Mini-Projet

Big Data Warehouse NoSQL – Simulation de flux de commandes multicanal

Encadré par : Professeur M.Saadi

Réalisé par :

CHAOUI Khalil

Akilou Illa Abdourazak

Boumhand Yassine

Filière : IID2

TABLE DES MATIÈRES

INTRODUCTION

Contexte du projet

Avec le développement rapide des technologies numériques, les entreprises génèrent aujourd'hui des volumes de données de plus en plus importants et variés. Les systèmes d'information modernes doivent gérer des données provenant de plusieurs sources, telles que les sites web, les applications mobiles et les points de vente physiques. Ces données arrivent de manière continue, avec des formats parfois différents et évolutifs.

Dans un contexte commercial, les entreprises multicanales doivent être capables de collecter, stocker et analyser les commandes clients afin de mieux comprendre le comportement des consommateurs, suivre les ventes et prendre des décisions stratégiques. Les bases de données relationnelles classiques, comme MySQL, bien qu'efficaces pour des données structurées et stables, montrent certaines limites face à ce type de données massives et hétérogènes, notamment à cause de la rigidité du schéma et de la complexité des relations.

C'est dans ce cadre que les bases de données NoSQL ont émergé. Elles offrent une plus grande flexibilité dans la modélisation des données, une meilleure scalabilité et une capacité à gérer des données semi-structurées, souvent représentées sous forme de documents JSON. MongoDB, en particulier, est une base de données NoSQL orientée documents largement utilisée dans les systèmes Big Data et les architectures décisionnelles modernes.

Ce mini-projet s'inscrit dans ce contexte et a pour objectif de simuler un environnement de vente multicanal (site web, application mobile et boutique physique), de collecter les commandes générées par ces différentes sources, puis de construire un Big Data Warehouse basé sur MongoDB. Les données collectées seront ensuite exploitées pour réaliser des analyses décisionnelles et produire des tableaux de bord permettant de visualiser les indicateurs clés de performance.

Définition de NoSQL

Le terme NoSQL, qui signifie Not Only SQL, désigne une famille de bases de données qui ne suivent pas le modèle relationnel classique basé sur des tables, des lignes et des relations strictes. Contrairement aux bases de données relationnelles, les bases NoSQL ne

nécessitent pas un schéma fixe et prédéfini, ce qui leur permet de gérer plus facilement des données volumineuses, variées et évolutives.

Les bases de données NoSQL sont particulièrement adaptées aux environnements Big Data, où les données peuvent être semi-structurées ou non structurées, et où la vitesse d'ingestion ainsi que la capacité de montée en charge (scalabilité) sont des critères essentiels. Elles permettent de stocker et de traiter de grandes quantités de données provenant de sources multiples, telles que les applications web, mobiles ou les systèmes distribués.

Il existe plusieurs types de bases de données NoSQL, notamment les bases orientées documents, les bases clé-valeur, les bases orientées colonnes et les bases orientées graphes. Chaque type répond à des besoins spécifiques selon la nature des données et les usages attendus.

Dans le cadre de ce projet, l'approche NoSQL permet de stocker les commandes clients sous forme de documents, sans imposer une structure rigide. Cette flexibilité facilite l'intégration de données hétérogènes issues de différents canaux de vente et simplifie l'évolution du modèle de données au fil du temps.

Présentation de MongoDB

MongoDB est une base de données NoSQL orientée documents, conçue pour stocker et gérer des données de manière flexible et scalable. Contrairement aux bases de données relationnelles traditionnelles, MongoDB ne repose pas sur des tables, mais sur des collections contenant des documents. Chaque document est stocké sous un format proche du JSON, appelé BSON (Binary JSON), ce qui permet de représenter des données complexes et hiérarchiques.

Dans MongoDB, un document correspond à une entité complète, par exemple une commande client. Les informations liées à cette commande, telles que les produits, les quantités ou le statut, peuvent être intégrées directement dans le même document, évitant ainsi l'utilisation de jointures complexes. Cette approche permet d'améliorer les performances de lecture et de simplifier la modélisation des données.

MongoDB offre plusieurs avantages importants, notamment la flexibilité du schéma, la haute performance pour les opérations de lecture et d'écriture, ainsi qu'une scalabilité horizontale, ce qui signifie qu'il peut facilement gérer de grands volumes de données en répartissant la charge sur plusieurs serveurs. Ces caractéristiques en font une solution largement utilisée dans les applications web modernes et les systèmes Big Data.

Dans le cadre de ce mini-projet, MongoDB est utilisé comme entrepôt de données NoSQL (Big Data Warehouse) pour centraliser les commandes provenant de différents canaux de vente (site web, application mobile et boutique physique). Grâce à ses capacités d'agrégation, MongoDB permet de réaliser des analyses décisionnelles efficaces et de produire des indicateurs clés tels que le chiffre d'affaires, les ventes par canal et les taux d'annulation.

1. ARCHITECTURE GÉNÉRALE DU SYSTÈME

1.1. Vue d'ensemble de l'architecture

L'architecture mise en place dans ce projet vise à simuler un environnement de vente multicanal et à construire un Big Data Warehouse NoSQL basé sur MongoDB. Le système est conçu pour collecter des données provenant de plusieurs sources, les intégrer dans un entrepôt centralisé, puis les exploiter à des fins d'analyse décisionnelle.

Le flux global des données peut être résumé comme suit : les commandes sont générées par différents simulateurs, stockées temporairement sous forme de fichiers JSON, puis collectées et intégrées dans une base MongoDB. Une fois les données centralisées, des requêtes d'agrégation sont utilisées pour produire des indicateurs décisionnels, qui sont ensuite visualisés sous forme de tableaux de bord.

1.2. Sources de données

Le système repose sur trois sources de données représentant les différents canaux de vente de l'entreprise :

- **Site web :** génère des commandes en ligne contenant des informations de livraison.
- **Application mobile :** produit des commandes similaires à celles du site web, avec des données de localisation et de livraison.
- **Boutique physique :** génère des commandes réalisées sur place, sans adresse de livraison.

Chaque source est simulée à l'aide d'un script Python qui crée des fichiers JSON représentant des commandes clients, déposés dans des répertoires dédiés.

1.3. Couche de collecte et d'intégration

Une couche de collecte des données est implémentée à l'aide d'un script Python chargé de surveiller en continu les répertoires des différentes sources. Ce script détecte automatiquement l'arrivée de nouveaux fichiers JSON, vérifie leur validité, puis extrait les informations nécessaires.

Les commandes valides sont ensuite insérées dans la base de données MongoDB, tandis que les fichiers traités sont déplacés vers un répertoire d'archivage. Cette étape permet d'assurer la fiabilité du processus d'intégration et d'éviter le retraitement des mêmes fichiers.

1.4. Entrepôt de données NoSQL

L'entrepôt de données est basé sur MongoDB, utilisé comme une base centralisée pour stocker l'ensemble des commandes collectées. Les données sont organisées dans une collection unique, où chaque document correspond à une commande complète.

Cette approche orientée documents permet de stocker les informations de manière flexible et cohérente, tout en facilitant les opérations de lecture et d'agrégation. MongoDB joue ainsi le rôle de Big Data Warehouse NoSQL, adapté à la gestion de données hétérogènes et évolutives.

1.5. Couche d'analyse et de reporting

La couche d'analyse repose sur les capacités d'agrégation de MongoDB, permettant de calculer des indicateurs clés tels que le chiffre d'affaires, les ventes par canal ou le taux de commandes annulées. Ces analyses constituent la base du processus décisionnel.

Les résultats obtenus sont ensuite visualisés à l'aide d'outils de reporting tels que MongoDB Charts et Power BI, afin de fournir des tableaux de bord clairs et exploitables par les décideurs.

1.6. Schéma global de l'architecture

L'architecture globale du système peut être résumée par le schéma suivant :

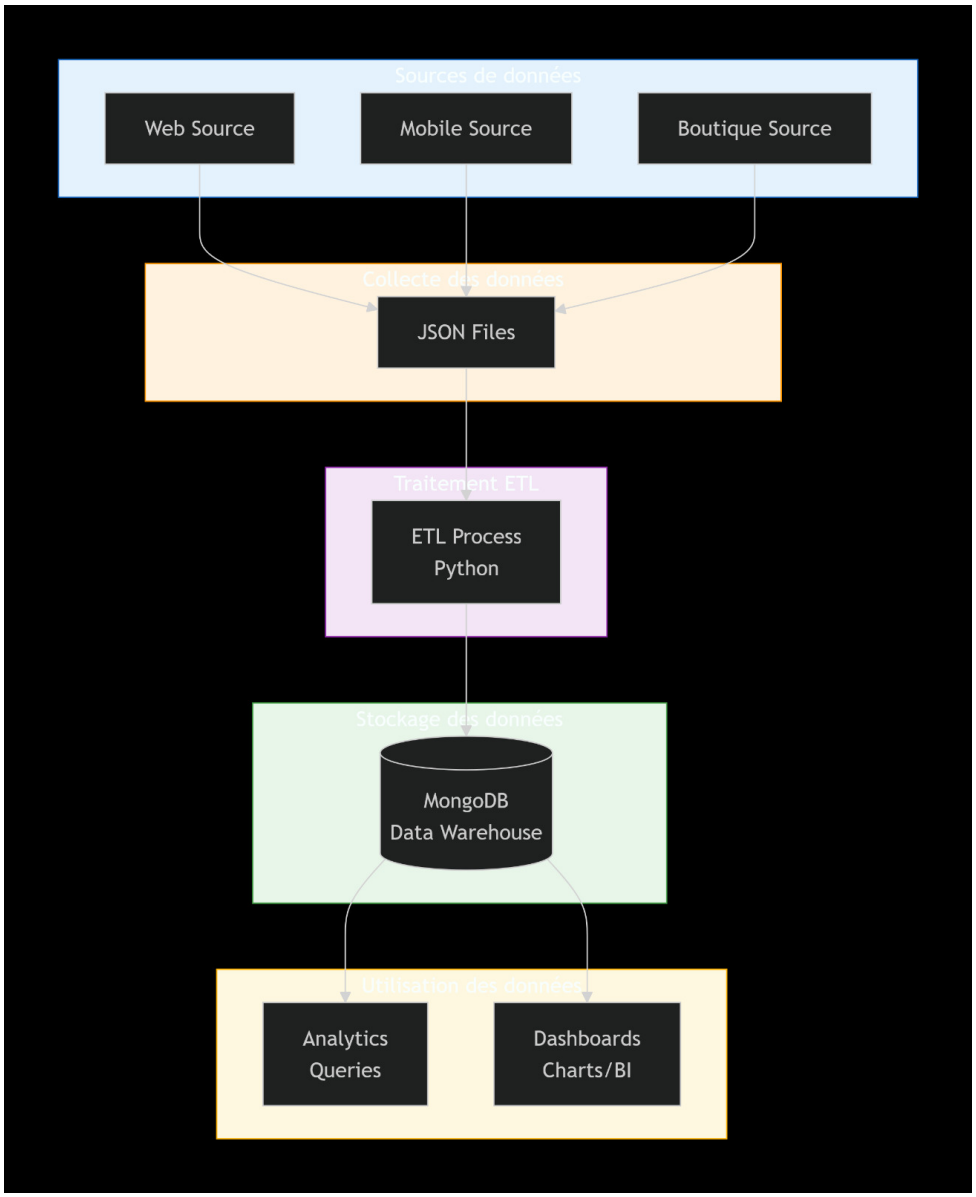

FIGURE 1.1 – Schéma global de l'architecture du système

2. SIMULATION DE LA PRODUCTION DE DONNÉES

2.1. Objectif de la simulation

L'objectif de cette étape est de simuler la génération de commandes clients provenant de différents canaux de vente. Dans un environnement réel, ces commandes seraient produites par des applications web, mobiles ou des systèmes de caisse en boutique. Dans le cadre de ce projet, cette production est simulée à l'aide de scripts Python afin de créer un flux continu de données réaliste.

La simulation permet de reproduire le comportement d'un système multicanal, où les commandes arrivent à des moments différents et sous des formats similaires mais légèrement variés selon la source.

2.2. Description des sources de données

Trois sources de données ont été définies, correspondant aux canaux de vente de l'entreprise :

- **Site web :** génère des commandes en ligne avec des informations de livraison.
- **Application mobile :** produit des commandes similaires au site web, avec des données de localisation.
- **Boutique physique :** génère des commandes effectuées sur place, sans adresse de livraison.

Chaque source est simulée indépendamment afin de refléter la diversité des canaux de vente.

2.3. Structure d'une commande

Chaque commande est représentée sous forme d'un fichier JSON, contenant les informations essentielles suivantes :

- Identifiant de la commande
- Date et heure de la commande
- Canal de vente

— Liste des produits commandés

— Quantité et prix unitaire des produits

— Statut de la commande (payée ou annulée)

— Adresse de livraison (uniquement pour le web et le mobile)

Cette structure permet de stocker l'ensemble des informations liées à une commande dans un seul document, conformément au modèle orienté documents de MongoDB.

Listing 2.1 – Exemple de structure d'une commande


```
{
  "id_commande": "WEB-20251231152635-2907",
  "canal": "site_web",
  "date_commande": "2025-12-31T15:26:35.235324",
  "client": {
    "nom": "Th ophile de la Lef vre",
    "email": "tmartins@example.com",
    "telephone": "05 86 42 23 57"
  },
  "produits": [
    {
      "nom_produit": "iPhone 15 Pro",
      "quantite": 3,
      "prix_unitaire": 1100.0,
      "prix_total": 3300.0
    },
    {
      "nom_produit": "iPad Air",
      "quantite": 3,
      "prix_unitaire": 650.0,
      "prix_total": 1950.0
    },
    {
      "nom_produit": "Laptop Dell XPS",
      "quantite": 2,
      "prix_unitaire": 1200.0,
      "prix_total": 2400.0
    }
  ],
  "montant_total": 7650.0,
  "statut": "confirm e",
  "mode_paiement": "virement"
```

```
}
```

2.4. Scripts de simulation

La simulation de la production de données est réalisée à l'aide de trois scripts Python distincts :

- `site_web.py`
- `application_mobile.py`
- `boutique_physique.py`

Chaque script génère de manière aléatoire des commandes et les enregistre sous forme de fichiers JSON dans un répertoire spécifique correspondant à la source de données. Une nouvelle commande est générée toutes les deux à cinq secondes afin de simuler un flux quasi continu.

2.5. Organisation des répertoires

Les fichiers générés par les scripts de simulation sont stockés dans une arborescence de dossiers structurée comme suit :

Listing 2.2 – Structure des répertoires de données

```
/data/sources/site_web/
/data/sources/application_mobile/
/data/sources/boutique_physique/
```

Cette organisation facilite la détection des nouveaux fichiers par le script de collecte et permet de distinguer clairement l'origine de chaque commande.

2.6. Résultat de la simulation

À l'issue de cette étape, un ensemble de fichiers JSON représentant les commandes clients est disponible pour chaque canal de vente. Ces fichiers constituent la matière première du processus d'intégration et d'analyse décisionnelle qui sera présenté dans les chapitres suivants.

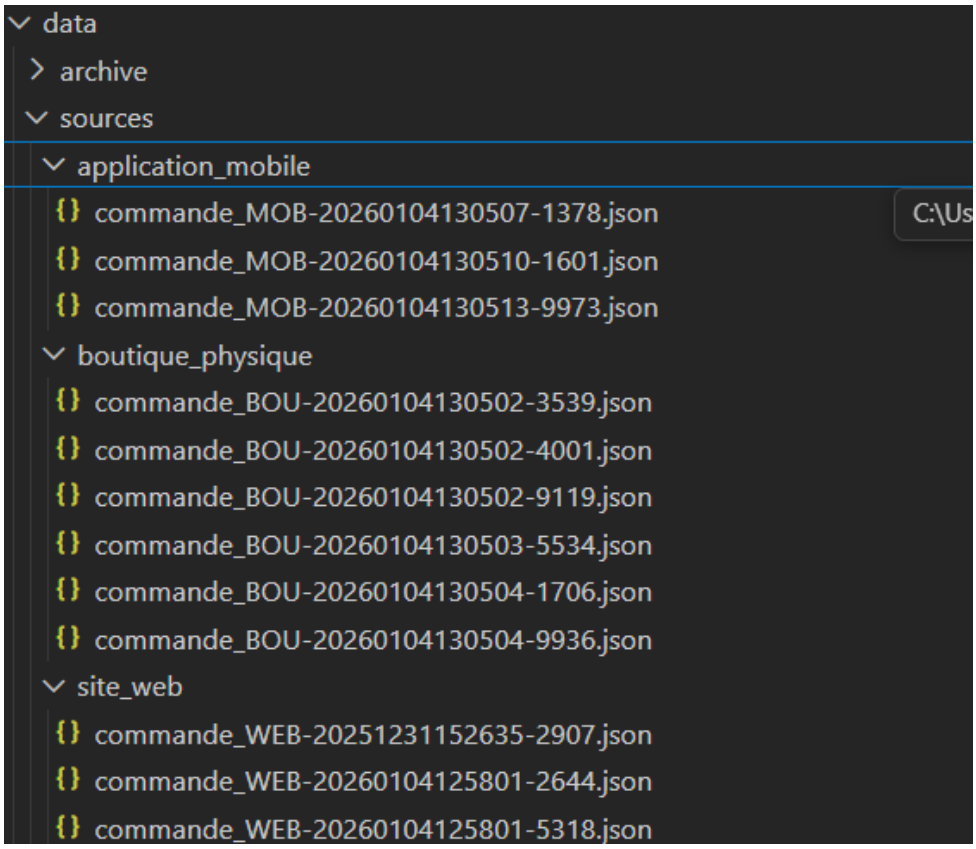

FIGURE 2.1 – Résultat de la simulation

3. PROCESSUS D'INTÉGRATION DES DONNÉES (DATA WAREHOUSE NOSQL)

3.1. Objectif du processus d'intégration

Le processus d'intégration des données a pour objectif de collecter, valider et centraliser les commandes générées par les différents canaux de vente dans un entrepôt de données NoSQL basé sur MongoDB. Cette étape constitue le cœur du système décisionnel, car elle permet de transformer des fichiers JSON dispersés en données exploitables pour l'analyse.

Dans un environnement réel, ce type de processus correspond à une chaîne d'ingestion de données continue. Dans ce projet, il est simulé à l'aide d'un script Python fonctionnant en quasi temps réel.

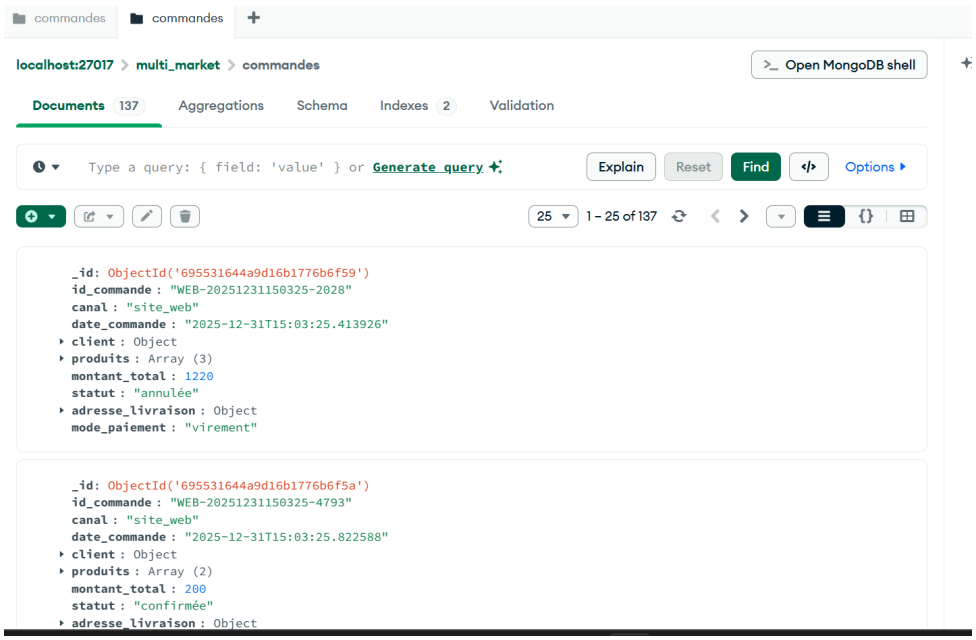

FIGURE 3.1 – Intégration dans MongoDB

3.2. Surveillance des sources de données

Un script de collecte est mis en place afin de surveiller en continu les répertoires contenant les fichiers JSON générés par les scripts de simulation. Ce script détecte automatiquement l'apparition de nouveaux fichiers dans les dossiers correspondant aux différentes sources (site web, application mobile et boutique physique).

La surveillance régulière des répertoires permet d'assurer une intégration rapide des nouvelles commandes, avec un délai maximal de quelques secondes entre la création du fichier et son insertion dans la base de données.

3.3. Validation des données

Avant l'intégration dans l'entrepôt de données, chaque fichier JSON est soumis à une étape de validation. Cette étape consiste à vérifier que le fichier respecte le format JSON et que les informations essentielles de la commande sont présentes.

Les fichiers corrompus ou mal formés sont ignorés afin de garantir la qualité des données stockées dans MongoDB. Cette approche permet de renforcer la robustesse du système et d'éviter l'insertion de données incorrectes.

3.4. Insertion dans l'entrepôt MongoDB

Une fois validées, les commandes sont insérées dans la base de données MongoDB. L'entrepôt de données est organisé comme suit :

— **Base de données :** `multi_market`

— **Collection :** `commandes`

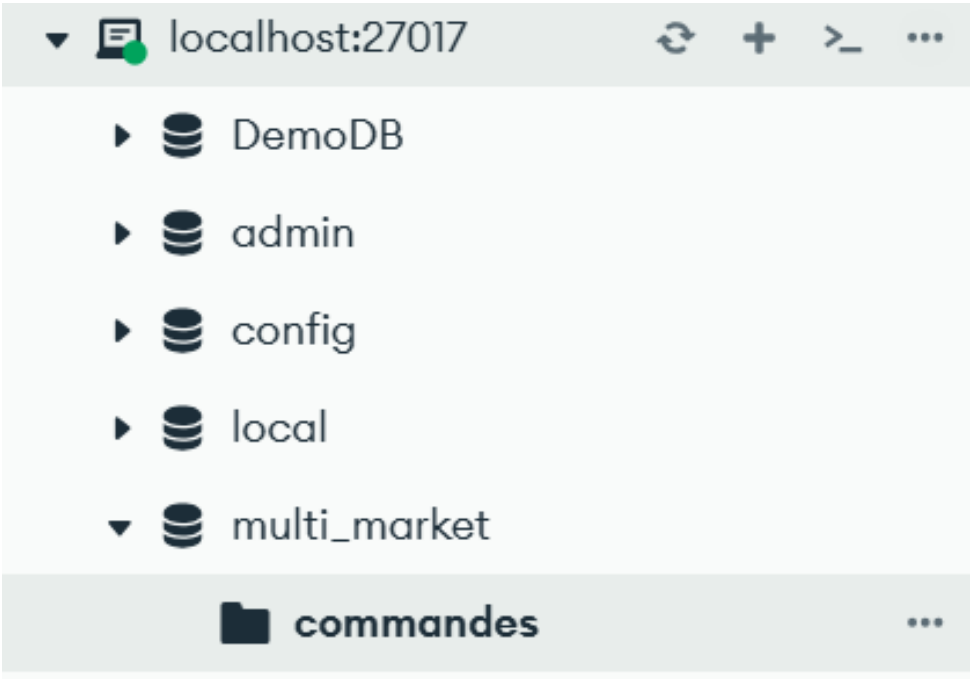

FIGURE 3.2 – l'entrepôt MongoDB

Chaque document inséré représente une commande complète, incluant les informations générales, le canal de vente, les produits commandés et le statut de la commande.

L'utilisation d'un modèle orienté documents permet de regrouper toutes les informations liées à une commande dans une seule entité.

3.5. Archivage des fichiers traités

Après l'insertion réussie dans MongoDB, les fichiers JSON traités sont déplacés vers un répertoire d'archivage. Cette étape permet d'éviter le retraitement des mêmes fichiers et de conserver une trace des données sources pour d'éventuelles vérifications ultérieures.

L'archivage contribue également à une meilleure organisation du système et à une séparation claire entre les données à traiter et celles déjà intégrées.

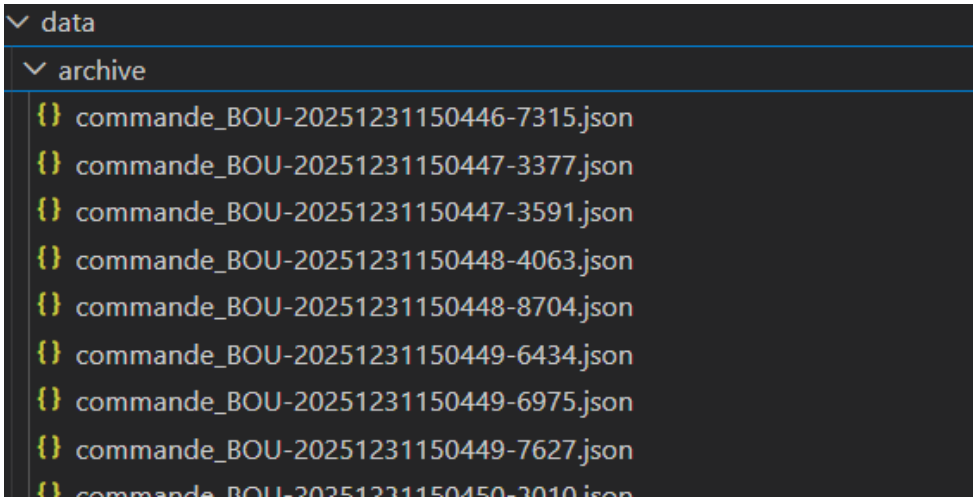

FIGURE 3.3 – Archivage des fichiers traités

3.6. Rôle du processus d'intégration dans le Data Warehouse NoSQL

Le processus d'intégration constitue le lien entre la production des données et leur exploitation décisionnelle. Il assure la continuité du flux de données, la qualité des informations stockées et la centralisation des commandes dans un entrepôt unique.

Grâce à ce mécanisme, MongoDB joue pleinement son rôle de Data Warehouse NoSQL, capable d'ingérer des données hétérogènes en continu et de les rendre disponibles pour les analyses décisionnelles présentées dans les chapitres suivants.

4. MODÉLISATION DES DONNÉES DANS MONGODB

4.1. Objectif de la modélisation NoSQL

La modélisation des données dans un système NoSQL vise à organiser les informations de manière efficace tout en profitant de la flexibilité offerte par l'absence de schéma strict. Contrairement aux bases de données relationnelles, où la modélisation repose sur des tables normalisées et des relations, la modélisation NoSQL est orientée vers l'usage et les besoins d'accès aux données.

Dans ce projet, l'objectif principal de la modélisation est de représenter une commande client comme une entité unique, facilement exploitable pour les analyses décisionnelles.

4.2. Approche orientée documents

MongoDB adopte une approche orientée documents, où chaque document regroupe l'ensemble des informations liées à une même entité. Dans le cas de ce projet, une commande est stockée sous forme d'un document unique dans la collection `commandes`.

Cette approche permet :

— d'éviter les jointures complexes,

— de simplifier les requêtes d'analyse,

— d'améliorer les performances de lecture.

Les informations relatives aux produits commandés sont intégrées directement dans le document de la commande sous forme de documents imbriqués.

4.3. Structure du document « commande »

Un exemple simplifié de document est présenté ci-dessous :

Listing 4.1 – Exemple de structure d'une commande

```
{
  "id_commande": "WEB-20251231152635-2907",
  "canal": "site_web",
  "date_commande": "2025-12-31T15:26:35.235324",
  "client": {
```

```
    "nom": "Th ophile de la Lef vre",
    "email": "tmartins@example.com",
    "telephone": "05 86 42 23 57"
  },
  "produits": [
    {
      "nom_produit": "iPhone 15 Pro",
      "quantite": 3,
      "prix_unitaire": 1100.0,
      "prix_total": 3300.0
    },
    {
      "nom_produit": "iPad Air",
      "quantite": 3,
      "prix_unitaire": 650.0,
      "prix_total": 1950.0
    },
    {
      "nom_produit": "Laptop Dell XPS",
      "quantite": 2,
      "prix_unitaire": 1200.0,
      "prix_total": 2400.0
    }
  ],
  "montant_total": 7650.0,
  "statut": "confirm e",
  "mode_paiement": "virement"
}
```

4.4. Gestion de la diversité des sources

Les commandes provenant des différents canaux ne possèdent pas exactement la même structure. Par exemple, les commandes issues du site web et de l'application mobile contiennent une adresse de livraison, tandis que celles provenant des boutiques physiques n'en possèdent pas.

Grâce à la flexibilité de MongoDB, ces différences sont gérées naturellement sans imposer de structure rigide. Les champs absents dans certains documents n'affectent pas le fonctionnement global de la base de données.

4.5. Comparaison avec la modélisation relationnelle

Dans une approche relationnelle, la modélisation d'une commande nécessiterait plusieurs tables (commande, produit, ligne de commande) reliées par des clés étrangères. En revanche, MongoDB permet de regrouper toutes ces informations dans un seul document.

Cette différence réduit la complexité du modèle et rend les requêtes d'analyse plus simples et plus performantes, ce qui est particulièrement adapté aux systèmes décisionnels et aux environnements Big Data.

Modèle Relationnel	**Modèle NoSQL (MongoDB)**
Tables normalisées	Document unique
Jointures nécessaires	Pas de jointure
Schéma rigide	Schéma flexible
Clés étrangères	Documents imbriqués
Complexité accrue	Simplicité

TABLE 4.1 – Comparaison entre modèle relationnel et NoSQL

4.6. Avantages du modèle adopté

Le modèle de données adopté dans ce projet présente plusieurs avantages :

— Simplicité de la structure

— Flexibilité face à l'évolution des données

— Facilité d'intégration des nouvelles sources

— Efficacité pour les opérations d'agrégation et d'analyse

Ainsi, la modélisation choisie permet à MongoDB de jouer efficacement le rôle d'un entrepôt de données NoSQL, capable de supporter les besoins analytiques du projet.

5. ANALYSE DÉCISIONNELLE ET AGRÉGATIONS MONGODB

5.1. Objectifs de l'analyse décisionnelle

L'analyse décisionnelle a pour objectif de transformer les données brutes stockées dans l'entrepôt de données en informations synthétiques et exploitables pour la prise de décision. Elle permet aux décideurs d'évaluer la performance de l'activité commerciale et d'identifier des tendances significatives.

Dans ce projet, les données de commandes stockées dans MongoDB sont analysées afin de produire des indicateurs clés de performance (KPI), tels que le chiffre d'affaires, les produits les plus vendus ou encore le taux d'annulation des commandes.

MongoDB propose un framework d'agrégation puissant permettant de réaliser ces analyses directement au niveau de la base de données, sans recourir à des outils externes.

5.2. Pipeline d'agrégation MongoDB

Les analyses décisionnelles sont réalisées à l'aide du framework d'agrégation de MongoDB, qui repose sur le principe de pipeline. Un pipeline est une succession d'étapes, où chaque étape transforme les données avant de les transmettre à la suivante.

Les principaux opérateurs utilisés dans ce projet sont :

— `$match` : filtre les documents selon des critères spécifiques,

— `$group` : regroupe les documents et applique des fonctions d'agrégation,

— `$project` : sélectionne et transforme les champs à afficher,

— `$sort` : ordonne les résultats.

Ces opérateurs permettent de reproduire des opérations similaires aux clauses `WHERE`, `GROUP BY`, `SELECT` et `ORDER BY` dans les bases de données relationnelles.

5.3. Chiffre d'affaires par mois et par canal

Cette analyse vise à calculer le chiffre d'affaires généré chaque mois par chaque canal de vente (web, mobile et boutique). Elle repose uniquement sur les commandes validées, c'est-à-dire celles dont le statut est `confirmée`.

Le chiffre d'affaires est calculé à partir des produits commandés, en multipliant la quantité par le prix unitaire, puis en regroupant les résultats par mois et par canal.

Listing 5.1 – Pipeline d'agrégation : chiffre d'affaires total par mois et par canal

```
[
  { "$match": { "statut": "confirme" } },
  {
    "$addFields": {
      "mois": { "$month": { "$toDate": "$date_commande" } }
    }
  },
  {
    "$group": {
      "_id": {
        "canal": "$canal",
        "mois": "$mois"
      },
      "chiffre_affaires": { "$sum": "$montant_total" }
    }
  },
  {
    "$project": {
      "_id": 0,
      "canal": "$_id.canal",
      "mois": "$_id.mois",
      "chiffre_affaires": 1
    }
  },
  {
    "$sort": {
      "mois": 1,
      "canal": 1
    }
  }
]
```

Cette analyse permet :

— d'identifier les canaux les plus performants,

— de suivre l'évolution des ventes dans le temps,

— d'aider à la prise de décisions stratégiques.

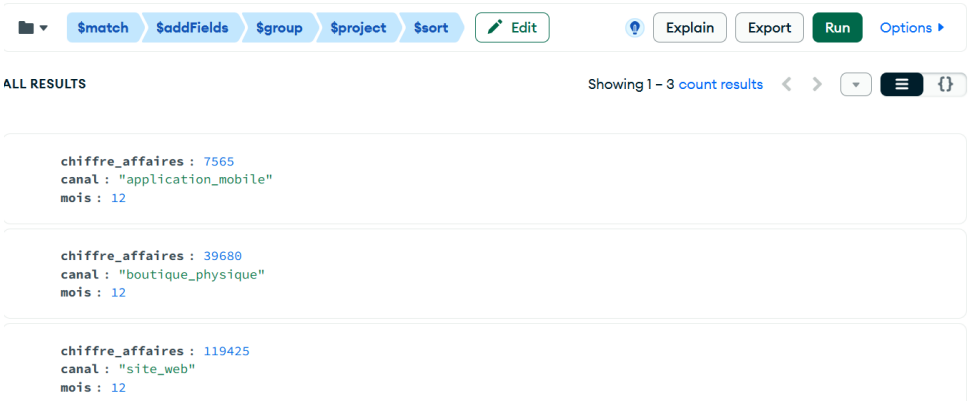

FIGURE 5.1 – Resultat de Pipeline

5.4. Top 10 des produits les plus vendus

L'objectif de cette analyse est d'identifier les produits les plus vendus en termes de quantité. Les données sont regroupées par nom de produit, puis triées par quantité totale vendue afin de sélectionner les dix produits les plus populaires.

Listing 5.2 – Pipeline d'agrégation : top 10 des produits les plus vendus en quantité

```
[
  { "$match": { "statut": "confirm e" } },
  { "$unwind": "$produits" },
  {
    "$group": {
      "_id": "$produits.nom_produit",
      "quantite_totale": {
        "$sum": "$produits.quantite"
      }
    }
  },
  {
    "$project": {
      "_id": 0,
      "produit": "$_id",
      "quantite_totale": 1
    }
  },
  {
    "$sort": {
      "quantite_totale": -1
    }
```

```
  },
  {
    "$limit": 10
  }

]
```

Cette analyse permet :

— de comprendre les préférences des clients,

— d'optimiser la gestion des stocks,

— d'orienter les actions marketing.

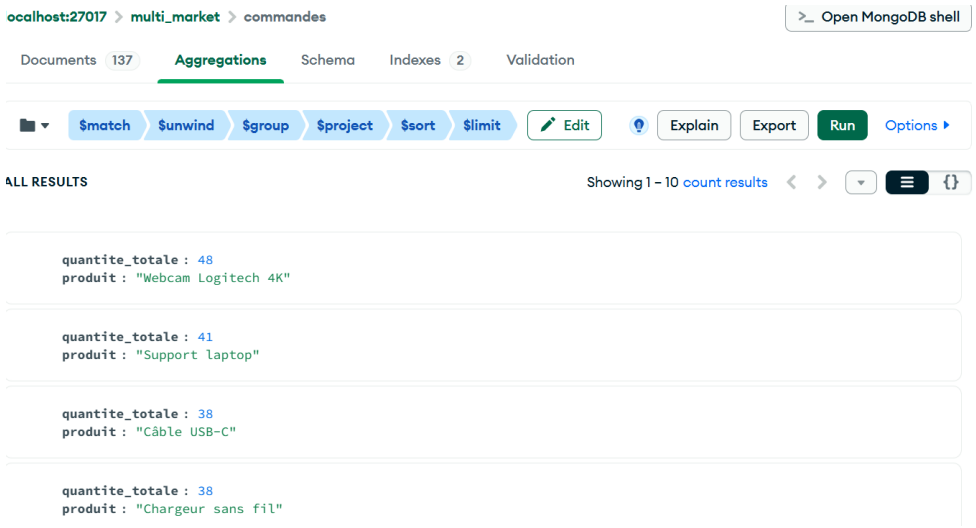

FIGURE 5.2 – Resultat de Pipeline

5.5. Taux de commandes annulées par canal

Le taux de commandes annulées est un indicateur important pour évaluer la qualité du service proposé par chaque canal de vente.

Listing 5.3 – Pipeline d'agrégation : top 10 des produits les plus vendus en quantité

```
[
  {
    "$group": {
      "_id": "$canal",
      "total_commandes": { "$sum": 1 },
      "commandes_annulees": {
        "$sum": {
          "$cond": [
            { "$eq": ["$statut", "annul e"] },
```

```
            1,
            0
          ]
        }
      }
    }
  },
  {
    "$addFields": {
      "taux_annulation": {
        "$multiply": [
          {
            "$divide": ["$commandes_annulees", "$total_commandes"]
          },
          100
        ]
      }
    }
  },
  {
    "$project": {
      "_id": 0,
      "canal": "$_id",
      "total_commandes": 1,
      "commandes_annulees": 1,
      "taux_annulation": 1
    }
  },
  {
    "$sort": {
      "taux_annulation": -1
    }
  }
]
```

Il est calculé en comparant le nombre de commandes annulées au nombre total de commandes pour chaque canal. Cette analyse permet d'identifier d'éventuels problèmes opérationnels ou logistiques liés à un canal spécifique.

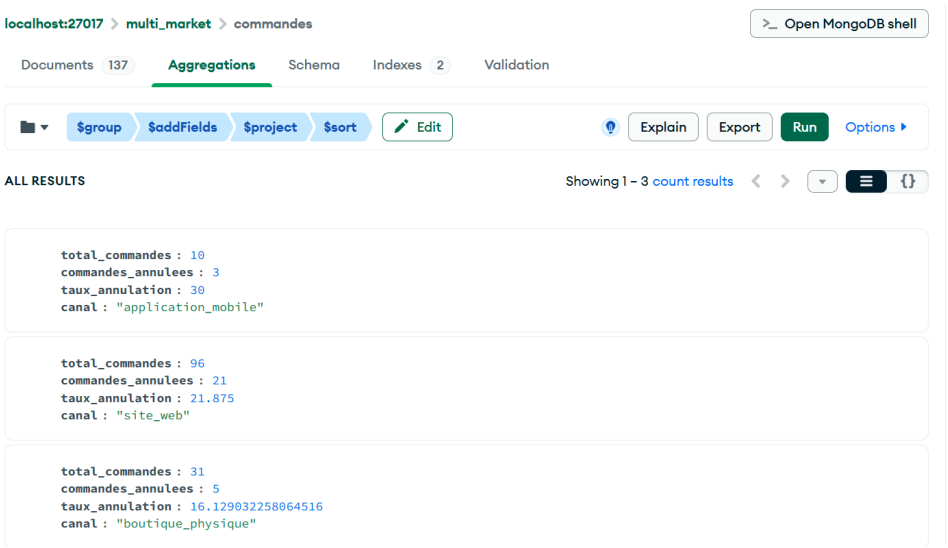

FIGURE 5.3 – Resultat de Pipeline

5.6. Chiffre d'affaires moyen par commande

Cette analyse consiste à calculer le chiffre d'affaires moyen généré par commande pour chaque canal de vente. Elle est obtenue en divisant le chiffre d'affaires total par le nombre de commandes validées.

Listing 5.4 – Pipeline d'agrégation : top 10 des produits les plus vendus en quantité

```
[
  {
    "$match": {
      "statut": "confirm e"
    }
  },
  {
    "$group": {
      "_id": "$canal",
      "total_ca": { "$sum": "$montant_total" },
      "nombre_commandes": { "$sum": 1 }
    }
  },
  {
    "$addFields": {
      "ca_moyen": {
        "$divide": ["$total_ca", "$nombre_commandes"]
      }
    }
```

```
  },
  {
    "$project": {
      "_id": 0,
      "canal": "$_id",
      "ca_moyen": 1,
      "total_ca": 1,
      "nombre_commandes": 1
    }
  }
]
```

Cet indicateur permet :

— d'évaluer la valeur moyenne des commandes,

— de comparer le comportement d'achat des clients selon le canal utilisé,

— d'identifier les canaux générant les commandes les plus rentables.

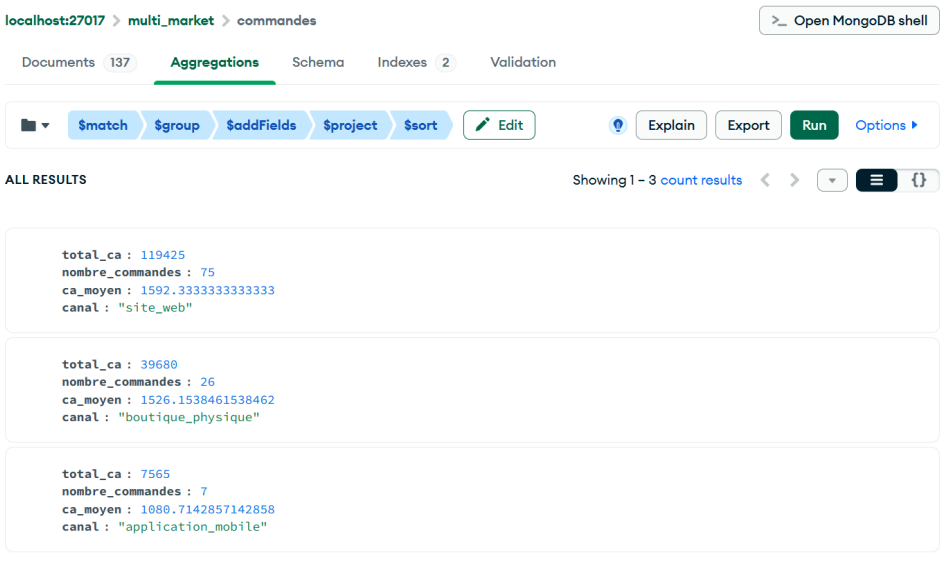

FIGURE 5.4 – Resultat de Pipeline

5.7. Importance des analyses décisionnelles

Les analyses réalisées dans ce chapitre constituent le socle du système décisionnel du projet. Elles permettent de synthétiser de grandes quantités de données et de fournir une vision globale et claire de l'activité commerciale.

Grâce aux capacités d'agrégation offertes par MongoDB, ces analyses sont réalisées de manière performante et flexible. Cela confirme l'adéquation de MongoDB pour la mise en place d'un entrepôt de données NoSQL orienté Big Data et analyse décisionnelle.

6. REPORTING ET VISUALISATION DES DONNÉES

6.1. Objectifs du reporting

Le reporting constitue la dernière étape du processus décisionnel. Il vise à présenter les résultats des analyses sous une forme visuelle claire, synthétique et facilement compréhensible, afin de faciliter l'interprétation des données et d'aider les décideurs à prendre des décisions stratégiques.

Dans ce projet, le reporting permet de transformer les indicateurs calculés à partir des données stockées dans MongoDB en tableaux de bord interactifs, offrant une vision globale de l'activité commerciale de l'entreprise.

6.2. Outils de visualisation utilisés

Deux outils de visualisation ont été utilisés pour la conception des tableaux de bord :

— **MongoDB Charts** : outil natif de MongoDB permettant de créer rapidement des graphiques directement à partir des collections de données, sans nécessiter de transformation supplémentaire.

— **Power BI** : outil de Business Intelligence offrant des fonctionnalités avancées de visualisation, d'interaction et de filtrage des données.

L'utilisation combinée de ces outils permet de répondre efficacement aux besoins de visualisation tout en assurant une bonne lisibilité et une exploitation optimale des résultats.

6.3. Tableaux de bord réalisés

Les tableaux de bord réalisés dans le cadre de ce projet présentent les principaux indicateurs de performance issus des analyses décisionnelles :

— Chiffre d'affaires total par mois et par canal de vente,

— Top 10 des produits les plus vendus en termes de quantité,

— Taux de commandes annulées par canal,

— Chiffre d'affaires moyen par commande et par canal.

Ces tableaux de bord permettent de comparer les performances des différents canaux de vente et de mettre en évidence les tendances majeures de l'activité commerciale.

FIGURE 6.1 – Dashboard Avec MongoDB Charts

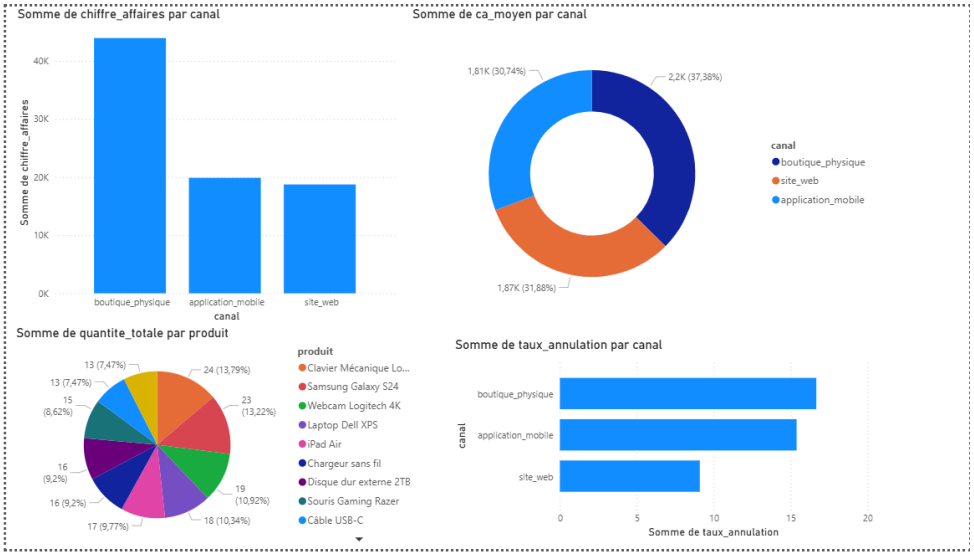

FIGURE 6.2 – Dashboard Avec Power BI

6.4. Apport du reporting à la prise de décision

Grâce aux tableaux de bord réalisés, les décideurs disposent d'une vue synthétique et interactive de l'activité commerciale. Le reporting permet d'analyser rapidement les

performances, de détecter d'éventuelles anomalies et d'ajuster les stratégies commerciales en conséquence.

Cette étape transforme les données brutes en informations décisionnelles à forte valeur ajoutée, renforçant ainsi la pertinence du système décisionnel mis en place.

6.5. Conclusion du processus de reporting

Le reporting et la visualisation des données constituent l'aboutissement du projet. Ils permettent de valoriser l'ensemble du travail réalisé, depuis la simulation et l'intégration des données jusqu'à l'analyse décisionnelle.

Cette dernière étape confirme l'efficacité de MongoDB comme solution de Big Data Warehouse NoSQL et souligne l'importance de la visualisation dans l'exploitation des données décisionnelles.

CONCLUSION

Ce projet a permis de mettre en œuvre un système complet de gestion des commandes multi-canaux basé sur MongoDB, en suivant une approche orientée Big Data et analyse décisionnelle. À travers la simulation des données, leur intégration et leur exploitation, les différentes étapes d'un Data Warehouse NoSQL ont été abordées de manière concrète et progressive.

L'utilisation de MongoDB comme entrepôt de données a démontré les avantages du modèle orienté documents, notamment en termes de flexibilité, de simplicité de modélisation et de performance pour les opérations analytiques. La gestion des données provenant de plusieurs canaux (site web, application mobile et boutique physique) a été facilitée grâce à l'absence de schéma rigide.

Les analyses décisionnelles réalisées à l'aide du framework d'agrégation de MongoDB ont permis de produire des indicateurs clés tels que le chiffre d'affaires, les ventes par canal, les produits les plus vendus et le taux de commandes annulées. Ces résultats ont ensuite été valorisés à travers des tableaux de bord clairs et interactifs grâce aux outils de reporting MongoDB Charts et Power BI.

En conclusion, ce mini-projet confirme que MongoDB constitue une solution efficace pour la mise en place d'un Data Warehouse NoSQL dans un contexte multi-canaux. Il met également en évidence l'importance du reporting et de la visualisation des données dans le processus décisionnel, en transformant les données brutes en informations utiles à forte valeur ajoutée.