

# Institut universitaire de Paris Rives de Seine

# SAE NoSQL Migration de données

Auteur:

Bastien EBELLY Cheick GUEYE Mehdi BENAYED

# 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 accrocs.
- 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.

# 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 NoSOL:

#### 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.

# Les requêtes

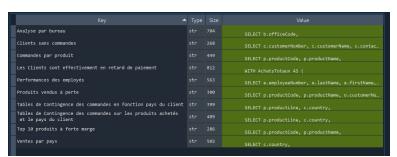
### 3.1 Comment nous avons exécuter les requêtes

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.

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



**Sortie 3** – extrait des requête sql

```
****Pour chaque Ligne dans Orders_table :

Créer un document 'order' :

- orderbate
- orderbate
- requiredate
- shippedbate
- shippedbate
- shippedbate
- shippedbate
- comments
- comments
- comments
- comments
- créer une sous-collection 'OrderDetails_table.orderNumber = Orders_table.orderNumber
- quantityOrdered
- pricetach
- orderLineNumber
- production
- pricetach
- orderLineNumber
- production
- produ
```

**Sortie 4**– schéma cible des données NoSQL choisie algorithme

Sortie 5 – extrait de code pour le Pseudo

# Schéma ciblé et conception du Pseudoalgorithme

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.