Introduction to Databases Checkpoint

:

Présentation de comparisent NoSQL (MongoDB) à SQL

Nom et prénom :

Zitouni Marwen

1. Définitions du SQL et du NoSQL

A. SQL, le langage des données structure

Le SQL, qui signifie Structured Query Language, est un langage informatique normalisé permettant de communiquer avec une base de données.

Les bases de données relationnelles ont longtemps été les plus populaires. Il s'agit de bases de données dans lesquelles l'information est organisée avec des tableaux à deux dimensions nommées « tables ». Les lignes correspondent aux enregistrements. Chaque enregistrement contient un groupe d'informations – les attributs – relatives à un sujet. La plupart des systèmes de gestion de bases de données relationnelles reconnaissent le SQL.

Plus concrètement, le SQL a pour but de stocker, de manipuler et de retrouver ces données. Il permet également d'effectuer des requêtes, de mettre à jour les données ou de les réorganiser, de créer et de modifier le schéma et la structure d'un système de base de données.

B. NoSQL, le SQL des bases de données non relationnelles

NoSQL signifie à la fois « Not only SQL », car certains langages NoSQL comprennent le langage SQL en plus de leur propre capacité, et « Non-relationnel » parce qu'il ne peut pas stocker facilement des données relationnelles.

La particularité des bases de données NoSQL est qu'elles n'utilisent pas le modèle relationnel. Il n'y a donc pas de tableau avec des caractéristiques et nombres d'attributs fixes. Les schémas sont donc absents ou flexibles. Cela permet de regrouper des données ayant des structures différentes. Les bases de données NoSQL peuvent également être distribuées, c'est-à-dire qu'elles peuvent être stockées sur plusieurs systèmes, plusieurs serveurs par exemple. Pour approfondir le sujet, nous vous conseillons l'article NoSQL : Tout comprendre sur les bases de données non relationnelles.

2. Avantages et inconvénients du SQL et du NoSQL

A. SQL et NoSQL solutions différentes au théorème de CAP

Le théorème de Brewer, aussi appelé théorème de **CAP**, a été formalisé en 2000 par Eric A. Brewer. Il s'intéresse aux trois propriétés fondamentales des bases de données :

- **Cohérence** (**C**onsistency en anglais) : la donnée a toujours un seul état visible à un instant précis, peu importe le nombre de copies ;
- **Disponibilité** (**A**vailability en anglais) : la donnée est disponible tant que le système tourne ;
- **Distribution** (**P**artition tolerance en anglais) : toute requête fournit un résultat correct, peu importe le nombre de serveurs.

Selon le théorème de CAP, dans toute base de données, vous ne pouvez respecter au plus que 2 propriétés parmi la cohérence, la disponibilité et la distribution.

Il faut donc prioriser les besoins du projet et choisir la technologie adaptée au bon couple de propriétés.

Pour le couple **CA** (cohérence et disponibilité), on utilisera des langages SQL comme **Oracle, MySQL, SQLServer**.

Pour le couple **CP** (cohérence et distribution), on choisira des langages NoSQL comme **HBase, BigTable, MongoDB* et CosmosDB*.**

Pour le couple **AP** (disponibilité et distribution), on privilégiera des langages NoSQL comme **Elasticsearch, Spark, Neo4j, OrientDB, FlockDB, Redis, SimpleDB*, Memcached*, DynamoDB*, CouchBase*, Cassandra*.**

***Les langages suivis d'un astérisque ont la capacité de pouvoir passer de CP à AP ou inversement.**

B. Cas d'usages du SQL et du NoSQL

La data est un champ en constante évolution à la fois par la quantité de données et leur diversité. La création du NoSQL après le SQL répond donc à ces changements.

Le SQL est à privilégier lorsque les données sont structurées et que leurs relations sont fondamentales. Si les bases de données sont complexes, ce système est principalement choisi.

Au contraire, si les données ne sont pas structurées ou changent de format avec le temps, le NoSQL sera pertinent. **Les bases de données NoSQL sont plus adaptées lorsque l'on manipule de très larges volumes de données dont les relations entre celles-ci ne sont pas particulièrement importantes.** Lors des augmentations de volumes de données, deux choix sont possibles :

- **la scalabilité verticale** : augmentation de capacité sur un serveur existant,
- **la scalabilité horizontale** : augmentation du nombre de serveurs.

Le SQL permet uniquement la scalabilité verticale alors que le NoSQL autorise à la fois la scalabilité verticale et horizontale, car il est distribué. On comprend donc rapidement la difficulté à laquelle peut faire face le SQL en cas de très large volume de données.

Le NoSQL est quant à lui plus limité dans la façon de faire des recherches. On accède à ces données par leur identifiant, nommé « clé primaire ». Il est donc facile de retrouver une donnée, mais trouver l'ensemble des données supérieures à une valeur, par exemple, devient beaucoup plus complexe. Le SQL n'a lui pas de soucis avec ce genre de besoin.

Le SQL et le NoSQL resteront tous les deux utilisés à l'avenir. Chaque langage a ses avantages. En revanche, les inconvénients peuvent amener, en fonction du besoin, à utiliser l'autre langage.

Tableau comparatif entre les terminologies Mongo et celles du SQL

SQL	MongoDB
database	database
table	collection
enregistrement	document
colonne	champ
index	index
jointure	objet, dénormalisation
clef primaire	clef primaire (Dans MongoDB la clef primaire est automatiquement attribué au champ _id)
clef étrangère	reférence