Avenue des Facultés
80000 Amiens

# Choix de la Technologie pour la Base de Données

## SAE Semestre 5

*Victor DA SILVA LOURENCO MARQUES, Lorenzo VATRIN, Justin LARGILLIÈRE, Liam TARGET, Malcolm WALTER, Lucas GONTHIER*

BUT3 Informatique
Tous parcours

Année
2024 - 2025

# Introduction

Dans ce document, nous allons expliquer les raisons pour lesquelles le modèle SQL a été choisi pour le projet de gestion de jeu vidéo, malgré la flexibilité offerte par une base de données NoSQL. Nous aborderons les spécificités du projet qui nécessitent une gestion stricte des relations entre les entités.

# NoSQL (MongoDB)

## Avantages

- **Flexibilité du schéma** : MongoDB permet une structure de données flexible où les documents peuvent évoluer facilement, ce qui est pratique si le projet nécessite des ajustements fréquents des entités (par exemple, ajouter de nouvelles propriétés aux joueurs ou aux items).

- **Gestion des données imbriquées** : Les données imbriquées, comme les items d'un joueur ou les ressources d'un craft, peuvent être stockées directement dans le même document, évitant ainsi les jointures coûteuses.

- **Performance en lecture/écriture** : MongoDB est optimisé pour des opérations fréquentes de lecture et d'écriture, car toutes les données nécessaires peuvent être rassemblées dans un seul document.

- **Évolutivité horizontale** : MongoDB permet un sharding natif, ce qui facilite la gestion de grandes volumes de données et permet de distribuer la charge entre plusieurs serveurs.

## Inconvénients

- **Pas de transactions complexes** : MongoDB n'offre pas la même rigueur que SQL pour gérer des transactions multi-documents complexes, ce qui peut être un inconvénient si des opérations nécessitent une forte cohérence des données.

- **Moins strict sur l'intégrité des données** : Il est plus facile d'avoir des données inconsistantes si les relations ne sont pas bien gérées, car MongoDB ne force pas les contraintes entre les collections (comme les clés étrangères en SQL).

- **Coût de la duplication des données** : Les documents imbriqués peuvent entraîner une duplication de certaines données, ce qui peut augmenter le volume de stockage nécessaire.

# SQL

## Avantages

- **Modèle relationnel robuste** : SQL excelle dans la gestion des relations complexes entre entités, grâce à des clés étrangères et des jointures, ce qui garantit une forte cohérence des données.

- **Transactions ACID** : SQL offre des garanties transactionnelles fortes, avec le support des propriétés ACID (atomicité, cohérence, isolation, durabilité), ce qui est utile pour des opérations complexes qui nécessitent une cohérence totale.

- **Contrôle de l'intégrité des données** : SQL impose des contraintes strictes (clés primaires, étrangères, etc.) qui garantissent l'intégrité des relations entre les tables, ce qui évite les incohérences.

- **Requêtes complexes optimisées** : Les bases SQL sont très efficaces pour exécuter des requêtes complexes avec des jointures entre plusieurs tables, ce qui est essentiel si les données doivent être interrogées de manière variée.

## Inconvénients

- **Rigidité du schéma** : Tout changement dans la structure des données nécessite des modifications du schéma qui peuvent être lourdes, surtout si de nombreuses entités sont affectées.

- **Performances réduites sur les grandes échelle**s : Avec de nombreuses jointures et des requêtes complexes, les performances peuvent chuter lorsque la taille de la base de données augmente.

- **Scalabilité verticale** : Les bases SQL se scalent principalement verticalement (en augmentant la puissance du serveur), ce qui peut être limitant pour les applications à grande échelle.

# Conclusion

Le choix entre NoSQL (MongoDB) et SQL dépend des besoins spécifiques de notre projet. Pour un projet de gestion de jeu vidéo avec des entités complexes et des relations dynamiques entre les joueurs, les items, les crafts et les ressources, le SQL semble être une meilleure option.

En conclusion, bien que MongoDB offre une flexibilité intéressante pour les projets évolutifs, la nature des relations complexes et interdépendantes du projet nécessite une structure plus rigide et cohérente, ce qui fait de SQL le choix le plus adapté. Avec SQL, le projet bénéficie d'une gestion stricte des relations, d'une intégrité des données assurée, ainsi que de transactions complexes sécurisées.