

MIF4 - SGBD non-relationnels

Fabien Duchateau

fabien.duchateau [at] univ-lyon1.fr

Université Claude Bernard Lyon 1

2020 - 2021



<http://emmanuel.coquery.pages.univ-lyon1.fr/enseignement/mif04/>

Depuis les années 1970, dominance du modèle relationnel

Émergence du web et du phénomène "Big Data" :

- ▶ Grandes plateformes ou applications web gérant des millions d'utilisatrices
- ▶ Explosion du volume de données à stocker et à traiter
- ▶ Données de plus en plus complexes et hétérogènes

Contexte

Depuis les années 1970, dominance du modèle relationnel

Émergence du web et du phénomène "Big Data" :

- ▶ Grandes plateformes ou applications web gérant des millions d'utilisatrices
- ▶ Explosion du volume de données à stocker et à traiter
- ▶ Données de plus en plus complexes et hétérogènes

Limites des SGBD relationnels (utilisant le langage SQL) pour ces nouveaux usages, à cause du mécanisme de jointures, des contraintes d'intégrité et des transactions

Le Big Data

Big Data : modélisation, stockage et traitement (analyse) d'un ensemble de données très volumineuses, croissantes et hétérogènes, dont l'exploitation permet entre autre :

- ▶ Prise de décisions, prédiction
- ▶ Découverte de nouvelles connaissances
- ▶ Possibilités de nouveaux "business models" (e.g., accès à un service contre des informations)

Causes :

- ▶ Faible coût du stockage
- ▶ Faible coût des processeurs
- ▶ Mise à disposition des données

Le Big Data (2)

Les "3V", caractéristiques du Big Data :

- ▶ **Volume** (plusieurs zettaoctets générés par an sur le web)
- ▶ **Vélocité** (fréquence de génération des données)
 - ▶ notion de flux (stream)
 - ▶ 4000 To par jour pour Facebook (2016)
 - ▶ 7000 To par seconde prévus pour le radiotélescope "Square Kilometre Array" (2020)
- ▶ **Variété** (hétérogénéité)
 - ▶ données brutes, structurées ou pas, etc.
 - ▶ images, texte, géo-démographiques, profils utilisatrices, etc.

http://fr.wikipedia.org/wiki/Big_data

<http://fr.wikipedia.org/wiki/Zettaoctet>

Exemples d'application

Sciences :

- ▶ Création de cartes de navigation par Fontaine Maury, au 19^{ème} siècle, à partir de vieux journaux de bord (précurseur)
- ▶ Large Hadron Collider (LHC), un accélérateur de particules qui génère un flux de 40 millions de données par seconde
- ▶ Décodage du génome humain

Politique :

- ▶ Prédiction du résultat des élections États-Uniennes 2012
- ▶ Transparence (Open Data)



http://en.wikipedia.org/wiki/Big_data#Big_science

<http://linkeddata.org>

http://fr.wikipedia.org/wiki/Open_data

Exemples d'application (2)

Industrie, secteur public, santé, etc. :

- ▶ Amazon gère des millions d'opérations par jour
- ▶ Programmes de surveillance (e.g., Prism)
- ▶ Réduction de 22% du gaspillage alimentaire (en GMS)
- ▶ Prédiction des analystes de Google quelques semaines avant l'épidémie de grippe aviaire en 2009
- ▶ Découverte d'un effet secondaire dû à la prise de deux médicaments par analyse des requêtes des internautes (Yahoo)
- ▶ Confirmation de la censure par analyse de la fréquence de noms de personnes (Chagall en Allemagne, Trotski en URSS)

<http://rue89.nouvelobs.com/2016/11/18/22-gaspillage-moins-les-supermarches-grace-big-data-265688>

http://labsoftnews.typepad.com/lab_soft_news/2013/03/

Motivation

Distribution des données sur différents serveurs et "data centers" :

- ▶ Passage d'un système vertical ("dopage" du serveur) à un système horizontal (ajout de machines "basiques")
- ▶ Contraignant avec des SGBD relationnels (*cf MIF24 - BDR*)

Nouveaux besoins :

- ▶ Passage à l'échelle (meilleures performances)
- ▶ Forte disponibilité, résistance aux pannes
- ▶ Gestion flexible des données (schémas dynamiques)

Motivation

Distribution des données sur différents serveurs et "data centers" :

- ▶ Passage d'un système vertical ("dopage" du serveur) à un système horizontal (ajout de machines "basiques")
- ▶ Contraignant avec des SGBD relationnels (*cf MIF24 - BDR*)

Nouveaux besoins :

- ▶ Passage à l'échelle (meilleures performances)
- ▶ Forte disponibilité, résistance aux pannes
- ▶ Gestion flexible des données (schémas dynamiques)

Mouvement NoSQL, NotOnlySQL, NoRel, NewSQL, ...

Plan

Caractéristiques des SGBD non-relationnels

Classification des SGBD non-relationnels

MongoDB

Généralités

SGBD non-relationnel \approx entrepôt clé-valeur fortement optimisé

Les SGBD non-relationnels sont "BASE", pour :

- ▶ Basically Available (haute disponibilité)
- ▶ Soft-state ("auto-modifications" de la BD)
- ▶ Eventually consistent (cohérence sur le long terme)

Dans la suite, caractéristiques partagées de ces SGBD :

- ▶ Modèle de données
- ▶ Distribution des données
- ▶ Théorème de CAP
- ▶ Gestion de la cohérence

Modèle de données

- ▶ Non relationnel (!), différentes représentations (*cf section classification*) qui complètent le modèle relationnel pour certains contextes d'utilisation
- ▶ Pas de schéma (ou schéma dynamique), ce qui permet l'ajout d'informations à la volée
- ▶ Structure de données complexe ou imbriquée
- ▶ Pas de jointures (données dénormalisées)
- ▶ Compromis sur les propriétés ACID des SGBD relationnels, pas de transactions

Distribution des données ("sharding")

- ▶ Partitionnement horizontal des données ("par tuples joints" selon la philosophie relationnelle)
- ▶ Réplication des données sur plusieurs serveurs / "data centers" : même fonction de hachage utilisée pour données et serveurs ("consistent hashing", e.g., DHT, cf *MIF24 - BDR*)
- ▶ Généralement peu d'écritures et beaucoup de lectures
- ▶ Traitements distribués adaptés à cette distribution (cf *cours Map Reduce*)

Certains SGBD non-relationnels (orientés colonnes) utilisent un système de partitionnement vertical

Théorème CAP - définition

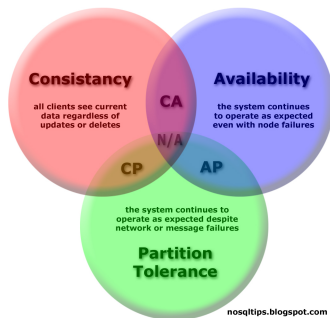
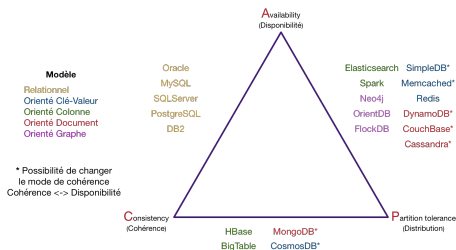
Le théorème de Brewer ou **théorème CAP** = un système distribué ne peut garantir en même temps les trois propriétés suivantes :

- ▶ Cohérence (consistency) : tous les noeuds du système voient la même information au même moment
- ▶ Disponibilité (*availability*) : toute requête reçoit une réponse (latence minimale)
- ▶ Résistance au morcellement (*partition tolerance*) : fonctionnement autonome en cas de morcellement du réseau (sauf panne totale)

<http://ksat.me/a-plain-english-introduction-to-cap-theorem/>
<http://www.julianbrowne.com/article/viewer/brewers-cap-theorem>

Théorème CAP - triangle

Les SGBD non-relationnels privilégient la **résistance au morcellement** (systèmes AP, et quelques CP)



<http://openclassrooms.com/fr/courses/4462426>

Cohérence

Dans la philosophie "non-relationnelle", on accède aux données par des clés

Cohérence d'une clé (distribuée) : toutes les lectures pour cette clé retournent la même donnée/valeur

La cohérence est un problème crucial lors d'écritures simultanées de données sur des serveurs et/ou "data centers" distribués

- ▶ Solution en relationnel avec les transactions distribuées
- ▶ Impact négatif sur la disponibilité / résistance au morcellement

⇒ Nouvelles techniques pour assurer une cohérence "partielle"

http://en.wikipedia.org/wiki/Eventual_consistency

Cohérence (2)

Différentes formes de cohérences :

- ▶ *Strong/strict consistency* : tous les serveurs sont cohérents après une mise à jour (e.g., transactions distribuées)
- ▶ *Weak consistency* : aucune garantie de cohérence à terme (e.g., perte d'une mise à jour suite à la défaillance d'un serveur)
- ▶ *Eventual consistency* (cohérence à long terme) : si aucune nouvelle mise à jour n'est effectuée pour une clé donnée, alors toutes les prochaines lectures finiront par lire la même valeur associée à cette clé (à plus ou moins long terme)

Sakr, Gaber. [Large Scale and Big Data : Processing and Management](#) (2014)
Bernstein, Sudipto. [Rethinking Eventual Consistency](#), SIGMOD (2013)

Cohérence (3)

Différentes formes de l'*eventual consistency* :

- ▶ *Read your writes* : garantie de lire les écritures précédentes de la session
- ▶ *Monotonic reads* : les lectures successives d'une session retournent toujours la dernière donnée lue ou une plus récente
- ▶ *Monotonic writes* : les écritures d'une session sont appliquées dans le même ordre pour toutes les copies
- ▶ *Causal consistency* : pas de valeur lue "au milieu" d'une séquence d'opérations (*session order* ou *reads-from order*)
- ▶ *Timeline consistency* : les écritures de n'importe quelle session sont appliquées dans le même ordre pour toutes les copies

Cohérence (3)

Différentes formes de l'*eventual consistency* :

- ▶ *Read your writes* : garantie de lire les écritures précédentes de la session
- ▶ *Monotonic reads* : les lectures successives d'une session retournent toujours la dernière donnée lue ou une plus récente
- ▶ *Monotonic writes* : les écritures d'une session sont appliquées dans le même ordre pour toutes les copies
- ▶ *Causal consistency* : pas de valeur lue "au milieu" d'une séquence d'opérations (*session order* ou *reads-from order*)
- ▶ *Timeline consistency* : les écritures de n'importe quelle session sont appliquées dans le même ordre pour toutes les copies

Un SGBD peut proposer plusieurs formes de cohérence, et l'application détermine la plus adaptée selon le contexte

Cohérence sur le long terme

Différentes techniques pour atteindre l'*eventual consistency* :

- ▶ **Two-Phase Commit** : protocole qui coordonne les processus impliqués dans une transaction atomique distribuée à commiter ou annuler (bloquant et non tolérant aux pannes)
- ▶ **Paxos-style consensus** : protocole qui trouve une valeur au consensus parmi les processus participants

http://en.wikipedia.org/wiki/Two-phase_commit_protocol

http://en.wikipedia.org/wiki/Paxos_algorithm

Cohérence sur le long terme (2)

- **Read-repair** : écriture de toutes les versions incohérentes, et lors d'une prochaine lecture, le conflit sera détecté et résolu (souvent en écrivant sur un certain nombre de répliques)

Exemple avec Dynamo (Amazon) : dans un panier, les ajouts de produits sont cruciaux, mais les suppressions de produits le sont moins car le client corrigera l'erreur. Dans ce cas, l'union du contenu des deux paniers en conflit permet de ne pas perdre d'ajout de produits. Utilisation des horloges vectorielles pour limiter l'incohérence des paniers au niveau des suppressions de produits

Philosophie "When in doubt, take customer's order!"

<http://the-paper-trail.org/> (post du 26/08/2008)

http://en.wikipedia.org/wiki/Vector_clock

En résumé

- ▶ Schéma dynamique [et non contraint]
- ▶ Distribution des données via partitionnement horizontal
- ▶ Systèmes AP (et CP) du théorème de CAP :
 - ▶ mécanismes pour atteindre la cohérence sur le long-terme



"YOUR CONTENT IS CONSISTENT. IT'S ALWAYS BAD."

Plan

Caractéristiques des SGBD non-relationnels

Classification des SGBD non-relationnels

MongoDB

Catégories des SGBD non-relationnels

Domaine en pleine évolution !

Propositions de classification selon :

- ▶ Le modèle de données
- ▶ Les caractéristiques non fonctionnelles
- ▶ Les propriétés du théorème de CAP

Dans la suite, classification selon le modèle de données

Davoudian et al. [A Survey on NoSQL Stores](#), CSUR (2018)

<http://hostingdata.co.uk/nosql-database/>

<http://fr.slideshare.net/bscofield/nosql-codemash-2010>

Le modèle relationnel

Un modèle que vous connaissez bien...

- ▶ Relations, attributs, tuples, etc.
- ▶ Propriétés de cohérence et de disponibilité
- ▶ SGBD : PostgreSQL, Oracle, MariaDB, MySQL, etc.

Exemple de tables relationnelles

Table **ECRIVAIN**

id	nom	pays	dateNaiss
2	GRR Martin	USA	20/09/48

Table **LIVRE**

id	titre	prix	datePubli	ecrivain
1	Le trône de fer	15	01/08/96	2

Entrepôt clé-valeur

Entrepôt clé-valeur ("key-value store") \approx tableau associatif

- ▶ La clé est un identifiant unique
- ▶ La valeur peut être scalaire ou structurée

Implémentation minimale :

- ▶ valeur = **get**(clé)
- ▶ **insert**(clé, valeur)
- ▶ **delete**(clé)

Utilisé pour profils utilisateur, paniers, données capteur, logs, etc.

Les implémentations proposent souvent des fonctionnalités ou propriétés supplémentaires

Entrepôt clé-valeur (2)

- ▶ Bonnes performances, partitionnement peu contraignant
- ▶ Modèle (trop) simple
- ▶ Langage d'interrogation limité
- ▶ Logique programmée côté application

Exemples de SGBD : Redis, Dynamo (Amazon), Voldemort (LinkedIn puis open-source), Memcached, [Riak], ...

<http://redis.io/>

<http://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf>

<http://www.project-voldemort.com/>

<http://www.memcached.org/>

<http://riak.com/>

Entrepôt clé-valeur (3)

Exemple de données en clé-valeur

```
"nom-ecrivain2" : "GRR Martin"  
"pays-ecrivain2" : "USA"  
"dateNaiss-ecrivain2" : 20/09/48  
"titre-livre1-ecrivain2" : "Le trône de fer"  
"prix-livre1-ecrivain2" : 15  
"datePubli-livre1-ecrivain2" : "01/08/96"
```

BD orientée colonnes

BD orientée colonnes = organisation des données en colonnes

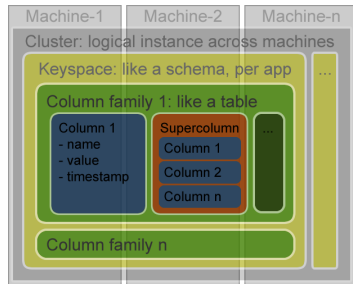
- ▶ Une colonne stocke un couple clé-valeur (et un timestamp)
- ▶ Une supercolonne stocke des colonnes (\approx ligne en relationnel)
- ▶ Une famille de colonnes stocke des colonnes ou supercolonnes

Utilisé pour logs et recommandations (Netflix), recherche de produits par catégorie (Ebay), agrégations et statistiques (Adobe, audiences télé), etc.

Ne pas confondre avec des BD relationnelles orientées colonnes, qui sérialisent les données par colonne (e.g., MonetDB, Vertica)

BD orientée colonnes (2)

- ▶ Bonnes performances
- ▶ Adapté aux données éparses, temps réel
- ▶ Requêtes pré-écrites (connues à l'avance)
- ▶ Pas pour les données fortement connectées



Exemples de SGBD : BigTable (Google), Cassandra (*BigTable* libre), HBase, Hypertable, ...

<http://cassandra.apache.org/>
<http://cassandra-php.blogspot.fr/>
<http://hbase.apache.org/>
<http://hypertable.com/>

BD orientée colonnes (3)

Stockage orienté lignes

id	type	lieu	spec	intérêts
Nicolas	prof	CNAM	BDD, NoSQL	BZH, Star Wars
Régis		OC	Machine Learning, Dev	escalade, nouilles chinoises
Luc	resp formation OC	OC	formation, audiovisuel	
Céline	prof	CentraleSupélec	Ontologie, logique formelle, visualisation	

Stockage orienté colonnes

id	type	id	lieu	id	spec	id	intérêts
Nicolas	prof	Céline	CentraleSupélec	Nicolas	BDD	Nicolas	BZH
Céline	prof	Nicolas	CNAM	Nicolas	NoSQL	Nicolas	Star Wars
Luc	resp formation OC	Régis	OC	Régis	Machine Learning	Régis	escalade
		Luc	OC	Régis	Dev	Régis	nouilles chinoises
				Luc	formation		
				Luc	audiovisuel		
				Céline	Ontologie		
				Céline	logique formelle		
				Céline	visualisation		

Stockage par ligne (relationnel) et stockage par colonne

<http://openclassrooms.com/fr/courses/4462426>

BD orientée colonnes (4)

Exemple de données en orienté colonnes

Un tableau représente une famille de colonnes (e.g., *écrivains*).

Une ligne représente une supercolonne.

Exemples de colonne : (1, nom, "GRR Martin"), (1, pays, "USA"),
(3, nom, "JRR Tolkien").

1	nom : GRR Martin	pays : USA	dateNaiss : 20/09/48
3	nom : JRR Tolkien	pays : GB	

2	titre : Le trône de fer	prix : 15	datePubli : 01/08/96	auteur : 1
---	----------------------------	--------------	-------------------------	---------------

BD orientée documents

BD orientée documents = collection de documents (clé-document)

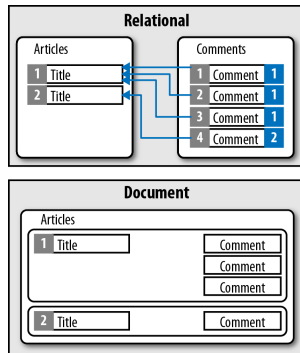
- ▶ Notion abstraite de "document"
- ▶ Structure arborescente : liste de champs (dont la valeur peut être une liste de champs)
- ▶ Deux documents peuvent avoir des champs différents

Utilisé pour gestion de contenu (CMS, bibliothèque numérique), événements complexes, analytique temps réel, catalogues, etc.

http://en.wikipedia.org/wiki/Document-oriented_database

BD orientée documents (2)

- ▶ Bonnes performances (dégradation avec l'inclusion)
- ▶ Modèle simple et puissant (imbrication)
- ▶ Facilité d'évolution du schéma
- ▶ Expressivité du langage de requêtes (sur contenu du document)
- ▶ Pas pour les données fortement connectées



Exemples de SGBD : MongoDB, CouchBase, CouchDB, ...

<http://www.mongodb.org/>

<http://www.couchbase.com/>

<http://couchdb.apache.org/>

BD orientée documents (3)

Exemple de données en orienté document

DOCUMENT 1

```
{
  _id : "livre1",
  "titre" : "Le trône de fer",
  "prix" : 15,
  "ecrivain" : "ecrivain2",
  "datePubli" : "01/08/96"
}
```

DOCUMENT 2

```
{
  _id : "ecrivain2"
  "nom" : "GRR Martin"
  "pays" : "USA"
  "dateNaiss" : "20/09/48"
}
```

OU

DOCUMENT 3

```
{ _id : "ecrivain2",
  "nom" : "GRR Martin",
  "pays" : "USA",
  "dateNaiss" : 20/09/48,
  "livres" : [
    {
      "titre" : Le trône de fer",
      "prix" : 15,
      "datePubli" : "01/08/96"
    }
  ]
}
```

BD orientée graphes

BD orientée graphes = éléments interconnectés avec un nombre indéterminé de relations entre eux (théorie des graphes)

- ▶ Noeuds pour représenter des entités
- ▶ Arcs étiquetés et directionnels entre deux noeuds

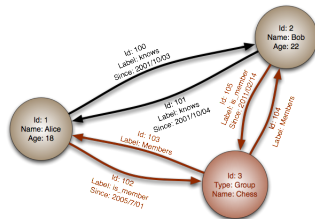
Les BD graphes peuvent représenter les autres modèles (clé-valeur, colonne, document)

Utilisé pour recommandations, web sémantique, web des objets, données géospatiales, réseaux sociaux, calcul scientifique, etc.

http://en.wikipedia.org/wiki/Graph_database

BD orientée graphes (2)

- ▶ Bonnes performances en lecture
- ▶ Facilité d'évolution du schéma
- ▶ Requête de type graphe (e.g., plus court chemin) ou détection de patterns
- ▶ Framework commun TinkerPop
- ▶ Pas/peu de sharding



Exemples de SGBD : Neo4J, OrientDB, BlazeGraph, AllegroGraph, Virtuoso, JanusGraph, Giraph, ...

<http://tinkerpop.apache.org/>

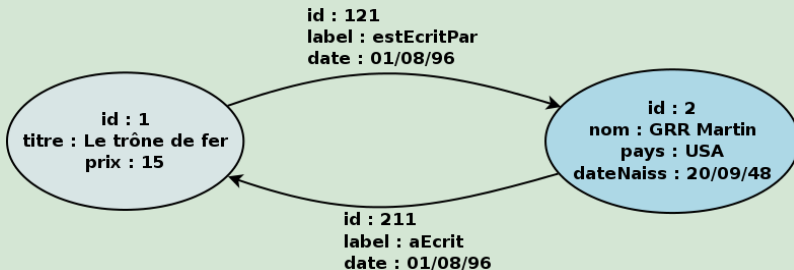
<http://www.neo4j.org/>

<http://www.franz.com/agraph/allegrograph/>

<http://virtuoso.openlinksw.com/>

BD orientée graphes (3)

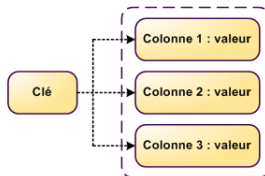
Exemple de données en orienté graphe



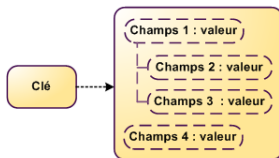
Synthèse des catégories de SGBD non-relationnels



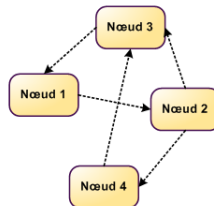
BDD Clé-Valeur



BDD Orientée colonnes



BDD Orientée document



BDD Orientée graphe

<http://blog.xebia.fr/>

D'autres catégories de SGBD non-relationnels

La plupart des SGBD supportent en réalité plusieurs modèles (multi-modèles), e.g., *DynamoDB est clé-valeur et document*

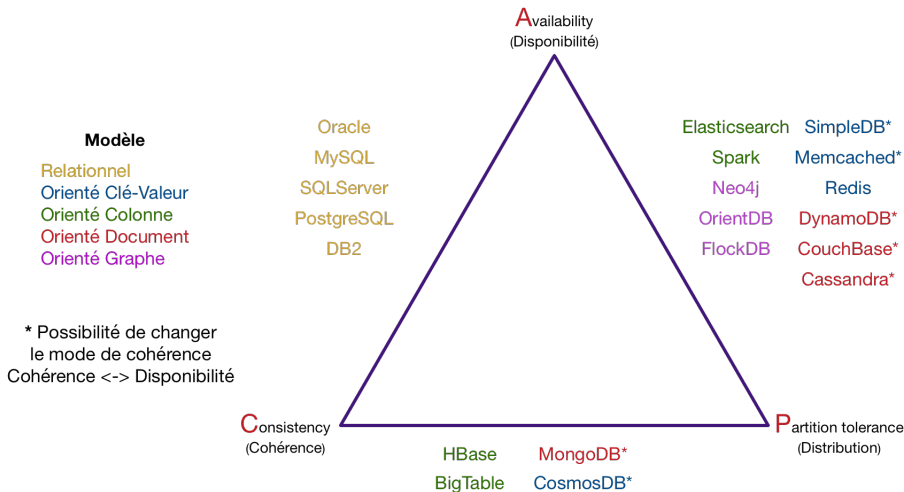
Autres modèles :

- ▶ Moteur de recherche (Solr, Elasticsearch, Splunk)
- ▶ Entrepôt RDF / triplestores (Jena, Virtuoso, RDF4J)
- ▶ BD XML natif (BaseX, eXistDB)
- ▶ BD multi-valeurs non 1FN, i.e., avec attributs multi-valués (SciDB, Rasdaman)
- ▶ BD événements, avec modifications historiées (Event Store)
- ▶ ...

<http://db-engines.com/>

<http://nosql-database.org/>

En résumé



Plan

Caractéristiques des SGBD non-relationnels

Classification des SGBD non-relationnels

MongoDB

Caractéristiques de MongoDB

- ▶ SGBD orienté documents
- ▶ Open-source
- ▶ Populaire (5^{ème} SGBD le plus utilisé)
- ▶ Passage à l'échelle horizontal (sharding) et réplication
- ▶ Système CP (cohérent et résistant au morcellement)
 - ▶ "strong consistency" (lectures sur serveur primaire)
 - ▶ "eventual consistency" (lectures sur différents serveurs)
- ▶ Traitements distribués (Map Reduce, *aggregation pipeline*)
- ▶ GUI (Compass, Robot 3T) et nombreux drivers



<http://www.mongodb.com/>

<http://robomongo.org/download>

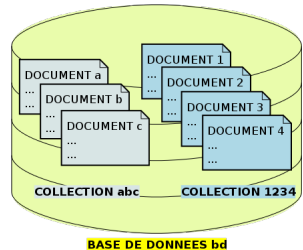
Concepts principaux - BD et collection

Base de données (\sim *base de données* en modèle relationnel) :

- ▶ Ensemble de collections
- ▶ Espace de stockage

Collection (\sim *table* en modèle relationnel) :

- ▶ Ensemble de documents qui partagent un objectif ou des similarités
- ▶ Pas de "schéma" prédéfini



<http://docs.mongodb.com/manual/core/databases-and-collections/>

Concepts principaux - document

Document (\sim *ligne*, *tuple* en modèle relationnel) :

- ▶ Un enregistrement dans une collection
- ▶ Syntaxe et stockage au format BSON
- ▶ Identifiant d'un document (clé "**_id**")

BSON = "Binary JSON" (JavaScript Object) avec améliorations :

- ▶ Ensemble de paires clé/valeur
- ▶ Une valeur peut être un objet complexe (liste, document, ensemble de valeurs, etc.)
- ▶ Représentation de nouveaux types (e.g., dates)
- ▶ Facilité de parsing (e.g., entiers stockés sur 32/64 bits)

<http://docs.mongodb.com/manual/core/document/>
<http://bsonspec.org/>

Concepts principaux - document (2)

Syntaxe d'un document en MongoDB
(format BSON) :

- ▶ **_id** est un identifiant (généré ou manuel)
- ▶ **att-1** est une clé dont la valeur est une chaîne de caractères
- ▶ **att-2** est une clé dont la valeur est un nombre
- ▶ **att-3** est une clé dont la valeur est une liste de valeurs
- ▶ **att-k** est une clé dont la valeur est un document inclus

```
{
  _id : <identifiant>,
  "att-1" : "val-1",
  "att-2" : val-2,
  "att-3" : ["val-31", "val-32", ...]
  ...
  "att-k" : {
    "att-k1" : "val-k1",
    ...
  }
}
```

Concepts principaux - document (3)

```
{  
  name: "sue",  
  age: 26,  
  status: "A",  
  groups: [ "news", "sports" ]  
}
```

← field: value
← field: value
← field: value
← field: value

Exemple de document au format BSON

Définition possible de contraintes sur les champs (validation)

<http://docs.mongodb.com/manual/introduction/>

<http://docs.mongodb.com/manual/core/document-validation/>

Relations inter-documents

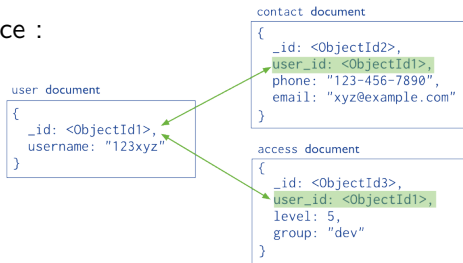
Relation entre les documents de différentes collections :

- ▶ Par référence : l'identifiant d'un document (son "_id") est utilisé comme valeur attributaire dans un autre document
 - ▶ se rapproche du modèle de données normalisées
 - ▶ nécessite des requêtes supplémentaires côté applicatif
- ▶ Par inclusion ("embedded") : un "sous-document" est utilisé comme valeur
 - ▶ philosophie "non-relationnelle" (pas de jointure)
 - ▶ meilleures performances en lecture
 - ▶ redondance possible

<http://docs.mongodb.com/manual/core/data-modeling-introduction/>

Relations inter-documents (2)

Relation par référence :



Relation par inclusion (embedded) :



Modélisation d'une BD

Considérations pour modéliser une BD au niveau physique :

- ▶ Accès par une clé (identifiant d'un document)
- ▶ Schéma optionnel \Rightarrow ajout d'un champ à tout moment
- ▶ Pas de jointure \Rightarrow redondances possibles
- ▶ Inclusion \Rightarrow moins de lectures
- ▶ Opérations atomiques sur un seul document, mais pas sur plusieurs \Rightarrow état incohérent temporaire (souvent tolérable)

<http://docs.mongodb.com/manual/core/data-model-design/>
<http://docs.mongodb.com/manual/core/schema-validation/>
<http://mongoosejs.com/>

Modélisation d'une BD

Considérations pour modéliser une BD au niveau physique :

- ▶ Accès par une clé (identifiant d'un document)
- ▶ Schéma optionnel \Rightarrow ajout d'un champ à tout moment
- ▶ Pas de jointure \Rightarrow redondances possibles
- ▶ Inclusion \Rightarrow moins de lectures
- ▶ Opérations atomiques sur un seul document, mais pas sur plusieurs \Rightarrow état incohérent temporaire (souvent tolérable)

"Application-driven" : les besoins applicatifs guident la conception pour identifier, stocker et accéder aux concepts (types de requêtes, ratio lecture/écriture, croissance du nombre de documents)

<http://docs.mongodb.com/manual/core/data-model-design/>

<http://docs.mongodb.com/manual/core/schema-validation/>

<http://mongoosejs.com/>

Modélisation d'une BD - recommandations

Relation **par inclusion** pour :

- ▶ les entités fréquemment lues ensemble (article/commentaires)
- ▶ une entité dépendante d'une autre (facture/client)
- ▶ des données mises à jour en même temps (nombre d'exemplaires d'un livre et informations sur les emprunteuses)
- ▶ besoin de rechercher des documents via un index multi-clés (catégories d'un livre)

Relation **par référence** pour :

- ▶ éviter une forte redondance sans "gain" (livre/éditeur)
- ▶ des entités fortement connectées "*many to many*" (livres/auteurs)
- ▶ des données hiérarchiques (catégories/sous-catégories/...)

<http://docs.mongodb.com/manual/core/data-modeling-introduction/>

CRUD = IFUD

Toute opération sur un seul document est atomique :

- ▶ INSERT
- ▶ FIND
- ▶ UPDATE
- ▶ DELETE

Lors d'insertion et mises à jour, la base de données et la collection sont automatiquement créées si elles n'existent pas

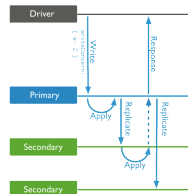
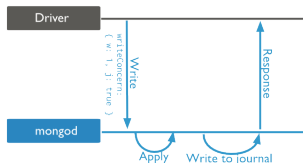
Langage de requête de MongoDB basé sur des motifs (patterns)

<http://docs.mongodb.org/manual/crud/>

Opération INSERT

```
db.users.insertOne(  ← collection
{
  name: "sue",        ← field: value
  age: 26,            ← field: value
  status: "pending"   ← field: value
}                    } document
)
```

Exemple d'insertion d'un document dans la collection "users"

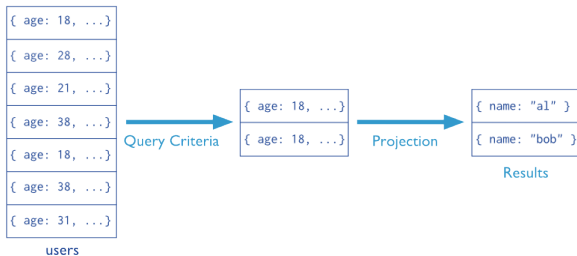


Exemples de writeConcern : "journal" et "2 écritures"

Opération FIND

Collection Query Criteria Projection

```
db.users.find( { age: 18 }, { name: 1, _id: 0 } )
```



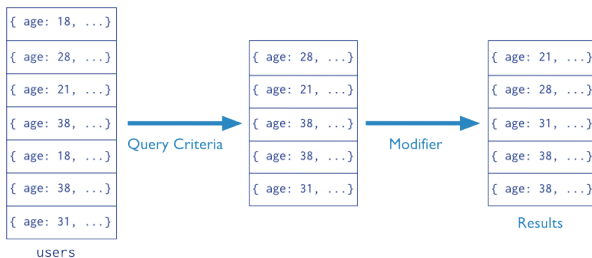
Exemple de requête avec sélection et projection : retourne tous les documents contenant un champ âge égal à 18 en ne gardant que le champ nom (exclusion du champ "_id")

<http://docs.mongodb.com/manual/crud/>

<http://docs.mongodb.org/manual/reference/operator/query/>

Opération FIND avec tri

Collection Query Criteria Modifier
`db.users.find({ age: { $gt: 18 } }).sort({age: 1 })`



*Exemple de requête avec fonction sur curseur de résultats :
retourne tous les documents contenant un champ âge supérieur à
18 et en triant les documents résultat par âge croissant*

<http://docs.mongodb.com/manual/reference/method/js-cursor/>

Opération UPDATE

```
db.users.updateMany(  
  { age: { $lt: 18 } },  
  { $set: { status: "reject" } }  
)
```

← collection
← update filter
← update action


Exemple de mise à jour du statut (nouvelle valeur "reject") pour tous les documents de la collection "users" contenant un champ âge inférieur à 18

<http://docs.mongodb.com/manual/tutorial/update-documents/>

Commande `db.coll.updateOne()` pour mettre à jour le premier document trouvé

Opération DELETE

```
db.users.deleteMany(  
  { status: "reject" }  
)
```



Exemple de suppression de tous les documents de la collection "users" dont le statut vaut "reject"

<http://docs.mongodb.com/manual/tutorial/remove-documents/>

Commande `db.coll.deleteOne()` pour supprimer le premier document trouvé

SQL versus MongoDB

SQL (MariaDB)	MongoDB
<pre>INSERT INTO people(user_id, age, status) VALUES ("bcd001", 45, "A")</pre>	<pre>db.people.insertOne({ user_id: "bcd001", age: 45, status: "A" })</pre>
<pre>SELECT user_id, status FROM people WHERE status = "A"</pre>	<pre>db.people.find({ status: "A" }, { user_id: 1, status: 1, _id: 0 })</pre>
<pre>UPDATE people SET status = "C" WHERE age > 25</pre>	<pre>db.people.updateMany({ age: { \$gt: 25 } }, { \$set: { status: "C" } })</pre>
<pre>DELETE FROM people WHERE status = "D"</pre>	<pre>db.people.deleteMany({ status: "D" })</pre>

<http://docs.mongodb.org/manual/reference/sql-comparison/>

En résumé

MongoDB, un SGBD orienté documents :

- ▶ Requêtes CRUD et requêtes agrégatives (regroupements)
- ▶ Indexation
- ▶ Aspects sécurité et administration
- ▶ Sharding (distribution des données) et réplication
- ▶ Map Reduce avec Javascript (distribution des traitements)
- ▶ Des "drivers" dans une quinzaine de langages (e.g., Jongo)

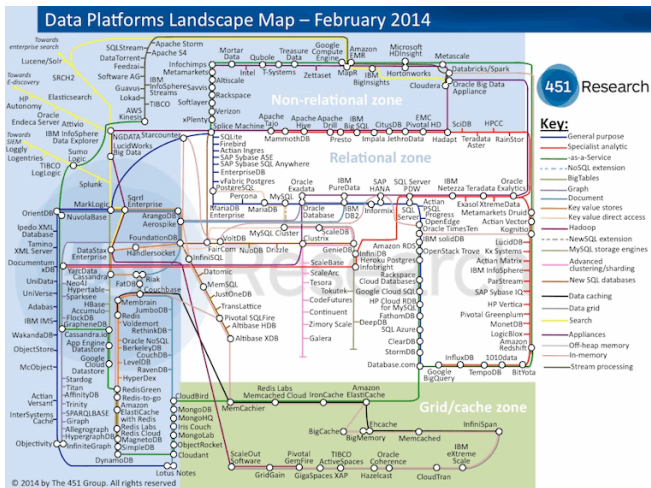
Vidéos tutorielles (~ une dizaine sur les concepts, < 20 minutes) :

<http://mrbool.com/course/introduction-to-mongodb/323>

<http://docs.mongodb.org/ecosystem/drivers/>

<http://jongo.org/> ou <https://pymongo.readthedocs.io/>

Bilan



http://blogs.the451group.com/information_management/2014/11/18/updated-data-platforms-landscape-map/

Bilan (2)

- ▶ **Big Data + web** \Rightarrow nouveaux besoins pour la modélisation, le stockage et le traitement de données **volumineuses**, véloces (**flux**) et variées (**hétérogénéité**)
- ▶ **Mouvement NoSQL / NoRel / NewSQL :**
 - ▶ paradigmes clé-valeur, colonne, document et graphe
 - ▶ théorème de CAP (consistency, availability, partition tolerance)
- ▶ **MongoDB** (concepts, modélisation, requêtes IFUD)

Perspectives : traitement distribué (Map-Reduce et chaînes d'opérations), répartition des données (MIF24 - BD réparties)