SGBD orienté colonnes : Cassandra

Louis Langrand, Benoît Deslandes, Melvin Altrach, Matthieu Scherrer, Brice Le Pape

ISAE-Supaero

18/02/2019

Overview

Généralités

ID Employé	Nom	Prénom	Salaire
1	Deslandes	Benoît	40000
2	Scherrer	Matthieu	45000
3	Altrach	Melvin	3
4	Langrand	Louis	1000000
5	Le Pape	Brice	50000

Table – Exemple de table contenue dans une base de données

Base de données orientée lignes

1,Deslandes,Benoît,40000 ; 2,Scherrer,Matthieu,45000 ; ...

Base de données orientée colonnes

1,2,3,4,5 ; Deslandes,Scherrer,Altrach,Langrand,Le Pape ; ...

Généralités

Avantages des SGBD orientés colonnes :

- Ajouter des colonnes plus facilement aux tables (les lignes n'ont pas besoin d'être redimensionnées)
- Compression par colonne, efficace lorsque les données de la colonne se ressemblent
- Création de tables de tailles presque infinies (2 milliards de colonnes), adaptées pour des problèmes de big data
- La base de données peut accéder plus précisément aux données dont elle a besoin pour répondre à une requête, plutôt que d'analyser et de supprimer des données non souhaitées dans chaque ligne

Histoire de Cassandra

1. 2008 : libérée par Facebook en open source
2. 2009 : devient un projet *Apache Incubator*
3. 2009-2019 : adoptée par Twitter, Netflix, Spotify...
4. Aujourd'hui : 11ème base la plus populaire

Apache Cassandra est un système de gestion de base de données (SGBD) de type NoSQL conçu pour gérer des quantités massives de données sur un grand nombre de serveurs.

Caractéristiques : Concepts

Keyspace

C'est l'équivalent d'une *database* en bases de données relationnelles. À noter qu'il est possible d'avoir plusieurs *keyspaces* sur un même serveur.

Colonne

Une colonne est composée d'un nom, d'une valeur et d'un *timestamp*.

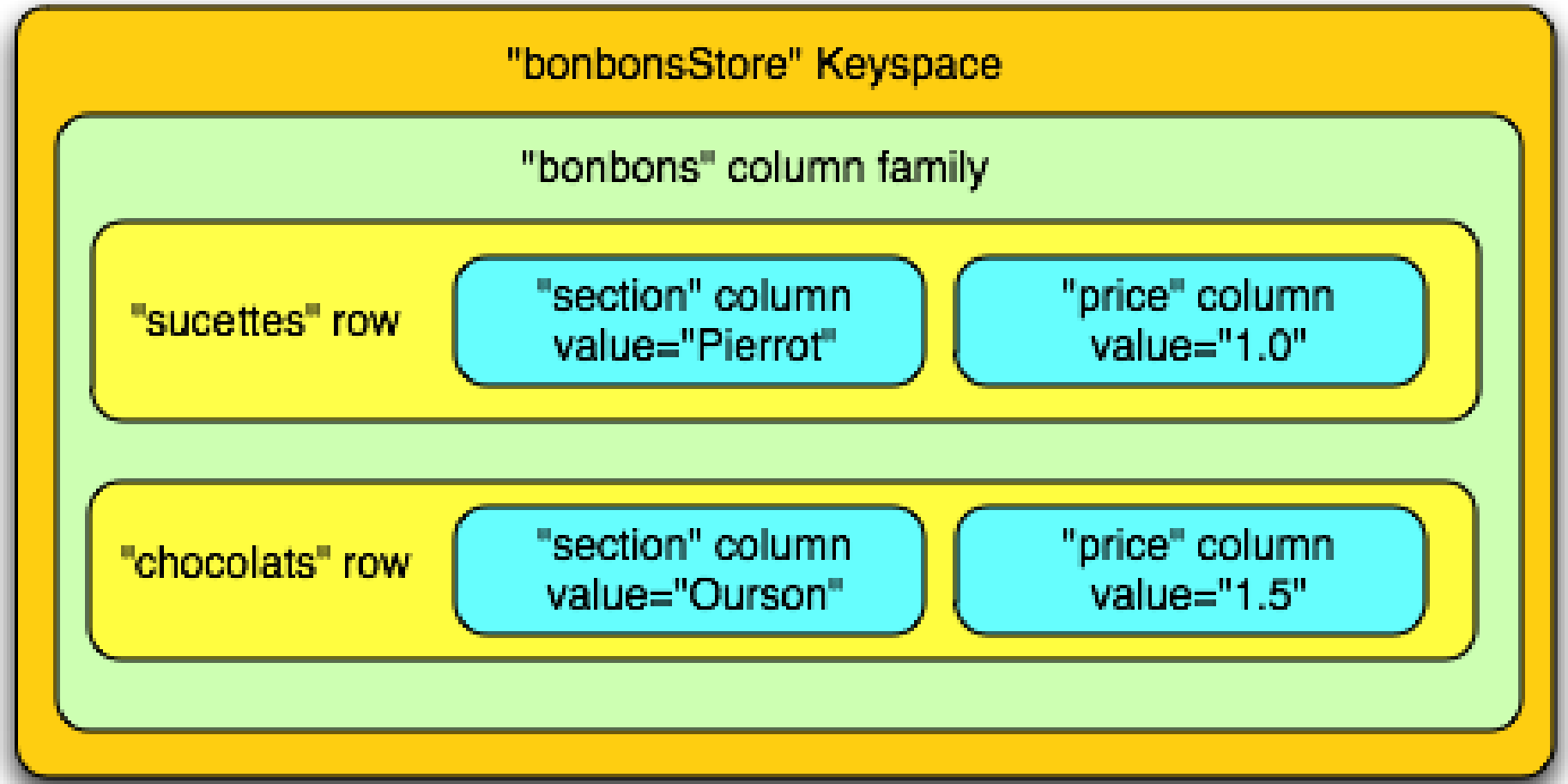
Ligne (ou *row*)

Une ligne est représentée par une clé et une valeur. Les « wide rows » permettant de stocker énormément de données, avec beaucoup de colonnes et les « skinny rows » permettant de stocker peu de données.

Famille de colonne

C'est l'objet principal de données et peut être assimilé à une table dans le monde des bases de données relationnelles.

Caractéristiques : Exemple



Caractéristiques : Architecture

Décentralisation

Tous les nœuds du cluster ont le même rôle. Il n'y a pas de point individuel de défaillance. Les données sont distribuées dans la grappe de serveurs (chaque nœud contient des données différentes), et il n'y a pas de master : tous les nœuds peuvent traiter toutes les requêtes.

Protocole *Gossip*

Le protocole Gossip est un protocole de communication de type « peer-to-peer » dans lequel les nœuds échangent périodiquement des informations sur leur état mais également sur ce qu'ils savent des autres nœuds (localisation et informations).

Caractéristiques : Réplication des données

La réplication est le processus permettant de stocker des copies des données sur de multiples nœuds afin de permettre leur fiabilité et la tolérance à la panne. Quand un keyspace est créé dans Cassandra, il lui est affecté la stratégie de distribution des réplicas, c'est-à-dire le nombre de réplicas et la manière dont ils sont répliqués dans le cluster.

Replication Factor

Un facteur de réplication de n signifie qu'il y a n copies de chaque ligne. La règle générale est que le facteur de réplication ne doit pas être supérieur au nombre de nœuds dans le cluster.

Caractéristiques : Modèle de données

Modèle orienté colonne

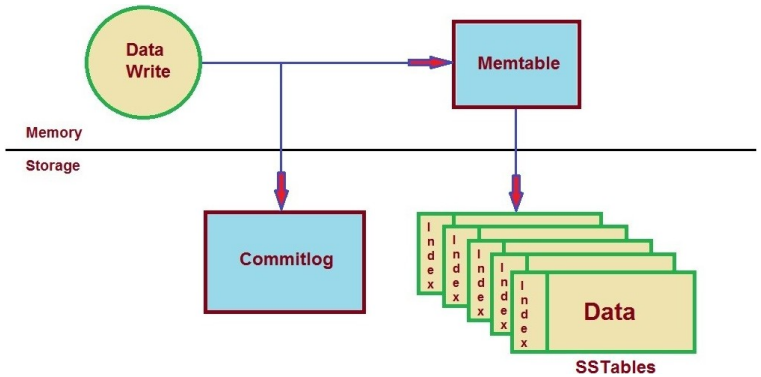
Contrairement à une base de données relationnelle, il n'est pas nécessaire de modéliser toutes les colonnes puisqu'une ligne n'a, potentiellement, pas le même ensemble de colonnes. Les colonnes et leurs métadonnées peuvent être ajoutées par l'application lorsque cela s'avère nécessaire.

Différence avec une base de donnée relationnelle

- Conçu pour répondre à des problématiques de **données distribuées**.
- Ne peut ni faire de jointures, ni sous-requêtes.
- Privilégie la dénormalisation des données.

Accès aux données : Écriture

Cassandra est optimisé pour permettre une écriture rapide et hautement disponible des données avec un débit élevé.



Sorted String Tables

Les SSTables sont immuables et, en tâche de fond, Cassandra fusionne périodiquement les SSTables dans une SSTable plus grande (*Compaction*).

Propriétés ACID (*Atomic*, *Consistent*, *Isolated*, *Durable*)

Les propriétés ACID permettent de garantir qu'une transaction est exécutée de façon fiable. Cassandra n'offre pas de transaction ACID .

Accès aux données : Insertion / Mise à jour

- Plusieurs colonnes peuvent être insérées en même temps.
- Lorsque des colonnes sont insérées ou mises à jour dans une famille de colonnes, l'application cliente précise la clé de la ligne pour identifier l'enregistrement à mettre à jour.
- La clé de la ligne peut être vue comme une clé primaire. Cependant, à la différence d'une clé primaire, il est possible d'insérer des données à une clé primaire déjà présente.

Accès aux données : Suppression

- La suppression d'une donnée du disque n'est pas immédiate. Des *tombstones* sont écrits sur les colonnes concernées, les colonnes marquées persistent pendant un laps de temps puis sont supprimées lors du processus de *decompaction* des SSTables.

- Une colonne supprimée peut réapparaitre si la routine de réparation de nœud n'est pas exécutée.

Accès aux données : Lecture

- (Rappel) Lorsqu'une requête de lecture d'une ligne est reçue par un nœud, la ligne doit être combinée de toutes les SSTables du nœud qui contiennent les colonnes de la ligne en question ainsi que de toutes les Memtables.

- Pour optimiser ce processus, Cassandra utilise une structure en mémoire appelée les *bloom filter* : chaque SSTable dispose d'un *bloom filter* associé qui est utilisé pour vérifier s'il existe des données dans la ligne avant de faire une recherche.

Accès aux données : Cohérence

Question

Comment est maintenue la cohérence des données et comment les lignes de données sont synchronisées entre tous les réplicas ?

Cassandra étend le concept de cohérence éventuelle en offrant une consistance réglable.
$\rightarrow$ Juste milieu entre le temps d'exécution de leurs requêtes et la consistance qu'ils souhaitent avoir sur les résultats obtenus.

Le niveau de consistance précise :

- sur combien de réplicas doit être écrite la donnée avant d'être acquitté du succès de l'opération. (écriture)
- combien de réplicas doivent répondre avant que le résultat ne soit fourni au client. (lecture)

Modèle de données

Une famille de colonne (ici "Etudiants") est l'équivalent d'une table en SGBD. On retrouve deux types de colonnes :
- statique : les colonnes sont définies lors de la création ou modification de la famille de colonnes ;
- dynamique : les colonnes sont définies lors de la création ou modification d'une ligne.

Etudiants

CLE	Nom	Prénom	Age	Taille
	Moulin	Mathieu	16	180

CLE	Nom	Prénom	Age
	Dupond	Jean	23

CLE	Nom	Prénom	Age	Taille
	Martin	Simon	22	175

Langage de requête : Creation

Le langage de requête spécifique à Cassandra s'appelle CQL (Cassandra Query Language). Des implémentations de ce langage de requête existe en Java, Python, C++.

Voici un exemple de création d'un espace de clés avec une famille en colonne.

```
CREATE KEYSPACE MonEspaceDeCle
  WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 3 };

USE MonEspaceDeCle;

CREATE COLUMNFAMILY MesColonnes (id text, Nom text, Prenom text, PRIMARY KEY(id));

INSERT INTO MesColonnes (id, Nom, Prenom) VALUES ('1', 'Doe', 'John');
```

Langage de requête : Synopsis

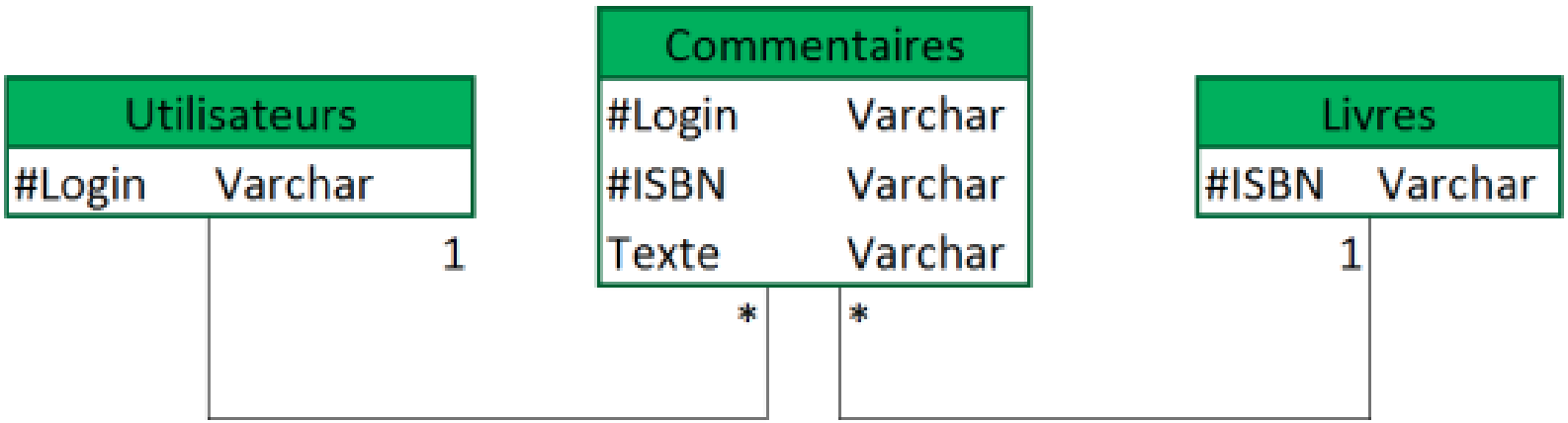
Voici le synopsis classique d'une requête utilisant du CQL

```
SELECT * | select_expression | DISTINCT partition
FROM [keyspace_name.] table_name
[WHERE partition_value
   [AND clustering_filters
   [AND static_filters]]]
[ORDER BY PK_column_name ASC|DESC]
[LIMIT N]
[ALLOW FILTERING]
```

Contrer l'absence de Jointure : la dénormalisation

Les jointures n'existent pas avec Cassandra, à cause de la distribution des données. Il faut donc "dénormaliser" les modèles.

Par exemple, prenons un ensemble de lecteurs (ici appelés "utilisateurs"), un ensemble de livres, et des commentaires laissés par les lecteurs sur les livres. Un lecteur peut commenter plusieurs livres et un livre peut être commenté par plusieurs lecteurs. Ce qui nous donne la forme normale suivante :

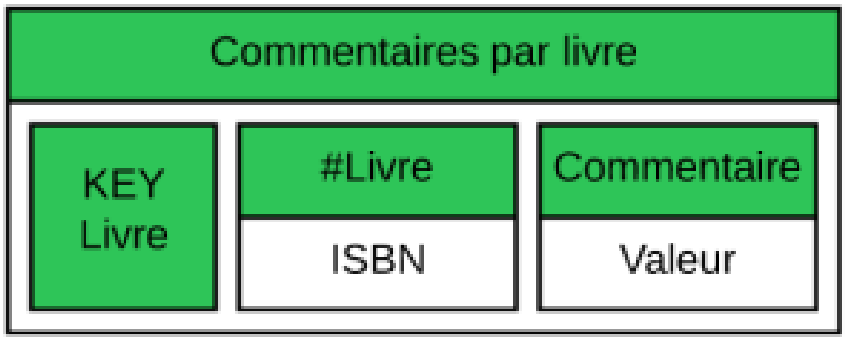


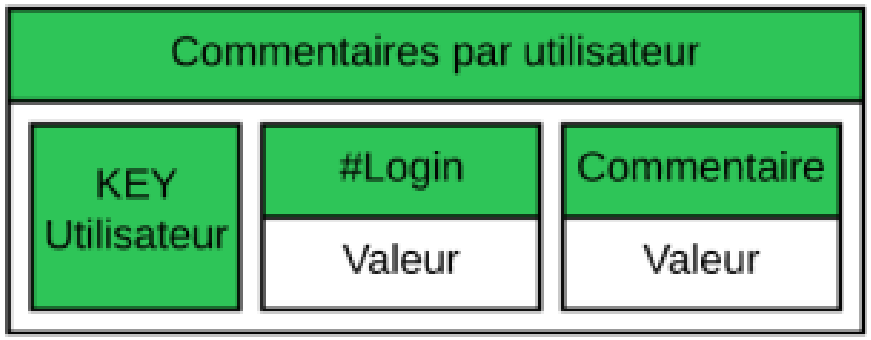
Contrer l'absence de Jointure : la dénormalisation

Problème : On souhaite obtenir tous les commentaires d'un livre et tous les commentaires d'un lecteur.
Solution : On crée 2 familles de colonnes.

```
CREATE COLUMNFAMILY comments_by_book (
  id text,
  book_isbn text,
  comment text,
  PRIMARY KEY(id));
```

```
CREATE COLUMNFAMILY comments_by_user (
  id text,
  user_login text,
  comment text,
  PRIMARY KEY(id));
```





Contrer l'absence de Jointure : la dénormalisation

Problème : On souhaite maintenant ajouter le commentaire "Pas mal pour un premier tome." de "voldemortDu31" sur le livre "Harry Potter à l'école des sorciers".
Solution : On l'ajoute dans les deux familles de colonnes.

```
INSERT INTO comments_by_book(id, book_isbn, comment) VALUES ('1', ' 2-07-051842-6',
'Pas mal pour un premier tome.');
```

```
INSERT INTO comments_by_user(id, user_login, comment) VALUES ('a', 'voldemortDu31',
'Pas mal pour un premier tome.');
```

Concurrents à Cassandra

- Dans la catégorie NoSQL des SGBD les plus populaires, Cassandra prend la deuxième place du palmarès juste derrière *MongoDB*
- Concurrents principaux en SGBD orientés colonnes : *HBase*, *Google Big Table*, *Microsoft Azure Cosmos DB*...

Les mots d'un architecte JAVA en cabinet IT, Abdelmajid Lali

"Cassandra associe le meilleur de Google Big Table (à savoir son modèle de données orienté colonnes et son mécanisme de persistance) et d'Amazon DynamoDB (avec son architecture distribuée où tous les nœuds sont équivalents)"

The End