medium.com

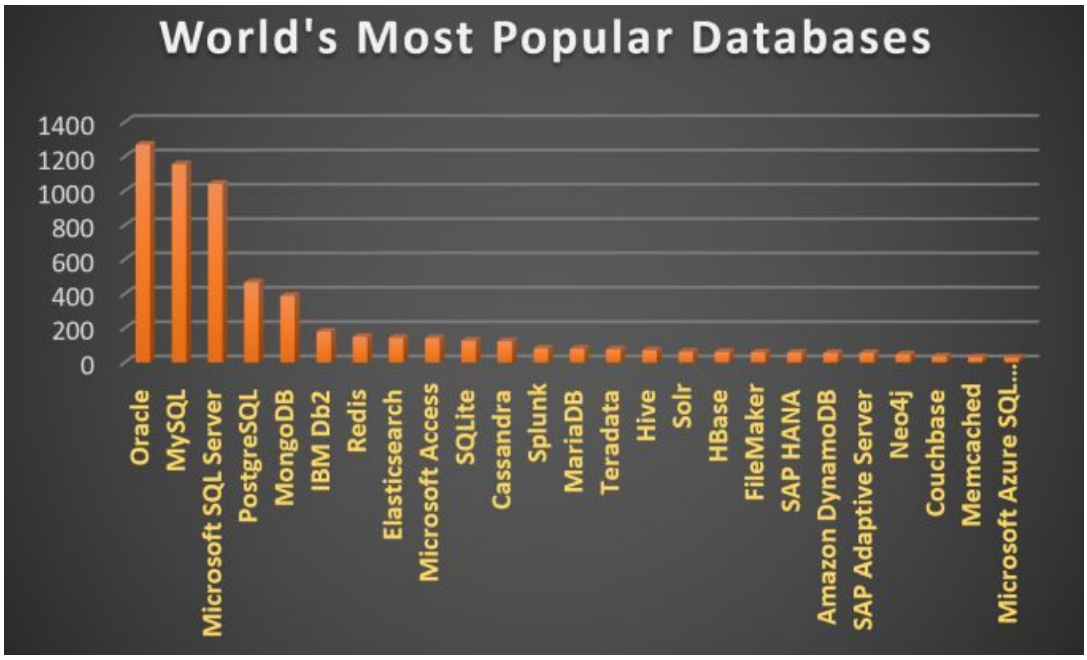
Significance of ACID vs BASE vs CAP Philosophy in Data Science

Shrey Shivam

7-9 minutes

There are 3 popular types of properties or theorems are associated with various database systems that are characterised by different properties based on these attributes. More often than not, data scientists and ML Engineers tend to overlook the importance of considering these properties while choosing an appropriate database for the required business use case.

Today, some of the most popular database storage systems are as below. They can be further classified into SQL vs No SQL and Row oriented vs Column oriented.



source: https://www.c-sharpcorner.com/article/what-is-the-most-

popular-database-in-the-world/

It is important to understand that what makes up for the existence of so many databases that are provide different characteristics. For example, Oracle is known for being a traditional OLTP database and Hive is a OLAP database. Similarly, Cassandra is optimised for O(1) read performance but MySQL is designed for O(log(n)).

What is ACID?

ACID is a set of properties of a database transaction with an intention to guarantee validity of data.

Atomicity: Multiple (SQL) statements are often composed into 1 transaction. In real world, one such example of a transaction could be a bank transfer where a debit from account A of ABC bank and credit into account X. A database that provides atomicity guarantees that either the entire transaction fails or succeeds as 1 single unit. This is also applicable in the case of power failures, crashes or any errors. In the case of ABC bank, either the money is debited and credited in respective amounts, or the operation fails keeping the database in a consistent state.

Consistency: Databases are governed by various defined rules such as relationship between 2 different entities (tables), primary key & foreign key constraints, various triggers and integrity constraints including checks on columns. Databases providing consistency ensure that it never goes into an inconsistent state and does not allow database to get corrupted due to an illegal transaction. For example, in a typical school database, a primary key-foreign key referential integrity constraint on student and course database ensures that every student is enrolled into at least 1 course and it is not possible to have a student record without a course record.

Isolation: In practicality, transactions happen concurrently. Multiple

users use the system with possibilities of interacting with same records. A very basic (and not the best) example could be 2 family members making a simultaneous; one through online net banking and the other using debit card transaction over the same underlying bank account. In a macro setting, this can be thought up as multiple transactions being written into the same table at the same time. Isolation is a property of databases that guarantees database to be in the same states with concurrency control as if it would have been done sequentially.

Durability: Availability of a committed transaction even after power failure or crash is ensured by recording all the completed transactions into a non volatile memory.

RDBMS such as MySQL, Oracle, SQL Server etc provide ACID compliance and hence are the obvious choice in financial systems use cases where these properties are required.

What is BASE?

In chemistry, Base is opposite to ACID. Even in the case of database systems, BASE is a set of properties of databases that are differently than ACID. In layman terms, BASE is a loose version of ACID characteristics and provide several advantages over ACID compliant database which are distributed systems by design and cannot guarantee safety from network failures.

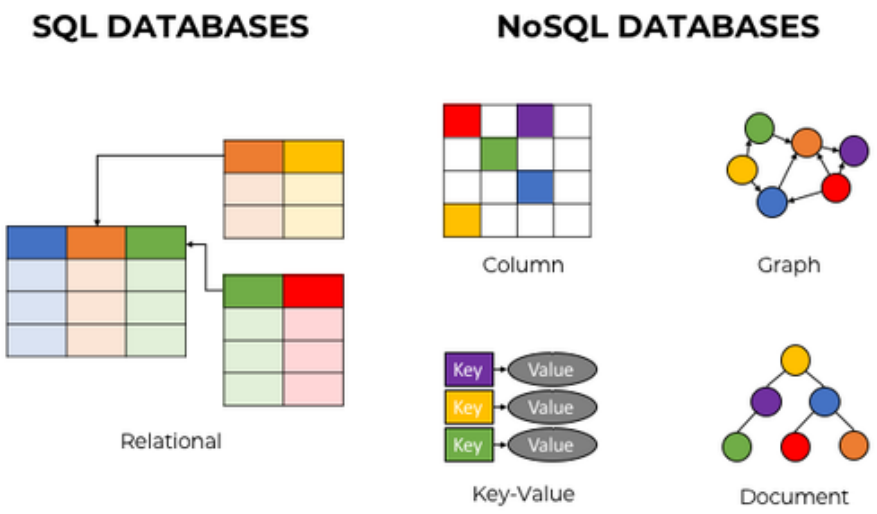
Basically Available : Basic Read and Write functionality is provided without any consistency guarantee. Hence there is a possibility of writes not getting persisted or read may not be providing latest data.

Soft State : The state of the system can be known after some time when it has converged but without consistency guarantees.

Eventually Consistent : In the case of a fully functional system, after a waiting time, we can get to the desired consistency state for

the given set of inputs enabling any further read operation to be consistent with desired expectations.

BASE properties gave birth to No SQL based databases that are not designed on Relational set theory of tables with rows and columns



DynamoDB and Cassandra are popular Key-Value stores, HBase is a popular column oriented database, Neo4J is a graph database and MongDB and Solr are popular Document databases.

BASE properties allow these databases to be distributed in nature and provides high scalability, fast read-write performance, easy replication and big data analytics capability.

SQL systems are vertically scalable where as NoSQL scale horizontally. ACID complaint traditional SQL based systems have static pre-defined schema where as most of the No SQL systems have dynamic schema.

In data science world, various distributed No SQL systems are used so that they can provide real time analytics (Cassandra), social network analysis (Neo4J), gaming and streaming media analytics (DynamoDB).

BASE does not mean No SQL systems lose messages or are not

reliable. They go by the principle that in a worst case scenario, they may not be able to guarantee all of the properties provided by a traditional database that are characterised by ACID compliance. Keep in mind that the design of these systems are also different from RDBMS so that high infrastructure availability and required processing and memory are available. In data science world, for use cases such as recommendation or network analysis, a single individual individual record or document does not provide any predictive power that is provided by aggregated or summarised data. For example, one click or 1 impression on amazon site does not impact the score or profile of the user as much as their action in the entire session or in the last 7 days. No SQL systems provide such capabilities.

What is CAP Theorem?

CAP Theorem states that in a case of a network failure in a distributed database systems, it is impossible to maintain both availability and consistency.

Consistency: Every read operation results in up to date information from database

Availability: Without guaranteeing that it has the most recent write, it provides a non error response

Partition Tolerance: Even after a few messages lost between nodes, or nodes not functional, system continues to be operational.

CAP Theorem says that in case of a network issue (one or more nodes not available, down etc), either the entire request operation is cancelled to maintain consistency and thus risking availability or the request proceeds to maintain availability and thus risking inconsistency.

It is important for data scientists and ML Engineers to understand these properties. Each of these database do their basic job well. It

is when the complexity (heavy read vs light write, heavy read & heavy write, high volume storage, vs joins, ad-hoc query, filtered query, schema changes etc) of the underlying data changes is when we start seeing issues.

For example, Cassandra is a very good database for high throughput read and write both but unless you design your data model in the right way (minimising partitions read, knowing which queries to support — grouping, filtering), it may provide miserably owing to our lack of ability to use it in an optimised way. You will be surprised to know that selection of HBase key and Row Key is the foundation of its table design which isn't a case with RDBMS.