The CAP theorem

In a distributed system, it cannot be guaranteed simultaneously:

- **C**onsistency: Any read receives the latest write or an error as a response
- **A**vailability: database requests always receive a response but without the guarantee that it contains the most recent writing
- **P**artition Tolerance: a network fault doesn't prevent messaging between nodes.

The distributed systems are always necessarily partition tolerant, this means there is always going to be a trade-off between consistency and availability

When we choose consistency over availability, the system will return an error or a timeout if particular information cannot be guaranteed to be up to date due to network partitioning.
When we choose availability over consistency, the system will always process the query and try to return the latest available version of the information, even if it cannot guarantee that it is up to date due to network partitioning.

Example where A is sacrificed (CP): In this case we prioritize **consistency.**
When you need multiple clients to have the same view of the data. For example, financial information, personal information. In those cases we need a database that gives you confidence that the data you are looking for is up to date. An example could be the website of a bank where customers access their bank accounts. The information in this case has to be updated and if it is not, it is better not to show it. Example: **MongoDB**

Example where C is sacrificed (AP): In this case we prioritize **availability** and it isn't critical that the database is constantly up to date.
An example would be Airbnb comments. At times it may happen that the number of comments that you see on the Airbnb apartment is 10, but when you actually open the link you see 20. This inconsistent behavior is OK for such systems. It's not a big concern that Airbnb isn't showing the correct number of comments. Example: **Cassandra**