BigData - distributed system and noSql

Module program

Panorama Big Data

2st Part: Big Data & NoSQL

8	Lecture - Big Data
9	Lecture/Lab - Column DB or Documentdb
10	Lecture - Data Lake
11	Lecture/Lab - Airflow Scheduler
12	Lecture/Lab - ELK
13	Lecture/Lab - Apache Spark
14	Lecture/Lab - Graph DB *Big Data report to give

isep
École d'ingénieurs du numérique

Big Data and NoSQL Databases

Plan

I. Context
   a. The 3Vs and Decision-Making
   b. Limitations of RDBMS
II. ACID vs BASE
   a. NoSQL
   b. Distribution
   c. The 4 families
   d. CAP theorem
   e. Map/Reduce

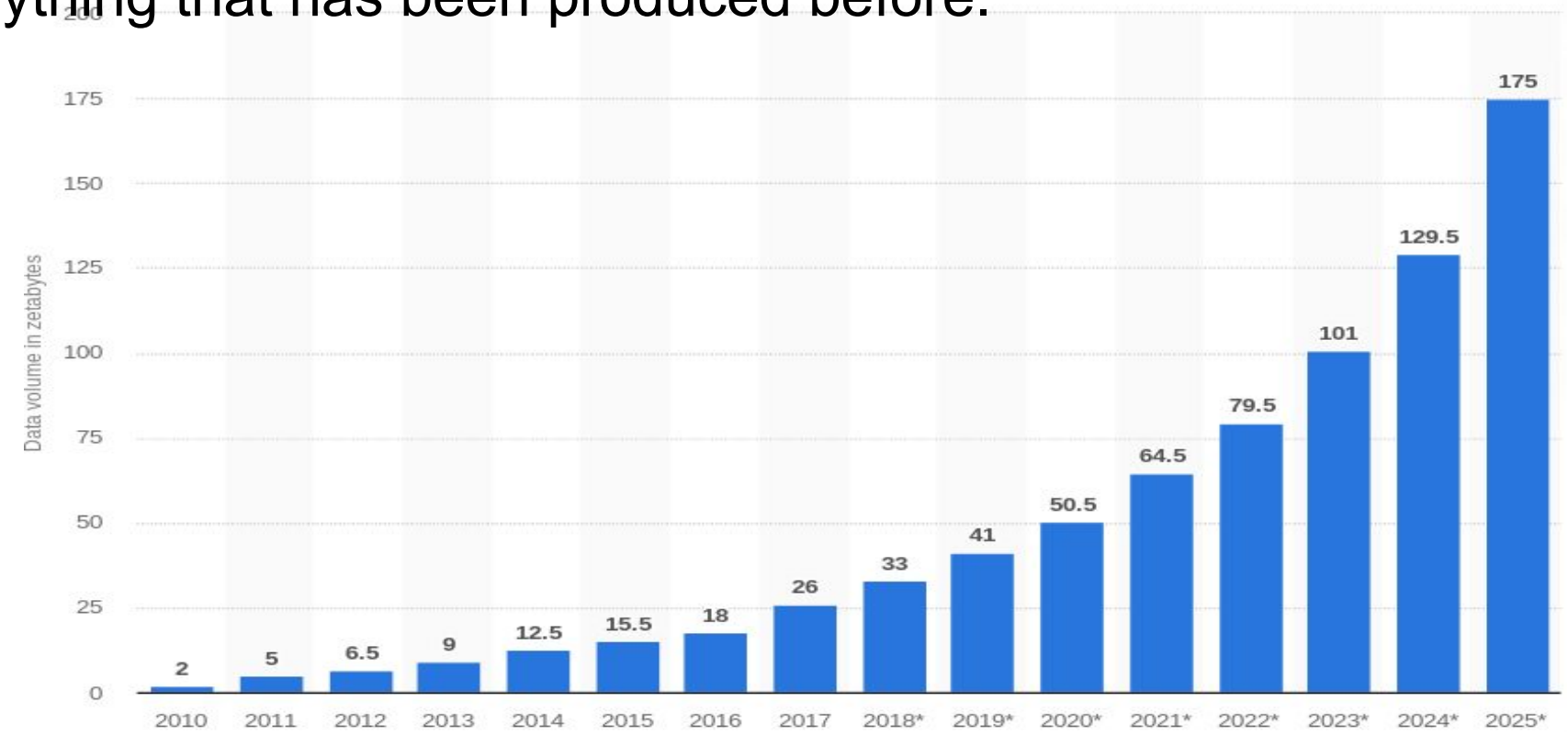
What is big data?

Context

Exponential growth in the amount of data

The growing volume of data

- The quantity of digital data produced doubles every 2 years.
- In other words, we have produced as much digital data in the last 2 year as anything that has been produced before.



source:
https://www.statista.com/statistics/871513/worldwide-data-created/

Volume

Volume: the evolution of the quantity of data stored and exchanged over time

Table 1: Data Measurement Units

Unit	Abbreviation	Decimal Value	Binary Value	Decimal Size
bit	b	0 or 1	0 or 1	1/8 of a byte
byte	B	8 bits	8 bits	1 byte
kilobyte	KB	$1{,}000^1$ bytes	$1{,}024^1$ bytes	1,000 bytes
megabyte	MB	$1{,}000^2$ bytes	$1{,}024^2$ bytes	1,000,000 bytes
gigabyte	GB	$1{,}000^3$ bytes	$1{,}024^3$ bytes	1,000,000,000 bytes
terabyte	TB	$1{,}000^4$ bytes	$1{,}024^4$ bytes	1,000,000,000,000 bytes
petabyte	PB	$1{,}000^5$ bytes	$1{,}024^5$ bytes	1,000,000,000,000,000 bytes
exabyte	EB	$1{,}000^6$ bytes	$1{,}024^6$ bytes	1,000,000,000,000,000,000 bytes
zettabyte	ZB	$1{,}000^7$ bytes	$1{,}024^7$ bytes	1,000,000,000,000,000,000,000 bytes
yottabyte	YB	$1{,}000^8$ bytes	$1{,}024^8$ bytes	1,000,000,000,000,000,000,000,000 bytes

Distribution of data centers around the world

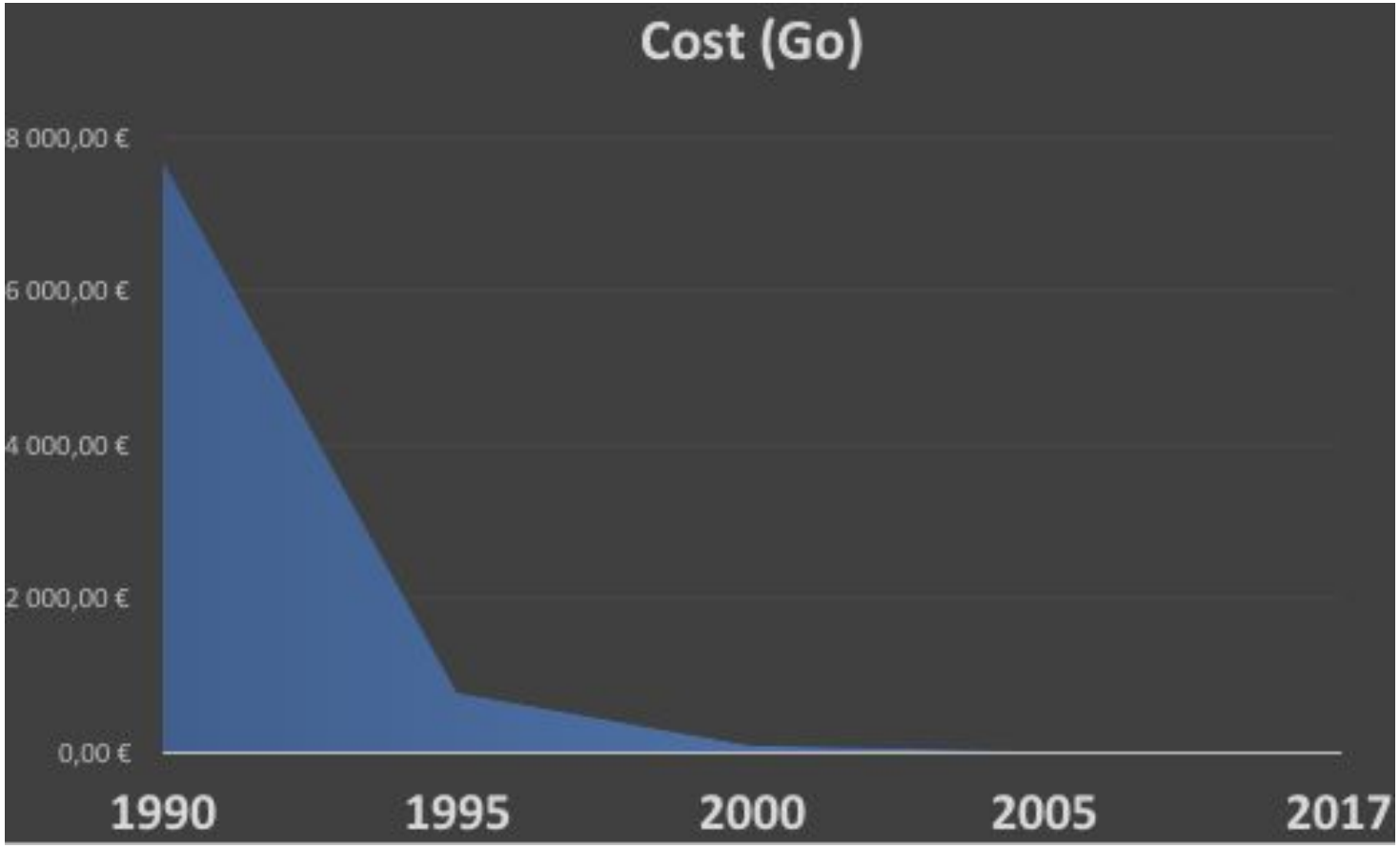
400+
Cloud Providers

5,000+
Data Centers

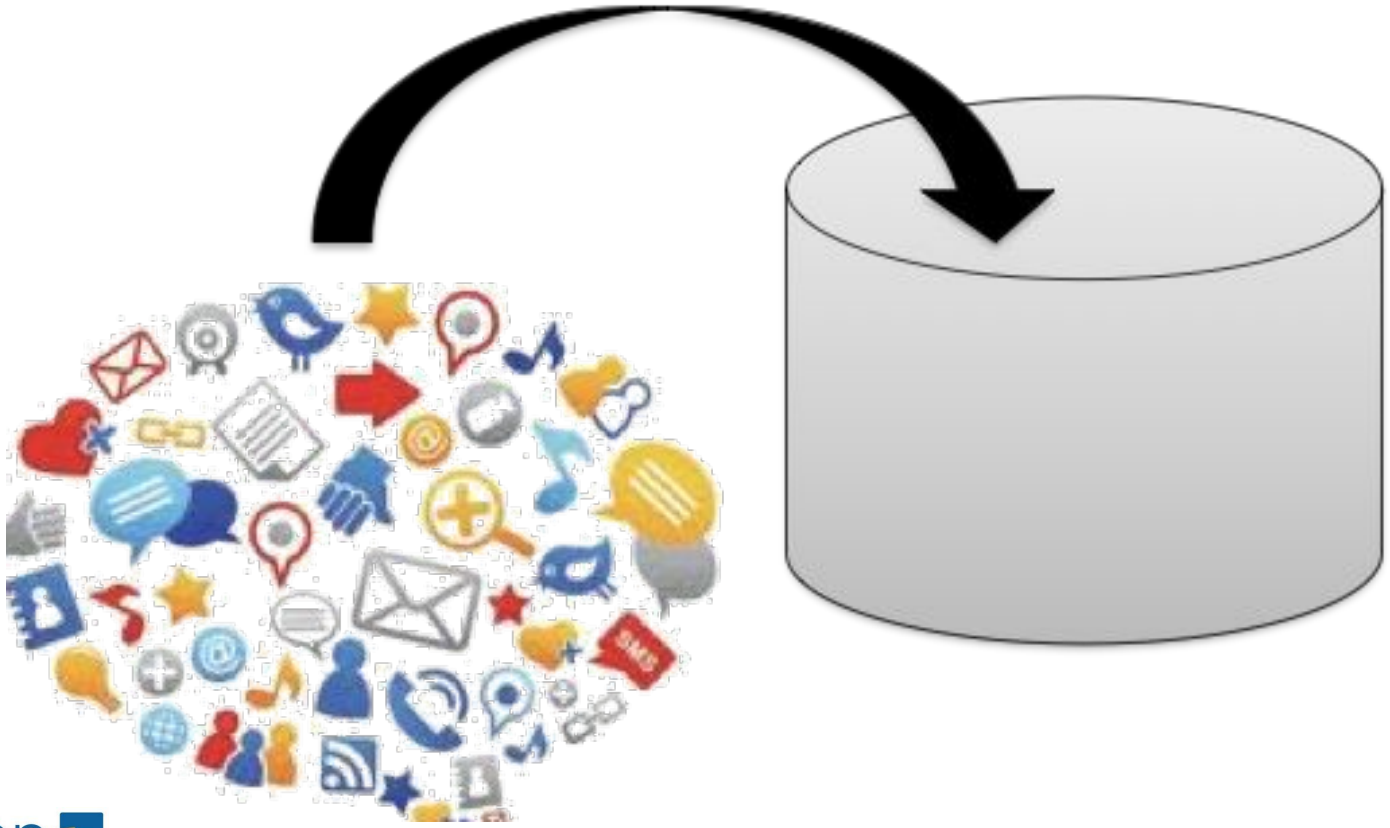
6,000+
Networks

https://www.datacentermap.com/

Cost


Cost (Go)
8 000,00 €
6 000,00 €
4 000,00 €
2 000,00 €
0,00 €
1990
1995
2000
2005
2017

Variety: format and sources

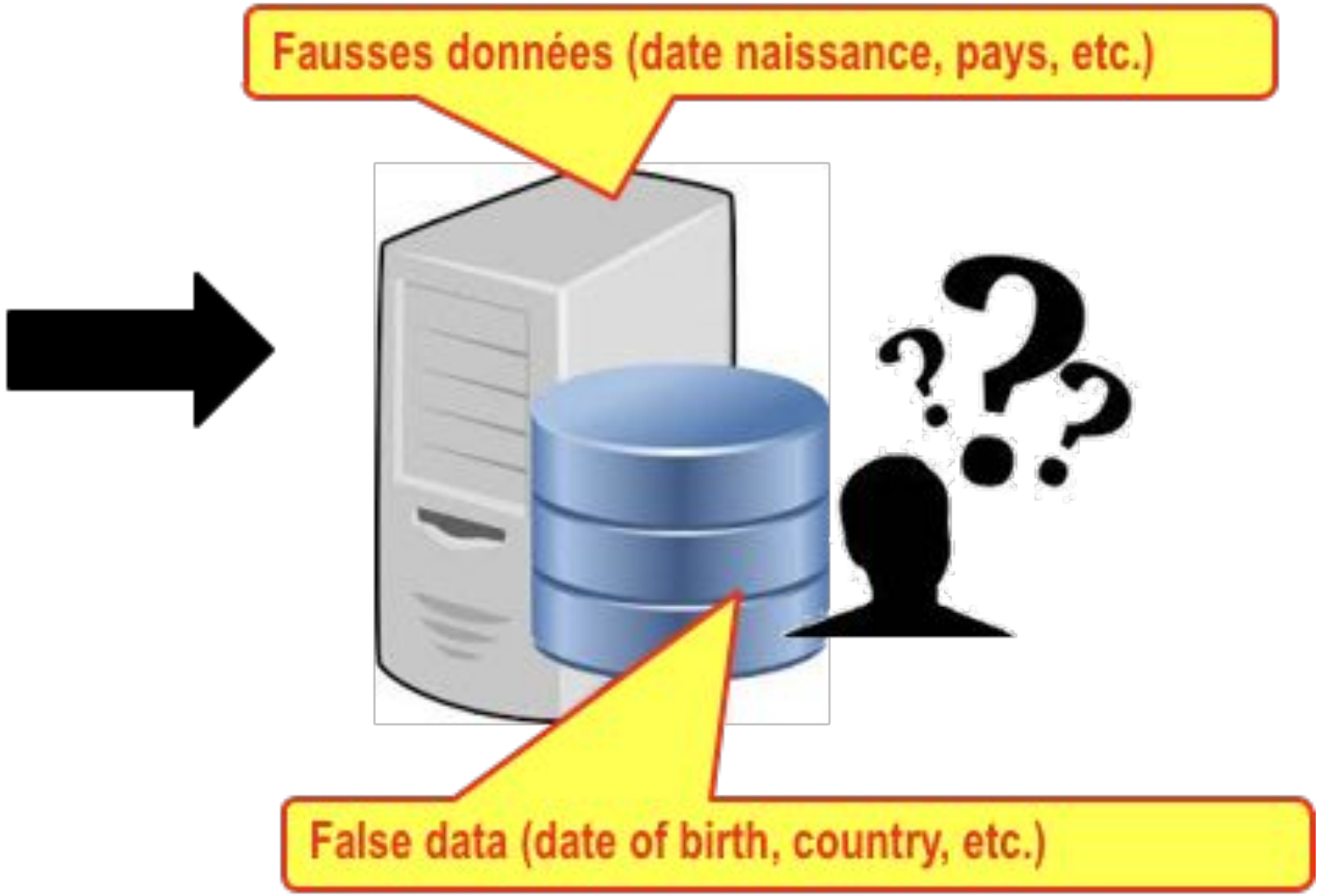


Velocity

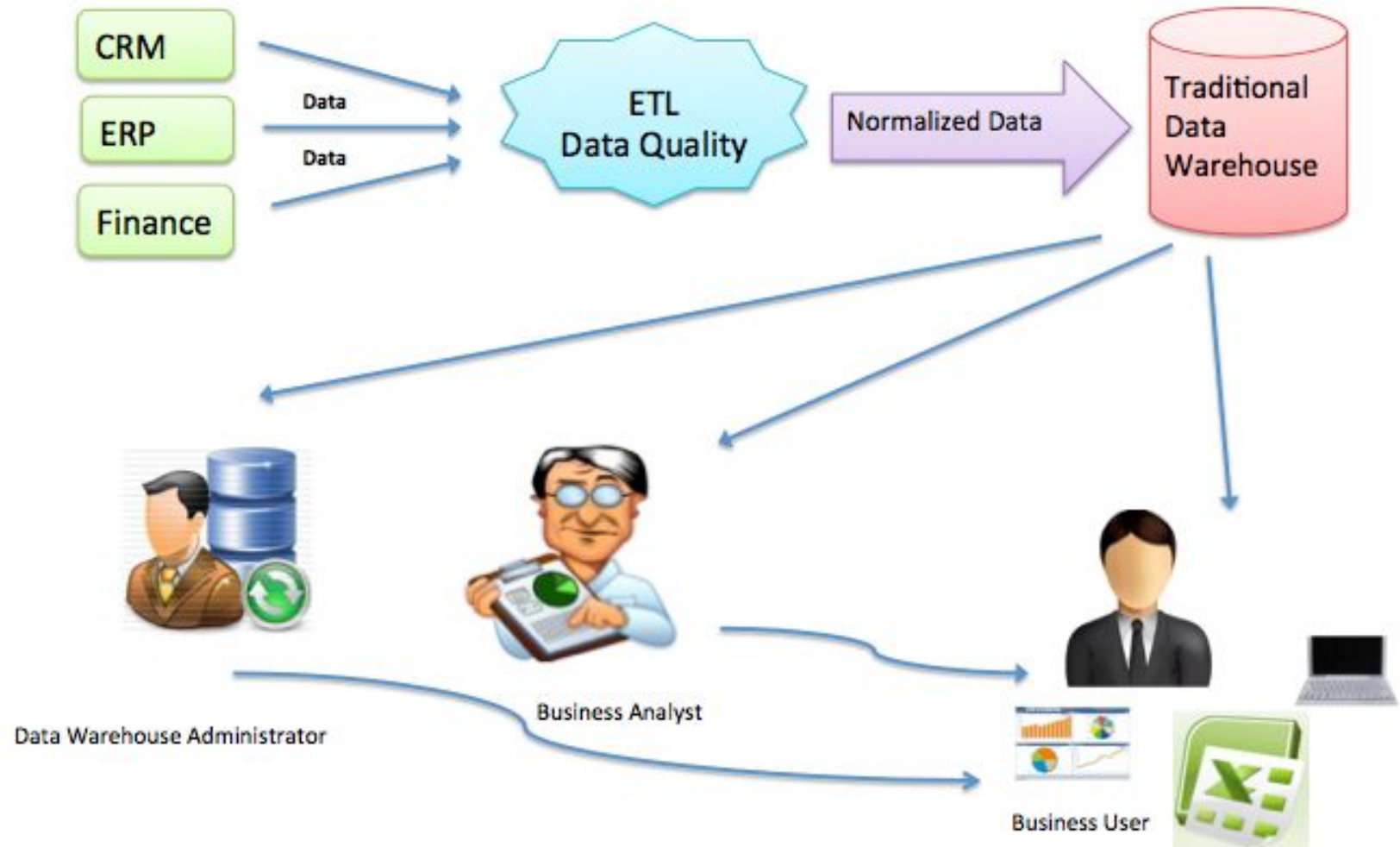
Veracity

fake profiles on social networks

Multiple profiles



Decision-making: old method



Decision-making vs 3V

The classic approach incompatible with the 3Vs of BigData:

- **Volume**: warehouses are designed to manage GB or TB of data while the exponential growth of data leads us to Po or Eo
- **Variety**: the number of types, including semi or unstructured textual data, is increasing
- **Velocity** (Speed): data is created faster and faster and requires real-time processing

10v of Big Data

What does this data look like?

- Generally, these are not structured data (as in traditional BD) standardized (3FN or FN)
- This is semi or unstructured data
  - Tweets
  - Server logs
  - Pages web
  - Linked data
  - CSV, JSON files, etc. bulky

Context: what about business?

- It is estimated that the volume of professional data doubles every 1.2 years
- ¾ of decision-makers believe that Big Data will significantly affect their storage systems
- The successes of web giants continue to demonstrate the value generated by data
- The technological innovations of ‘Big Data’, the result of solutions developed by web giants, bring new possibilities
- In Europe, the use of Big Data to improve the efficiency of “processing” would save $100 billion

Netflix

- Recommendations based on user data

"there are 33 million different versions of Netflix. 75% of the videos viewed come from Netflix's suggestions to its users

- Big data motivated the purchase of the rights (for US $100 million) and the shooting of the House of Cards series

(popularity of similar series, choice of producer, director, actors)

House of Cards brought 2 million new users to the US and 1 million outside the USA

Source: http://www.infoq.com/fr/news/2014/01/netflix-bigdata-decisions

Exploitation of social networks

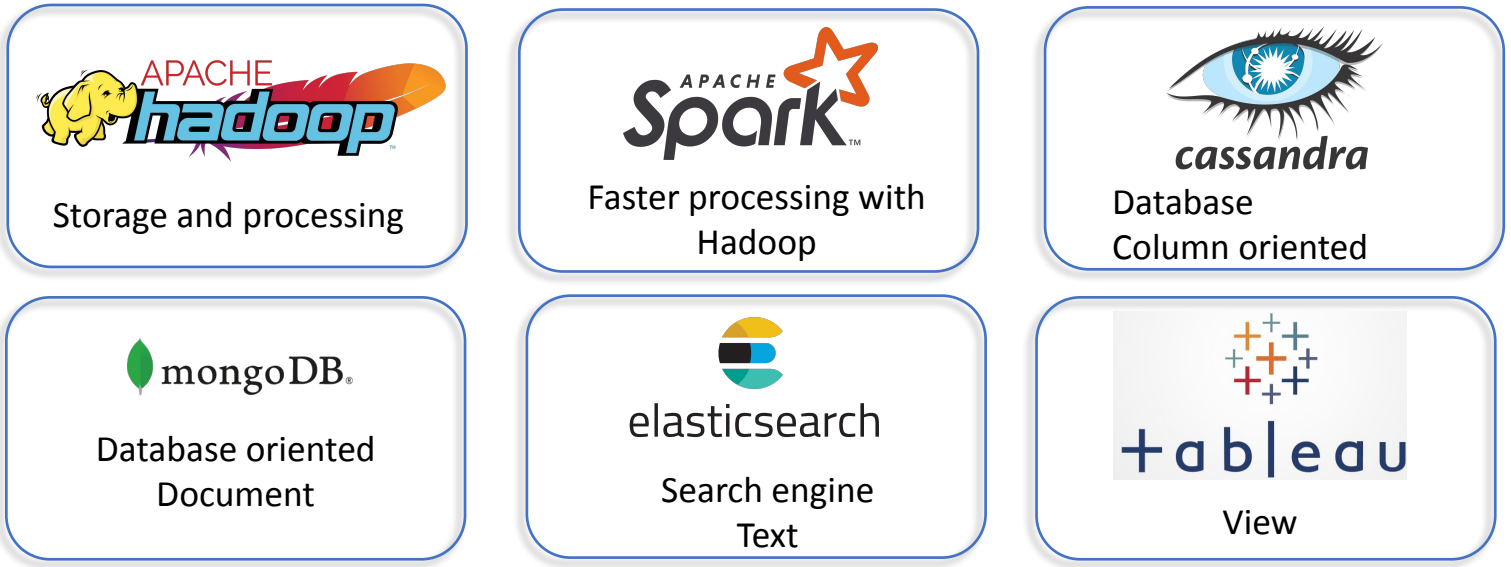
- Understanding the audience
  - What is the value of the audience?
  - Who are the followers (Twitter account for example)?
  - What are the subscribers' interests?
  - Identification of the most important tweets
- E-reputation
  - Monitoring of opinion
- Participatory innovation
- Ideas box on the Internet
  - Analysis of opinion for the creation of innovative products and services

Data analysis

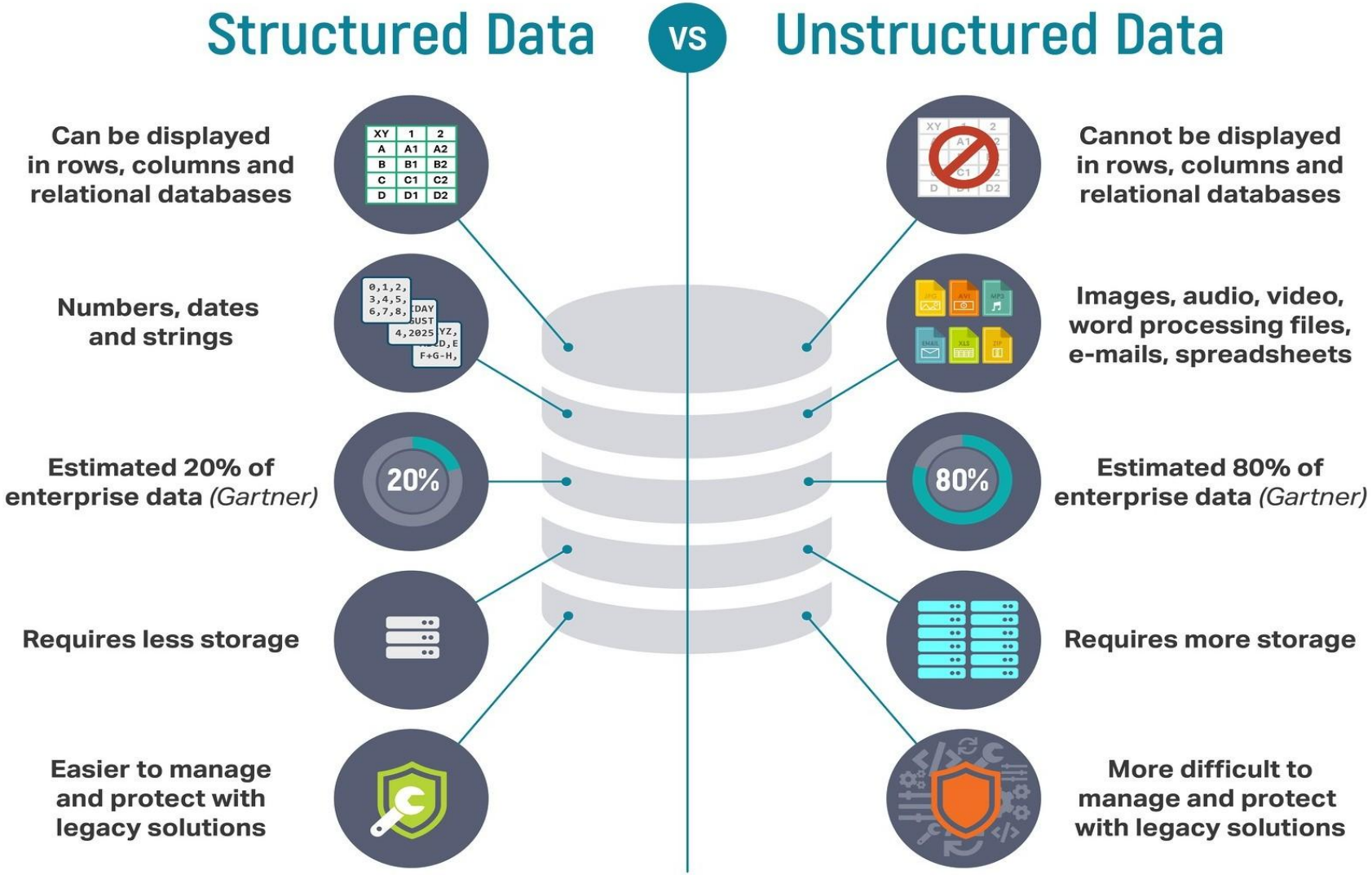
Characteristics	Traditional analysis	Big data analysis
Data characteristics	Structured data Volume measured in megabytes and gigabytes	All data: structured, semi-structured and unstructured Volume measured in terabytes and petabytes
Purpose of analysis	Sample of a known population	The whole population
Facts and conclusions	Answers to the questions already identified	New and unexpected facts and discoveries
Knowledge requirements	Analytical tools and techniques, reporting	Advanced analytical, mathematical and statistical knowledge and computer skills

1Lukić, Jelena. (2017). The impact of big data technologies on competitive advantage of companies. Facta Universities: series Economics and Organization.

Big data technologies



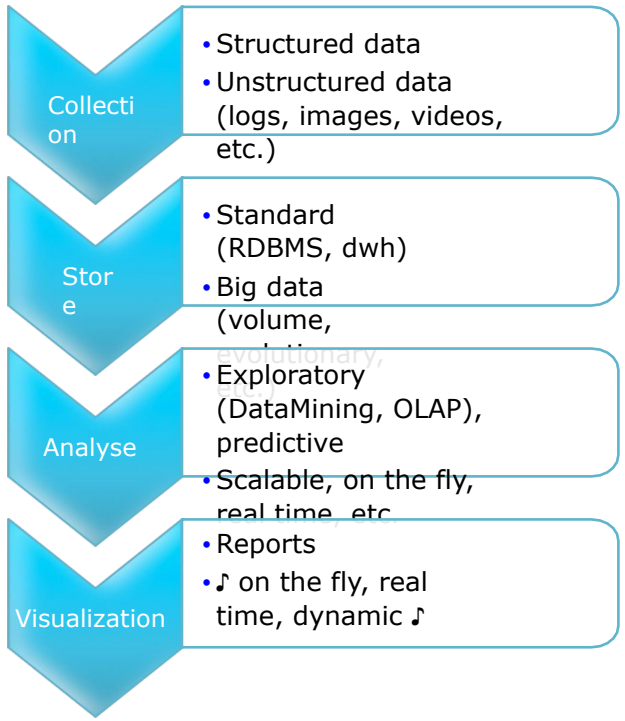
Structured vs unstructured data



source:
https://lawtomated.com/structured-data-vs-unstructured-data-what

Big data challenges

- Take into account exponential growth in data volume
- Manage the variety of data and extract relevant information (value)
- Real-time processing
- Make information available as soon as possible
- Anticipating and facilitating decision-making



Context: consequences

- Unprecedented volumes to manage involve:
  - **Heterogeneous, complex and often linked data**
    - Produced by sometimes different applications,
    - By different users,
    - With explicit links (e.g. quotes, URL anchors, etc.) or implicit links (to be extracted or learned)
  - **Many servers / clusters**
    - A single server cannot store this amount of information, guarantee access times for a large number of users, do quick calculations, etc.
  - **Need to distribute calculations and data:**
    - Like many servers / clusters, need algorithms to calculate and distribute data on a large scale

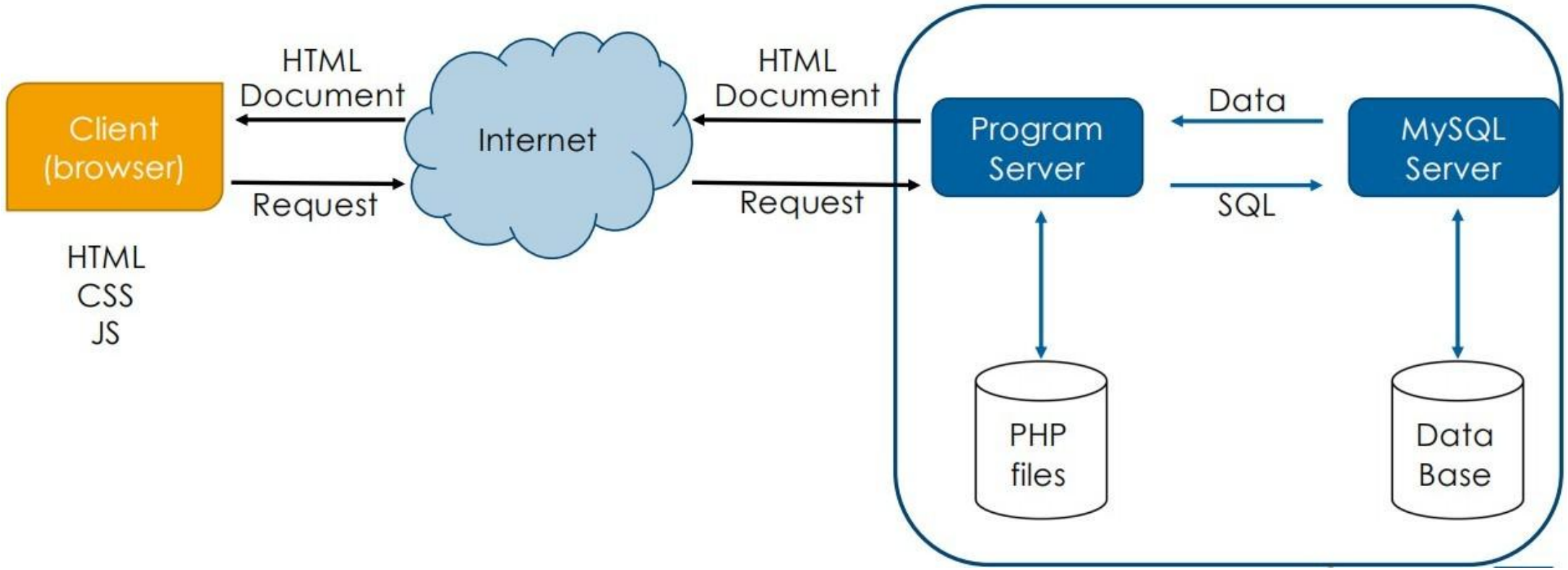
Reaching Limits of an SGBD

Reaching the limits of traditional databases

- Suppose you are hired to create a simple web analysis application. The goal is:
  - Track the number of page views for any URL the client wishes to monitor.
  - Inform you at any time of the first 100 URLs in terms of page views
  - The client's web page sends a Ping to the application's web server with its URL each time a page is viewed.

Column name	Type
id	integer
user_id	integer
url	varchar(255)
pageviews	bigint

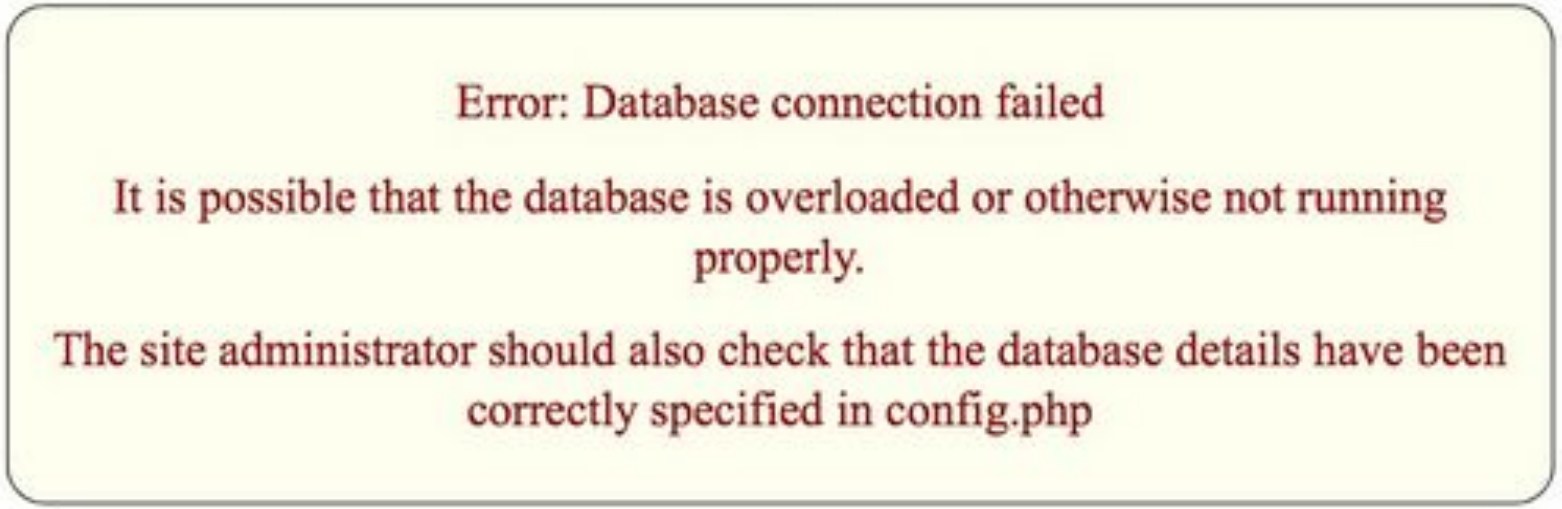
Reaching the limits of traditional databases

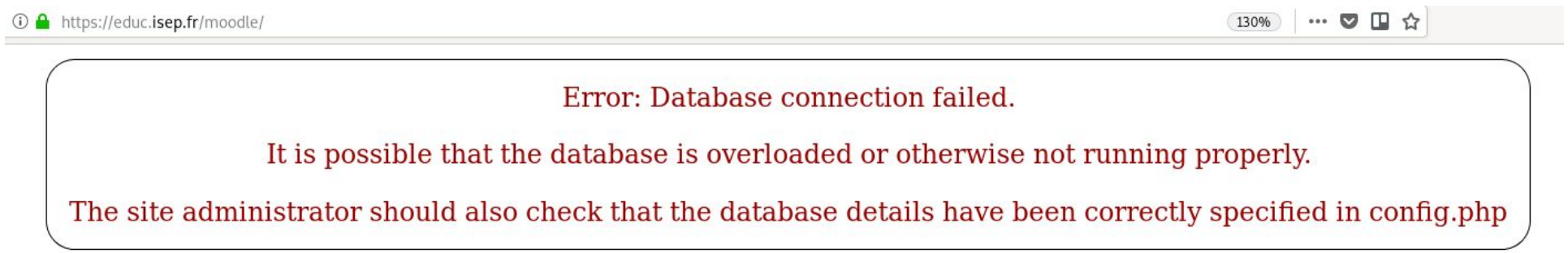


Reaching the limits of traditional databases

The web analytics product is a huge success, and traffic to your application is increasing exponentially.

Your company is having a big party, but in the middle of the celebration, you start receiving a lot of e-mails from your surveillance system. They all say the same thing: "wait time error when inserting into the database."





You check the logs and the problem is obvious. The database cannot keep up with the workload, causing timeouts on write requests to increment pageviews.

Reaching the Limits of Traditional Databases

Timeout expired with SQL Server insert

Asked 1 year, 8 months ago Active 1 year, 8 months ago Viewed 592 times

0

I am trying to insert 200,000 documents from a folder into the SQL Server database into a `varbinary` column. I get a timeout expiration message after inserting 80,000 documents. Average file size is about 250kb and max file size is 50MB. I am running this c# program on the server where the database is located.

Please suggest.

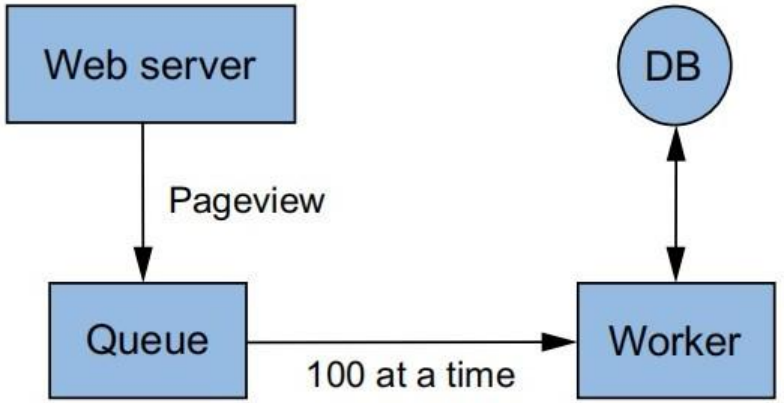
For this, I would suggest using SQLBulkCopy methods, since it should be able to handle data insertion much more easily. Further, as others have pointed out, you might want to increase the timeout condition for your command.

share improve this answer

answered Mar 25 '18 at 11:01

Reaching the Limits of Traditional Databases

Instead of allowing the web server to directly access the database, you insert a queue between the web server and the database.



Reaching the Limits of Traditional Databases

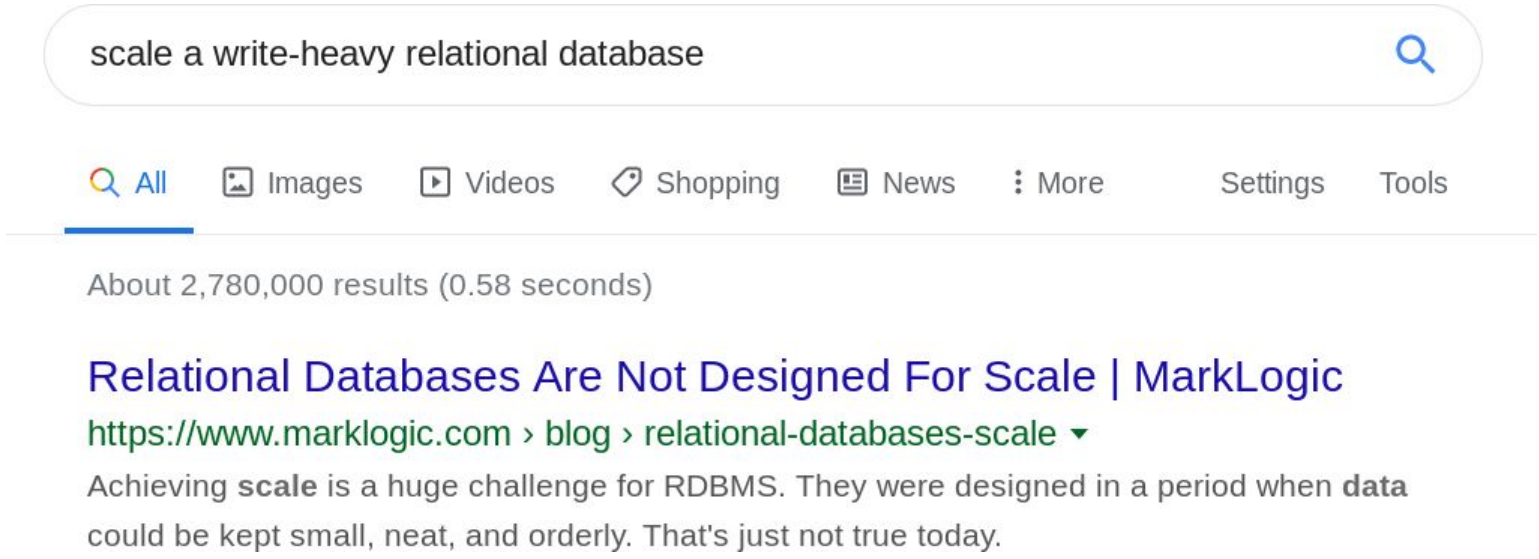
Unfortunately, adding a queue and doing batch updates was only a stopgap solution for the scalability problem.

Your app continues to gain popularity, and once again the database is overloaded.

Your workers can't keep up with the writes, so you try to add more workers to parallelize the updates.

Unfortunately, this doesn't help; the database is clearly the bottleneck.

Reaching the Limits of Traditional Databases



RDBMS vs Data Distribution

- In this simple example, we have the limitations of an RDBMS
- Features in RDBMS
  - Joins between tables
  - Rich query language
  - Strong integrity constraints
- Limitations in the distributed context:
  - **How to distribute/partition data**
  - Links between entities -> Same server
  - But the more links we have, the more complex the data placement is

Principles of distributed systems

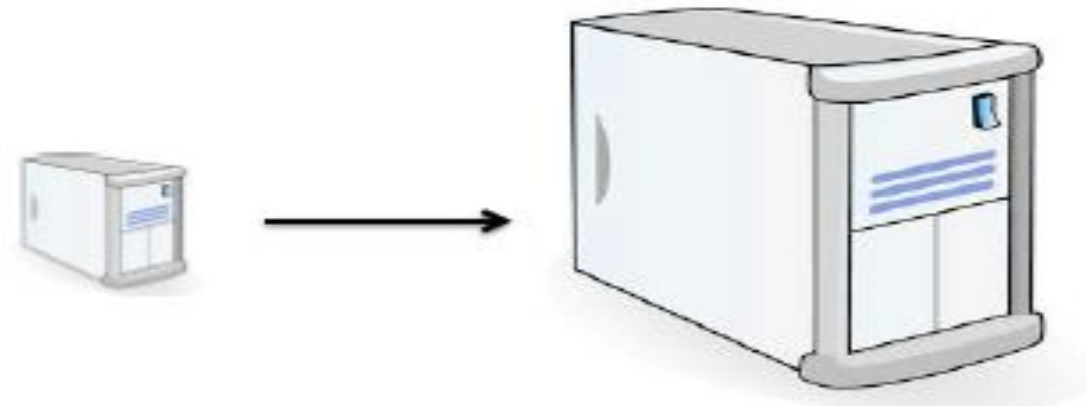
Extensibility

- Scalability is the property of a system, network or process that demonstrates its ability to handle large workloads flexibly or to be expanded without difficulty
- Two types:
  - Vertical
  - Horizontal

Definition of scalability: A system is scalable if its performance is proportional to the resources allocated to it: network transfers are a limiting factor in scalability

Vertical scalability

- In practice
  - Replace servers with more powerful machines
- Significant cost
- Easy installation



Horizontal scalability

- Added ordinary machines
- Lower cost
- More delicate installation

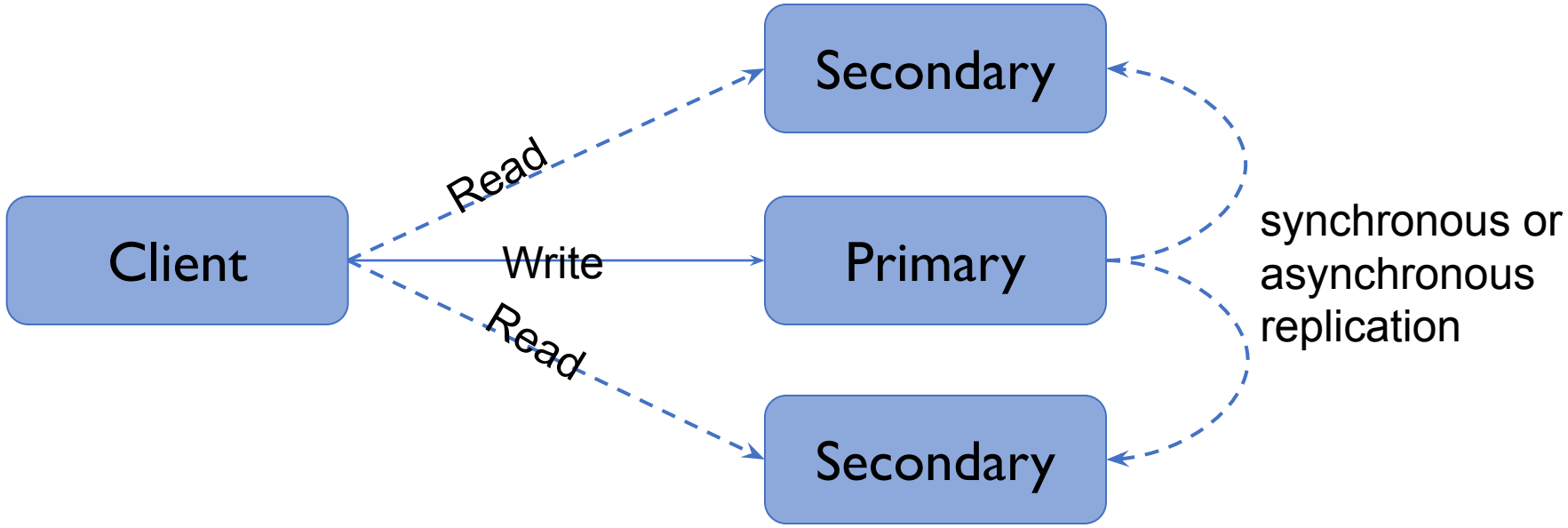
Volume Scaling for an RDBMS

- Scaling issues when the dataset is too large
- RDBMS were not designed to be distributed
- Distributed database solutions (multi-nodes) => “horizontal scaling”
- Different approaches:
  - Master-slave
  - Partitioning

Volume Scaling for RDBMS

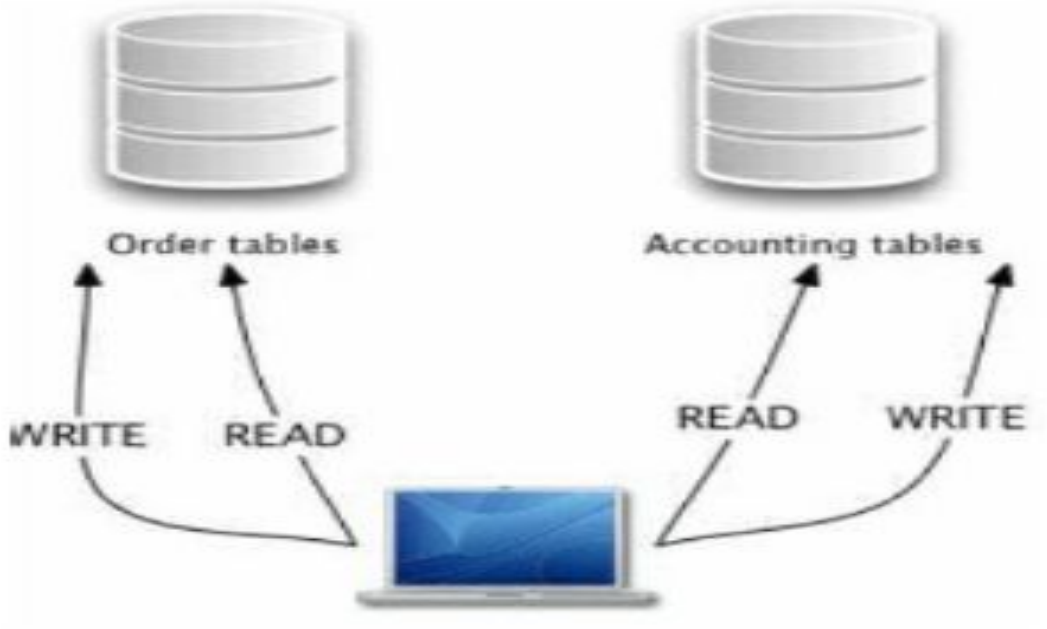
- Master/slave
  - All writes are sent to the master
  - Reads may be incorrect until writes are propagated
  - Data volume can cause issues if the master needs to duplicate data to slaves
- Partitioning
  - Efficient for reads and writes
  - Lack of transparency, applications must adapt to partitioning
  - Loss of referential constraints (relations) and join between partitions

Primary/Secondary Nodes

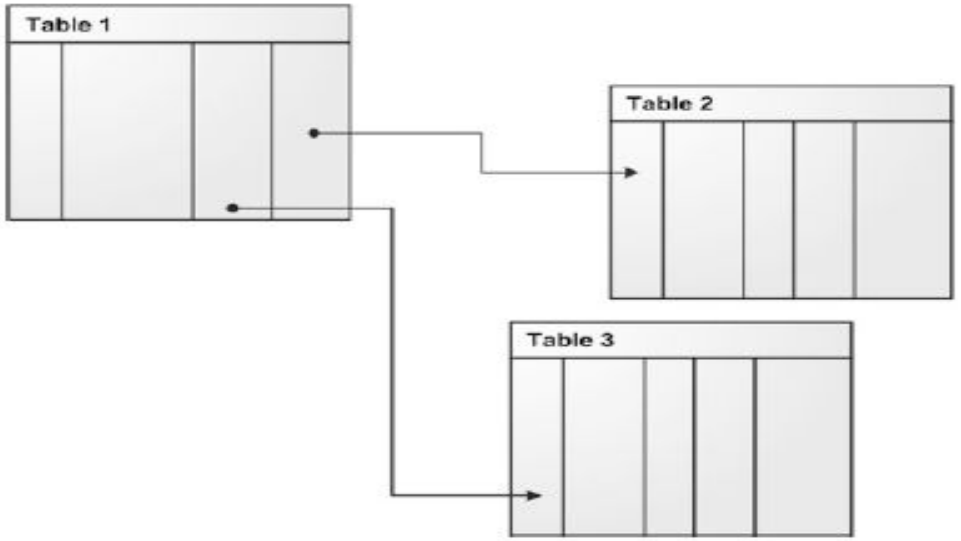


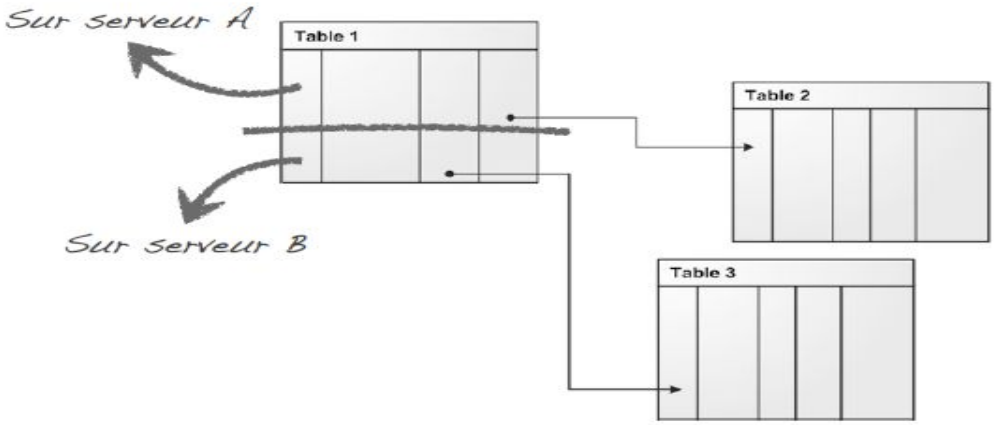
Vertical partitioning

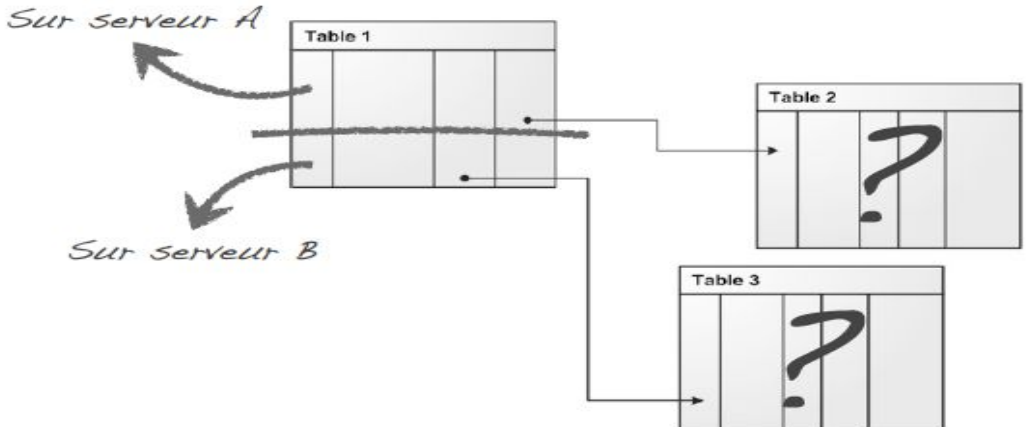
- Servers store different tables of a database on different machines

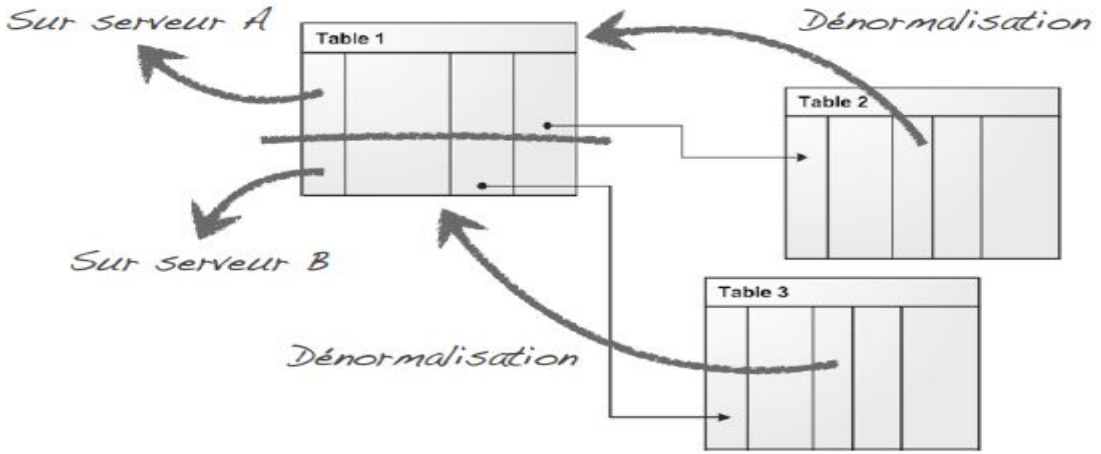


Horizontal partitioning





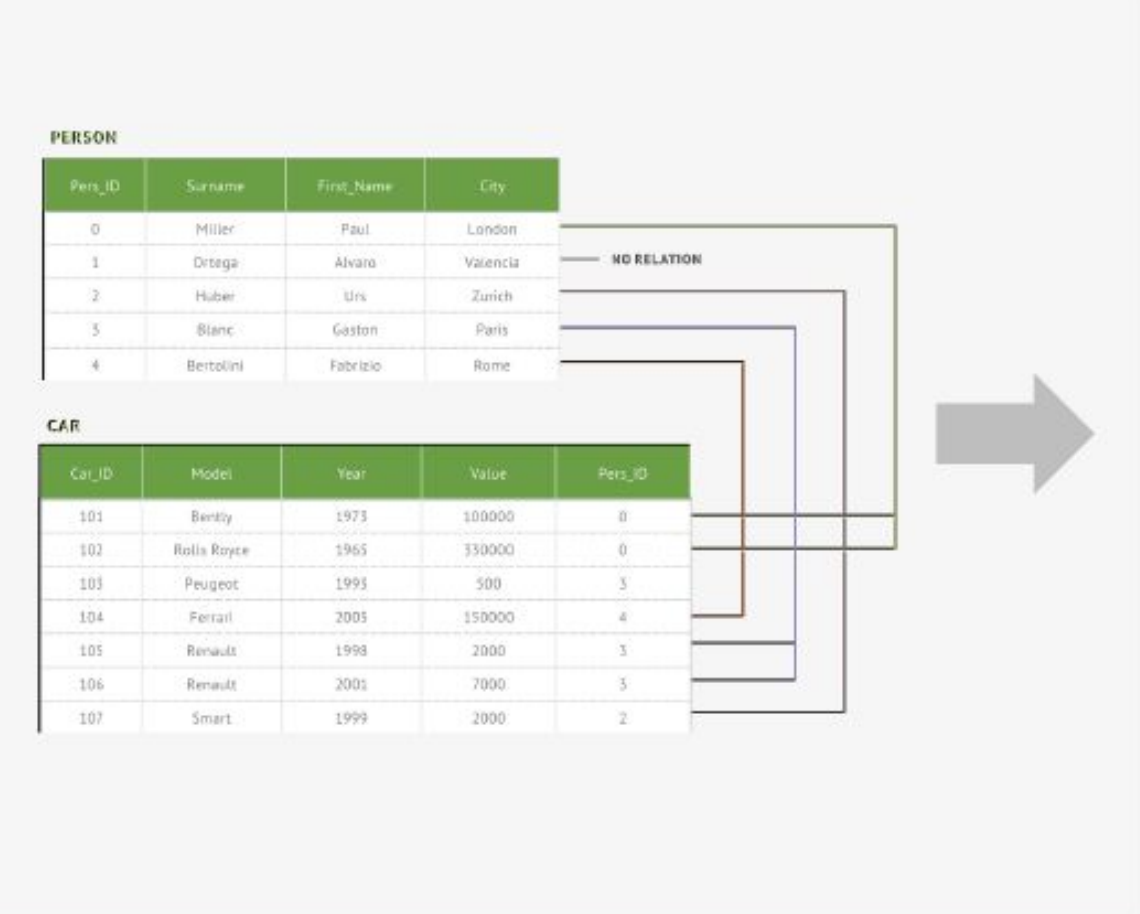




We then lose a lot of the interest in relationships!

Denormalization

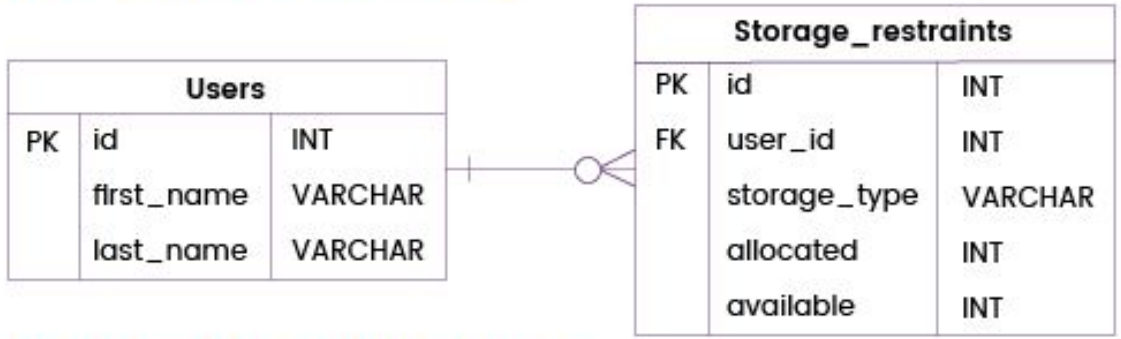
Column denormalization: allows you to repeat a column in multiple tables to avoid having to create joins between tables.



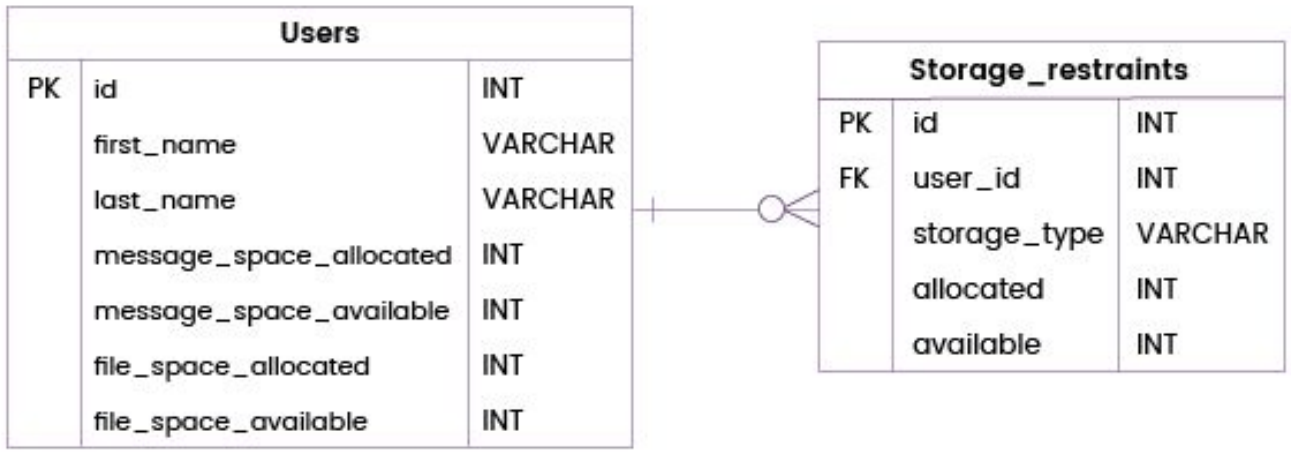

```
{
  first_name: 'Paul',
  surname: 'Miller',
  city: 'London',
  location:
[45.123,47.232],
  cars: [
    { model: 'Bentley',
      year: 1973,
      value: 100000, … },
    { model: 'Rolls Royce',
      year: 1965,
      value: 330000, … }
  ]
}
```

Denormalization

Normalized database



Denormalized database



ACID vs BASE

ACID properties for transactions

- ***A**tomicity*: *a transaction is carried out entirely or not at all*
- ***C**onsistency*: *the content of a database must be consistent at the start and end of a transaction*
- ***I**solation*: *modifications to a transaction are only visible/modifiable when it has been validated*
- ***D**urability*: *once the transaction is validated, the state of the database is permanent (unaffected by failures or otherwise)*

Distributed systems: BASE model

- ***B**asically **A**vailable*: *minimum guarantee for availability rate in the face of a large quantity of requests*
- ***S**oft-state:* *the state of the system can change over time even without new inputs (this is due to the consistency model).*
- ***E**ventually consistent*: *all replicas reach the same state, and the system becomes consistent at some point, if we stop the inputs*

NoSQL: a solution

- **NoSQL : Not Only SQL**
  - New approach to data storage and management
  - Enables scaling through a highly distributed context
  - Management of complex and heterogeneous data: No schema for objects

- **Does not replace RDBMS !**
  - Huge amount of data (PetaBytes)
  - Need for response time
  - Low data consistency

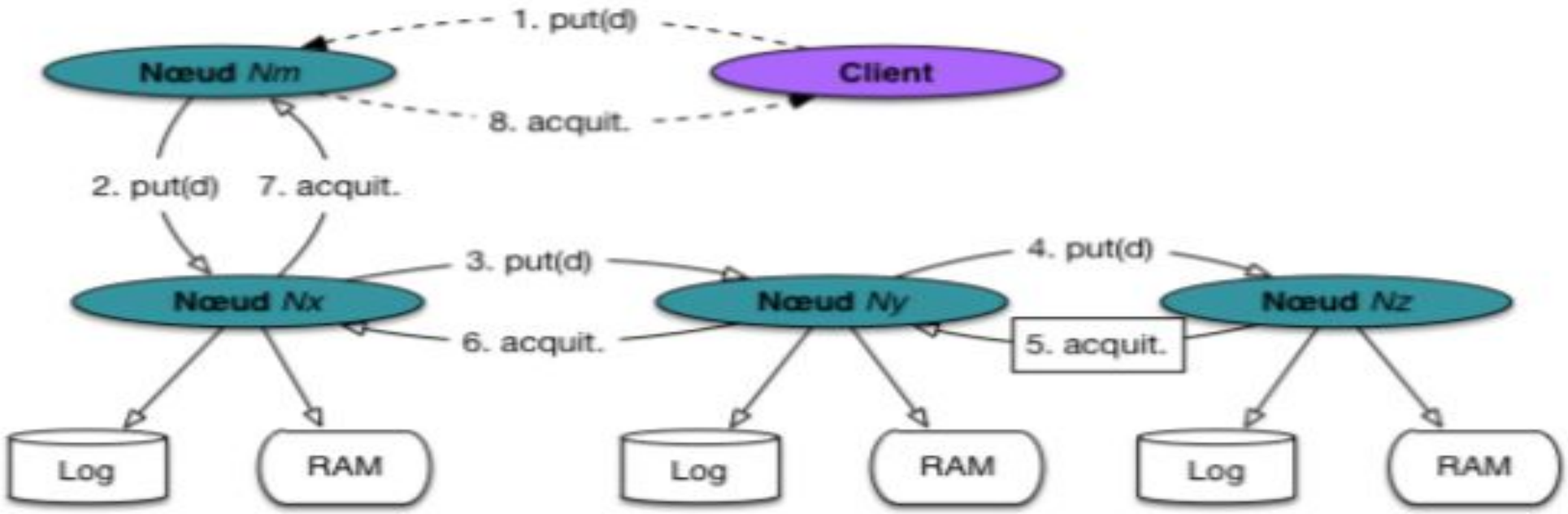
The technical choices of NoSQL

- **Distributed Architecture**
  - Processing and data distribution techniques with or without a master
  - Replication
- **Consistency Level**
  - strong, eventually, weak
- **Abandoning the relational model**
  - need to design applications for specific use of data
- **Transferring complexity to application (client-side)**
  - conflict management

Replication

- Basic principle of NoSQL
  - Failure is the rule (disk, application, system, server, etc.)
    - Redundancy is the answer to dealing with outages.
    - We duplicate: data on different disks
    - We avoid SPOF: Single Point Of Failure
- Replication Objectives
  - Availability (immediate and automatic failure recovery)
  - **read scalability: distribute read requests across multiple machines when data is replicated**
  - **write scalability: concurrent write problems**
  - load distribution
  - data locality
- Main problem with replication
  - **Consistency**

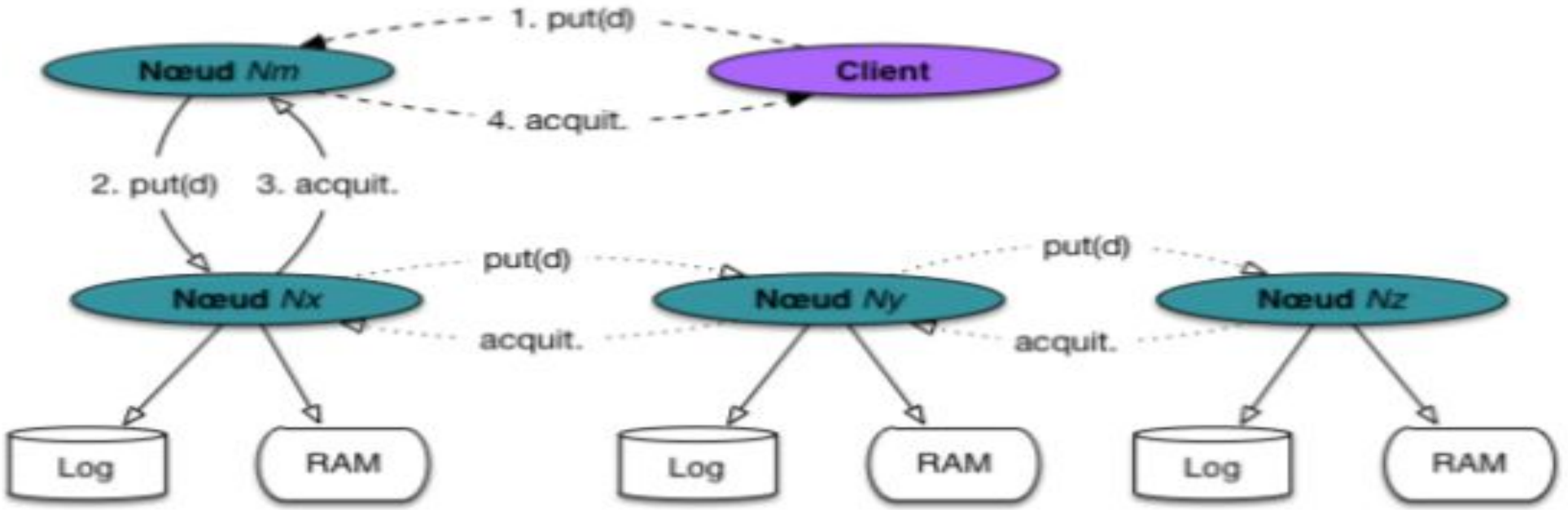
Synchronous replication



Chain of 8 messages for 3 copies, waiting time can be extended in the event of overload of a server for example.

Source : polycopié de P. Rigaux – Bases de Données Documentaires et Distribuées

Asynchronous replication

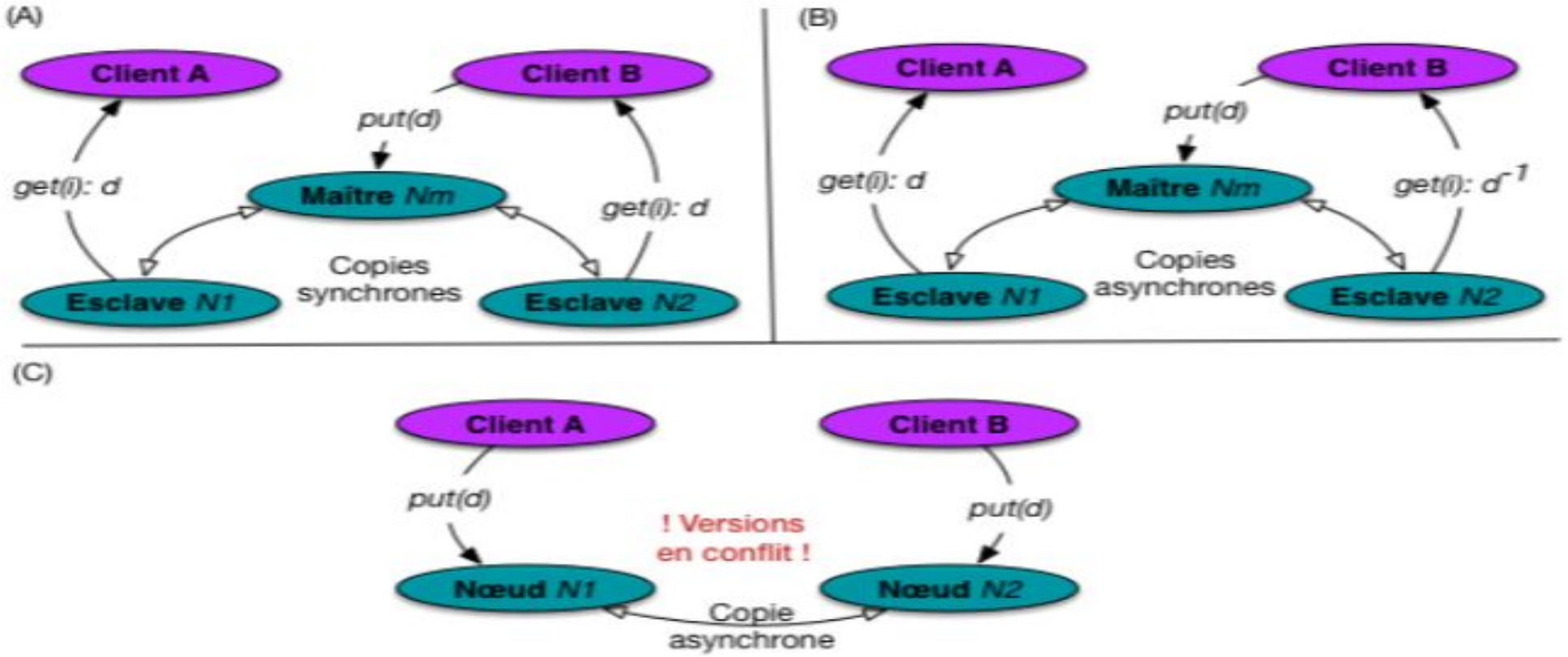


Faster, but:

- the client receives an acknowledgment even though the replication is not complete,
- the client continues its execution even though all copies are not yet updates ; it is therefore possible that a read returns obsolete copies

Consistency

- Consistency depends on 3 factors in NoSQL: system topology (master-slave or without a master) and the asynchronous nature or not of the writes



Consistency

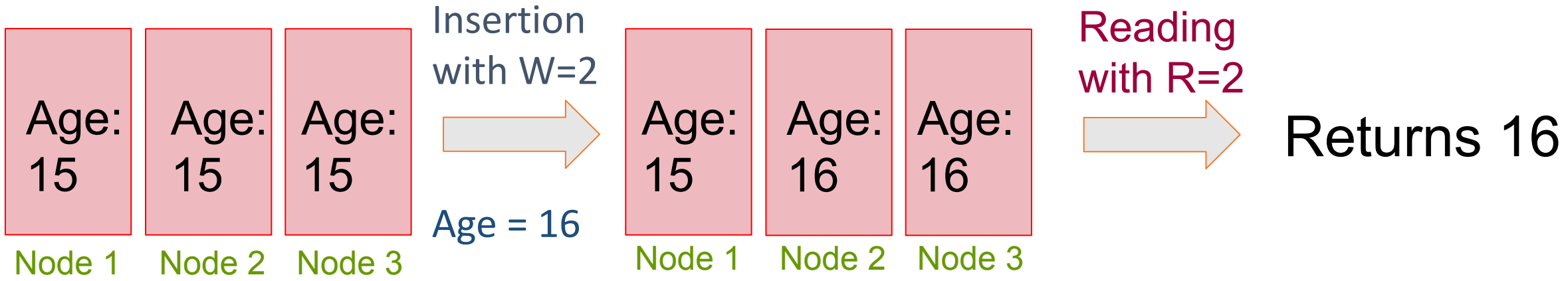
Consistency levels:

- **Strong consistency (ACID properties):** synchronous replication and strong locking mechanism
- **Eventually consistency (Eventual consistency):** the system guarantees eventual convergence towards a coherent state (choice of the majority of NoSQL engines)
- **Low consistency:** promotes data availability, never waiting for read and write operations ⇒ some queries can be performed on old data

Long-term consistency

- To always be consistent:
  - $R + W > N$
  - R = number of nodes that must agree when reading a value
  - W = number of nodes that must validate an insertion
  - N = replication factor
- Example: N = 3

  We only take into account here the nodes concerned



Here we have: W + R > 3

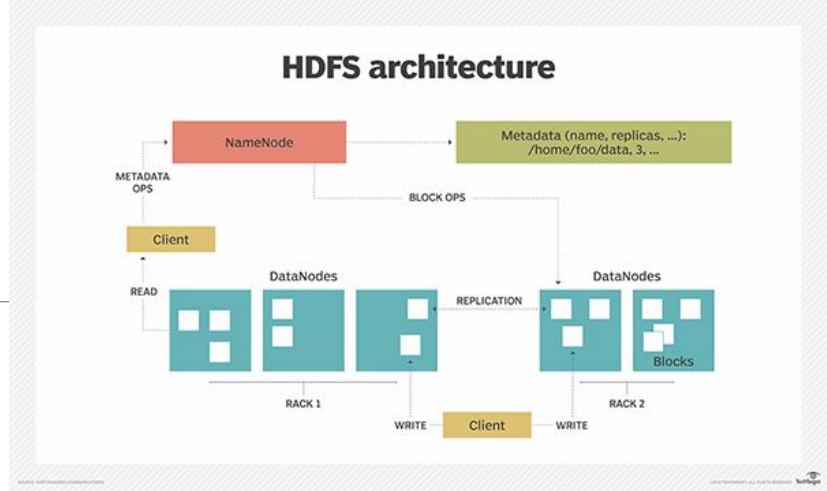
Sharding: Scaling up

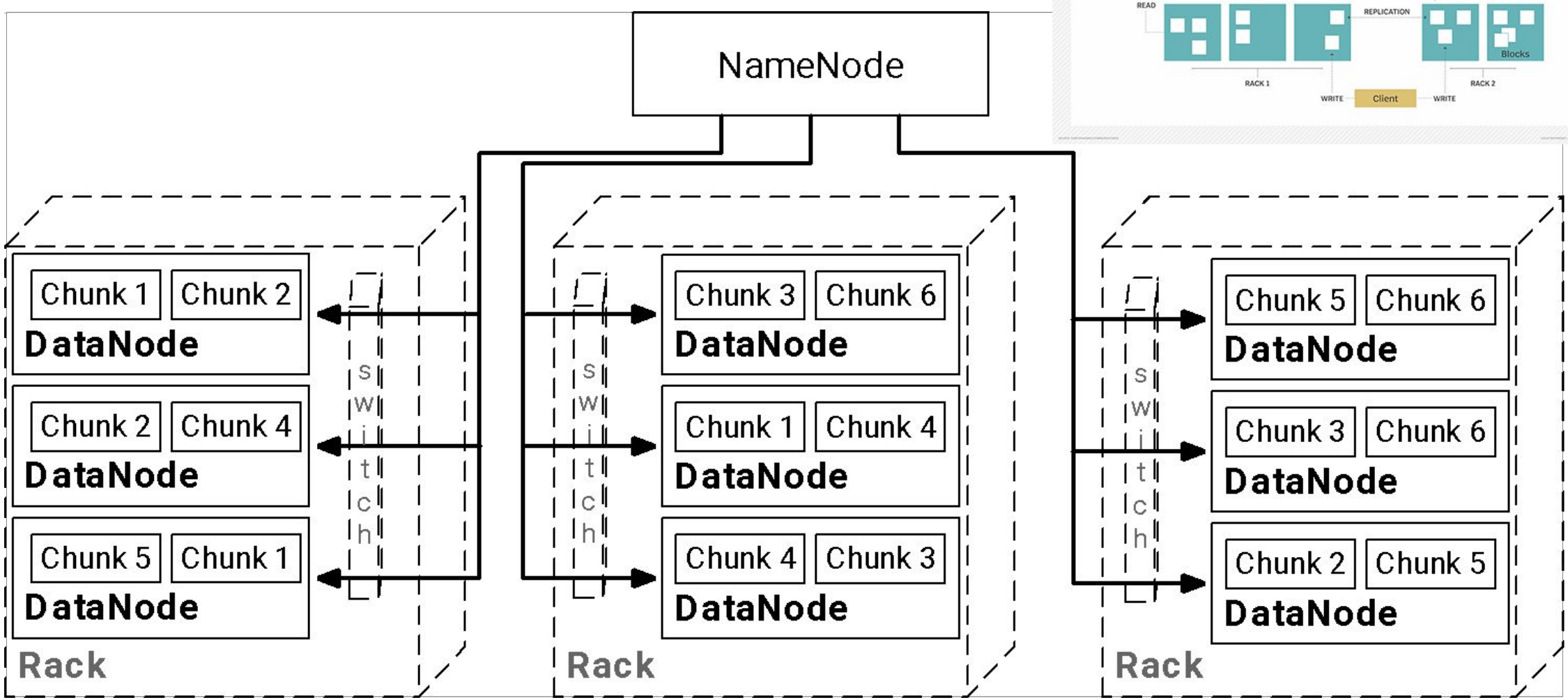
- Distribution of data blocks across a set of servers
- Horizontal partitioning
- Three main techniques:
  1. Based on resource allocation: HDFS
  2. Based on a tree structure: Non-dense index
  3. Hash-based: Consistent Hashing

Sharding : HDFS

- Resource allocation
  - **HDFS =>** Hadoop Distributed File System
    - Depends on server load
    - Distribution, fault tolerance
    - Dynamic and optimized allocation

Sharding : HDFS



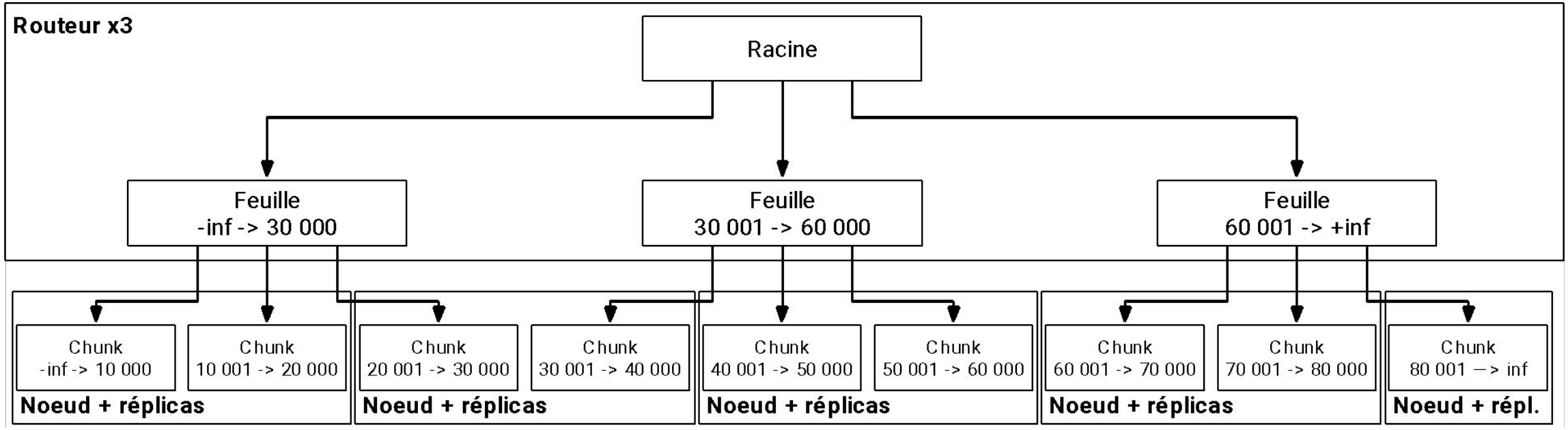


Sharding : Clustered index

- Tree technique
  - **Index non-dense distribué (Distributed sparse index)**
    - Physically sorted data
  - Cut into blocks on the nodes (often 256MB)
  - Distribution, tolerance
  - ➡ Interval queries
  - ⚠️ Choose the right key for sorting

Sharding : Clustered index

- Example

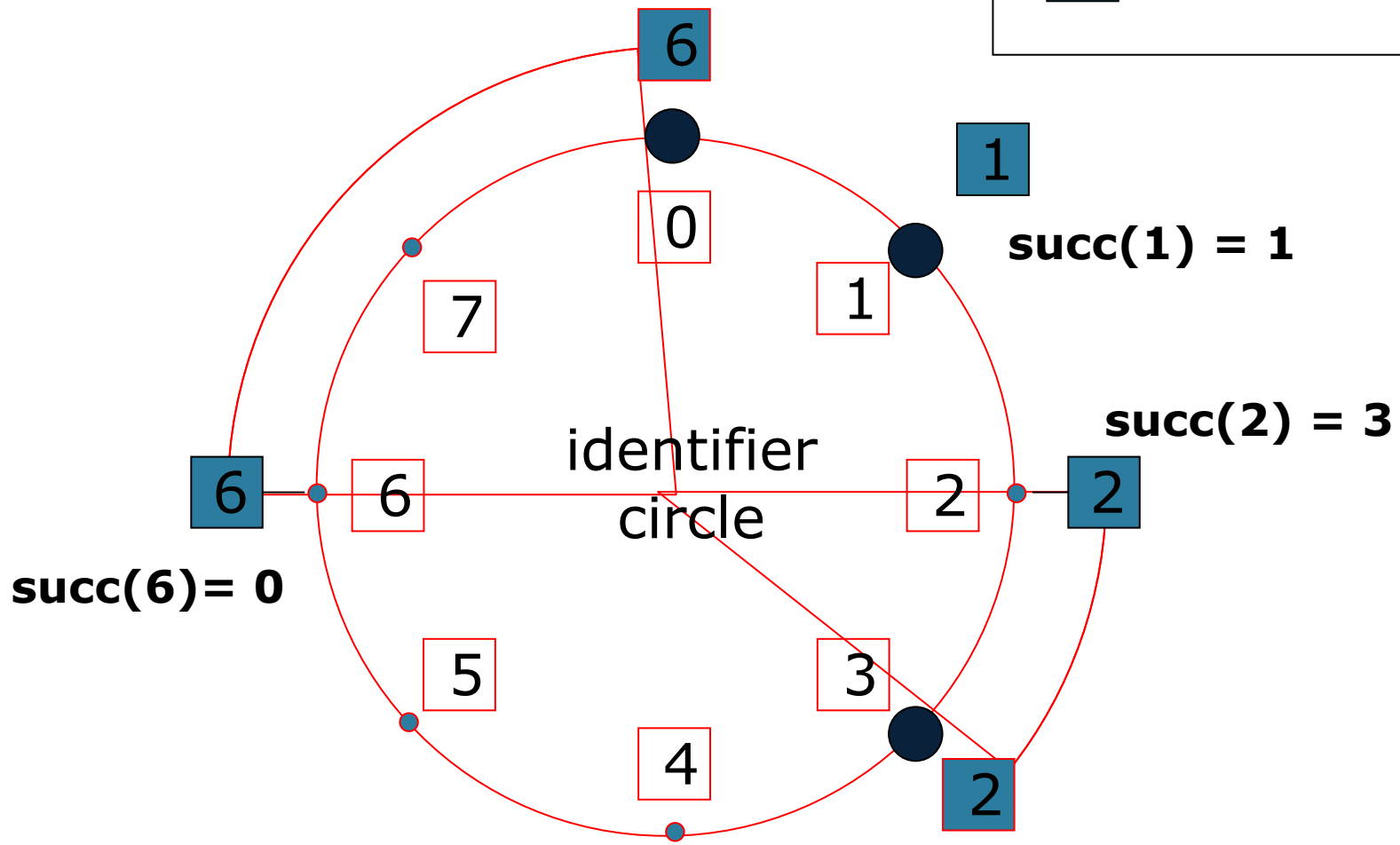


Sharding : Consistent Hashing

- A distributed hash table
- **Consistent Hashing (DHT)**
  - Unique and dynamic hashing technique for data and servers
  - Ring distribution (virtual)
  - No centralized server (everything is client/server)
  - Management autonomy

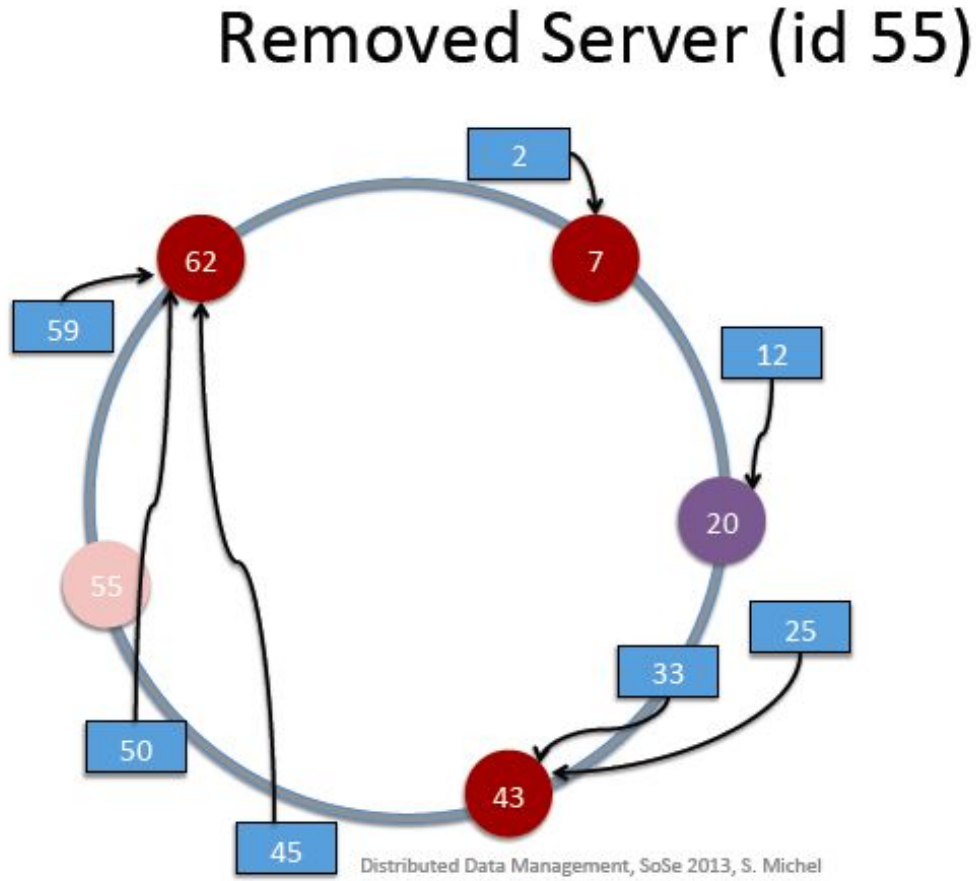
Sharding : Consistent Hashing

- Exemple



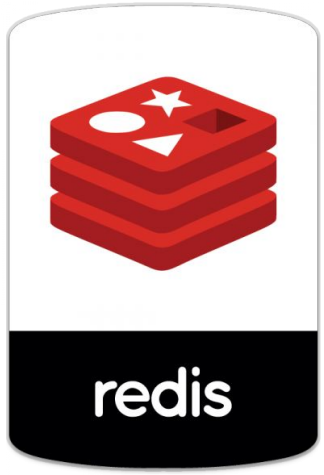
Sharding : Consistent Hashing

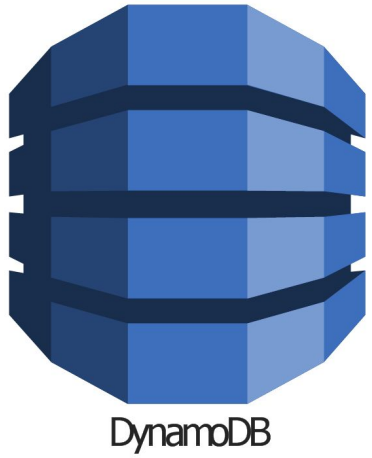
- Exemple



Sharding : Consistent Hashing

- solutions





Elasticity

Self management

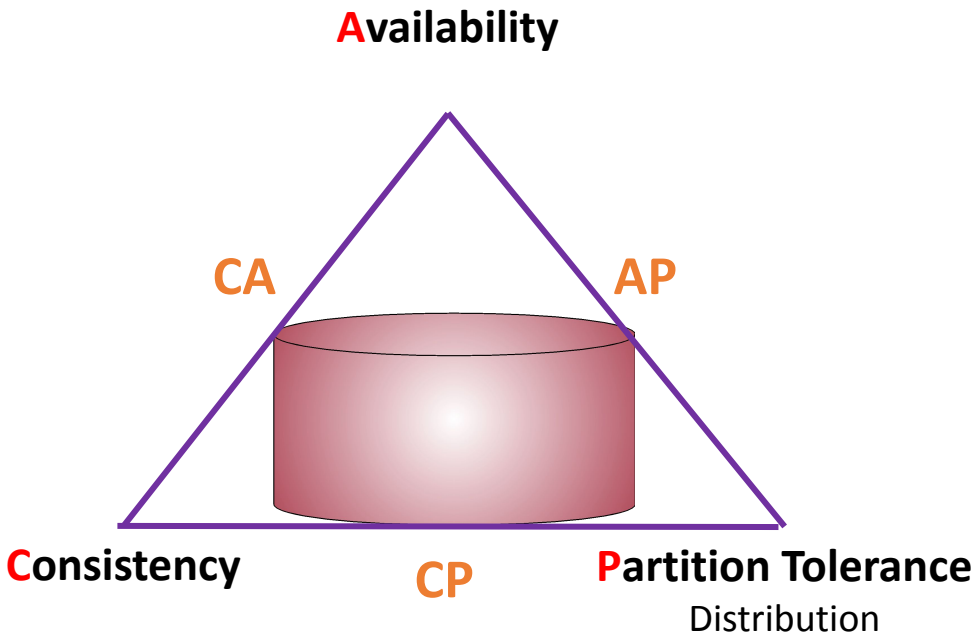
“CAP theorem”

Brewer's theorem (2000)

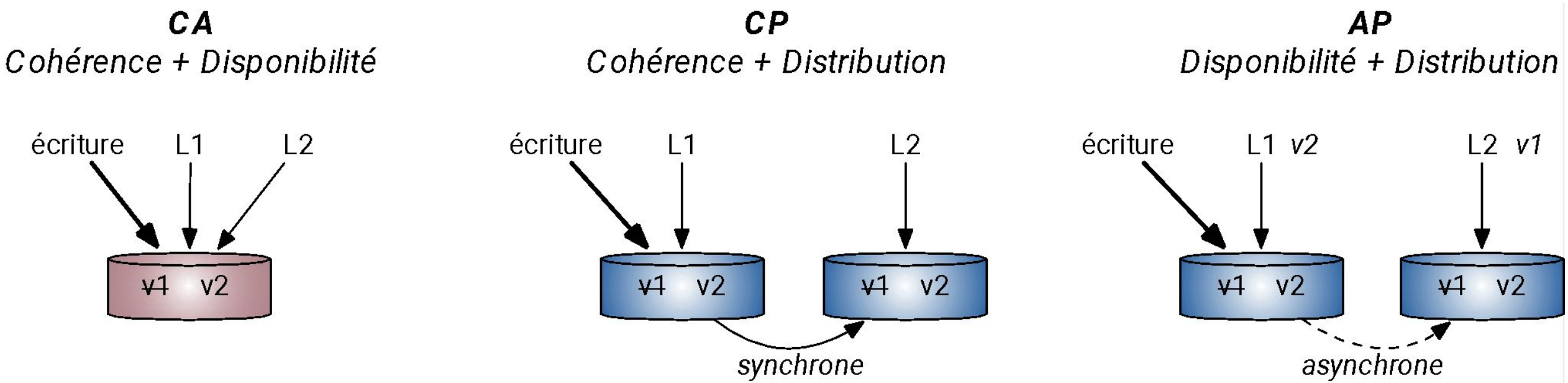
3 fundamental properties for distributed systems

1. ***Consistency: All servers see the same data (value) at the same time (or Consistency)***
2. ***Availability: If a server goes down, data remains available***
3. ***Partition Tolerance: The system, even partitioned, must respond correctly to any request (except in the event of a network failure)***

Theorem: “In a distributed system, it is impossible for these 3 properties to co-exist, you must choose 2 of them”.



“CAP theorem”


CA
Cohérence + Disponibilité
écriture
L1
L2
v1
v2
CP
Cohérence + Distribution
écriture
L1
L2
v1
v2
v1
v2
synchrone
AP
Disponibilité + Distribution
écriture
L1 v2
L2 v1
v1
v2
v1
v2
asynchrone

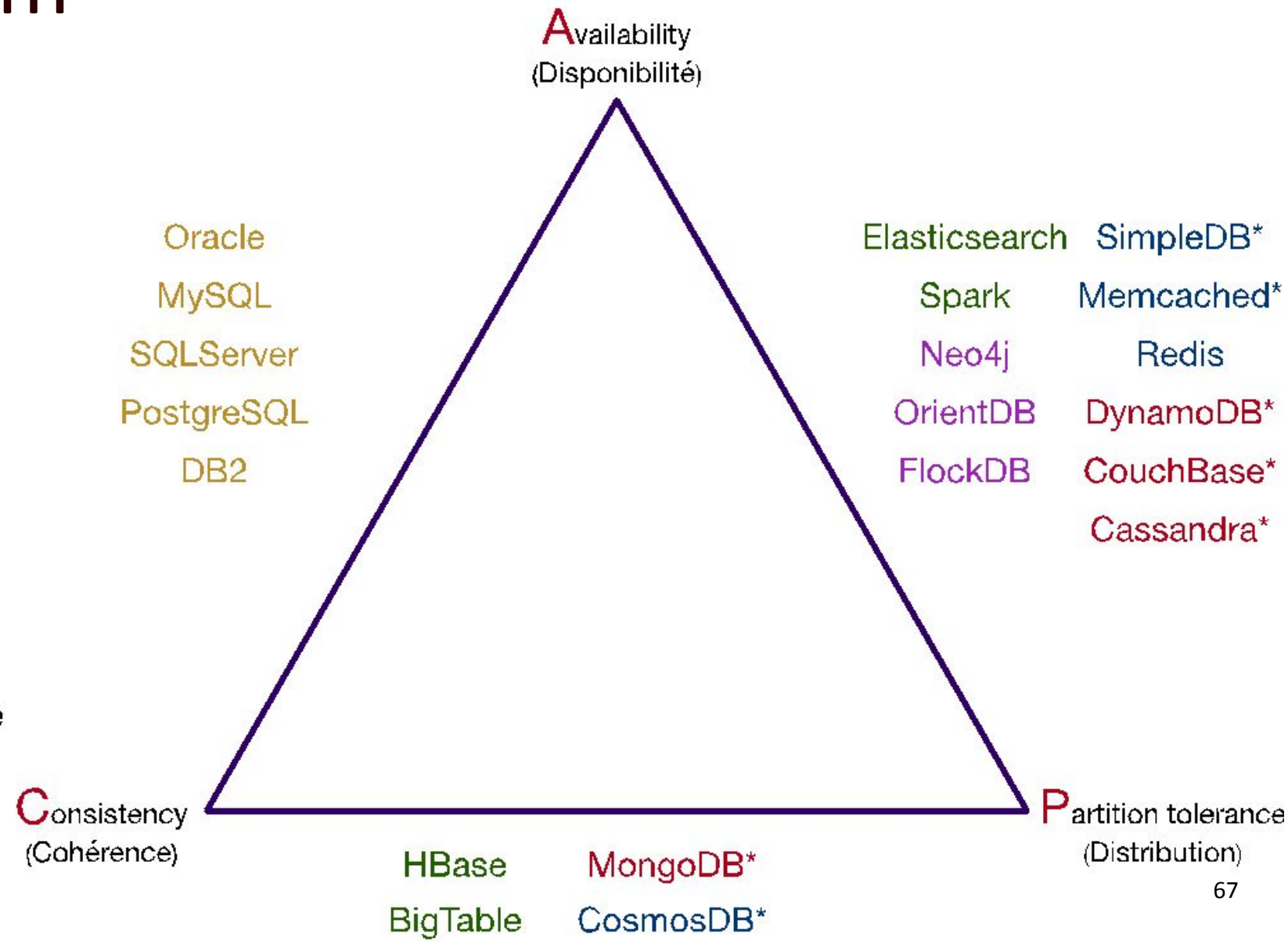
“CAP theorem”

- CAP Triangle

Modèle

Relationnel
Orienté Clé-Valeur
Orienté Colonne
Orienté Document
Orienté Graphe

* Possibilité de changer
le mode de cohérence
Cohérence <-> Disponibilité

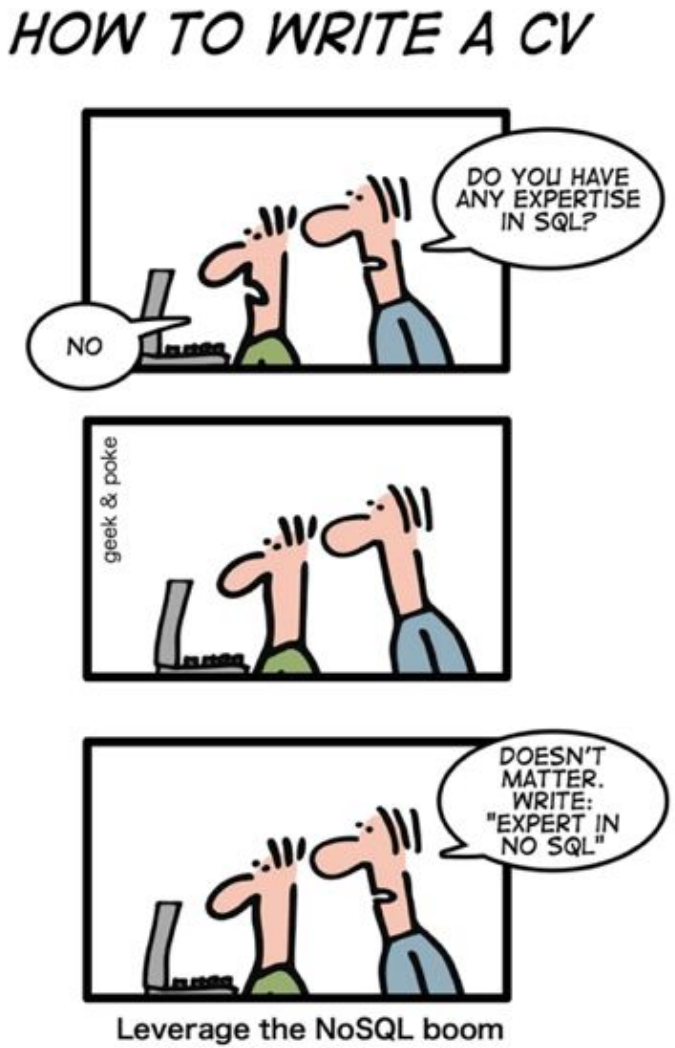


Take away: Qu’avez-vous retenu?

- 5 minutes pour répondre …

Bases noSql

NoSQL does not replace RDBMS



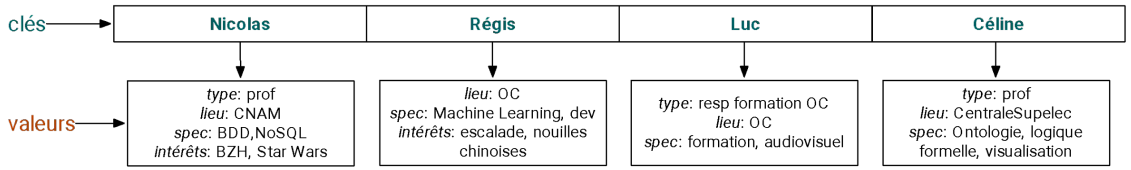
Limitations of RDBMS

“ If the only tool you have is a hammer, you tend to see every problem as a nail.”

Abraham Maslow

NoSQL: a big family

Keys-Values



Column oriented

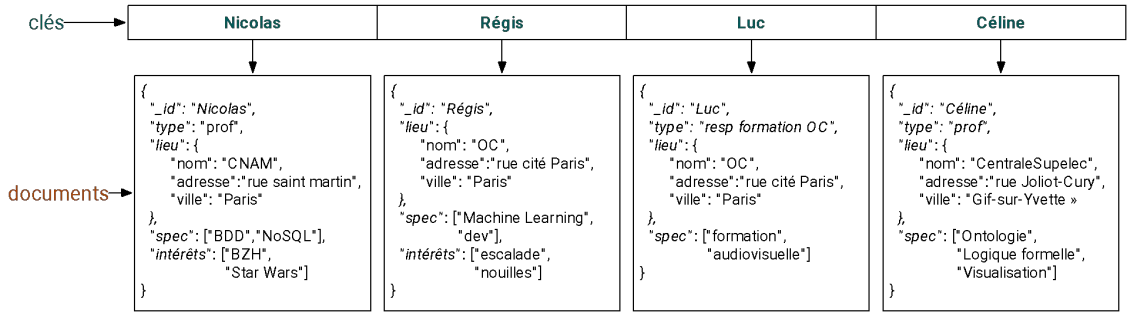
id	type
Nicolas	prof
Céline	prof
Luc	resp formation OC

id	lieu
Céline	Centrale Supelec
Nicolas	CNAM
Régis	OC
Luc	OC

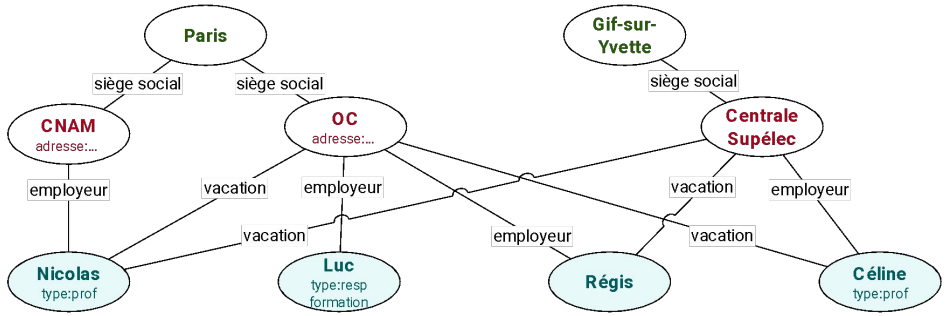
id	spec
Nicolas	BDD
Nicolas	NoSQL
Régis	Machine Learning
Régis	Dev
Luc	formation
Luc	audiovisuel
Céline	Ontologie
Céline	logique formelle
Céline	visualisation

id	intérêts
Nicolas	BZH
Nicolas	Star Wars
Régis	escalade
Régis	nouilles chinoises

Document oriented



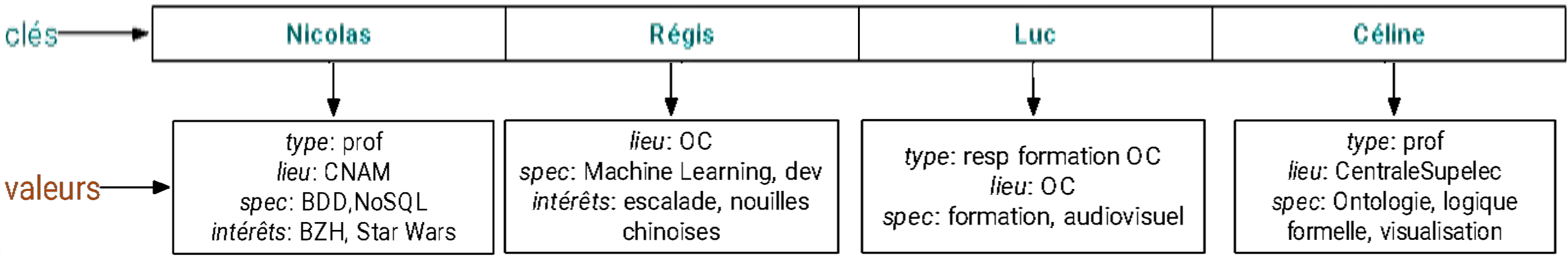
Graph oriented



I - NoSQL & Key-Values: example

Relational

id	type	lieu	spec	intérêts
Nicolas	prof	CNAM	BDD, NoSQL	BZH, Star Wars
Régis		OC	Machine Learning, Dev	escalade, nouilles chinoises
Luc	resp formation OC	OC	formation, audiovisuel	
Céline	prof	CentraleSupelec	Ontologie, logique formelle, visualisation	



I - NoSQL & Key-Values

- “HashMap” distributed
- Key + Value Pair
  - No schema for the value (string, object, integer, binary, etc.) which can therefore be different for each record
- Consequences
  - No structure or types
  - No expressivity of querying (pre/post processing to concretely manipulate the data)

II - NoSQL & Columns

- Example

id	type	lieu	spec	intérêts
Nicolas	prof	CNAM	BDD, NoSQL	BZH, Star Wars
Régis		OC	Machine Learning, Dev	escalade, nouilles chinoises
Luc	resp formation OC	OC	formation, audiovisuel	
Céline	prof	CentraleSupelec	Ontologie, logique formelle, visualisation	

id	type
Nicolas	prof
Céline	prof
Luc	resp formation OC

id	lieu
Céline	Centrale Supelec
Nicolas	CNAM
Régis	OC
Luc	OC

id	spec
Nicolas	BDD
Nicolas	NoSQL
Régis	Machine Learning
Régis	Dev
Luc	formation
Luc	audiovisuel
Céline	Ontologie
Céline	logique formelle
Céline	visualisation

id	intérêts
Nicolas	BZH
Nicolas	StarWars
Régis	escalade
Régis	nouilles chinoises

II - NoSQL & Columns: applications

Usage examples:

- Counting (online voting, counter, etc.)
- Logging
- Search for products in a category (Ebay)
- Large-scale reporting (aggregates calculated on a column)

III - NoSQL & Documents : example

id	type	lieu	spec	intérêts
Nicolas	prof	CNAM	BDD, NoSQL	BZH, Star Wars
Régis		OC	Machine Learning, Dev	escalade, nouilles chinoises
Luc	resp formation OC	OC	formation, audiovisuel	
Céline	prof	CentraleSupelec	Ontologie, logique formelle, visualisation	

documents

```
{
  "_id": "Nicolas",
  "type": "prof",
  "lieu": {
      "nom": "CNAM",
      "adresse":"rue saint martin",
      "ville": "Paris"
  },
  "spec": ["BDD","NoSQL"],
  "intérêts": ["BZH",
               "Star Wars"]
}
```

```
{
  "_id": "Régis",
  "lieu": {
      "nom": "OC",
      "adresse":"rue cité Paris",
      "ville": "Paris"
  },
  "spec": ["Machine Learning",
           "dev"],
  "intérêts": ["escalade",
               "nouilles"]
}
```

```
{
  "_id": "Luc",
  "type": "resp formation OC",
  "lieu": {
      "nom": "OC",
      "adresse":"rue cité Paris",
      "ville": "Paris"
  },
  "spec": ["formation",
           "audiovisuelle"]
}
```

```
{
  "_id": "Céline",
  "type": "prof",
  "lieu": {
      'nom': 'CentraleSupelec',
      'adresse':' rue Joliot-Cury',
      'ville': 'Gif-sur-Yvette »
  },
  "spec": ['Ontologie',
           'Logique formelle',
           'Visualisation']
}
```

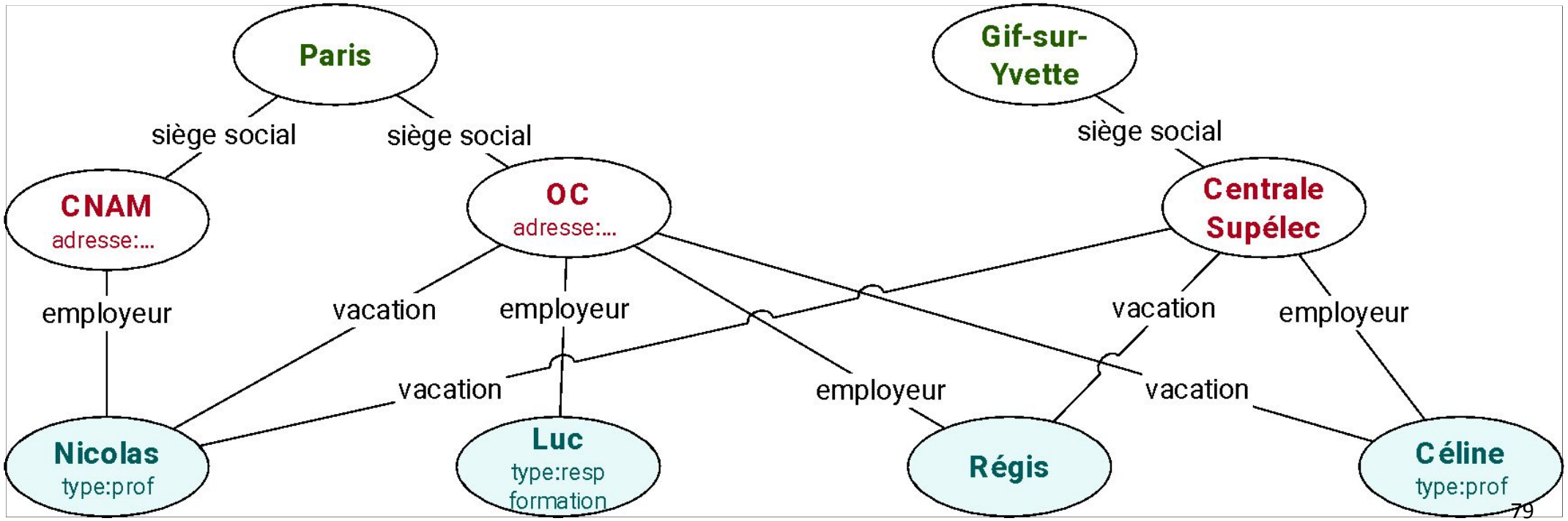
III - NoSQL & Documents

- Based on the key-value model
  - Adding semi-structured data (JSon or XML)
  - Same as “key-value” but “value = document”
    - Document composed of keys/values
    - Types simples (Int, String, Date)
    - Diagram not necessary (may vary from one document to another)
    - Data nesting (tree schema)
    - Lists of values
- Requests:
  - More complex than CRUD
  - Each key in the document can be queried

IV - NoSQL & Graph : example

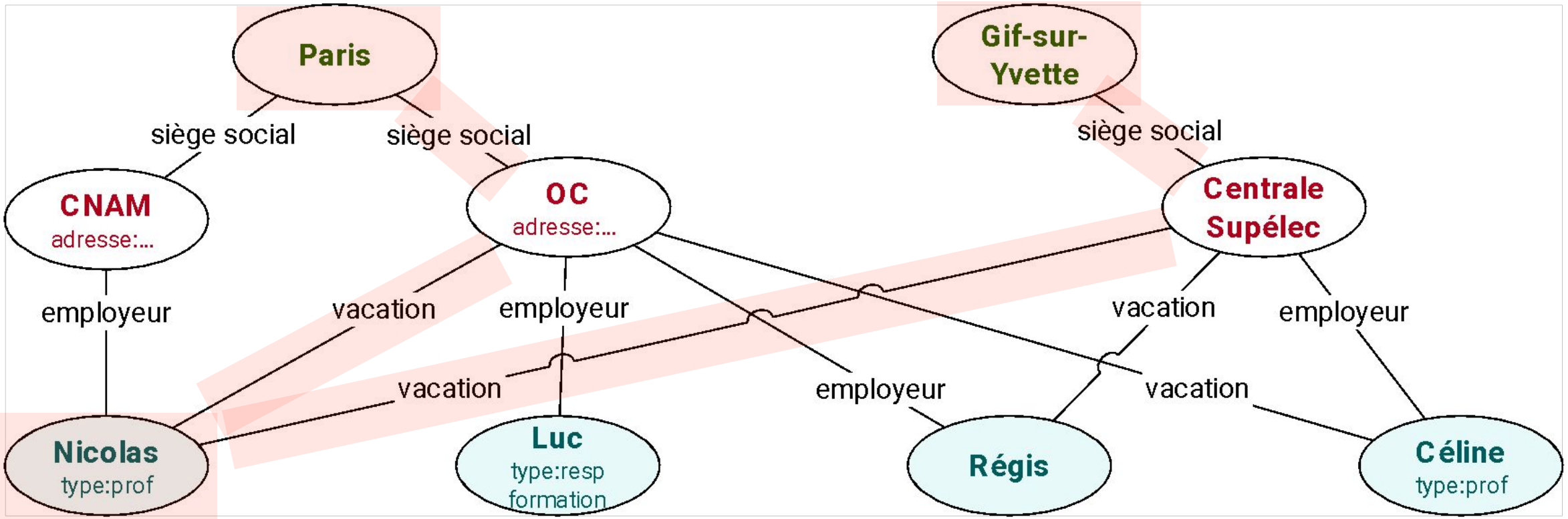
Azure Cosmos DB:

id	type	lieu	spec	intérêts
Nicolas	prof	CNAM	BDD, NoSQL	BZH, Star Wars
Régis		OC	Machine Learning, Dev	escalade, nouilles chinoises
Luc	resp formation OC	OC	formation, audiovisuel	
Céline	prof	CentraleSupelec	Ontologie, logique formelle, visualisation	



IV - NoSQL & Graph : query-ing

- Graph queries:
  - **Person** doing **vacation** at **Paris** *and at* **Gif-sur-Yvette**

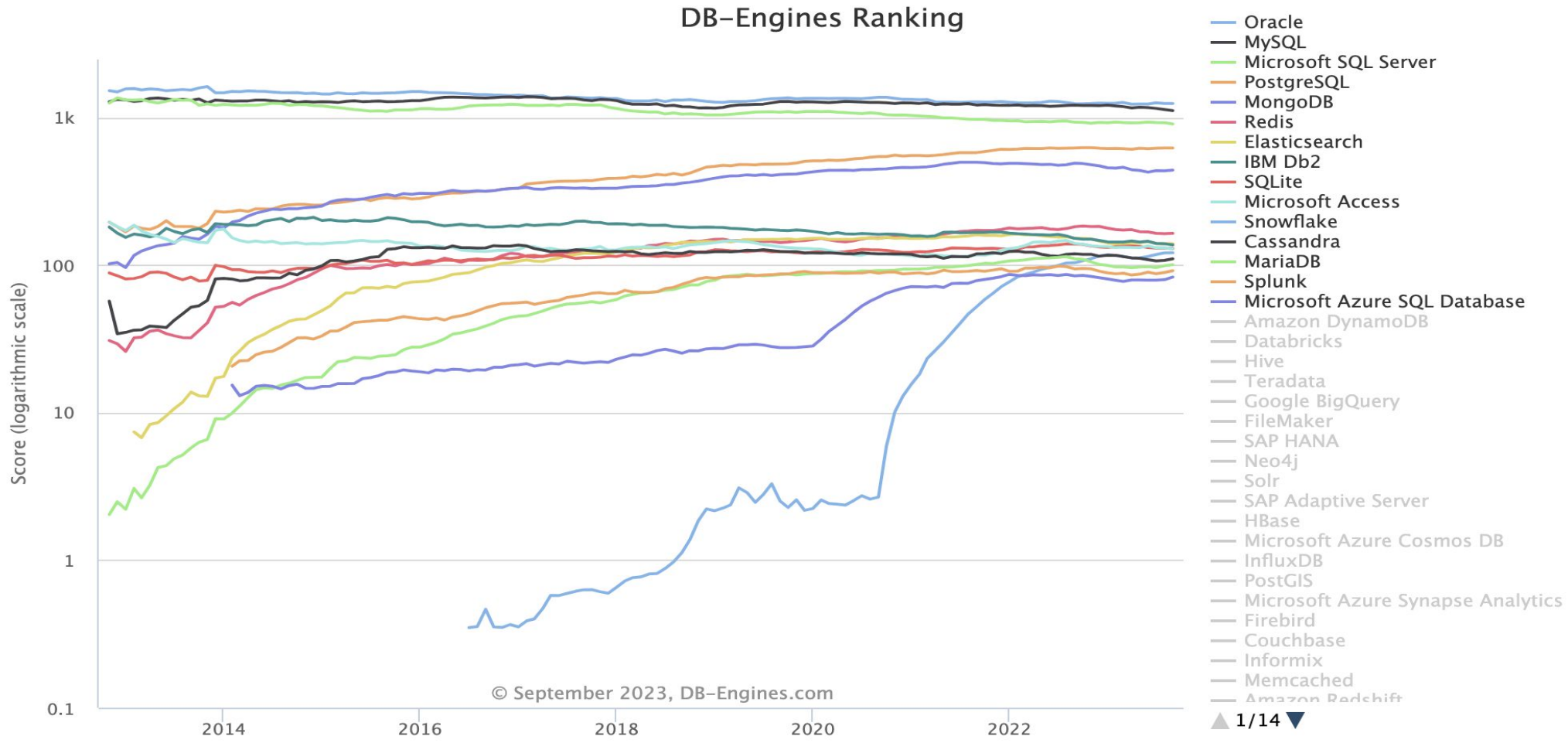


IV - NoSQL & Graph

- Storing nodes, relationships and properties
  - Graph theory
  - Querying by graph traversals
  - Calling up data on request (efficient routes)
  - Non-trivial modeling

DB-Engines

Ranking table
September 2023



Conclusion

- NoSQL
  - Dedicated to an extremely distributed context
  - Highly distributed computing
  - 4 types of complex calculations (key-value, document, columns, graphs)
  - CAP theorem
- Should not automatically replace a DBMS
  - ACID properties
  - Complex queries
  - Join performance

SQL is everywhere even in noSql

- Cassandra => CQL
- Couchbase => N1ql
- Phoenix => SQL on Hbase
- Hive / Spark => SQL
- DBT => Data Build Tool

Hadoop Eco-system

Hadoop

- Hadoop est un éco-système open source fournissant tout les outils nécessaires pour travailler sur la donnée de façon distribuée.

L'objectif d'Hadoop est d'utiliser de nombreuses machines peu chères et de les faire travailler ensemble pour traiter de large volume de données en parallèle.

Les composants sont tolérants à la panne.

Hadoop history

Doug Cutting adds DFS & MapReduce support to Nutch

Doug Cutting & Mike Cafarella started working on Nutch

NY Times converts 4TB of image archives over 100 EC2s

Fastest sort of a TB, 3.5mins over 910 nodes

Fastest sort of a TB, 62secs over 1,460 nodes
Sorted a PB in 16.25hours over 3,658 nodes

2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009

Google publishes GFS & MapReduce papers

Yahoo! hires Cutting, Hadoop spins out of Nutch

Facebooks launches Hive: SQL Support for Hadoop

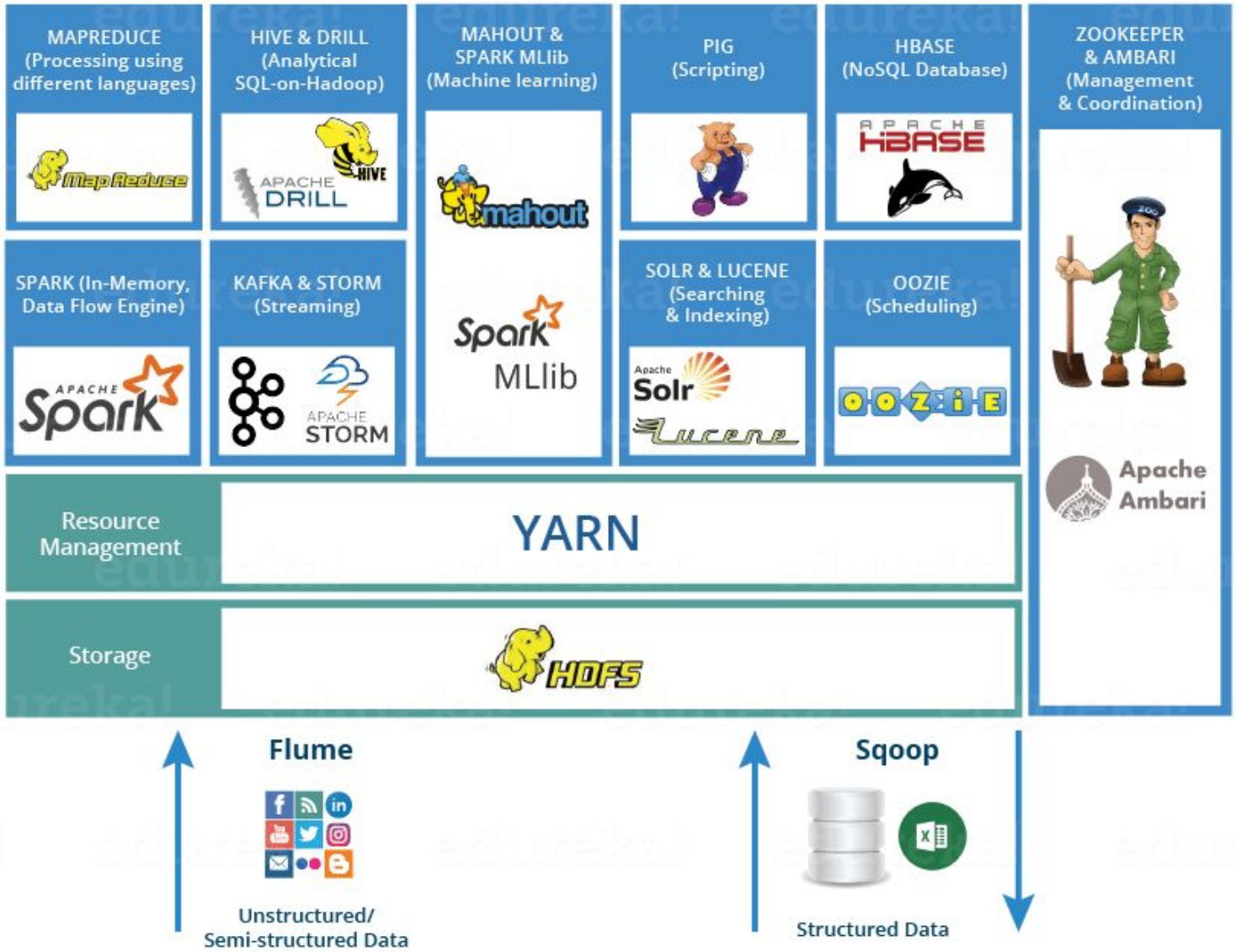
Founded

Doug Cutting joins Cloudera

Hadoop Summit 2009, 750 attendees

Source: Cloudera, Inc.

Hadoop Components



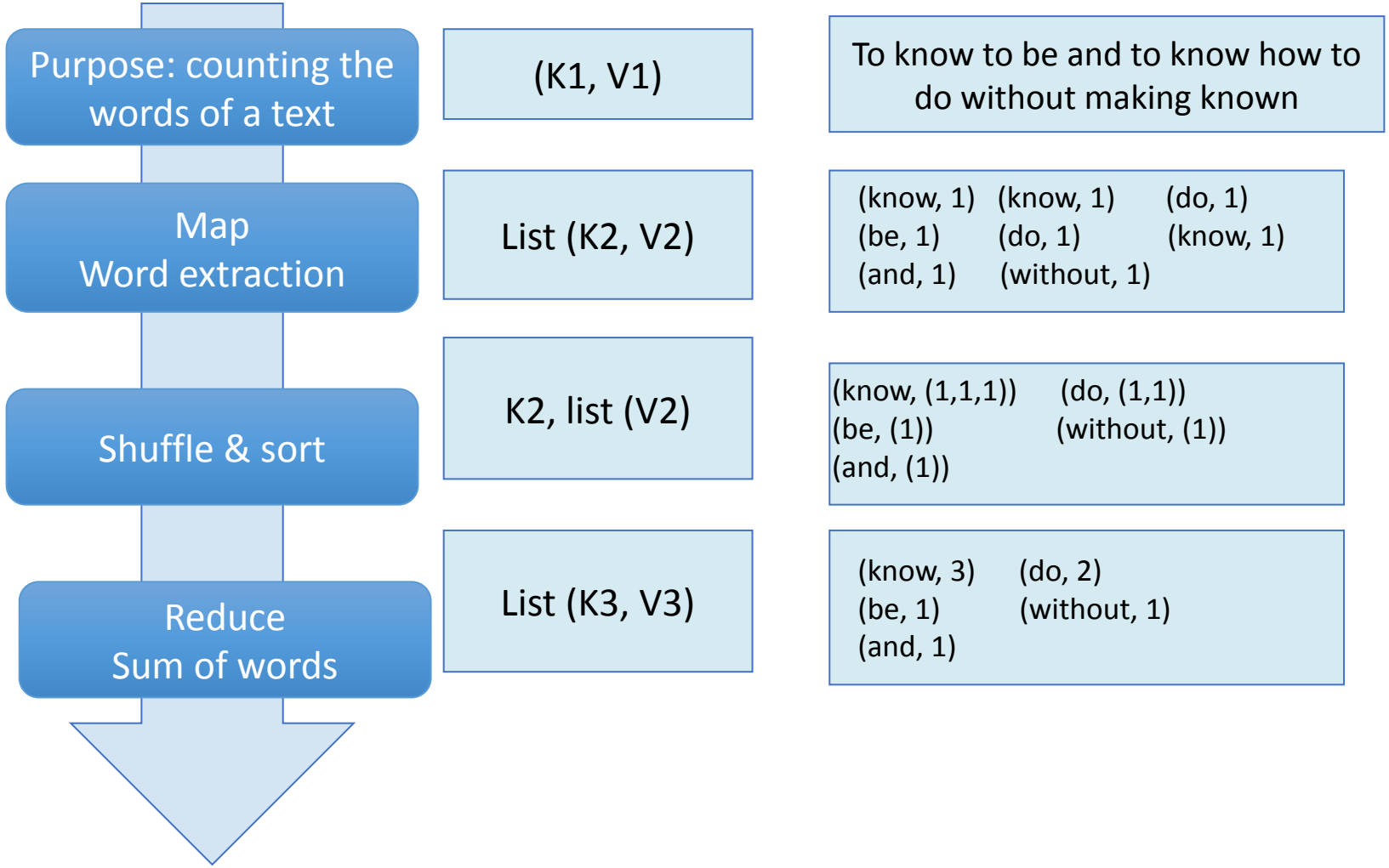
Mapreduce

Introduced by Google in 2004, its goal is to make parallel treatments (indexing, datamining, …)

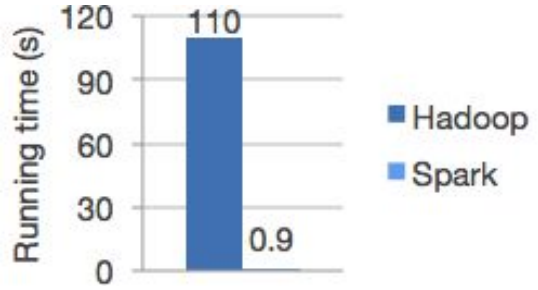
Borrowed from functional programming:

- **Map**: is a data transformation step in the form of key / value pairs
- **Shuffle / spell**: distributed all data that have the same key are the same machine.
- **Reduce**: is a step of merging key records to form the final result

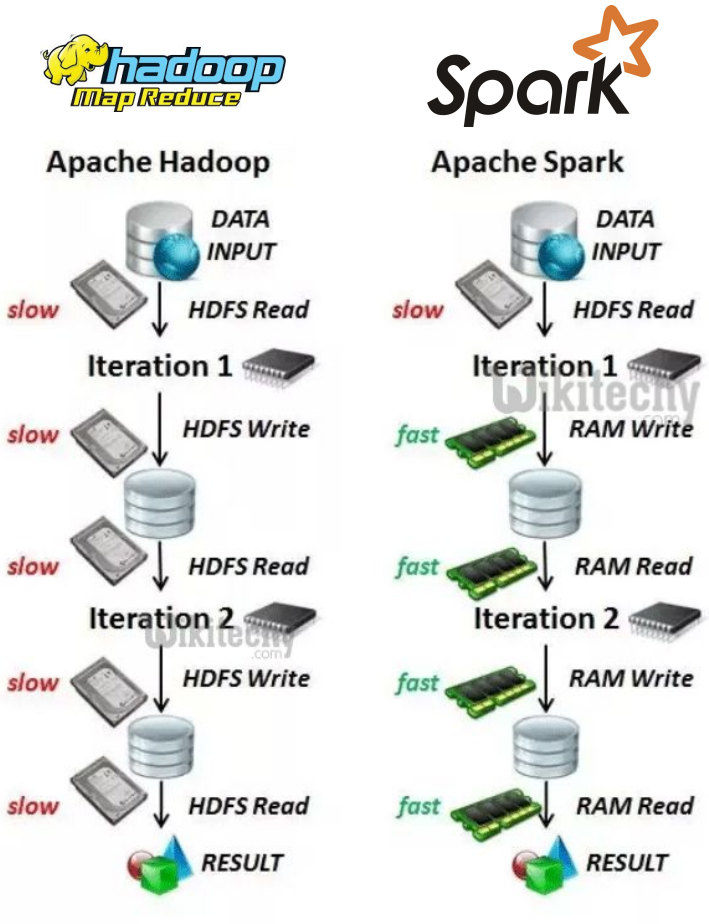
Algorithm principle

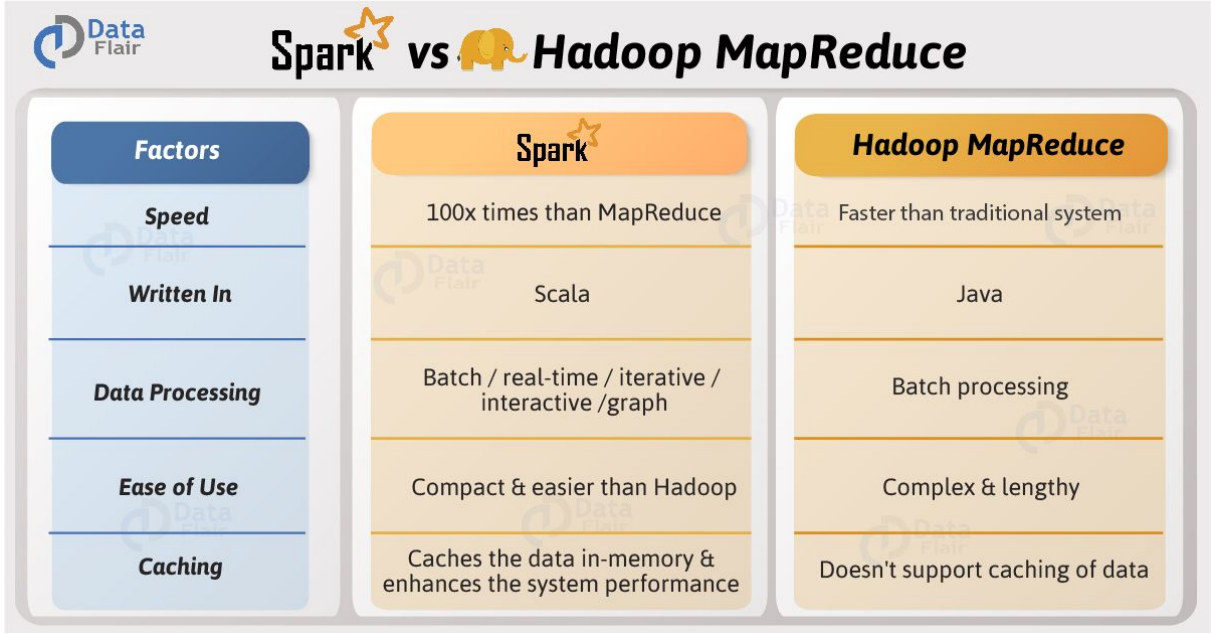


Birth of spark

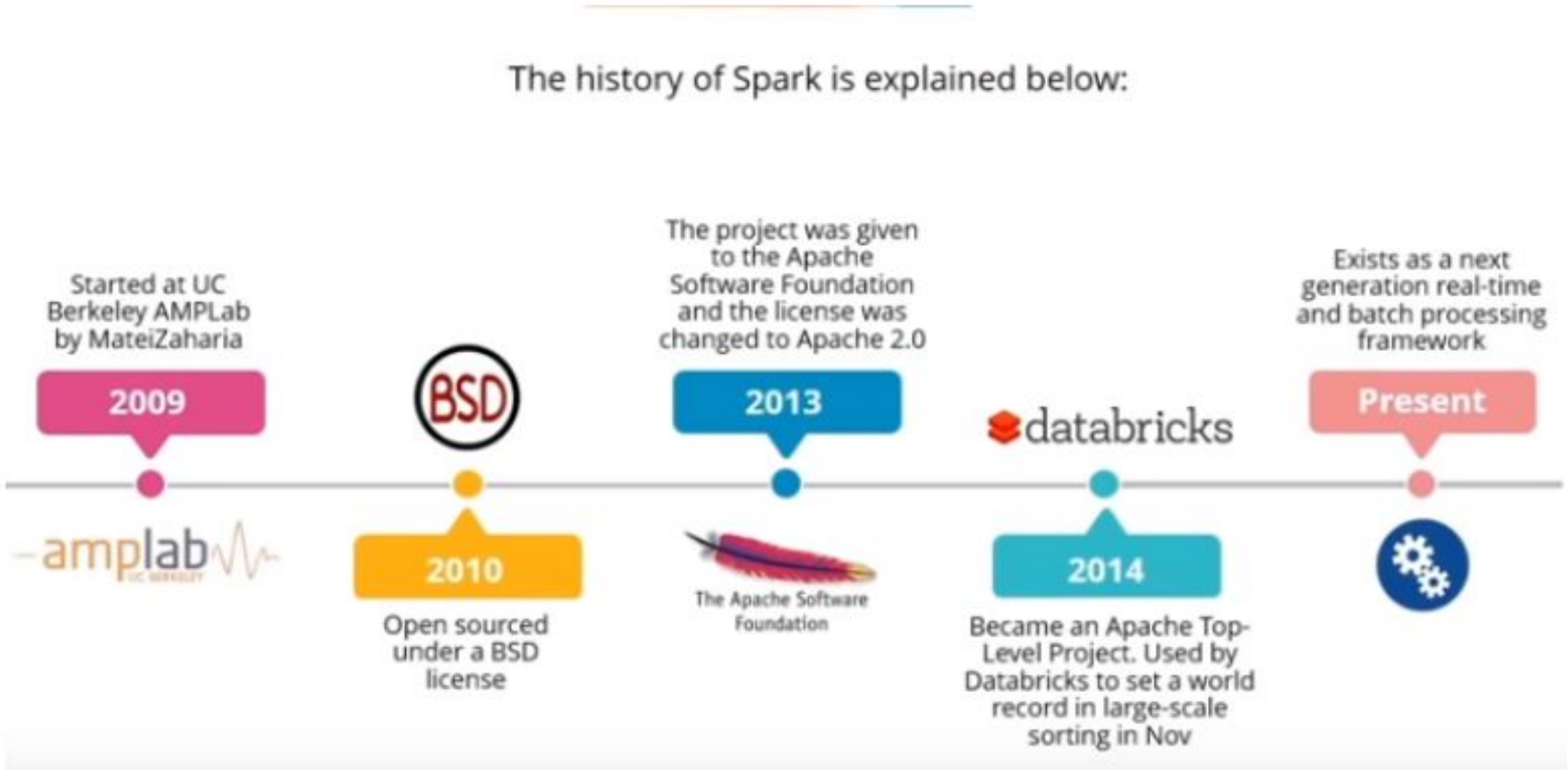


On mapReduction, you have to write on disk at every step what slows down the treatments.





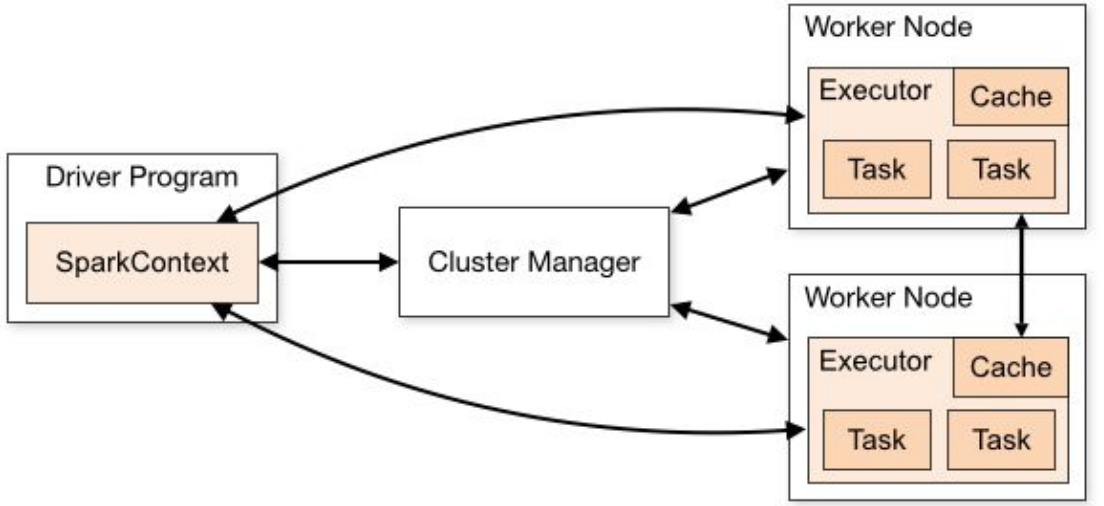
History of Spark

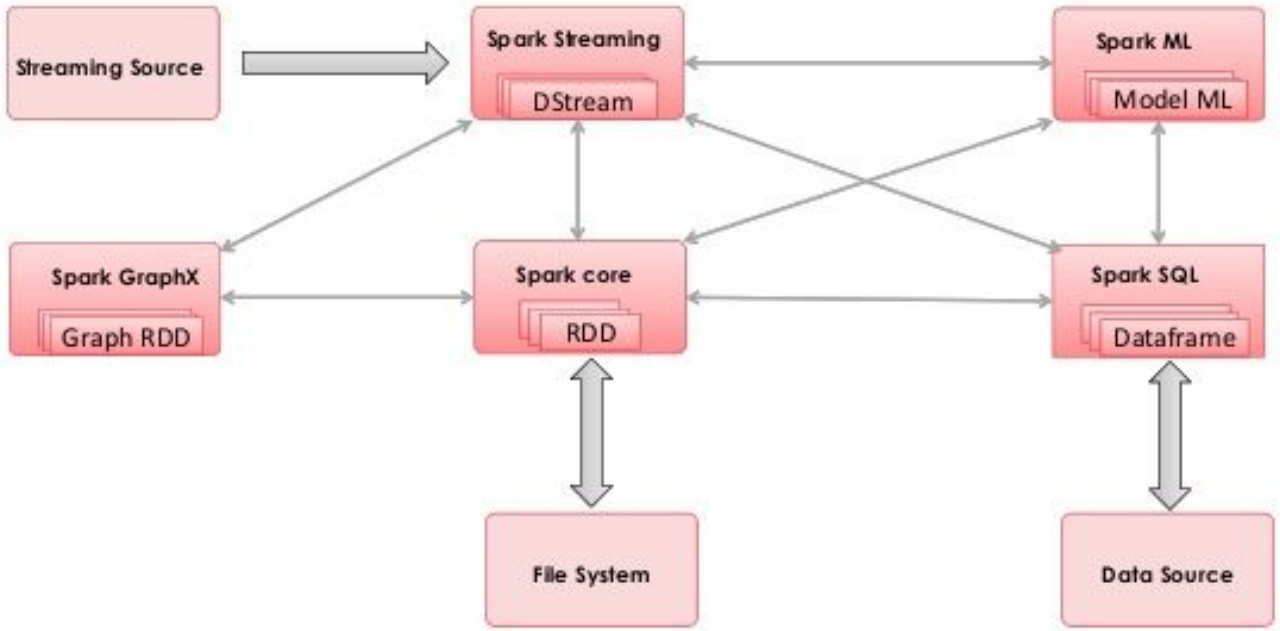


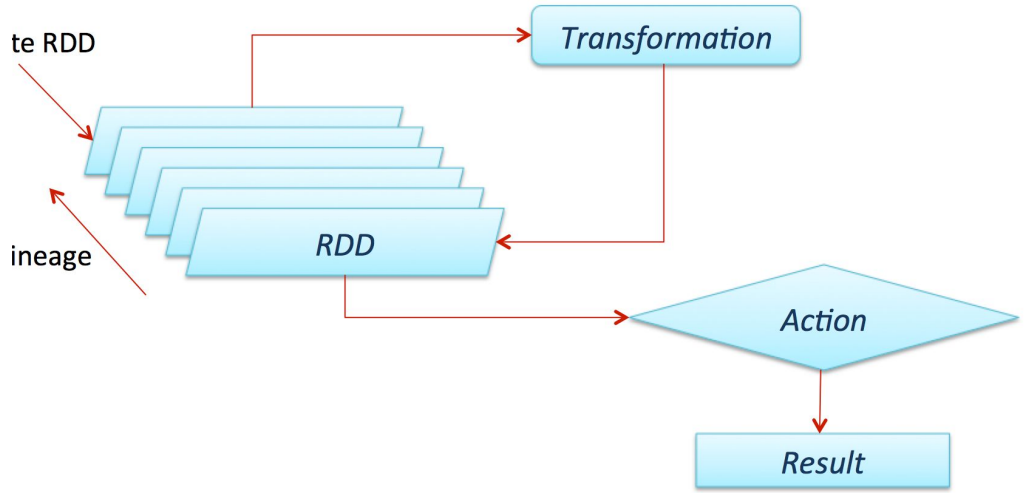
In 2021: Databricks (‘founder’ of Spark) raises 1 Billion:
https://www.lebigdata.fr/databricks-leve-1-milliard

Spark Components

Spark invents RDDs (Resilient Distributed Dataset) to distribute calculations in memory. It allows you to do Batch, streaming, ML, Graph.

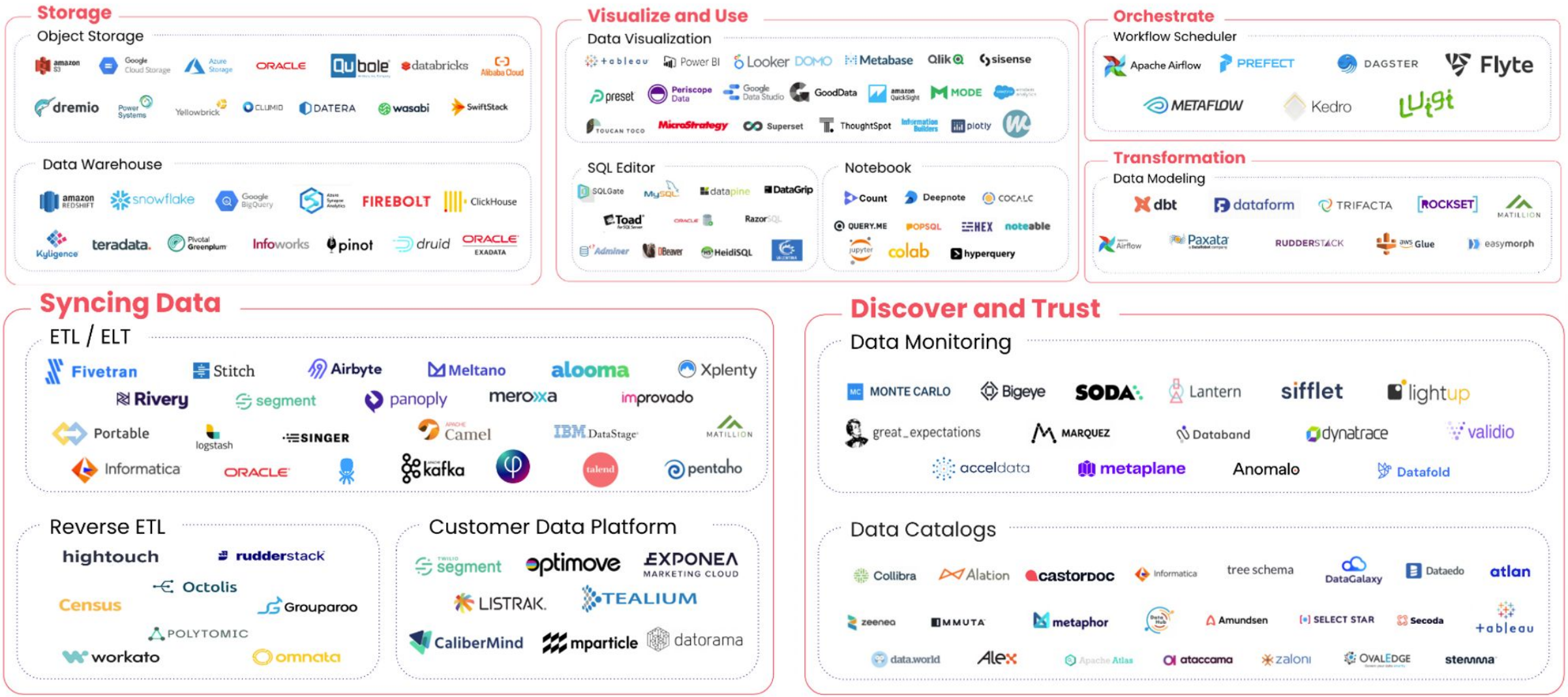






Modern data stack landscape

Careful, not everything here is open-source:



mapReduce

Cassandra Deep-dive

Design Social network app (Instagram)

I hope you learnt a lot !