Compléments sur le moteur NoSQL MongoDB et d'un Moteur NoSQL de votre Choix

Cette section présente une analyse théorique et pratique de plusieurs concepts liés aux moteurs NoSQL. Nous nous focalisons principalement sur **MongoDB** et, à titre d'exemple, sur **RavenDB**, une base NoSQL orientée documents qui repose sur le modèle clé-document et fournit des fonctionnalités avancées comme les index automatiques et les requêtes LINQ.

---

1. Modèles de données supportés

MongoDB

- **Type de modèle :**
  MongoDB est une base de données orientée documents.
- **Format des données :**
  Utilise le format BSON (Binary JSON), une extension binaire de JSON, qui permet de stocker des données complexes (documents, tableaux, sous-documents, etc.).
- **Structure :**
  Les données sont organisées en collections (similaires aux tables en SQL) qui contiennent des documents (similaires aux enregistrements), où chaque document possède des champs pouvant être de types variés et possiblement imbriqués.
- **Application pratique :**
  Dans notre système, chaque entité telle que **Member**, **Publisher**, **Category** ou **Book** est stockée sous forme de document, ce qui offre une grande flexibilité dans l'évolution du schéma des données sans nécessiter de migrations lourdes.

RavenDB

- **Type de modèle :**
  RavenDB est une base de données orientée documents.
- **Format des données :**
  Utilise le format JSON pour le stockage des documents.
- **Structure :**
  Les documents sont stockés dans des collections et peuvent contenir des objets imbriqués, des tableaux, et des références à d'autres documents.
- **Application pratique :**
  Dans notre système, **Member**, **Publisher**, **Book**, etc., peuvent être modélisés comme des documents JSON simples, et enrichis avec des sous-documents (ex. contactInfo, activeLoans). RavenDB permet également de projeter des données à la volée via des index personnalisés.

---

2. Réévaluer la procédure d'installation du moteur et des utilitaires

MongoDB

- **Procédure d'installation :**
  L'installation de MongoDB repose sur des packages précompilés disponibles pour divers systèmes d'exploitation ou via des gestionnaires de packages comme `apt` ou `yum`.
- **Utilitaires :**
  - `mongod` : Le serveur MongoDB.
  - `mongo` : Le shell de commande pour interagir avec la base.
  - Des outils comme `MongoDB Compass` permettent d'effectuer des analyses visuelles et administratives.
- **Pratique dans notre système :**
  Une procédure d'installation bien documentée et automatisée (via des scripts ou une configuration Docker) garantit que tous les développeurs et systèmes de production déploient une configuration identique et optimisée.

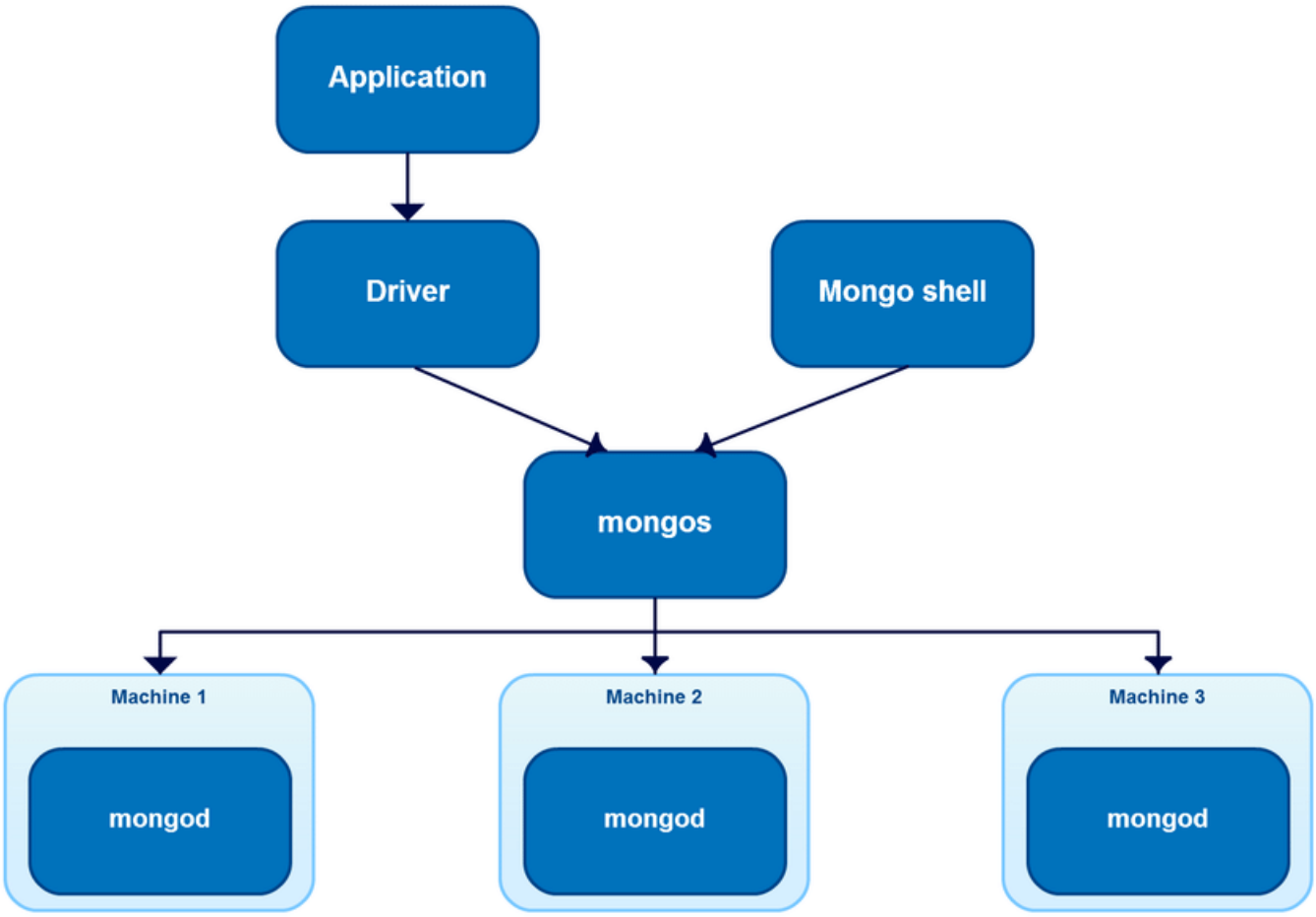
RavenDB

- **Procédure d'installation :**
  RavenDB est très facile à installer, car il est disponible en version auto-hébergée avec une interface web embarquée. Il peut être déployé localement, via Docker, ou dans le cloud.
- **Utilitaires :**
  - Interface Web intégrée : permet d'explorer, interroger, indexer, administrer.
  - Outils en ligne de commande et API REST disponibles.
- **Pratique dans notre système :**
  Grâce à son interface intégrée, le monitoring, les tests et l'administration des collections peuvent être réalisés facilement, sans outil tiers. Un serveur RavenDB peut être lancé en quelques minutes avec une configuration minimale.

---

3. Architecture du moteur NoSQL (schémas expliqués)

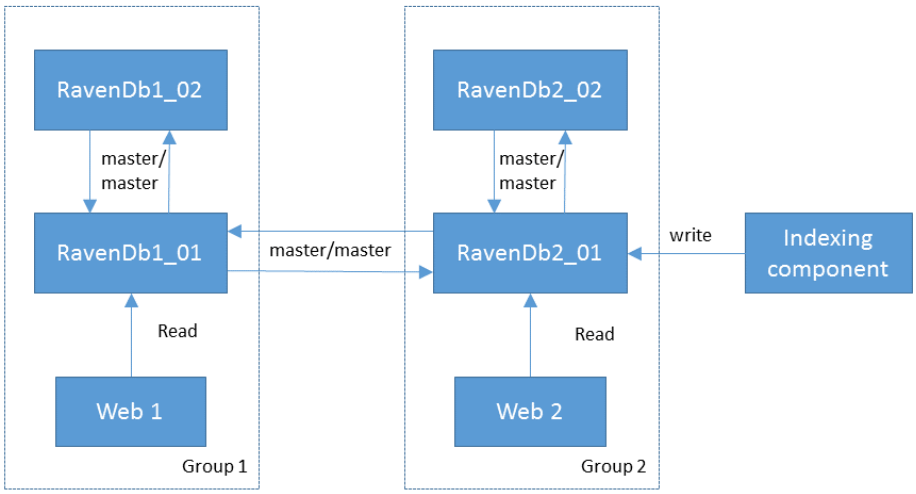
MongoDB



Cette architecture montre une application connectée à un mongos (routeur MongoDB), qui redirige les requêtes vers différents mongod (instances de données) répartis sur plusieurs machines. Le mongos agit comme une interface unique pour le client, simplifiant l'accès aux données distribuées sur plusieurs nœuds du cluster.

- **Architecture générale :**
  MongoDB utilise une architecture maître-esclave (ou réplication multi-maître) dans des configurations de réplication, et supporte le sharding pour la partition des données.
- **Schéma explicatif (conceptuel) :**
  - **Instance de MongoDB (mongod) :** Chaque instance gère des collections et des documents.
  - **Replica Set :** Un ensemble d'instances répliquant les mêmes données pour la haute disponibilité.
  - **Shard Cluster :** Les données sont partitionnées (shardées) sur plusieurs nœuds, chaque shard pouvant être un replica set.
- **Application pratique dans notre système :**
  En production, notre système pourrait déployer un cluster shardé pour répartir la charge, avec des replica sets assurant la redondance et la tolérance aux pannes.

RavenDB



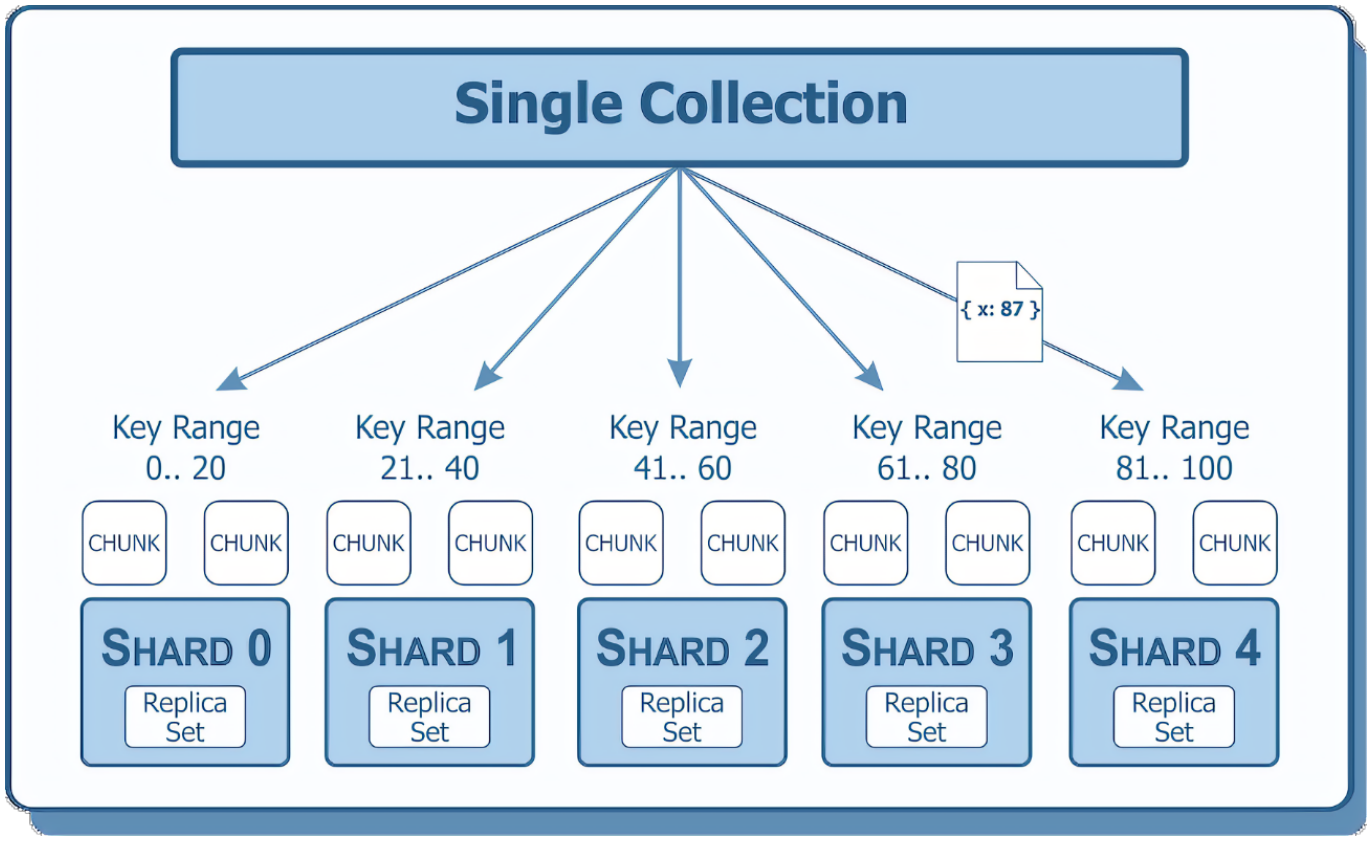
RavenDB suit une architecture maître-maître distribuée avec des nœuds pouvant répliquer entre eux de manière bidirectionnelle. Chaque nœud peut accepter des lectures et des écritures. Une base de données peut être partagée entre plusieurs nœuds du cluster.

- **Architecture générale :**
  RavenDB peut être utilisé en mode autonome ou en cluster. En cluster, chaque base de données peut être répliquée sur plusieurs nœuds.
- **Schéma conceptuel :**
  - **Cluster :** Composé de plusieurs nœuds capables de se synchroniser.
  - **Base de données distribuée :** Répliquée automatiquement sur plusieurs serveurs.
- **Application pratique dans notre système :**
  L'architecture distribuée permet d'améliorer la résilience de l'application et la disponibilité des données sans modifier le code applicatif.

---

4. Méthode de partitionnement (schémas expliqués)

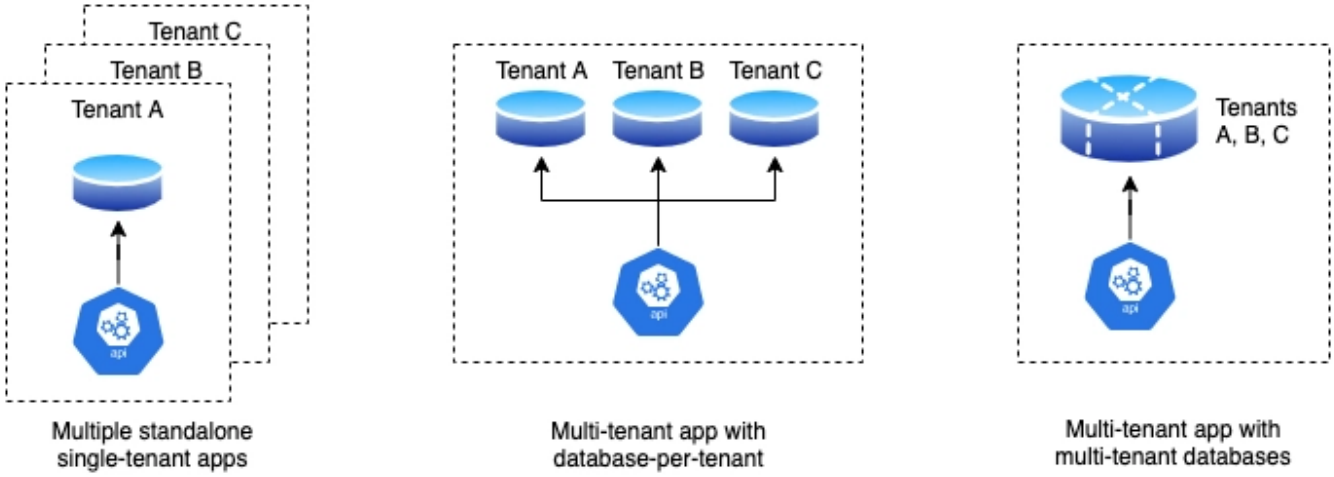
MongoDB



MongoDB utilise un partitionnement appelé sharding où chaque fragment (shard) contient une portion des données. Le mongos est responsable de router les requêtes vers le bon shard. Les documents sont distribués en fonction d'une clé de partition définie par l'utilisateur (shard key).

- **Partitionnement (Sharding) :**
  - **Principe :** Les données sont réparties sur plusieurs shards en utilisant une clé de sharding.
  - **Schéma conceptuel :**
    - **Shard Key :** Un champ ou une combinaison de champs sélectionné pour partitionner les documents.
    - **Mongos :** Le routeur qui distribue les requêtes vers les shards correspondants.
- **Application pratique dans notre système :**
  En choisissant une shard key judicieuse (par exemple, par région géographique pour des **Members** ou par catégories pour des **Books**), nous pouvons garantir une distribution équilibrée des données et améliorer les performances des requêtes.

RavenDB



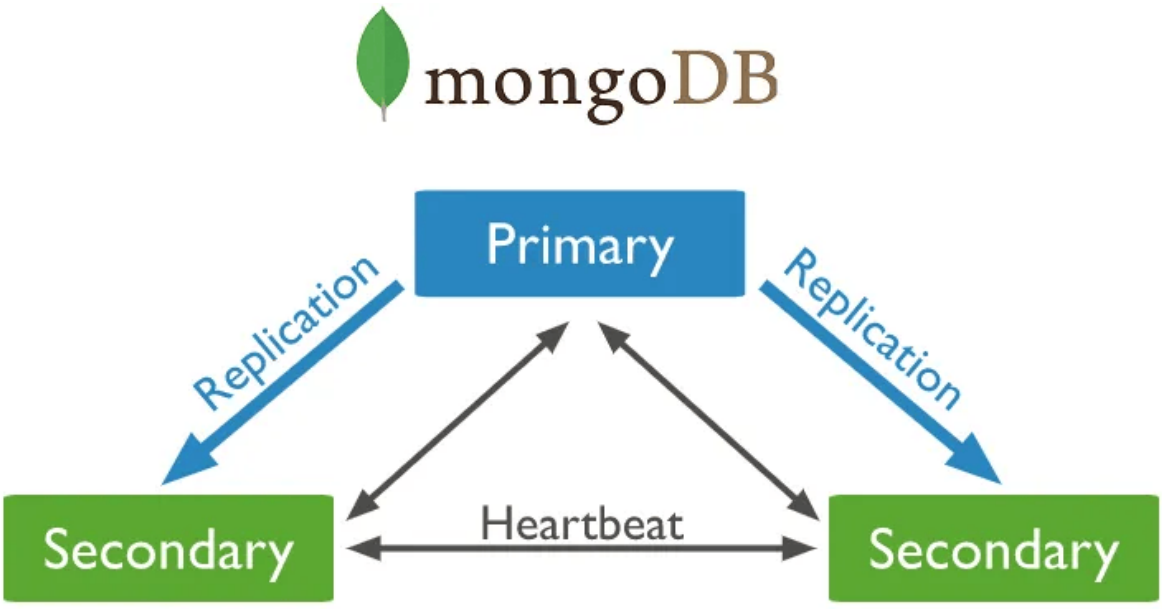
RavenDB ne propose pas de partitionnement automatique comme MongoDB, mais supporte le **multi-tenant sharding** via le concept de **bases de données multiples**. Chaque collection ou client peut être isolé dans une base de données dédiée.

- **Partitionnement :**
  - **Principe :** L'administrateur peut répartir les données manuellement ou par convention (ex. une base par région ou par client).
  - **Schéma conceptuel :**
    - Plusieurs bases de données logiques réparties sur des nœuds physiques différents.
- **Application pratique dans notre système :**
  Pour les très grandes volumétries, on peut affecter des bases différentes à des groupes de **Members**, ou stocker des livres de différentes langues dans des bases séparées pour équilibrer la charge.

---

5. Méthode de réplication (schémas expliqués)

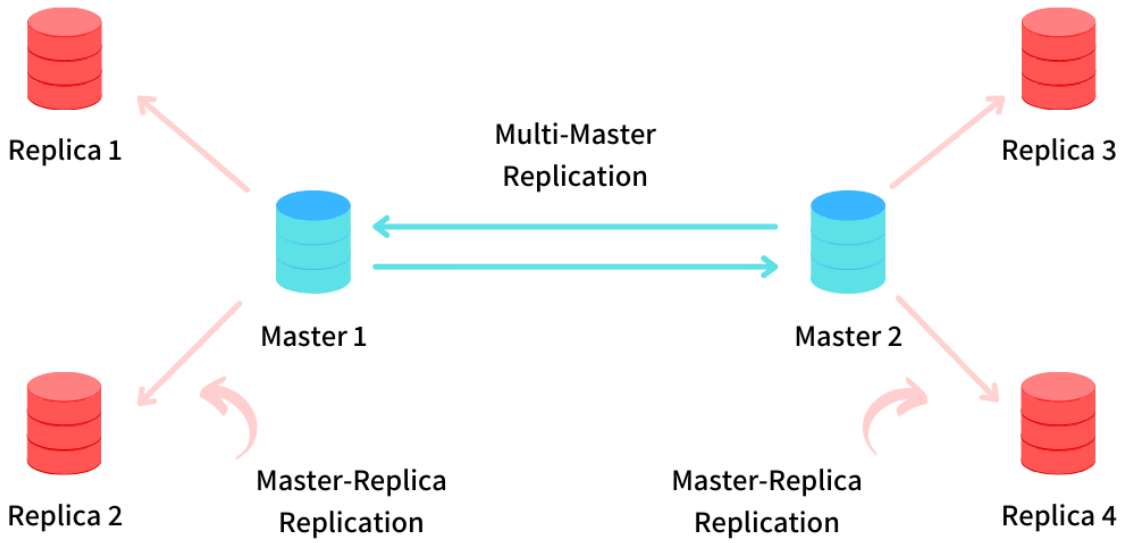
MongoDB



Chaque shard dans MongoDB peut être un replica set : un groupe de nœuds où un nœud est primaire (écriture) et les autres secondaires (lecture / failover). En cas de panne du primaire, un secondaire est élu automatiquement pour assurer la continuité de service.

- **Réplication :**
  - **Principe :** Utilisation de Replica Sets pour répliquer les données sur plusieurs serveurs.
  - **Schéma conceptuel :**
    - **Primary Node :** Responsable de toutes les écritures.
    - **Secondary Nodes :** Copient les opérations du primary pour assurer la redondance.
- **Application pratique dans notre système :**
  La réplication assure la haute disponibilité. En cas de panne du nœud primaire, l'un des secondaires peut être promu pour minimiser l'interruption du service.

RavenDB



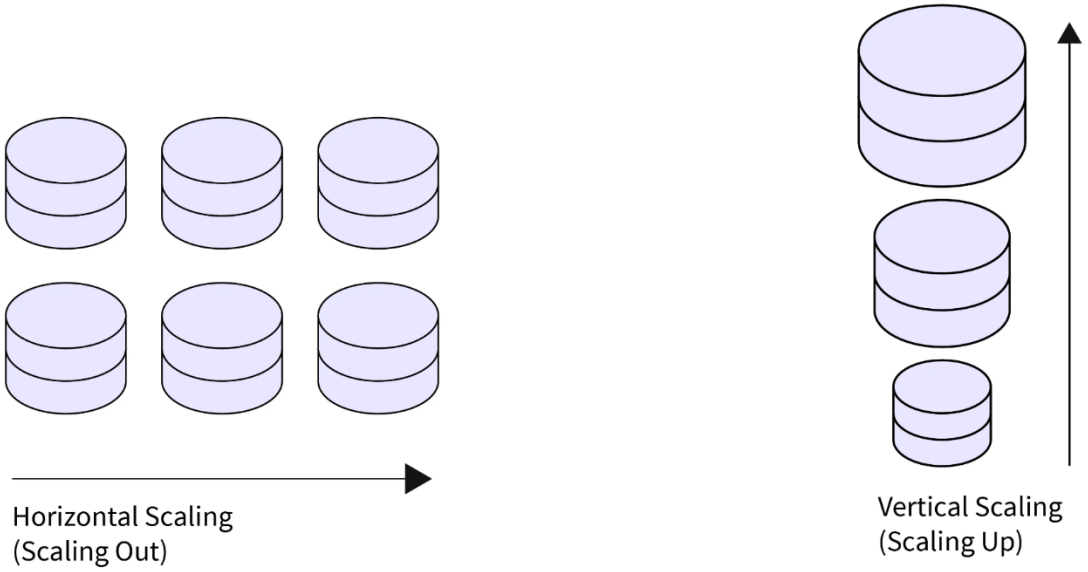
Multi-Master Replication

RavenDB permet une réplication automatique entre nœuds pour chaque base de données. Il s'agit d'une réplication **asynchrone multi-maître**, ce qui signifie que chaque nœud peut accepter des modifications qui seront synchronisées avec les autres nœuds.

- **Réplication :**
  - **Principe :** Chaque base peut être répliquée sur plusieurs nœuds selon des topologies définies.
  - **Schéma conceptuel :**
    - **Répliques actives :** Toutes les copies peuvent servir à la lecture ou à l'écriture.
    - **Replication Hub/Spoke ou Mesh :** Plusieurs configurations possibles pour synchroniser les bases.
- **Application pratique dans notre système :**
  Cette réplication facilite le déploiement multi-région et garantit la résilience en cas de panne d'un serveur. Les lectures/écritures peuvent continuer depuis un autre nœud sans interruption.

---

6. Montée en charge (schémas expliqués)



Ces figure illustre que MongoDB comme RavenDB sont capables de monter en charge horizontalement. Il suffit d'ajouter de nouveaux nœuds au cluster, et les données (ou leurs partitions) sont redistribuées automatiquement. Cela permet de traiter plus de requêtes, stocker plus de données, et améliorer la tolérance aux pannes.

MongoDB

- **Scalabilité horizontale (sharding) :**
  - **Principe :** Ajout de nouveaux nœuds pour distribuer la charge et le volume des données.
  - **Schéma conceptuel :**
    - **Shard Cluster :** Chaque shard contient une partie des données et fonctionne en replica set.
- **Application pratique dans notre système :**
  Lorsque le volume de données et le nombre de requêtes augmentent, nous pouvons ajouter des shards pour étendre les capacités de traitement sans interruption.

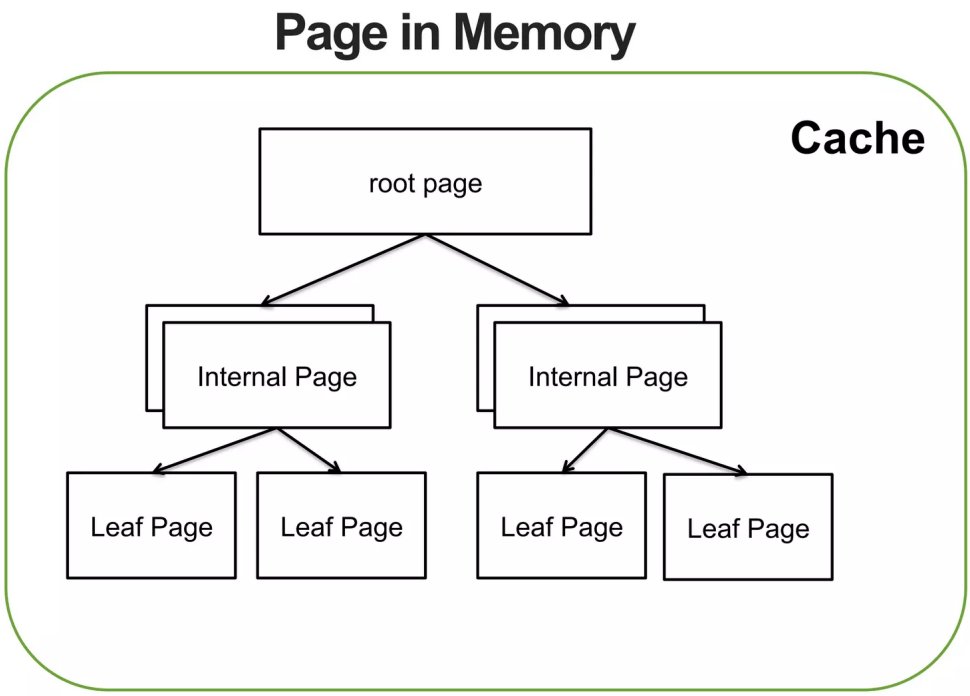
RavenDB

- **Scalabilité horizontale :**
  - **Principe :** Ajout de nœuds au cluster et distribution des bases de données entre ces nœuds.
  - **Schéma conceptuel :**
    - Chaque base peut être répliquée ou déplacée automatiquement sur les nouveaux nœuds.
- **Application pratique dans notre système :**
  En cas d'augmentation du nombre de **Books** ou d'**emprunts**, on peut facilement créer de nouvelles bases ou répliquer les existantes pour répartir la charge, sans reconfigurer les applications clientes.

---

7. Gestion du ou des caches mémoire (schémas expliqués)

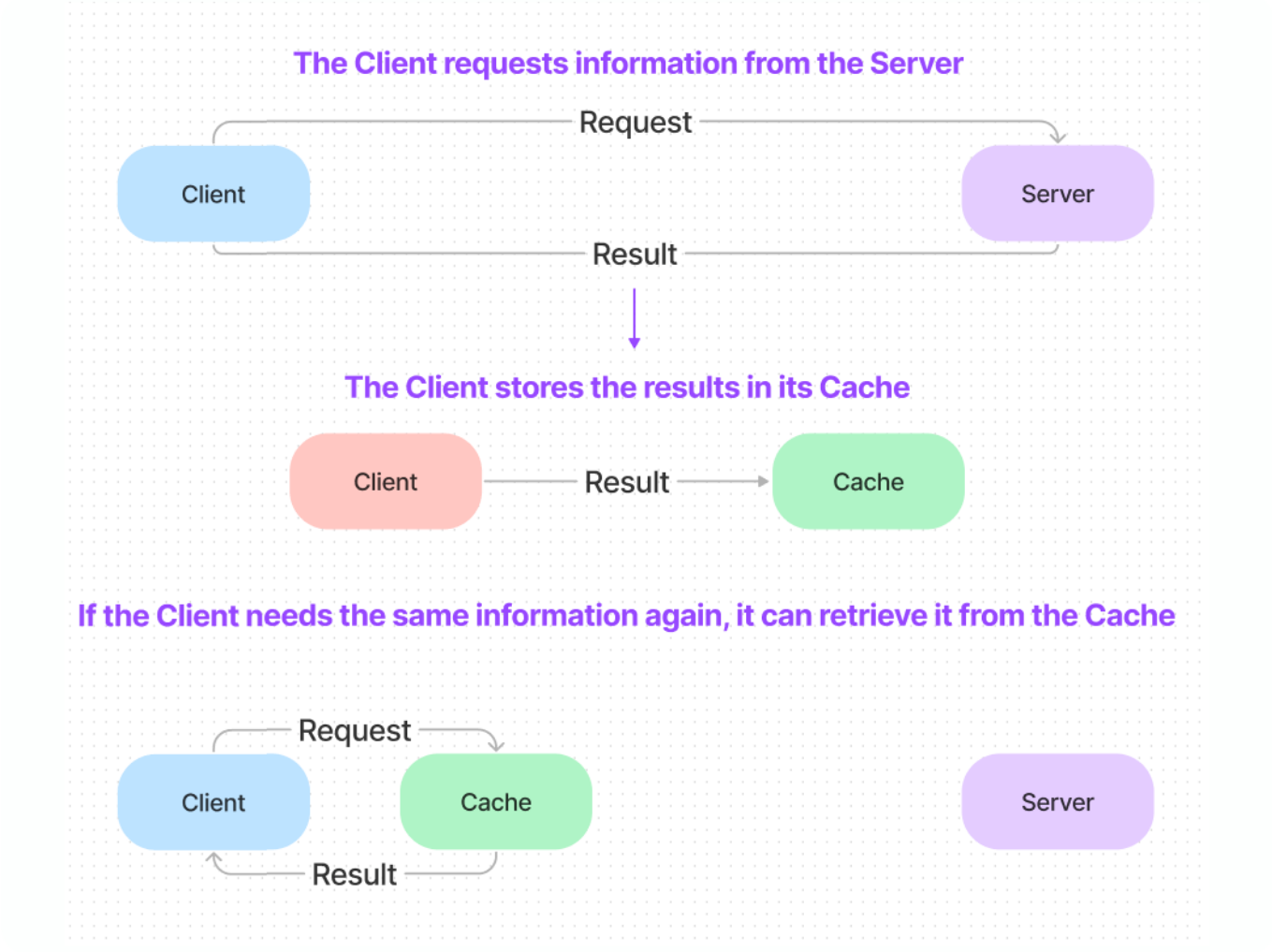
MongoDB



MongoDB utilise un moteur de stockage basé sur la mémoire (WiredTiger), qui garde les données récemment utilisées dans le cache mémoire (RAM). Cela accélère les lectures en évitant d'accéder au disque à chaque fois. La gestion du cache est automatique et adaptative.

- **Gestion du cache :**
  - **Principes :**
    - **WiredTiger Cache :** MongoDB utilise WiredTiger (son moteur de stockage par défaut) qui intègre un cache mémoire pour accélérer les opérations de lecture/écriture.
    - **OS Cache :** MongoDB exploite également la mémoire disponible du système d'exploitation pour stocker les pages de données.
  - **Schéma conceptuel :**
    - **Cache Memory Layer :** Combine le cache spécifique à WiredTiger et la mémoire de l'OS pour réduire l'accès disque.
- **Application pratique dans notre système :**
  Une configuration adéquate (taille du cache, paramètres de WiredTiger) peut améliorer significativement la performance des requêtes sur les collections telles que **Members** ou **Publishers**.

RavenDB



RavenDB exploite la mémoire disponible pour maintenir en cache les documents récemment lus ou fréquemment consultés. Le système de cache est intégré et optimisé pour minimiser les accès disque.

- **Gestion du cache :**
  - **Principes :**
    - **Documents récemment consultés :** stockés en mémoire pour accélérer les lectures.
    - **Index également mis en cache** pour des recherches rapides.
  - **Schéma conceptuel :**
    - Cache segmenté par base de données, avec politique de remplacement LRU (Least Recently Used).
- **Application pratique dans notre système :**
  Les requêtes fréquentes sur des collections comme **Books** ou **Members** peuvent tirer avantage du cache intégré sans configuration manuelle, améliorant la latence des opérations.

Ces concepts, bien compris et correctement mis en œuvre, permettent de concevoir une architecture NoSQL robuste et évolutive pour notre système. MongoDB et RavenDB offrent tous deux des approches complémentaires adaptées à des contextes différents : MongoDB pour une scalabilité extrême avec sharding automatisé, RavenDB pour une réplication souple et un développement rapide grâce à ses outils intégrés.