree data structures which consist of maps, collections, and scalar values.
* The stored documents can be similar to each other, but not necessarily the same. It stores documents in the value part of the key-value database. You can consider the document databases as key-value stores where you can examine the values.

In the next section, we will focus on examples of a document database.

## Document Database Example

Examples of Document databases are:

* MongoDB: Provides a rich query language and many useful features such as built-in support for MapReduce-style aggregation and geospatial indexes.
* Apache CouchDB: Uses JSON for documents, JavaScript for MapReduce indexes, and regular HTTP for its API.

Example: (MongoDB) document  
{Name:“SimpliLearn",  
Address:" 10685 Hazelhurst Dr, Houston, TX 77043, United States”,  
Courses: [“Big Data”,”Python”,”Android”,”PMP”,”ITIL”],  
Offices: [ “NYK”,”Dubai”,”BLR”],  
}

In the next section, we will learn about the Column-based database.

## Column-Based Database

Column-based databases store data in column families as rows. These rows contain multiple columns associated with a row key.  
  
Column families are groups of related data that is accessed together.  
  
For example, you may access customer profile information at the same time, but not their order history.

* Each column family is like a container of rows in an RDBMS table where the key identifies the rows.
* Each row consists of multiple columns. However, the various rows need not have the same columns. Moreover, you can add a column to any row at any time without adding it to other rows.

We will continue our discussion on the Column-based database in the next section of the tutorial.

**Want to test your MongoDB skills?** [Take the MongoDB free practice test](https://www.simplilearn.com/mongodb-developer-exam-prep-free-practice-test?source=lecture-page)

## Goals of Column-Based Database

The goal of a Column-based database is to efficiently read and write data to and from hard disk storage to quickly return a query. In this database, all column one values are physically together, followed by all the column two values.  
  
The data is stored in record order so that the 100th entry for column one and the column two are from the same input record. This allows you to access individual data elements, such as customer name, as a group in columns, rather than individually row-by-row.   
  
The compression permits columnar operations like MIN, MAX, SUM, COUNT, and AVG— to be performed very rapidly. A column-based database management system or DBMS is self-indexing, therefore it uses less disk space than an RDBMS containing the same data.  
  
We will continue our discussion column-based database in the next section of this tutorial.

## Diagramatic Representation of Column-Based Database column based database

The diagram provided in the section depicts that data is getting stored in a column rather than row format. It shows columns for the same column family are stored together in one file on the hard disk.

Therefore, these data can be retrieved fast in an efficient manner. In the next section, we will focus on an example of a column-based database.

## Column-Based Database Example

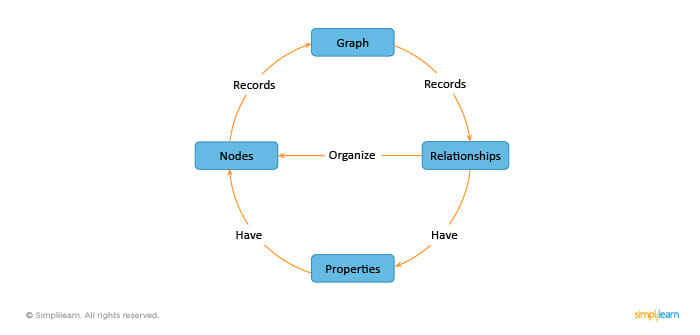
Cassandra is a column-based database.

* Cassandra is fast and easily scalable with write operations spread across the cluster.
* The cluster does not have a master node, hence, any node can handle the read and write operations.

 In the next section of this tutorial, we will discuss Graph database.

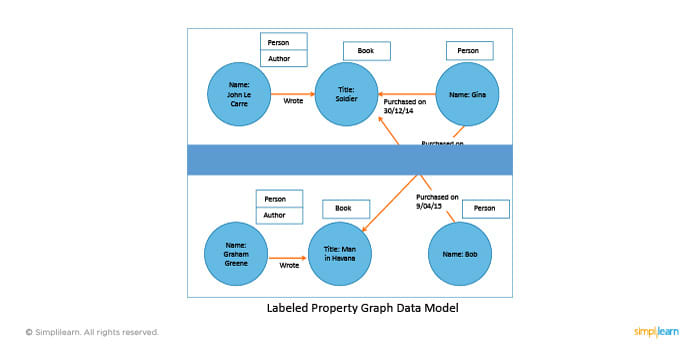
## Graph Database

A graph database lets you store data and its relationships with other data in the form of nodes and edges. Each relation can have a set of properties. Edges have a direction which has its own significance and enables you to explore the relationship in both the direction.

All the nodes in the graph are organized by relationships that help explore interesting and hidden patterns between the nodes.   
  
We will continue our discussion on Graph database in the next section.

## Examples of Graph Database

* Examples of graph database are Neo4J (pronounce as Neo- four-J), Infinite Graph, OrientDB, and FlockDB.
* Neo4J is one of the most popular graph databases, which is ACID compliant. It is the product of the company Neo Technologies. It is Java based but has bindings for other languages, including Ruby and Python.
* FlockDB was created by Twitter for relationship related analytics.

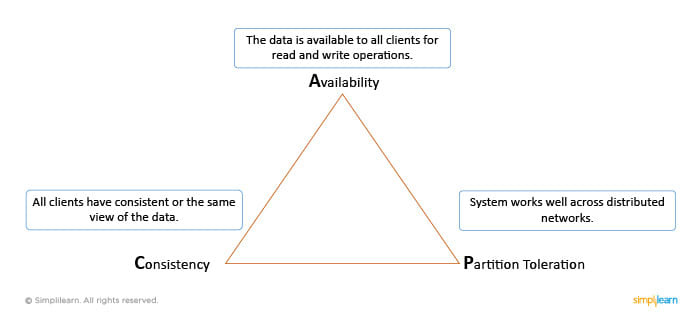
  
In the graph database, the labeled property graph model is used for modeling the data. It is same as the entity relationships or ER model used in RDBMS. The property graph contains connected entities, such as the nodes which can hold any number of attributes or key-value-pairs.

In the next section, we will discuss Consistency, Availability, and Partition tolerance or CAP theorem.

## CAP Theorem

In a distributed system, the following three properties are important.

* Consistency: Each client must have consistent or the same view of the data.
* Availability: The data must be available to all clients for read and write operations.
* Partition toleration: System must work well across distributed networks.



We will continue with the CAP Theorem in the next section.

## CAP Theorem (contd.)

The CAP theorem was proposed by Eric Brewer.

According to this theorem, in any distributed system, you can use only two of the three properties—consistency, availability, or partition tolerance simultaneously.  
  
Many NoSQL databases provide options for a developer to choose to adjust the database as per requirement. For this, understanding the following requirements is important:

* How the data is consumed by the system.
* Whether the data is read or write heavy.
* If there is a need to query data with random parameters.
* If the system is capable of handling inconsistent data.

We will focus on consistency in the next section of this tutorial.

**Related Read:**[3 Reasons Why Your Big Data Career Needs a MongoDB Certification](https://www.simplilearn.com/reasons-why-big-data-career-needs-mongodb-certification-article?souce=lecture-page)

## Consistency

### **Consistency in CAP theorem and ACID are explained below.**

### **Consistency in CAP theorem**

Consistency in CAP theorem refers to atomicity and isolation. Consistency means consistent read and write operations for the same sets of data so that concurrent operations see the same valid and consistent data state, without any stale data.

### **Consistency in ACID**

Consistency in ACID means if the data does not satisfy predefined constraints, it is not persisted. Consistency in CAP theorem is different. In a single-machine database, consistency is achieved using the ACID semantics.

However, in the case of NoSQL databases which are scaled out and distributed providing consistency gets complicated.

We will discuss availability in the next section of this tutorial.

## Availability

According to the CAP theorem, availability means:

* The database system must be available to operate when required. This means that a system that is busy, uncommunicative, unresponsive, or inaccessible is not available.
* If a system is not available to serve a request at a time it is needed, it is unavailable.

In the next section, we will discuss partition tolerance.

## Partition Tolerance

Partition tolerance or fault-tolerance is the third element of the CAP theorem. Partition tolerance measures the ability of a system to continue its service when some of its clusters become unavailable.

In the next section, we will learn about MongoDB in terms of the CAP theorem.

## MongoDB as Per CAP

By default, MongoDB offers strong consistency. This means after you perform a write operation, you cannot read the same data until the write operation is successful.

MongoDB is a single-master system and by default, all reads go to the primary node.

Optionally, if you enable reading from the secondary node, MongoDB becomes eventually consistent and allows reading of out-of-date results. In addition, MongoDB handles network partition very well by keeping same data on multiple nodes or replica set.

Therefore, MongoDB is a consistent and partition tolerant database which comprises the availability aspect.

## Summary

Here is a quick recap of what was covered in this lesson:

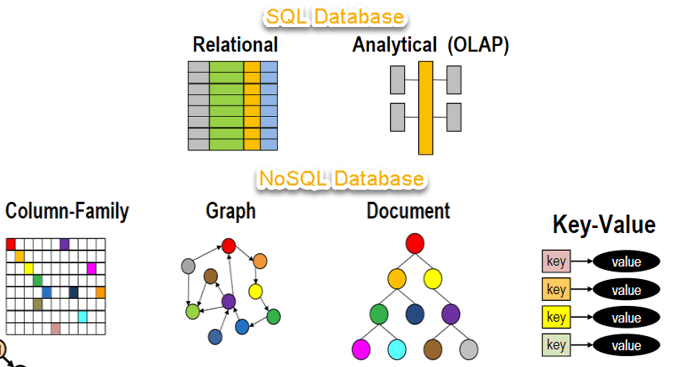
* NoSQL represents a class of products and a collection of diverse or related data concepts for storage and manipulation.
* NoSQL databases are used to efficiently manage large-volume and semi-structured data.
* The four basic NoSQL database types are— Key-Value, Document-based, Column-based, and Graph-based.
* According to the CAP theorem, a distributed computer system cannot provide all the three properties together—consistency, availability, and partition tolerance.

## Conclusion

This concludes the lesson on Introduction to NoSQL databases. In the next chapter, we will discuss [databases for the modern Web](https://www.simplilearn.com/mongodb-a-database-for-the-modern-web-tutorial-video).

### **Find our MongoDB Developer and Administrator Online Classroom training classes in top cities:**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Place |  |
| [MongoDB Developer and Administrator](https://www.simplilearn.com/big-data-and-analytics/mongodb-certification-training?source=lp-date-table) | 22 Feb -15 Mar 2020, Weekend batch | Your City | [View Details](https://www.simplilearn.com/big-data-and-analytics/mongodb-certification-training?source=lp-date-table) |

[](https://www.guru99.com/images/1/101818_0537_NoSQLTutori1.png)

In this tutorial, you will learn-

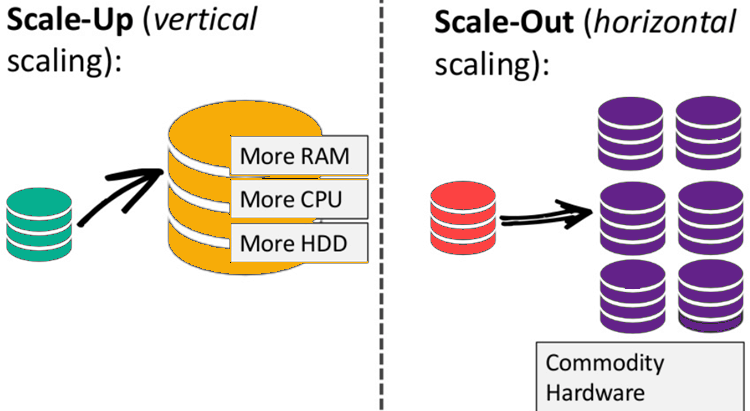
* [What is NoSQL?](https://www.guru99.com/nosql-tutorial.html#1)
* [Why NoSQL?](https://www.guru99.com/nosql-tutorial.html#2)
* [Brief History of NoSQL Databases](https://www.guru99.com/nosql-tutorial.html#3)
* [Features of NoSQL](https://www.guru99.com/nosql-tutorial.html#4)
* [Types of NoSQL Databases](https://www.guru99.com/nosql-tutorial.html#5)
* [Query Mechanism tools for NoSQL](https://www.guru99.com/nosql-tutorial.html#6)
* [What is the CAP Theorem?](https://www.guru99.com/nosql-tutorial.html#7)
* [Eventual Consistency](https://www.guru99.com/nosql-tutorial.html#8)
* [Advantages of NoSQL](https://www.guru99.com/nosql-tutorial.html#9)

## 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."

[](https://www.guru99.com/images/1/101818_0537_NoSQLTutori2.png)

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

## Brief History of NoSQL Databases

* 1998- Carlo Strozzi use the term NoSQL for his lightweight, open-source relational database
* 2000- Graph database Neo4j is launched
* 2004- Google BigTable is launched
* 2005- CouchDB is launched
* 2007- The research paper on Amazon Dynamo is released
* 2008- Facebooks open sources the Cassandra project
* 2009- The term NoSQL was reintroduced

## Features of NoSQL

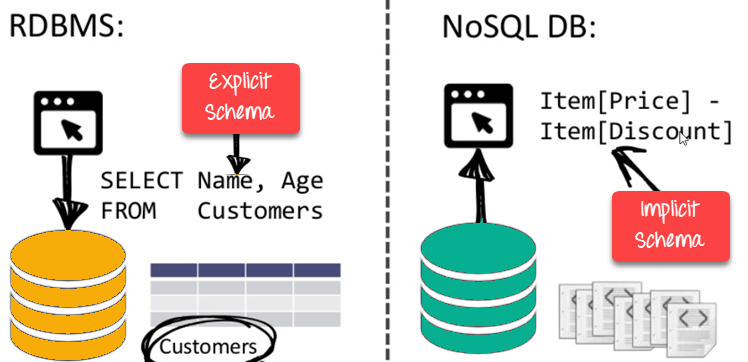
**Non-relational**

* NoSQL databases never follow the relational model
* Never provide tables with flat fixed-column records
* Work with self-contained aggregates or BLOBs
* Doesn't require object-relational mapping and data normalization
* No complex features like query languages, query planners,

referential integrity joins, ACID

**Schema-free**

* NoSQL databases are either schema-free or have relaxed schemas
* Do not require any sort of definition of the schema of the data
* Offers heterogeneous structures of data in the same domain

[](https://www.guru99.com/images/1/101818_0537_NoSQLTutori3.png)

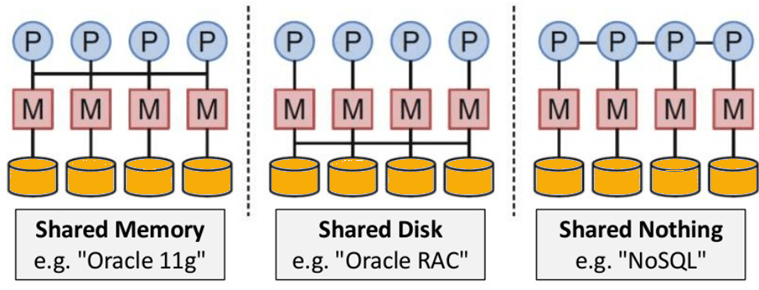
NoSQL is Schema-Free

**Simple API**

* Offers easy to use interfaces for storage and querying data provided
* APIs allow low-level data manipulation & selection methods
* Text-based protocols mostly used with HTTP REST with JSON
* Mostly used no standard based query language
* Web-enabled databases running as internet-facing services

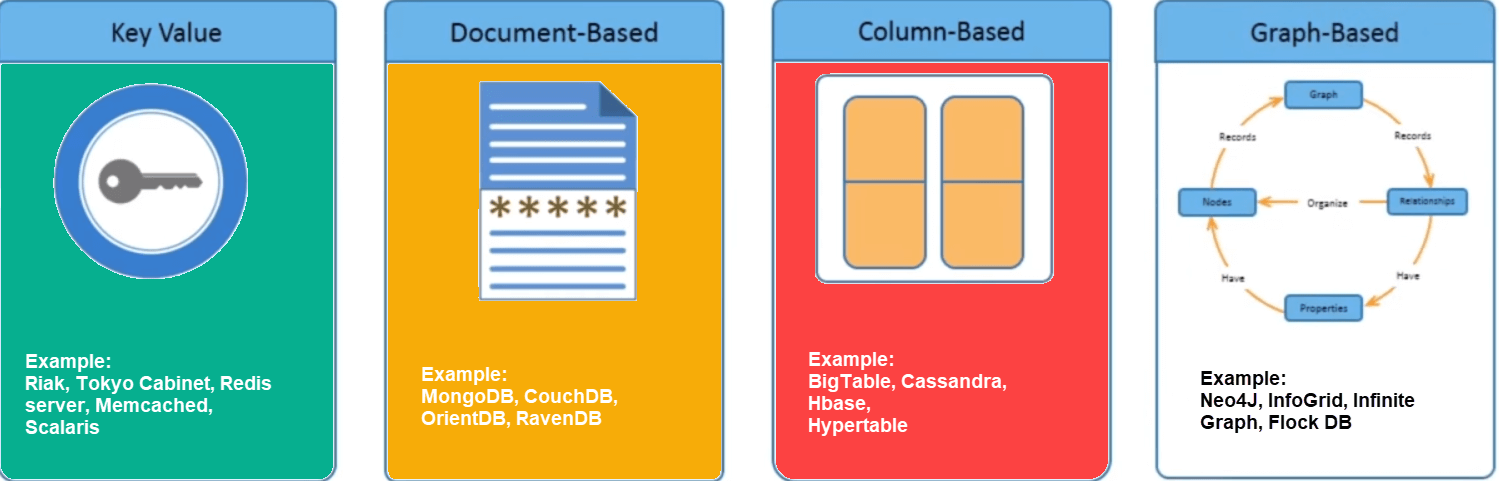
**Distributed**

* Multiple NoSQL databases can be executed in a distributed fashion
* Offers auto-scaling and fail-over capabilities
* Often ACID concept can be sacrificed for scalability and throughput
* Mostly no synchronous replication between distributed nodes Asynchronous Multi-Master Replication, peer-to-peer, HDFS Replication
* Only providing eventual consistency
* Shared Nothing Architecture. This enables less coordination and higher distribution.

[](https://www.guru99.com/images/1/101818_0537_NoSQLTutori4.png)

NoSQL is Shared Nothing.

## Types of NoSQL Databases

[](https://www.guru99.com/images/1/101818_0537_NoSQLTutori5.png)

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. You should select a database based on your product needs.

Let see all of them:

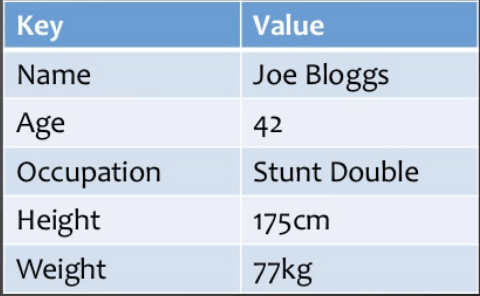
* Key-value Pair Based
* Column-oriented Graph
* Graphs based
* Document-oriented

### Key Value Pair Based

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.

For example, a key-value pair may contain a key like "Website" associated with a value like "Guru99".

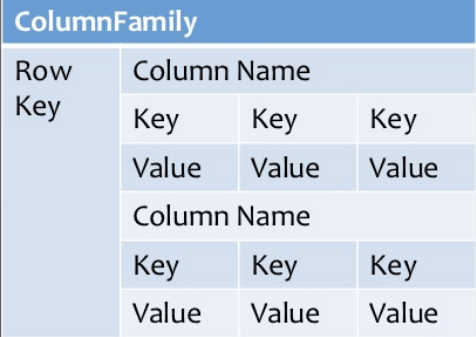
[](https://www.guru99.com/images/1/101818_0537_NoSQLTutori6.png)

It is one of the most basic types of NoSQL databases. This kind of NoSQL database is used as a collection, dictionaries, associative arrays, etc. Key value stores help the developer to store schema-less data. They work best for shopping cart contents.

Redis, Dynamo, Riak are some examples of key-value store DataBases. They are all based on Amazon's Dynamo paper.

### Column-based

Column-oriented databases work on columns and are based on BigTable paper by Google. Every column is treated separately. Values of single column databases are stored contiguously.

[](https://www.guru99.com/images/1/101818_0537_NoSQLTutori7.png)

Column based NoSQL database

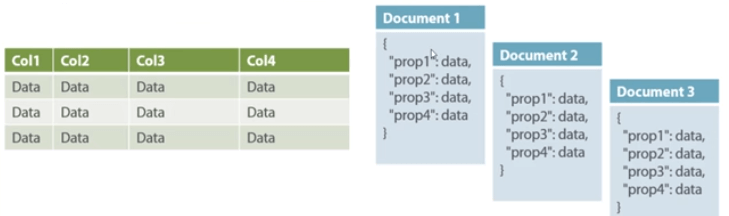
They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column.

Column-based NoSQL databases are widely used to manage data warehouses, business intelligence, CRM, Library card catalogs,

HBase, Cassandra, HBase, Hypertable are examples of column based database.

### Document-Oriented:

Document-Oriented NoSQL DB 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.

[](https://www.guru99.com/images/1/101818_0537_NoSQLTutori8.png)

Relational Vs. Document

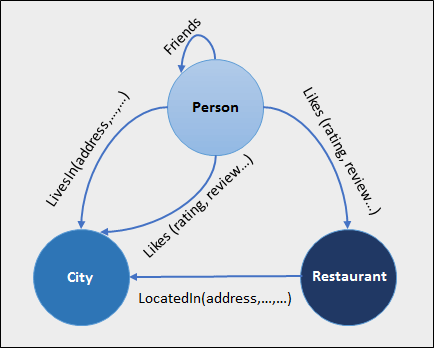
In this diagram on your left you can see we have rows and columns, and in the right, we have a document database which has a similar structure to JSON. Now for the relational database, you have to know what columns you have and so on. However, for a document database, you have data store like JSON object. You do not require to define which make it flexible.

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.

Amazon SimpleDB, CouchDB, MongoDB, Riak, Lotus Notes, MongoDB, are popular Document originated DBMS systems.

### Graph-Based

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.

[](https://www.guru99.com/images/1/101818_0537_NoSQLTutori9.png)

Compared to a relational database where tables are loosely connected, a Graph database is a multi-relational in nature. Traversing relationship is fast as they are already captured into the DB, and there is no need to calculate them.

Graph base database mostly used for social networks, logistics, spatial data.

Neo4J, Infinite Graph, OrientDB, FlockDB are some popular graph-based databases.

## Query Mechanism tools for NoSQL

The most common data retrieval mechanism is the REST-based retrieval of a value based on its key/ID with GET resource

Document store Database offers more difficult queries as they understand the value in a key-value pair. For example, CouchDB allows defining views with MapReduce

## What is the CAP Theorem?

CAP theorem is also called brewer's theorem. It states that is impossible for a distributed data store to offer more than two out of three guarantees

1. Consistency
2. Availability
3. Partition Tolerance

**Consistency:**

The data should remain consistent even after the execution of an operation. This means once data is written, any future read request should contain that data. For example, after updating the order status, all the clients should be able to see the same data.

**Availability:**

The database should always be available and responsive. It should not have any downtime.

**Partition Tolerance:**

Partition Tolerance means that the system should continue to function even if the communication among the servers is not stable. For example, the servers can be partitioned into multiple groups which may not communicate with each other. Here, if part of the database is unavailable, other parts are always unaffected.

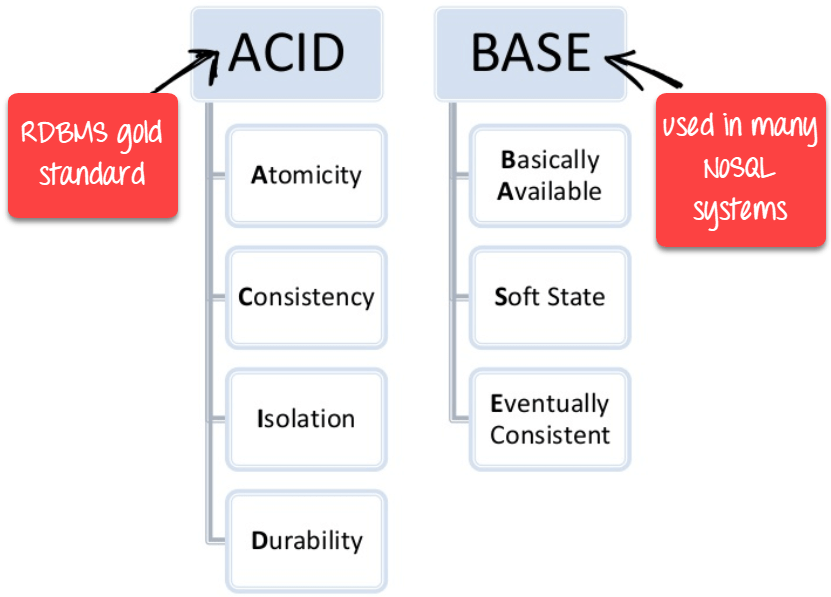
## Eventual Consistency

The term "eventual consistency" means to have copies of data on multiple machines to get high availability and scalability. Thus, changes made to any data item on one machine has to be propagated to other replicas.

Data replication may not be instantaneous as some copies will be updated immediately while others in due course of time. These copies may be mutually, but in due course of time, they become consistent. Hence, the name eventual consistency.

BASE: **B**asically **A**vailable, **S**oft state, **E**ventual consistency

* Basically, available means DB is available all the time as per CAP theorem
* Soft state means even without an input; the system state may change
* Eventual consistency means that the system will become consistent over time

[](https://www.guru99.com/images/1/101818_0537_NoSQLTutori10.png)

## Advantages of NoSQL

* Can be used as Primary or Analytic Data Source
* Big Data Capability
* No Single Point of Failure
* Easy Replication
* No Need for Separate Caching Layer
* It provides fast performance and horizontal scalability.
* Can handle structured, semi-structured, and unstructured data with equal effect
* Object-oriented programming which is easy to use and flexible
* NoSQL databases don't need a dedicated high-performance server
* Support Key Developer Languages and Platforms
* Simple to implement than using RDBMS
* It can serve as the primary data source for online applications.
* Handles big data which manages data velocity, variety, volume, and complexity
* Excels at distributed database and multi-data center operations
* Eliminates the need for a specific caching layer to store data
* Offers a flexible schema design which can easily be altered without downtime or service disruption

## Disadvantages of NoSQL

* No standardization rules
* Limited query capabilities
* RDBMS databases and tools are comparatively mature
* It does not offer any traditional database capabilities, like consistency when multiple transactions are performed simultaneously.
* When the volume of data increases it is difficult to maintain unique values as keys become difficult
* Doesn't work as well with relational data
* The learning curve is stiff for new developers
* Open source options so not so popular for enterprises.

## Summary

* NoSQL is a non-relational DMS, that does not require a fixed schema, avoids joins, and is easy to scale
* The concept of NoSQL databases became popular with Internet giants like Google, Facebook, Amazon, etc. who deal with huge volumes of data
* In the year 1998- Carlo Strozzi use the term NoSQL for his lightweight, open-source relational database
* NoSQL databases never follow the relational model it is either schema-free or has relaxed schemas
* Four types of NoSQL Database are 1).Key-value Pair Based 2).Column-oriented Graph 3). Graphs based 4).Document-oriented
* NOSQL can handle structured, semi-structured, and unstructured data with equal effect
* CAP theorem consists of three words Consistency, Availability, and Partition Tolerance
* BASE stands for **B**asically **A**vailable, **S**oft state, **E**ventual consistency
* The term "eventual consistency" means to have copies of data on multiple machines to get high availability and scalability
* NOSQL offer limited query capabilities

v