24/11/24

Rapport de migration : Base relationnelle vers NoSQL

ANGELIKIA KAVUANSIKO
EKTA MISTRY

BUT 3 SD FA VCOD - NOSQL

UNIVERSITE PARIS CITÉ

Sommaire

Introduction

CONTEXTE

Dans le cadre de la modernisation de son infrastructure de données, Paula Dupont, directrice d'une entreprise automobile, souhaite migrer son système actuel basé sur une base de données relationnelle vers une base de données NoSQL. En effet, la base de données relationnelle actuelle présente plusieurs limitations majeures.

D'une part, les requêtes sont de plus en plus lentes, entraînant une latence élevée lors de la consultation des données. D'autre part, le système subit régulièrement des pertes de données dues à des défaillances de serveurs. Ces problèmes impactent directement la fiabilité et la performance globale du système.

Pour répondre à ces besoins, une migration sera réalisée pour passer du format SQLite à un format NoSQL plus adapté. La migration s'appuie sur un jeu de données nommé ClassicModel.

OBJECTIFS

L'objectif de ce projet est de changer de technologie pour stocker les données de l'entreprise de Mme Dupont. Pour cela, nous avons choisi d'adopter un type de base de données NoSQL adapté aux besoins de l'entreprise. Ce projet de migration sera réalisé en plusieurs étapes essentielles.

Dans un premier temps, des requêtes SQL seront créées sur la base de données relationnelle pour se familiariser avec les données. Par la suite, un schéma optimal pour la base NoSQL sera défini, intégrant des relations imbriquées ou dénormalisées pour répondre aux exigences spécifiques du modèle NoSQL.

Un pseudo-algorithme sera élaboré pour automatiser le passage du modèle relationnel au modèle NoSQL. Celui-ci prendra en compte la transformation des relations entre les tables en documents imbriqués. Enfin, des requêtes seront exécutées dans le nouvel environnement NoSQL pour vérifier que la migration a été correctement effectuée, que les données ont été transférées avec succès et qu'elles sont accessibles et conformes aux attentes.

Méthodologie

PROCESSUS DE MIGRATION

Le processus de migration s'est déroulé en plusieurs étapes clés, visant à assurer une transition fluide et efficace de la base de données relationnelle SQLite vers MongoDB:

1. Création de requêtes SQL sur la base de données initiale
   - Des requêtes SQL ont été écrites pour explorer et analyser les données dans SQLite. Les relations complexes entre les tables, comme Orders, OrderDetails, et Products, ont été étudiées. Ces requêtes ont permis de créer des vues complètes, facilitant la compréhension de la structure des données et la préparation à la migration.
2. Transformation des données
   - Les données relationnelles extraites ont été converties en documents JSON, le format de MongoDB. Une dénormalisation a été réalisée, regroupant des informations liées (par exemple, les détails des commandes) dans des documents uniques, simplifiant ainsi les futures requêtes.
3. Insertion dans MongoDB
   - Les documents JSON transformés ont été insérés dans des collections MongoDB (Orders, Customers, Employees, Products). Chaque collection a été optimisée pour répondre à des besoins spécifiques des requêtes, en tenant compte des cas d'utilisation identifiés lors de l'analyse initiale.
4. Validation de la migration
   - Des requêtes NoSQL ont été exécutées pour vérifier l'exactitude des données migrées. Des tests ont été réalisés pour s'assurer que les données étaient cohérentes et accessibles, avec des résultats conformes à ceux obtenus dans l'ancien modèle relationnel.

EXEMPLE DE PSEUDO-ALGO

```
Début
  Fonction MigrationVersMongoDB(baseDeDonneesSQL, baseDeDonneesMongoDB)
    // Connexion à la base de données relationnelle (SQL)
    ConnexionSQL = Connexion(baseDeDonneesSQL)

    // Connexion à la base de données MongoDB
    ConnexionMongoDB = Connexion(baseDeDonneesMongoDB)

    // Récupérer la liste des tables SQL à migrer
    tables = ObtenirTables(ConnexionSQL)

    Pour chaque table dans tables
      Si table a des relations
        DonneesSQL = EffectuerJointure(ConnexionSQL, table)
      Sinon
        DonneesSQL = ObtenirDonnees(ConnexionSQL, table)

      DonneesMongoDB = TransformerEnDocumentMongoDB(DonneesSQL)
      InsérerDansMongoDB(ConnexionMongoDB, DonneesMongoDB)
    Fin Pour

    FermerConnexion(ConnexionSQL)
    FermerConnexion(ConnexionMongoDB)
  Fin Fonction
Fin
```

Architecture cible

CHOIX DU MODELE NOSQL

Pour notre projet de migration des données d'un format SQL vers NoSQL, nous avons décidé d'opter pour une base de données de type document, spécifiquement MongoDB. MongoDB est une base de données NoSQL orientée document. Cela signifie que les données sont sous la forme de JSON/BSON. Dans ce type de base de données, il peut y avoir plusieurs collections contenant des documents similaires. Chaque document constitue un enregistrement, permettant ainsi de stocker des données complexes et hiérarchisées.

Nous avons choisi ce modèle de données NoSQL pour plusieurs raisons.
Dans un premier temps, ce format de données offre une grande flexibilité dans le schéma. Chaque document peut avoir une structure différente. En effet, nous disposons d'une base de données avec plusieurs tables comme ***Products, Orders, Customers***. Le modèle de données document permet d'adapter facilement la structure des données au fur et à mesure des évolutions de notre projet.

Le type de base de données document est le modèle le plus adapté pour notre projet car il est avantageux en termes de scalabilité horizontale grâce à la partition des documents. De plus MongoDB respecte les propriétés CP (Consistency et Partition tolerance) dans le théorème CAP. En effet, MongoDB assure que les données sont cohérentes et est conçu pour continuer à fonctionner, même si une partition du réseau empêche certains nœuds de communiquer entre eux.

SCHÉMA CIBLE DES DONNÉES DANS LA BASE DE DONNÉES NOSQL

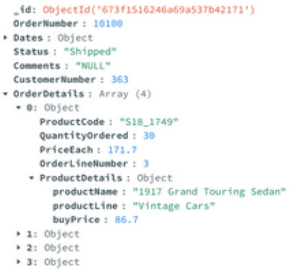
Nous avons décidé d'organiser les données en plusieurs collections. Avant la migration, nous avons réaliser une analyse du modèle relationnel qui a permis d'identifier les relations clés entre les tables et leur pertinence pour le modèle NoSQL. Les tables incluaient les produits, les commandes, les clients, les employés, les détails des commandes, les paiements et les bureaux. Nous avons regroupées les données, dispersées et reliées par des clés primaires et étrangères, en collections NoSQL pour simplifier les relations et optimiser l'accès. Pour cela, nous avons créé 4 collections afin d'organiser les données de manière structurée et faciliter leur accès dans le nouveau format NoSQL.

- **Products** : Représente les produits avec des informations essentielles pour la gestion commerciale.
- **Orders** : Combine les données des commandes (Orders) et des détails des commandes (OrderDetails) pour chaque commande. Chaque document représente une commande avec les détails de la commande et les produits commandés (Products).
- **Customers** : Inclut les informations des clients (Customers) ainsi que leurs détails de paiement (PaymentDetails).
- **Employees** : Contient les informations sur les employés (Employees) et les bureaux associés(Offices).

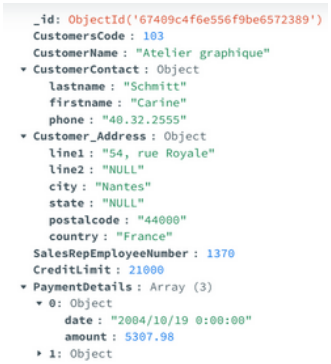
EXEMPLE DE DOCUMENT DANS NOS COLLECTIONS

PRODUCTS

ORDERS



CUSTOMERS



EMPLOYEES

Difficultés rencontrés

TECHNIQUES

La migration des données d'une base relationnelle SQLite vers MongoDB a soulevé plusieurs défis techniques. Le traitement des relations complexes entre les tables (Orders, OrderDetails, Products) a nécessité des jointures précises pour regrouper correctement les données dans le modèle NoSQL. De plus, la construction des collections adaptées à MongoDB a été un point clé : il fallait équilibrer dénormalisation et performance, tout en veillant à maintenir la cohérence des données. Enfin, les requêtes post-migration ont révélé des problèmes tels que les doublons ou des calculs incorrects des montants, dus à des erreurs dans le regroupement ou à des relations mal gérées.

ORGANISATIONNELLES

La validation des résultats intermédiaires a parfois ralenti le projet, car il fallait s'assurer que les nouvelles collections respectaient à la fois les besoins métiers et les contraintes de performance.

SOLUTIONS APPORTÉES

Une méthodologie structurée a été mise en place pour gérer chaque étape de la migration : analyse, transformation, insertion, validation. Des tests rigoureux ont permis d'assurer la cohérence des données, et une communication régulière au sein de l'équipe a facilité la résolution des problèmes rapidement et efficacement.

Conclusion

Dans ce projet de migration de données d'une base relationnelle SQLite vers une base de données NoSQL, nous avons suivi une méthodologie claire et structurée pour garantir une transition efficace et sans erreur. Nous avons d'abord extrait les données de la base SQLite en utilisant des requêtes SQL adaptées, puis réfléchi au format le plus approprié pour les intégrer dans MongoDB, en tenant compte des spécificités des bases de données NoSQL. Le script Python que nous avons développé a permis d'effectuer la migration des données. Enfin, une étape de validation rigoureuse a été mise en place pour garantir l'intégrité et la cohérence des données après la migration.

En conclusion, ce projet illustre l'importance d'une modélisation adaptée lors du passage d'un modèle relationnel à un modèle orienté documents, ainsi que les bénéfices qu'apporte MongoDB en termes de gestion des données non structurées ou semi-structurées. Ainsi en suivant les étapes, depuis l'extraction des données relationnelles via des requêtes SQL, jusqu'à leur transformation et insertion dans MongoDB, nous avons pu assurer une transition fluide et cohérente malgré certaines difficultés rencontrées.