Bases de Données: Nouveaux paradigmes

Erick STATTNER

Maître de Conférences en Informatique

Université des Antilles

erick.stattner@univ-antilles.fr

www.erickstattner.com

Description de l'enseignement

Objectifs pédagogiques:

- Se familiariser aux nouveaux paradigmes des BD
- Maitriser les concepts autour du NoSQL
- Construire/manipuler une base NoSQL avec MongoDB
- Sensibiliser aux perspectives ouvertes avec Big Data

Organisation:

- 15h
- 1 CC

Sommaire

Chapitre I. Introduction

I. Introduction

Présentation

Contexte:

- Depuis les années 2000, un déluge de données considérable
- De tels volumes ne sont plus gérables par des solutions traditionnelles
- Problèmes des 3V
  - **V**olume
  - **V**élocité
  - **V**ariété

Solution

- Nouvelle façon de gérer les données
- Emergence du NoSQL

I. Introduction

Présentation

NoSQL

- **Not Only SQL**
- Apparait dans les années 2010
- Famille de SGBD qui s'écarte du paradigme classique des BD relationnelles

Objectifs

- Se soustraire aux contraintes lourdes du relationnel
- Dénormaliser les modèles
- Favoriser les performances

I. Introduction

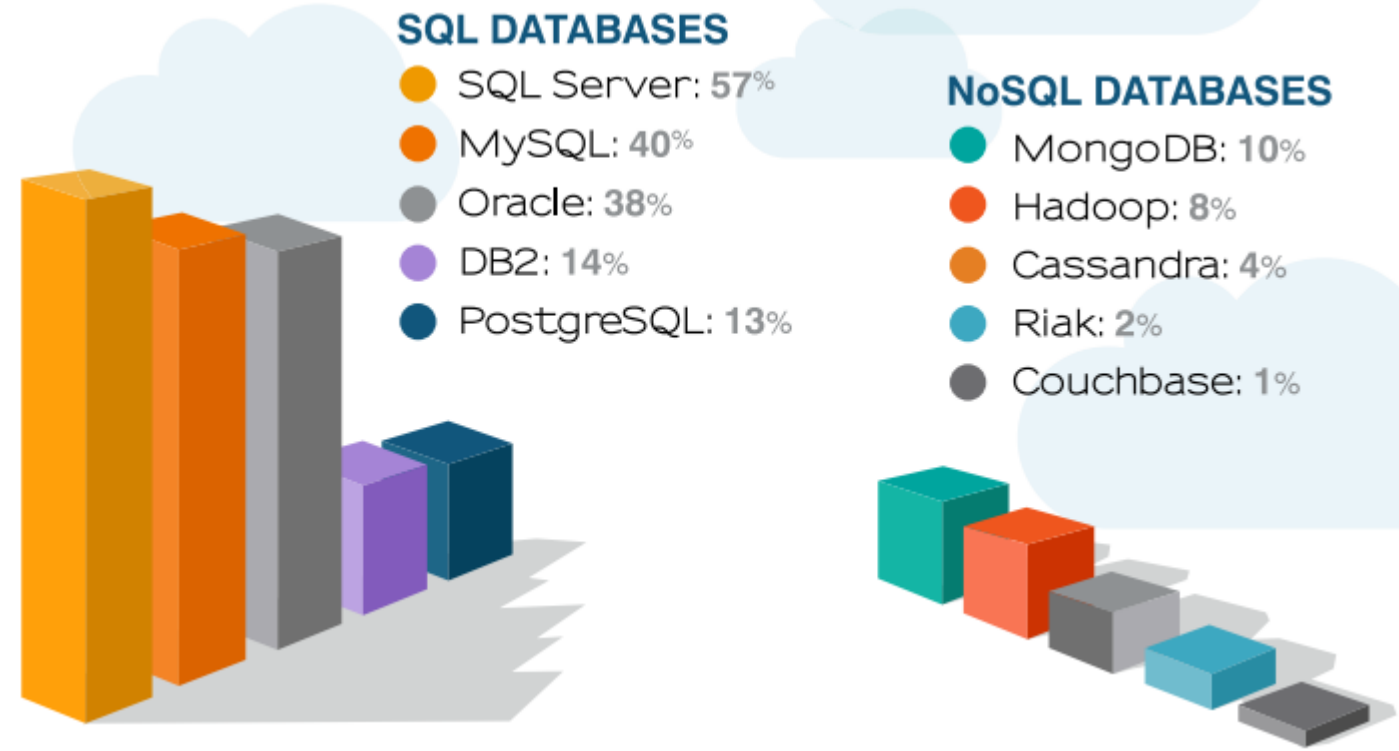
Présentation

SQL vs NoSQL

	SQL	NoSQL
Type	Relational	Non-Relational
Data	Structured Data stored in Tables	Un-structured stored in JSON files but the graph database does supports relationship
Schema	Static	Dynamic
Scalability	Vertical	Horizantal
Language	Structured Query Language	Un-structured Query Language
Joins	Helpful to design complex queries	No joins, Don't have the powerful interface to prepare complex query
OLTP	Recommended and best suited for OLTP systems	Less likely to be considered for OLTP system
Support	Great support	community depedent, they are expanding the support model
Integrated Caching	Supports In-line memory(SQL2014 and SQL 2016)	Supports integrated caching
flexible	rigid schema bound to relationship	Non-rigid schema and flexible
Transaction	ACID	CAP theorem

I. Introduction

Présentation



Source: STRATOSCALE

Chapitre II.
Principes du NoSQL

II. Principes du NoSQL

Familles de bases NoSQL

Trois principales familles de bases NoSQL

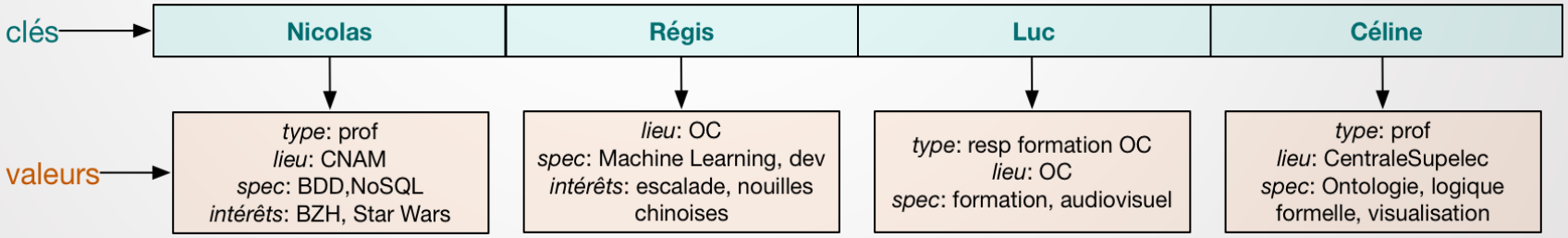
- Les clés-valeurs
- Colonne
- Documents

II. Principes du NoSQL

Familles de bases NoSQL

Clés-valeurs

- Repose sur le couple clé-valeur
- Fonctionne comme une table de hachage
- Solution simple et efficace
- La valeur contient des données hétérogènes
- Donc pas de schéma, pas de structure



source: Openclassrooms

II. Principes du NoSQL

Familles de bases NoSQL

Colonnes

- Se focalise sur les attributs
- Evite d'avoir à traiter les colonnes inutiles lors des requetes
- Adaptée pour traitements de masse (stats, calculs analytiques, …)
- Moins appropriée pour la lecture de données spécifiques

Stockage orienté colonnes

id	type
Nicolas	prof
Céline	prof
Luc	resp formation OC

id	lieu
Céline	Centrale Supelec
Nicolas	CNAM
Régis	OC
Luc	OC

id	spec
Nicolas	BDD
Nicolas	NoSQL
Régis	Machine Learning
Régis	Dev
Luc	formation
Luc	audiovisuel
Céline	Ontologie
Céline	logique formelle
Céline	visualisation

id	intérêts
Nicolas	BZH
Nicolas	Star Wars
Régis	escalade
Régis	nouilles chinoises

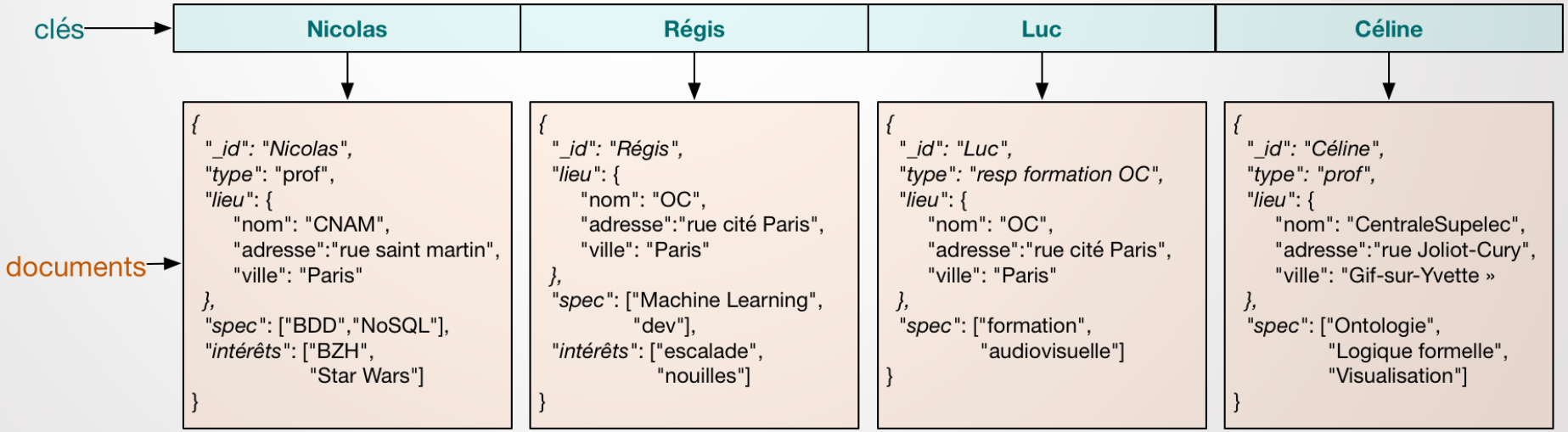
source: Openclassrooms

II. Principes du NoSQL

Familles de bases NoSQL

Documents

- Repose sur la notion de clé-valeur (également sur les champs)
- Les valeurs sont structurées
- Manipuler des informations avec données complexes (listes, imbrications, etc.)
- Langages d'interrogation riches



source: Openclassrooms

II. Principes du NoSQL

Théorème de CAP

Dans un SGBR, les transactions sont ACID

- **Atomicité**:
  Toutes les opérations d'une transaction sont effectuées, autrement aucune n'est réalisée.
- **Cohérence**
  Le contenu de la base doit être cohérent du début à la fin
- **Isolation**
  Les modifications d'une transaction ne sont visibles que quand celle-ci a été validée
- **Durabilité**
  Une fois la transaction validée, l'état de la base est permanent
  (quel que soit l'incident qui survient, panne, coupure, etc.)

II. Principes du NoSQL

Théorème de CAP

Les bases NoSQL relâchent certaines contraintes pour garantir les performances

- Elles sont **B.A.S.E**
  - **Basically Available**
    Le système garantie un taux de disponibilité de la donnée
  - **Soft-state**
    La base n'est pas nécessairement cohérente à tout instant
  - **Eventually consistent**
    La base atteindra, à terme, un état cohérent

II. Principes du NoSQL

Théorème de CAP

Pour caractériser les bases de données

- **3 propriétés CAP**
  - **Consistency (Cohérence)**
    Le contenu de la base doit être cohérent (une donnée n'a qu'un seul état).
  - **Availability (Disponibilité)**
    La donnée doit toujours être disponible
  - **Partition (Distribution)**
    Quel que soit le nombre de serveurs, toute requête doit fournir un résultat correct

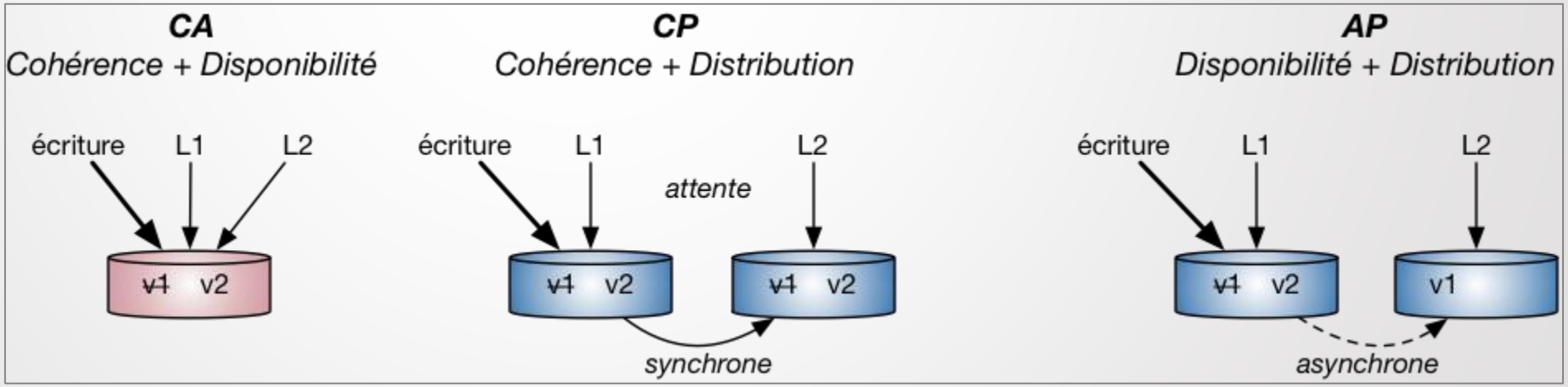
II. Principes du NoSQL

Théorème de CAP

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"

Eric A. Brewer, 2000



source: Openclassrooms

Chapitre III. MongoDB

III. MongoDB

Presentation

MongoDB

- Base de données NoSQL
- Sortie en 2009
- Orientée documents
- Pas de schémas, pas de structure de table, pas de typage
- Fournit un langage basé sur JavaScript

III. MongoDB

Presentation

Installation

- 3 étapes nécessaires
  - **Le serveur**
    https://www.mongodb.com/download-center#community
    le serveur se lance via l'exécutable ***mongod.exe***
  - **Un répertoire stockant les données**
    Par défaut
    - Windows: c:/data/db
    - Linux: /data/db
  - **Interface**
    - Une interpréteur de commande
      mongo.exe
    - Une interface graphique
      Robo 3T
      https://robomongo.org/download

III. MongoDB

Presentation

MongoDB

- Information modélisée dans format JSon
- Nécessité de dénormaliser les tables traditionnelles

id	titre
1	Her
2	Avengers

film_id	acteur_id
1	1
1	2
2	1

id	nom	prenom
1	Johansson	Scarlett
2	Phoenix	Joaquim

Base de données relationnelle classique
Normalisé

```
1 {_id: "Her", acteurs : [{nom:"Johansson", prenom:"Scarlett"}, {nom:"Phoenix",
  prenom:"Joaquim"}]}
2 {_id: "Avengers", acteurs : [{nom:"Johansson", prenom:"Scarlett"}]}
```

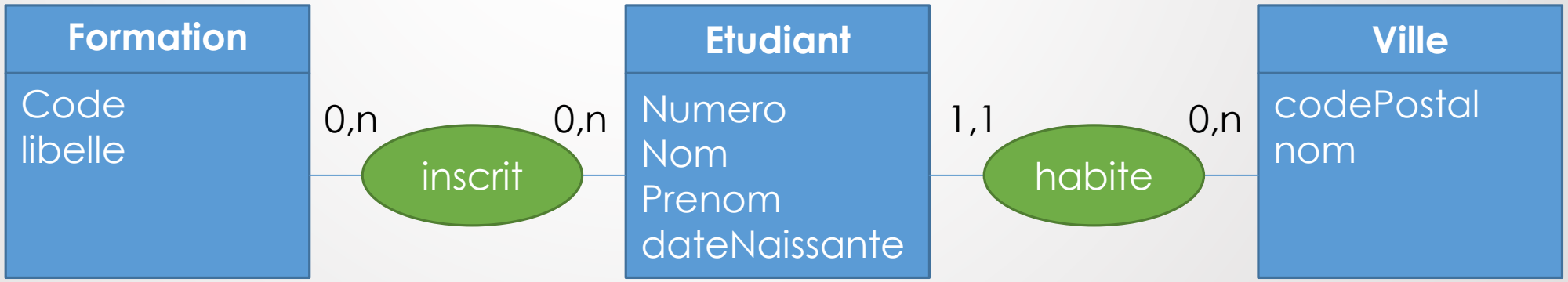
Base de données NoSQL
Dénormalisé

III. MongoDB

Presentation

Dénormalisation des schémas classiques

- Proscrire les jointures
- Fusionner les tables
- Pas d'algorithmes/pas de solutions miracles
- A l'issue, on identifie le ou les **collections**



III. MongoDB

Presentation

Eléments à prendre en compte pour identifier les collections

- Données fréquemment interrogées conjointement
- Données indépendantes
- Attention aux associations (*,n)-(*,n)
- Fréquences des mises à jour

Chapitre IV.
Requêtes sur MongoDB

IV. Requêtes sur MongoDB

Requetes

Interroger/Manipuler la base

- Une action sur une collection doit être préfixée du nom de la collection **db.<nomCollection>.<actionARealiser>**
- Actions
  - Insérer (*insert*)
  - Lire (*find*)
  - Mettre à jour (*update*)
  - Supprimer (*remove*)
- Possibilité d'utiliser du JavaScript
  - Facilite appels de fonctions
  - Création de librairies
  - Création de programme reposant sur Map/Reduce

IV. Requêtes sur MongoDB

Requetes

Insérer un document

- Commande
  db.<nom de la collection>.insert(<document>)
- Exemple

```
db.acteurs.insert(
   {nom:"Johansson",
   prenom:"Scarlett"}
)
```

- Dans une collection, les documents n'ont pas nécessairement la même structure

```
db.acteurs.insert(
   {nom:"Toto",
   prenom:"Alain" ,
   naissance:"1/1/1990"}
)
```

IV. Requêtes sur MongoDB

Requetes

Sélectionner un document

- Commande
  db.<nom de la collection>.find(<critère>)
- Exemples
  Sélectionner un enregistrement
  db.acteurs.findOne()

  Sélectionner tous les enregistrements
  db.acteurs.find({})

IV. Requêtes sur MongoDB

Requetés

Restriction sur une sélection

- Commande
  db.<nom de la collection>.find(<critère>)
- Exemple
  db.acteurs.find({ "nom" : "Toto" })
- avec plusieurs critères
  db.acteurs.find({
  "nom" : "Toto" ,
  "prenom" : "alain"
  })
- Avec objets imbriqués
  db.acteurs.find({
  "nom" : "Toto" ,
  "prenom" : "alain" ,
  "ville.codePostal" : "97180"
  })

IV. Requêtes sur MongoDB

Requetés

Restriction sur une sélection

- Conserver certaines colonnes dans le résultat
  db.<nom de la collection>.find(<critère>, <colonne>)
- Même principe: clé-valeur
- Exemple

```
db.acteurs.find(
{
   "nom" : "Toto" ,
   "prenom" : "alain"
} ,
{
   "nom" : 1 }
}
)
```

IV. Requêtes sur MongoDB

Requetés

Restriction sur une sélection

- Compter résultats
  db.<nom de la collection>.find(<critère>, <colonne>).count()
- Exemple

```
db.acteurs.find(
{
  "nom" : "Toto" ,
  "prenom" : "alain"
} ,
{
  "nom" : 1 }
}
).count()
```

IV. Requêtes sur MongoDB

Requetés

Restriction sur une sélection

- Valeurs distinctes
  db.<nom de la collection>.distinct.find(<critère>)
- Exemple
  db.acteurs.distinct ("ville.nom")

IV. Requêtes sur MongoDB

Requetés

Opérateurs arithmétiques

$gt, $gte	>, ≥	Plus grand que (*greater than*)	"a" : {"$gt" : 10}
$lt, $lte	<, ≤	Plus petit que (*less than*)	"a" : {"$lt" : 10}
$ne	≠	Différent de (*not equal*)	"a" : {"$ne" : 10}
$in, $nin	∈, ∉	Fait parti de (ou ne doit pas)	"a" : {"$in" : [10, 12, 15, 18] }
$or	∨	OU logique	"a" : {"$or" : [{"$gt" : 10}, {"$lt" : 5}] }
$and	∧	ET logique	"a" : {"$and" : [{"$lt" : 10}, {"$gt" : 5}] }
$not	¬	Négation	"a" : {"$not" : {"$lt" : 10} }
$exists	∃	La clé existe dans le document	"a" : {"$exists" : 1}
$size		test sur la taille d'une liste (uniquement par égalité)	"a" : {"$size" : 5}

Source: Openclassrooms - Liste des opérateurs ici

IV. Requêtes sur MongoDB

Requetés

Restriction sur une sélection

- Exemple 1

```
db.acteurs.find(
{
      "nom" : "Toto" ,
      "prenom" : "alain" ,
      "anneeNaissance"  : {$lt : 1950}
})
```

- Exemple 2

```
db.acteurs.find(
{
   "$or" :
         [
               {"prenom" : "alain"}
               {"ville.codePostal" : "97180"}
         ]
} )
```

IV. Requêtes sur MongoDB

Requetés

Mise à jour d'un document

- Commande
  db.<nom de la collection>.update(<critère>, <nouvelle valeur>)
- Exemple
  db.acteurs.update({nom:"toto"}, {$set : {prenom:"benoit"}})
- **ATTENTION**
  MongoDb ne met à jour que le premier document qu'il trouve avec le critère
  Solution: {multi:true}
  db.acteurs.update({nom:"toto"}, {$set : {prenom:"benoit"}}, {multi:true})

IV. Requêtes sur MongoDB

Requetés

Modifier structure

- Commande
  db.<nom de la collection>.update(<critère>, <nouvelle valeur>)
- Exemple

```
db.acteurs.update(
{
      "prenom" : "Scarlette" ,
      "nom" : "Johansson" ,
},
{
          $set : {note : 10}
}
)
```

IV. Requêtes sur MongoDB

Requetés

Suppression d'un document

- Commande
  db.<nom de la collection>.remove(<critère>, <nouvelle valeur>)
- Supprime tous les documents qui vérifient le critère
- Exemple
  db.acteurs.remove({nom:"toto"})
  db.acteurs.remove({nom:"toto"} , {multi:true})

Chapitre V. Limites

V. Limites

Limites

Pourquoi faire encore des BD relationnelle?

- Gérer l'intégrité des données
- Possibilité de requêtes complexes
  - Jointure
  - Agrégation
- Langage de haut niveau

V. Limites

Limites

Attention aux choix qui impactent

- La conception
  modélisation des données, modélisation des process, …
- L'application
  connexion, requêtes, fonctionnalités, …
- Les équipes de dev.
  langage, drivers, méthodes, formation, …
- Le système
  sécurité, distribution, flux, …
- Le cout
  cout administration, cloud, stockage, …

V. Limites

Limites

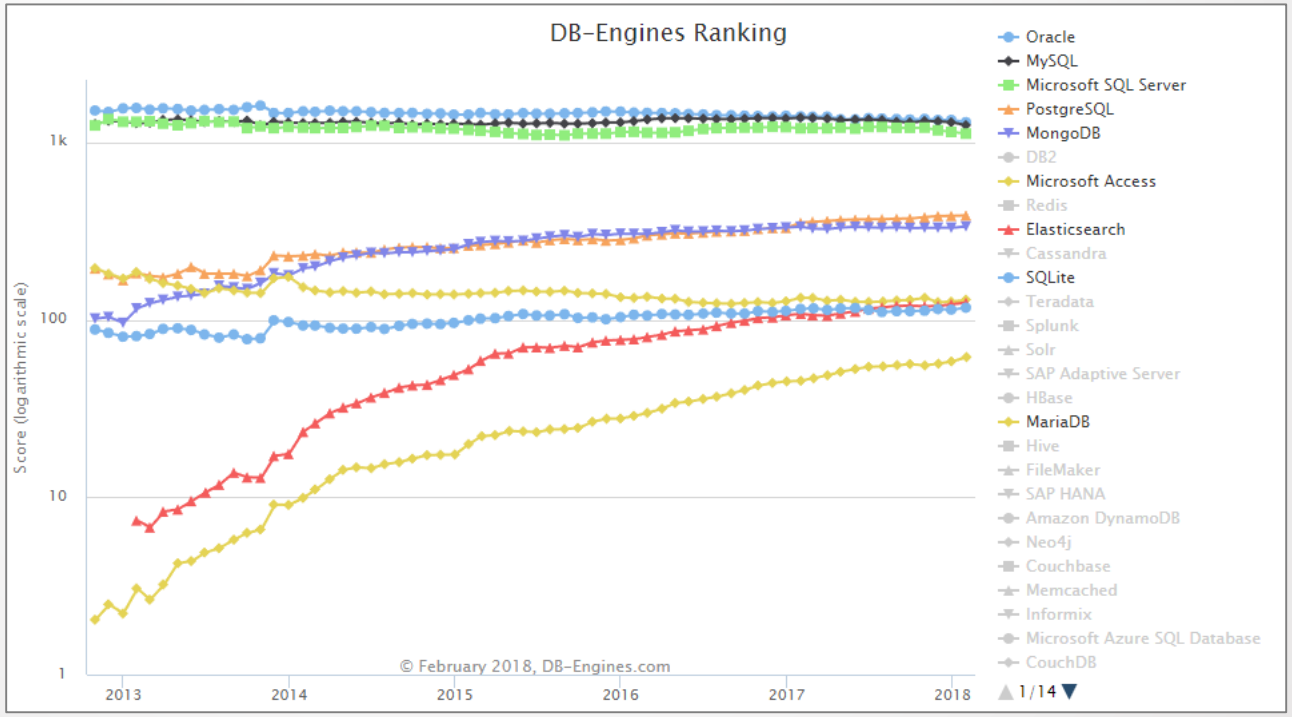
Comment choisir

- Performances ?
- Cout ?
- Disponibilité ?
- Langage d'interrogation ?
- Fonctionnalités ?

V. Limites

Limites

Base de données, les chiffres



Source: db-engines.com

Webo/Bibliographie

A lire pour aller plus loin:

- Guide de démarrage pour utiliser MongoDb
  https://openclassrooms.com/courses/guide-de-demarrage-pour-utiliser-mongodb

- Maîtrisez les bases de données NoSQL
  https://openclassrooms.com/courses/maitrisez-les-bases-de-donnees-nosql

- Les bases de données NoSQL et le Big Data: Comprendre et mettre en œuvre
  R. Bruchez, Eyrolles

- MongoDB : Base de donnée orientée documents
  Emmanuel Caruyer, CNRS