Cycle Ingénieur - Troisième Année

Projet bases de données

F-SD312 2019 - 2020

Pourquoi utiliser MongoDB ?

Étudiants :
Lorène Authier
Axel Blanc
Claire Fourdan
Willy Lao
Paul Templier

Encadrant :
Dennis Wilson

Version du 31 mars 2020

1 La ville SQL et la ville NoSQL

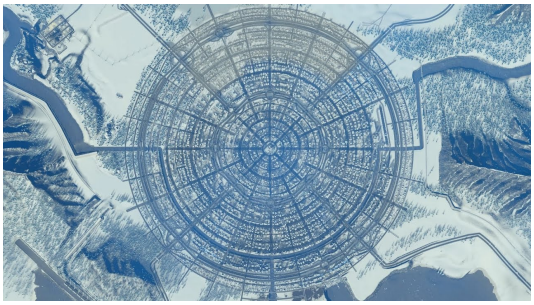

FIGURE 1 – Ville SQL

FIGURE 2 – Ville NoSQL

Imaginons deux villes très différentes. D'un côté, la **ville SQL** est imposante par la rigueur de sa structure et l'uniformité de ses bâtiments. Chacun de ses habitants parle le même langage, sans quoi ils ne pourraient pas se comprendre. De l'autre, la **ville NoSQL** frappe par ses bâtiments hétérogènes, comme si ses architectes ne s'étaient pas soucié de l'entourage de leurs édifices lors de leur construction. Ses habitants peuvent parler une langue différente sans qu'il n'y ait de friction.

SQL : "la ville des banquiers"	**NoSQL : "la ville des développeurs"**
Basé sur des tableaux (jointures)	Basé sur des documents, des graphes...
Très pertinent pour des transactions multi-lignes	Flexibilité (JSON) : créer des documents sans spécifier leur structure
Changer la structure est difficile	Ajouter des champs est très facile

La ville SQL est la "ville des banquiers" car elle se prête très bien aux documents structurés et codifiés. La ville NoSQL est la "ville des développeurs" car sa flexibilité est pertinente pour des structures de données changeantes ou des mises à jour fréquentes comme nous le verrons dans la suite.

MongoDB, sujet de notre étude, se place sous la bannière du NoSQL contrairement aux autres leaders sur le marché.

2 Pourquoi MongoDB est-il différent de ses concurrents ?

MongoDB est souvent considéré comme la nouvelle génération de système de stockage de documents. Son développement en modèle open source commence en **2009** et il devient complètement gratuit en **2018**. Contrairement aux quatre premiers leaders du marché, MongoDB utilise le NoSQL [14].

Popularité (db-engines.com)	**Bases de données**
1	Oracle
2	MySQL
3	Microsoft SQL Server
4	PostgreSQL
5	MongoDB

Les entreprises qui ont déjà adopté MondoDB incluent Adobe, EA, Google, Ebay, UK Government, IBM... Prenons tout de même le temps de nous renseigner sur les concurrents de MongoDB qui utilisent aussi le NoSQL [13].

Voici quelques unes de leurs caractéristiques :

- **Postgre-SQL** : Hybride SQL / No-SQL. Base de données importantes, requêtes plus complexes.
- **Apache Cassandra (Facebook)** : IoT, perspectives rapides en temps réel (log activities, error logging, and sensor data).
- **Apache HBase** : Pour des accès de lecture ou écriture en temps réel, avec des milliards de lignes et des millions de colonnes.

Contrairement à la plupart des gestionnaires de base de données, avec MongoDB, on peut s'abstraire de la création de schéma de tables relationnelles (ORM) ou des requêtes SQL complexes. MongoDB est, comme nous allons le voir, orientée objet. Au lieu d'utiliser des tables et des lignes comme dans les bases de données relationnelles, MongoDB est construit sur une architecture de collections et de documents, ce qui lui confère une grande souplesse d'utilisation et nous permet de faire évoluer le schéma de la base de données à la volée. Ces données peuvent ensuite être exploitées par du Javascript, directement intégré dans MongoDB, ou par d'autres langages comme Python.

3 Fonctionnement de MongoDB

3.1 Une base de données "Document-based"

MongoDB utilise un modèle de données basé sur les **documents**.

Un document est un ensemble de clés et de valeurs, similaire à un dictionnaire, contenant l'information. Des données qui, en SQL, se trouveraient dans des tables relationnelles séparées sont alors stockées dans un seul et même document, identifié par une unique clé primaire. Ce document est l'unité de base des données dans MongoDB.

Ces données sont alors stockées par MongoDB sous forme de documents **BSON**, une représentation binaire qui étend la représentation JSON pour inclure des types des données supplémentaires (int, long, date...). Cette représentation augmente la fiabilité du stockage, du traitement et de la manipulation des données [9, 12].

Une **collection** est un ensemble de documents.

```
{
  "_id":
      ObjectId("5ad88534e3632e1a35a58d00"),
  "name": {
    "first": "John",
    "last": "Doe" },
  "address": [
    { "location": "work",
      "address": {
        "street": "16 Hatfields",
        "city": "London",
        "postal_code": "SE1 8DJ"},
      "geo": { "type": "Point", "coord": [
         51.5065752,-0.109081]}},
  ],
  "phone": [
    { "location": "work",
      "number": "+44-1234567890"},
     ],
  "dob": ISODate("1977-04-01T05:00:00Z"),
  "retirement_fund":
        NumberDecimal("1292815.75")
}
```

FIGURE 3 – exemple de document JSON

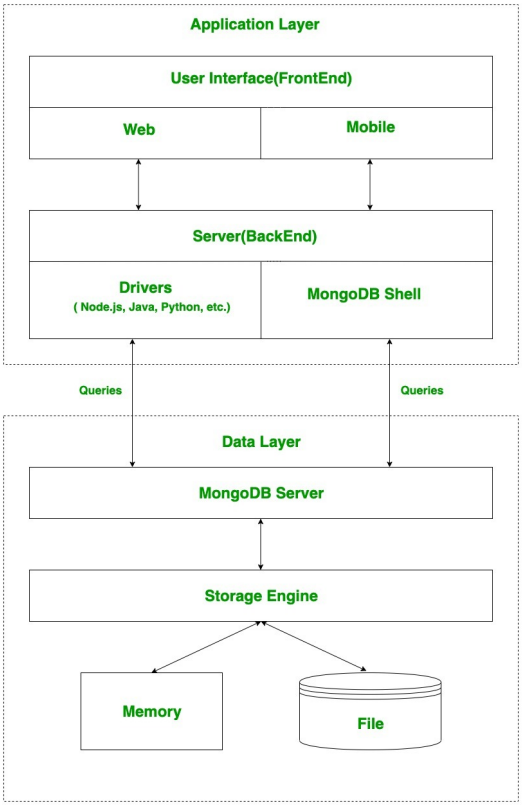

FIGURE 4 – Schéma de l'architecture de MongoDB

3.2 Architecture de MongoDB

Le fonctionnement de MongoDB se base sur deux couches :
— la couche d'application (*Application Layer*)
— la couche de données (*Data Layer*)
La couche d'application contient une partie Frontend auquel l'utilisateur aura accès par le biais du web ou d'un mobile.

La partie Server (Backend) contiens les Drivers et le Shell MongoDB (une interface JavaScript interactive) qui communiquent avec MongoDB et la couche des données grâce aux requètes.

C'est le serveur MongoDB qui réceptionne les requètes et les transmet au moteur de stockage, qui gère les données (écriture, lecture).

Le schéma ci-dessus à droite, en anglais, représente l'architecture de MongoDB telle que décrite dans les lignes précédentes [6].

Le partitionnement automatique est une autre caractéristique intrasèque à MongoDB qui permet aux données d'une collection MongoDB d'être réparties sur plusieurs systèmes pour une évolutivité horizontale à mesure que les volumes de données et les exigences de débit augmentent. L'architecture physique de la distribution des serveurs est appelée un cluster. Ce dernier comporte au moins 2 routeurs (mongos), 3 serveurs de configuration (Config Servers) et 2 serveurs de données (shards). Le Mongos s'occupe du routage des requêtes, il répartit et traite les données rapidement, et au nombre de 2, rend possible la tolérance aux pannes de MongoDB [2].

4 Démonstration

Un notebook de démonstration est fourni pour accompagner ce rapport. Cette démonstration permet d'illustrer l'un des nombreux usages possibles d'une base de données MongoDB. Les données choisies sont issues du jeu League Of Legends, qui permet de récupérer de nombreuses statistiques à propos des matchs passés grâce à son API.

Après un pré-traitement permettant de les intégrer à Tableau, les données présentées dans cette démonstration ont également été utilisées dans le projet de DataViz, où des statistiques plus avancées sont présentées. Ce notebook a pour but de montrer la manipulation des données plus que leur exploration ou exploitation pour l'analyse.

Les documentations des outils et API utilisés sont accessibles en cliquant sur leurs noms.

4.1 Création et remplissage d'une base de données

Plus de 8500 matchs de plusieurs joueurs du Supaero Gaming Club ont été récupérés grâce à l'API de Riot, l'éditeur de League of Legends. La documentation de l'API est disponible ici.

Les données ont été récupérés grâce à des appels successifs à l'API [4, 5, 3]. 3 types de requêtes ont été nécessaires :

1. L'API GET_getBySummonerName permet d'accéder à l'identifiant unique d'un joueur à partir de son pseudo
2. L'API GET_getMatchlist donne accès à la liste des identifiants uniques des parties jouées par chaque personne.
3. Enfin, l'API GET_getMatch donne pour une partie jouée les statistiques associées à chaque participant.

Les statistiques complètes de matchs ont été récupérées pour 6 comptes actifs, sur 2 ans (depuis mars 2018). Le facteur limitant fut la limite de l'API de Riot, fixée à 100 appels par 2min pour une clé de développement.

Les résultats, sous forme de documents JSON, ont été injectés dans une base MongoDB locale grâce à la bibliothèque python PyMongo.
Pour cette démonstration, les statistiques de matchs ont été mises à disposition dans une base de données en ligne MongoDB Atlas.

4.2 Requêtes effectuées sur la base de données créée

Une fois la base de données remplie, nous pouvons effectuer des requêtes dessus pour en sortir des statistiques intéressantes sans itérer sur tous les documents. Afin de caractériser et étudier les profils des joueurs, des informations comme le *win rate*(pourcentage de victoire) ou le *KDA*(ratio entre les participations aux éliminations et les morts) permettent d'avoir une idée de la performance sur un personnage. Le groupe de personnages les plus joués par une personne parmi les 150 disponibles dans le jeu s'appelle le *champion pool*, et peut être varié ou spécialisé selon les profils.

Le format *aggregate* permet de lister des opérations à faire sur les documents d'une collection. Le résultat de chaque étape est utilisé comme entrée pour l'étape suivante, permettant des opérations en cascade. Les résultats intermédiaires sont donc observables, en exécutant seulement une partie de la requête, rendant la conception plus

simple.
Une fois définie, la requête est exécutée en entier sur la base de données, et seul le résultat final est renvoyé.

Les opérations et leur paramètres sont définis dans des dictionnaires imbriqués. Généralement, la clé est le nom de l'opération tandis que la valeur associée est le paramètre de l'opération, qui lui même peut être une opération définie par un dictionnaire.

Plusieurs types d'opérations sont présentées dans le notebook, dont :
- ***$match*** sélectionne les documents dont les champs correspondent aux valeurs indiquées
- ***$sample*** sélectionne aléatoirement un certain nombre de documents, permettant notamment de tester une requête sur un plus petit dataset pour gagner du temps pendant son élaboration
- ***$unwind*** prend dans chaque document une liste et crée un nouveau document par élément de cette liste, avec tous les champs identiques au document parent. A partir d'un document ayant une liste de 10 joueurs comme valeur au champ "participants", il va donc renvoyer 10 documents ayant chacun un des joueurs comme valeur au champ "participants".
- ***$project*** permet de ne garder que certains champs dans chaque document, et donc d'en réduire la taille
- ***$set*** assigne une nouvelle valeur à un champ dans chaque document, le créant si nécessaire
- ***$sort*** trie les documents renvoyés dans un ordre défini

4.3 Quelques retours sur cette première utilisation

Le gestionnaire MongoDB se prête particulièrement bien à des données de formats similaires à celles de notre démonstration. En effet, les listes de tailles variables présentes pas les documents JSON seraient difficiles à intégrer dans une base SQL, mais s'utilisent sans efforts dans une base MongoDB grâce au format identique de stockage des données.

La création de requêtes demande un temps d'adaptation en raison de sa différence par rapport à SQL, mais le format permet de monter rapidement en compétences dessus et la documentation en ligne est assez complète pour permettre une utilisation extensive.

5 Comparaison avec les systèmes de gestion de base de données traditionnelles

Nous avons constaté que des organisations de toutes tailles adoptent MongoDB. Alors, quelles sont les différences observées par rapport à un système de gestion de base de données relationnelle ?

Nous allons comparer un SGBD relationnel, comme MySQL, et MongoDB sur plusieurs points.

- **L'ajout de données dans la base.** Le développement est simplifié car les documents MongoDB correspondent naturellement aux langages de programmation modernes orientés objet. L'utilisation de MongoDB supprime la nécessité de créer une couche de mappage relationnel-objet, qui traduit les objets dans le code en tables relationnelles et peut être complexe.
- **Stockage en mémoire.** MongoDB a tendance naturellement à utiliser plus de mémoire car il doit stocker les noms de clés dans chaque document. Ceci est dû au fait que la structure des données n'est pas nécessairement cohérente parmi les objets.
- **Flexibilité** MongoDB n'exige pas une structure de données unifiée sur tous les objets. Ce modèle souple permet au schéma de la base de données d'évoluer avec les besoins de l'entreprise, là où la structure relationnelle rigide du SQL ajouterait une surcharge aux applications et ralentirait les développeurs forcés d'adapter les objets à cette structure précise [8]. En conséquence, MongoDB est moins fortement conforme au principe ACID (Atomic, Consistency, Isolation, Durability) que les gestionnaires relationnels classiques.
- **Rapidité des requêtes.** Comme expliqué plus tôt, MongoDB est très pratique pour stocker une quantité importante de données sous forme de documents. Les requêtes MongoDB sont donc dans ce cas d'utilisation bien plus rapides que celles qui auraient été nécessaires en MySQL. En effet, les données à récupérer sont stockées "à un seul endroit" et donc peuvent être récupérées en une seule recherche. Cet avantage disparaît si les données imitent en partie un modèle relationnel : le code doit effectuer plusieurs requêtes indépendantes et peut devenir plus lent qu'un SGBD classique.
- **Disponibilité et Répartition de charge** MongoDB intègre une méthode, le "sharding", décrite plus haut. Elle permet de distribuer les données sur plusieurs instances de base de données elles-même réparties

sur une ou plusieurs machines. Ces machines forment un cluster MongoDB, appelé sharded cluster, qui a pour atout principal de permettre une répartition des charges : les requêtes se font sur des plus petits jeux de données et peuvent être parallélisées. Il est également possible d'ajouter des serveurs supplémentaires sans interruption de service si les besoins augmentent. Cette scalabilité horizontale est beaucoup plus délicat à réaliser avec MySQL [1, 7].

5.1 Alors, dans quels cas utiliser MongoDB ?

Commençons par un petit exemple : nous souhaitons que notre base de données contiennent un "jeu" de séries télévisées, chaque série a plusieurs saisons elles-mêmes composées de plusieurs épisodes qui eux-même ont plusieurs acteurs et plusieurs critiques. Lorsqu'un utilisateur va visiter le site, il va se rendre sur la page d'une de ces séries pour y voir toutes les saisons, tous les épisodes, acteurs, revues, de cette série : le but est donc de récupérer toute l'information reliée à cette série.

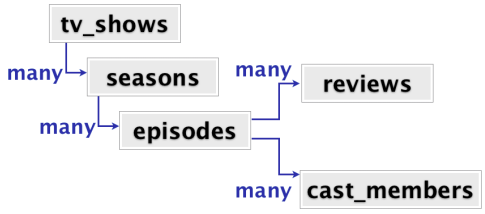

FIGURE 5 – ORM d'une base relationnelle

```
{title: 'Babylon 5',
 seasons: [
  {season_number: '1',
    episodes: [
     {ordinal_within_season: '1',
      title: 'Midnight on the Firing Line'
      reviews: [{...}],
      cast_members: [{...}]
     }
    ]
  }
 ]
}
```

FIGURE 6 – Exemple d'une unité de donnée format document

Dans une table relationelle classique, les données seraient organisées selon plusieurs tables : par exemple, une table *tv-shows*, une table *seasons*, avec une clé permettant de la relier à *tv-shows*, une table *episodes* avec une clé dans *seasons*, ainsi que des tables *reviews* et *cast-members* reliées par des clés à *episodes*. Pour récupérer toute l'information sur une série, il faudrait procéder à une jointure sur cinq tables.

Au contraire, on peut modéliser ces informations de façon imbriquée, dans un document. Les informations sur une série serait contenue dans une structure assez large, avec un tableau des saisons, chacune étant elle-même un document contenant le tableau des épisodes... C'est ainsi que MongoDB stocke les données. Cette fois, il est beaucoup plus rapide de récupérer les informations, cela se fait en une seule fois [15] ! MongoDB est particulièrement adaptée dans le cas de ce type de données, lorsque la taille de la collection varie, et lorsque des informations "fonctionnent ensemble", par un exemple pour un post et les commentaires qui lui sont rattachés.

MongoDB sera notamment un un excellent choix pour la mise en place de projets Web qui se basent sur une importante masse de données déstructurées, pour le développement d'applications mobiles, pour de l'analyse en temps réel, de la gestion de contenu ou de l'archivage de données récupérées par des objets connectés [17, 16, 18]...

6 Conclusion :

MongoDB est une base de données généraliste, qui a plusieurs avantages dont nous pouvons tirer profit, comme son schéma dynamique, sa mise à l'échelle facile, ses excellentes performances pour des requêtes simples, l'écriture et la lecture intuitive des documents JSON...

Il faudra néanmoins faire attentions à quelques points ! MongoDB sera un mauvais choix si vous n'avez pas pour priorité la mise à l'échelle, si vos features sont fortement changeantes pour des prestataires de services par exemple, si de multiples opérations sont à prévoir pour le cas de transactions, ou si beaucoup de logiciels et d'équipes seront amenées à utiliser la base de données. Attention aussi, le langage de requête est riche mais les requêtes MongoDB ne sont pas plus simples à écrire qu'en SQL dans certains cas [10, 11].

Si vous recherchez un gestionnaire de base de données proposant de très bonnes performances, traitant des données complexes, sans restrictions liées à la conception de schéma, si vous prévoyez une augmentation de la taille de votre base de données dans le futur, nous vous recommendons d'opter pour MongoDB !

Références

[1] Claudia BROUCHE. *MongoDB et le sharding : Scaling vertical vs. Scaling horizontal.* [En ligne : consulté le 28 mars 2020]. SupInfo. URL : https://www.supinfo.com/articles/single/4761-mongodb-sharding.

[2] *Distribuez vos données avec MongoDB.* [En ligne : consulté le 30 mars 2020]. OpenClassrooms. URL : https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql/4474616-distribuez-vos-donnees-avec-mongodb.

[3] *Get By Summoner Name.* [En ligne : consulté le 20 mars 2020]. API RiotGames. URL : https://developer.riotgames.com/apis#summoner-v4/GET_getBySummonerName.

[4] *Get Match.* [En ligne : consulté le 20 mars 2020]. API RiotGames. URL : https://developer.riotgames.com/apis#match-v4/GET_getMatch.

[5] *Get Matchlist.* [En ligne : consulté le 20 mars 2020]. API RiotGames. URL : https://developer.riotgames.com/apis#match-v4/GET_getMatchlist.

[6] *How MongoDB works ?* [En ligne : consulté le 29 mars 2020]. GeeksForGeeks. URL : https://www.geeksforgeeks.org/how-mongodb-works/?fbclid=IwAR3lj9smShA_TbzHWZBZG1H4I1m5OSjJgXAqwZgMPaayQSV9Hmogv

[7] *Mongo DB : Editeur de Base de données NoSQL ("Not Only SQL").* [En ligne : consulté le 28 mars 2020]. NextDecision. URL : https://www.peerbits.com/blog/mongodb-vs-mysql-which-database-choose-for-your-business.html.

[8] *MongoDB : présentation et comparaison avec MySQL.* [En ligne : consulté le 26 mars 2020]. Digital Guide - Ionos. URL : https://www.ionos.fr/digitalguide/sites-internet/developpement-web/mongodb-presentation-et-comparaison-avec-mysql/.

[9] *MongoDB Architecture Guide :Overview.* [En ligne : consulté le 29 mars 2020]. Info-MongoDB.com. URL : https://info-mongodb-com.s3.us-east-1.amazonaws.com/MongoDB_Architecture_Guide.pdf?fbclid=IwAR33TT_Q7DSzqSnUWckNgPY9PJP-EP9UM_P5VqzbcgTD79_V2aHlKn1L4Vw.

[10] *MongoDB Vs MySQL : Which Database You Should Choose for your Business ?* [En ligne : consulté le 28 mars 2020]. Peerbits. URL : https://www.peerbits.com/blog/mongodb-vs-mysql-which-database-choose-for-your-business.html.

[11] *MongoDB : Retour d'expérience.* [En ligne : consulté le 26 mars 2020]. <InsertAfter/>. URL : https://insertafter.com/fr/blog/retour_xp_mongodb.html.

[12] Sacha SCHUTZ. *MongoDB, la base de donnée pour dire adieu à Sql.* [En ligne : consulté le 29 mars 2020]. URL : http://dridk.me/MongoDB.html.

[13] *SQL vs NoSQL.* [En ligne : consulté le 20 mars 2020]. Mark Smallcombe. URL : https://www.xplenty.com/blog/the-sql-vs-nosql-difference/.

[14] *The rise of MongoDB.* [En ligne : consulté le 20 mars 2020]. Joe Rezendes. URL : https://medium.com/@joerez/the-rise-of-mongodb-8ac6f7cb8a95.

[15] *Un exemple du mauvais usage de MongoDB.* [En ligne : consulté le 30 mars 2020]. Blog. URL : https://blog.seboss666.info/2015/02/un-exemple-du-mauvais-usage-de-mongodb/.

[16] *Usage de MongoDB.* [En ligne : consulté le 27 mars 2020]. Scenari-community. URL : https://stph.scenari-community.org/contribs/nos/Mongo2/co/activiteapprentissage_4.html.

[17] *Use Cases.* [En ligne : consulté le 28 mars 2020]. MongoDB. URL : https://www.mongodb.com/use-cases.

[18] *When to Use (and Not to Use) MongoDB.* [En ligne : consulté le 29 mars 2020]. DZone. URL : https://dzone.com/articles/why-mongodb.