SAÉ NoSQL

Migration de données d'un format
SQLite à un format NoSQL

Baptiste TIVRIER
Hsiao-Wen-Paul LO
BUT 3 SD - VCOD - G34

IUT de Paris - Rives de Seine
Université Paris Cité

Introduction :

La directrice d'une entreprise de voitures, Paula Dupont, a constaté plusieurs problèmes affectant la performance et la fiabilité de sa base de données "ClassicModel". Parmi ceux-ci, la latence élevée des requêtes et la perte de données en raison de défaillances serveur fréquentes figurent parmi les plus préoccupantes. Afin de pallier ces difficultés, elle souhaite adopter une solution NoSQL, qui permettra d'améliorer la scalabilité, la performance des requêtes et la tolérance aux pannes.

Objectifs de la demande :

L'objectif principal de la demande est de transférer efficacement les données de la base actuelle, "ClassicModel", d'un format SQLite vers une base de données NoSQL, tout en garantissant l'intégrité des données et en optimisant les performances dans le but de répondre aux limitations de l'infrastructure existante. Le processus de migration de la base de données vers NoSQL se fera en plusieurs étapes majeures :

1) Création de requêtes SQL sur la base de données initiale : Cette étape consiste à extraire les données nécessaires de la base de données SQLite à l'aide de requêtes SQL spécifiques.

2) Réflexion sur le format des données NoSQL : Une analyse sera réalisée pour déterminer comment les données extraites seront structurées dans le format NoSQL. Ce format sera choisi en fonction des besoins de l'entreprise en matière de performance, de scalabilité et de flexibilité.

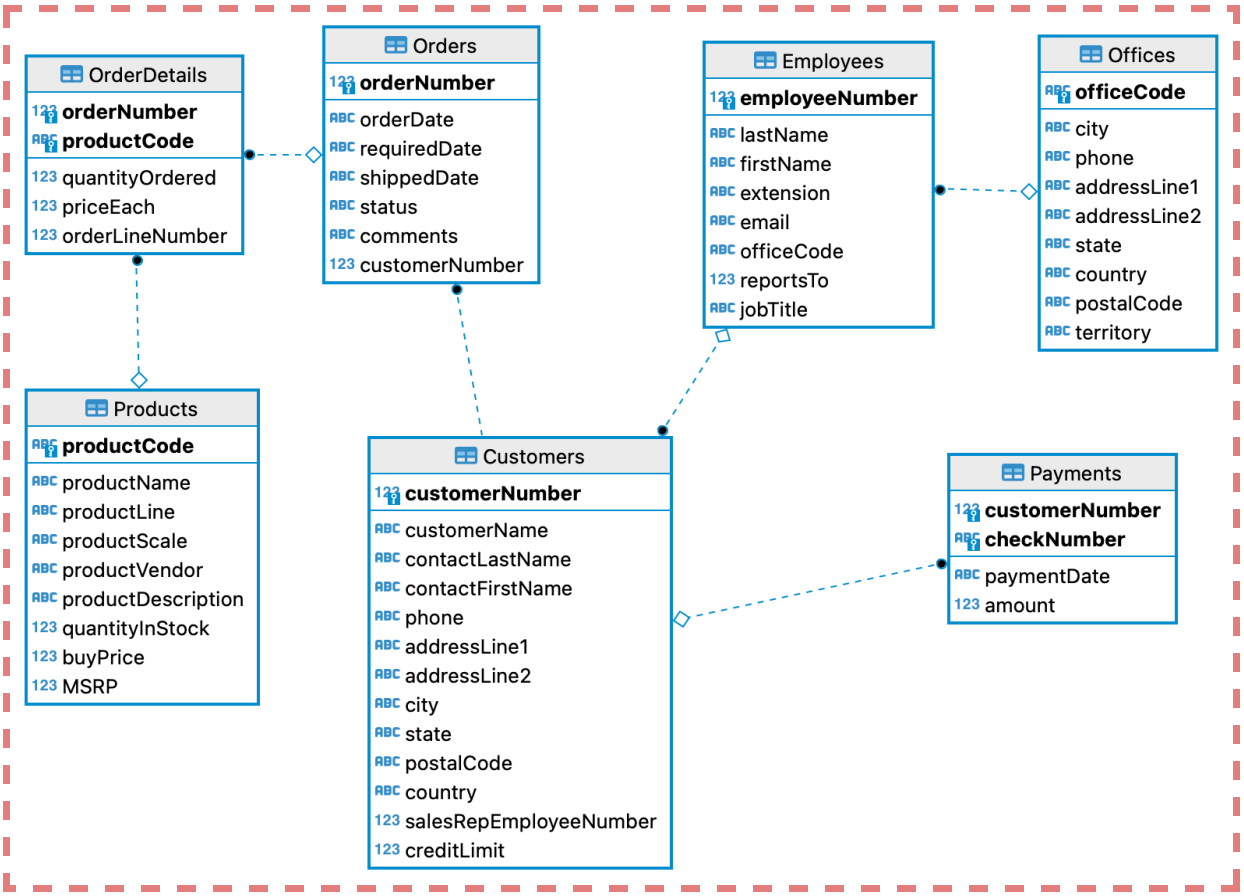
3) Écriture du script Python de migration : Un script Python sera développé pour réaliser la migration des données depuis SQLite vers NoSQL. Ce script prendra en charge les connexions, l'extraction des données et leur insertion dans la nouvelle base de données NoSQL.

4) Création de requêtes dans le nouveau format NoSQL : Une fois la migration effectuée, il sera important de s'assurer que les données sont correctement transférées en créant des requêtes dans le format NoSQL, en utilisant MongoDB, pour valider que la migration a été réussie.

Description de la base :

La base “ClassicModel” est une base de données relationnelle contenant des informations sur une entreprise spécialisée dans la vente de répliques miniatures de véhicules de tout type. Elle est organisée en plusieurs tables interconnectées, chacune dédiée à la gestion des informations commerciales, des clients, des commandes et des paiements. Voici une description des tables de la base, illustrée par le schéma joint :

“Customers” : Cette table enregistre les informations relatives à chaque client de l'entreprise ; **“Payments”** : Cette table enregistre les paiements effectués par les clients, y compris les dates et les montants ; **“OrderDetails” :** Cette table contient des détails sur chaque produit inclus dans une commande spécifique ; **“Orders” :** Cette table contient des informations générales sur les commandes passées par les clients, y compris les dates et les statuts ; **“Products” :** Cette table contient des informations sur les produits disponibles à la vente, y compris les prix et descriptions ;**“Employees” :** Cette table contient les informations relatives aux employés de l'entreprise ; **“Offices” :** Cette table contient des informations sur les bureaux de l'entreprise.



Réflexion sur la migration - ‘Type de base’ :

Pour cette migration, nous avons choisi d’adopter un modèle document pour structurer les données dans la base NoSQL, plutôt que d'opter pour d'autres modèles tels que clé-valeur, graphe ou colonne. Chaque approche NoSQL présente des avantages spécifiques, mais le modèle document est mieux adapté aux besoins de la demande pour plusieurs raisons :

Flexibilité : Le format document permet de stocker des données semi-structurées ou non structurées, tout en permettant des modifications faciles à la structure des documents au fur et à mesure de l'évolution des besoins métier ; **Lecture optimisée :** Contrairement au modèle clé-valeur, qui est souvent utilisé pour des accès très simples à des paires clé/valeur, ou au modèle colonne qui peut devenir complexe à gérer pour des structures de données plus riches, le modèle document permet de stocker des données complexes et hiérarchiques dans un seul document. Cela est particulièrement utile pour les tables de ClassicModel, qui contiennent des relations complexes entre les entités (comme les clients, commandes, employés, etc.) ; **Modélisation intuitive :** En regroupant les données liées sous forme de documents, nous pouvons facilement encapsuler les informations pertinentes dans des objets indépendants, facilitant ainsi les requêtes complexes tout en maintenant une structure naturelle pour les utilisateurs et les applications.

Réflexion sur la migration - ‘Schéma des données’ :

Concernant la structuration des données, nous avons décidé de regrouper les tables de ClassicModel dans trois collections principales pour refléter la logique métier de manière optimale :

1) Collection "customers" : Cette collection regroupe les informations des clients (tirées de la table “Customers”) et leurs paiements associés (en intégrant la table “Payments” dans la collection). Chaque document représentera un client, et les paiements seront directement inclus dans ce même document sous forme de sous-documents ou tableaux. Cela permet d'avoir un accès direct et rapide à tous les paiements d’un client en particulier, simplifiant ainsi les requêtes relatives aux transactions financières.

2) Collection "offices" : Dans cette collection, nous regroupons les informations sur les bureaux (table “Offices”) et les employés (table “Employees”). Chaque document représentera un bureau et contiendra une liste des employés qui y sont affectés. Ce modèle permet de visualiser rapidement tous les employés d’un même bureau sans avoir à effectuer des jointures complexes. Chaque employé sera encapsulé comme un sous-document au sein de son bureau, ce qui offre une gestion centralisée des informations liées à un bureau donné.

3) Collection "orders" : Cette collection contiendra des informations détaillées sur les commandes des clients (table “Orders”) et les produits associés à ces commandes (en intégrant les tables “OrderDetails” et “Products” dans un même document). Chaque document représentera une commande, et les détails des produits (quantités, prix, etc.) seront inclus comme sous-documents dans cette commande. Cette approche facilite les recherches sur les commandes spécifiques et leurs produits, tout en conservant une structure cohérente et facilement navigable pour chaque commande passée.

Ce choix de structure vise à maximiser la performance des requêtes en réduisant la complexité des jointures tout en optimisant la lisibilité et l'efficacité des opérations de lecture dans le nouveau système NoSQL. En regroupant les données de manière logique et naturelle dans ces collections, nous nous assurons également que la migration reflétera fidèlement les relations métier tout en exploitant les avantages d’un modèle document flexible.