Livrable 2 : Migration MongoDB

Phase 2 - Bases de Données Avancées

Projet CinéExplorer

Plateforme Web de Découverte de Films

Étudiant : SAHNOUN SALAH EDDINE
Groupe : FISE 4A
Encadrants : AL-KHARAZ M. et HADDOU BEN DERBAL H.
Année universitaire : 2025-2026

Aix-Marseille Université - Polytech Marseille
Département Informatique

26 décembre 2025

Table des matières

Introduction

Ce livrable présente les résultats de la Phase 2 du projet CinéExplorer, consacrée à la migration des données IMDB vers MongoDB. L'objectif était de maîtriser les différences entre les approches SQL et NoSQL, d'effectuer une migration complète des données, et de comparer les performances des deux technologies.

Objectifs de la Phase 2

— Comprendre les différences fondamentales entre SQL et NoSQL
— Maîtriser la migration de données relationnelles vers MongoDB
— Utiliser le pipeline d'agrégation MongoDB
— Comparer les performances des deux approches
— Implémenter des documents structurés dénormalisés

Configuration technique

— **MongoDB :** Version 6.0.27 Community Edition
— **PyMongo :** Version 4.6.1
— **SQLite :** Version 3.37.2 (intégré à Python)
— **Python :** Version 3.10+
— **Dataset :** IMDB small (36,859 films, 2.3M documents)

T2.1 : Installation et configuration

Résultats de la connexion

La connexion à MongoDB a été établie avec succès :

— Version serveur : 6.0.27
— Connexion : localhost :27017
— Base de données : imdb_flat accessible

Configuration réussie

Composant	Version	Statut
MongoDB Community Edition	6.0.27	OK
PyMongo	4.6.1	OK
Connexion serveur	localhost :27017	OK
Base de données	imdb_flat	OK

Table 1 – Configuration MongoDB

T2.2 : Migration des collections plates

Résultats de la migration

Collection	Documents	Statut
characters	152,877	OK
directors	40,933	OK
genres	85,426	OK
knownformovies	252,924	OK
movies	36,859	OK
persons	145,847	OK
principals	361,858	OK
professions	314,779	OK
ratings	36,859	OK
titles	775,705	OK
writers	82,855	OK
TOTAL	**2,286,922**	**OK 100%**

Table 2 – Résultats de la migration

Performances de migration

— **Temps total :** 22.44 secondes
— **Débit moyen :** 102,000 documents/seconde
— **Tables migrées :** 11/11 (100% de succès)
— **Intégrité :** Tous les comptages correspondent parfaitement

T2.3 : Requêtes MongoDB équivalentes

Résultats comparatifs détaillés

#	Description	MongoDB (ms)	SQLite (ms)	Ratio	Plus rapide
Q1	Filmographie Tom Hanks	7,794.76	319.73	24.4	SQLite 24×
Q2	Top 5 Drama 1990-2000	369.42	41.63	8.9	SQLite 9×
Q3	Acteurs multi-rôles	1,035.57	854.58	1.2	Similaire
Q4	Collaborations acteur-réalisateur	7,256.56	1,487.31	4.9	SQLite 5×
Q5	Genres populaires	501.78	161.84	3.1	SQLite 3×
Q6	Évolution carrière Tom Hanks	**16.75**	920.57	0.02	MongoDB 55×
Q7	Top 3 par genre	9,275.97	417.97	22.2	SQLite 22×
Q8	Percée grâce à un film	**585.27**	861.00	0.7	MongoDB 1.5×
Q9	Acteur-Réalisateur	9,720.25	858.48	11.3	SQLite 11×

TABLE 3 – Comparaison détaillée des performances

Statistiques globales

Métrique	Valeur
Requêtes MongoDB réussies	9/9 (100%)
Temps moyen MongoDB	4,062 ms (4.06 s)
Temps moyen SQLite	658 ms (0.66 s)
Rapport SQLite/MongoDB	0.16
SQLite plus rapide en moyenne	**6.2×**
Victoires MongoDB	**2 requêtes** (Q6, Q8)
Victoires SQLite	**6 requêtes** (Q1, Q2, Q4, Q5, Q7, Q9)
Égalités	**1 requête** (Q3)

TABLE 4 – Statistiques globales de performance

Analyse des résultats

Points forts de MongoDB

— **Agrégations natives :** Excellentes performances pour les opérations d'agrégation (Q6 : 55× plus rapide)
— **Flexibilité du pipeline :** Capacité à transformer les données de manière complexe
— **Schéma flexible :** Adaptabilité aux données semi-structurées

Limitations identifiées

— **Coût des jointures :** Les opérations de jointure multiples impactent significativement les performances
— **Complexité des pipelines :** Difficulté de débogage et maintenance
— **Performances variables :** Dépendance forte à l'indexation et structure des données

T2.4 : Documents structurés

Modèle de document structuré

Le document structuré suit exactement le schéma demandé dans la consigne :

— `_id` : Identifiant du film
— `title`, `year`, `runtime` : Informations de base
— `genres` : Tableau des genres
— `rating` : Objet avec moyenne et nombre de votes
— `directors` : Tableau des réalisateurs
— `cast` : Tableau du casting avec personnages et ordre
— `writers` : Tableau des scénaristes avec catégorie
— `titles` : Tableau des titres alternatifs par région

Résultats de la création

Métrique	Valeur
Temps de création	45.00 secondes
Documents créés	36,859
Index source créés	19
Index cible créés	8

TABLE 5 – Résultats de la création de movies_complete

Comparaisons demandées

1. Temps pour récupérer un film complet

Approche	**Temps (ms)**	**Nombre requêtes**
Collections plates	4.879 ms	N requêtes (15)
Documents structurés	**0.724 ms**	**1 requête**
Gain	**6.74× plus rapide**	**Réduction de 93%**

TABLE 6 – Comparaison temps de récupération

2. Taille de stockage

Approche	**Taille (MB)**
Collections plates (somme)	60.10 MB
Collection structurée	**17.48 MB**
Réduction	**71% d'économie**

TABLE 7 – Comparaison taille de stockage

3. Complexité du code

Collections plates	**Documents structurés**
+ Normalisation des données	+ 1 requête pour toutes les données
+ Mises à jour simplifiées	+ Données prêtes à l'emploi
+ Flexibilité des requêtes	+ Meilleures performances lecture
− Multiples requêtes nécessaires	− Redondance des données
− Assemblage côté application	− Mises à jour complexes
− Risque d'incohérence	− Taille document importante

TABLE 8 – Comparaison complexité du code

Analyse comparative et recommandations

Avantages et inconvénients

SQLite (SQL)	MongoDB (NoSQL)
+ Jointures relationnelles performantes	+ Scalabilité horizontale
+ Requêtes SQL standardisées	+ Flexibilité du schéma
+ Cache mémoire efficace	+ Performances agrégation native
+ Transactionnalité ACID	+ Documents structurés pour lecture
− Limité à un seul fichier	− Coût élevé des jointures
− Pas de scalabilité horizontale	− Pipeline d'agrégation complexe
− Schéma rigide	− Cohérence éventuelle

Table 9 – Comparaison SQLite vs MongoDB

Justification technique

Pourquoi cette architecture hybride ?

1. **Performance :** Exploiter les points forts de chaque technologie
   — SQLite excelle en jointures → navigation/filtrage
   — MongoDB excelle en agrégations → détails/analyses
2. **Maintenance :** Séparation des responsabilités
   — SQLite : données transactionnelles normalisées
   — MongoDB : données analytiques dénormalisées
3. **Évolutivité :** Préparation pour la Phase 3
   — MongoDB prêt pour Replica Set (haute disponibilité)
   — SQLite reste pour les fonctionnalités basiques
4. **Cohérence avec les résultats :**
   — MongoDB 6.74× plus rapide pour la lecture de documents complets
   — SQLite 6.2× plus rapide en moyenne pour les requêtes relationnelles
   — Chaque technologie utilisée là où elle excelle

Conclusion de la Phase 2

Bilan des objectifs atteints

Objectif	Réalisation	Statut
Comprendre SQL vs NoSQL	Analyse comparative complète	Atteints
Maîtriser la migration	2.3M documents migrés en 22s	Atteints
Utiliser pipeline d'agrégation	9 requêtes MongoDB implémentées	Atteints
Comparer performances	Benchmark complet réalisé	Atteints
Documents structurés	Collection movies_complete créée	Atteints

TABLE 10 – Bilan des objectifs atteints

Principaux enseignements

1. **MongoDB excelle** pour les agrégations natives et les documents structurés
2. **SQLite reste supérieur** pour les jointures relationnelles complexes
3. **La dénormalisation** offre des gains significatifs en lecture (6.74×) et stockage (71%)
4. **L'architecture hybride** est optimale pour CinéExplorer
5. **L'indexation** est cruciale pour les performances MongoDB

Annexes

Résultats d'exécution complets

Migration (T2.2)

```
DÉMARRAGE DE LA MIGRATION SQLite → MongoDB
Temps total: 22.44 secondes
Tables migrées: 11/11
Documents totaux: 2,286,922
```

Requêtes (T2.3)

```
Requêtes MongoDB réussies: 9/9
Temps moyen MongoDB: 4.062 secondes
Temps moyen SQLite: 0.658 secondes
SQLite 6.2x plus rapide en moyenne
```

Documents structurés (T2.4)

```
movies_complete construit en 45.00s - 36,859 documents
Structuré: 0.724 ms (1 requête)
Flat: 4.879 ms (N requêtes) - Gain: 6.74x
Stockage: 17.48 MB vs 60.10 MB - Économie: 71%
```