MongoDB

https://www.mongodb.com/try/download/shell

OLTP: OLTP stands for Online Transaction Processing. It refers to a type of computer system that is designed for processing transactions in real-time. These transactions could be any type of business activity, such as sales, orders, or inventory management.

OLTP systems are typically used in business environments where fast response times and data accuracy are critical.

Some common examples of OLTP systems include banking and financial transaction processing systems, airline reservation systems, and e-commerce websites. In these systems, data is typically stored in a relational database, and transactions are processed using SQL (Structured Query Language) queries.

The data access pattern for OLTP is to retrieve a small set of data, usually by doing a lookup on an ID.

Transaction support is arguably the most important characteristic that OLTP offers, as reflected in the name itself. A database transaction means a set of actions that are either all completed, or none of them are completed; there is no middle ground. For example, an ATM has to guarantee that it either gave the customer the money and debited their account, or did not give the customer money and did not debit their account. Only giving the customer money, but not debiting their account, harms the bank; only debiting the account, but not giving the customer money, harms the customer.

OLAP: OLAP stands for Online Analytical Processing. It refers to a type of computer system that is designed for analyzing and querying large volumes of data in real-time. OLAP systems are typically used in business environments where decision-making and analysis are critical.

Unlike OLTP systems, which are optimized for transaction processing, OLAP systems are optimized for querying and analysis of data. OLAP systems are designed to handle complex queries and generate reports quickly, even when dealing with large amounts of data. They provide advanced analytical capabilities, including multi-dimensional analysis, data mining, and predictive analytics.

Some common examples of OLAP systems include business intelligence tools, such as Tableau and Power BI, which enable users to analyze and visualize data in a variety of ways.

NewSQL is a term that’s used to describe product offerings that support the relational data model while delivering the same scalable performance of NoSQL database systems.

### What are the Concepts of NoSQL?

The concepts are listed below:

#### Lack of schema

Support for structured, semi-structured, and unstructured data. No specific schema require to define before entering data into its databases. New fields can be added and also supports nested data implementation and retrieval. Developers can use data type and query options requisite for the specific application resulting in faster development. Faster Development time is considerably reduced due to no complex SQL queries or join statements.

#### Auto Balancing

Division of data among multiple servers automatically, with no assistance required from applications.

#### Integrated Caching

Its database cache data in system memory to increase data throughput and increase the performance in advance. High scalability, reliability with a simple data model and simple query language.

#### The BASE Principle For Transaction

The base is to NoSql as what ACID is to SQL. It ensures that NoSQL databases ensure their reliability despite the loss of consistency. The base stands for Basically Available Soft-state, Eventually consistent. Eventually consistent – System can become eventually consistent, information is updated wherever necessary.

**What are the Concepts of SQL Relational Databases?**

the Concepts of SQL Relational Databases below:

**ACID**

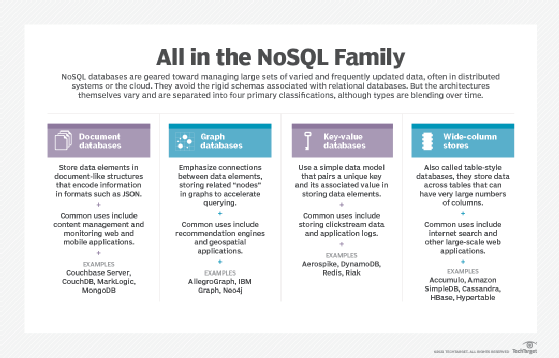
Atomicity, Consistency, Isolation, Durability to maintain the reliability of transactions.

1. Atomicity – completion of the transaction as a whole or none at all
2. Consistency – assures the stable state of the database with or without changes
3. Isolation – multiple transactions do not interfere with each other
4. Durability – permanent effect on the database by the changes

**Scalability**

The capability of the database to handle growing amounts of data. Vertical scaling helps to enhance the existing capacity of the database server. Most SQL databases support vertical scaling. They can, however, scale-up, not scale out.

NoSQL



#### Examples of NoSQL database systems

There is a plethora of popular NoSQL database systems used for production workloads today, including:

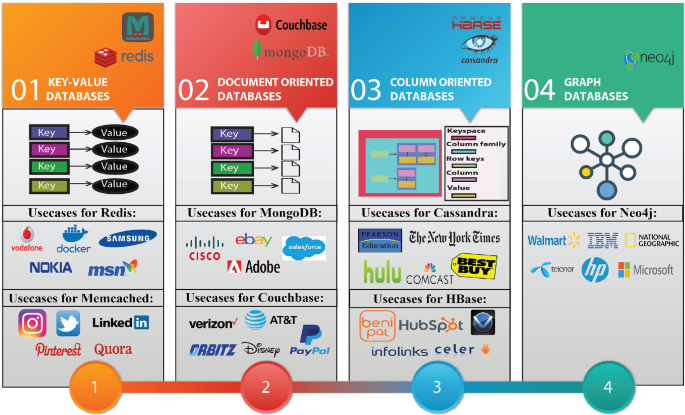
* MongoDB, a document database
* Redis, a key/value database
* Cassandra, a wide-column database
* Neo4j, a graph database

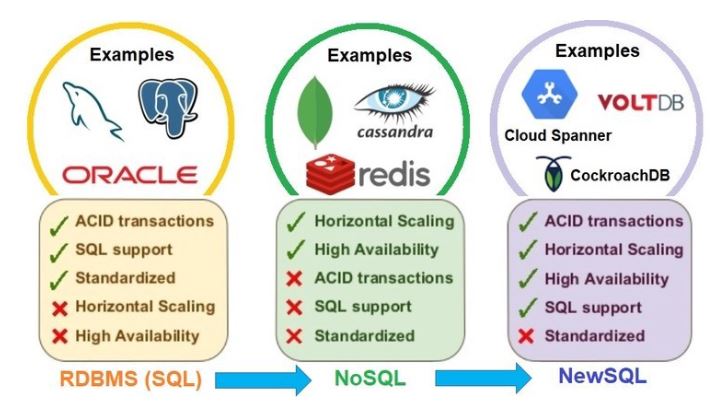
#### Examples of NewSQL database systems

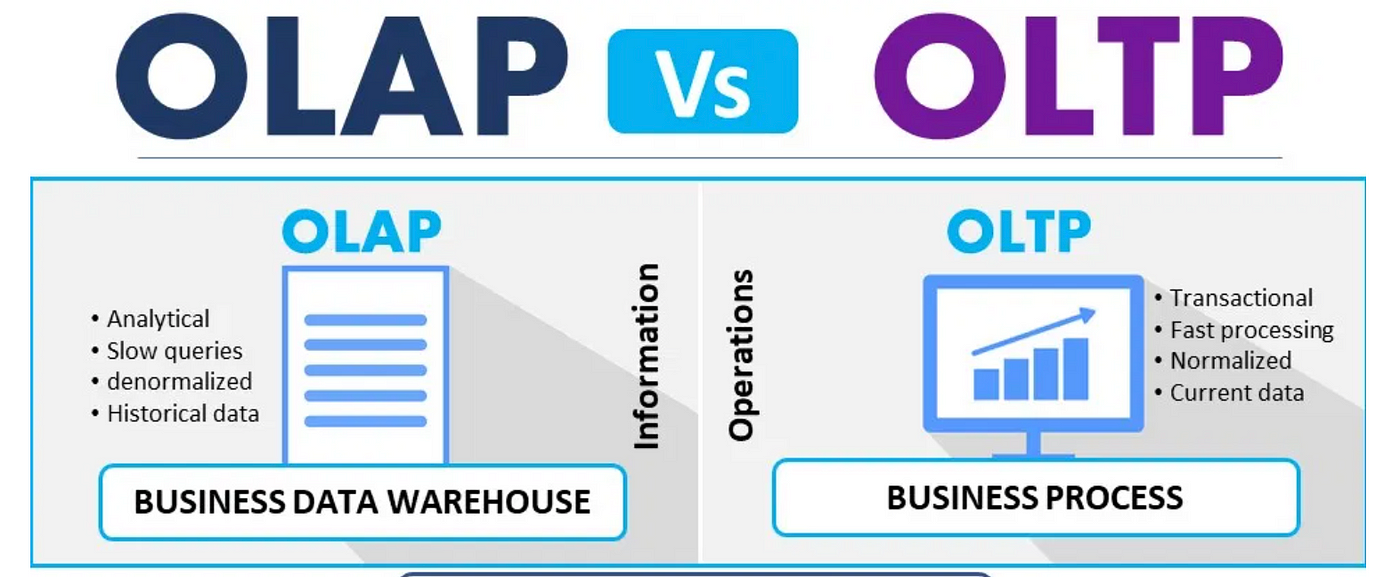
An increasing number of NewSQL DBMS offerings are being implemented for production workloads, including:

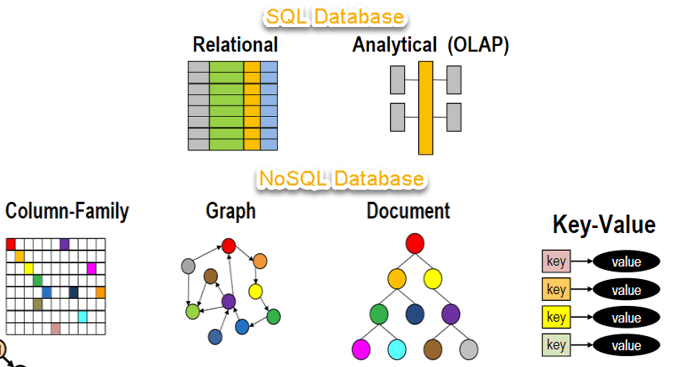
* CockroachDB
* VoltDB
* NuoDB
* SingleStore
* Clustrix

https://www.techtarget.com/whatis/feature/SQL-vs-NoSQL-vs-NewSQL-How-do-they-compare









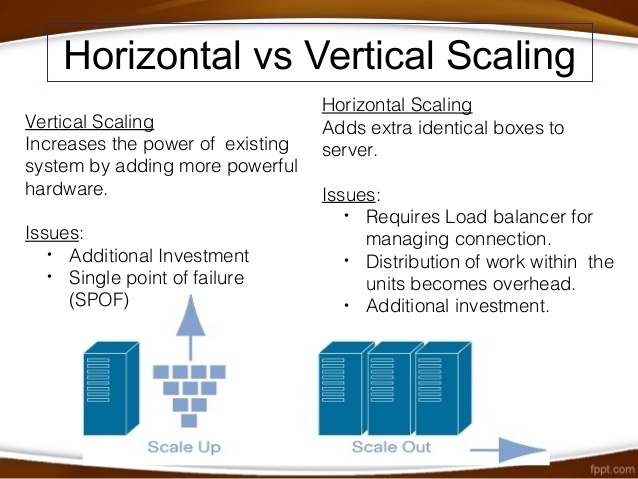
BASE

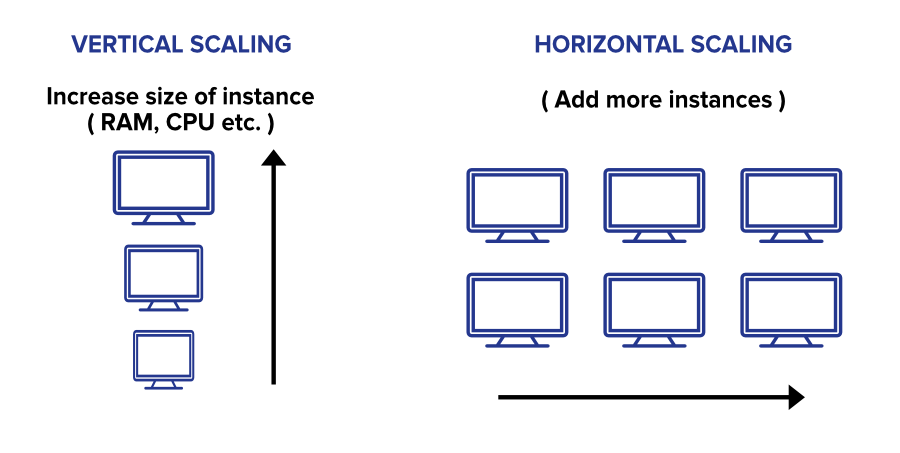
The term BASE is an acronym that stands for "Basically Available, Soft state, Eventually consistent". It is a set of properties that describe a class of distributed systems that prioritize availability and partition tolerance over strict consistency.

MongoDB, being a NoSQL database, is designed to follow the BASE properties. Specifically, MongoDB is considered to be "Basically Available" because it is designed to be highly available and provide fast response times, even in the face of network partitions or hardware failures.

MongoDB is also "Soft state," which means that the state of the system may change over time due to factors such as data replication or network partitions. However, MongoDB is designed to eventually converge to a consistent state, making it "Eventually consistent."

In practice, this means that MongoDB can provide a highly available and fault-tolerant database system, but at the cost of eventual consistency. This is often acceptable in use cases where real-time consistency is not critical, such as in certain types of web applications or big data analytics.





Horizontal vs vertical scaling

Horizontal scaling, also known as scaling out, involves adding more machines or nodes to a system to distribute the workload across multiple machines. In the context of a database, this means sharding the data across multiple servers. Horizontal scaling can improve fault tolerance and availability because if one server fails, the others can still continue to operate. It can also improve performance by distributing the workload across multiple machines, which can increase throughput.

Vertical scaling, also known as scaling up, involves adding more resources, such as CPU, memory, or storage, to a single machine to increase its capacity. This is typically done by upgrading the hardware, such as replacing the CPU or adding more RAM. Vertical scaling can be more expensive than horizontal scaling because it requires purchasing more powerful hardware, but it can be simpler to implement because it does not require distributing the workload across multiple machines.

In summary, horizontal scaling involves adding more machines to a system, while vertical scaling involves adding more resources to a single machine.

Vertical scaling is useful in several scenarios, including:

Limited budget: If the budget is limited, it may be more cost-effective to upgrade the hardware of a single machine rather than purchasing multiple machines.

Low to moderate workload: If the workload is low to moderate, upgrading the hardware of a single machine may be sufficient to handle the load.

Single application: If the workload is primarily driven by a single application, vertical scaling can be an effective way to improve performance of that application.

Simpler administration: Vertical scaling can be simpler to administer because it involves managing a single machine rather than multiple machines.

Limited space: In some environments, such as data centers, there may be limited space for additional machines. In these cases, vertical scaling can be a better option.

However, there are also some limitations to vertical scaling. One of the biggest limitations is that it is not infinitely scalable. Eventually, a single machine will reach its limits in terms of hardware capacity, and it will be necessary to add more machines to handle the load.

Advantages of Nosql

NoSQL databases offer several advantages in terms of velocity, volume, variability, and veracity compared to traditional relational databases:

1. Velocity: NoSQL databases are designed to handle high-velocity data streams, such as real-time data from IoT devices or social media platforms. They can handle high read and write loads, providing fast response times for queries.
2. Volume: NoSQL databases are designed to handle large volumes of data, from terabytes to petabytes or more. They can scale horizontally across multiple nodes, allowing for near-unlimited storage capacity.
3. Variability: NoSQL databases are designed to handle semi-structured and unstructured data, such as JSON, XML, or binary data. This flexibility allows for more complex and diverse data models.
4. Veracity: NoSQL databases are designed to handle data with varying levels of accuracy and completeness, including data that may be missing or inconsistent. They can handle data quality issues and still provide valuable insights.

MongoImport

https://www.mongodb.com/try/download/database-tools

Unfortunately, you have to download MongoDB tools separately from [mongodb website](https://www.mongodb.com/try/download/database-tools).

***.zip***

By default, the .zip option is enabled. After downloading the MongoDB tools and extracting them, open the bin folder and copy it. Then go to the bin directory of MongoDB (C:\Program Files\MongoDB\Server\<your MongoDB version>\bin) and paste them in it.

***.msi***

If you downloaded .msi instead of the .zip file, after the installation is completed, go to the C:\Program Files\MongoDB directory. You can see a folder with the Tools name right side of the Server folder. You should go to the bin folder of the Tools and copy all files in it, go back to the Server folder, open the bin folder and paste all of them on it right side of the mongod file.

open cmd from mongodb server/bin and write

mongoimport --version

mongoimport --uri mongodb+srv://sudhamangla:ifbAgvkraul4n6IC@cluster0.mgrqz5r.mongodb.net/mydb --collection products --jsonArray --file C:\All\_Works\Sudha\SimpliLearn\MEAN\_Phase3\_April\_Original\MongoDB\Atlas\products.json

MVC Application

npm install express mongoose body-parser

secure the password

npm install dotenv

<https://medium.com/geekculture/set-up-mongodb-sharded-cluster-for-windows-with-just-a-double-click-6eedbb7b79e>

<https://github.com/arun2pratap/mongodbClusterForWindowsOneClick>

<https://adelachao.medium.com/create-a-mongodb-replica-set-in-windows-edeab1c85894>

Indexes

Note that creating a compound index can be more efficient than creating separate single-field indexes, because it allows MongoDB to use the index for queries that involve both fields. However, be aware that compound indexes can have trade-offs in terms of storage space and write performance, especially for collections with a large number of documents or high write throughput.

Back

>mongod --config mongod-1.cfg

mongod --config mongod-2.cfg

cfg = rs.conf()

rs.status()

rs.initiate()

db.hello() -   
Displays information about this member's role in the replica set, including whether it is the primary.

show databases

use mydb

db.collection.insertOne({name:'sudha'});rs

mongosh --port 27019

db.getMongo().setReadPref('secondary')

db.getMongo().setReadPref('secondaryPreferred')

|  |
| --- |
| cfg = rs.conf(); |
| cfg.members[1].priority = 2; |
| rs.reconfig(cfg); |

[rs.printReplicationInfo()](https://www.mongodb.com/docs/manual/reference/method/rs.printReplicationInfo/#mongodb-method-rs.printReplicationInfo)

Prints a formatted report of the replica set status from the perspective of the primary.

rs.addArb("localhost:27030")

In a MongoDB replica set, an arbiter node is a special type of member that does not store any data, but participates in the replica set election process and can cast a tie-breaking vote. The purpose of an arbiter node is to ensure that the replica set can still function in the event of a network partition or other failure that prevents a majority of the replica set members from communicating with each other.