# Persistence

Much larger volumes of data are captured in modern computing, and an effort to offer scalable solutions. We examine relational, key/value, document, columnar, and graph databases. The last four types are collectively called NoSQL. Databases are now distributed in order to handle this increased volume of data and resulting requests. We provide an overview of the current technology.

We discuss these topics:

* Differences between the various database systems
* The CAP theorem – an observation about characteristics of distributed computing
* Characteristics of NoSQL database

# Database Characteristics

RDBMS systems tend to store structured data, meaning the data has a schema, and offer ACID properties (Atomic, Consistent, Isolated, and Durable.) They offer excellent transactional capabilities with guaranteed preservation of order of updates and consistency of reads. Providing these capabilities requires significant overhead and present challenges to partitioning.

RDBMS with SQL has several very big advantages:

* Strong mathematical basis.
* Declarative syntax with a well-known language in Structured Query Language (SQL).
* Provides indexing to accelerate ad hoc queries

NoSQL databases primarily address large-scale data storage and unstructured data. The definition from NoSQL.org is:

*NOSQL* is the next Generation Database Management Systems mostly addressing some of the points: being non-relational, distributed, open-source and horizontally scalable.

## Some NoSQL Systems

# CAP Summary

Modern distributed persistence is required to manage the large volume of data generated by 21rst century computing. Distributed computing forces network partitioning; and that adds failure modes and latency compared to single machine implementations.

CAP Theorem: A distributed database system can have at most 2 of 3 properties: Consistency, Availability and Partition Tolerance.

1. ***Consistency***: This condition states that all nodes see the same data at the same time.
2. ***Availability***: This condition states that every request gets a response on success/failure. Achieving availability in a distributed system requires that the system remains operational 100% of the time.
3. ***Partition Tolerance:*** This condition states that the system continues to run, despite the number of messages being delayed by the network between nodes. A system that is partition-tolerant can sustain any amount of network failure that doesn’t result in a failure of the entire network.

## From Wiki

No distributed system is safe from network failures, thus network partitioning generally has to be tolerated. In the presence of a partition, one is then left with two options: consistency or availability. When choosing consistency over availability, the system will return an error or a time-out if particular information cannot be guaranteed to be up to date due to network partitioning. When choosing availability over consistency, the system will always process the query and try to return the most recent available version of the information, even if it cannot guarantee it is up to date due to network partitioning.

In the absence of network failure – that is, when the distributed system is running normally – both availability and consistency can be satisfied.

CAP is frequently misunderstood as if one has to choose to abandon one of the three guarantees at all times. In fact, the choice is really between consistency and availability only when a network partition or failure happens; at all other times, no trade-off has to be made.[5][6]

Database systems designed with traditional ACID guarantees in mind such as RDBMS choose consistency over availability, whereas systems designed around the BASE philosophy, common in the NoSQL movement for example, choose availability over consistency.[7]

The PACELC theorem builds on CAP by stating that even in the absence of partitioning, another trade-off between latency and consistency occurs.

## From Book #1 – NoSQL Overview

Persistence systems may be categorized as: relational, key/value, document, columnar, and graph. The last four types are collectively called NoSQL. It is ironic that all NoSQL databases are developing SQL-like interfaces to leverage the wide-spread experience of data users with SQL.

NoSQL database Characteristics:

* Consistency versus availability
* ACID guarantees
* Hash versus range partition
* In-place updates versus appends
* Row versus column versus column-family storage models
* Strongly versus loosely enforced schemas

Amazon Dynamo uses Eventual Consistency to come close to get all three properties. The paper Dynamo: Amazon’s Highly Available Key-value Store is worth reading when learning about NoSQL databases and distributed systems.

# References

### Books

1. Seven NoSQL Databases in a Week ()

Links:

1. ***Consistency*** is a little complex, but a technical article on Wiki is <https://en.wikipedia.org/wiki/Eventual_consistency>.
2. ***Wiki CAP***: <https://en.wikipedia.org/wiki/CAP_theorem>.
3. Amazon’s Dynamo DB: <https://www.allthingsdistributed.com/2007/10/amazons_dynamo.html>.
4. ***NoSQL vs. RDBMS: Apples and Oranges?***: <https://it.toolbox.com/blogs/madgreek/nosql-vs-rdbms-apples-and-oranges-032810>.
5. ***Your Ultimate Guide to the Non-Relational Universe!***: <http://nosql-database.org/>.
6. Cloud NoSQL DB comparison: <https://searchcloudcomputing.techtarget.com/tip/Compare-NoSQL-database-types-in-the-cloud>.
7. AWS NoSQL description: <https://aws.amazon.com/nosql/>.
8. A great NoSQL overview: <https://medium.com/@Grigorkh/nosql-in-the-cloud-a-scalable-alternative-to-relational-databases-f5dc35555fdb>.
9. ***Good SQL vs NoSQL compariso***n: <https://blog.panoply.io/sql-or-nosql-that-is-the-question>.
10. ***Index usage in SQL vs NoSQL***: <https://sql-vs-nosql.blogspot.com/2013/11/indexes-comparison-mongodb-vs-mssqlserver.html>.

## CAP tutorials:

1. Overview: <https://towardsdatascience.com/cap-theorem-and-distributed-database-management-systems-5c2be977950e>.