le cnam

Bases Relationnelles vs Bases NoSQL

Exercices

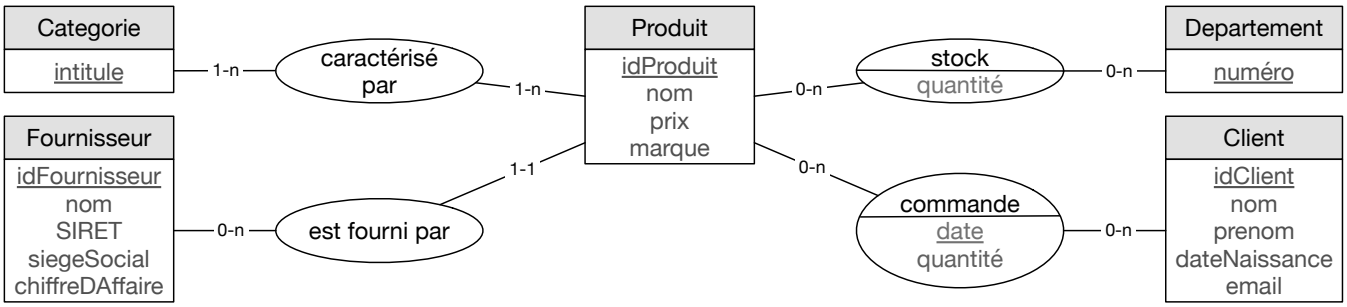
CNAM Paris
nicolas.travers (at) cnam.fr

Chapitre 1

Schéma relationnel vs NoSQL

Ce chapitre est construit de manière à faire comprendre les problématiques liées au passage d'une base de données relationnelle vers une base NoSQL. Etant donné un contexte fortement distribué, nous serons confrontés aux problèmes de jointures traditionnelles du relationnel.

Dans ce chapitre, nous utiliserons une base gérant une vente de produits à des clients sur internet. Le schéma Entité/Association est le suivant :



Par ailleurs, nous avons quelques statistiques associées qui nous servirons de base pour les calculs :

— 10^7 clients, 10^4 produits, 10^9 produits commandés, 100 départements ;

— Un client fait en moyenne 33 commandes contenant chacune 3,3 produits différents ;

— 1 à 5 catégories par produit (en moyenne 2) ;

— 50 produits de marque "Apple", répartis sur tous les départements.

1.1 Relationnel vers Documents JSon

JSon (JavaScript Object Notation) est un modèle de représentation de données semi-structurées très utilisé sur le Web. Il est notamment utilisé dans les systèmes NoSQL orientés documents.

Dans cette section, nous allons étudier la transformation d'un schéma relationnel vers des documents JSon. Nous prendrons pour exemple la table `Produit`.

1.1.1 Donner pour l'entité `Produit` la correspondance directe en JSon sous forme d'exemples de documents (nous n'avons pas de schéma JSon) ;

1.1.2 Le prix doit être détaillé avec une devise et un taux de TVA ;

1.1.3 Les catégories sont indépendantes les unes des autres. Fusionner les deux entités dans le document JSon ;

1.1.4 Le fournisseur d'un produit est fréquemment interrogé avec le produit.

1.2 Dénormalisation

Nous souhaitons éviter de faire de trop nombreuses communications réseaux pour des requêtes soumises à notre base de données. Pour ce faire, proposez un nouveau schéma Entité/Association en appliquant les fusions nécessaires en fonction des informations fournies ci-dessous. Vous y associerez un document JSon exemple.

1.2.1 Au vu des informations données dans la section 1.1, donner le nouveau schéma Entité/Association. Nous noterons en plus que les numéros de département sont indépendant les uns des autres ;

1.2.2 Les produits sont fréquemment interrogés pour y consulter le stock ;

1.2.3 Le stock d'un département est fréquemment interrogé, associé aux produits ;

1.2.4 Les commandes sont fréquemment interrogées pour y retrouver le produit et le client ;

1.2.5 Quels sont les problèmes liés à cette dénormalisation ?

1.3 Jointures

Les systèmes NoSQL ne favorisent pas les jointures du fait de nombreuses communications réseaux générées pour regrouper les données corrélées. Nous allons comparer différentes solutions pour en estimer le coût.

Pour cela nous utiliserons les 4 requêtes de jointure suivantes :

R1 Stock (liste des noms et prix de produits, ainsi que leur quantité) du département n°92 ;
R2 Distribution des produits (nom et quantité) de marque "Apple" dans les départements ;
R3 Nom du produit le plus commandé par le client n°125 ;
R4 Les 100 noms de produits les plus commandés (somme des quantités).

Le coût de chaque requête prendra en compte le nombre de messages échangés sur le réseau. Il se décompose en :

— Messages "**Aller**", interrogeant des données spécifiques ou l'ensemble des serveurs. Cette partie peut être considérée également comme le "*Map*"
— Messages "**Retour**" donnant le nombre de documents retournés
— "**Shuffle**" pour le regroupement des données en préparation d'un "*Reduce*". Les données sont redistribuées sur le réseau selon une clé de regroupement.
— Messages "**Reduce**" correspond au nombre de messages retournés par l'opération *Reduce*. Il est souvent égal au nombre de clé de regroupement.

s et **retour**s.

Les collections sont réparties sur 1000 serveurs. Les choix de répartitions dans les serveurs seront précisés pour chaque requête.

1.3.1 Jointure applicative

Calculer le nombre de messages échangés sur le réseau avec l'algorithme suivant :

1) On interroge la première collection (`Aller` : nb de serveurs interrogés, `Retour` : nb de documents qui répondent)
2) On interroge la seconde collection avec le résultat de la première (`Aller` : nb de documents retournés précédemment, `Retour` : nb de documents qui répondent)

Si une agrégation est nécessaire, il sera effectué par un regroupement par clé (i.e. Map/Reduce). Le nombre de messages sera réparti en "regroupement" et retour de groupement (après agrégation).

Les choix de stockage pour chaque requête est le suivant :

R1 La collection "stock" est placée sur le réseau grâce au département (hachage sur la clé "département"), et les "produits" selon leur identifiant de produit ;
R2 Stockage identique au précédent ;
R3 Les commandes sont placées grâce à l'identifiant du client (hachage). Le comptage groupé par `idproduit` sera effectué de manière distribuée (i.e. MapReduce), il y a 20 idproduits différents.
R4.1 Même stockage, on groupe par `idclient`.
R4.2 Stockage par `idproduit`, le groupement se fera alors localement (pas de messages pour regroupement).

1.3.2 Jointure par Map/Reduce

Une autre possibilité est de réunir les deux types de documents dans une seule et même collection. Les deux "collections" co-existent donc et peuvent répondre aux requêtes.

Pour effectuer la jointure, il faut dans le `MAP` émettre les documents avec la clé de jointure et les données demandées. Dans le `REDUCE`, les données sont jointes pour produire le résultat demandé. Pour améliorer le coût de la jointure, les données sont regroupées grâce à la clé de la jointure.

A chaque requête, tous les serveurs sont interrogés, le reduce est effectué localement grâce à la clé de regroupement sur la jointure. Le résultat est envoyé (s'il existe).

Calculer pour chaque requête le coût en nombre de messages :

R1 Produit et Stock, regroupés sur le numéro de département
R2 Produit et Stock, regroupés sur le numéro de département
R3 Produit et Commande, regroupés sur l'identifiant de produit
R4 Produit et Commande, regroupés sur l'identifiant de produit

1.3.3 Dénormalisation

En utilisant des collections contenant des documents normalisés (ceux obtenus dans la section 1.2), nous pouvons éviter d'avoir à effectuer des jointures.

Pour chaque requête, calculer le nombre de messages générés sur chacune des collections de documents générées, avec les répartitions suivantes :

R1 `Produit-Stock` : regroupement sur "idproduit"

R1 `Stock-Produit` : regroupement sur "département"

R2 `Produit-Stock` : regroupement sur la "marque"

R2 `Stock-Produit` : regroupement sur la "marque"

R3 `Produit-Commande` : regroupement sur "id"

R3 `Commande-Produit` : regroupement sur "idclient". Le comptage groupé par `idproduit` sera effectuée de manière distribuée (i.e. MapReduce).

R4 `Produit-Commande` : regroupement sur "id"

R4 `Commande-Produit` : regroupement sur "idproduit".

1.3.4 Conclusions

Créer un tableau récapitulant pour chaque type de jointure les coûts en nombre de messages échangés. Que peut-on en conclure ?