# Fiche Synthèse : Recommandations

**Recommandation Finales :**

1. **SQL (MySQL/PostgreSQL)**
   * **Pourquoi ?**
     + **Relations complexes** : Les bases SQL sont adaptées pour gérer les entités relationnelles complexes, comme les utilisateurs, les livres, et les avis.
     + **Performance pour les requêtes complexes** : Les jointures et les agrégations sur des tables relationnelles sont optimisées.
     + **Cohérence des données** : Grâce au modèle ACID, SQL garantit une haute intégrité et cohérence des données, particulièrement important pour des opérations impliquant des transactions (ajout d'avis, modification des recommandations).
     + **Limite de Scalabilité** : SQL peut devenir moins performant avec un volume massif de données, mais dans ce cas précis, la scalabilité verticale est suffisante.
2. **NoSQL (MongoDB)**
   * **Pourquoi ?**
     + **Flexibilité du schéma** : Idéal pour gérer les **avis utilisateurs**, qui peuvent varier (contenu, métadonnées). NoSQL permet d'ajouter facilement des informations supplémentaires sans perturber la structure existante.
     + **Scalabilité horizontale** : Bien que votre volume de données ne soit pas massif, MongoDB peut supporter une croissance modérée tout en étant flexible à l’échelle.
     + **Optimisation pour les lectures rapides** : MongoDB est conçu pour des écritures et lectures rapides, ce qui est important pour les systèmes de recommandation en temps réel.
     + **Base de données non relationnelle** : Bien que moins performante pour les requêtes complexes impliquant plusieurs relations, NoSQL est parfaitement adapté pour les données semi-structurées.

**Modèle Polyglotte (combinaison des deux bases)**

* **Recommandation finale** : Utilisez **SQL** pour les entités fortement structurées et leurs relations complexes (utilisateurs, livres, recommandations), et **NoSQL** pour gérer des données semi-structurées, comme les avis utilisateurs.
  + **SQL** : Gestion des relations (avis, livres, utilisateurs) avec des opérations ACID pour garantir l'intégrité.
  + **NoSQL** : Stockage flexible des avis, évolutivité horizontale et gestion efficace des données non structurées.

**Conclusion :**

En combinant SQL et NoSQL, vous bénéficiez des avantages de chaque modèle : la puissance de SQL pour les relations complexes et la flexibilité de NoSQL pour les données semi-structurées. Ce modèle polyglotte répondra aux besoins de **flexibilité**, **performance** et **scalabilité** dans le cadre de votre projet de gestion des avis et recommandations.