mongoDB
FOR GIANT IDEAS

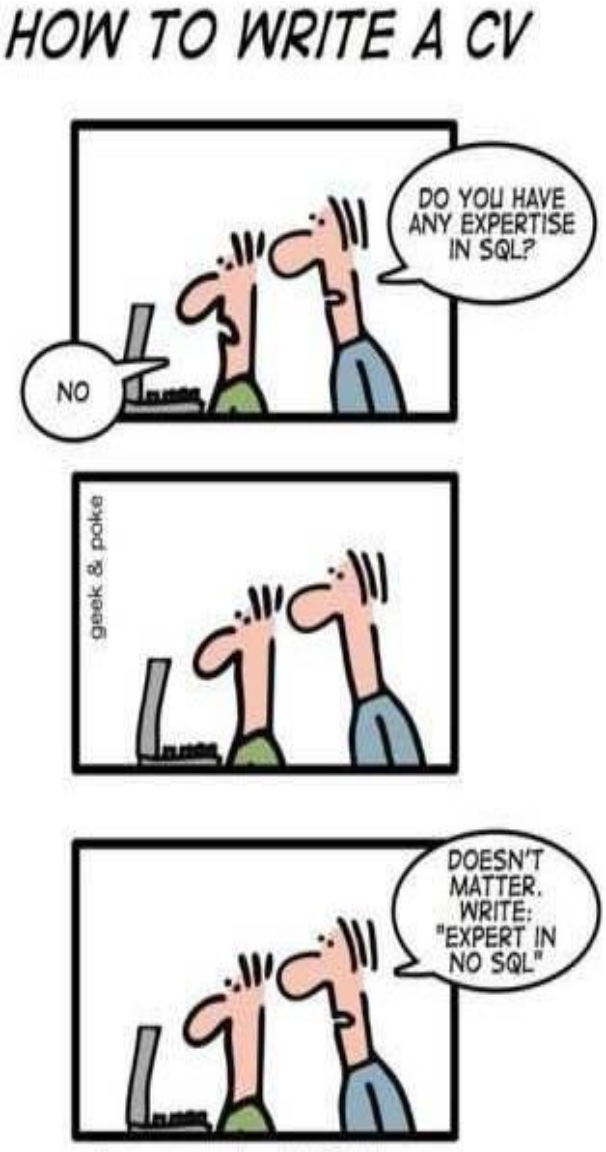
mongoDB
{ name: mongo, type: DB }

Cours de Bases de données NOSQL (MongoDB)

madaniabdellah@gmail.com

Plan du cours

- **Rappels :**
  - **JSON**
  - **Bases de Données Relationnelles**
- Introduction : Mouvement NoSQL
- MongoDB
  - Caractéristiques de MongoDB
  - Requêtes avec MongoDB
  - MongoDB et Java
  - MongoDB et PHP
  - Réplication et reprise sur panne dans MongoDB
  - Sharding dans MongoDB
  - TextSearch



JSON

madaniabdellah@gmail.com

Introduction

- JSON (JavaScript Object Notation – Notation Objet issue de JavaScript)
- JSON est un format de données, basé sur du texte facile à lire ou à écrire pour des humains.
- JSON est largement utilisé pour stocker des données ou échanger des données notamment sur Internet
- JSON connait un fort engouement (admiration) car il possède quelques points forts :
  - standard ouvert
  - syntaxe simple et compacte
  - facile à parser et à écrire
  - format offrant une structuration des données compacte

Introduction


```
{
    "_id" : 1,
    "title" : "Toy Story (1995)",
    "genres" : "Animation|Children's|Comedy"
}
{
    "_id" : 2,
    "title" : "Jumanji (1995)",
    "genres" : "Adventure|Children's|Fantasy«
}
{
    "_id" : 3,
    "title" : "Grumpier Old Men (1995)",
    "genres" : "Comedy|Romance"
}
```

Syntaxe

- La syntaxe de JSON est très simple ce qui explique une partie de son succès.
- Elle ne définit que deux types de structure :
  - un **objet** qui est un ensemble de paires clé/valeur
  - un **tableau**.
- JSON définit 4 types de données prédéfinis : **string**, **number**, **boolean** (**true** et **false**) et **null**.

Syntaxe

- **L'objet**

Il contient un membre ou une liste de membres, chaque membre étant de la forme:

"nom" : "valeur"

La syntaxe de l'objet est:

{ **membre, membre, ...** }

- **Le tableau**

Contient une ou plusieurs valeurs séparées par des virgules.

[valeur, valeur, ….]

- **Les valeurs**

Une valeur peut être: un objet, un tableau, un littéral (chaîne, nombre, true, false, null).

JSON vs XML

```
{
 "menu": "Fichier",
 "commandes": [
    {
       "titre": "Nouveau",
       "action":"CreateDoc"
    },
    {
       "titre": "Ouvrir",
       "action": "OpenDoc"
    },
    {
       "titre": "Fermer",
       "action": "CloseDoc"
    }
  ]
}
```

```
<?xml version="1.0" ?>
<racine>
 <menu>Fichier</menu>
 <commandes>
   <item>
      <titre>Nouveau</titre>
      <action>CreateDoc</action>
   </item>
   <item>
      <titre>Ouvrir</titre>
      <action>OpenDoc</action>
   </item>
   <item>
      <titre>Fermer</titre>
      <action>CloseDoc</action>
   </item>
 </commandes>
</racine>
```

JSON vs XML

JSON est comme XML parce que :

- Les deux JSON et XML sont "auto-descriptif" (lisible par l'homme)
- JSON et XML sont tous les deux hiérarchiques (valeurs dans les valeurs)
- JSON et XML peuvent être analysés et utilisés par de nombreux langages de programmation
- JSON et XML peuvent être récupérés avec XMLHttpRequest

JSON vs XML

JSON est différent de XML :

- JSON n'utilise pas de balise de fin
- JSON est plus court
- JSON est plus rapide à lire et à écrire
- JSON peut utiliser des tableaux
- Mais, la plus grande différence est:

  XML doit être analysé avec un analyseur XML. JSON peut être analysé par une fonction JavaScript standard.

Exemple d'utilisation de JSON

```
<body>
<script type="text/javascript">
    var text = '{ "employees" : '+
    '[' +
            '{ "firstName":"John" , "lastName":"Doe" },' +
            '{ "firstName":"Anna" , "lastName":"Smith" },' +
            '{ "firstName":"Peter" , "lastName":"Jones" }'+
    ']}';
    var obj = JSON.parse(text);
    document.write(obj.employees[0].firstName+'
    '+obj.employees[0].lastName);
</script>
</body>
```

Bases de Données Relationnelles

madaniabdellah@gmail.com

Bases de Données Relationnelles

Depuis plus de 40 ans maintenant, la méthode la plus communément utilisée pour stocker des informations de manière permanente est le **modèle relationnel** , avec les Systèmes de Gestion de Bases de données Relationnels (SGBDR) comme :

- MySQL
- SQLite
- Oracle
- SQL Server
- MariaDB.

Bases de Données Relationnelles

- Le modèle relationnel consiste à stocker l'information dans des tables, définies par leur **schéma** (leurs différentes colonnes, clés primaires, clés étrangères).
- Cela permet de ne pas stocker l'information plusieurs fois, et de pouvoir facilement consolider les données avec des requêtes SQL et des jointures.

Bases de Données Relationnelles

- Une base de données sert à stocker des informations, les consulter et les mettre à jour.
- Les informations sont stockées de manière non redondantes
- Dans un programme sur un serveur web, ces informations seront stockées sur le serveur.
- Une base de données relationnelle enregistre les informations ligne après lignes ➔ notion de table.
- Les SGBDR utilisent un langage de requêtes de la famille SQL (Structured Query Language).

Bases de Données Relationnelles

- Les bases de données relationnelles respectent le modèle ACID :
  - Atomicité
    - Tout ou rien (utilisation des transactions)
  - Cohérence
    - Les transactions qui modifient l'état de la base font en sorte que les données restent cohérentes
    - Contraintes d'intégrité référentielle (clés, types, valeurs par défaut, …)
  - Isolation
    - Une transaction ne peut pas accéder aux données mise à jour par une autre transaction avant qu'elle ne soit validée par son propriétaire
  - Durabilité
    - Toute transaction validée (commitée) est assurée d'être prise en compte quelque soit la panne.

Bases de Données Relationnelles

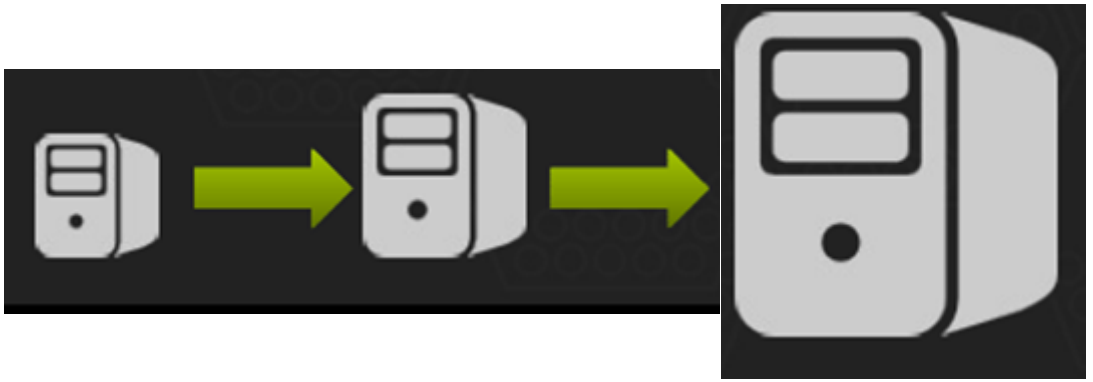
Limites

- Pas de Scalabilité (montée en charge)
- Le modèle de consistance des RDBMS empêche l'utilisation de plusieurs machines pour répartir la charge (au moins en écriture)

Bases de Données Relationnelles

Limites

- Pas de Scalabilité
- Pour augmenter la performance d'accès à la base, pas d'autres moyens que d'acheter un plus gros serveur, puis un autre, puis un autre, …



Bases de Données Relationnelles

Limites

- Une autre limite : son schéma est statique. Prenons par exemple un site de e-commerce qui vendrait toute sorte de choses :
  - De l'électroménager (machine à laver, aspirateur…)
  - Livres, DVD, Jeux vidéos
  - Matériel audio/vidéo/photo, informatique
  - Places de concert
  - Matériel de jardin (tondeuse à gazon)
- Chaque article possède ses propres caractéristiques :
  - prix
  - fabricant
  - nombre de tours par minute (machine à laver)
  - volume du tambour (machine à laver)
  - …

Bases de Données Relationnelles

Limites

- Pour stocker toutes ces informations, deux solutions :
  - soit on créé un nombre interminable de colonnes pour la table **catalogue**, en étant parfois obligé d'ajouter de nouvelles colonnes pour les nouveaux types de produits ➔ Table trouée
  - soit on utilise plusieurs tables, avec un nombre importants de jointures.

Bases de Données Relationnelles

Solution

- C'est à partir de ce constat (limites du modèle relationnel) qu'une nouvelle idée a émergé au début des années 2000
- Cette idée c'est de se passer du schéma
- Utiliser une nouvelle structure pouvant ainsi stocker des objets assez hétérogènes.
- Divers projets existent, les principaux sont :
  - **MongoDB**
  - **CouchDB**
  - **Cassandra**
  - **BigTable**
  - **HBase**