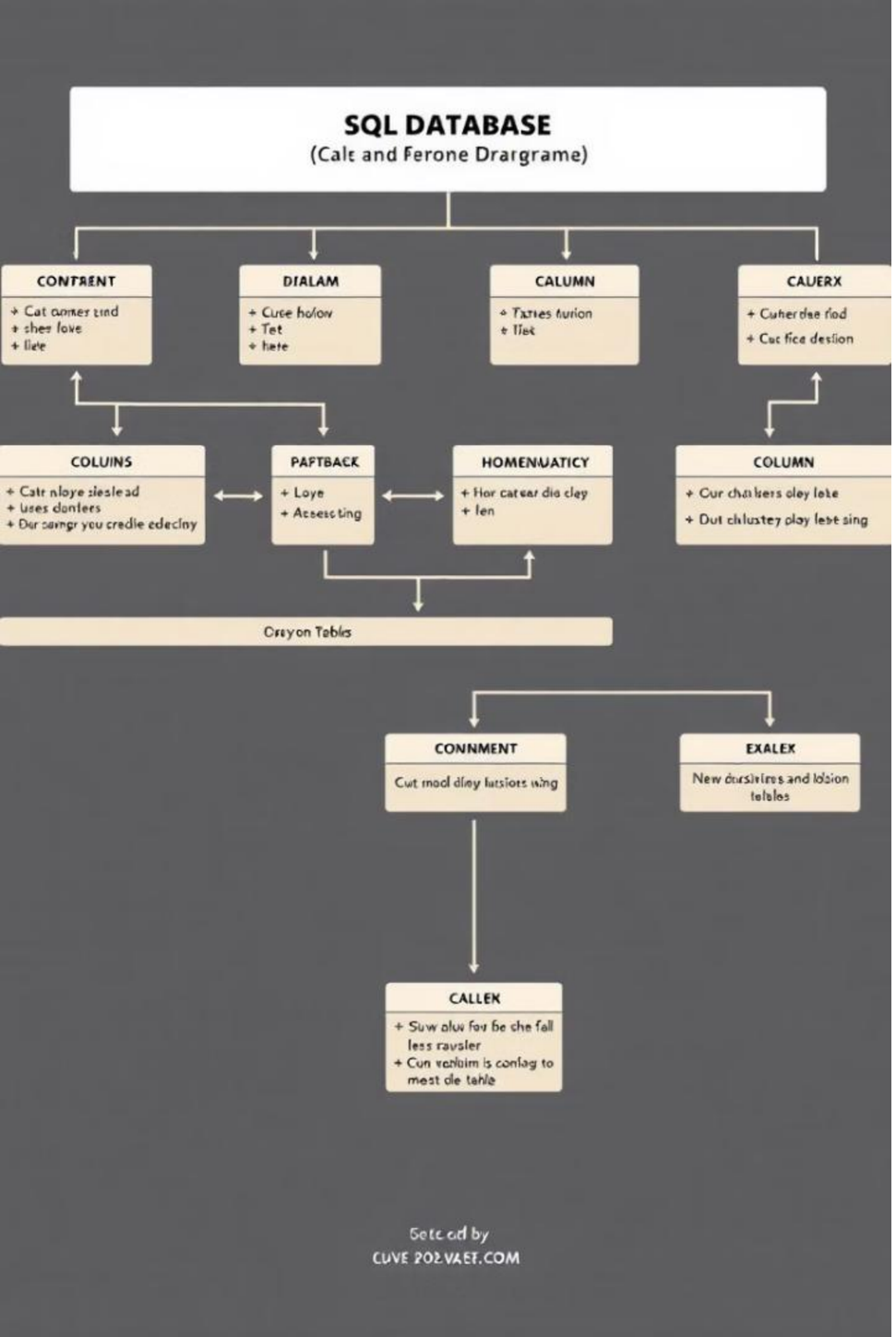
NoSQL
Not Only SQL

Pr. Zainab OUFQIR



Il était une fois ... SQL

1 Signification

SQL signifie "Structured Query Language" (Langage de requête structuré).

2 Type

C'est un système de gestion de base de données relationnel grâce à l'utilisation de clés étrangères

3 Structure

Utilise des tables avec des colonnes définies et des lignes de données.

Jointures en SQL

Définition

Les jointures permettent de lier des données de plusieurs tables.

1

Utilisation

Utilisées pour des requêtes complexes impliquant plusieurs tables.

2

Exemple

utiliser SELECT, INSERT, UPDATE et DELETE pour interagir avec la base de données..

3

Histoire des Bases de Données SQL

Années 1970

Création des premiers systèmes de bases de données relationnelles.

1

Évolution

Développement continu pour répondre aux besoins des entreprises.

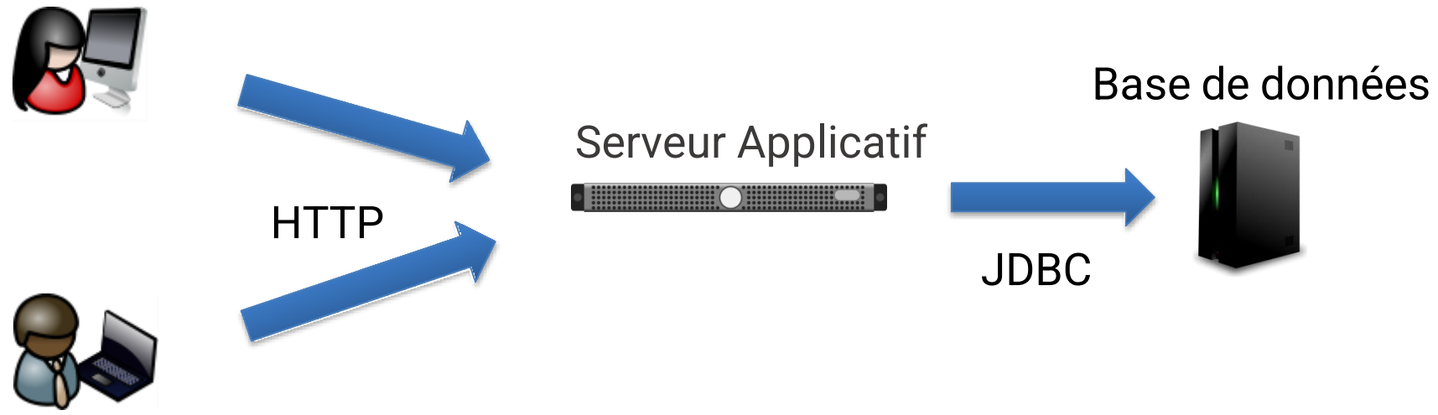
2

Aujourd'hui

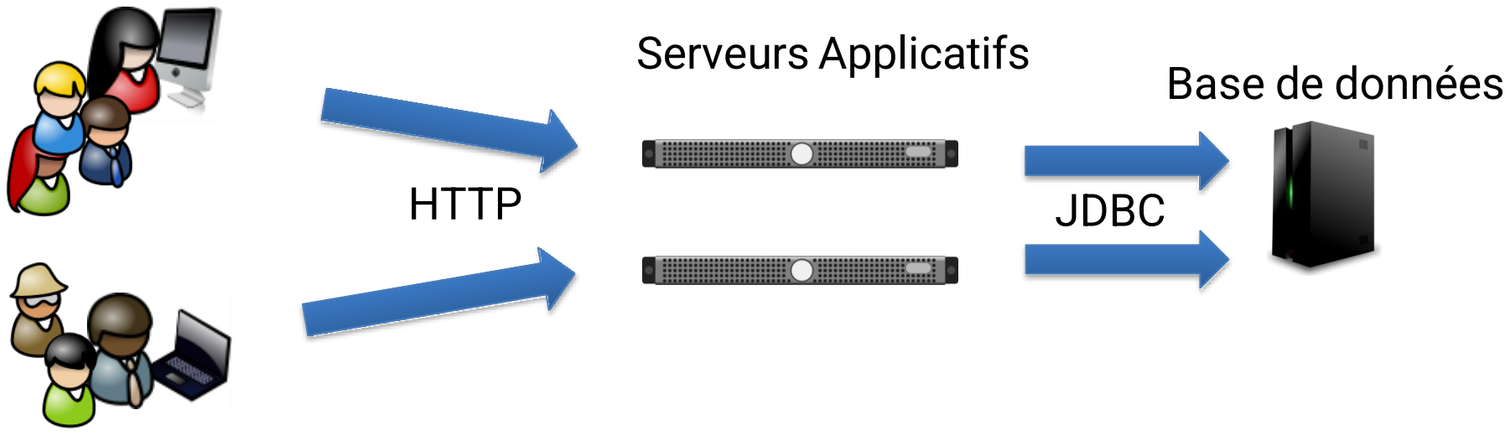
Toujours largement utilisé pour de nombreuses applications.

3

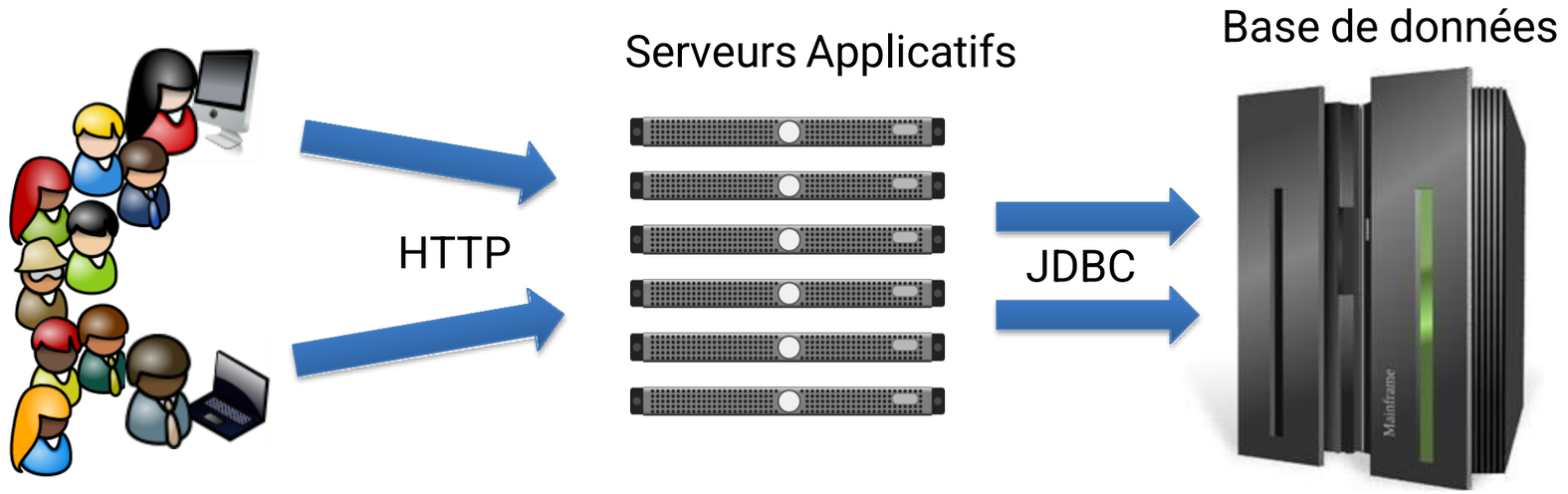
Mise en œuvre d'un SGBDR- cas 1



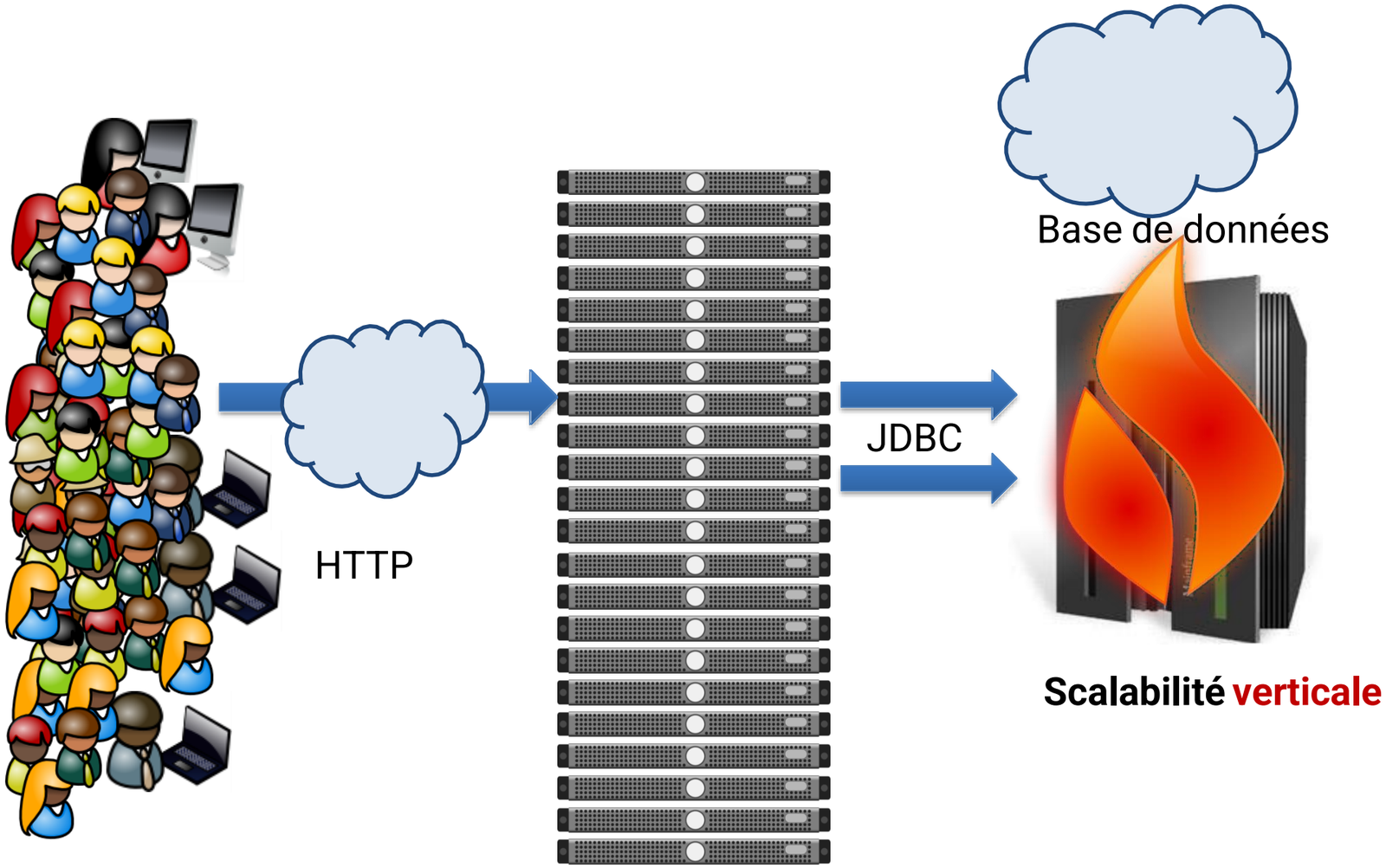
Mise en œuvre d'un SGBDR- cas 2



Mise en œuvre d'un SGBDR- cas 3



Mise en œuvre d'un SGBDR- cas 4



Évolution des Bases SQL : Scaling Vertical

1

Définition

Augmentation des ressources de la machine hébergeant la base de données.

2

Méthode

Ajout de stockage, de RAM ou amélioration du processeur.

3

Limite

Limité par les capacités maximales d'une seule machine.

Limites des SGBD classiques

SGBD Relationnels offrent :

- Un système de jointure entre les tables permettant de construire des requêtes complexes impliquant plusieurs entités.

Contexte fortement distribué : Ces mécanismes ont un coût considérable:

- Avec la plupart des SGBD relationnels, les données d'une BD liées entre elles sont placées sur le même nœud du serveur.
- Si le nombre de liens important, il est de plus en plus difficile de placer les données sur des nœuds différents.

Définition de NoSQL

1

Signification

NoSQL signifie "Not Only SQL" (Pas seulement SQL).

2

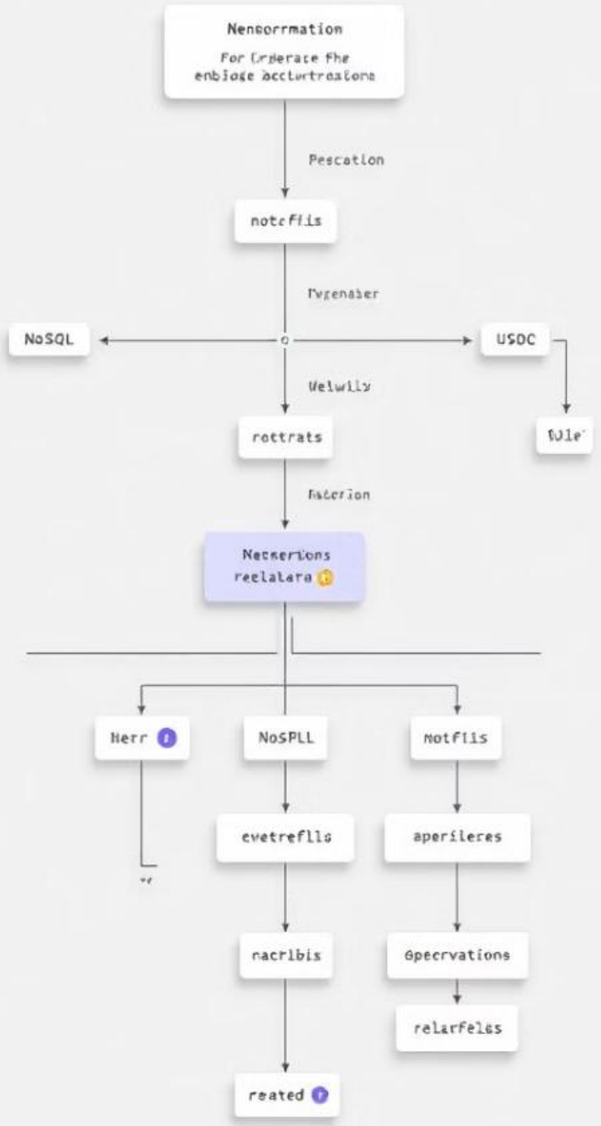
Type

C'est un système de gestion de base de données non relationnel.

3

Structure

Gère efficacement les volumes massifs de données non structurées avec une scalabilité facile.



Absence de Jointures en NoSQL

Concept

Pas de jointures traditionnelles dans les systèmes NoSQL.

Raison

Structure non relationnelle ne nécessite pas de jointures.

Alternative

Utilisation d'autres méthodes pour lier les données si nécessaire.

Histoire des Bases de Données NoSQL

1

Fin des années 2000

Émergence des systèmes NoSQL pour répondre aux nouveaux besoins.

2

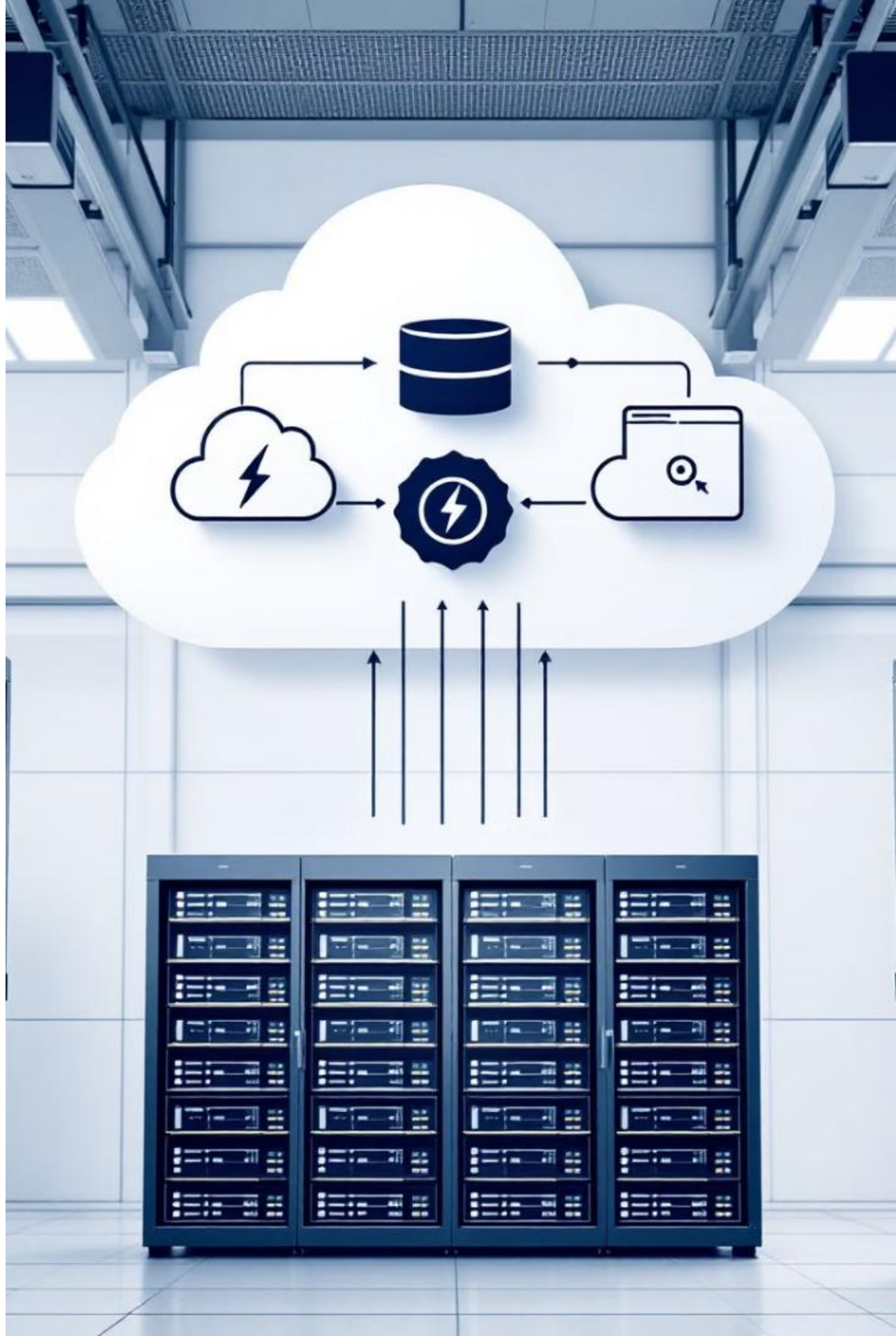
Développement

Croissance rapide avec l'essor du Big Data et du Cloud.

3

Aujourd'hui

Solution privilégiée pour de nombreuses applications web modernes.

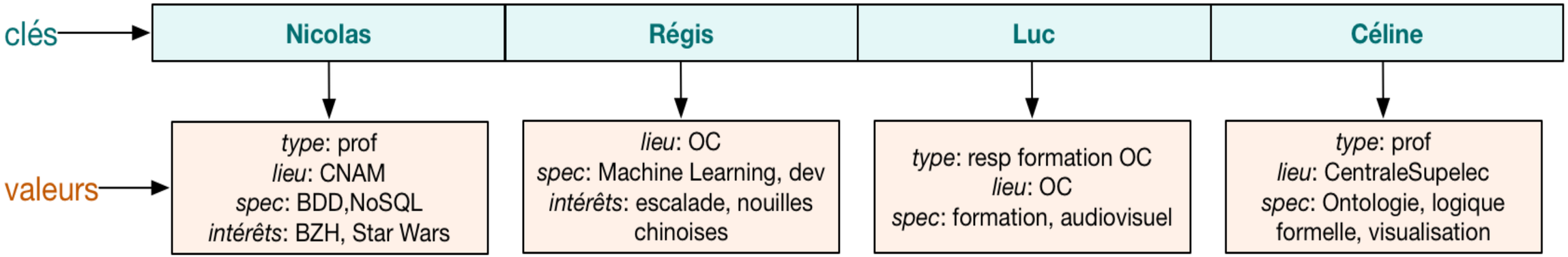


Types de Bases de Données NOSQL

Les clés-valeurs

Une base de données NoSQL de type Clé-valeur est la forme la plus simple de stockage NoSQL où chaque élément est stocké comme une paire composée d'un identifiant unique (la clé) et de sa valeur associée.

- La clé identifie la donnée de manière unique et permet de la gérer.
- La valeur contient n'importe quel type de données.



Les clés-valeurs

Les seules opérations de type CRUD peuvent être utilisées :

- Create (key,value)
- Read (key)
- Update (key,value)
- Delete (key)

Exemple:

- **Redis** (VMWare): Vodafone, Trip Advisor, Nokia, Samsung, Docker
- Memcached **(Danga)** : LiveJournal, Wikipédia, Flickr, Wordpress
- Azure Cosmos DB **(Microsoft)** : Real Madrid, Orange tribes, MSN, LG, Schneider Electric
- SimpleDB **(Amazon)**

Orientées Colonnes

Une base de données NoSQL orientée colonnes stocke les données par colonnes plutôt que par lignes (comme en SQL).

Il est alors possible de focaliser les requêtes sur une ou plusieurs colonnes, sans avoir à traiter les informations inutiles (les autres colonnes).

Stockage orienté lignes

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	

Stockage orienté colonnes

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

Orientées Colonnes

\+ Cette solution est très adaptée pour effectuer des traitements sur des colonnes comme les agrégats (comptage, moyennes, co-occurences...). D'une manière plus concrète, elle est adaptée à de gros calculs analytiques.

\- Toutefois, cette solution est beaucoup moins appropriée pour la lecture de données spécifiques comme pour les clés/valeurs.

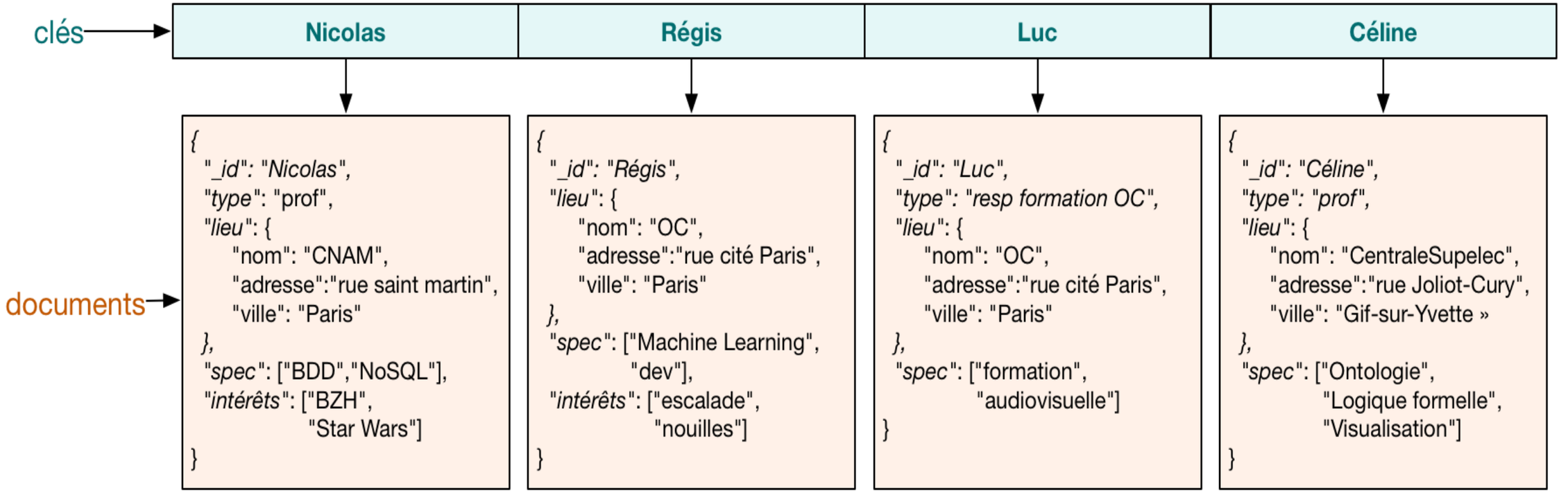
Qui fait de l'orienté-colonnes ?

- **BigTable** (Google)
- **HBase** (Apache, Hadoop)
- **Spark SQL** (Apache)
- **Elasticsearch** (elastic) -> moteur de recherche

Orientée Documents

Le but de ce stockage est de manipuler des documents contenant des informations avec une structure complexe (types, listes, imbrications).

Modèle clé-valeur où la valeur est un document (lisible par un humain) au format semi-structuré hiérarchique (XML,YAML,JSON ou BSON, etc.)



Orientée Documents

Avantage : pouvoir récupérer, via une seule clef, un ensemble d'informations structurées de manière hiérarchique (Dans les bases relationnelles, cela impliquerait plusieurs jointures)

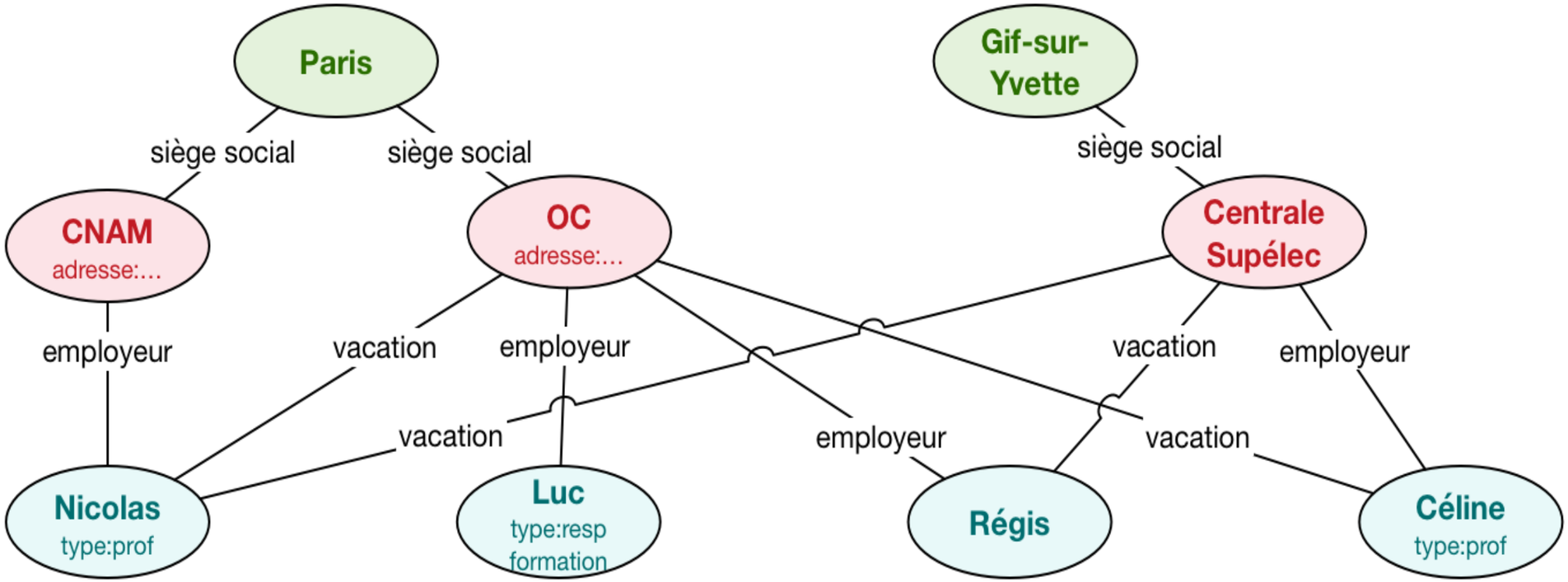
Quelles solutions pour gérer vos documents :

- **MongoDB** (MongoDB) : ADP, Adobe, Bosch, Cisco, eBay, Electronic Arts, Expedia, Foursquare
- **CouchBase** (Apache, Hadoop) : AOL, AT&T, Comcast, Disney, PayPal, Ryanair
- **DynamoDB** (Amazon) : BMW, Dropcam, Duolingo, Supercell, Zynga
- Cassandra (Facebook -> Apache) : NY Times, eBay, Sky, Pearson Education

Orienté Graphes

Conçues pour les données dont les relations sont représentées comme graphes, et ayant des éléments interconnectés, avec un nombre indéterminé de relations entre elles.

Exemple d'un réseau social : dans certains cas, il devient très complexe de calculer la distance entre deux personnes non directement connectées. Et c'est ce type d'approche que résolvent les bases orientées Graphe.



Orienté Graphes

C'est ce type de relations et de calculs de distance que les bases de données orientées graphes gèrent efficacement, notamment pour :

- Les recommandations d'amis ("Vous connaissez peut-être...")
- L'analyse des réseaux d'influence
- La détection de communautés

Quelles bases de données gèrent de gros graphes ?

- **Neo4j** : eBay, Cisco, UBS, HP, TomTom, The National Geographic Society
- **OrientDB** (Apache) : Comcast, Warner Music Group, Cisco, Sky, United Nations, Verisign
- **FlockDB** (Twitter) : Twitter

NoSQL et BigData

Avec l'émergence du Big Data, les besoins en base de données ont rapidement dépassé les capacités des bases de données SQL. En conséquence, la technologie NoSQL a été créée pour résoudre les problèmes d'évolutivité.

Adaptation des BD NOSQL au BigData:

- **Volume**: NoSQL gère des pétaoctets de données en les distribuant automatiquement sur plusieurs serveurs, sans limite pratique de stockage.
- **Variété**: Accepte tous types de données (structurées, semi-structurées, non structurées) sans nécessiter de schéma prédéfini.
- **Vélocité**: Permet le traitement en temps réel grâce à des écritures/lectures rapides et des opérations parallèles.
- **Distribution**: Les données sont automatiquement réparties sur plusieurs serveurs pour optimiser performances et disponibilité.

Évolution des Bases NoSQL : Scaling Horizontal

1

Définition

Répartition de la base de données sur plusieurs serveurs.

2

Méthode

Ajout de nouveaux serveurs pour répartir la charge.

3

Avantage

Permet une meilleure gestion des pics de trafic.

Pourquoi SQL

Et pourquoi donc faire encore de la base de données relationnelle? Le NoSQL est très tentant lorsque l'on voit toutes les possibilités qu'il peut offrir. Mais il ne faut pas négliger certains fondamentaux qui rendent les SGBDR incontournables.

Cela se résume dans une combinaison qu'il ne faut pas négliger :

1. Faire des jointures entre les tables de la base de données
2. Faire des requêtes complexes avec un langage de haut niveau sans se préoccuper des couches basses
3. Gérer l'intégrité des données de manière infaillible, ça veut dire être capable de garantir le contenu de la base de données, quelles que soient les mises à jour effectuées sur les données, ainsi que sa pérennité.

ACID : Propriétés Fondamentales des SGBDR

ACID est un acronyme qui définit les quatre propriétés essentielles garantissant la fiabilité des transactions dans une base de données relationnelle : Atomicité, Cohérence, Isolation et Durabilité.

Une transaction dans une base de données relationnelle est un ensemble d'opérations (INSERT, UPDATE, DELETE, SELECT) qui sont traitées comme une seule unité de travail indivisible.

Une transaction suit le principe du "tout ou rien" : soit toutes les opérations de la transaction réussissent et sont validées (COMMIT), soit aucune n'est appliquée en cas d'erreur (ROLLBACK).

Atomicité : Tout ou Rien

1 Définition

L'atomicité garantit qu'une transaction s'exécute entièrement ou pas du tout.

2 Principe

Pas de résultat partiel. La transaction réussit complètement ou échoue totalement.

3 Importance

Assure l'intégrité des données en évitant les états intermédiaires incohérents.

Cohérence : Intégrité des Données

1

Avant Transaction

Toutes les règles sont respectées (pas de solde négatif, stocks valides, relations entre tables correctes)

2

Pendant Transaction

Chaque opération est vérifiée en temps réel (validation des montants, vérification des contraintes, contrôle des relations)

3

Après Transaction

Si succès : nouvelles données cohérentes, toutes les règles respectées

Si erreur : retour automatique à l'état initial, aucune modification partielle

Isolation : Transactions Indépendantes

1

Principe

Une transaction ne doit pas être affectée par ce que font les autres transactions en cours, comme si elle était seule dans la base de données.

2

Exemple

Deux transferts bancaires simultanés ne doivent pas interférer entre eux.

3

Objectif

Garantir que plusieurs opérations simultanées donnent le même résultat que si elles étaient faites l'une après l'autre, évitant ainsi les erreurs de calcul.

Durabilité : Persistance des Données

Définition

Une fois qu'une transaction est validée (commit), les changements sont définitivement enregistrés et ne peuvent pas être perdus.

Application

Si l'ordinateur s'arrête brutalement ou qu'il y a une coupure de courant, les données validées seront toujours présentes au redémarrage.

Importance

La durabilité assure qu'aucune transaction validée ne peut être perdue, ce qui est crucial pour la confiance dans le système.

ACID et Bases SQL

1

Conception

Les bases SQL sont conçues pour respecter les principes ACID.

2

Transactions

Elles gèrent efficacement les transactions conformes à ACID.

3

Performance

L'adhésion à ACID peut impacter les performances et la complexité.

NoSQL : Une Alternative à ACID

Les propriétés ACID ne sont pas applicables dans un contexte distribué tel que le NoSQL.

Exemple d'une transaction de cinq opérations (lecture/écriture) :

- Cela implique une synchronisation entre cinq serveurs pour garantir l'atomicité, la cohérence et l'isolation.
- Cela se traduit par des latences dans les transactions (en cours et en concurrence).
- Ce qui n'est pas tolérable lorsque justement on veut éviter ces latences en distribuant les calculs.

Du coup, les propriétés BASE ont été proposées pour caractériser les bases NoSQL :

- **B**asically **A**vailable : Le système reste toujours disponible et répond aux requêtes même en cas de panne partielle ou de forte charge, quitte à retourner des résultats partiels ou non à jour.

  Exemple : Si un serveur tombe en panne, les autres continuent à fonctionner

- **S**oft-state : Les données peuvent changer au fil du temps sans intervention directe car le système distribué met progressivement à jour ses différents nœuds, acceptant une incohérence temporaire.

  Exemple : Un post sur Facebook peut apparaître à des moments différents chez différents utilisateurs

- **E**ventually consistent : Le système garantit qu'à terme, si aucune nouvelle mise à jour n'est effectuée, toutes les copies des données finiront par être identiques et cohérentes sur tous les nœuds.

  Exemple : Quand vous changez votre photo de profil, certains amis la voient immédiatement, d'autres quelques secondes plus tard

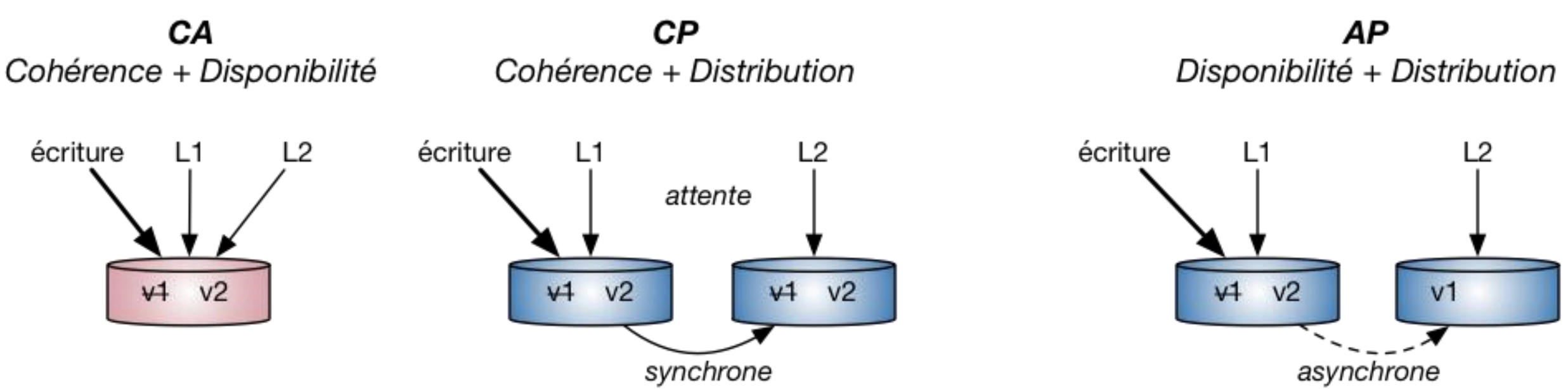
Théorème de Brewer dit "théorème de CAP"

En 2000, Eric A. Brewer a formalisé un théorème très intéressant reposant sur 3 propriétés fondamentales pour caractériser les bases de données (relationnelles, NoSQL et autres) :

1. Consistency (Cohérence) : Une donnée n'a qu'un seul état visible quel que soit le nombre de réplicas.
2. Availability (Disponibilité) : Tant que le système tourne (distribué ou non), la donnée doit être disponible.
3. Partition Tolerance (Distribution) : Le système continue à fonctionner et à répondre, même si une panne empêche certains serveurs de communiquer entre eux.

Le théorème de CAP dit :

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



CA (**C**onsistency-**A**vailability), il représente le fait que lors d'opérations concurrentes sur une même donnée, les requêtes L1 et L2 retournent la nouvelle version (v2) et sans délai d'attente. Cette combinaison n'est possible que dans le cadre de bases de données transactionnelles telles que les SGBDR.

CP (**C**onsistency-**P**artition) propose de distribuer les données sur plusieurs serveurs en garantissant la tolérance aux pannes (réplication). La gestion de la cohérence nécessite un protocole de synchronisation des réplicas, introduisant des délais de latence dans les temps de réponse (L1 et L2 attendent la synchronisation pour voir v2). C'est le cas de la base NoSQL MongoDB.

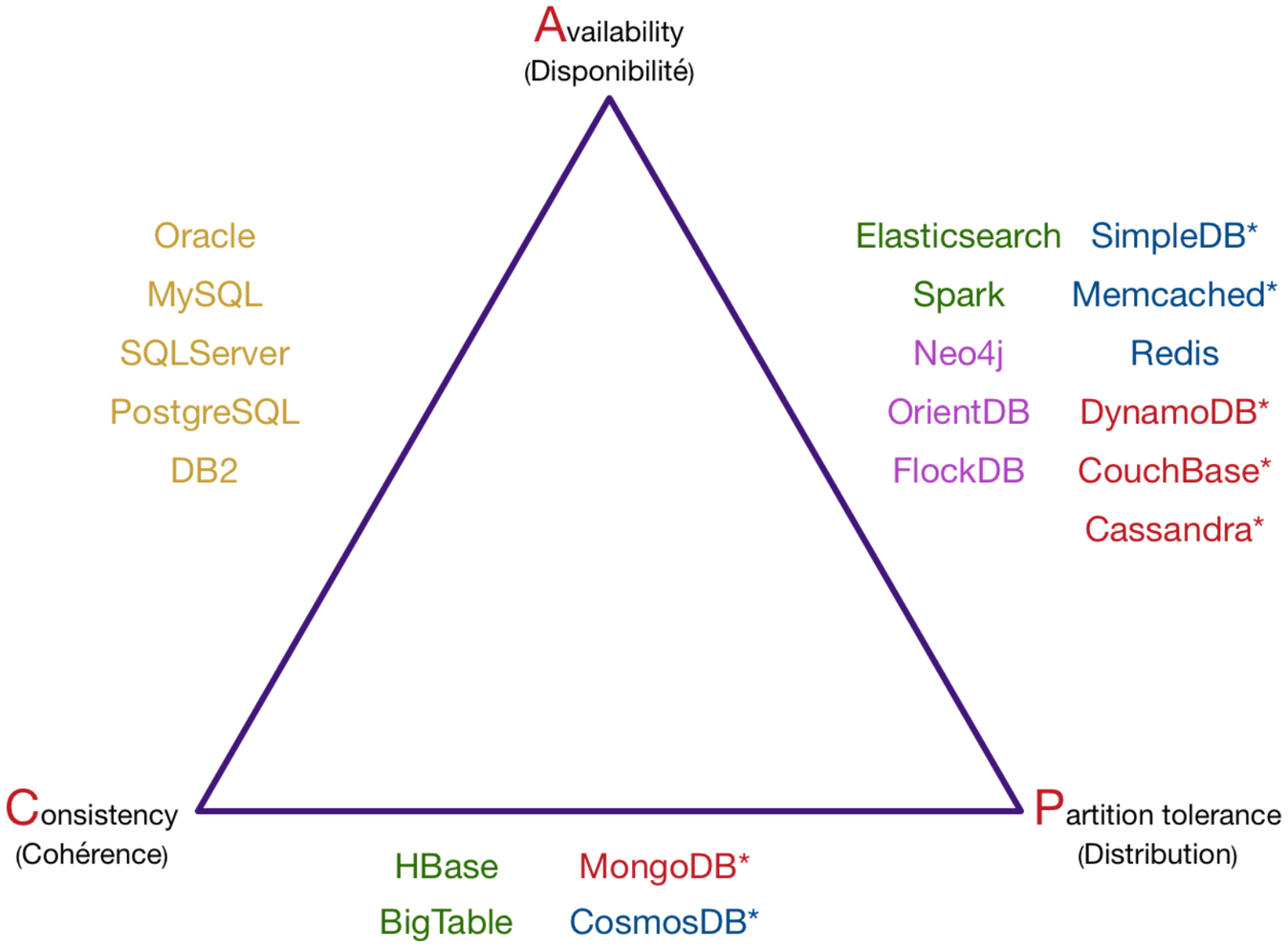
AP (**A**vailability-**P**artition) à contrario s'intéresse à fournir un temps de réponse rapide tout en distribuant les données et les réplicas. De fait, les mises à jour sont asynchrones sur le réseau (**L1** voit la version v2, tandis que L2 voit la version v1).

Le triangle de CAP et les bases de données

Modèle

Relationnel
Orienté Clé-Valeur
Orienté Colonne
Orienté Document
Orienté Graphe

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



Quels sont les critères de choix de NoSQL ?

Les cas principaux où il est préférable d'utiliser NoSQL :

- **Grands Volumes de Données**: Lorsque vous traitez des quantités massives de données (Big Data) nécessitant une scalabilité horizontale facile et économique.
- **Données Non Structurées/Semi-structurées**: Quand vos données ont une structure variable ou évolutive, comme des profils utilisateurs, des contenus de blogs, des logs ou des données IoT.
- **Haute Disponibilité**: Requise Pour les applications nécessitant un temps de réponse rapide et une disponibilité constante (réseaux sociaux, jeux en ligne, applications temps réel).
- **Scalabilité Horizontale**: Lorsque vous devez pouvoir ajouter facilement des serveurs pour gérer plus de charge sans interruption de service.