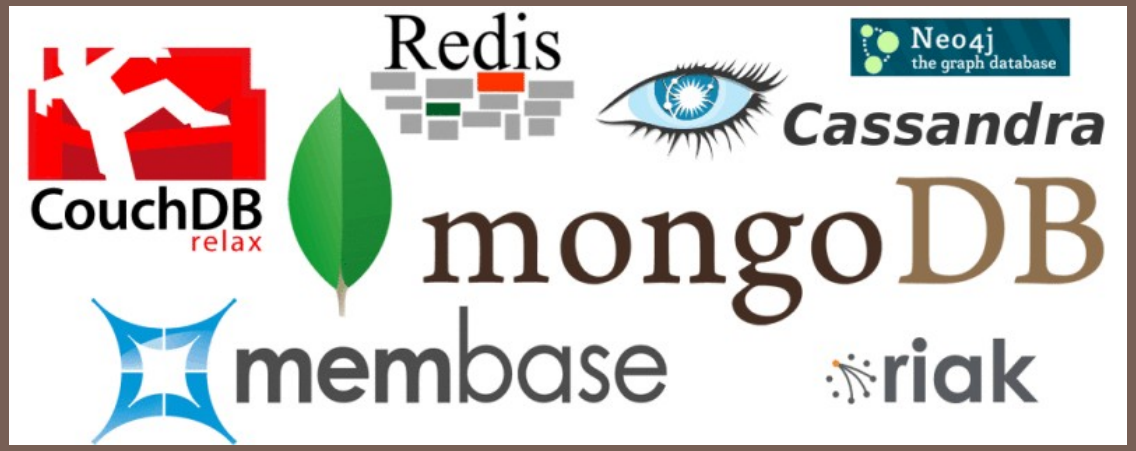

NoSQL : Cassandra

BD et SI

Frédéric Moal
Master 1 MIAGE

Qu’est-ce que Cassandra ?

- Cassandra, conçu à l’origine par Facebook, maintenant un projet de la fondation Apache, distribué par la société Datastax
  - Initialement basé sur le système BigTable de Google (stockage orienté colonnes)
  - A fortement évolué vers un modèle relationnel étendu (N1NF = Non First Normal Form)
- Un des rares systèmes NoSQL à proposer un typage fort
- Un langage de définition de schéma et de requêtes, CQL (pas de jointure)

Qu’est-ce que Cassandra ?

- Construit dès l’origine comme un système scalable et distribué
  - Propose des méthodes de passage à l’échelle inspirées du système Dynamo (Amazon)
  - distribution par hachage (consistent hashing)
  - tolérance aux pannes par réplication en mode multi-nœuds
- Très utilisé, réputé très performant.

Installation

- Serveur(s) Docker :

  docker run --name cassandra -d cassandra:tag

  - En cluster :

    docker run --name cassandra2 -d --link cassandra:cassandra cassandra:tag

- Client cqlsh docker :

  docker run -it --link cassandra:cassandra --rm cassandra cqlsh cassandra

  docker exec -it cassandra cqlsh localhost

Cassandra : modèle

- Cassandra gère des bases (keyspaces), contenant des tables (a.k.a. column families)
- Une table contient des lignes (rows) constituées de valeurs simples ou complexes
- Perspective A
  - c’est du relationnel étendu en rompant la première règle de normalisation (type atomique)
- Perspective B
  - chaque ligne est un document structuré
- Rem : le vocabulaire confus et parfois déroutant : columns, column families, etc.

Cassandra : structure

- La structure de base est la paire (clé, valeur)
  - Clé : un identifiant
  - Valeur atomique (entier, chaîne de caractères) ou complexe (dictionnaire, ensemble, liste) ou un type nommé
- Une ligne (row, document) est un identifiant (row key) associé à un ensemble de paires (clé, valeur)
- Les lignes sont typées par un schéma, y compris les données imbriquées
- Cassandra n'autorise pas (au moins avec CQL) l'insertion de données ne respectant pas le schéma !
- D'autres fonctionnalités : le versionnement…

Cassandra : bases

- Une table (anciennement column family) est un ensemble de rows conformes au schéma de la table
- Des notions de colonnes et super-colonnes sont (semble-t-il) en voie de disparition.
- Une base est un ensemble de tables. On l’appelle keyspace

Cassandra : conception

- Cassandra ne propose pas de jointure (NoSQL)
- Il est donc préférable de regrouper les données le plus possible dans des lignes (notion de document autonome)
- La conception devrait se baser sur les requêtes les plus fréquentes de l'application
- Au pire, on représente les mêmes données sous plusieurs formes (redondance)
  - Tous les films, avec leurs réalisateurs et acteurs
  - Tous les artistes, avec les films qu'ils ont tournés ou dans lesquels ils ont joué

Un exemple : schéma

- Création d'un keyspace

```
CREATE KEYSPACE IF NOT EXISTS Movies
WITH REPLICATION = {
'class' : 'SimpleStrategy',
'replication_factor': 3
};
```

- Création d'une table (sans imbrication)

```
create table artists (id text,
last_name text, first_name text,
birth_date int,
primary key (id)
);
```

Exemple : insert

- A la SQL :

```
insert into artists (id, last_name, first_name,
birth_date)
values ('artist1', 'Depardieu', 'Gerard', 1948);
insert into artists (id, last_name, first_name,
birth_date)
values ('artist2', 'Baye', 'Nathalie', 1948);
insert into artists (id, last_name, first_name)
values ('artist3', 'Marceau', 'Sophie');
```

- En JSON :

```
insert into artists JSON '{
"id": "artist1",
"last_name": "Coppola",
"first_name": "Sofia",
"birth_date": "1971"
}'
```

Exemple : select

- select * from artists;

```
 id          | last_name    | first_name      | birth_date
-------------+--------------+-----------------------------
  'artist1' | Depardieu    | Gérard          | 1948
  'artist2' | Baye         | Nathalie        | 1948
  'artist3' | Marceau      | Sophie          | null
```

Exemple : imbrication

- Création d'un type :

```
create type artist (id text,last_name text,
first_name text,
birth_date int,
```

- Table des films :

```
create table movies (id text,
title text,
Year int,
genre text,
country text,
director frozen<artist>,
primary key (id) );
```

Exemple : insert

- Insert en JSON :

```
INSERT INTO movies JSON '{
"id": "movie:1",
"title": "Vertigo",
"year": 1958,
"genre": "drama",
"country": "USA",
"director": {
  "id": "artist:3",
  "last_name": "Hitchcock",
  "first_name": "Alfred",
  "birth_date": "1899"
 },
}';
```

Exemple

- Au final :

```
create table movies (id text,
title text,
Year int,
genre text,
country text,
director frozen<artist>,
actors set <frozen<artist>>,
primary key (id) );
```

Exemple

- Au final :

```
INSERT INTO movies JSON '{
"id": "movie:1",
"title": "Vertigo",
"year": 1958,
"genre": "drama",
"country": "USA",
"director": { "id": "a:3",
              "last_name": "Hitchcock",
              "first_name": "Alfred",
              "birth_date": 1899  },
"actors": [ { "id":"a:4" }, { "id":"a:5" } ]
}';
```

Exemple

- Au final :

```
select * from movies;
```

```
movie:1 | {{id: 'a:4', last_name: null,
first_name: null, birth_date: null},
{id: 'a:5', last_name: null, first_name:
null, birth_date: null}} |     USA |
{id: 'a:3', last_name: 'Hitchcock',
first_name: 'Alfred', birth_date: 1899}
| drama | Vertigo | 1958
```