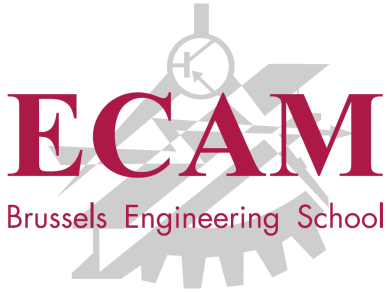

I404C NoSQL

Séance 9

Mise en œuvre d'une base de données NoSQL

Sébastien Combéfis

2017–2018

Rappels

- Index et recherche dans des données avec ElasticSearch
  - Démarrage et ajout de données dans l'index
  - Recherche de données et requêtes sur l'index
- Opération et calculs sur des données avec Map-Reduce
  - Définition des opérations Map et Reduce
  - Partitionnement et combinaison de données
  - Exemple de calcul et plateforme Pig

Objectifs

- Migration de schéma de données
  - Bases de données sans schéma
  - Modification de schéma en relationnel et en NoSQL
- Persistence polyglotte
  - Caractéristiques des différents modèles et comparaison
  - Combinaison de plusieurs types de moteurs différents

Migration de données

Bases de données sans schéma

- Les bases de données NoSQL sont sans schéma

  Ajout libre de valeurs, documents, colonnes, nœuds et arêtes...

- Offre une grande liberté et meilleure souplesse
  - Pas besoin de penser à l'avance aux données à stocker
  - On peut arrêter de stocker des éléments sans perdre le passé

- Stockage de données non uniformes

  Les enregistrements peuvent avoir des champs différents

Schéma implicite

- Parfois nécessité de connaitre les champs disponibles

  Sauf si on génère juste un rapport sous forme « champs : valeur »

- Existence d'un schéma implicite de données

  Hypothèses sur la structure dans le code qui manipule les données

- Définition du schéma dans le code de l'application
  - La base de données ignore l'existence de ce schéma
  - Difficultés lorsque plusieurs applications accèdent la même base

Vue matérialisée (1)

- Structure en agrégats facilite la récupération d'information

  Stockage et accès aux agrégats comme des unités

- Une vue en relationnel est une relation calculée sur d'autres
  - Cache au client si les données sont dérivées ou non
  - Peut être très lourd à calculer

- Vue matérialisé calculée à l'avance et cachée sur disque

  Pratique pour données souvent lues sans devoir être vite à jour

Vue matérialisée (2)

- Requêtes précalculées et cachées dans le monde NoSQL

  Appelées vues matérialisées et gérées par Map-Reduce

- Deux principales stratégies de construction
  - Mise à jour de la vue en même temps que les données
  - Mise à jour à intervalles réguliers
- Possible de construire la vue matérialisée à l'extérieur
  - Appel d'une fonction de construction et écriture vers la DB
  - Utilisé avec le Map-Reduce incrémental

Changement de schéma

- Changement de schéma fréquent en développement Agile

  Important de pouvoir s'adapter aux changements d'exigences

- Discussions avec experts du domaine lors de chaque itération

  Diminution des frictions avec les développeurs

- Bases de données NoSQL très utilisées en prototypage

  Éventuelle migration (partielle) vers le relationnel en production

Migration en relationnel (1)

- Développement d'objets, tables et relations

  Corrélation à avoir entre les modèles objet et entité-relation

- Modification des objets doit être répercutée sur les données
  - Maintenir synchronisation entre application et DB
  - Migration est un projet en soi (scripts de migration)
  - Pas adapté à l'agile et error-prone

Migration en relationnel (2)

- Pour les Greenfield projects (nouvel environnement)
  - Scripter les changements de DB pendant le développement
  - Maintenir l'historique des modifications (versioning)
  - Support d'outils : Liquibase, MyBatis Migrator, DBMaintain
- Pour les Legacy projects (intégration avec déjà existant)
  - Reverse engineering sur bases existantes comme baseline
  - Pas de données transactionnelles dans la baseline
  - Transition avec compatibilité avec les applications existantes
  - Scaffolding (trigger, vue et colonne virtuelle)

Migration en NoSQL

- Changement fréquents du schéma des DBs NoSQL

  S'adapter aux changements du marché et innovation des produits

- Pas besoin de penser au schéma avant le développement

  Type relation pour graphe, nom colonnes, organisation clés...

- Pas de schéma rigide dans le moteur de la DB

  Mais schéma implicite au niveau de l'application (ou mixte)

- Difficultés lors de changements alors que données existantes

Migration incrémentale

- Risque d'avoir des incompatibilités de schéma

  Par exemple lors du renommage d'un champ dans les documents

- Deux options pour migrer de schéma
  - Convertir toutes les données vers le nouveau schéma (couteux)
  - Lecture dans les deux schémas et écriture vers nouveau

Autre migration

- Migration des bases de données orientée graphe

  Ajout d'arêtes pour nouvelle relation, plutôt que mise à jour

- Modification de la structure d'agrégats

  Couper un agrégat en plus petits morceaux


welcome
Persistence polyglotte

Persistence polyglotte

- Bases de données conçues pour résoudre différents problèmes

  Utiliser un seul moteur pour toutes les contraintes pas performant

- Plusieurs types de problèmes différents
  - Stocker des données transactionnelles
  - Cacher des informations de session
  - Parcourir des clients avec les produits achetés par leurs amis

- Différences entre systèmes OLTP et OLAP

  Basés sur les transactions ou sur l'analyse statistique

Besoins en stockage

- Propriétés nécessaires à satisfaire pour des données stockées

  Disponibilité, consistence, nécessite de backups...

- Programmation polyglotte, Neal Ford en 2006

  Utiliser plusieurs langages de programmation pour une application

- Persistence polyglotte dans le domaine des bases de données

  Approche hybride de la persistence, combinant plusieurs moteurs

Plateforme de e-commerce (1)

- Plusieurs types de données dans une application e-commerce
  - Données du panier d'achats
  - Informations sur la session de l'utilisateur
  - Historique des commandées finalisées
  - Business Intelligence/Data Warehousing (stratégie, reporting...)
- Classiquement géré par un seul moteur RDBMS

  Mais pas mêmes besoins en backup, recovery, disponibilité...

Plateforme de e-commerce (2)

- Base de données orientée clé-valeur
  - Données transitoires identifiables par une clé
  - Panier d'achats avant confirmation de l'achat et session
- Base de données orientée graphe

  Réseau social des clients pour moteur de recommandation
- Base de données orientée documents

  Informations de logs et historique des commandes
- Base de données relationnelles pour inventaire et prix

Couche de service

- Interrogation des données stockées par une autre application

  Lourd de devoir connaitre la structure et interroger les moteurs

- Placement d'une/plusieurs bases de données derrière une API

  Définition d'un service exposé à l'extérieur

- Évolution de la structure des bases de données transparente

  Stabilité de l'API au vu de l'extérieur

Extension

- Changer une base de données pour un nouvel usage spécifique

  Difficile pour assurer service auprès des applications

- Possible d'ajouter des fonctionnalités
  - Système de cache avec memcached, par exemple
  - Moteur d'indexation avec Solr, par exemple

- Prévoir mécanisme de synchronisation des données

  Systématique, régulière, à la demande, par opération...

Choisir la bonne technologie

- Modèle de données à stocker et requêteq à réaliser

  Initialement tout dans une seule base de données relationnelle

- Toutes les solutions sont viables de prime abord

  Bon choix de persistence et attributs pour répondre aux requêtes

- Bon choix de technologie par rapport à l'évolution

  Changement ou nouvelle requête doit être possible facilement

En entreprise

- Les DBAs doivent comprendre les nouveaux types de stockage

  Nécessité de devenir poly-skilled

- Plusieurs compétences à acquérir
  - Fonctionnement des moteurs NoSQL
  - Comment effectuer un monitoring de ces moteurs
  - Stratégie de backup et de recovery des données
  - Extraction de données depuis ces moteurs

- Choix d'un moteur, souvent open source et outils ad-hoc

  Aspects de sécurité déplacés du moteur NoSQL vers l'application

Déploiement

- Le déploiement des bases de données devient complexe

  Tous les moteurs doivent tourner en même temps

- Installation souvent faite par clone du dépôt officiel

  Après déploiement, nécessité de suivre les mises à jour

- Configuration facilitée avec choix minimaux par défaut

  Lien à faire avec la communauté open source des projets

Choix d'une base de données

- Pas de règle établie pour choisir une base de données

  Dépend fortement du domaine et des circonstances individuelles

- Le monde NoSQL est encore jeune et immature

  On en reparle dans quelques années...

- Deux principaux critères à considérer

  Productivité du programmeur et performance d'accès

Productivité du programmeur

- Grosse frustation de devoir travailler avec le relationnel
  - Récupération et affichage des informations en agrégats
  - Transformation pour stockage sous forme de relations
  - Utilisation d'ORM : Hibernate, iBATIS, Rails Active Record...
- NoSQL supprime ORM et stocke naturellement des agrégats

  Et graphes adaptés pour un grand nombre de relations différentes
- NoSQL très adapté pour données non uniformes

  Pas de schéma (fort) ou possibilité schéma mixte

Performance d'accès

- Besoin d'accéder rapidement à beaucoup de données

  Gros sites web désireux de croissance horizontale sur un cluster

- Lecture d'un agrégat faite pour être très rapide

  Contrairement au relationnel avec plusieurs tables et jointure

- Sharding et réplication horizontale sur un cluster facilités

  Permet une mise à l'échelle d'une application

Pourquoi NoSQL en entreprise ? (1)

- Six besoins principaux pour l'adoption de NoSQL
  - Besoin de **vitesse**

    Diminuer le temps de réponse des demandes extérieures
  - Besoin de **mise à l'échelle**

    Supporter nombre grandissant d'utilisateurs/volumes de stockage
  - Besoin de **disponibilité continue**

    Limiter au maximum les downtime des nœuds

Pourquoi NoSQL en entreprise ? (2)

- Six besoins principaux pour l'adoption de NoSQL
  - Besoin d'**être indépendant du lieu**

    Servir rapidement des données selon le lieu
  - Besoin de **gérer plusieurs types de données**

    Modèle de données le plus flexible possible
  - Besoin de **réduire les couts**

    Prix réduit de solutions open source

Pourquoi pas de NoSQL en entreprise ?

- Plusieurs freins à l'adoption du NoSQL en entreprise
  - Différence du modèle de données par rapport au relationnel
  - Problèmes potentiels de sécurité
  - Non support des transactions ACID
- Existence d'inhibiteurs non techniques
  - Transition du personnel vers le monde NoSQL
  - *« Guerres de religion »* techniques
  - Viabilité des vendeurs des solutions

Exemples d'application

- Applications en ligne
  - Flux de données temporisées (financière...), récolte de senseurs
  - Vente en ligne
  - Analyse de données en temps réel (Google Analytics...)
- Applications analytiques
  - Détection de fraude, analyse de risque
  - Analyse du comportement d'acheteurs
- Applications de recherche d'entreprise
  - Recherche générale ou catégorisée
  - Recherche à l'intérieur de documents (PDF...)

Crédits

- https://www.flickr.com/photos/heimatiater/15509652894
- https://www.flickr.com/photos/prayitnophotography/8014375021