Dimitri Deramond
Antoine ROBIN
BUT 31 EMS

22/12/2024

SAE Bases de données NoSQL

Table des matière

Introduction

Suite aux limites que la base de données d'une entreprise de voiture rencontre, on nous missione de rendre ces données plus exploitables et simples d'utilisation. Notre objectif va donc être de réaliser la migration d'une base de données relationnelle vers une base de données NoSQL.
Pour cela nous aurons d'abord besoin de nous familiariser et de comprendre les données, afin de choisir le type de base de données NoSQL le plus adapté. Une fois que nous aurons choisi le format des données, nous réaliserons la migration des données vers le type choisi.

Choix du type de base de données NoSQL

La première étape, après avoir analysé les données, est donc de choisir le type de base de données NoSQL, parmi les 4 suivantes: clé-valeur, colonne, document et graphique.
Chacun de ces 4 types de base NoSQL présente ses avantages et ses inconvénient:

Clé-valeur

Dans ce modèle, les données présentes des clés uniques qui agissent comme un identifiant, et pour chacune de ces clés, une valeur est attribuée. Cette valeur peut-être de tout type, chaîne de caractères, JSON, numérique…Plusieurs logiciels permettent de manipuler des données organiser en clé-valeurs, nous pouvons par exemple utiliser redis, memcached ou encore riak.
Les avantages de ce type de base de données NoSQL sont par exemple la facilité avec laquelle on peut lire et écrire les données, car elles sont organisées en ligne et chaque

donnée est rapidement identifiable grâce à la clé, ce qui permet aussi de traiter de grands ensembles de données. En revanche, ce modèle est peu adapté si l'on veut réaliser des requêtes complexes.

Colonne

Le modèle en colonne présente les données sous forme de colonne plutôt que par des lignes, ainsi chaque colonne regroupe les données similaires, par paires de nom-colonne:valeur, ce qui peut rendre la lecture de données massives plus facile. Parmi les logiciels permettant la manipulation des données sur un modèle de type colonne, on retrouve cassandra, druid ou encore scylla par exemple. En revanche, ce modèle n'est pas adapté si par la suite on souhaite gérer les relations entre colonnes, ou si on veut réaliser des transactions complexes.

Document

Les base de données NoSQL document, stocke les données sous forme de documents au format JSON, BSON ou xml, où chaque document représente un enregistrement. La manipulation de données de type document est possible via différents logiciels comme mongoDB, RavenDB ou encore couchbase par exemple. Les avantages de ce type de base de données, est que chaque document peut avoir une structure différente, de plus les données sont bien organisées, chaque document présente toutes les informations de chaque données. Pour les inconvénients, ce modèle rend les requêtes compliquées à réaliser si des jointures entre documents sont nécessaires, de plus les mise à jour simultanées des données peut-être compliquées à gérer.

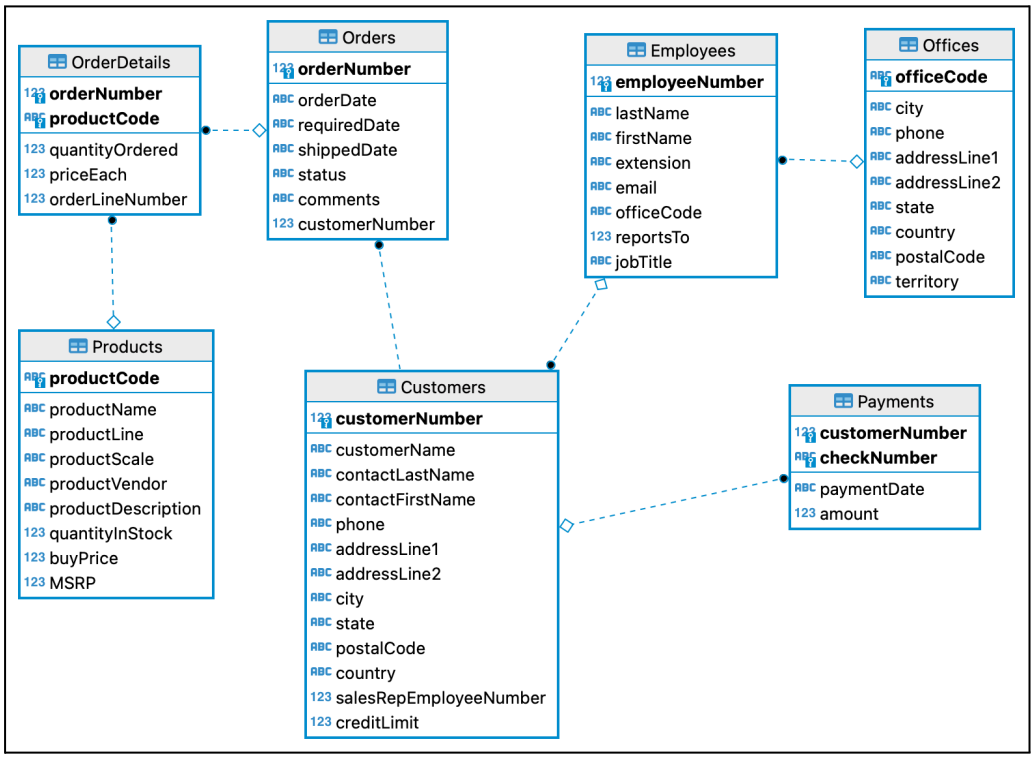
Graphique

Les base de données graphes organisent les données sous forme de nœuds représentant des entités, de liens correspondant au relation entre ces nœuds, et de propriété permettant de compléter l'information, ce type de données nous permet de modéliser des relations complexes entre les entités. La manipulation de ces graphes est possible sur neo4j, arangoDB ou encore OrientDB par exemple.
Les avantages de ce modèle est qu'il permet de réaliser des requêtes entre les différentes relations plus facilement, et permet de modéliser les données de façon intelligible. En revanche, il n'est pas adapté si nous souhaitons réaliser des requêtes n'impliquant pas les relations, et ce modèle n'est pas non plus facilement utilisable par des utilisateurs non familiers avec ce type de données.

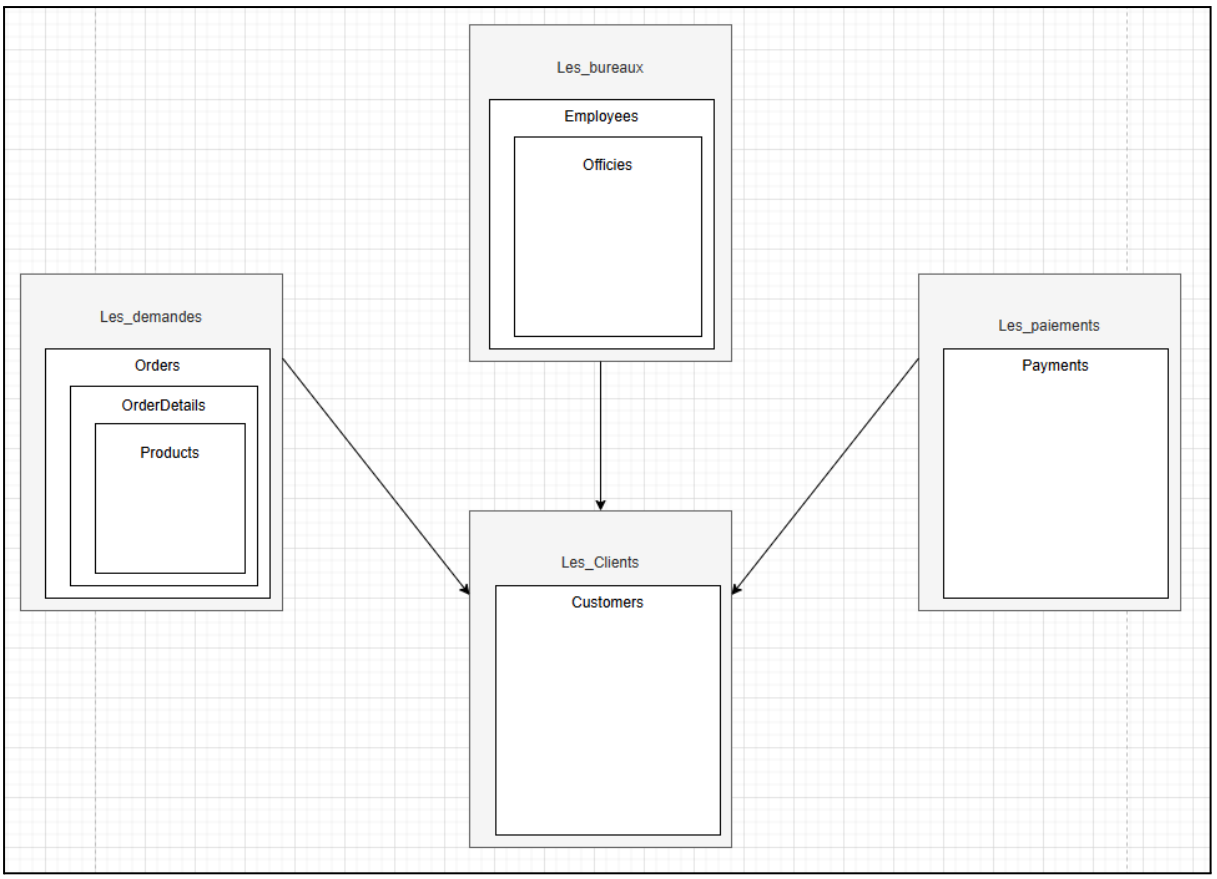
Etant donnée que nous travaillons sur les données d'une entreprise de voiture, on peut s'attendre à un volume important de données, où chaque observations ne disposeras peut-être pas de données pour chaque variable de la base, ainsi nous optons pour une base de données NoSQL de type document, qui nous permettra de stocker chaque données avec les informations dont elle dispose, indépendamment des autres.

Migration des données

Du fait des limites rencontrées par la base de données qui nous a été transmise de la part d'une entreprise de voiture, et afin d'optimiser la gestion des données, notre objectif est de réaliser la migration des données actuelles vers une base de données NoSQL de type document.
Pour cela, nous avons effectué des modifications sur la base de données afin de la préparer à la migration vers le nouveau format. Nous sommes ainsi passé du modèle initiale suivant :



à un modèle simplifié comme ci-dessous:



Une fois la transformation faite, et afin de réaliser la migration vers le nouveau modèle NoSQL de type document, nous avons modifié les données pour les avoir au bon format. Ainsi les tables “Les_demandes”, “Les_bureaux” et “Les_paiements” du nouveau modèle simplifié ont été ajoutées directement dans la table “Les_Clients” sous forme de liste. Nous avons ensuite importé nos données sur MongoDB qui est le logiciel permettant la manipulation des bases de données NoSQL de type document.

Pseudo code :

Début

Fonction ConvertirModèleRelationnelEnNoSQL(baseRelationnelle)

// Traitement de la collection "Les_demandes" pour regrouper Orders, OrderDetails et Products

Pour chaque commande dans baseRelationnelle.Orders

Créer un documentCommande vide

Remplir documentCommande avec les informations de la commande

// Récupérer et ajouter les détails de commande imbriqués

documentCommande.OrderDetails = []

```
        Pour chaque détail dans RechercherDétailsCommande(commande.orderNumber)
            Créer documentDétail et ajouter les informations du détail
            Ajouter les informations du produit associé à documentDétail
            Ajouter documentDétail à documentCommande.OrderDetails
        Fin Pour

        // Insérer le document final dans "Les_demandes"
        InsérerDocumentDansCollection("Les_demandes", documentCommande)
        Fin Pour

        // Traitement de la collection "Les_bureaux" pour regrouper Employees et Offices
        Pour chaque employé dans baseRelationnelle.Employees
        Créer documentEmployé vide
        Remplir documentEmployé avec les informations de l'employé
        Ajouter les informations du bureau associé à documentEmployé
        Insérer documentEmployé dans la collection "Les_bureaux"
        Fin Pour

        // Création des documents indépendants pour "Payments" et "Customers"
        Pour chaque paiement dans baseRelationnelle.Payments
        Créer et remplir documentPaiement avec les informations du paiement
        Insérer documentPaiement dans "Payments"
        Fin Pour

        Pour chaque client dans baseRelationnelle.Customers
        Créer et remplir documentClient avec les informations du client
        Insérer documentClient dans "Customers"
        Fin Pour

        Fin Fonction
Fin
```