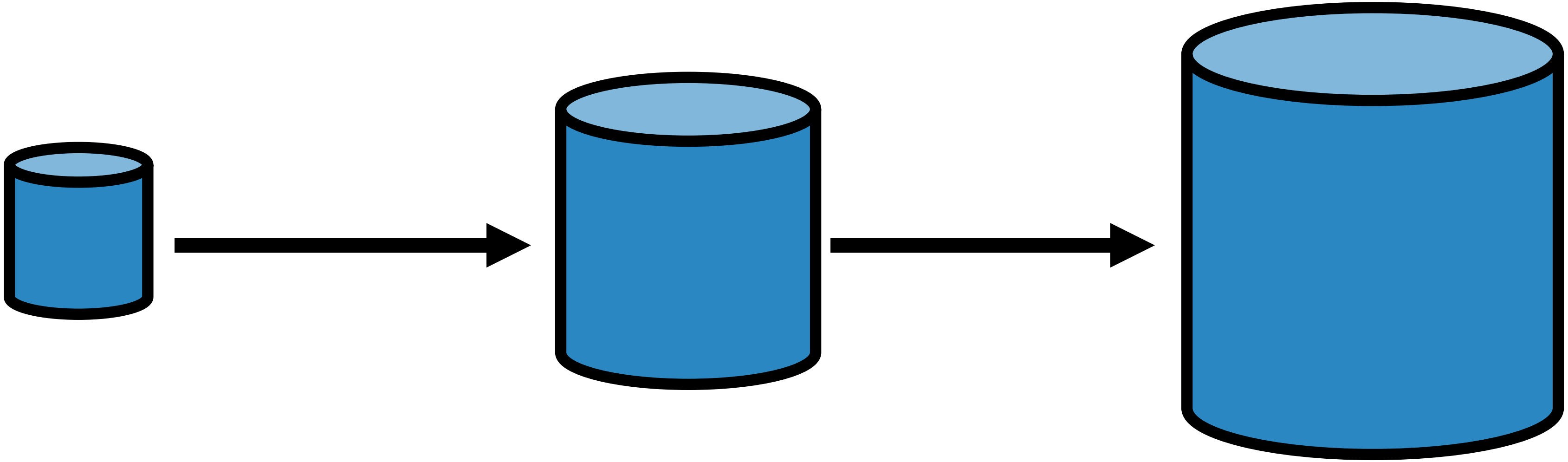
Introduction to NoSQL Databases

datasciencedojo
unleash the data scientist in you

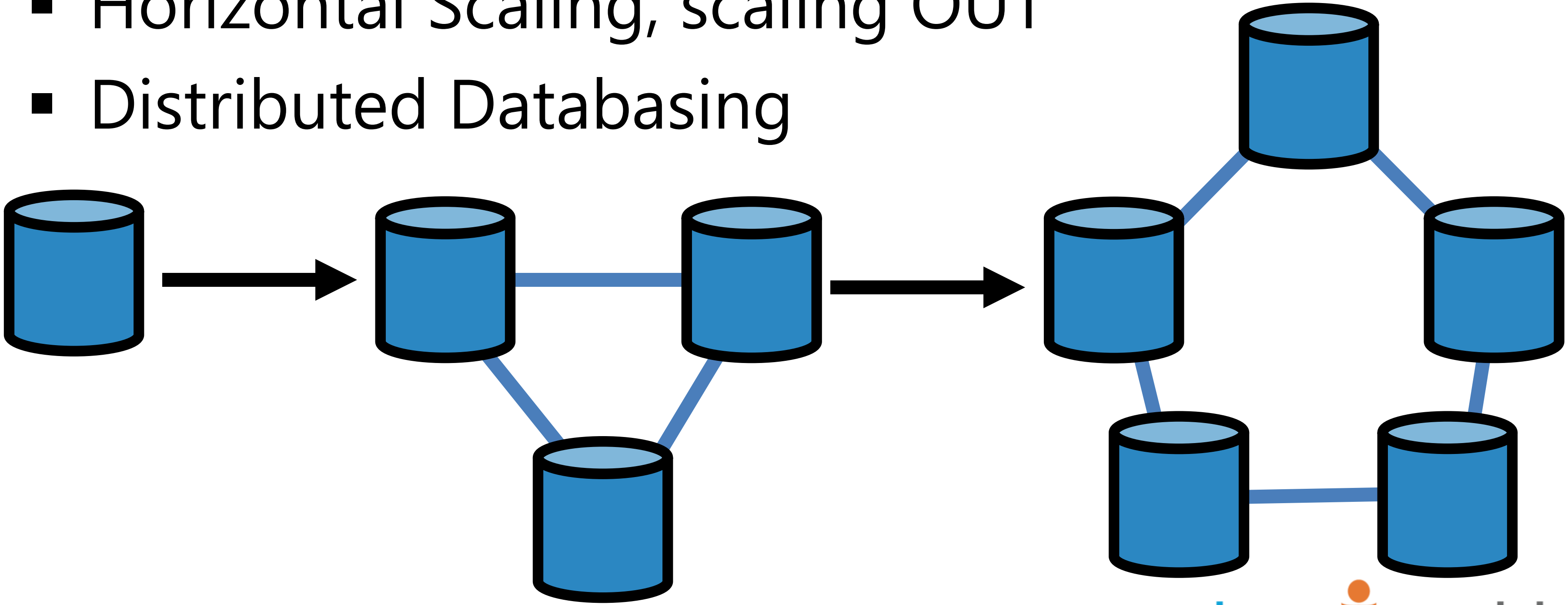
Scaling, Traditional Relational DB

- Vertical Scaling, scaling UP



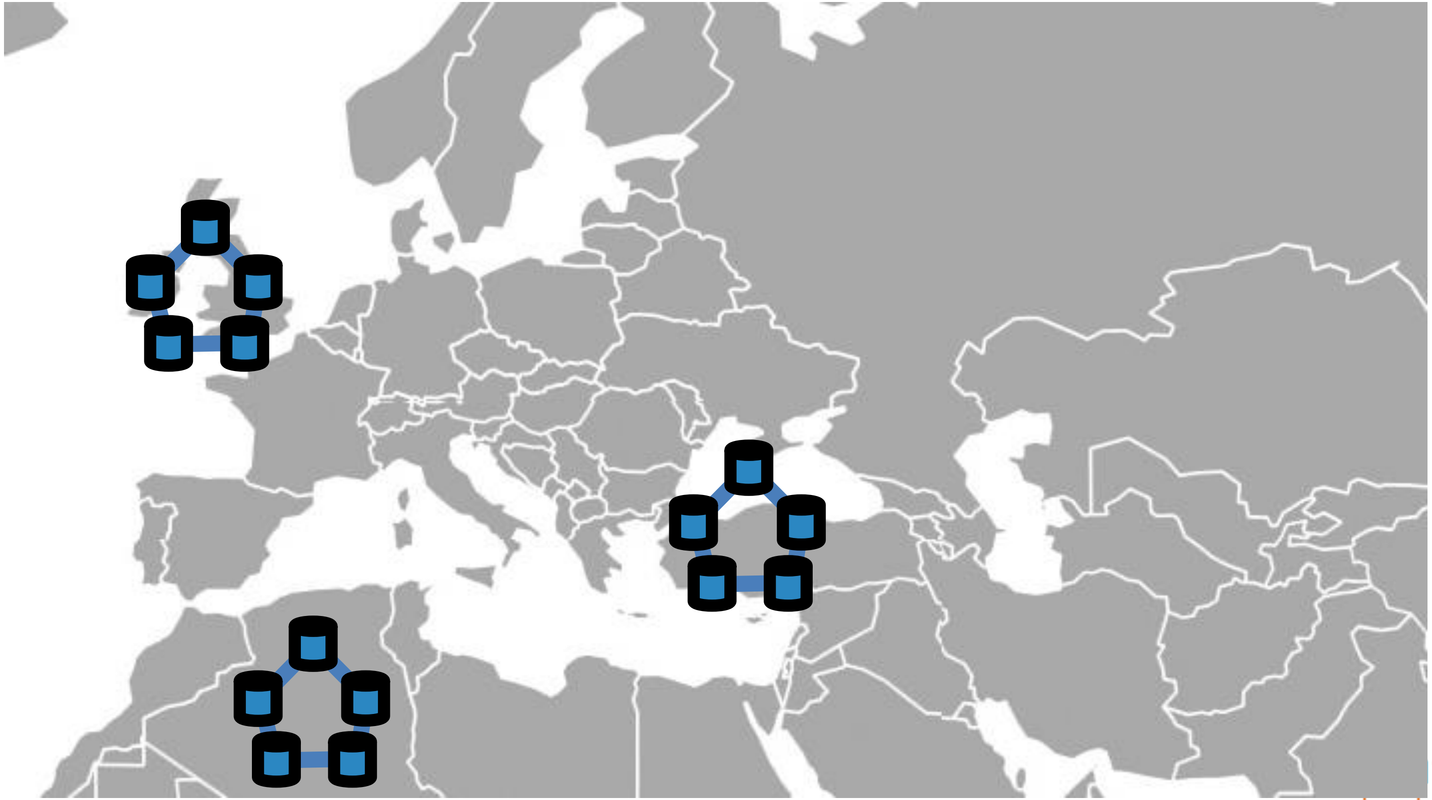
Scaling, NoSQL Era

- Horizontal Scaling, scaling OUT
- Distributed Databasing



datasciencedojo
unleash the data scientist in you

Scaling, NoSQL Era



sciencedojo
unleash the data scientist in you

Data Architecture

- No standard solution that fits all
- Business and data defines the architecture
- Multiple databases, different types depending on the characteristics of each data subset

CAP

Consistency

This means that the data in the database remains consistent after the execution of an operation. For example after an update operation all clients see the same data.

Availability

This means that the system is always on (service guarantee availability), no downtime.

Partition Tolerance

This means that the system continues to function even the communication among the servers is unreliable, i.e. the servers may be partitioned into multiple groups that cannot communicate with one another.

CAP Theorem

- It is impossible for a distributed processing system to simultaneously provide all three of the following guarantees
  - **Consistency** - A read is guaranteed to return the most recent write for a given client.
  - **Availability** - A non-failing node will return a reasonable response within a reasonable amount of time (no error or timeout).
  - **Partition Tolerance** - The system will continue to function when network partitions occur.

CAP Theorem

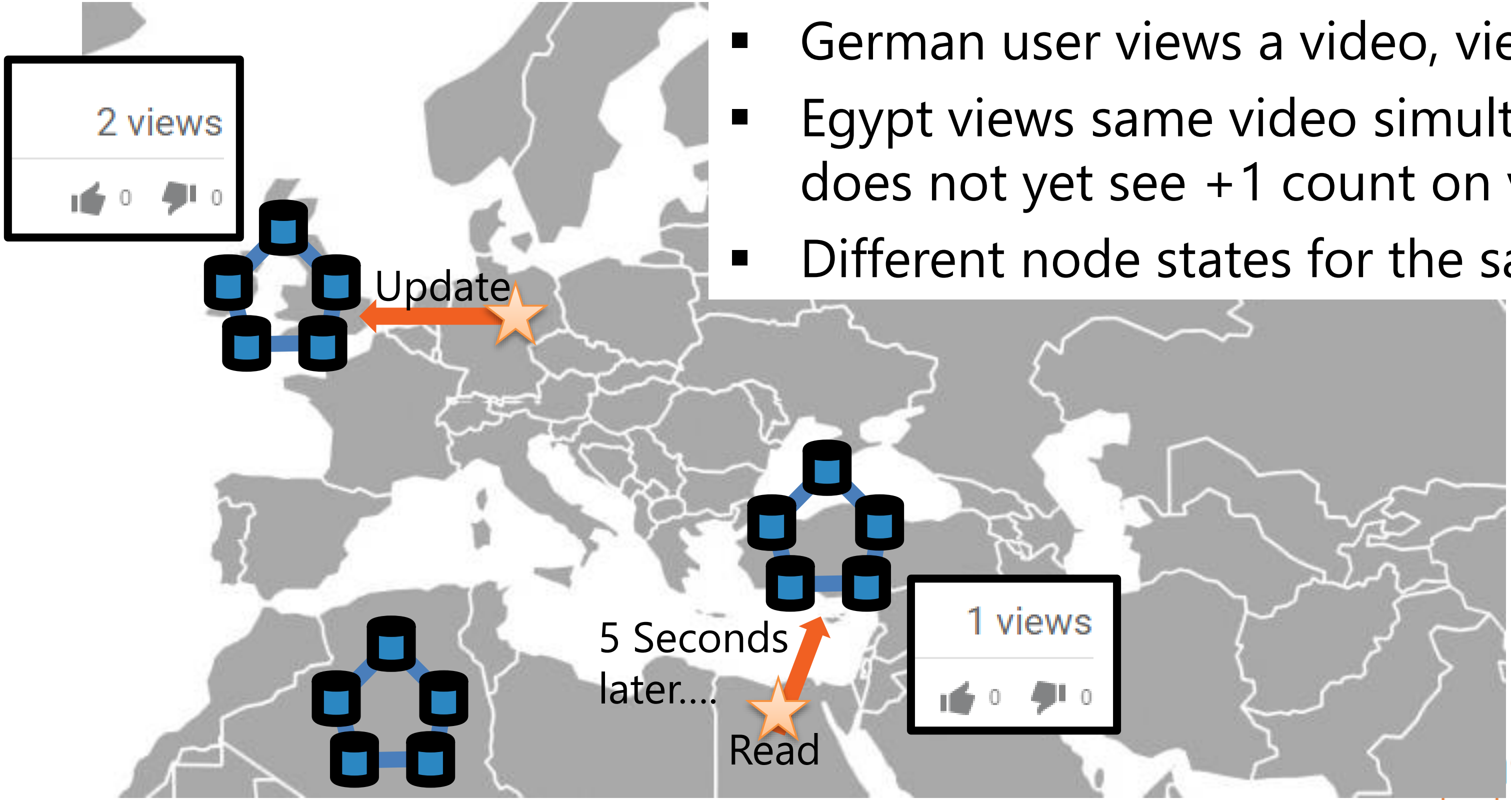
CA - Single site cluster, therefore all nodes are always in contact. When a partition occurs, the system blocks.

CP - Some data may not be accessible, but the rest is still consistent/accurate.

AP - System is still available under partitioning, but some of the data returned may be inaccurate.

AP – Lack of (Immediate) Consistency

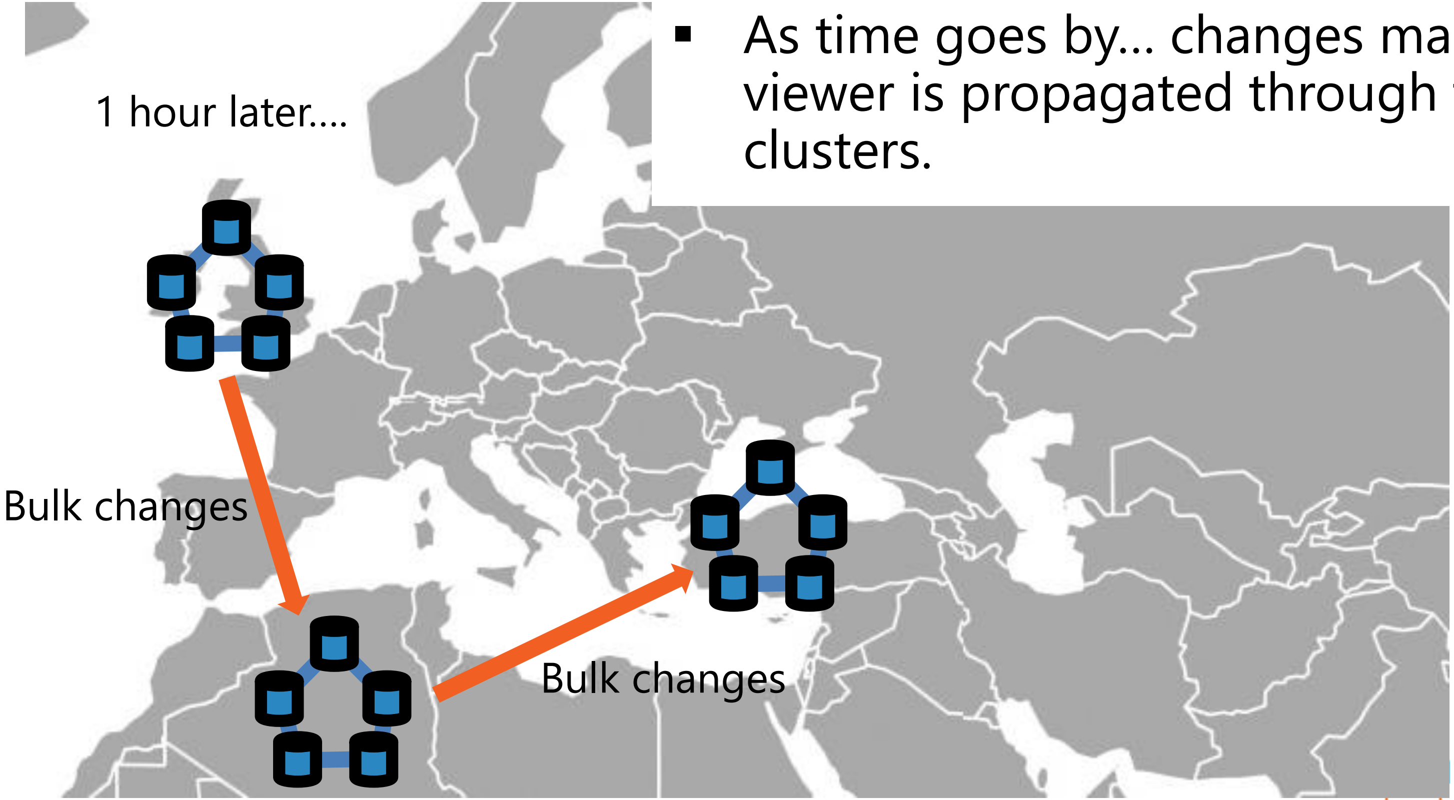
- German user views a video, view count +1
- Egypt views same video simultaneously, does not yet see +1 count on video
- Different node states for the same record



sciencedojo
unleash the data scientist in you

AP – Eventual Consistency

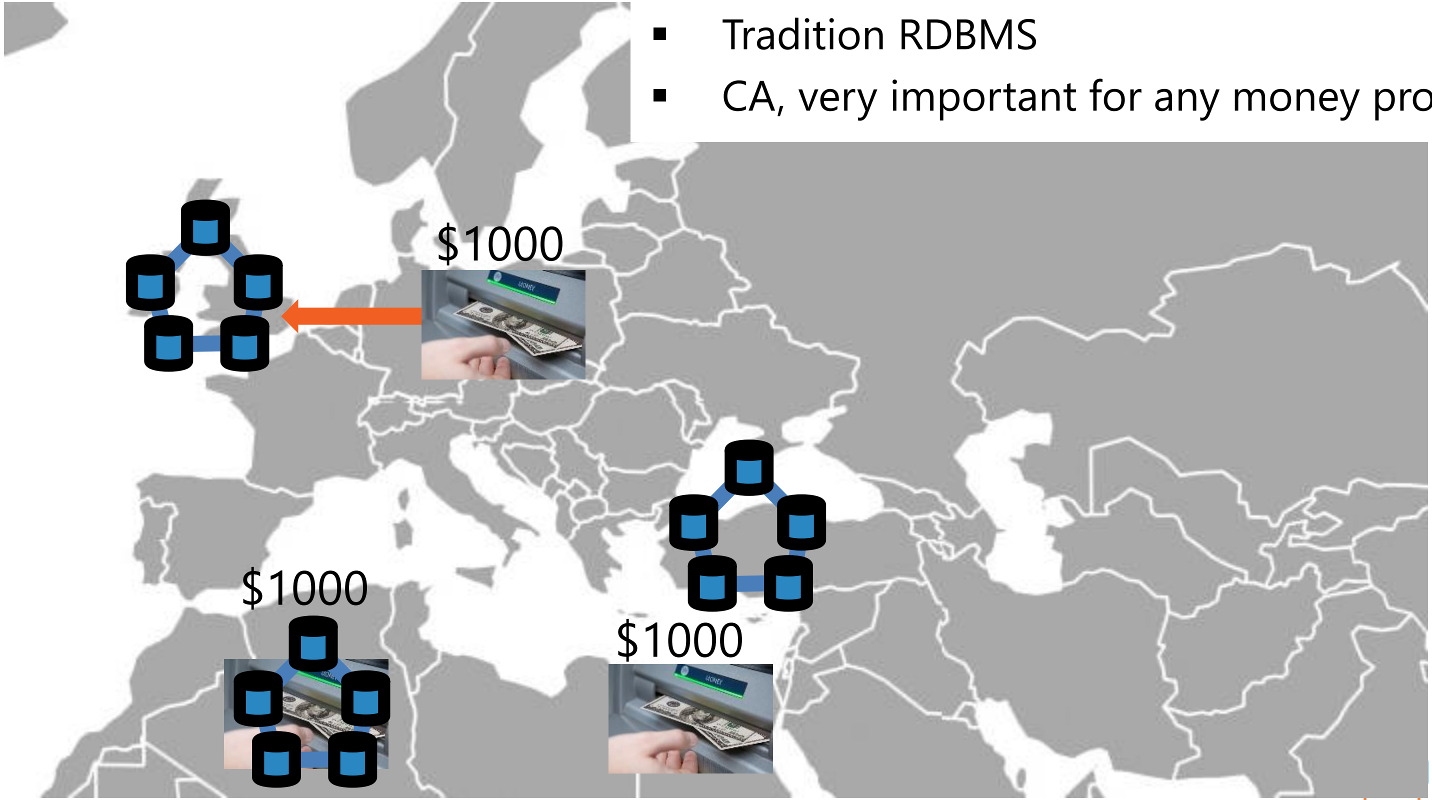
- As time goes by… changes made by German viewer is propagated through the other DB clusters.



ATMs

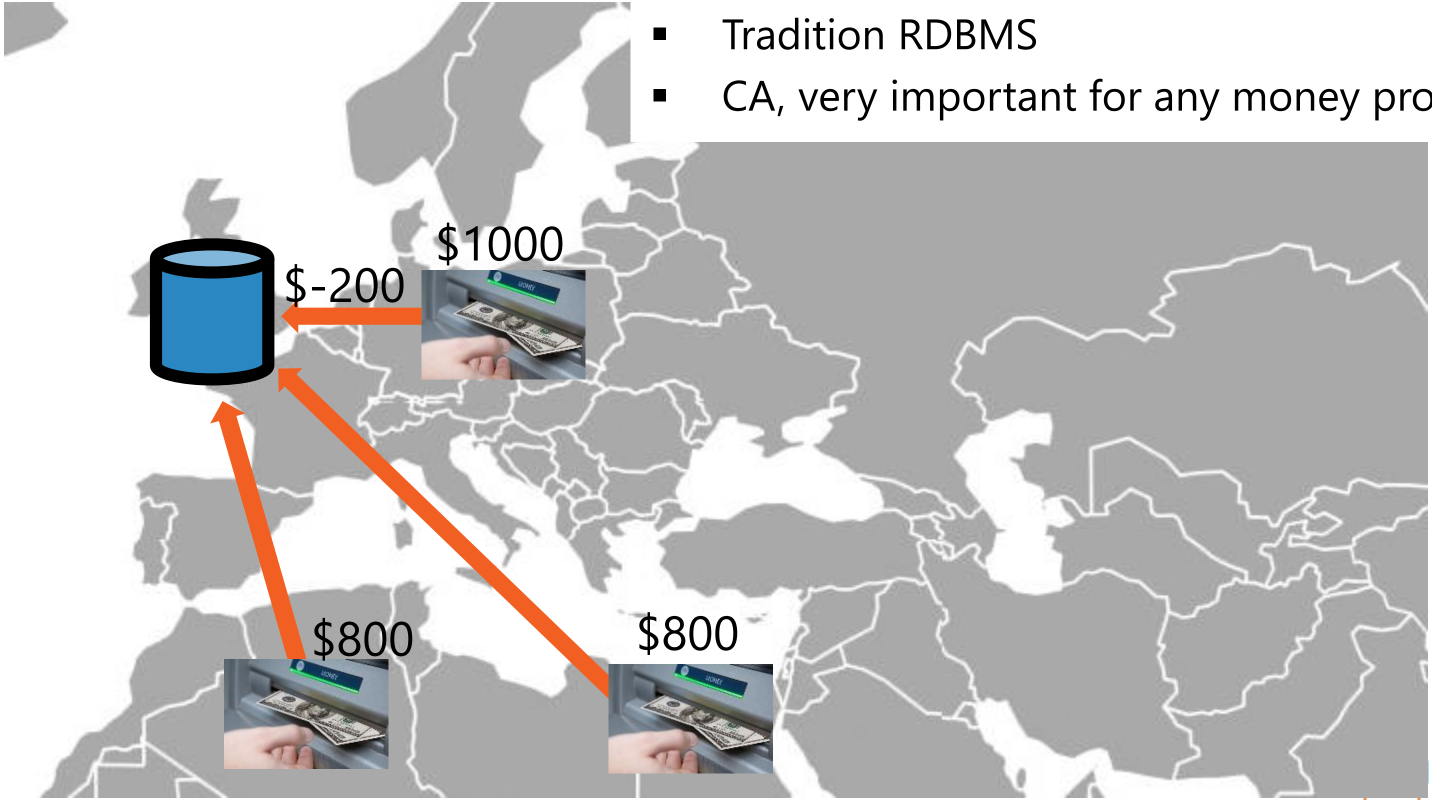
Why Consistency Matters

- Tradition RDBMS
- CA, very important for any money processing

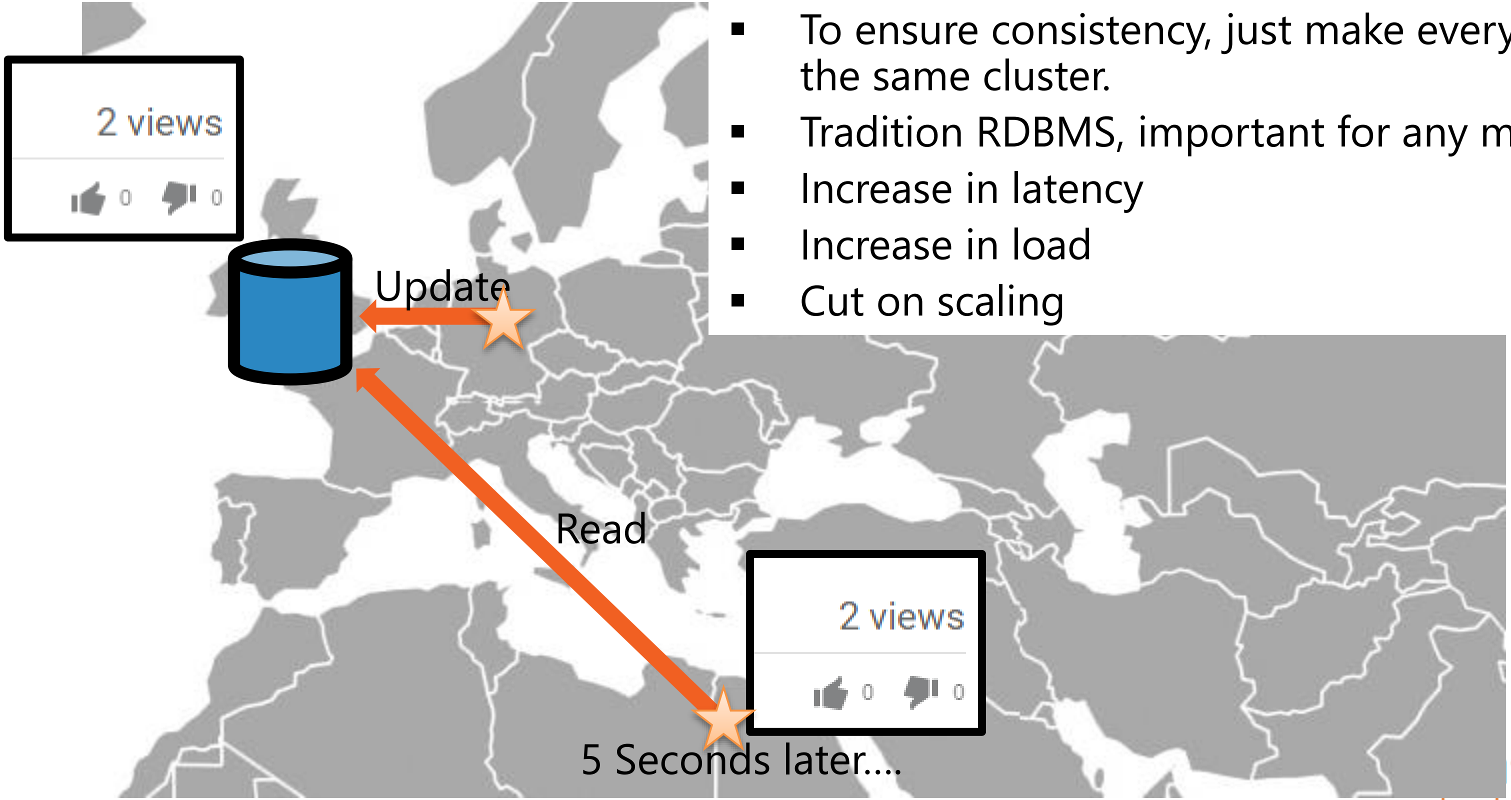


Why Consistency Matters

- Tradition RDBMS
- CA, very important for any money processing



CA – Lack of Partition Tolerance

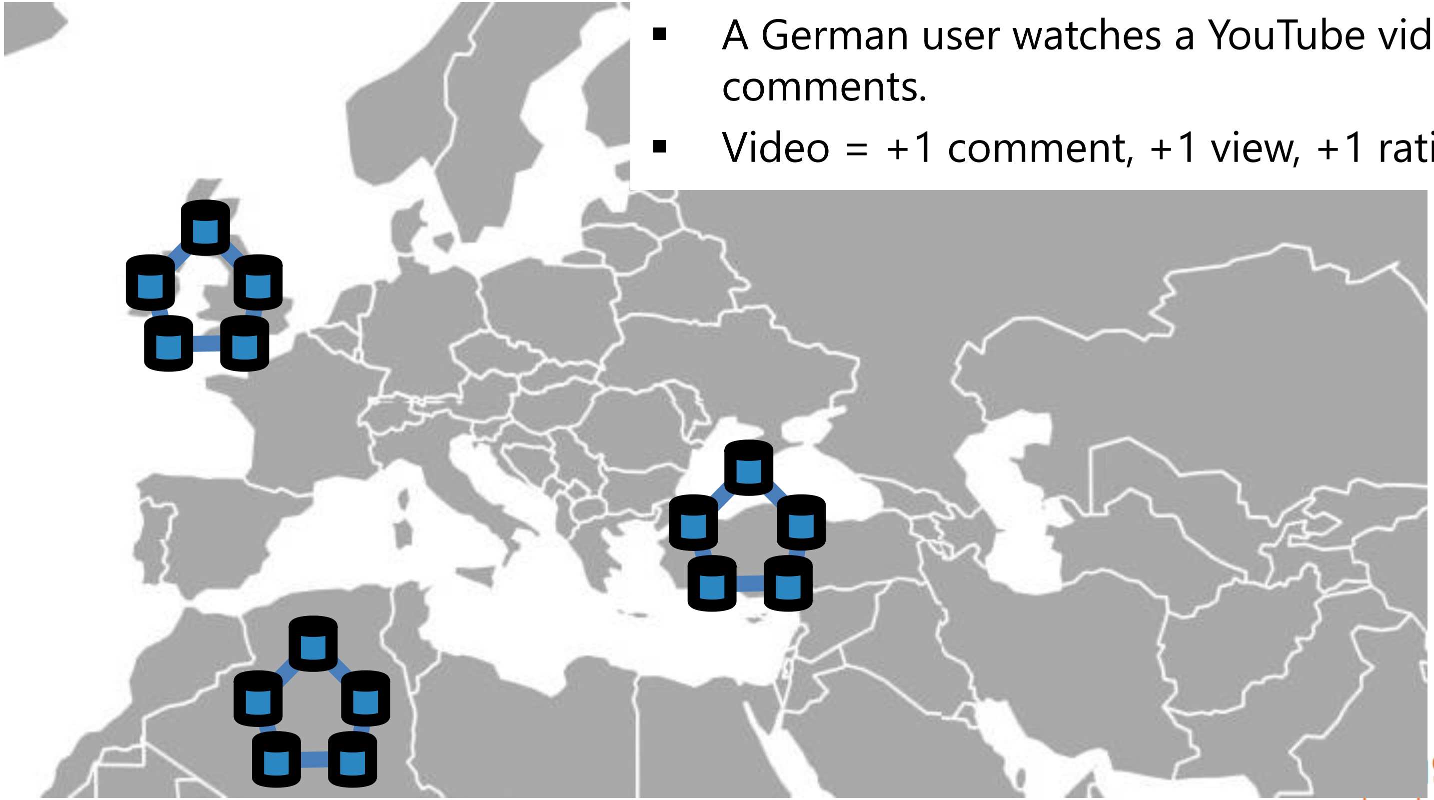


- To ensure consistency, just make everyone read from the same cluster.
- Tradition RDBMS, important for any money processing
- Increase in latency
- Increase in load
- Cut on scaling

sciencedojo
unleash the data scientist in you

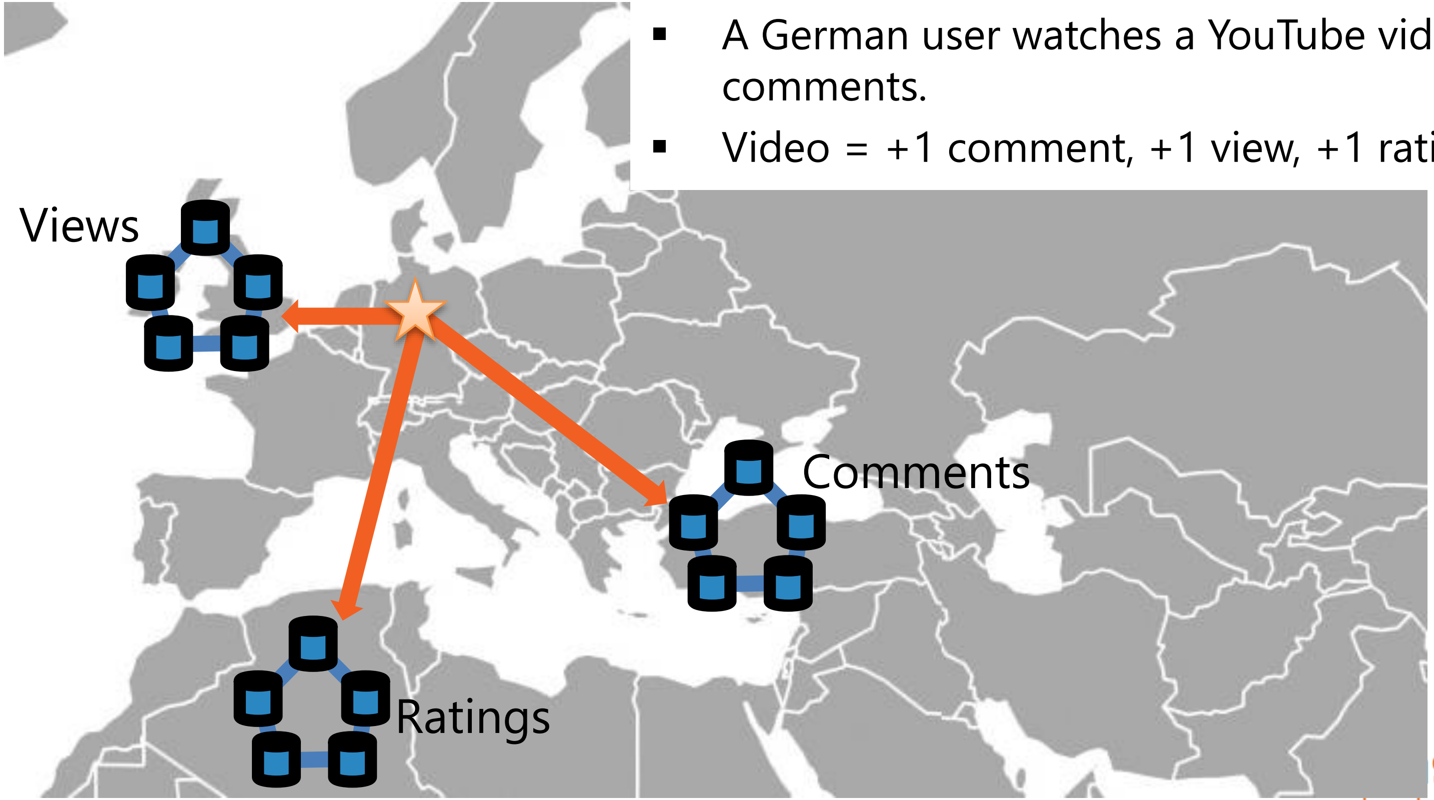
Consistency + Partitions?

- A German user watches a YouTube video, rates it, then comments.
- Video = +1 comment, +1 view, +1 rating



Consistency + Partitions?

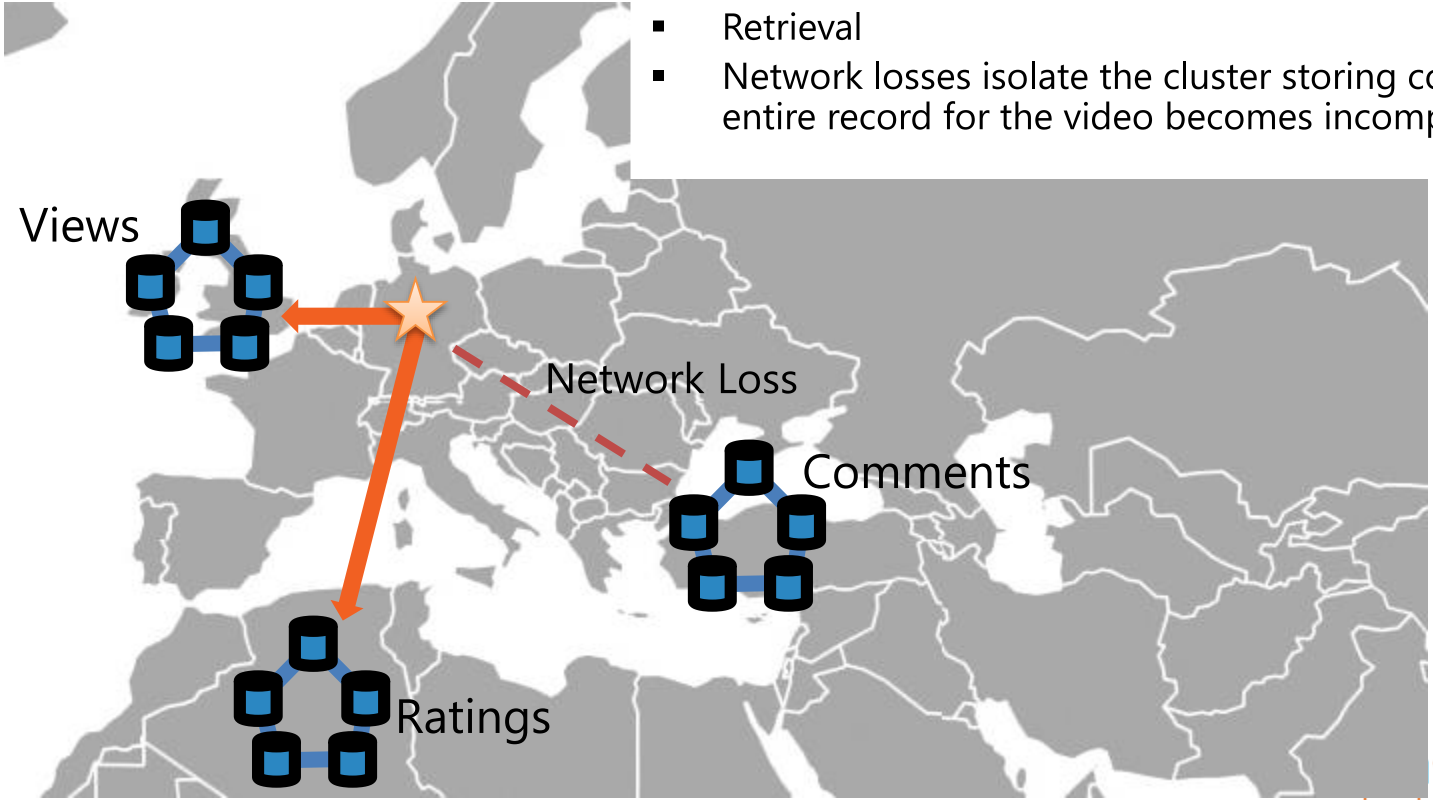
- A German user watches a YouTube video, rates it, then comments.
- Video = +1 comment, +1 view, +1 rating



sciencedojo
unleash the data scientist in you

CP: Loss of Availability

- Retrieval
- Network losses isolate the cluster storing comments. The entire record for the video becomes incomplete (temporarily)



CAP Theorem

CAP provides the basic requirements for a distributed system to follow **2 of the 3 requirements.**

In theoretically it is **impossible** to fulfill all 3 requirements.

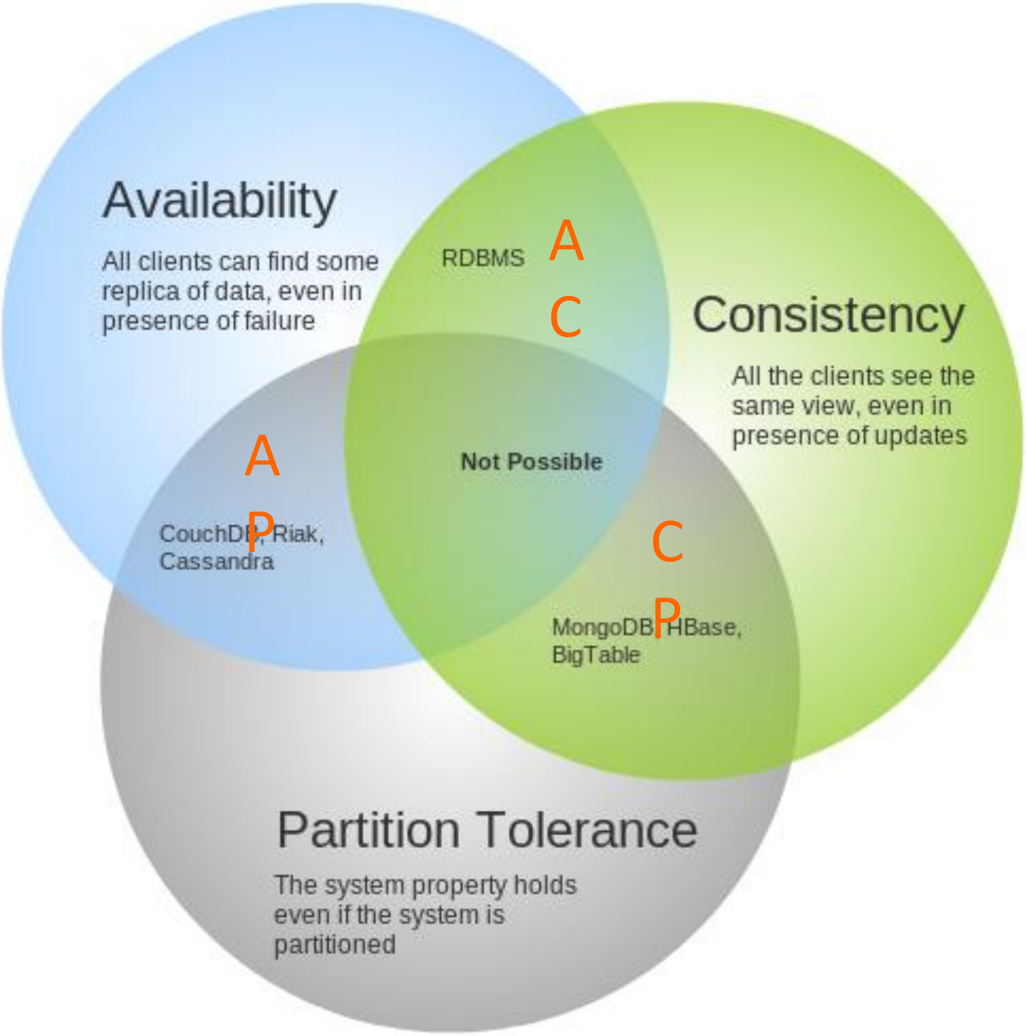
Therefore all the current NoSQL database follow the different **combinations of the C, A, P** from the CAP theorem.

datasciencedojo
unleash the data scientist in you

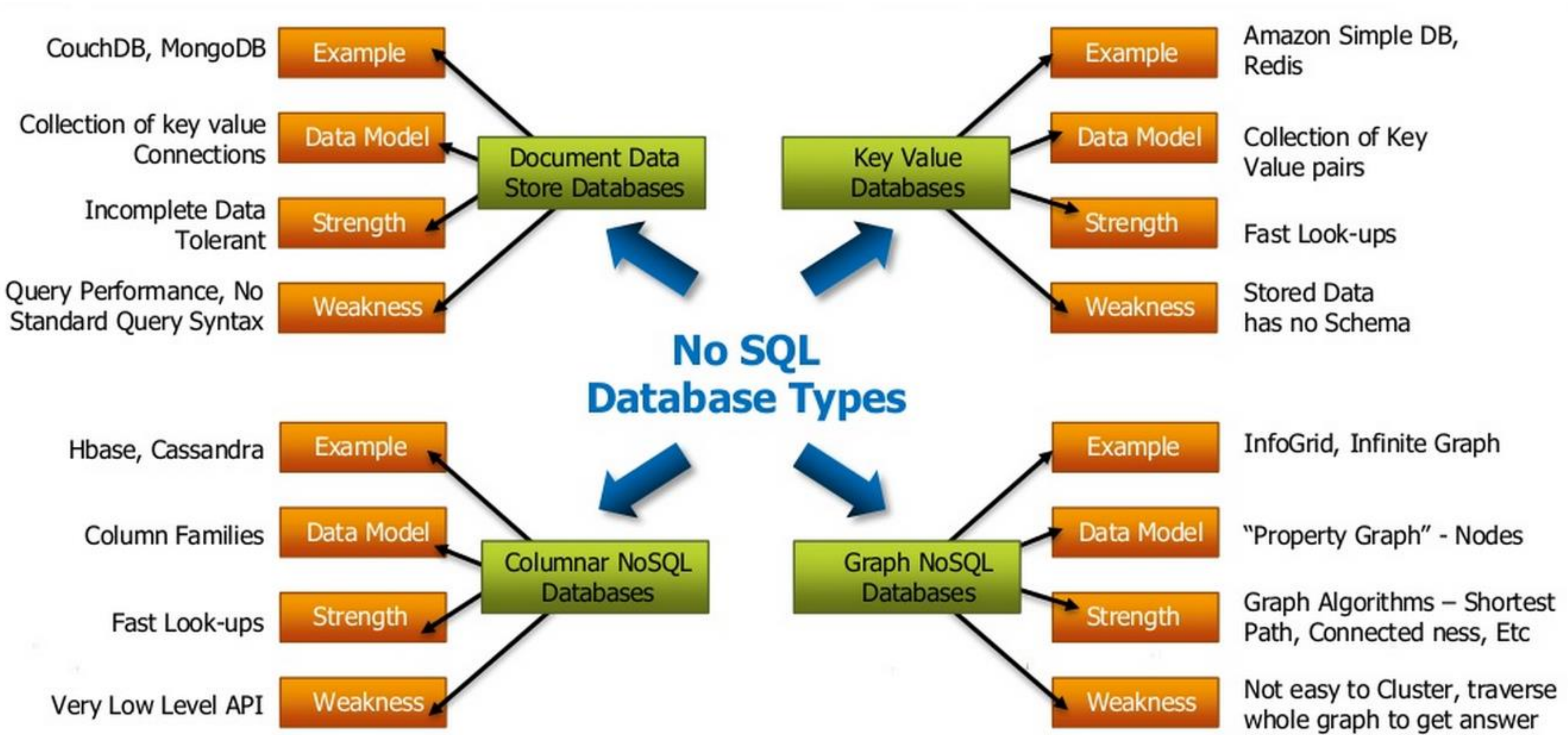
NoSQL vs. SQL

- NoSQL
  - Availability first (Consistency second)
- SQL (Traditional RDBS databases)
  - Consistency first (Availability second)

datasciencedojo
unleash the data scientist in you



datasciencedojo
unleash the data scientist in you



datasciencedojo
unleash the data scientist in you

What is HBase

- Distributed, non-relational database
  - Columnar, schema-free data model
  - NoSQL on top of Hadoop
- Large scale
  - Linear scalability
  - Billions of rows X millions of columns
  - Many deployments with 1000+ nodes, PBs of data
- Low latency
  - Real-time random read/writes
- Open Source
  - Modeled after Google's BigTable
  - Started in 2006

Row Store

Table

	Country	Product	Sales
Row 1	India	Chocolate	1000
Row 2	India	Ice-cream	2000
Row 3	Germany	Chocolate	4000
Row 4	US	Noodle	500

Pros:

- Fast record query
- Relationships
- Less redundancy
- Single line insert

Cons:

- Thin tables
- Getting a single column value, retrieves the entire record
  - Terrible with wide tables
- Aggregations must sift through all columns for each row

Columnar

Table

	Country	Product	Sales
Row 1	India	Chocolate	1000
Row 2	India	Ice-cream	2000
Row 3	Germany	Chocolate	4000
Row 4	US	Noodle	500

Pros

- High speed aggregations
- Compression
- Wide tables are now possible (billions of columns, instead of hundreds)
- High speed snap shot retrieval
- Easily Distributable

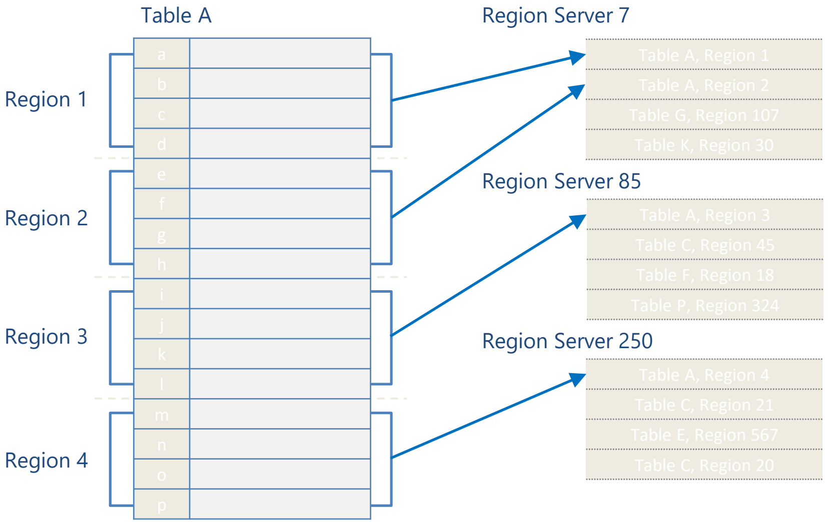
Cons

- Bad for record retrieval
- Terrible at relationships
- M line insert

datasciencedojo
unleash the data scientist in you

Data Model

- Scale-out architecture
  - Automatic sharding of tables
  - Automatic failover
  - Strong consistency for reads and writes
- APIs
  - Get/Put
  - Scan
  - Coprocessors



Performance Features

- Column Families
- In-memory caching
- High throughput streaming writes

Row Key	Customer		Sales	
Customer Id	Name	City	Product	Amount
101	John White	Los Angeles, CA	Chairs	$400.00
102	Jane Brown	Atlanta, GA	Lamps	$200.00
103	Bill Green	Pittsburg, PA	Desk	$500.00
104	Jack Black	St. Louis, MO	Bed	$1600.00

Column Families

Sharding

- Holding rows of database on different partitions
- Same table divided onto different servers, even different geographies
- Reduces index size

Sharding

- More reliance on interconnection between servers
- Increased latency in querying when more than one shard must be searched
  - Some searches are fast, others are slow
- Often no guarantees about cross shard consistency

Notable Capabilities

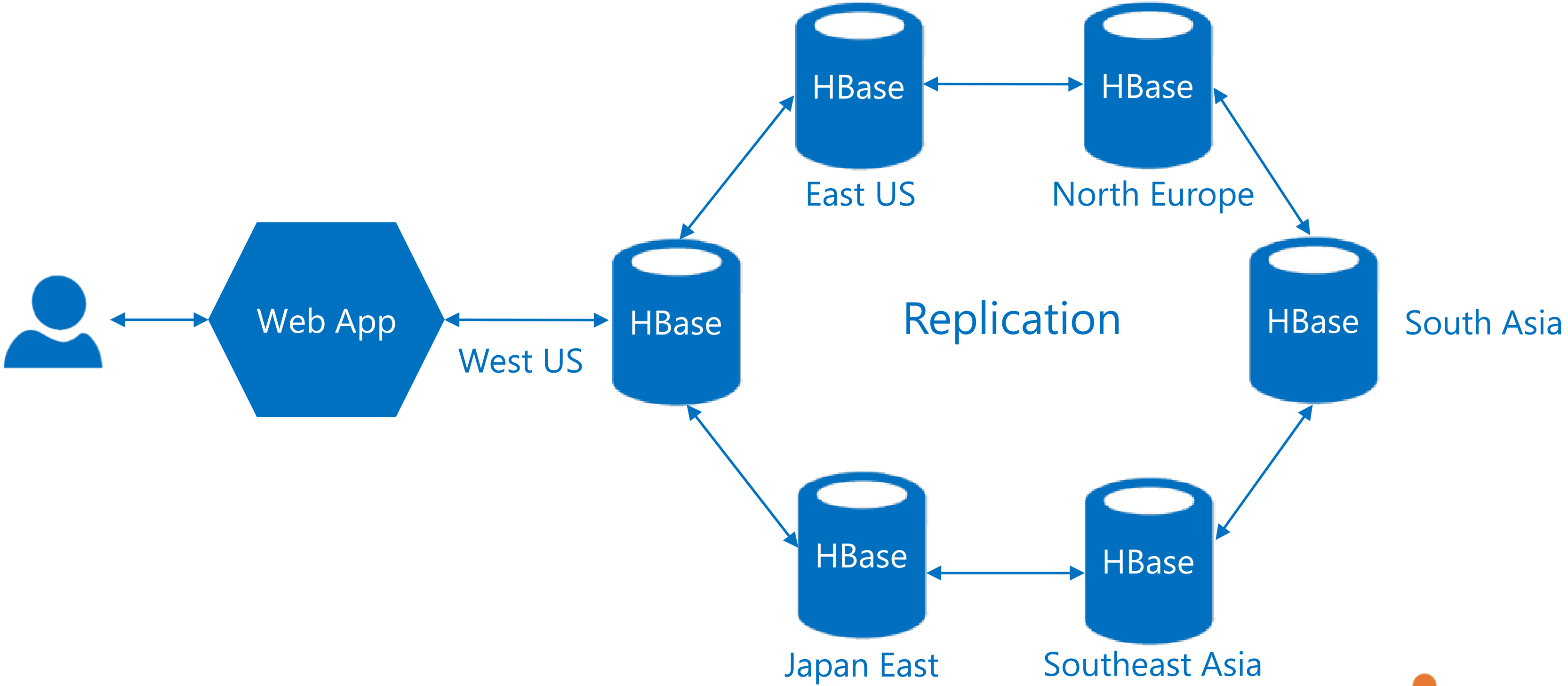
- Integration features
  - Integration with Hadoop MapReduce, Hive, Tez, Spark (hardware pending)
  - Bulk import of large amounts of data
- Client APIs
  - Java, REST, python, node.js, php, .NET

Use case #1: key value store

- Key value store
  - Message systems
  - Content management systems
- Examples
  - Facebook Messages
  - Twitter-like messages
  - Webtable – web crawler/indexer

datasciencedojo
unleash the data scientist in you

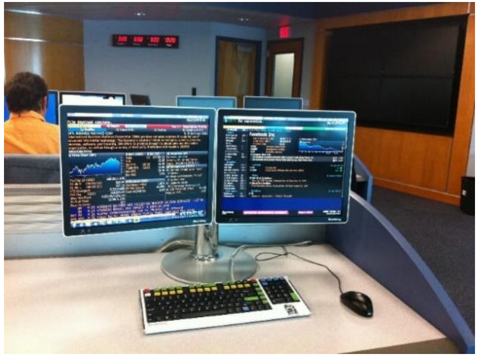
Use case #1: key value store

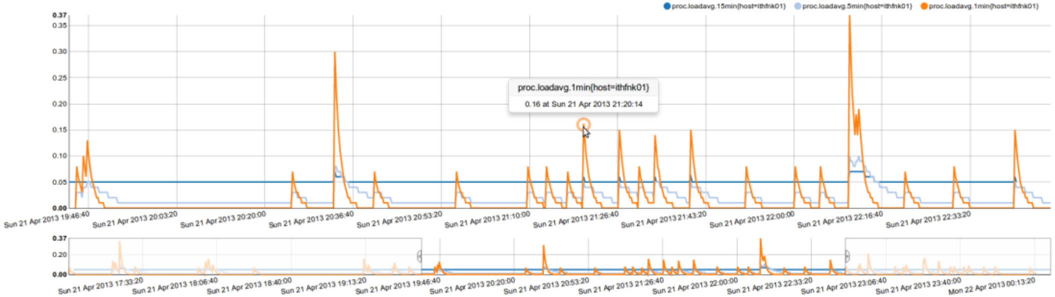


- Architecture

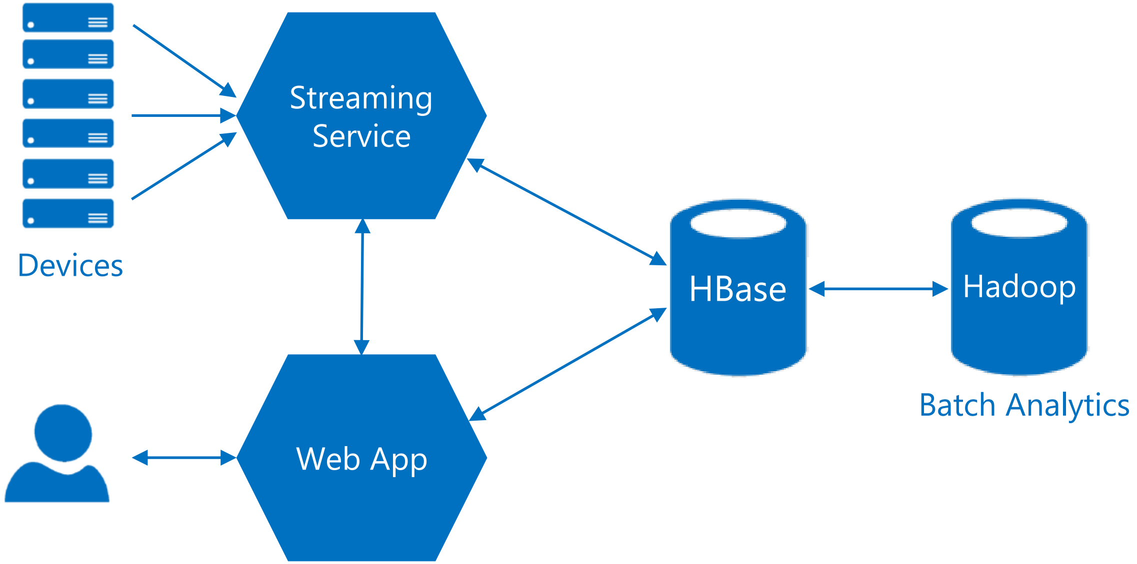
Use case #2: sensor data

- Sensor data
  - Social analytics
  - Time series databases
  - Interactive dashboards with trends, counters, etc
  - Audit log systems





Use case #2: sensor data



- Architecture

Questions?