NFA 008

le cnam

Introduction à NoSQL et MongoDB
25/05/2013

NoSQL, c'est à dire ?

- Les bases de données NoSQL restent des bases de données mais on met l'accent sur
  - L'aspect NON-relationnel
  - L'architecture distribuée
  - La rapidité d'exécution
  - L'absence de SQL comme DML
- Utilisé pour des *data stores* de grande à très grande taille

Les grands noms du NoSQL

- Hadoop, Cassandra (Java)
- MongoDB, CouchDB
- Redis
- MemcacheDB

MongoDB

- Existe depuis 2007, développé en C++
- Les données existent sous forme de documents BSON (Binary JSON)
- En ligne de commande : la notation JSON
- Utilisable avec les principaux langages de développement (C, Java, .NET, PHP, Python...) via des pilotes

JSON

- Une notation JavaScript pour décrire un objet

```
{
    "employees": [
        {
            "prenom": "Jean",
            "nom": "Bon"
        },
        {
            "prenom": "Eric",
            "nom": "Rack"
        },
        {
            "prenom": "Raymond",
            "nom": "Godibi"
        }
    ]
}
```

Relationnel vs. MongoDB

- Terminologie différente, vous vous en doutez...

Relationnel	MongoDB
table	collection
tuple	document
champ	champ
index	index

Relationnel vs. MongoDB

- En SQL, le schéma constitue la fondation de la base de données (contraintes d'intégrité, clés primaires, clé étrangères...)
- MongoDB n'a pas de notion schéma (« schema-less »)

Relationnel vs. MongoDB

- Les tables ont une structure fixe qui ne varie pas d'un tuple à l'autre : tout auditeur a un nom, un prénom, un âge, même vide, même marqué NULL.
- Dans une collection, les documents peuvent varier : l'auditeur A a un nom, un prénom et un âge, l'auditeur A2 peut avoir en plus de ces trois champs une adresse.

Relationnel vs. MongoDB

- Les champs d'une table ont le même type de données pour tous les tuples.
- Les documents peuvent avoir un attribut de même nom contenant un type de valeur différent : l'âge d'un auditeur peut être un numérique dans un document, une chaîne de caractères dans un autre.

  Possible mais à éviter, évidemment !

Dé-normalisation

- MongoDB encourage la *dé-normalisation*, vous foulez donc au pied tout ce que vous avez appris en NFA 008 !
- La dé-normalisation favorise l'extraction rapide des documents, souvenez-vous que la rapidité de traitement est placée au centre des ambitions du NoSQL

La dé-normalisation en exemple – relation un-à-plusieurs

- Un livre a un seul auteur (pour simplifier)
- Table **AUTEUR** : id, nom, prenom
- Table **LIVRE** : id, titre, isbn, id_auteur (not null)
- En relationnel, on a un lien entre le livre et son auteur (1:1), id_auteur est une clé étrangère
- En NoSQL, on va embarquer l'auteur dans le document *livre* pour matérialiser ce lien

La dé-normalisation en exemple

```
{
        "_id": 1,
        "titre": "Orages  d'acier",
        "isbn": "2253048429",
        "auteur": {
            "nom": "Jünger",
            "prenom": "Ernst"
        },
        "editeur": {
            "nom": "Le Livre de Poche"
        }
}
```

La dé-normalisation en exemple

- La tentation est grande pour le développeur de faire toujours plus de dé-normalisation et donc à terme de produire des « monstro-documents » contenant tout et (surtout) n'importe quoi...
- MongoDB limite la taille des documents à 16M et à 100 niveaux d'imbrication

La dé-normalisation – relations un-à-plusieurs

```
{
    "_id": 1,
    "titre": "PHP5 avancé",
    "isbn": "2212134355",
    "auteurs": [
        {
            "nom": "De Geyer",
            "prenom": "Cyril"
        },
        {
            "nom": "Daspet",
            "prenom": "Eric"
        },
        {
            "nom": "Séguy",
            "prenom": "Damien"
        }
    ]
}
```

La dé-normalisation : avantages

- Une opération faite par le serveur sur un seul document et pas plusieurs

  → Donc un seul accès au disque
- Performances accrues pour les lectures...
- Mais potentiellement ralenties sur les mises à jour

La normalisation est possible

- Il est possible de faire référence à un document depuis un autre, comme on le fait en relationnel
- Notre document livre contiendra une référence vers son éditeur
- Mais ceci nous oblige à faire deux requêtes :
  - La première pour obtenir le livre
  - La seconde vers la collection *editeurs*

La normalisation reste possible

Normaliser ou dé-normaliser ?

- La réponse est...ça dépend !
  - Votre application est elle amenée à faire plus de lectures que d'écritures ou de modifications ?
  - Quelle est la volumétrie de vos données et comment celle-ci va-t-elle évoluer ?
  - Quelles requêtes ferez-vous le plus souvent ?
  - Normalisez si vous souhaitez privilégier l'intégrité des données
  - Dé-normalisez si vous privilégiez la rapidité

MongoDB sur le poste de travail

- Une interface en ligne de commande : le shell
  - On y exécute des commandes Mongo qui font usage de la notation JSON
- Des interfaces web :
  - phpMoAdmin
  - RockMongo (ma préférée !)

MongoDB vs SQL

- Lister les bases de données :
  - SQL : SHOW DATABASES
  - MongoDB : show dbs
- Se brancher sur une base de données :
  - SQL : use mabase
  - MongoDB : use mabase

MongoDB vs SQL

- Créer une table SQL :
  - CREATE TABLE livre (id tinyint unsigned primary key, titre varchar(20), isbn bigint unsigned)
- Créer une collection MongoDB :
  - db.createCollection('livre')

MongoDB vs SQL

- Lister des tables SQL :
  - SHOW TABLES ;
- Lister des collections MongoDB :
  - show collections ;

MongoDB vs SQL - insertions

- SQL :
  - INSERT INTO livre VALUES (1, 'Orages d'acier', '12134567891') ;
- MongoDB :
  - db.livre.insert({id:"1", titre:"Orages d'acier", isbn:"1234567891"});

Vous notez que nous faisons usage de la notation JSON dans le shell Mongo.

MongoDB vs SQL - projections

- SQL : SELECT * FROM livre LIMIT 5;
- MongoDB : db.livre.find().limit(5);
- SQL : SELECT titre, isbn FROM livre
- MongoDB : db.livre.find({}, {titre:1, isbn:1});
- SQL : SELECT titre FROM livre WHERE id = 1
- MongoDB : db.livre.find({id:"1"},{titre:1});

MongoDB vs SQL - projections

- SQL : SELECT titre FROM livre ORDER BY id DESC;
- MongoDB : db.livre.find({}, {titre:1}).sort({id:-1});
- SQL : SELECT titre FROM livre WHERE titre LIKE 'Or %'
- MongoDB : db.livre.find({titre:/^Or/});
- SQL : SELECT DISTINCT(titre) FROM livre
- MongoDB : db.livre.distinct("titre");

MongoDB vs SQL - projections

- SQL : SELECT COUNT(*) FROM livre
- MongoDB : db.livre.count();
- SQL : SELECT * FROM livre WHERE titre like 'Or %' OR ID = 1 ;
- MongoDB : db.livre.find({ $or: [{titre:/^Or/}, {id:"1"}]});

MongoDB vs SQL – mises à jour

- SQL : UPDATE livre SET titre= 'Jeux Africains' WHERE id = 1 ;
- MongoDB : db.livre.update({id:"1"}, {$set: {titre:"Jeux Africains"}});
- SQL : UPDATE livre SET prix = prix + 3 WHERE id =1
- MongoDB : db.livre.update({id:"1"}, {$inc:{prix:3}});

MongoDB vs SQL – suppressions

- SQL : DELETE FROM livre
- MongoDB : db.livre.remove();
- SQL : DELETE FROM livre WHERE id = 1
- MongoDB : db.livre.remove({id:"1"});
- SQL : DELETE FROM livre WHERE id >= 1 LIMIT 1
- MongoDB : db.livre.remove({id: {$gte: 1}}, true);

MongoDB vs SQL – agrégation

- MongoDB met en place les mêmes fonctionnalités de base que SQL pour ce qui est des fonctions agrégatives :
- $max
- $min
- $avg
- $group
- $sum

MongoDB – les index

- Les index existent aussi dans MongoDB :
  - Unique
  - Composés
  - Géospatiaux
  - Texte (depuis 2.4)
- Tout ceci existe aussi dans MySQL
- Pour poser un index sur le champ d'une collection : *ensureIndex*

MongoDB - particularités

- _id est le champ qui sert de clé primaire par défaut
- Si on ne précise pas de valeur pour le champ _id, MongoDB en attribue une (longueur = 12 octets)
- La gestion des identifiants auto-incrémentés que nous avons utilisé avec MySQL n'est pas aussi intuitive en MongoDB, il faut des développements supplémentaires !
- Le type utilisé par MongoDB pour gérer les dates est ISODate

MongoDB - bilan

- Pas d'intégrité référentielle
- Pas de jointures ni de transactions
- Réplication et partitionnement sont implémentés
- Schéma flexible (même si on parle de *schema-less design*)
- Privilégie la rapidité en minimisant les lectures sur le disque (et donc la dé-normalisation)
- Pratique pour le Web où le temps de réponse est un paramètre critique

MongoDB - bilan

- Nécessite de repenser sa façon de raisonner : ne pas chercher à faire du relationnel en NoSQL
- Technologie encore jeune mais qui évolue avec le concours des plus grandes entreprises
- Des pilotes pour de nombreux langages
- Une communauté active
- A démarrer sur de « petits » projets pour une première prise en main...à vous de jouer !

MongoDB – Ressources en ligne

- Valider du JSON en ligne : jsonlint.com
- RockMongo : rockmongo.com
- phpMoAdmin : phpmoadmin.com
- Et, évidemment : mongodb.org