

# Bases de données aujourd'hui et bases de données NoSQL

Antoine Augusti – `antoine.augusti.fr`

Thibaud Dauce – `thibaud.dauce.fr`

4 mars 2016

## 1 Architecture web et modèle relationnel

- Architecture classique
- Bilan architecture classique
- Qu'est-ce qu'un ORM ?

## 2 Au delà du SQL : les BDs NoSQL

- Pourquoi changer du relationnel ?
- Relâchement des contraintes de transactions
- À quoi ressemble une BD NoSQL

## 3 Types de bases de données NoSQL

- Bases de données clé-valeur
- Bases de données orientées documents
- Bases de données orientées graphes

## 4 Conclusion

- Bonnes pratiques
- NoSQL ou relationnel ?

## 1 Architecture web et modèle relationnel

- Architecture classique
- Bilan architecture classique
- Qu'est-ce qu'un ORM ?

## 2 Au delà du SQL : les BDs NoSQL

- Pourquoi changer du relationnel ?
- Relâchement des contraintes de transactions
- À quoi ressemble une BD NoSQL

## 3 Types de bases de données NoSQL

- Bases de données clé-valeur
- Bases de données orientées documents
- Bases de données orientées graphes

## 4 Conclusion

- Bonnes pratiques
- NoSQL ou relationnel ?

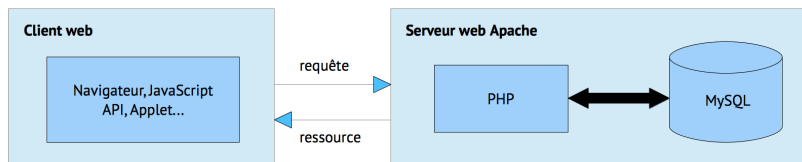


Figure – La stack LAMP.

## Stacks habituelles :

- LAMP : Linux, Apache, MySQL, PHP — la plus commune ;
- LEMP : Linux, Nginx, MySQL, PHP-FPM — commence à remplacer LAMP ;
- MEAN : MySQL, Express, AngularJS, Node.js — pour du JavaScript côté serveur.

## Avantages :

- Rapide à mettre en place (installation en un clic généralement) ;
- Optimisation possible par **scalabilité verticale** (augmenter les capacités physiques du serveur) ou séparation des services (BDD sur une autre machine par exemple).

## Inconvénients :

- Il faut une expertise base de données.

# Qu'est-ce qu'un ORM ?

## Définition : ORM

L'**Object-Relationnal Mapping** est une technique qui simule une base de données orientée objet à partir d'une base de données relationnelle.

- Fait la liaison entre le monde relationnel dans la couche stockage et le monde objet dans l'application ;
- S'intègre généralement dans un DAO (*Data Access Object*) pour respecter le pattern repository ;
- Facilité de développement : *pas besoin* d'une connaissance poussée du SQL ;
- Facilite les interactions avec la base de données pour les développeurs.

## Les limites des ORM

Toujours **beaucoup** moins performant que des requêtes SQL optimisées dans les cas complexes.



# ORM : exemple de requêtes

```
1 // Création d'un utilisateur
2 Map<String, String> userData = new HashMap<String, String>();
3 userData.put("prenom", "Antoine");
4 userData.put("nom", "Augusti");
5 User user = User.create(userData);
6
7 // Sélection des utilisateurs majeurs et articles qu'ils ont écrits
8 ArrayList<User> users = User.with("articles")
9     .where("age", ">=", 18)
10    .get();
11
12 // Suppression des utilisateurs vivant à Paris
13 User.where("ville", "Paris").delete();
14
15 // Les derniers articles d'un utilisateur (3ème page)
16 final int NOMBRE_ARTICLES_PAR_PAGE = 10;
17 ArrayList<Article> articles = Article.whereUserId(user.getId())
18     .latest()
19     .skip(2*NOMBRE_ARTICLES_PAR_PAGE)
20     .take(NOMBRE_ARTICLES_PAR_PAGE)
21     .get();
```

Listing 1: Quelques requêtes basiques avec un ORM imaginaire.

D'autres ORM : Hibernate (Java), Eloquent (PHP), SQLAlchemy (Python),  
Mongoose (Node.js)...



## 1 Architecture web et modèle relationnel

- Architecture classique
- Bilan architecture classique
- Qu'est-ce qu'un ORM ?

## 2 Au delà du SQL : les BDs NoSQL

- Pourquoi changer du relationnel ?
- Relâchement des contraintes de transactions
- À quoi ressemble une BD NoSQL

## 3 Types de bases de données NoSQL

- Bases de données clé-valeur
- Bases de données orientées documents
- Bases de données orientées graphes

## 4 Conclusion

- Bonnes pratiques
- NoSQL ou relationnel ?



- Service réparti sur plusieurs continents ?
- Accès concurrent de plusieurs centaines de milliers de personnes ?
- Stockage d'objets avec prise en compte de l'héritage ?
- Besoin d'analyser de gros volumes de données ?
- Reconnaissance automatique d'images ou traitement automatique de textes ?

# Pourquoi changer du relationnel ?

Besoin d'une alternative vers les années 2004 avec l'arrivée du *Big Data*.

- Des volumes de données important (plusieurs gigas, téraoctets, pétaoctets) ;
- Un nombre de transactions très important, une forte demande de disponibilité et de temps de réponse ;
- Des bases de données réparties sur plusieurs centres de données ou continents ;
- Préférence pour l'ajout de petites machines plutôt qu'une configuration poussée des BDs (concept de **scalabilité horizontale**).



Impossible de garantir les propriétés ACID des BDs relationnelles avec les nouvelles contraintes.

De nouvelles propriétés **BASE**, suite au théorème CAP :

- *Basic Availability* : système disponible dans son ensemble bien que certaines machines soient indisponibles ;
- *Soft state* : l'état du système distribué peut changer, même sans nouvelles transactions ;
- *Eventual Consistency* : En l'absence de nouvelles transactions, le système sera cohérent au bout d'un temps.

# À quoi ressemble une BD NoSQL

- Un SGBD qui n'est pas structuré en tables et dont l'élément de base n'est pas un tuple mais dépend du type de BD NoSQL ;
- Un langage de requête non uniformisé, propre à chaque BD. Souvent au format JSON avec une API REST ;
- Une dénormalisation des données où certains enregistrements sont en partie ou entièrement dupliqués ;
- Type de base de données NoSQL à choisir en fonction de l'usage souhaité ;
- Types de base de données NoSQL existants : clé-valeur, colonnes, documents, graphe. . .

## 1 Architecture web et modèle relationnel

- Architecture classique
- Bilan architecture classique
- Qu'est-ce qu'un ORM ?

## 2 Au delà du SQL : les BDs NoSQL

- Pourquoi changer du relationnel ?
- Relâchement des contraintes de transactions
- À quoi ressemble une BD NoSQL

## 3 Types de bases de données NoSQL

- Bases de données clé-valeur
- Bases de données orientées documents
- Bases de données orientées graphes

## 4 Conclusion

- Bonnes pratiques
- NoSQL ou relationnel ?

- **Modélisation** : la plus simple. À une clé, on associe une valeur. La valeur peut être de n'importe quel type (chaîne de caractères, entier, structure, objet sérialisé...).

Des exemples de paires clé-valeur avec des types différents.

Clé	Valeur
pays.id-42	{"id" :42,"name" :"Chad"}
statistiques.nombre-visiteurs	1337
configuration.periode-gratuite	false
articles.categories-sport.latest	[22, 45, 67, 200, 87]

- **Opérations** : création d'une paire clé-valeur, suppression, accès à une valeur à l'aide de la clé, incrémentation et décrémentation d'une valeur ;
- On peut définir la durée de vie d'une paire clé-valeur ou adopter une politique *least recently used* ;
- Stockage en RAM pour accélérer les temps de lecture. Mécanisme de reprise en cas de crash ;
- **Cas d'utilisation** : cache d'une autre BD, comptage d'éléments, gestion de files d'attente, opérations ensemblistes. . .
- **Principaux acteurs** : Redis, Memcached, Riak.

# Commandes sous Redis

```
1 # Assignment de chaîne "bonjour" à la clé "cleTexte"
2 redis> SET cleTexte bonjour
3 OK
4 # Récupération de la valeur de la clé "cleTexte"
5 redis> GET cleTexte
6 "bonjour"
7
8 # Incrémentation d'un compteur
9 redis> INCR compteur
10 (integer) 42
11 # Suppression du compteur
12 redis> DEL compteur
13 (integer) 1
14
15 # Création d'une liste avec insertion en fin de liste
16 redis> RPUSH liste "Hello"
17 (integer) 1
18 # Insertion en fin de liste
19 redis> RPUSH liste "World"
20 (integer) 2
```

Listing 2: Quelques commandes Redis en console.





- **Modélisation** : aucun schéma fixe, un document peut contenir n'importe quel type d'information ;
- Optimisation horizontale ;
- **Opérations** : utilise le JSON pour les requêtes et l'améliore pour le stockage (BSON) ;
- Facile d'accès (API REST, shell...);
- **Cas d'utilisation** : base de données principalement utilisées pour du stockage ;
- **Principaux acteurs** : MongoDB, CouchDB, CouchBase.

# Exemple d'un document

```
1  {
2    "id":1500,
3    "nom":"Thibaud",
4    "prénom":"Dauce",
5    "professeur_préférée":{
6      "numero":42,
7      "prenom":"Géraldine",
8      "nom":"Del Mondo",
9      "matières": ["BD", "Réseaux"]
10   },
11   "commentaires":[
12     {
13       "id":646,
14       "contenu":"J'adore la BD."
15     },
16     {
17       "id":647,
18       "contenu":"Boule et Bill aussi."
19     },
20     {
21       "id":648,
22       "contenu":"Et pleins d'autres trucs."
23     }
24   ]
25 }
```

Listing 3: Exemple d'un document JSON.

```
db.inventory.find( { type: "snacks" } )
```

Listing 4: Exemple de requête find sur MongoDB.

# Exemples de requêtes MongoDB

```
db.inventory.find(  
  {  
    type: 'food',  
    $or: [ { qty: { $gt: 100 } }, { price: { $lt: 9.95 } } ]  
  }  
)
```

Listing 5: Exemple de requête find sur MongoDB avec des opérateurs spéciaux.

# Exemples de requêtes MongoDB

```
db.inventory.insert(  
  {  
    item: "ABC1",  
    details: {  
      model: "14Q3",  
      manufacturer: "XYZ Company"  
    },  
    stock: [  
      { size: "S", qty: 25 },  
      { size: "M", qty: 50 }  
    ],  
    category: "clothing"  
  }  
)
```



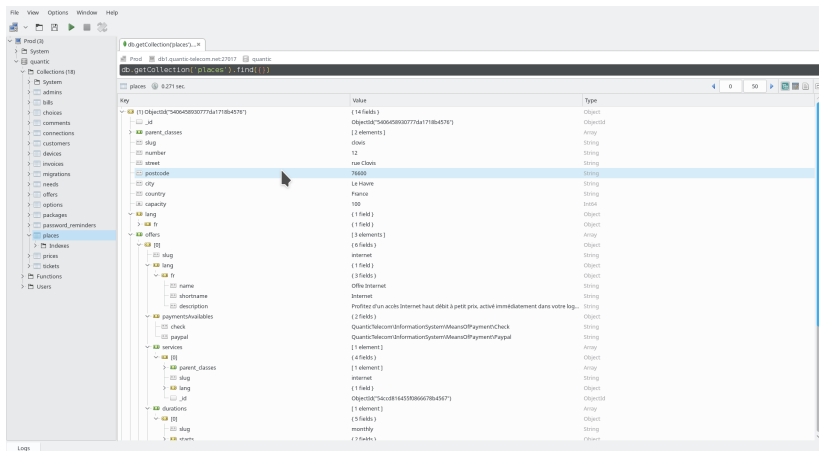
Listing 6: Exemple de requête insert sur MongoDB.

# Exemples de requêtes MongoDB

```
db.inventory.update(  
  { item: "MNO2" },  
  {  
    $set: {  
      category: "apparel",  
      details.model: "14Q2"  
    },  
    $currentDate: { lastModified: true }  
  }  
)
```

Listing 7: Exemple de requête update sur MongoDB.

# Mais on peut aussi utiliser une interface graphique



The screenshot shows the Robomongo interface displaying a document from the 'places' collection. The document is expanded to show its nested structure. The 'parent\_classes' field contains an array of two objects. The 'offers' field contains an array of three objects. The 'services' field contains an array of one object. The 'payments/availables' field contains an array of two objects. The 'services' object is further expanded to show its 'parent\_classes' and 'lang' fields.

Key	Value	Type
[1] ObjectID("40645893077da1718b4578")	(14 fields)	Object
.._id	ObjectID("40645893077da1718b4578")	ObjectID
..parent_classes	[2 elements]	Array
..slug	String	String
..class	String	String
..number	12	String
..street	rue Clauz	String
..postcode	76400	String
..city	Le Havre	String
..country	France	String
..capacity	100	Int64
..lang	(1 field)	Object
..fr	(1 field)	Object
..offers	[3 elements]	Array
..fr	(6 fields)	Object
..slug	internet	String
..lang	(1 field)	Object
..fr	(3 fields)	Object
..name	Offre Internet	String
..shortname	Internet	String
..description	Profitez d'un accès Internet haut débit à petit prix, activé immédiatement dans votre logi...	String
..payments/availables	(2 fields)	Object
..check	QuantaTelecomInformationSystem\MeanOfPaymentCheck	String
..paypal	QuantaTelecomInformationSystem\MeanOfPaymentPaypal	String
..services	[1 element]	Array
..fr	(4 fields)	Object
..parent_classes	[1 element]	Array
..slug	internet	String
..lang	(1 field)	Object
.._id	ObjectID("5ac08164559386678b4567")	ObjectID
..durations	[1 element]	Array
..fr	(5 fields)	Object
..slug	monthly	String
..events	(7 fields)	Object

Figure – La vue d'un document avec Robomongo.

# Bases de données orientées graphes

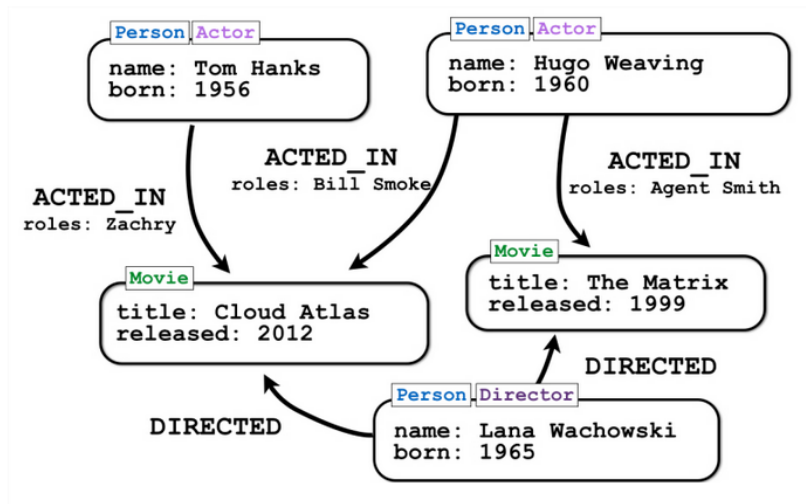


Figure – Exemple de graphe.



- **Modélisation** : représentation de l'information de manière très particulière (nœuds et arcs de même importance) ;
- **Opérations** : traitement très particulier de l'information (voir slide précédente) ;
- **Cas d'utilisation** : utilisé principalement pour les réseaux (liens entre des équipements ou des personnes) ;
- Base de données rarement utilisée pour du stockage ;
- **Principaux acteurs** : Neo4j, Titan, OrientDB.

# Bases de données orientées graphes

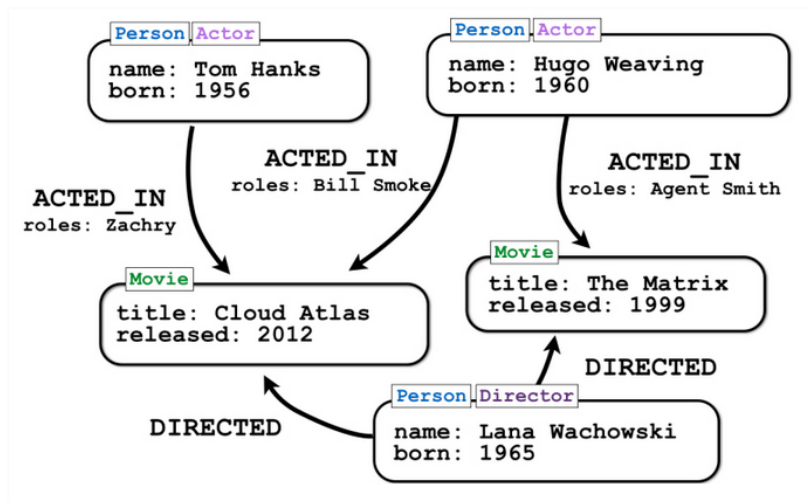


Figure – Exemple de graphe.

# Exemple de requête

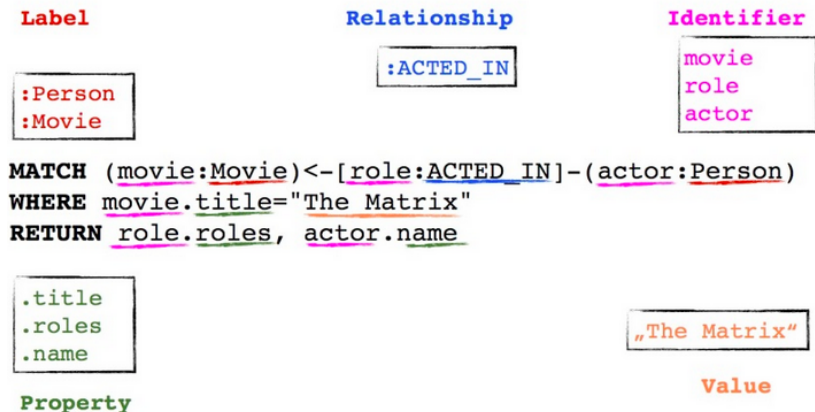


Figure – Exemple de requête Cypher pour Neo4j.

## 1 Architecture web et modèle relationnel

- Architecture classique
- Bilan architecture classique
- Qu'est-ce qu'un ORM ?

## 2 Au delà du SQL : les BDs NoSQL

- Pourquoi changer du relationnel ?
- Relâchement des contraintes de transactions
- À quoi ressemble une BD NoSQL

## 3 Types de bases de données NoSQL

- Bases de données clé-valeur
- Bases de données orientées documents
- Bases de données orientées graphes

## 4 Conclusion

- Bonnes pratiques
- NoSQL ou relationnel ?

- Ne pas commencer par du NoSQL ;
- Bien connaître ses données ;
- Ne pas avoir peur de la redondance ;
- Ne pas trop redonder ;
- Et surtout, bien quantifier ses besoins.

## RTFM

La majorité des bases de données NoSQL connues sur le marché possèdent une très bonne documentation, souvent faite à destination de personnes venant du monde du relationnel.

# D'autres cas d'usage du NoSQL

Première ligne en prod', mais pas que

Le NoSQL ne permet pas uniquement de contenir la charge, de répondre aux problématiques du distribué.

D'autres attraits du NoSQL :

- **Dénormalisation** : business intelligence, machine learning, data mining (BDD colonnes, documents)...
- **Stockage** : entrepôt d'agrégation de données (BDD colonnes, documents)
- **Modélisation** : systèmes de recommandation (BDD graphe)
- ...



# NoSQL ou relationnel ?

## Les deux mon capitaine !

Les bases de données NoSQL ont été inventées afin de résoudre des problèmes insolubles par les bases de données relationnelle et non **pas pour les remplacer.**

NoSQL	Relationnel
stockage de masse	stockage fiable
données diverses	données formatées
scalabilité horizontale	scalabilité verticale

# Choix du type de BD NoSQL

	Modélisation	Cas d'utilisation
Clé / valeur	Modélisation simple, permettant d'indexer des informations diverses via une clé	Mise en cache
Documents	Modélisation souple permettant de stocker des documents au format JSON dans des collections	Stockage de masse
Graphes	Modélisation optimisée pour les problèmes de graphes	Stockage provisoire pour traiter les données

Non exhaustif : il existe d'autres types de BDD NoSQL !

