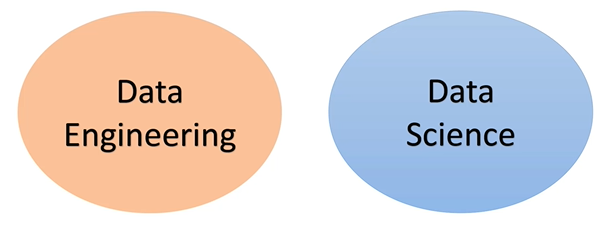
**Data-driven organization**

* Collecting data
* Refine data, remove noise
* Integrate data with other sources
* Build models to describe, predict and optimize business processes and outcomes

**The data driven infrastructure**



**Data engineer**: produces and maintains the data infrastructure

**Data scientist**: Analyst that uses data as raw material to produce intelligence

Data driven organizations

https://hbr.org/2012/10/making-advanced-analytics-work-for-you

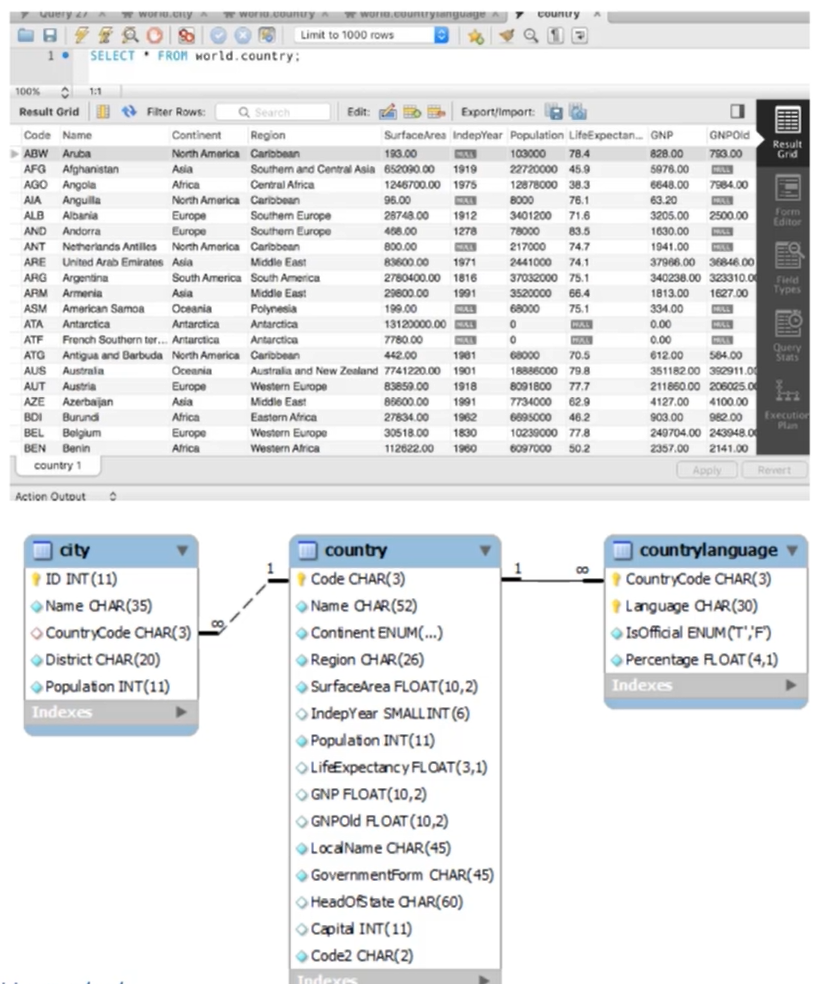
DB ranking

<https://db-engines.com/en/ranking>

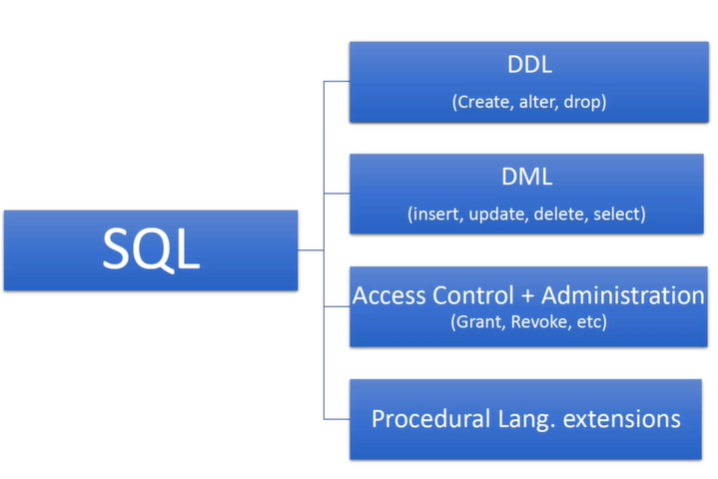
Git

<https://github.com/okmich/rdbms_2_nosql>

**Relational Databases**

* DB built on a relational model
* **Relations / Tables**
  + Basic unit of data storage
  + 2D structure
* **Record / Row / Tuple**
  + Instances of the data
* **Primary Key(s)**
  + Unique identification of the table
* **Fields / Columns / Properties**
  + Attributes
* **Data Types**
  + Specified in the creation of the table for each field
* **NULL**
  + Can be added in any field
* **Constraints**
  + Ex: Column can’t receive NULL, must be unique values
* **Uniqueness**
  + Value unique in a particular field
* **Relationships**
  + Connects two tables
  + Foreign keys
* **Transactions (ACID)**
  + Performs actions on a database
* **Index**
  + Help improve the performance of read operations
  + Database objects
  + DB views
* **ERD**
  + Entity relationship diagram
  + Layout the entities and relationships between tables
* **RDBMS**
  + Relational database management system
  + Tools, programs to manage, monitor, secure the DB. Ex: MySQL, Postgres

**Structured Query Language (SQL)**

* Based on relational algebra

Data definition language

Data manipulation language

**Data modeling**

**1**

* Table for **Actors**
  + Ex: Store, customer, Seller

**2**

* Table for **Actions** of actors
  + Ex: Customers make orders, complains, returns

**3**

* Table for **properties of actors** that don’t fit to 2D diagram
  + Ex: Possible payment methods for customer

**4**

* Table for **properties of actions** that don’t fit to 2D diagram
  + Ex: Possible reasons for return

**5**

* **Optimization**
  + Normalization: separate tables for redundancy reduction (Ex: doc, doctor, med doc)
    - Cons: You know less from only one table. Have to read more than one table
  + Denormalization: application of JOINS to integrate tables

**MySQL**

* Opensource RDBMS
* Second most used DBMS (behind Oracle)
* Written in C and C++

**Database systems**

* Two groups
  + **OLTP** (Online Transaction Processing)
    - * Many inserts
      * Crud applications
      * Always favor normalization
      * Must include START TRANSACTION; COMMIT; ROLLBACK;
  + **OLAP** (Online Analytical processing)
    - * It’s more read than written
      * Provide a source for Business Intelligence
      * Data mining

**Data Processing in RDBMS**

u,

* In the application space
  + LAMP stack or Django
  + Application data structure
  + Suited for OLTP application (small amount of data)
* In the database space
  + MS Access Forms or Oracle Apex
  + Database objects
  + Large amount of data
  + MapReduce, Spark, Pig: Executes the logic in the location the data exists

**Indexes**

→ Reduce the time to make a query

→ Queries no longer have to make a table scan

**ETL (Extract, Transform, Load)**

→ Integrate the business data with data from other sources

→ Saved in a Data Warehouse

**Analytical Processing**

→ Aggregation Queries

* Aggregation functions
* Sum, count ...

→ Analytical Queries

**Logs**

→ Transaction logs

→ Online logs

→ Write ahead logging: Log before writing

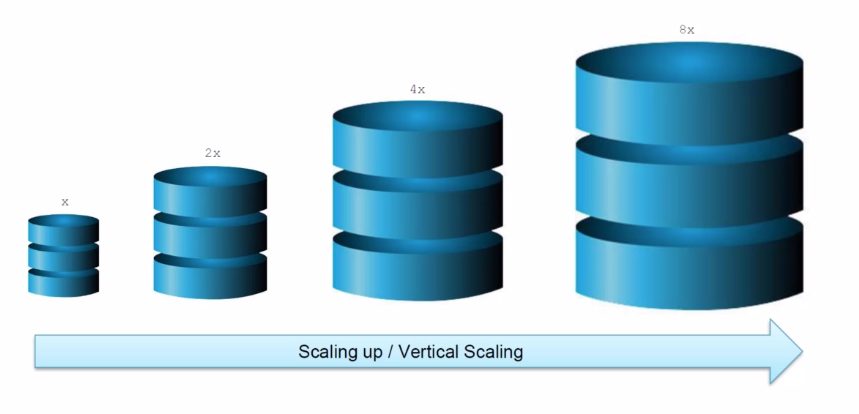
→ Replication

* In case something wrong happens to the data set, system can continue in another database

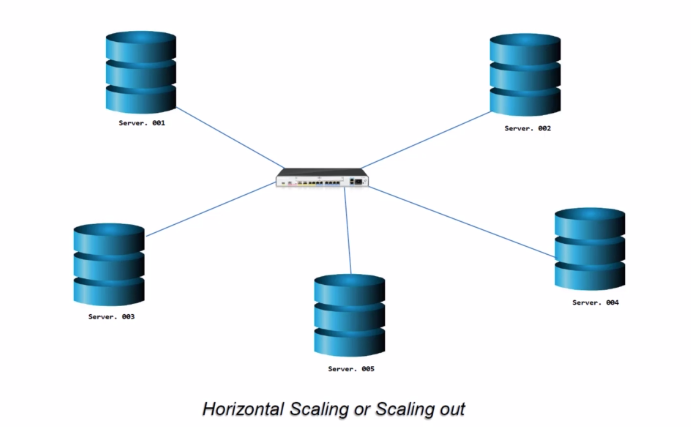
**Relation Database Wrap-up**

Scaling the RDBMS Architecture

**→ Vertical scaling**



**→ Horizontal Scaling**

→ Machines added to network

→ Distributed databases

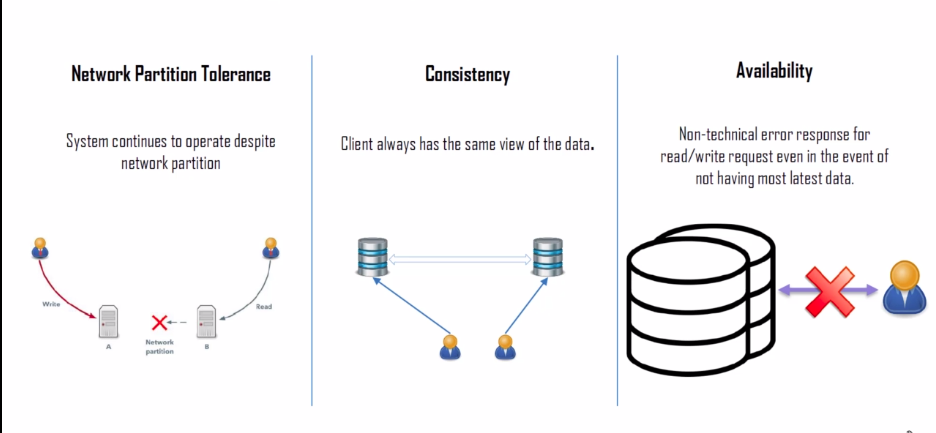
→ Tables broken into partitions. Each partition in a machine

→ Must have REPLICATION of all nodes. Replication FACTOR

→ RF 3. Any time data is written it’s copied 3 times in other machines.

→ Copy = Replica

**Concerns of Distributed Databases**



→ **Network Partition Tolerance**: DB continues to function

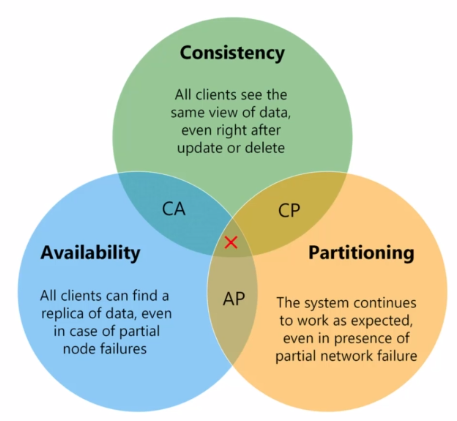
→ **Consistency**: All values are the same for any machine or request

→ **Availability**: System available for read/write requests

**CAP Theory**

**C**: Consistency, **A**: Availability, **P:** Partitioning

→ Cannot have always the 3 of them!

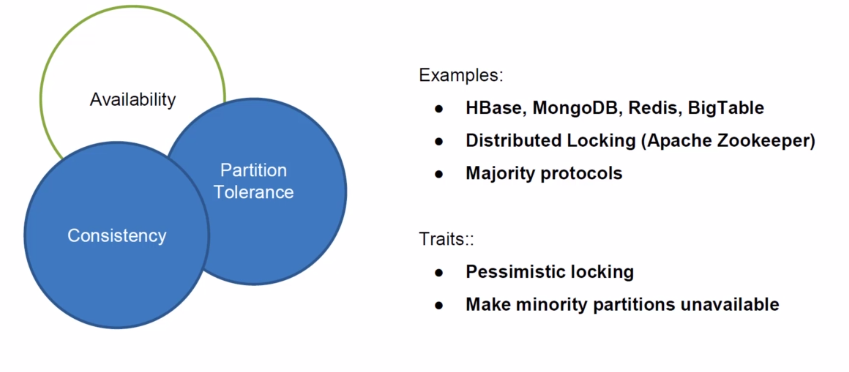


**CP**

→ The write is incomplete until it’s replicated to all the other machines

→ But what if at the write there was a partition failure? Write will never be complete

→ Availability not be reached!

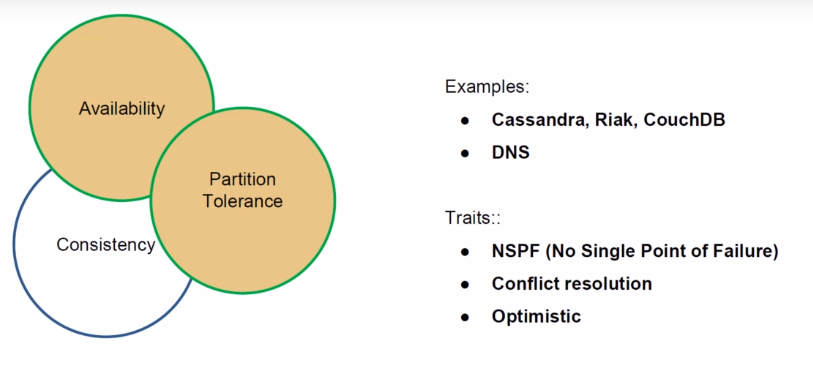


**AP**

→ The failed server will not be updated

→ Failed server will show outdated results

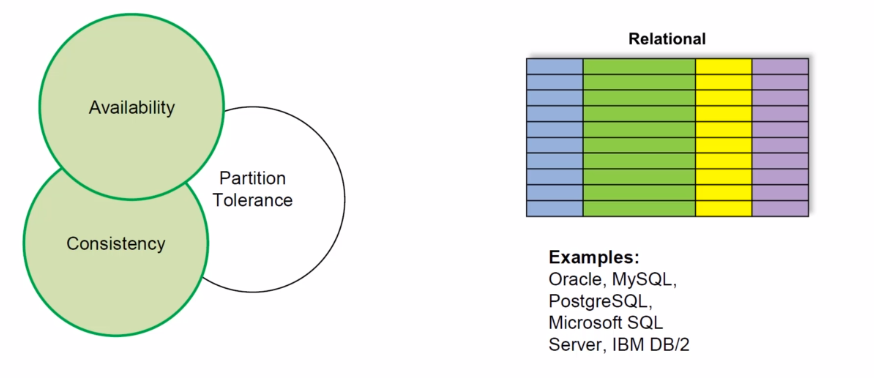
→ Consistency not reached!



**CA**

→ In a single RDB it’s already possible

→ Partitioning is not reached!



ACID vs. BASE

**Other classifications of DB’s**

Data Model

→ Key-value model

→ Graph model

Data Variability

→ ...

Operational Capacity

→ …