Introduction aux SGBD NoSQL

Par A. DAAIF
ENSET Mohammedia
Université Hassan II, Casablanca

Plan

- Limites des bases de données relationnelles
- Avènement du NoSQL
- Théorème de CAP
- Les propriétés BASE
- Différents types de bases de données NoSQL
- Avantages et inconvénients du NoSQL

Bases de données relationnelles

- Apparues aux année 70 du siècle dernier(IBM)
- Leurs concept fondamental est **la relation** qui définit l'association entre deux entités
- Les entités sont stockées dans des tables
- Les associations peuvent exiger des tables supplémentaires
- Utilisent SQL comme langage de manipulation (données et schéma)
- Des indexes sont utilisés pour accélérer l'accès aux données
- Selon les SGBDR, plusieurs mécanismes complémentaires sont utilisés pour faciliter l'accès aux données (caches, vues, procédures stockés, etc)

SGBDR – Propriétés ACID

- ACID représente les quatre attributs d'une transaction de données.
- Une transaction représente un ensemble d'opérations
- A : Atomicity (Atomicité)
- C : Consistency (Cohérence)
- I : Isolation (Isolation)
- D : Durability (Durabilité)

SGBDR – ACID - Atomicity

- Pour qu'une transaction devienne effective, il faut que **toutes les opérations** qui la composent **réussissent**. (COMMIT)

- Si une opération échoue, l'ensemble des opérations est annulé (ROLLBACK)

SGBDR –ACID - Consistency

- La propriété de cohérence assure que chaque transaction amènera le système d'un état valide à un autre état valide. Tout changement dans la base de données doit être valide selon toutes les règles définies, incluant mais non limitées aux contraintes d'intégrité etc.

SGBDR –ACID - Isolation

- Toute transaction doit s'exécuter comme si elle était la seule sur le système. Aucune dépendance possible 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 : si T1 et T2 s'exécutent simultanément, alors chacune doit demeurer indépendante de l'autre.

SGBDR –ACID - Durability

- La propriété de durabilité assure que lorsqu'une transaction a été confirmée, elle demeure enregistrée même à la suite d'une panne d'électricité, d'une panne de l'ordinateur ou d'un autre problème.

- Par exemple, dans une base de données relationnelle, lorsqu'un groupe d'énoncés SQL a été exécuté, les résultats doivent être enregistrés de façon permanente, même dans le cas d'une panne immédiatement après l'exécution des énoncés.

Limites des bases de données relationnelles

- Problème lié à l'application des propriétés ACID en milieu **distribué**
  - La cohérence (consistency) est très difficile à assurer car cela exige que les serveurs doivent être des miroirs les uns des autres. Cela implique que :
  - Toutes les données de la base doivent être présentes sur chacun des serveur => Coût de stockage
  - Les délais des opérations d'insertion, modification et suppression peuvent devenir important car une modification sur un serveur implique la même modification sur tous les serveurs.
- Problème de performance des requêtes utilisant des jointures
- Problème lié à la gestion des objets hétérogènes
- Types de données limités
- Incompatibilité des types (App ⇔ SGBD)

SGBDR ▫ NoSQL

- L’accroissement exponentiel des données, la prise en compte des données faiblement structurées, et les avancées technologiques sont autant d’arguments qui poussent à la migration des SGBD relationnels vers une nouvelle façon de stockage et de manipulation des données; le NoSQL

Bases de données NoSQL

- NoSQL signifie **Not only SQL**
- Regroupe l'ensemble des initiatives de systèmes de bases de données autres que les SGBDR
- Ces initiatives sont conduites par les besoins spécifiques des grands acteurs du web et des réseaux sociaux. (Google, LinkedIn, Facebook, Amazon, etc…)
- Deux défis majeurs :
  - Gérer les grands volumes de données
  - Assurer le service lors des montées en charge (performances d'accès et d'insertion)
- Le NoSQL privilégie la **haute disponibilité** et le **partitionnement horizontal** au détriment de **la cohérence**.

Pourquoi le NoSQL ?

- Taille : Supporter un nombre important d'opérations, de données, d'utilisateurs etc.
- Performances en écriture
- Type de données flexibles (semi-structurées)
- ACID : il existe des solutions pour conserver certains aspects de ces propriétés. Voir théorème de CAP
- Simplicité de développement.
- Scalabilité horizontale
- …

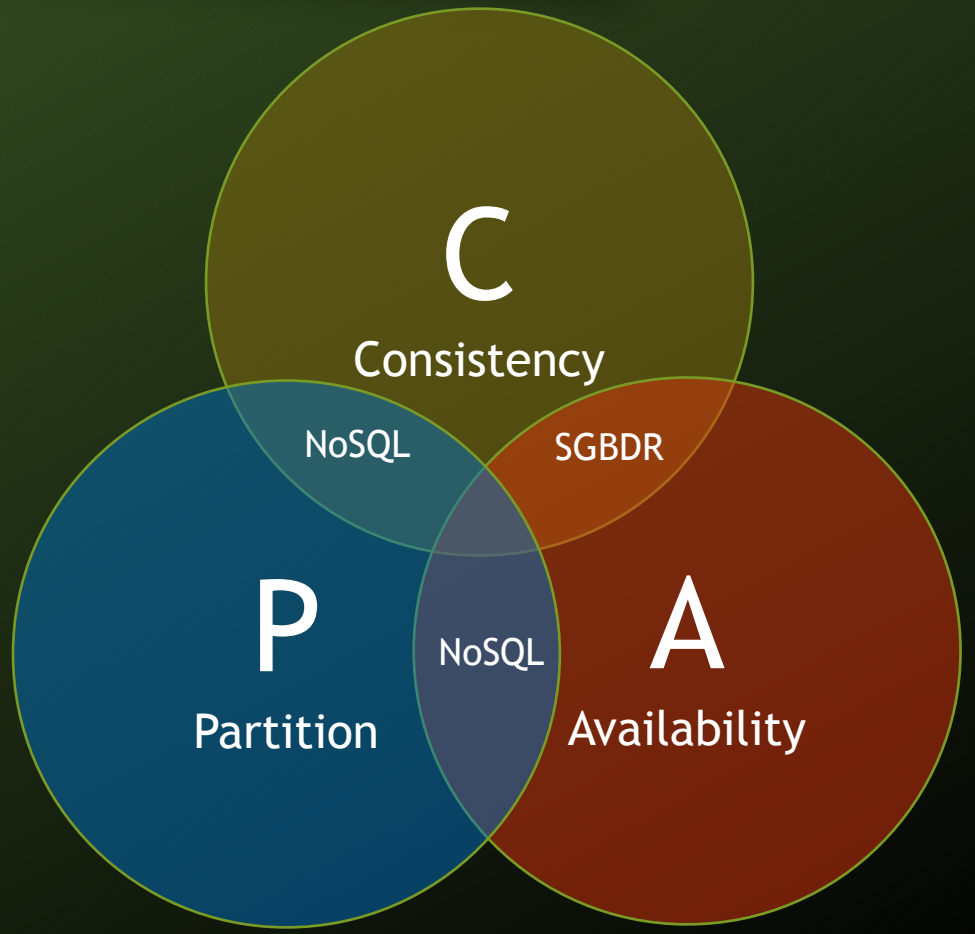
Théorème de CAP

- Ou théorème Brewer (2000)
- **CAP** = Consistency, Availability, Partition Tolerance
- **Cohérence** (Consistency) : tous les nœuds du système voient exactement les mêmes données au même moment
- **Disponibilité** (Availability) : garantie que toutes les requêtes reçoivent une réponse;
- **Tolérance au partitionnement** (Partition Tolerance) : aucune panne moins importante qu'une coupure totale du réseau ne doit pas empêcher le système de répondre correctement

Théorème de CAP

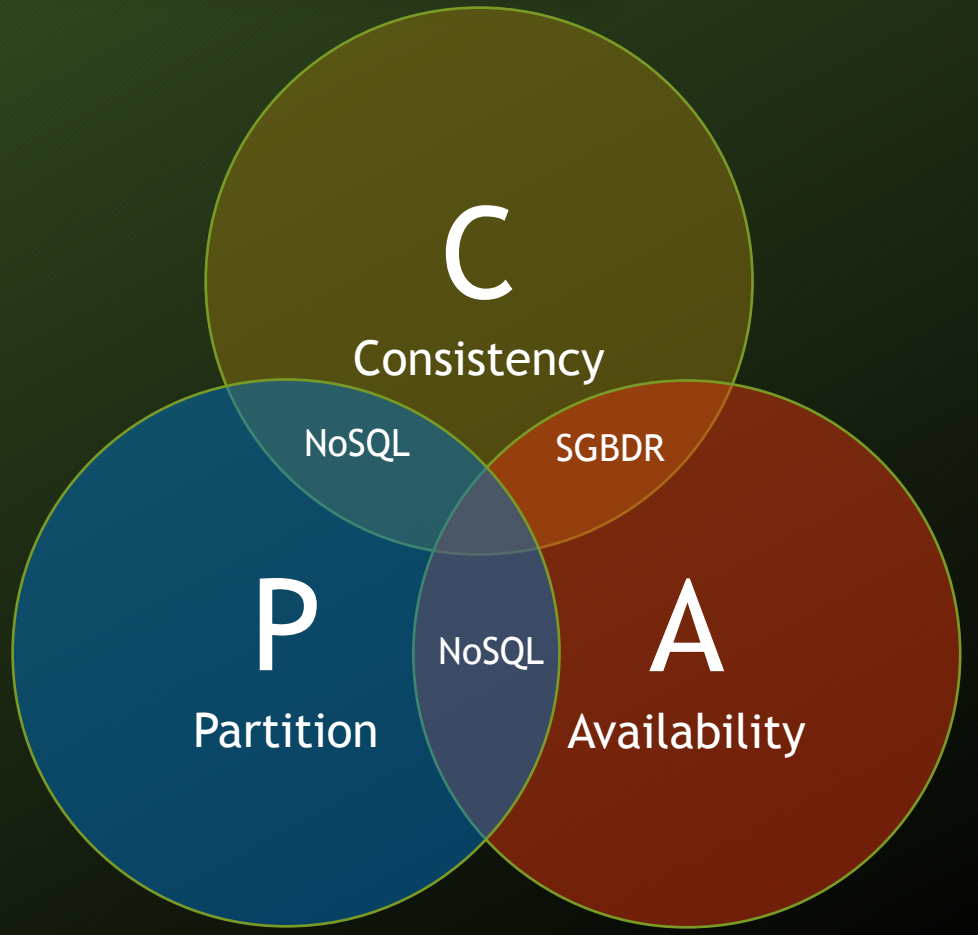
- Dans toute base de données, vous ne pouvez respecter au plus que 2 propriétés parmi la **cohérence**, la **disponibilité** et la **distribution**.

- Cela s'illustre assez facilement avec les bases de données relationnelles, elles gèrent la cohérence et la disponibilité, mais pas la distribution



Théorème de CAP

- **CP** : Les données sont consistantes entre tous les nœuds et le système possède une tolérance aux pannes, mais il peut aussi subir des problèmes de latence ou plus généralement, de disponibilité
- **AP** : Le système répond de façon performante en plus d'être tolérant aux pannes. Cependant rien ne garantit la consistance des données entre les nœuds
- **CA** : Les données sont accessibles et consistantes entre tous les nœuds tant que tous les nœuds sont en ligne



Les propriétés BASE

- On abandonne ACID pour adopter BASE
- BASE c’est :
- **Basic Availability** (Disponibilité basique): Même en cas de panne d’un des nœuds du cluster, le système reste disponible selon le théorème CAP
- **Soft State** (Cohérence légère) : Cela indique que l’état du système risque de changer au cours du temps, sans pour autant que des données soient entrées dans le système. La cohérence devient la responsabilité du développeur et non pas celle du SGBD.
- **Eventual Consistency** (Cohérence à terme) : La seule exigence des bases de données NoSQL en matière de cohérence est d'exiger qu'à l'avenir, les données convergent vers un état cohérent. Aucune garantie n'est toutefois donnée quant au moment où cela se produira. Cela constitue un écart total par rapport à l'exigence de cohérence immédiate d'ACID qui interdit l'exécution d'une transaction jusqu'à la fin de la transaction précédente et la convergence de la base de données vers un état cohérent.

Différents types de bases de données NoSQL

- Les bases de données orientées Clé/Valeur
- Les bases de données Orientées Document
- Les bases de données Orientées Colonnes
- Les bases de données orientées Graphe

Les bases de données orientées Clé/Valeur

- Une base de données de valeurs-clés, ou magasin de clés-valeurs, est un paradigme de stockage de données conçu pour stocker, extraire et gérer des tableaux associatifs, une structure de données plus couramment appelée aujourd'hui dictionnaire ou table de hachage.
- Les dictionnaires contiennent une collection d'objets, ou enregistrements, contenant à leur tour de nombreux champs différents, chacun contenant des données. Ces enregistrements sont stockés et récupérés à l'aide d'une clé qui identifie de manière unique l'enregistrement.
- Exemples : Redis, Riak, Etcd, Hazelcast, memcached …

Les bases de données Orientées Document

- Ces bases de données sont une évolution des bases de données de type clé-valeur. Ici les clés ne sont plus associées à des valeurs sous forme de bloc binaire mais à un document dont le format n'est pas imposé. Il peut être de plusieurs types différents comme par exemple du JSON ou du XML, pour autant que la base de données soit en mesure de manipuler le format choisi afin de permettre des traitements sur les documents. Dans le cas contraire, cela équivaudrait à une base de données clé-valeur. Bien que les documents soient structurés, ce type de base est appelée « schemaless ».
- Exemples :
  - JSON ->MongoDB, CouchBase
  - XML -> eXist, BaseX

Les bases de données Orientées Colonne

- La table à deux dimensions vue par l'utilisateur est représentée par le SGBD comme une suite d'octets pour que le système d'exploitation puisse l'écrire en mémoire ou sur le disque.
  - Une table de donnée **orientée lignes** sérialise toutes les valeurs d'une ligne ensemble, puis les valeurs de la ligne suivante, etc.
  - Une base de données **orientée colonne** sérialise les valeurs d'une colonne ensemble, puis les valeurs de la colonne suivante, etc
- Cette technique permet d'ajouter facilement de nouvelles colonnes. Mais aussi d'améliorer la compression des données puisque les données ont plus de chance de se répéter sur les colonnes.
- Exemples : Cassandra, Hbase,

Les bases de données orientées Graphes

- Une base de données orientée graphe représente les données sous forme de nœuds mis en relation avec des arcs.
- Une telle base de données se caractérise par les critères suivants :
  - stockage des données représentées sous forme d'un graphe, avec des nœuds et des arcs ;
  - lecture et parcours des données sans recours à un index, en utilisant les arcs pour passer d'un nœud à l'autre ;
  - flexibilité du modèle de données ; il n'est pas nécessaire de créer une entité pour les nœuds ou les arêtes contrairement au modèle d'une base de données relationnelle ;
  - intégration d'une API utilisant des algorithmes classiques de la Théorie des graphes (plus-court-chemin, Algorithme de Dijkstra, …) permettant une exploitation différente des bases de données relationnelles.
- Exemples : Neo4J, ArangoDB