HAI914I - Gestion des données au delà de SQL (NoSQL)

Cours: 1
Date: 16/09/2022
Professeur/intervenant: Isabelle Mougenot

Notes de Cours :

- triplestore : gestionnaire de stockage de triplets
- un triplet est construit sut la base d'un sujet, d'une propriété et d'un objet.
- On a des langages qui vont venir ajouter des propriétés aux triplets

Les motivations du NoSQL :

- impulsion autour du NOSQL grâce à Google, Facebook, Twitter....
    - car besoin de données ouvertes
    - plus de volume de données
    - limites des bdd relationnelles
        - flexibilité
        - adaptabilité
        - des milliers voir des millions d'usagers

Le NoSQL est une alternative au relationnel dans certains cas mais pas partout.
Le relationnel est souvent suffisant dans la plupart des cas.
Les cas spécifiques sont :

- recours fréquent à de l'évolution de schémas
- un flux transactionnel
- données distribuées dès l'origine

Le NoSQL ne s'oppose pas au relationnel, il est à l'inverse complémentaire.
L'idée est de s'en démarquer.
Le NoSQL dénormalise les schémas pour rendre le processus plus efficace, si on a besoin de faire évoluer les schémas.
Par contre, si on a pas besoin de les faire évoluer, cela est inutile et le relationnel est plus efficace.

Voir propriété ACID
En NoSQL, on peut ajouter des serveurs au niveau de l'architecture physique :

- diminution du temps de réactivité

Scalabilité horizontale : Établir une relation linéaire entre les ressources ajoutée et l'accroissement des performances.

Pas aussi idéal que ce que l'on pourrait penser

Besoin applicatif à large échelle

- Le partitionnement c'est pouvoir exploiter les performances d'un schéma en partitionnant les données et en les dispatchant sur les noeuds de mon système (qui contiennent l'information dont j'ai besoin).

Cela existait déjà dans le relationnel. La différence est que l'on va pouvoir le faire en horizontal, vertical ou hybride.

- La réplication permet d'avoir des copies de certains noeuds et d'avoir un noeud qui prend le relais d'un autre lors du mal fonctionnement mais aussi de distribuer la charge de travail de chaque noeud pour qu'ils soient moins surchargés.

C'est au programmeur de définit comment il veut partitionner et répliquer.

Les grands principes du NoSQL :

- **Simplicité**
- **Flexibilité**
- **Efficacité**
- **Passage à l'échelle : gros volumes de données distribués et interconnectés**

En BDD on peut utiliser plusieurs systèmes de données. Ils sont complémentaires. On doit choisir le système en fonction de nos besoins.

Principe CAP

1. Consistency : Toute modification de donnée est suivie d'effet pour tous les noeuds du système
    - Dans un système cohérent, tous les noeuds du système rendront la même valeur une fois interrogé
2. Availability : Toute requête émise et traitée par un noeud du système, reçoit une réponse
    - Un système disponible doit pouvoir répondre à n'importe quel moment à un appel
3. Partition tolerance : assurer une continuité du fonctionnement en cas d'ajout/suppression de noeuds du système

- SGBDR : Cohérence et haute disponibilité
- Système NoSQL : Choix de la tolérance et sélection de soit la cohérence, soit la disponibilité.

Typologie du système NoSQL

- Principe de base : clé/valeur -> meilleurs systèmes

    - clé/valeur distribué
    - orienté colonne
    - orienté document
- Système orienté graphe
- Dans une certain mesure les triplestores et les SGBDOO(bdd orientée objet) (Doctrine permet de mapper, c'est à dire d'utiliser une bdd relationnelle mais de faire comme si c'était de l'orienté objet)

Absence de standards

1. APIs Spécifiques
2. Terminologies propriétaires
3. Mécanismes de requêtage à géométrie variable

Définitions :

- **Scalabilité ou passage à l'échelle** : Capacité de l'architecture à s'adapter à une montée en charge sans besoin de refonte des appli