



Institut universitaire de
Paris Rives de Seine

SAE NoSQL

Migration de données

Auteur :

Bastien EBELLY
Cheick GUEYE
Mehdi BENAYED

26 septembre 2024

Partie 1

Introduction et définition de l'objectif finale

Dans ce projet, nous avons pour objectif de migrer une base de données d'un format relationnel SQLite vers un format NoSQL. Pour y parvenir, nous allons suivre une démarche structurée en quatre étapes :

1. **Créer des requêtes SQL** pour extraire les données dont nous avons besoin. C'est essentiel pour avoir une bonne base d'informations.
2. **Définir le format des données** que nous voulons dans le système NoSQL et élaborer l'algorithme nécessaire pour cette migration.
3. **Écrire un script Python** qui facilitera ce transfert de données. On veut que ce soit fluide et sans accros.
4. **Vérifier la migration** en formulant des requêtes dans le nouveau format NoSQL, pour s'assurer que tout est en ordre.

Nous commencerons par analyser les données pour bien comprendre ce dont nous avons besoin, puis nous choisirons le type de base de données NoSQL le plus adapté, qu'il s'agisse d'une base clé-valeur, de documents ou de graphes.

Ce processus nous permettra de garantir une transition réussie vers le modèle NoSQL tout en respectant les spécificités de nos données.

Partie 2

Migration vers une base de données NoSQL

2.1 Description des jeux de données

Les données proviennent de la base de données SQLite initiale utilisée pour stocker des informations de gestion des commandes nommée *ClassicModel*. Pour garantir l'objectif final, notre groupe a envisagé MongoDB, bien que nous soyons encore en discussion sur le choix final. Voici les collections que nous allons créer dans cette nouvelle base de données NoSQL :

2.1.1 Dictionnaire de données - Customers

Cette collection contient toutes les informations relatives aux clients. Elle inclut :

- Les coordonnées du client (nom, prénom, téléphone, etc.).
- Les détails financiers comme la limite de crédit.
- Les relations avec les employés qui les gèrent.

2.1.2 Orders

La collection **Orders** stocke les commandes passées par les clients. Chaque document contient les informations suivantes :

- Le numéro de commande, la date de commande, la date de livraison.
- Le statut de la commande et d'éventuels commentaires.

2.1.3 OrdersDetails

Cette collection détaille chaque produit associé à une commande spécifique. Elle inclut :

- Le code du produit, la quantité commandée, le prix unitaire.
- L'ordre d'apparition du produit dans la commande...

2.1.4 Products

La collection **Products** liste les produits vendus. Chaque document comprend des informations telles que :

- Le nom du produit, la ligne de produit, la description.
- Le prix d'achat, le prix de vente recommandé, et le stock disponible...

2.1.5 Employees et Offices

Ces collections sont liées aux employés et aux bureaux dans lesquels ils travaillent. Elles permettent de suivre la hiérarchie des employés et de localiser leur bureau (etc..)

2.1.6 Payments

La collection Payments suit les paiements effectués par les clients, en incluant la date et le montant de chaque paiement. (etc.)

2.2 Définition des requêtes à utiliser

Après analyse des données, le type de base de données NoSQL choisi est **MongoDB**, une base orientée documents. MongoDB est très adaptée aux besoins de ce projet en raison de sa capacité à gérer des données semi-structurées et à modéliser les relations entre entités via des documents imbriqués et des références entre collections. Il est souple, sur les schémas, il est plus optimisable et permet d'imbriquer les tables les unes sur les autres pour faire moins de requête.

Partie 3

Les requêtes

3.1 Comment nous avons exécuter les requêtes

Voici les requêtes SQL qui nous permettront de vérifier si la migration vers MongoDB s'est déroulée correctement. L'objectif est de comparer les résultats des requêtes avant et après migration pour s'assurer que les données et leurs relations sont fidèles.

```
# Importation des modules nécessaires
import sqlite3 # Pour interagir avec une base de données SQLite
import pandas as pd # Pour la manipulation et l'analyse de données

# Connexion à la base de données SQLite
conn = sqlite3.connect("ClassicModel.sqlite") # WARNING aux importations au format ".data" supprimer et remettre à jour

# Dictionnaire contenant 10 requêtes SQL différentes
requetes_sql = {
    "Clients sans commandes": """
        SELECT c.customerNumber, c.customerName, c.contactLastName, c.contactFirstName, c.country
        FROM Customers c
        LEFT JOIN Orders o ON c.customerNumber = o.customerNumber
        WHERE o.customerNumber IS NULL
        ORDER BY c.customerNumber;
    """,
    "Performances des employés": """
        SELECT e.employeeNumber, e.lastName, e.firstName,
            COUNT(DISTINCT c.customerNumber) AS nb_clients,
            COUNT(DISTINCT o.orderNumber) AS nb_commandes,
            SUM(od.quantityOrdered * od.priceEach) AS total_ventes
        FROM Employees e
        LEFT JOIN Customers c ON e.employeeNumber = c.salesRepEmployeeNumber
        LEFT JOIN Orders o ON c.customerNumber = o.customerNumber
        LEFT JOIN OrderDetails od ON o.orderNumber = od.orderNumber
        GROUP BY e.employeeNumber
        ORDER BY e.employeeNumber;
    """,
    "Analyse par bureau": """
        SELECT b.officeCode,
            COUNT(DISTINCT c.customerNumber) AS nb_clients,
            COUNT(DISTINCT o.orderNumber) AS nb_commandes,
            SUM(od.quantityOrdered * od.priceEach) AS montant_total,
            COUNT(DISTINCT CASE WHEN c.country != b.country THEN c.customerNumber END) AS clients_internationaux
        FROM Offices b
        LEFT JOIN Employees e ON b.officeCode = e.officeCode
        LEFT JOIN Customers c ON e.employeeNumber = c.salesRepEmployeeNumber
        LEFT JOIN Orders o ON c.customerNumber = o.customerNumber
        LEFT JOIN OrderDetails od ON o.orderNumber = od.orderNumber
        GROUP BY b.officeCode
    """
}
```

Sortie 1 – extrait de validation des données par requêtes

Index	customerNumber	total_achats	total_paiements
0	114	200995	195365
1	119	180125	136340
2	124	654858	647596
3	141	912294	793051
4	145	145042	119029
5	148	172990	172990

Sortie 2 – extrait du dataframe

Key	Type	Size	Value
Analyse par bureau	str	704	SELECT b.officeCode,
Clients sans commandes	str	268	SELECT c.customerNumber, c.customerName, c.contac...
Commandes par produit	str	449	SELECT p.productCode, p.productName,
Les Clients sont effectivement en retard de paiement	str	812	WITH AchatsTotaux AS (
Performances des employés	str	563	SELECT e.employeeNumber, e.lastName, e.firstName,...
Produits vendus à perte	str	300	SELECT p.productCode, p.productName, o.customerNu...
Tables de Contingence des commandes en fonction pays du client	str	399	SELECT p.productline, c.country,
Tables de Contingence des commandes sur les produits achetés et le pays du client	str	409	SELECT p.productline, c.country,
Top 10 produits à forte marge	str	286	SELECT p.productCode, p.productName,
Ventes par pays	str	502	SELECT c.country,

Sortie 3 – extrait des requête sql

```

1  """Pour chaque ligne dans Orders_table :|
2
3      Créer un document 'order' :
4      - orderNumber
5      - orderDate
6      - requiredDate
7      - shippedDate
8      - status
9      - comments
10
11      Pour chaque ligne dans OrderDetails_table où OrderDetails_table.orderNumber = Orders_table.orderNumber
12      Créer une sous-collection 'OrderDetails' :
13      - quantityOrdered
14      - priceEach
15      - orderLineNumber
16
17      Pour chaque ligne dans Products_table où Products_table.productCode = OrderDetails_table.productCode
18      Imbriquer les informations du produit :
19      - productCode
20      - productName
21      - productLine
22      - productScale
23      - productVendor
24      - productDescription
25      - quantityInStock
26      - buyPrice
27      - MSRP
28      Fin pour
29
30      Fin pour (OrderDetails)
31
32      Pour chaque ligne dans Customers_table où Customers_table.customerNumber = Orders_table.customerNumber
33      Créer une sous-collection 'Customers' :
34      - customerName

```

Sortie 4– schéma cible des données NoSQL choisie
algorithme

```

1  """
2  Orders= { - OrderDetails = { - orderNumber
3                                - quantityOrdered
4                                - priceEach
5                                - orderLineNumber
6                                - Products = { - productCode
7                                                  - productName
8                                                  - productLine
9                                                  - productScale
10                                                 - productVendor
11                                                 - productDescription
12                                                 - quantityInStock
13                                                 - buyPrice
14                                                 - MSRP
15                                             }
16                                }
17      - orderDate
18      - requiredDate
19      - shippedDate
20      - status
21      - comments
22      - Customers = { - payments = { - customerNumber
23                                    - checkNumber
24                                    - paymentDate
25                                    - amount
26                                }
27      - customerName
28      - contactLastName
29      - contactFirstName
30      - phone
31      - addressLine1
32      - addressLine2
33      - city

```

Sortie 5 – extrait de code pour le Pseudo

Partie 4

Schéma ciblé et conception du Pseudo-algorithme

Le pseudo-algorithme et le schéma des données que nous présentons ici ne sont pas encore définitifs. Il est possible que certaines parties doivent être ajustées après la validation finale de notre code Python, car nous devons encore vérifier si la logique et la structure des données s'alignent bien avec le fonctionnement du programme. Cependant, dans l'avancement de notre projet, nous avons inclus ces éléments dans le rapport pour montrer la direction de notre travail. Ils serviront de base pour la suite de l'implémentation et pourront être adaptés lors de l'intégration finale dans la base NoSQL.