NoSQL

Quick Introduction

Alberto Rossotto

03 / March / 2022

SQL

To understand NoSQL, we need to understand SQL

SQL databases:

- Scale well vertically
- Implement ACID transactions (Atomicity, Consistency, Isolation, Durability)
- Have a standardised query language
- Have a clearly defined schema
- Organise information in tables connected by relationships

NoSQL

Origin

Tech giants (Google, Amazon,…) hit the limits of SQL databases

- Vertical scalability was not enough: better to have horizontal scalability with many small boxes
- ACID transactions were too slow: availability (=speed) was preferred
- A standardised query language was not a requirement for custom made products
- Tables were a limit: easier to ingest the data and process it later

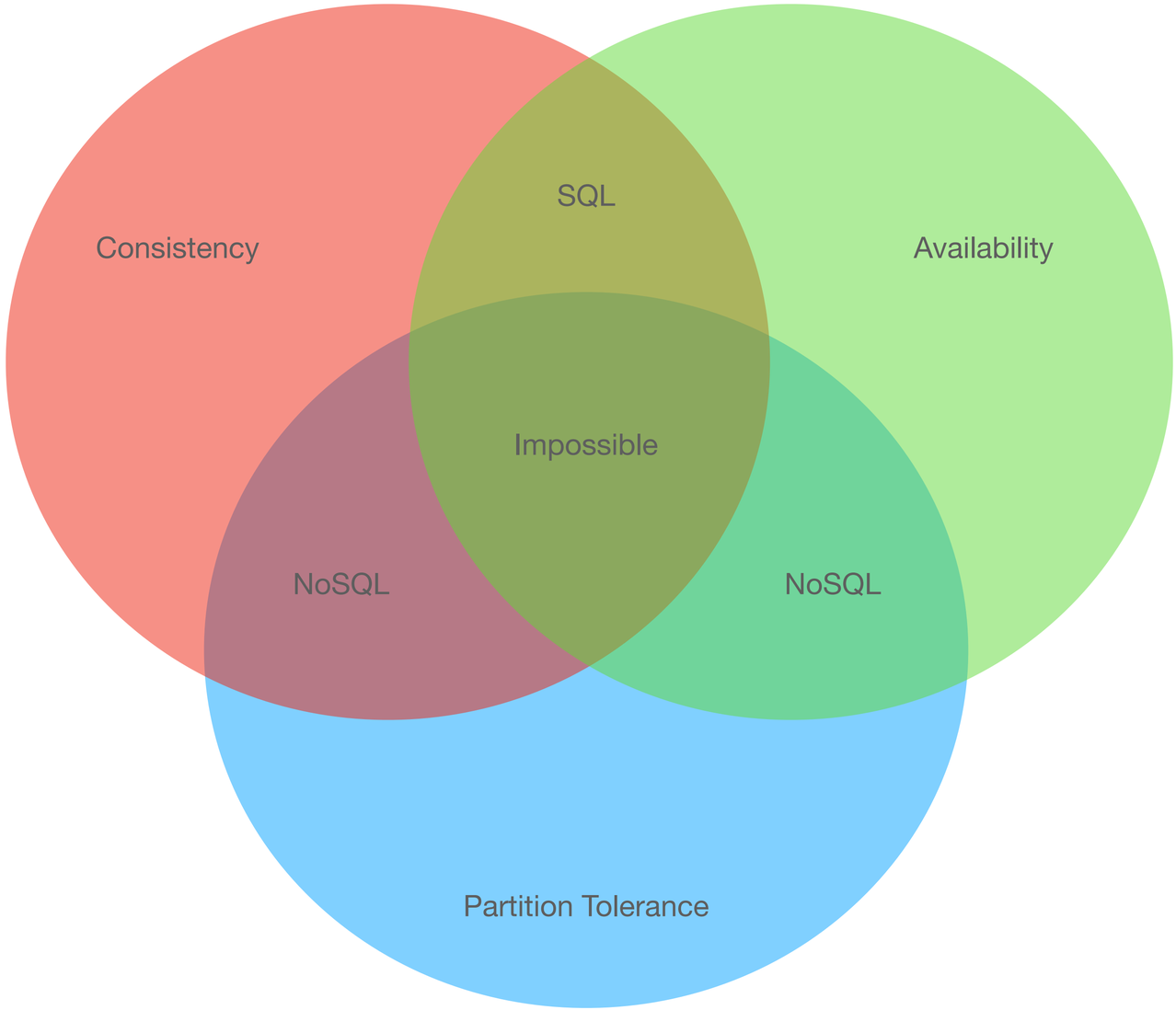
CAP theorem

Limits of all databases

Consistency, Availability, and Partition tolerance: pick two

CAP theorem

Limits of all databases



NoSQL

Overall characteristics

NoSQL databases, **in general**:

- Scale well horizontally
- Don’t implement ACID transactions
- Don’t have a standardised query language
- Organise information in unrelated aggregates
- Don’t require a schema
- Can handle large amount of data

NoSQL

About not having a schema

It does not mean storing whatever it comes with no structure.

Schema-less only means that the database does not enforce a schema.

Some NoSQL databases still use mandatory or optional schemas.

A “de-facto” schema is required to be able to query and parse the data.

Some databases support indexes requiring some form of structured data.

NoSQL types

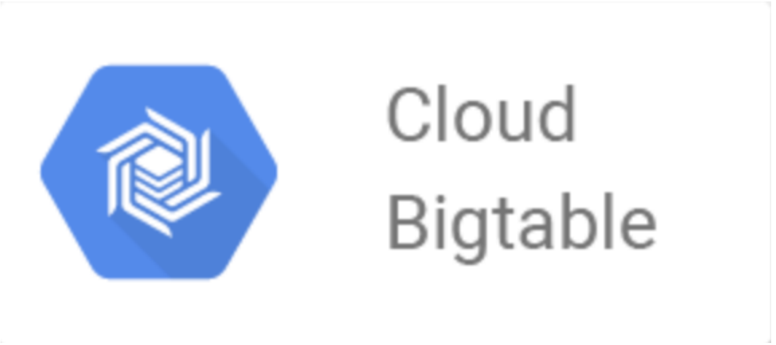
Key - Value

- Generally they are the fastest in Read/Write
- Don’t offer much flexibility in terms of query
- They work perfectly as a cache. Products like Hazelcast or Datagrid blur the difference between a cache and a (NoSQL) database.

NoSQL types

Column databases

- They organise data in tables
- They query tables per column or per row



NoSQL types

Document based

- Generally they are the slowest in Read/Write, but can work as key-value db
- Very flexible query language if the document is XML or JSON
- They may support transactions up to a degree. Transactionality is discouraged for performances. The border of the ideal transaction is the within a single document.

NoSQL types

Graph database

- Used to store relations
- Very fast to retrieve data with complex relational queries
- Slow at performing insertion or updates
- Limited use-cases
- One of the most famous product are Neo4j

NoSQL types

Time-series

- Used to store values associated to a timestamp
- Ideal for monitoring because they can ingest data rapidly and aggregate values
- Limited use-cases
- One of the most famous product is Druid

NoSQL

There is no clear cut

Many NoSQL databases do not fit in just one category. MarklogicDb is primarily a document-based db, but it implements graphs too.

Specifications change rapidly. MongoDb did not support ACID until a couple of years ago.

It is very common to have REST apis, some products support libraries, some are integrated in Spring. In general integration with NoSQL is harder.