

Bases de Données NoSQL et Big Data

Chapitre 1 : Introduction aux BD NoSQL

Pr. ZAIDOUNI Dounia

Filière : SUD, Semestre 3, INE2
Institut National des Postes et Télécommunications (INPT)

21 Novembre 2024

Présentation du syllabus du cours

- Chapitre 1: Introduction aux BD NoSQL
 - SGBD Relationnels transactionnels (Définition, propriétés ACID, limites, ...)
 - Propriété BASE et théorème du CAP (ou théorème de Brewer)
 - Présentation des BD NoSQL
- Chapitre 2: Fondements et Typologie des BD NoSQL
- Chapitre 3: Introduction aux Big Data
- Chapitre 4: MongoDB
- Chapitre 5: Apache Hbase

Outline

- 1 Contexte
- 2 SGBD Relationnels transactionnels
- 3 Propriété BASE et Théorème du CAP (ou théorème de Brewer)
- 4 Présentation des BD NoSQL

Plan

1 Contexte

2 SGBD Relationnels transactionnels

- Définition de transaction et d'intégrité référentiel
- Les propriétés ACID
- Limites des SGBD Relationnels transactionnels

3 Propriété BASE et Théorème du CAP (ou théorème de Brewer)

4 Présentation des BD NoSQL

Contexte

- Plusieurs sources de données :
 - Les informations générées par les grandes plateformes et applications Web (Google, Facebook, Twitter, LinkedIn, Amazon, ...) comme :
 - Emails, images, vidéos, audios, transactions d'achat, logs, etc.
 - Les informations récupérées par des réseaux de capteurs pour gérer des Smart cities ou des Smart Building, comme :
 - Climat, trafic, données de géolocalisation, pression, température, etc.
- Un volume considérable de données à gérer par ces applications nécessitant une distribution des données et leur traitement sur de nombreux serveurs.
- Manipulation de données associées à des objets complexes, non structurées et hétérogènes et qui ne nécessite pas de déclarer au préalable l'ensemble des champs représentant l'objet.

Problèmes

- ⇒ Limites de l'utilisation des SGBD traditionnels (relationnels et transactionnels) basés sur SQL.
- ⇒ Nécessité de nouvelles approches de stockage et de gestion des données permettant une meilleure scalabilité dans des contextes fortement distribués.

Plan

1 Contexte

2 SGBD Relationnels transactionnels

- Définition de transaction et d'intégrité référentiel
- Les propriétés ACID
- Limites des SGBD Relationnels transactionnels

3 Propriété BASE et Théorème du CAP (ou théorème de Brewer)

4 Présentation des BD NoSQL

- Les SGBD Relationnels offrent un système de jointure entre les tables permettant de construire des requêtes complexes impliquant plusieurs entités.
- Ils sont généralement transactionnels :
 - Ils réalisent des **transactions** respectant les propriétés **ACID** (Atomicité, Cohérence, Isolation et Durabilité).
- Les SGBD Relationnels offrent aussi un système d'**intégrité référentielle** permettant d'assurer la cohérence du contenu de la base de données.

Définition de transaction

- C'est une opération sur les données telle qu'une réservation, achat, paiement, transfert d'argent, etc.
- Elle est mise en œuvre via une suite de tâches qui font passer la base de données d'un état A (antérieur à la transaction) à un état B postérieur.
- Exemple: Lors d'une opération informatique de transfert d'argent d'un compte bancaire sur un autre compte bancaire :
 - Il y a une tâche de retrait d'argent sur le compte source et une de dépôt sur le compte cible.
 - Le programme qui effectue cette transaction va s'assurer que les deux opérations peuvent être effectuées sans erreur, et dans ce cas, la modification deviendra alors effective sur les deux comptes. Si ce n'est pas le cas l'opération est annulée. Les deux comptes gardent leurs valeurs initiales.
 - On garantit ainsi la cohérence des données entre les deux comptes.

Définition d'intégrité référentielle

- C'est une situation dans laquelle pour chaque information d'une table A qui fait référence à une information d'une table B, l'information référencée existe dans la table B.
- Une clé étrangère, dans une base de données relationnelle, est une contrainte qui garantit l'intégrité référentielle entre deux tables.
- L'intégrité référentielle est un gage de cohérence du contenu de la base de données.
- Exemple: On définira qu'un livre est écrit par un auteur. Une contrainte d'intégrité référentielle :
 - interdira l'effacement d'un auteur, tant que dans la base de données il existera au moins un livre se référant à cet auteur.
 - interdira également d'ajouter un livre si l'auteur n'est pas préalablement inscrit dans la base de données.

Les propriétés ACID

C'est un ensemble de propriétés qui garantissent qu'une transaction informatique est exécutée de façon fiable.

- Atomicité
- Cohérence
- Isolation
- Durabilité

Atomicité

- Il assure qu'une transaction se fait au complet ou pas du tout :
 - Si une partie d'une transaction ne peut être faite, il faut effacer toute trace de la transaction et remettre les données dans l'état où elles étaient avant la transaction.
- Il n'y a jamais de transactions à moitié terminées.
- L'atomicité doit être respectée dans toutes situations, comme une panne d'électricité, une défaillance de l'ordinateur, ou une panne d'un disque magnétique.

- Elle assure que chaque transaction amènera le système d'un état valide à un autre état valide.
- Tout changement à la base de données doit être valide selon toutes les règles fonctionnelles définies.

Exemples de règles fonctionnelles :

- Contrainte d'intégrité référentiel.
- Contraintes d'intégrité :

Ils sont des clauses permettant de contraindre la modification de tables afin que les données saisies dans la base soient conformes aux données attendues. Par exemple en SQL :

- Le mot clé 'NOT NULL' permet de forcer la saisie d'un champ.
- La clause 'UNIQUE' permet de vérifier que la valeur saisie pour un champ n'existe pas déjà dans la table.

- Toute transaction doit s'exécuter comme si elle était la seule sur le système. Aucune dépendance entre les transactions.
- La propriété d'isolation assure que l'exécution simultanée de transactions produit le même état que celui qui serait obtenu par l'exécution en série des transactions. Chaque transaction doit s'exécuter en isolation totale.
- La transaction va travailler dans un mode isolé où elle seule peut voir les données qu'elle est en train de modifier, cela en attente d'un nouveau point de synchronisation (checkpoint).
 - Grâce au **Checkpoint et au Rollbacks-Recovery**, le système garantit la tolérance aux pannes et la visibilité sur les données antérieures.
 - Un Rollback-Recovery se produit dans les systèmes de base de données quand une transaction 'T1' provoque un échec et qu'une récupération doit être effectuée.

- Lorsque la transaction est achevée, le système est dans un état stable durable :
 - soit à l'issue d'une modification transactionnelle réussie,
 - soit à l'issue d'un échec qui se solde par le retour à l'état stable antérieur.
- La durabilité assure que lorsqu'une transaction est confirmée, elle doit être enregistrés de façon permanente même à la suite d'une panne d'électricité, d'une panne de l'ordinateur ou d'un autre problème.

Points forts des SGBD Relationnels transactionnels

Les SGBD Relationnels offrent :

- Un système de jointure entre les tables permettant de construire des requêtes complexes impliquant plusieurs entités.
- Un système d'intégrité référentielle permettant :
 - Fiabilité des Données : Elles Garantissent que les données sont valides et cohérentes.
 - Automatisation des Règles Métier : Appliquent automatiquement des règles sans intervention manuelle.
 - Réduction des Erreurs : Empêchent les erreurs humaines ou logicielles, comme l'insertion de données incohérentes.
 - Efficacité des Requêtes : Simplifient l'écriture des requêtes SQL en s'assurant que les relations sont cohérentes.
- Les SGBDR sont performants pour les systèmes où :
 - Les données sont hautement structurées.
 - La cohérence stricte est essentielle.
 - Le volume des données reste modéré.
 - Les besoins transactionnels sont critiques (comme les banques).

Limites des SGBD Relationnels transactionnels

- Les SGBDR sont principalement conçus pour évoluer verticalement (ajouter plus de ressources matérielles à un seul serveur). Ils peinent à évoluer horizontalement. Dans des environnements avec de grandes quantités de données (Teraoctets et Zettaoctets) ou avec un nombre élevé de requêtes simultanées, la scalabilité verticale atteint vite ses limites.
- Les SGBDR ne sont pas bien optimisés pour gérer des types de données comme les documents, les graphiques ou les données hiérarchiques. Par exemple, manipuler des structures JSON complexes ou traverser des graphes sociaux (comme les réseaux d'amis) peut être inefficace dans un SGBDR.
- Les SGBDR ne sont pas conçus pour gérer efficacement des volumes massifs de données non structurées ou semi-structurées générées à grande vitesse comme les systèmes de streaming en temps réel.

Limites des SGBD Relationnels transactionnels

- Si on doit distribuer les traitements de données entre différents serveurs alors il est coûteux et difficile de maintenir les contraintes 'ACID' à l'échelle du système distribué entier tout en maintenant des performances correctes.
 - Si le nombre de liens est important, il est de plus en plus difficile de placer les données sur des nœuds différents.
 - Difficulté à mettre à l'échelle les opérations d'écriture telles que les insertions et les mises à jour, elles peuvent devenir un goulot d'étranglement lorsque la taille des données augmente.
- Les SGBDR sont basées sur un schéma fixe défini à l'avance. Cela peut rendre difficile la gestion de types de données non structurées ou semi-structurées. Toute modification du schéma (ajouter ou supprimer une colonne, par ex.) peut entraîner des interruptions, des temps d'arrêt ou des efforts de migration importants, surtout avec de grandes bases.

⇒ La plupart des SGBD adaptés aux systèmes distribués relâchent les contraintes ACID, ou même ne proposent pas de gestion de transactions.

Stratégies possibles des traitements sur des données

Dans un environnement distribué, il y a deux stratégies possibles pour pouvoir appliquer des traitements sur des données :

- **Par distribution des traitements** : on distribue ces traitements sur un nombre de machines important afin d'absorber des charges très importantes et on envoie les données aux endroits appropriés, et on enchaîne les exécutions distantes (scénario type Workflow implémentable avec des web services).
- **Par distribution des données** : on distribue les données sur un nombre important de serveurs afin de stocker de très grands volumes de données, et on pousse les programmes vers ces serveurs (plus efficace de transférer un petit programme sur le réseau plutôt qu'un grand volume de données. Ex: algorithme MapReduce).

Plan

1 Contexte

2 SGBD Relationnels transactionnels

- Définition de transaction et d'intégrité référentiel
- Les propriétés ACID
- Limites des SGBD Relationnels transactionnels

3 Propriété BASE et Théorème du CAP (ou théorème de Brewer)

4 Présentation des BD NoSQL

Propriété BASE

- A l'opposé des propriétés 'ACID', les propriétés 'BASE' caractérisent les SGBD issus du cloud computing et des systèmes distribués.
 - Ces SGBD privilégient la haute disponibilité des données distribuées, la rapidité, la simplicité au détriment de la cohérence et de l'exactitude de la réponse.
- Les propriétés 'BASE' sont :
 - **Basically Available, Soft state et Eventual consistency.**

Propriété BASE

- **Basically Available** : Le système garantit la disponibilité des données et il y aura une réponse à toute demande. Mais cette réponse pourrait être un "échec" pour obtenir les données demandées, on peut avoir des indisponibilités sur de courtes périodes).
- **Soft state** : L'état de la BD n'est pas garanti à un instant donné (les mises à jour ne sont pas immédiates). L'état du système peut changer avec le temps, des changements peuvent survenir en raison de la "cohérence éventuelle", ainsi l'état du système est toujours "ouple".
- **Eventual consistency** : La cohérence des données à un instant donné n'est pas primordiale mais assurée plus tard (en utilisant différents mécanismes par exemple le 'verrouillage optimiste'). Au fur et à mesure que des données sont ajoutées au système, l'état est progressivement répliqué sur les autres nœuds.

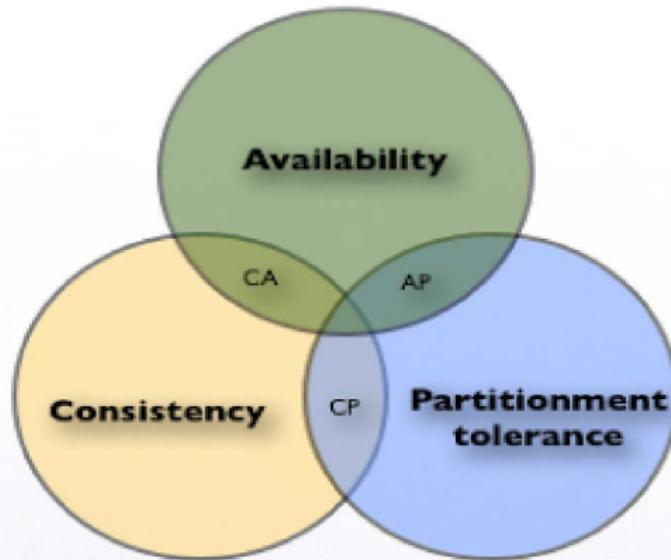
Propriété 'CAP'

3 propriétés fondamentales pour les systèmes distribués :

- **Consistency** : toutes les nœuds dans un système distribué renvoient la même valeur.
- **Availability** : Chaque nœud non défaillant renvoie une réponse pour toutes les demandes de lecture et d'écriture dans un délai raisonnable.
- **Partitionment tolerance** : Le système continue de fonctionner et maintient sa cohérence malgré le partitionnement dans le système distribué.

Propriété 'CAP'

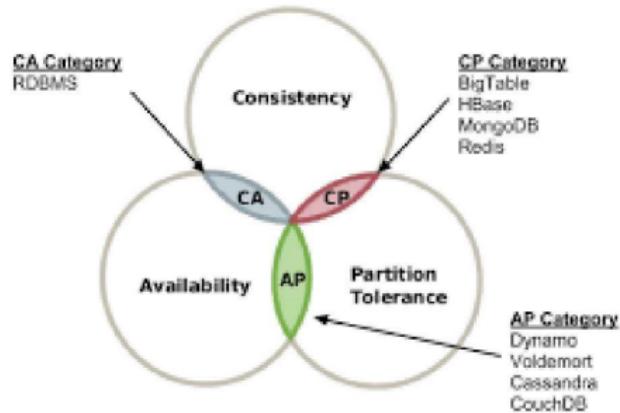
- Consistency (Consistance ou Cohérence), Availability (Disponibilité) et Partition tolerance (Tolérance au partitionnement).



Théorème du CAP (ou théorème de Brewer)

Theorem

Dans un système distribué, il est impossible d'obtenir ces 3 propriétés en même temps, il faut en choisir 2 parmi les 3.



- Les SGBDR sont AC (Available and Consistent).
- Les SGBD "NoSQL" sont CP (Consistent and Partition tolerance) ou AP (Available and Partition tolerance).

Consistance (Consistency) (1/2)

- Un système S est dit consistant si on peut garantir qu'une fois qu'on y enregistre un état, il donnera le même état à chaque opération suivante, jusqu'à ce que cet état soit explicitement changé par quelque chose en dehors du système S.

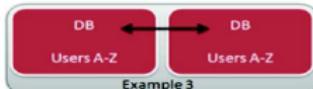


- Ex1 : Une instance de la BD est automatiquement pleinement consistante puisqu'il n'y a qu'un seul noeud qui maintient l'état.



- Ex2 : Si 2 serveurs de BD sont impliqués, et si le système est conçu de telle sorte que toutes les clés de 'a' à 'm' sont conservées sur le serveur 1, et les clés 'n' à 'z' sont conservées sur le serveur 2, alors le système peut encore facilement garantir la consistance.

Consistance (Consistency) (2/2)



- Ex3 :

Supposons 2 BD avec des répliques. Si une des BD fait une opération d'insertion de ligne, cette opération doit être aussi faite (committed) dans la seconde BD avant que l'opération soit considérée comme complète.

- Pour avoir une consistance de 100% dans un tel environnement répliqué, les noeuds doivent communiquer.
- Plus le nombre de répliques est grand plus les performances d'un tel système sont mauvaises.

Disponibilité (Availability)

Les BD dans Ex1 ou Ex2 ne sont pas fortement disponibles :

- dans Ex1 : si le noeud tombe en panne, on perd 100% des données.
- dans Ex2 : si le noeud tombe en panne, on perd 50% des données.
- dans Ex3 : une simple réPLICATION de la BD sur un autre serveur assure une disponibilité de 100%.
- augmenter le nb de noeuds avec des copies des données (répliques) augmente directement la disponibilité du système, en le protégeant contre les défaillances matérielles.
- les répliques aident aussi à équilibrer la charge d'opérations concurrentes, notamment en lecture.

Tolérance au partitionnement (Partitionment tolerance) (1/2)

- Supposons que la "consistance" et la "disponibilité" sont assurées.
- Dans le cas Ex3, supposons que les 2 serveurs de BD sont dans 2 Data Centers différents, et que l'on **perde la connexion réseau** entre les 2 Data Centers. Faisant que les 2 BD seront incapables de synchroniser leurs états.
- Si on parvient à gérer les opérations de lecture/écriture sur ces 2 BD, on peut peut-être prouver que les 2 serveurs ne seront plus consistants.

Tolérance au partitionnement (Partitionment tolerance) (2/2)

- Une application bancaire gardant à tout moment "l'état de votre compte" est l'exemple parfait du problème des enregistrements inconsistants :
 - Si un client retire 1000 Dirhams à Rabat, cela doit être immédiatement répercuté à Casablanca, afin que le système sache exactement combien il peut retirer à tout moment.
 - Si le système ne parvient pas à le faire, cela pourrait mécontenter de nombreux clients.
- Si les banques décident que la "consistance" est très importante, et désactivent les opérations d'écriture lors de la panne, alors la "disponibilité" du cluster sera perdu puisque tous les comptes bancaires dans les 2 villes seront désormais gelés jusqu'à ce que le réseau soit de nouveau opérationnel.

Plan

1 Contexte

2 SGBD Relationnels transactionnels

- Définition de transaction et d'intégrité référentiel
- Les propriétés ACID
- Limites des SGBD Relationnels transactionnels

3 Propriété BASE et Théorème du CAP (ou théorème de Brewer)

4 Présentation des BD NoSQL

Définition du NoSQL (1/2)

- Ce n'est pas (comme son nom le laisse supposer) No SQL (pas de SQL) mais plutôt NoSQL "Not only SQL" (pas seulement SQL).
- NoSQL désigne une famille de SGBD qui s'écarte du paradigme classique des bases relationnelles.
- La raison principale de l'émergence et de l'adoption des SGBD NoSQL serait le développement des clusters de serveurs et la nécessité de posséder des SGBD adaptées à ce modèle d'infrastructure distribuées.
- Cette architecture distribuée fonctionnant avec des **agrégats** répartis sur différents serveurs permettant des accès et modifications concurrentes mais imposant également de remettre en cause de nombreux fondements de l'architecture SGBD relationnelle traditionnelle, notamment les propriétés 'ACID'.

Définition du NoSQL (2/2)

- Le terme NoSQL a été proposé par 'Carl Strozzi' en 1998.
- Les SGBD NoSQL ont vu le jour à la fin des années 2000 avec le développement de grandes entreprises internet (Google, Amazon, Ebay...) brassant des quantités énormes de données.
- Les SGBD NoSQL n'interdisent pas le langage de requête structurée SQL.
 - Certains systèmes NoSQL sont intégralement non relationnels,
 - D'autres se contentent d'éviter certaines fonctions relationnelles, telles que les schémas de tables fixes et les opérations de jointure.

Principaux atouts des SGBD NoSQL

Les principaux atouts des SGBD NoSQL sont :

- Évolutivité,
- Disponibilité,
- Tolérance aux pannes.

Caractéristiques générales des SGBD NoSQL

- **Pas forcément de schéma normalisé** (schema-less) ou schéma dynamique :
 - Les SGBD NoSQL peuvent ignorer cette étape de définition de schéma et stocker des données hétérogènes au fur et à mesure de leur alimentation. Cela permet une grande flexibilité dans l'application.
- Pas forcément de jointure mais multiplication des requêtes.
- Structures de données complexes, imbriquées et hétérogènes.
- **Architecture distribué** avec un partitionnement horizontal des données sur plusieurs serveurs généralement par utilisation d'algorithmes "MapReduce".
- RéPLICATION des données sur plusieurs noeuds.
- Généralement, pas de gestion de transactions.
 - Mode d'utilisation : peu d'écritures, beaucoup de lectures. Par exemple: SGBD d'annuaires.
- Transactions pas forcément 'ACID' mais plutôt 'BASE'.

Les premiers précurseurs du modèle NoSQL

- Pour répondre aux limites des SGBDR, les entreprises ont commencé à développer leurs propres SGBD pouvant fonctionner sur des architectures matérielles distribuées et permettant de traiter des volumes de données importants.
- Les systèmes propriétaires qui en ont résulté :
 - Google (BigTable)
 - Amazon (Dynamo)
 - LinkedIn (Project Voldemort)
 - Facebook (Cassandra Project puis HBase)
 - SourceForge.net (MongoDB)
 - Ubuntu One (CouchDB)

Comparaison NoSQL et SGBDR (1/3)

Choix des BD NOSQL en opposition aux SGBDR conduits par les contraintes du marché et les besoins techniques :

- **BigData :**
 - Adaptation des BD NOSQL au BigData :
 - Vitesse : beaucoup de données qui arrivent rapidement, à partir de plusieurs sources.
 - Variété : données structurées, semi-structurées ou non-structurées
 - Volume : données très nombreuses (To et Zo).
 - Complexité : données stockées et gérées dans plusieurs data centers.
- **Disponibilité continue des données :**
 - Le manque de disponibilité peut être fatal pour une entreprise.
 - BD NOSQL utilisent une architecture hautement distribuée :
 - Pas de SPOF.
 - Redondance des données et traitements : tolérance aux pannes.
 - La disponibilité continue des BD NoSQL à travers les data centers et le cloud.
 - Toute mise à jour ou modification est faite sans déconnecter la base.

Comparaison NoSQL et SGBDR (2/3)

- **Indépendance de l'emplacement :**

- Possibilité de consulter et modifier une BD sans savoir où est-ce que ces opérations ont réellement lieu.
- Toute fonctionnalité d'écriture propagée à partir d'un emplacement, pour être disponible aux utilisateurs à partir d'autres sites.
- Difficile à appliquer aux SGBDR, surtout pour l'écriture.

- **Modèles de données flexibles :**

- Dans le modèle relationnel, les relations entre les tables sont prédefinies, fixes et organisées dans un schéma strict et uniforme :
 - Problèmes d'évolutivité et de performance en gérant de grands volumes de données.
- Les BD NOSQL peuvent accepter tout type de données (structurées, semi-structurées ou non-structurées) plus facilement.
- Dans les SGBDR, les performances posent problème, surtout quand des lignes "larges" sont utilisées et les actions de modification sont nombreuses.

Comparaison NoSQL et SGBDR (3/3)

- **Business Intelligence et Analyse :**

- Capacité des bases NOSQL à exploiter les données collectées pour dériver des décisions pertinentes.
- Extraction d'informations décisionnelles à partir d'un grand volume de données, difficile à obtenir avec des bases relationnelles.
- Bases NoSQL permettent le stockage et la gestion des données des applications métier, et fournissent une possibilité de comprendre les données complexes, et de prendre des décisions.

SGBDR Vs NoSQL

SGBDR	NoSQL
Consistance forte	Consistance éventuelle
Scalabilité vertical, peu horizontale	Scalabilité horizontale
Grande quantité de données	Enorme quantité de données
Supporte les transactions	Difficile de supporter la transaction
Schéma / tables	pas de schéma
SQL	Map-Reduce
Bonne disponibilité	Très haute disponibilité