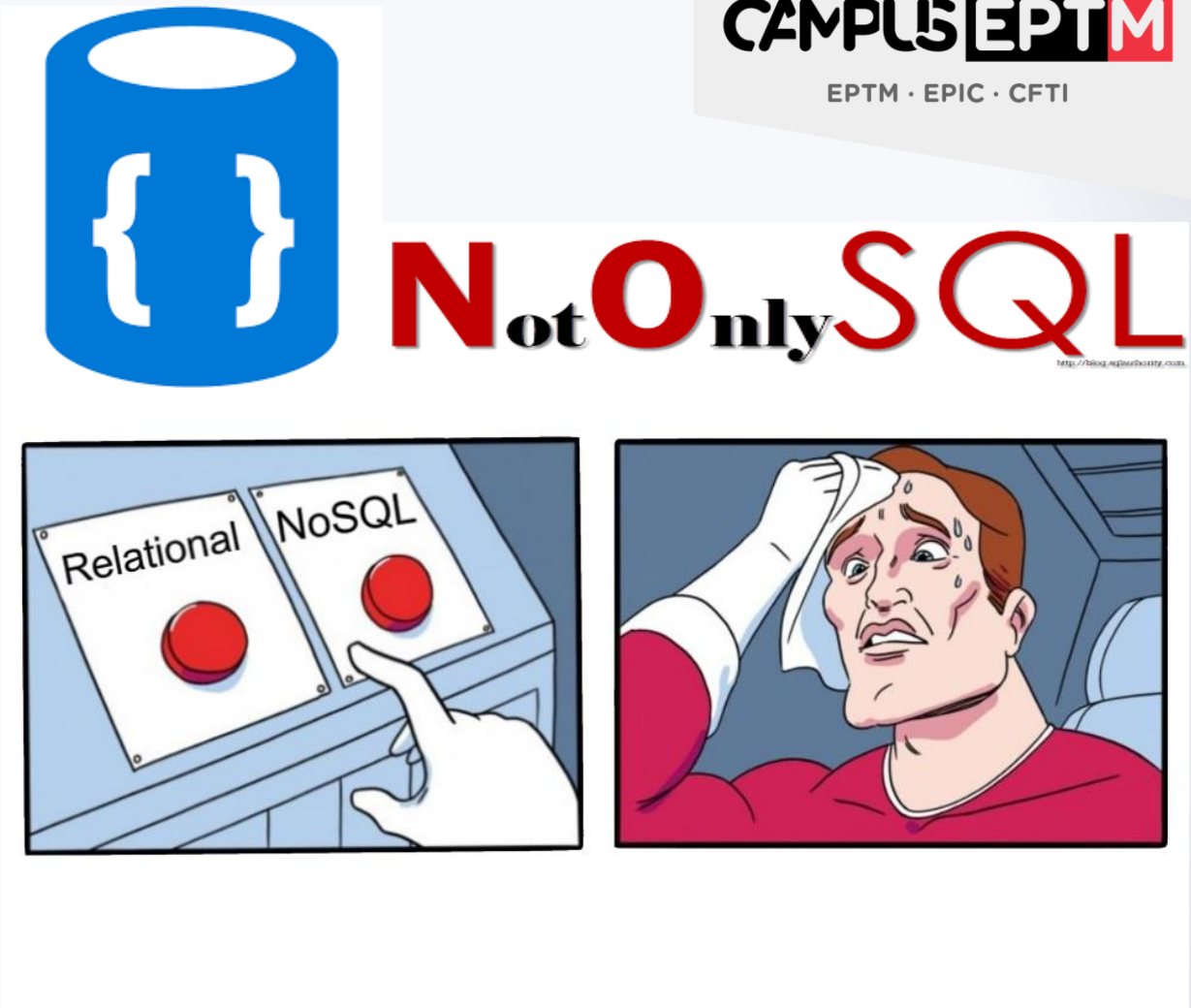
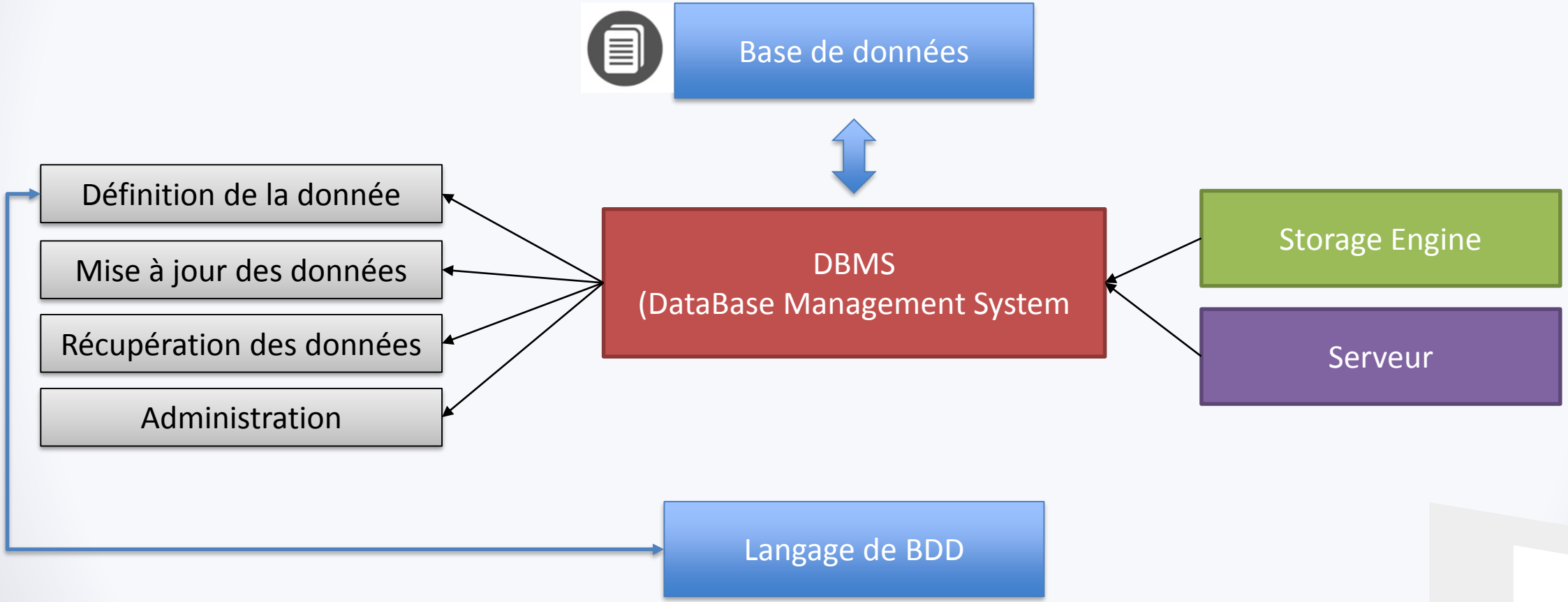
M165
Utiliser des bases de données NoSQL

Version mars 2023



Eléments d'une BDD



Historique NoSQL

- Terme utilisé la première fois en 1998 par Carlo Strozzi pour désigner un système relationnel n'utilisant pas le langage SQL
  - Coïncidence sans lien avec la famille de SGBD NoSQL
- 2000 : Création de la base de données graphique Neo4j
- 2004 : Bigtable lancée par Google (orientée colonnes)
- 2005 : Apache CouchDB
- 2007 : Amazon Dynamo (clé valeur)
- 2008 : Facebook rend Cassandra «opensource». Remet le terme NoSQL sous le feu des projecteurs et sa popularité actuelle
- 2009 : MongoDB
- 2011 : Google Firebase

SQL - NoSQL

CAMPUS EPTM
EPTM · EPIC · CFTI

SQL

Base de données relationnelle

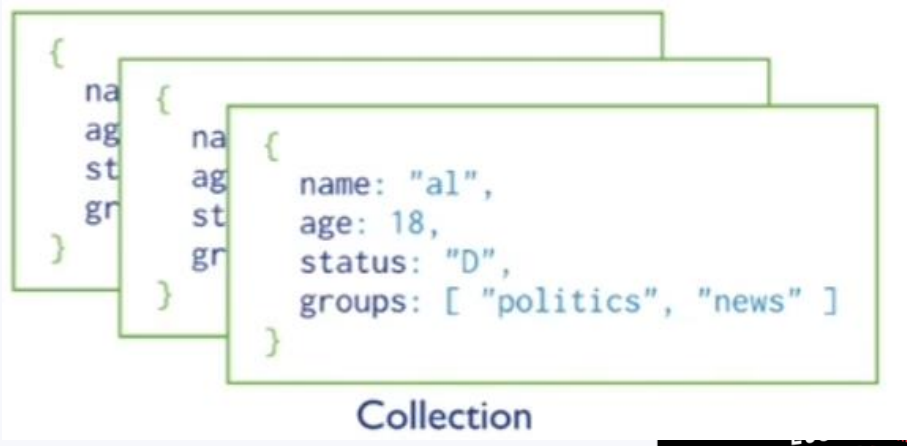
Tables et rangées

Id	Name	Department	Salary	Gender	Age	City
1001	John Doe	IT	35000	Male	25	London
1002	Mary Smith	HR	45000	Female	27	Mumbai
1003	James Brown	Finance	50000	Male	28	Delhi
1004	Mike Walker	Finance	50000	Male	28	London
1005	Linda Jones	HR	75000	Female	26	Mumbai
1006	Anurag Mohanty	IT	35000	Male	25	London
1007	Priyanla Dewangan	HR	45000	Female	27	Mumbai
1008	Sambit Mohanty	IT	50000	Male	28	London
1009	Pranaya Kumar	IT	50000	Male	28	London
1010	Hina Sharma	HR	75000	Female	26	Mumbai

NoSQL

Base de données NON relationnelle

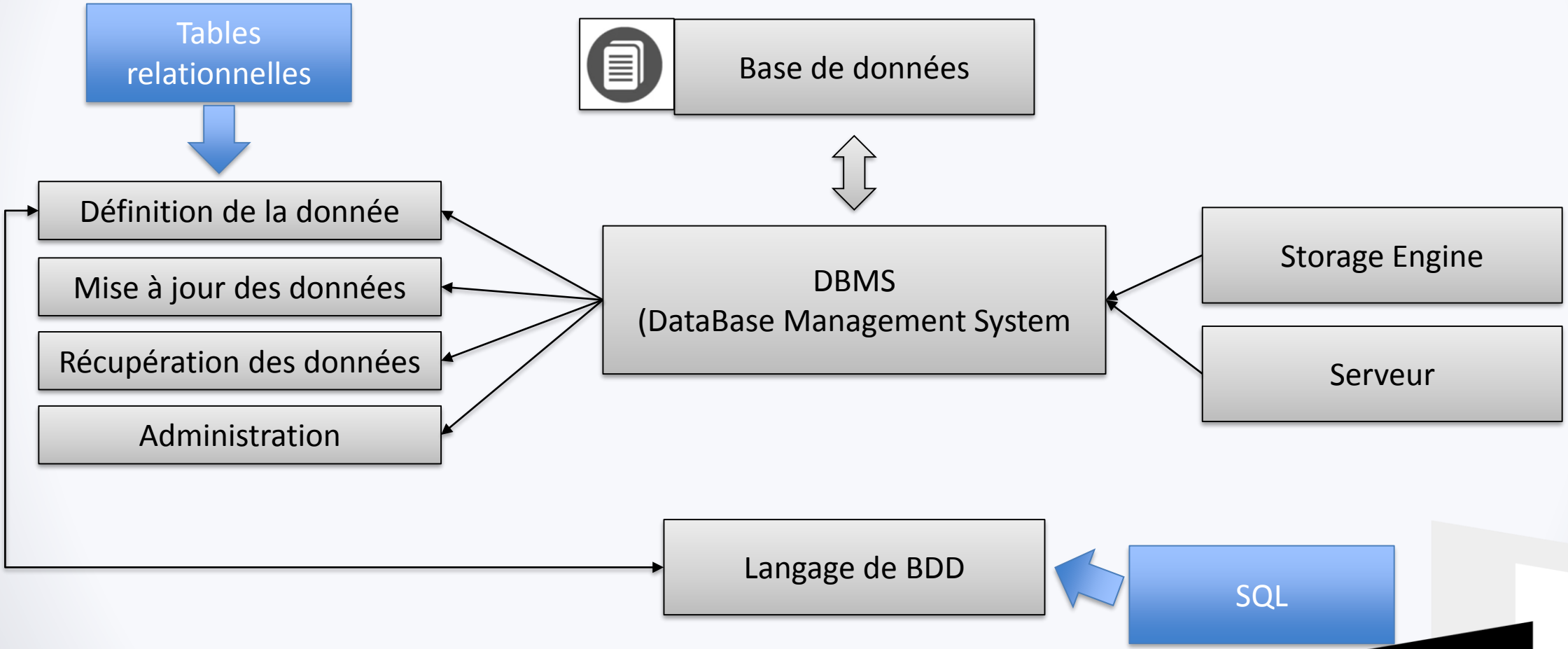
- Graphe
- Clé-valeur
- Colonne-large
- Document



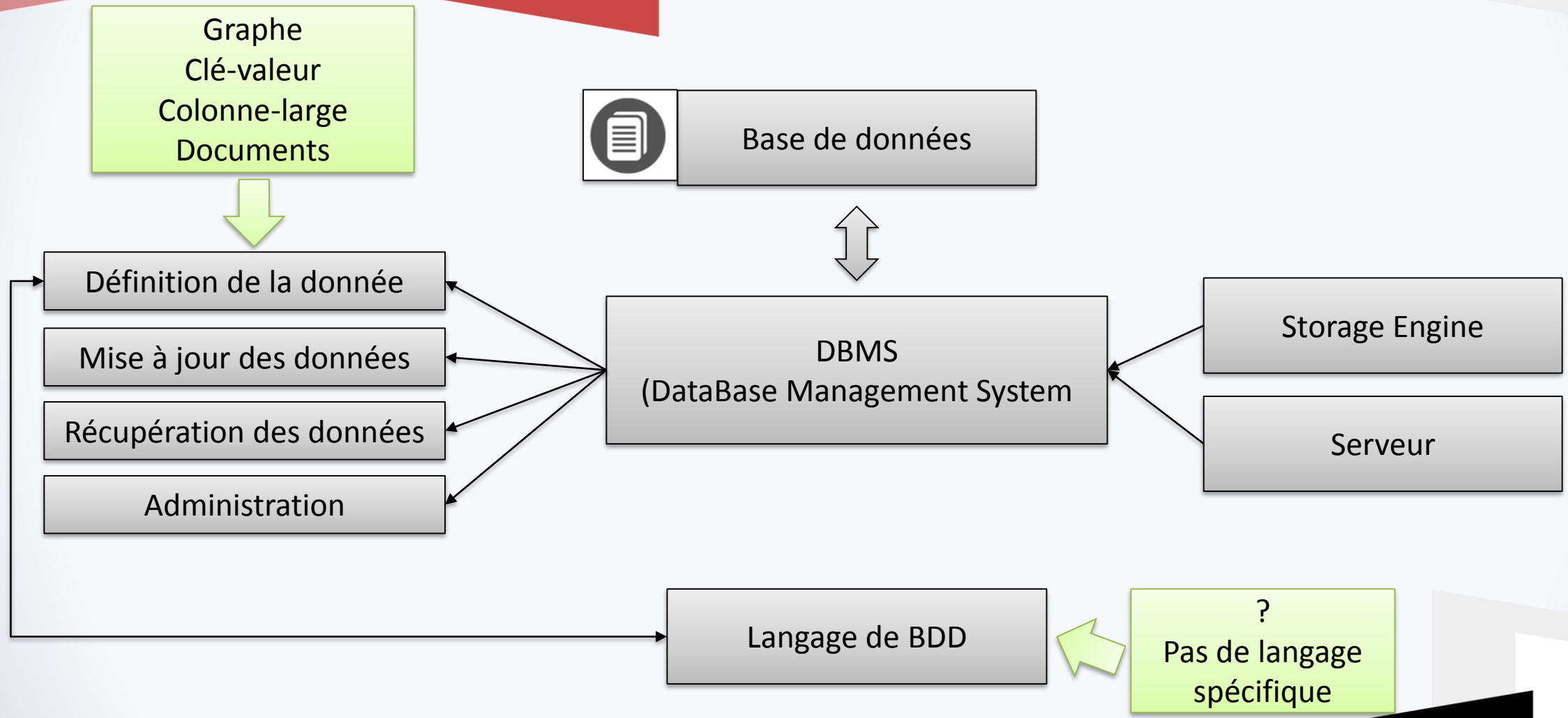
PROFESSIONNELLE DES METIERS

CANTON DU VALAIS
KANTON WALLIS

Structure en SQL



Structure en NoSQL



Quelques acteurs SQL et NoSQL

CAMPUS EPTM
EPTM · EPIC · CFTI

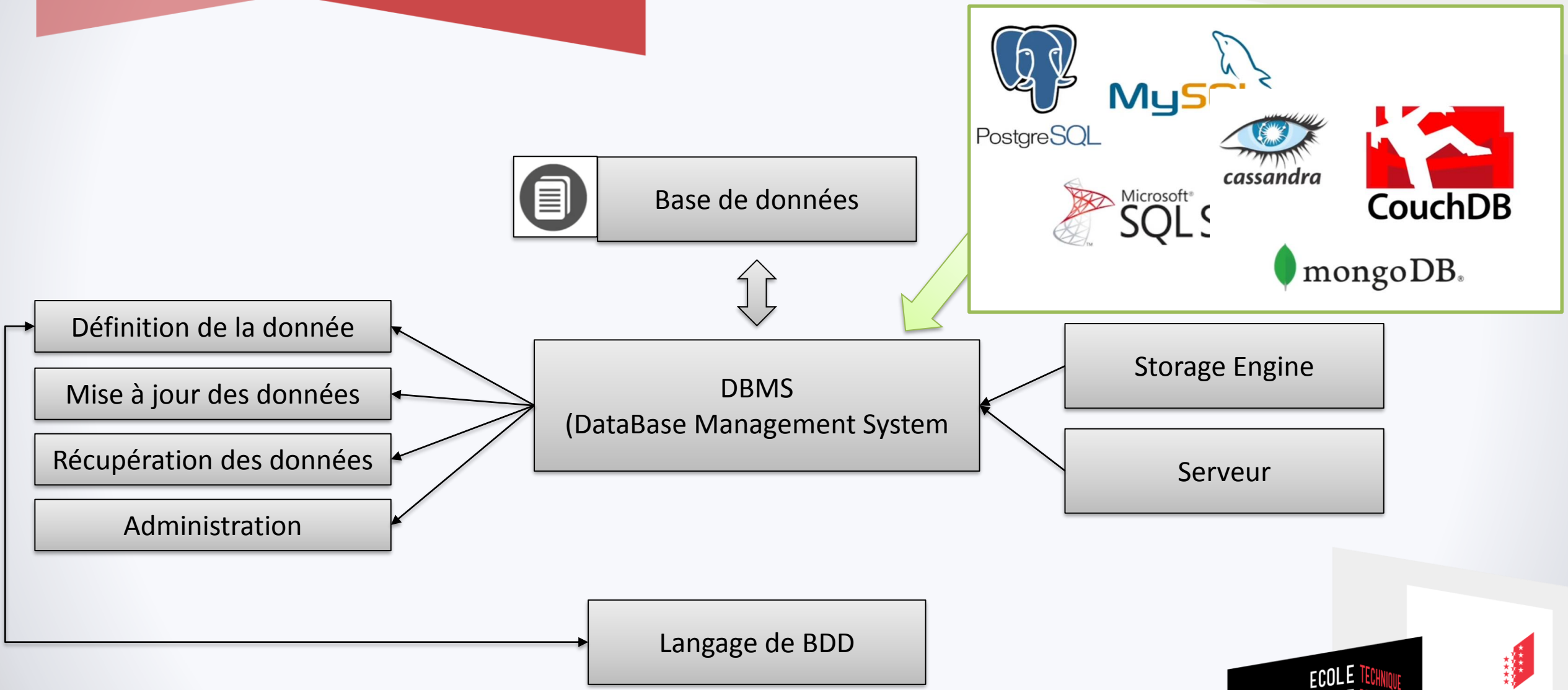
SQL

PostgreSQL

Microsoft SQL Server

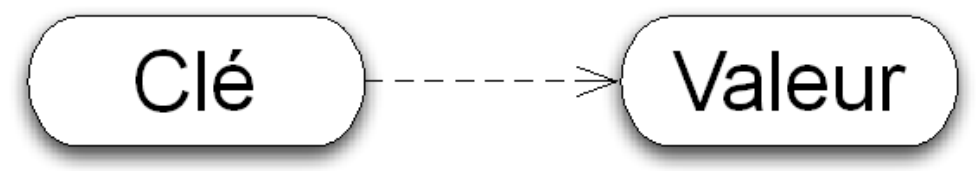
NoSQL

Acteurs = DBMS



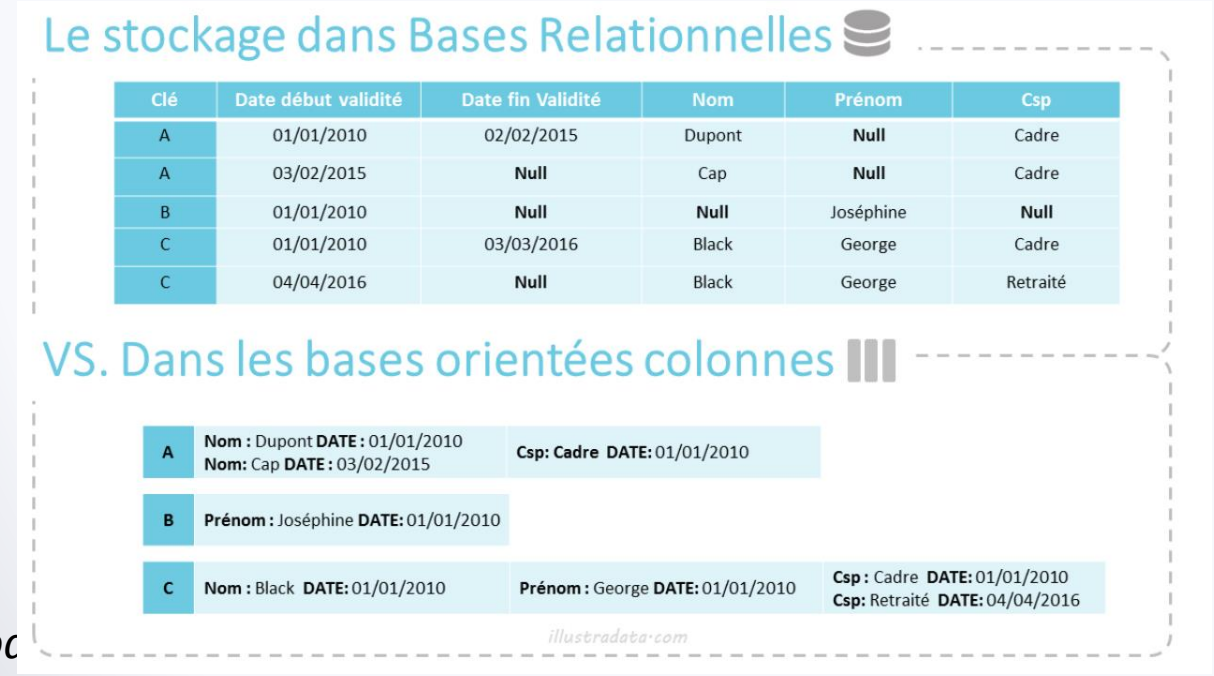
NoSQL
Orientée clé-valeur

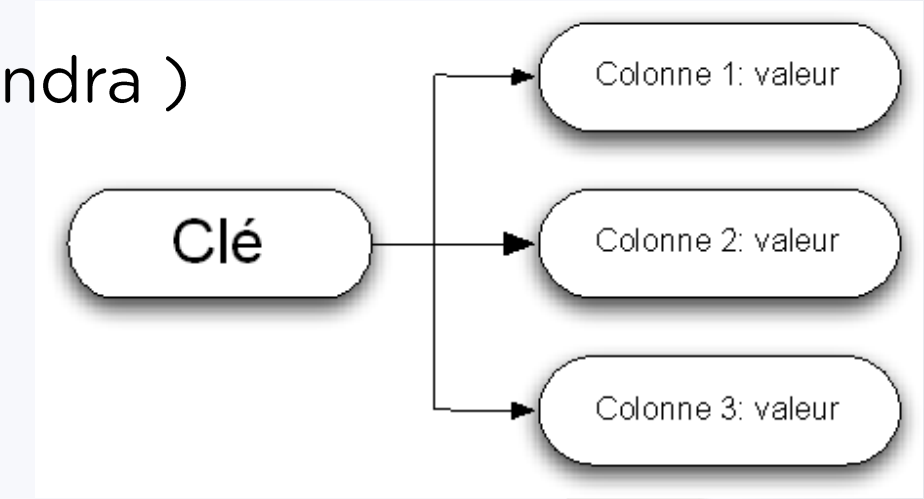
- La plus simple
- 3 opérations PUT – GET – DELETE
- Pas de langage de requête
- Adaptée aux caches et aux accès rapides aux informations
- Lecture / Ecriture = accès disque simple
- Exemples (Amazon DynamoDB, Redis, Voldemort (linkedin))



NoSQL
Orientée Colonnes

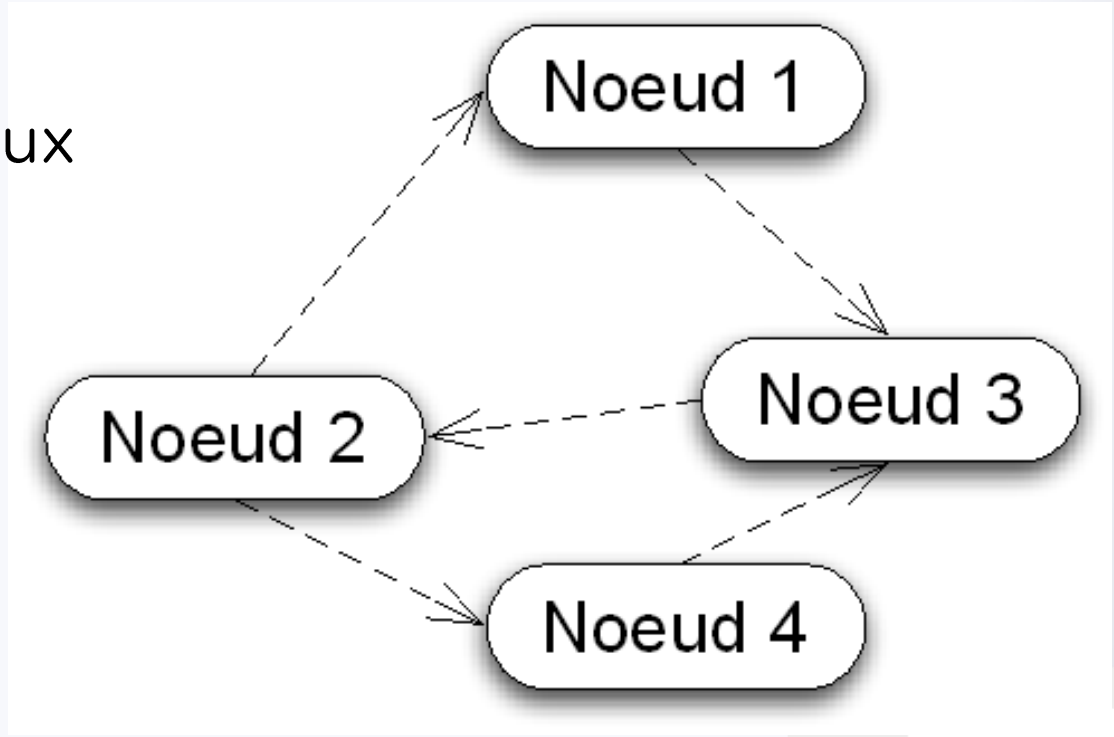
- Se rapproche le plus des tables dans une base relationnelle
  - Tables / lignes / colonnes
  - Mais colonnes dynamiques (pas de valeur null)
- Plus évolutive et flexible que clé-valeurs
- Exemples (Big Table, Apache Hbase, Cassandra)





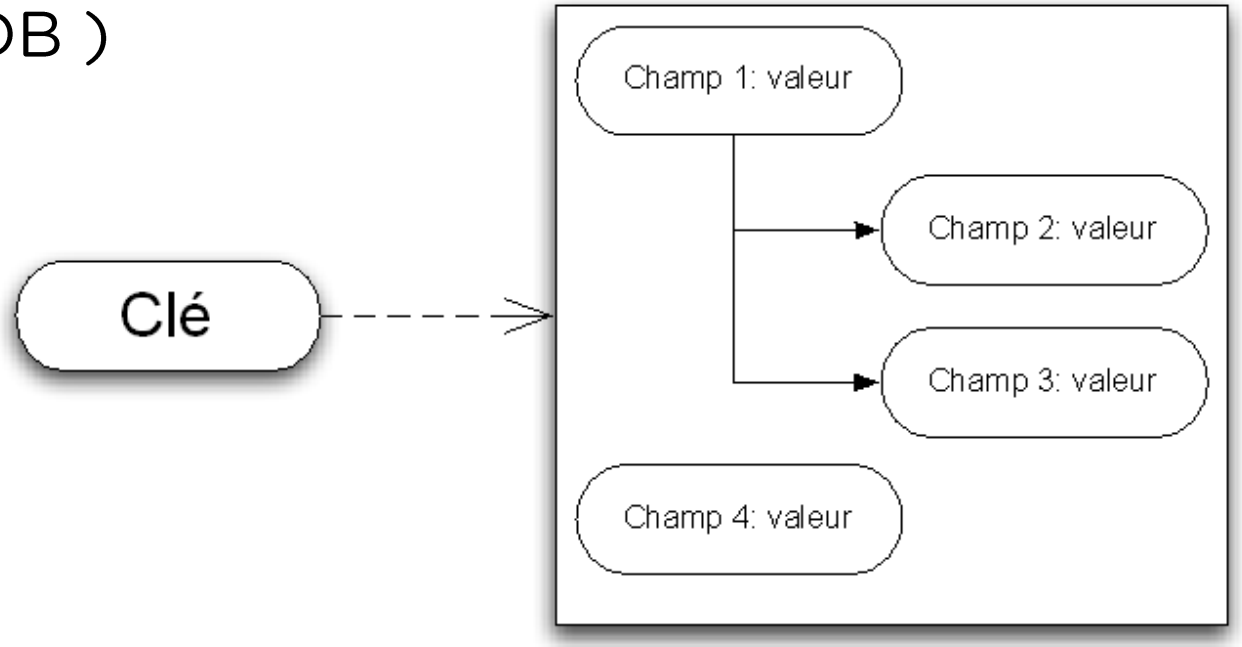
NoSQL
Orientée Graphes / Noeuds

- Palie à des problèmes impossibles à résoudre avec des BDD relationnelles
- Forme de graphe
- Utilisation typique des réseaux sociaux
- Exemples
  - (Neo4j, JanusGraph, Amazon Neptune)



NoSQL
Orientée Documents

- Proche de la version « clé-valeurs »
- Chaque valeur est représentée sous la forme d'un document
- Dans un document les données sont organisées de manière hiérarchique. (json)
- Exemples (CouchDB, MongoDB)



Points forts NoSQL
Orienté Document

- Modèle de données flexible
- Scale peu coûteux
- Requêtes Rapides
- Facile à utiliser pour les Devs

→ Scaling Horizontal

VS

Scaling Vertical

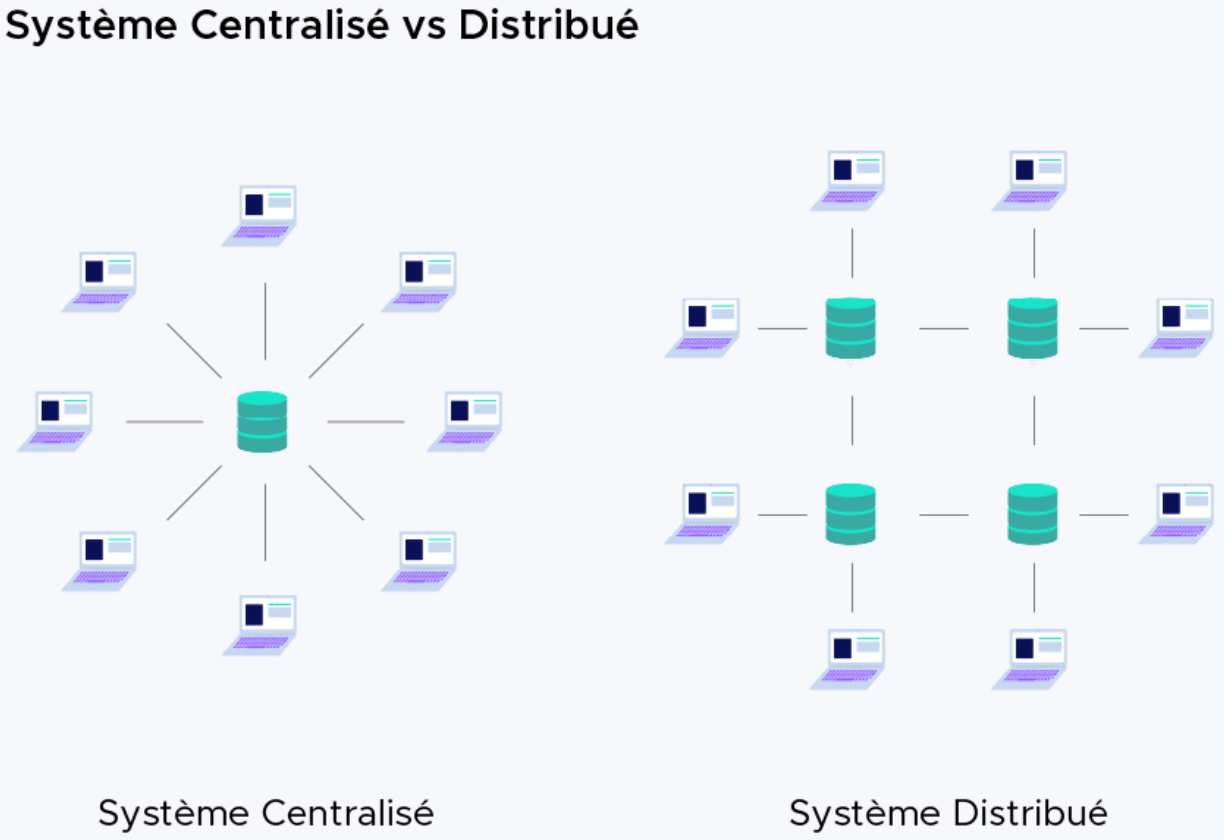
Faut-il abandonner le Relationnel ?

- Le Relationnel (SGBDR) restent incontournables concernant :

  1) Faire des jointures entre les tables d'une base de données
  2) Faire des requêtes complexes, avec un langage haut niveau
  3) Gérer l'intégrité des données de manière infaillible

- Ce dernier point est à prendre en compte dans le cadre NoSQL. La gestion de la cohérence d'une donnée n'est pas forcément garantie

Système distribué VS Système centralisé



ACID vs BASE

Atomicité : une transaction s'effectue entièrement ou pas du tout

Cohérence : le contenu d'une base doit être cohérent au début et à la fin d'une transaction

Isolation : Les modifications d'une transaction sont visibles uniquement lorsqu'elle est validée

Durabilité : Une fois la transaction validée, l'état est permanent

→ Pas forcément applicable dans un contexte distribué tel que NoSQL

ACID vs BASE

Basically Available:
quelle que soit la charge de la base de données (données/requêtes), le système garanti un taux de disponibilité de la donnée

Soft-state:
La base peut changer lors des mises à jour / ajout / suppression de serveurs. La base NoSQL n'a pas à être cohérente à tout instant

Eventually consistent :
A terme, la base atteindra un état cohérent

Le théorème de CAP

Le théorème de CAP (théorème de Brewer)

- 3 propriétés fondamentales caractérisant une base de données

Consistency (Cohérence)
Une donnée n'a qu'un seul état visible quel que soit le nombre de réplicas. Peut importe le nœud sur lequel le client se connecte.

Availability (Disponibilité)

Tant que le système tourne (distribué ou non), la donnée doit être disponible.

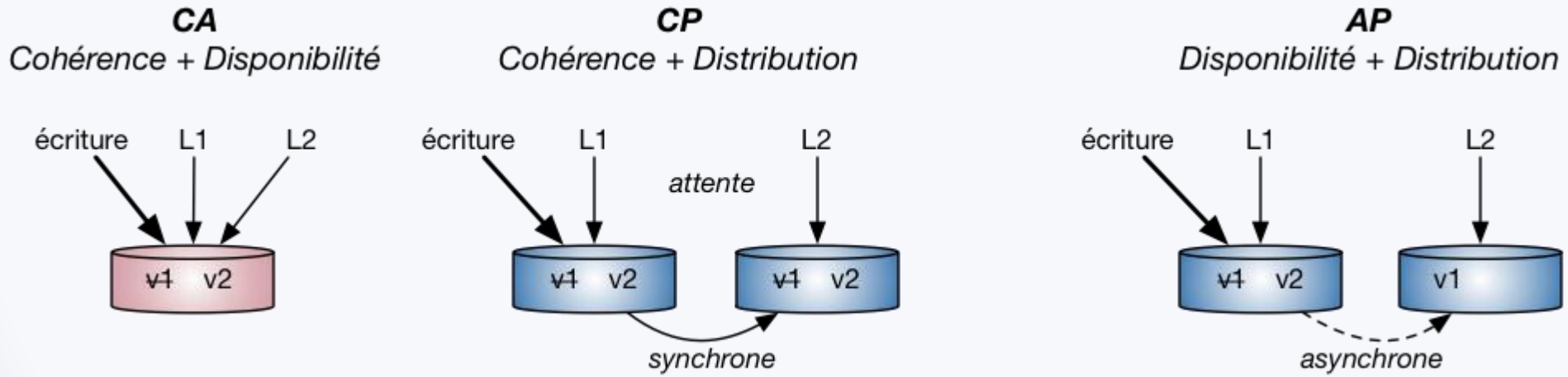
Partition Tolerance (Distribution)

Quel que soit le nombre de serveurs, toute requête doit fournir un résultat correct. Le cluster doit continuer à fonctionner

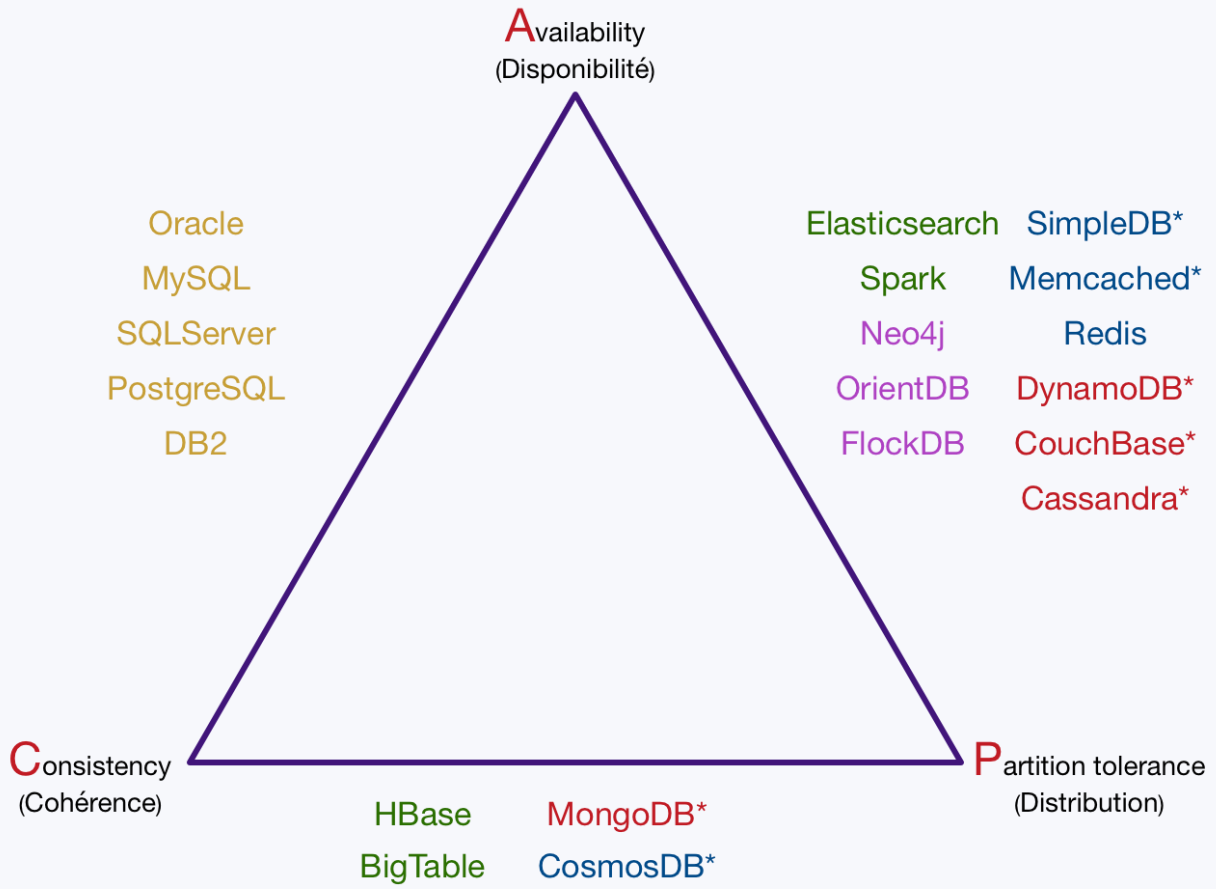
Le théorème de CAP

- Le théorème dit :

« Dans toute base de données, vous ne pouvez respecter au plus que 2 propriétés parmi la cohérence, la disponibilité et la distribution.»



Le triangle de CAP



Modèle

Relationnel
Oriété Clé-Valeur
Oriété Colonne
Oriété Document
Oriété Graphe

Oracle
MySQL
SQLServer
PostgreSQL
DB2

Elasticsearch
Spark
Neo4j
OrientDB
FlockDB

SimpleDB*
Memcached*
Redis
DynamoDB*
CouchBase*
Cassandra*

HBase
BigTable

MongoDB*
CosmosDB*

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

Liens utiles et références

- https://db-engines.com/en/ranking_trend
- https://datascientest.com/nosql-tout-savoir
- https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql