**MongoDb**

[**https://beginnersbook.com/2017/09/mongodb-tutorial/**](https://beginnersbook.com/2017/09/mongodb-tutorial/)

[**https://www.guru99.com/nosql-tutorial.html**](https://www.guru99.com/nosql-tutorial.html)

MongoDB is a NoSQL database. There are different types of NoSQL databases, so to be specific MongoDB is an open-source document-based NoSQL database.

[**MongoDB**](https://intellipaat.com/blog/tutorial/mongodb-tutorial/introduction-to-mongodb/) is an open-source, cross-platform, document-oriented database system which is written in C++ programming language. MongoDB database allows several functioning such as insert, update, delete query, projection, sort () and limit () methods, create and drop collection and many to list.

MongoDB is a document-oriented NoSQL database mainly used for high volume data storage.

MongoDB is a database which came into existence around the mid-year of the 2000s. It falls under the category of a NoSQL database.

Despite having a similar functionality with the other NoSQL technologies, MongoDB has created a distinctive place for itself. How and why? Read below :

* Suitable for real-time analytics and processing massive amounts of queries
* Faster access to data as it uses internal memory for storing data
* Embedded documents and arrays reduce need to use expensive joins
* Ability to run over multiple servers
* The load is balanced automatically by keeping data in shards
* Failures are easily administered during failures.

## MongoDB versus MySQL

Over a decade MySQL has enjoyed its supremacy as it was an inexpensive and widely used relational database management system for storing the structured data. However, with the advent of the digital age, the data has not remained limited to a single type but can be in any form and in massive amounts. MongoDB is one of the new age databases that solve some of the conventional issues.

How is MongoDB better than MySQL? Let’s compare these two technologies :

|  |  |  |
| --- | --- | --- |
| Criteria | MongoDB | MySQL |
| Data Models | Many alternative data models | Relational database management system |
| Architecture | Schemaless | Schema |
| Operations | Supports atomicity | A transactional relation datastore |
| Query Language | JSON query | SQL |
| Performance Enhancement by | Server-side scripting | Stored procedures implemented |
| Scalability | Highly scale | Not Scalable |
| Speed | Faster | Slower |

## What is NoSQL?

NoSQL is a non-relational DMS, that does not require a fixed schema, avoids joins, and is easy to scale. The purpose of using a NoSQL database is 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 collect terabytes of user data every single day.

NoSQL database stands for "Not Only SQL" or "Not SQL."

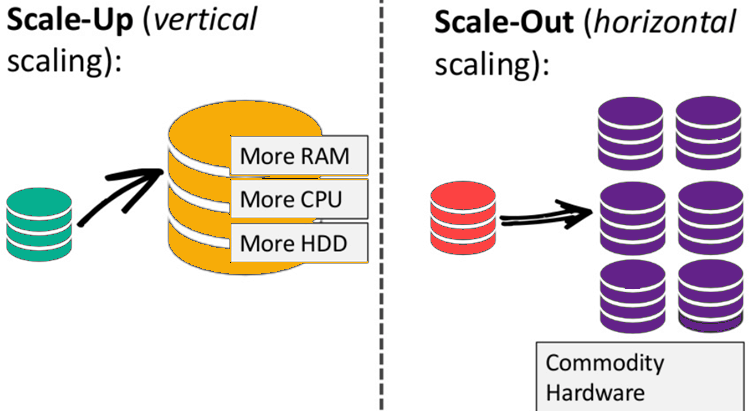
Traditional RDBMS uses SQL syntax to store and retrieve data for further insights. Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data.

## 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.

## 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

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.

## 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.

## **MongoDB Features**

## Each database contains collections which in turn contain documents. Each document can be different with a varying number of fields. The size and content of each document can be different from each other.

## The document structure is more in line with how developers construct their classes and objects in their respective programming languages. Developers will often say that their classes are not rows and columns but have a clear structure with key-value pairs.

## As seen in the introduction with NoSQL databases, the rows (or documents as called in MongoDB) don’t need to have a schema defined beforehand. Instead, the fields can be created on the fly.

## The data model available within MongoDB allows you to represent hierarchical relationships, to store arrays, and other more complex structures more easily.

## Scalability – The MongoDB environments are very scalable. Companies across the world have defined clusters with some of them running 100+ nodes with around millions of documents within the database

## **Why Use MongoDB?**

## Below are the few of the reasons as to why one should start using MongoDB

## Document-oriented – Since MongoDB is a NoSQL type database, instead of having data in a relational type format, it stores the data in documents. This makes MongoDB very flexible and adaptable to real business world situations and requirements.

## Ad hoc queries - MongoDB supports search by field, range queries, and regular expression searches. Queries can be made to return specific fields within documents.

## Indexing - Indexes can be created to improve the performance of searches within MongoDB. Any field in a MongoDB document can be indexed.

## Replication - MongoDB can provide high availability with replica sets. A replica set consists of two or more mongo DB instances. Each replica set member may act in the role of the primary or secondary replica at any time. The primary replica is the main server that interacts with the client and performs all the read/write operations. The Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically switches over to the secondary and then it becomes the primary server.

## Load balancing - MongoDB uses the concept of sharding to scale horizontally by splitting data across multiple MongoDB instances. MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure.

## **Difference between MongoDB & RDBMS**

## Below are some of the key term differences between MongoDB and RDBMS

|  |  |  |
| --- | --- | --- |
| RDBMS | MongoDB | Difference |
| Table | Collection | In RDBMS, the table contains the columns and rows which are used to store the data whereas, in MongoDB, this same structure is known as a collection. The collection contains documents which in turn contain Fields, which in turn are key-value pairs. |
| Row | Document | In RDBMS, the row represents a single, implicitly structured data item in a table. In MongoDB, the data is stored in documents. |
| Column | Field | In RDBMS, the column denotes a set of data values. These in MongoDB are known as Fields. |
| Joins | Embedded documents | In RDBMS, data is sometimes spread across various tables and in order to show a complete view of all data, a join is sometimes formed across tables to get the data. In MongoDB, the data is normally stored in a single collection, but separated by using Embedded documents. So there is no concept of joins in MongoDB. |

## Apart from the differences in the terms, a few other differences are shown below

## Relational databases are known for enforcing data integrity. This is not an explicit requirement in MongoDB.

## RDBMS requires that data be [normalized](https://www.guru99.com/database-normalization.html) first so that it can prevent orphan records and duplicates Normalizing data then has the requirement of more tables, which will then result in more table joins, thus requiring more keys and indexes. As databases start to grow, performance can start becoming an issue. Again this is not an explicit requirement in MongoDB. MongoDB is flexible and does not need the data to be normalized first.

## 

## 

## 

## 

## **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
* The BASE stands for Basically Available, Soft state, Eventual 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

**Horizontal vs Vertical Scaling:**

[**https://github.com/vaquarkhan/vaquarkhan/wiki/Difference-between-scaling-horizontally-and-vertically**](https://github.com/vaquarkhan/vaquarkhan/wiki/Difference-between-scaling-horizontally-and-vertically)

[**https://stackoverflow.com/questions/11707879/difference-between-scaling-horizontally-and-vertically-for-databases**](https://stackoverflow.com/questions/11707879/difference-between-scaling-horizontally-and-vertically-for-databases)