Des BDDR aux bases de données NoSQL

Houcine Senoussi

September 25, 2017

Introduction

- Big Data $\rightarrow$ Développement des bases de données non relationnelles.
- Notre objectif dans ce chapitre est de comprendre les raisons de ce développement et d'étudier les propriétés et le fonctionnement des ces nouvelles bases de données.

Généralités

- Modèle simple et puissant.
  - Données sous forme de tables (relations) à deux dimensions.
- Dispose d'un langage de requête normalisé (ANSI 1986) : SQL.
  - Langage intuitif (4^{eme} génération).
  - Langages de définition (DDL) et de manipulation (DML) des données.
  - Possibilité d'agréger, de résumer et de rapprocher les données.
  - Gestion des droits avec la même structure et la même syntaxe.
  - Extensions procédurales (propriétaires) : PL/SQL d'Oracle et T-SQL de Microsoft.
  - Intégration possible dans les langages de programmation (jdbc, odbc, ...).

Normalisation des bases de données

- Première forme normale :
  - Une relation est en première forme normale si tous ses attibuts sont **atomiques** (mono-valués).
- Deuxième forme normale :
  - 1-NF + tout attribut qui n'est pas dans la clé primaire est en **dépendance fonctionnelle** par rapport à la clé primaire.
- Troisième forme normale :
  - 2-NF + Pas de dépendance fonctionnelle entre les attributs non-clés.

Qu'est-ce qu'une transaction

- Origine du concept : juridique.
- Transformation de l'état du système.
  - Requêtes **insert**, **update** et **delete**.
- S'exécute "virtuellement" jusqu'à la validation (**commit**) ou l'annulation (**rollback**).

Exemple de transaction

- Gestion commerciale : tables Clients, Produits et Commandes.
- Passer une commande :
  - Enregister la commande (insert/table : Commandes).
  - Augmenter l'en-cours client la commande (update/table : Clients).
  - Décrémenter les quantités de stock (update/table : Produits).

Propriétés d'une transaction

- Caractéristiques résumées par l'acronyme **ACID** :
  - **Atomicité** : Fonctionnement en Tout-Ou-Rien. Soit toutes requêtes sont exécutées, soit aucune ne l'est.
  - **Cohérence** : fait passer le système d'un état valide à un autre valide.
    - Exemples : Si un client est supprimé toutes ses commandes le sont aussi. Si une nouvelle commande est créée l'en-cours du client corresponsant est mis à jour.
  - **Isolation** : Deux transactions concurrentes ne s'enchevêtrent pas. Si deux transactions essaient de modifier la même donnée, l'une des deux sera bloquée jusqu'à ce l'autre valide ou annule sa modification.
  - **Durabilité** : Une fois validés, les changements ne peuvent plus être annulés.

Travail pratique sur les transactions

- Faire le TP 'BDDR:les transactions'.
- Analyser les limites du modèle relationnel mises en évidence par la manipulation des transactions.

Limites du modèle relationnel

- Ce sont les transactions qui assurent la **cohérence** des bases de données.
- Ces règles (celles des transactions) commencent à poser problème lorsque les données deviennent volumineuses.
  - Problème de disponibilité (**Availability**) des données.

Limites du modèle relationnel-2

- Le schéma lui même commence à poser problème :
  - Les données ne correspondent pas toujours aux objets du business.
    - Exemple : En plus de la table "Etudiants" et "Matières" nous devons avoir une table "tels étudiants suivant tels cours".
    - C'est une conséquence de la **normalisation** des données.
    - Nous devons faire des **jointures** pour relier ces tables : coûteuses en temps dès que les données deviennent **volumineuses**.

Limites du modèle relationnel-3

- Relier les traitements (programmes) et les données : passage relationnel - objet (mapping objets des applications - données de la base) pas toujours 'naturel'. Le problème se pose notamment pour les grandes masses de données (besoins importants en mémoire).
- Certains types de données ne sont pas adaptées au format relationnel : systèmes de traitement d'événements complexes-changements d'états fréquents (vélocité).

Limites du modèle relationnel-4

- En résumé :
  - Problèmes de **performance** pour les données **volumineuses** (Web scale).
- Que faire ?
  - Affiner les programmes ?
  - Augmenter la puissance des machines ?
  - Dénormaliser les données ?
  - Répliquer/Distribuer les données ?

Limites du modèle relationnel-5

- Affiner les programmes :
  - Utilisation des indexes.
  - Optimisation des requêtes.
  - Autres astuces au niveau du code.
  - Problème : effet limité.
- Vertical scaling : utiliser du matériel plus puissant (plus de mémoire, processeurs plus rapides et disques de plus grande capacité).
  - Problème 1 : les données augmentent plus vite que la loi de Moore.
  - Problème 2 : la bande passante des disques est limitée.

Limites du modèle relationnel-6

- Dénormaliser les données :
  - Autoriser la redondance des données.
  - Concevoir les données en fonction des traitements et non indépendamment d'eux.
  - Avantages : accélérer les traitements (réduire le nombre de jointures), avoir une image définitive des données (exemple des factures).
  - Inconvénients : on sort du cadre du relationnel, risque d'incohérence des données.

Limites du modèle relationnel-7

- Répliquer les données : avoir plusieurs exemplaires de la même donnée mais sur plusieurs machines.
  - Avantage : accès aux données plus facile.
  - Inconvénients : risque d'incohérence des données, gestion plus compliquée (relation maître/esclave entre les machines).

Limites du modèle relationnel-8

- Distribuer les données. Cela peut se faire selon plusieurs principes :
  - segmentation fonctionnelle : par exemple la table *Clients* sur une machine et la table *Produits* sur une autre.
  - distribution sur la base d'une clé : à l'intérieur de chaque table, calculer une clé pour chaque ligne et répartir les lignes en fonction de la valeur de la clé.
  - utilisation d'un annuaire (lookup) : l'un des serveurs sert de 'pages jaunes', c'est-à-dire qu'il contient l'information concernant la machine contenant chaque ligne de chaque table.

Vers les bases de données NoSQL

- La discussion précédente nous conduit à la (triple) conclusion suivante :
  - Tant que le volume des données reste 'raisonnable' on peut conserver les bases de données relationnelles.
  - Lorsque les données deviennent trop volumineuses on est amené à renoncer à certains aspects du relationnel (schéma, normalisation, ...)
    - De là sont nées les bases de données non relationnelles.
  - Il n'est pas toujours possible de respecter toutes les exigences du business : notamment la cohérence permanente des données et le temps d'attente limité lors de l'accès à ces dernières.

Bases de données NoSQL

- L'essentiel de ces bases ont été inventées par les équipes de grands acteurs du web (Google, Amazon, Facebook, ...).
- Objectif : répondre aux besoins de ces entreprises confrontées à de très grands volumes de données que le modèle relationnel de permettait plus de traiter.
- Plusieurs formes : les principales sont les BDD orientées document, BDD orientées colonne, BDD graphe.

Théorème de Brewer

- Comme nous l'avons remarqué plus haut il n'est pas possible d'avoir un système de gestion des données répondant à toutes les exigences du business. D'où la nécessité d'un compromis : alléger certaines exigences pour en obtenir d'autres.
- Cette impossibilité/compromis a été formulée par Brewer dans un théorème :
  - Théorème CAP : dans un système de gestion de données (volumineuses) on ne peut garantir que deux des trois exigences suivantes : la cohérence des données (Consistency), la disponibilité des données (Availability) et la tolérance aux pannes (Partition tolerance).

Conclusion

- Bases de données relationnelles : caractéristiques et limites.
- Chapitre suivant :
  - Bases de données NoSQL : principes, familles et fonctionnement.
  - Théorème de Brewer.

Bibliographie

- Rudi Bruchez."Les bases de données NoSQL et le Big Data". Eyrolles 2015.
- Eben Hewitt."Cassandra : the definitive guide". O'Reilly, 2011.