NOSQL

Les bases de données

Au programme

JSON

MONGODB

NOSQL

ELASTICSEARCH

Définition

Définition

NoSQL = Not Only SQL

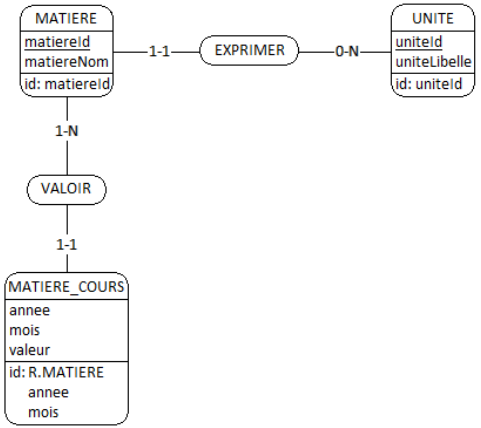
- Ne pas le considérer comme étant juste une opposition au monde SQL
- Propose des structures de données diverses et variées pour répondre à de nouveaux besoins.
- Historique :
  - SQL : années 1980 : Modèle qui domine le stockage de données relationnel et transactionnel
    - Oracle : 1983
    - PostGreSQL : 1985
    - MySQL : 1994
  - NoSQL : Courant 2000 porté par les GAFA et des besoins spécifiques lié à leurs métier et au volume de données :
    - MongoDB : 2007
    - Cassandra : 2008
    - Redis : 2009
    - ElasticSearch : 2010
    - InfluxDB : 2013

Définition

SQL

- Les bases de données SQL reposent sur un modèle relationnel et transactionnel.
- Les données sont organisées en schéma et table.
- Une table contient des colonnes pour définir les attributs de la donnée.
- Les lignes correspondent aux enregistrements.

- On parle de MCD (Modèle Conceptuel de Données)



productID	productName	categoryName
1	Chai	Beverages
2	Chang	Beverages
3	Aniseed Syrup	Condiments
4	Chef Anton's Cajun Seasoning	Condiments
5	Chef Anton's Gumbo Mix	Condiments
6	Grandma's Boysenberry Spread	Condiments
7	Uncle Bob's Organic Dried Pears	Produce
8	Northwoods Cranberry Sauce	Condiments

Théorème de CAP

Théorème de CAP

SQL : Les transactions ACID

Une transaction est une suite d'instructions de lecture/écriture qui garantit la cohérence des données.

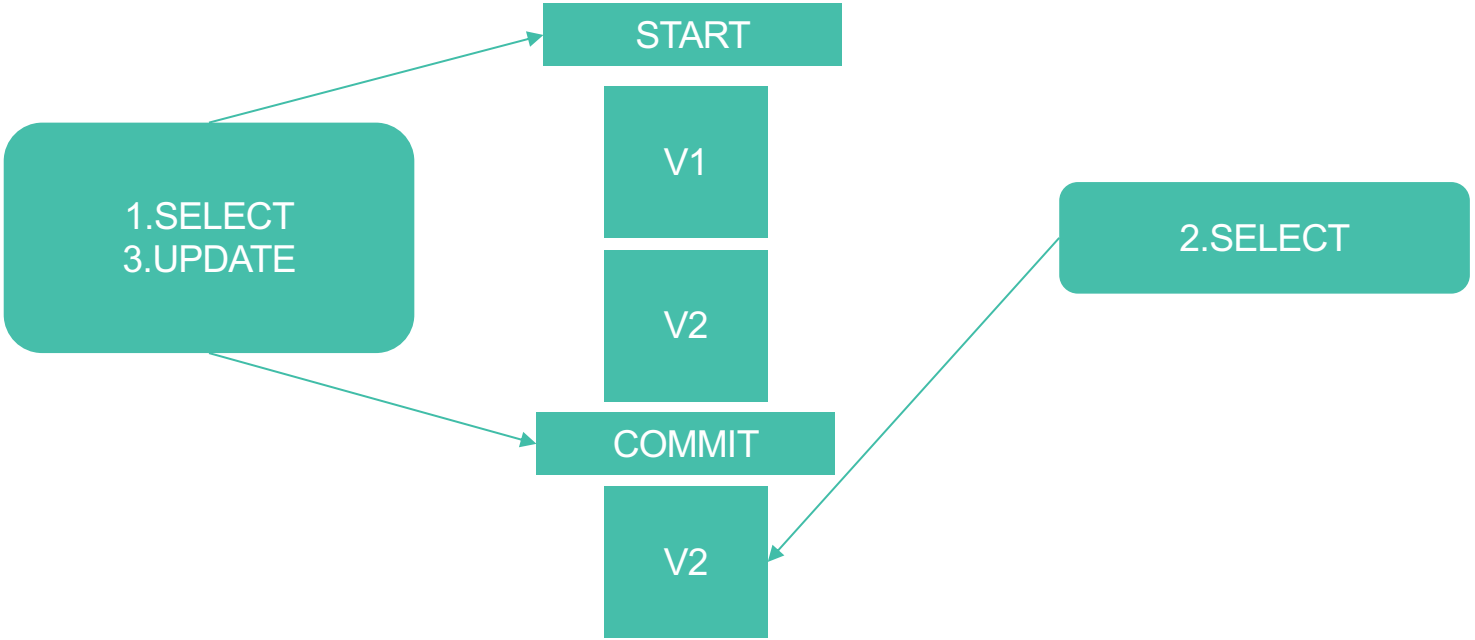
- **Atomicité** : Une transaction s'effectue entièrement ou pas du tout.
- **Cohérence** : Le contenu d'une base doit être cohérent au début et à la fin d'une transaction.
- **Isolation** : Les modifications d'une transaction ne sont visibles/modifiables que quand celle-ci a été validée.
- **Durabilité** : Une fois la transaction validée, l'état de la base est permanent (non affecté par les pannes ou autres).

```
START TRANSACTION
SELECT * FROM ….
UPDATE personnes set nom=« toto » where id=1
UPDATE personnes set age=« 30 » where id=1
COMMIT;
```

- Le NOSQL veut se défaire de ces bases afin de permettre d'améliorer la distribution des données, la scalabilité des bases et les performances.

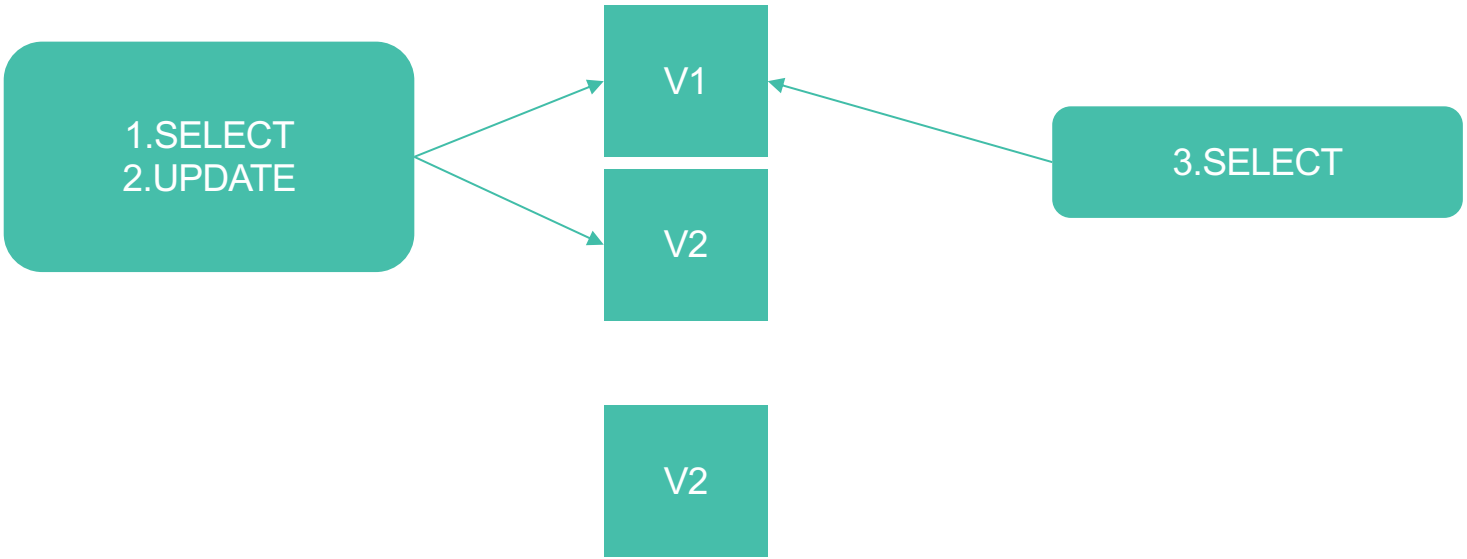
Théorème de CAP

SQL : Les transactions ACID



Théorème de CAP

Les lectures impropres



Théorème de CAP

NOSQL : Les propriétés BASE

- **Basically Available** : quelle que soit la charge de la base de données (données/requêtes), le système garanti un taux de disponibilité de la donnée.
- **Soft-state** : La base peut changer lors des mises à jour ou lors d'ajout/suppression de serveurs. La base NoSQL n'a pas à être cohérente à tout instant.
- **Eventually consistent** : À terme, la base atteindra un état cohérent.

- Le relâchement des contraintes liées aux transactions permet d’améliorer significativement les performances.

Théorème de CAP

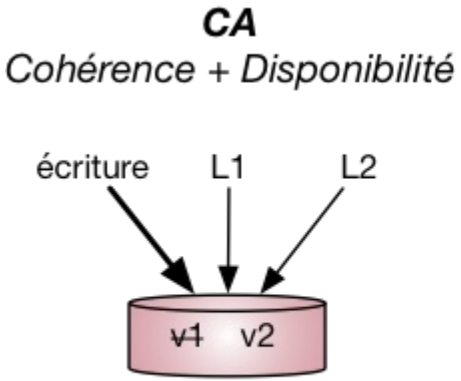
Le théorème

- **Consistency (Cohérence)** : Une donnée n'a qu'un seul état visible quel que soit le nombre de réplicas
- **Availability (Disponibilité)** : Tant que le système tourne (distribué ou non), la donnée doit être disponible
- **Partition Tolerance (Distribution)** : Quel que soit le nombre de serveurs, toute requête doit fournir un résultat correct

- Dans toute base de données, vous ne pouvez respecter au plus que 2 propriétés parmi la cohérence, la disponibilité et la distribution.

Théorème de CAP

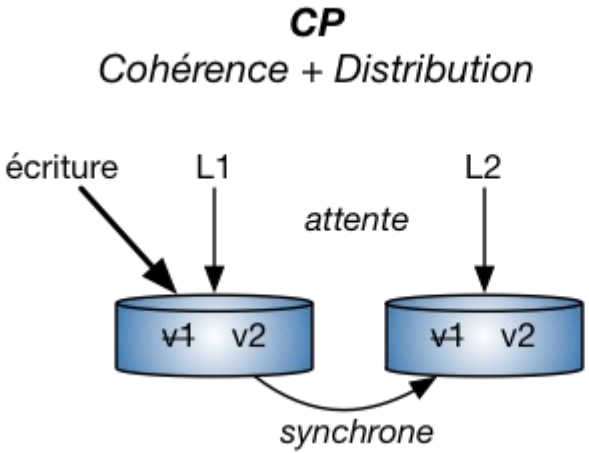
Le théorème



- le couple CA (Consistency-Availability) : il représente le fait que lors d'opérations concurrentes sur une même donnée, les requêtes L1 et L2 retournent la nouvelle version (v2) et sans délai d'attente. Cette combinaison n'est possible que dans le cadre de bases de données transactionnelles telles que les SGBDR.

Théorème de CAP

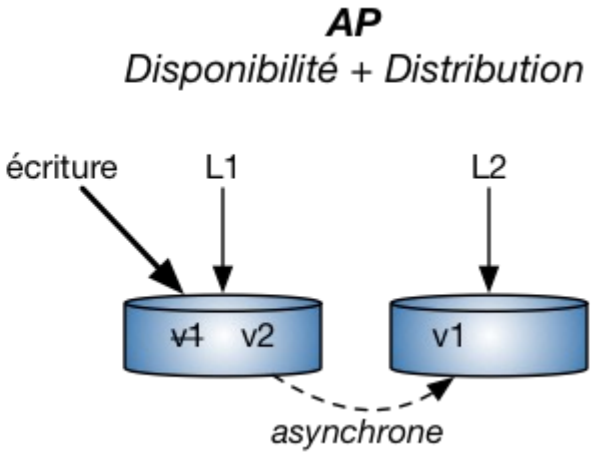
Le théorème



- Le couple CP (Consistency-Partition Tolerance) : propose maintenant de distribuer les données sur plusieurs serveurs en garantissant la tolérance aux pannes (réplication). En même temps, il est nécessaire de vérifier la cohérence des données en garantissant la valeur retournée malgré des mises à jour concurrentielles. La gestion de cette cohérence nécessite un protocole de synchronisation des réplicas, introduisant des délais de latence dans les temps de réponse (L1 et L2 attendent la synchronisation pour voir v2). C'est le cas de la base NoSQL MongoDB.

Théorème de CAP

Le théorème



- Le couple AP (Availability-Partition Tolerance) à contrario s'intéresse à fournir un temps de réponse rapide tout en distribuant les données et les réplicas. De fait, les mises à jour sont asynchrones sur le réseau, et la donnée est "Eventually Consistent" (L1 voit la version v2, tandis que L2 voit la version v1). C'est le cas de Cassandra dont les temps de réponses sont appréciables, mais le résultat n'est pas garanti à 100% lorsque le nombre de mises à jour simultanées devient important.

- Exemple : les serveurs DNS

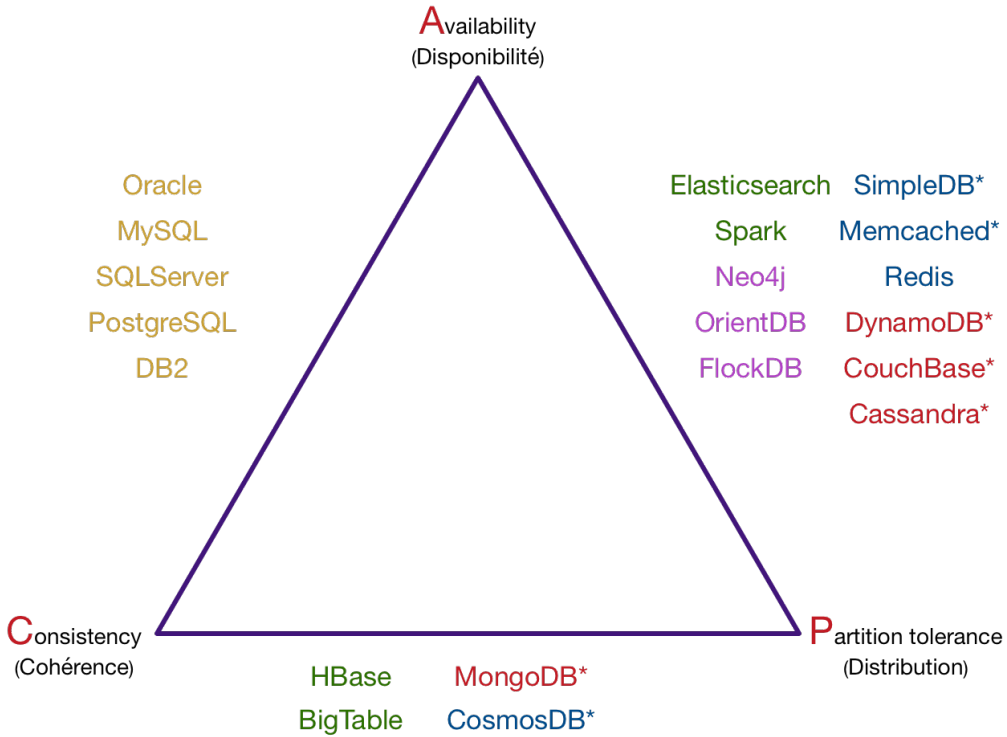
Théorème de CAP

Le triangle de CAP

Modèle

Relationnel
Orienté Clé-Valeur
Orienté Colonne
Orienté Document
Orienté Graphe

* Possibilité de changer
le mode de cohérence
Cohérence <-> Disponibilité

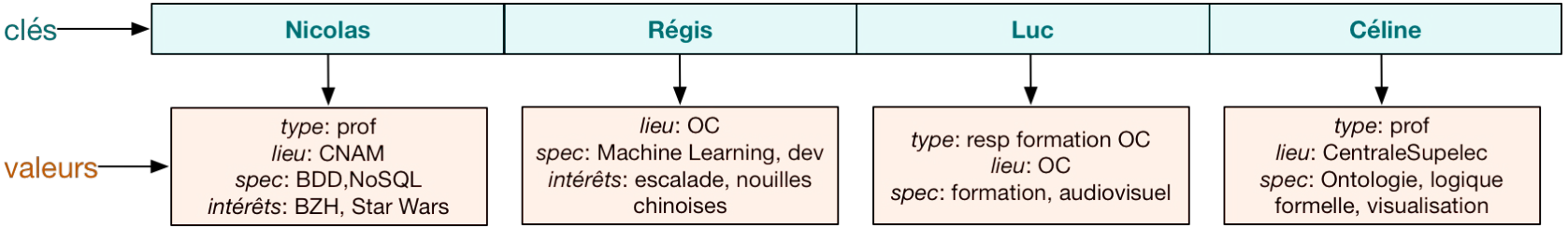


Bases de données NoSQL

Bases de données NoSQL

Orientée clé/valeur

- Chaque élément est stocké sous la forme d'une paire de clé/valeur. Les magasins de clé/valeur sont les plus simples parmi les bases de données NoSQL.
- Stockage de données uniquement accessibles par la clé.
- Pas de langage de requête.
- Souvent utilisés pour des données volatiles

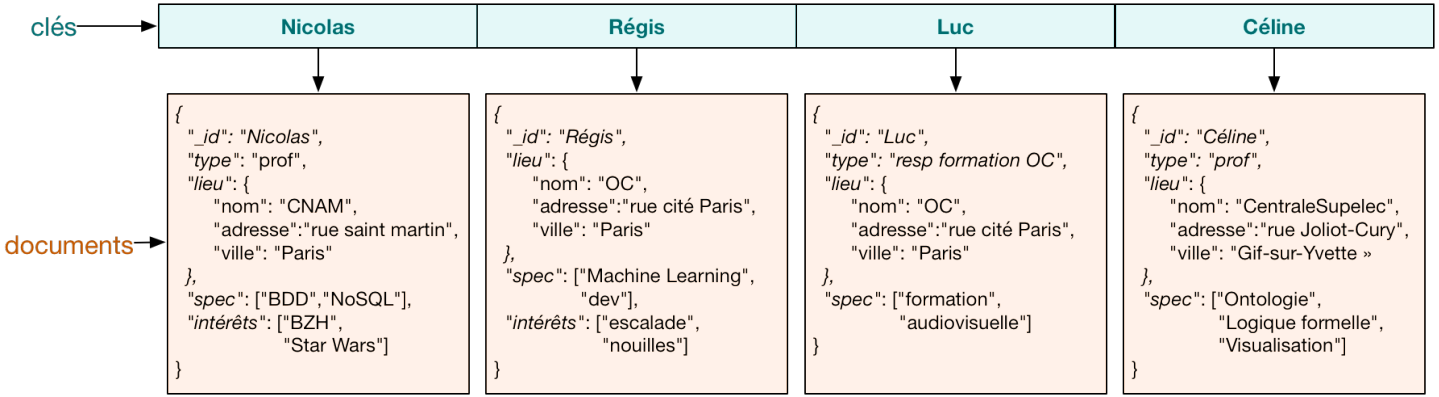


- Utilisation : Gestion de cache, panier d'achat, …
- Exemple : Redis

Bases de données NoSQL

Orientée documents

- Ces bases de données associent généralement chaque clé à une structure de données complexe qui s'appelle un document. Les documents peuvent contenir des paires de tableaux de clés ou des paires clé-valeur ou même des documents imbriqués. Elle possède un langage de requête complexe. On y retrouve couramment des modèles Json.

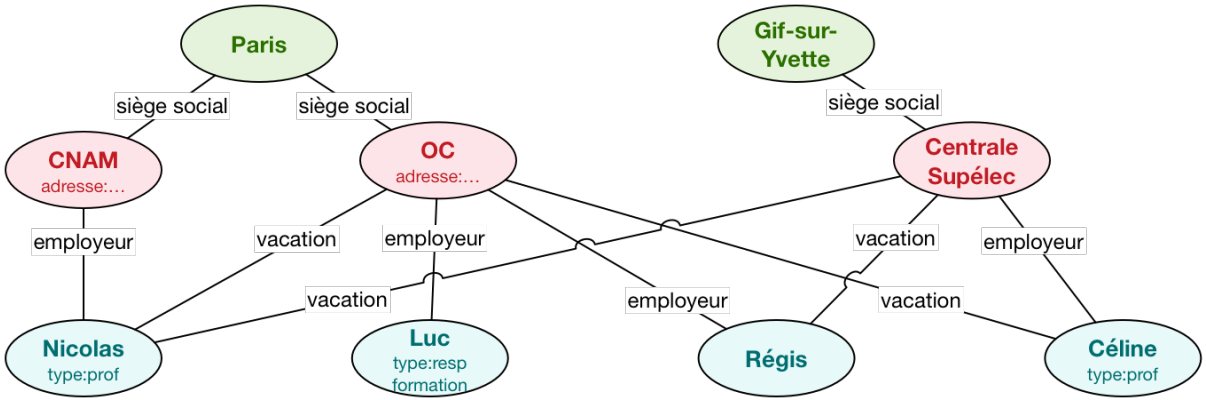


- Utilisation : gestion de contenu (bibliothèques numériques, collections de produits, dépôts de logiciels, collection multimédia, etc.), collection d'événements complexes, gestion des historiques d'utilisateurs sur réseaux sociaux.
- Exemple : MongoDB

Bases de données NoSQL

Orientée Graph

- Ces DB stockent des informations sur les graphiques, les réseaux, tels que les connexions sociales.



- Utilisation : réseaux sociaux (recommandation, plus court chemin, cluster...), réseaux SIG (routes, réseau électrique, fret...), web social (Linked Data)
- Exemple : Neo4J

Bases de données NoSQL

Orientée colonnes

- Ces types de bases de données sont optimisés pour les requêtes sur de grands ensembles de données, et au lieu de lignes, ils stockent des colonnes de données ensemble.

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	CentraleSupelec	Ontologie, logique formelle, visualisation	

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

- Utilisation : Comptage (vote en ligne, compteur, etc), journalisation, recherche de produits dans une catégorie
- Exemple : ElasticSearch

Bases de données NoSQL

Times Series

- Ces types de bases de données sont optimisés pour stocker et requêter des données horodatés.

	Date	Open	High	Low	Close	Adj Close	Volume
0	2009-12-21	33.119999	33.480000	33.029999	33.470001	24.608217	1952800
1	2009-12-22	33.549999	33.799999	33.389999	33.779999	24.836138	1715700
2	2009-12-23	33.680000	34.369999	33.630001	34.340000	25.247873	2565000
3	2009-12-24	34.220001	34.520000	34.119999	34.509998	25.372852	1043700
4	2009-12-28	34.400002	34.400002	34.080002	34.270000	25.398802	1247800

- Utilisation : Internet des Objet (capteur, position, …), Monitoring, …
- Exemple : InfluxDB

Au programme

JSON

MONGODB

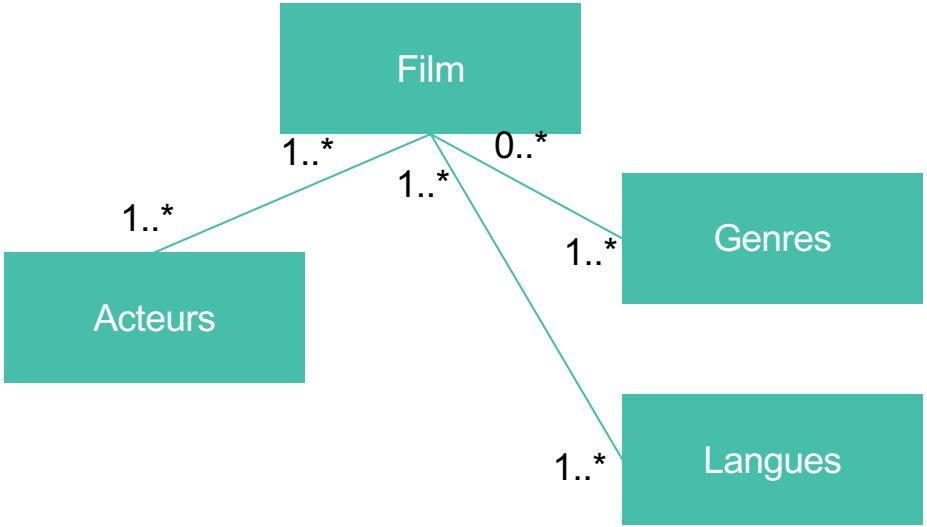
NOSQL

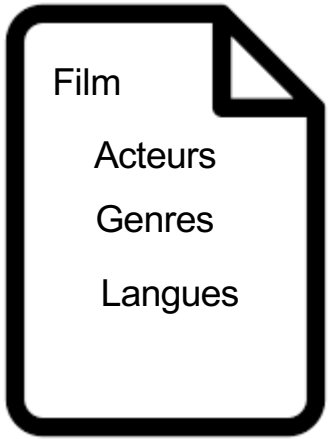
ELASTICSEARCH

MongoDB

Présentation

- MongoDB est une base de données orientées documents (et non pas relationnelle)



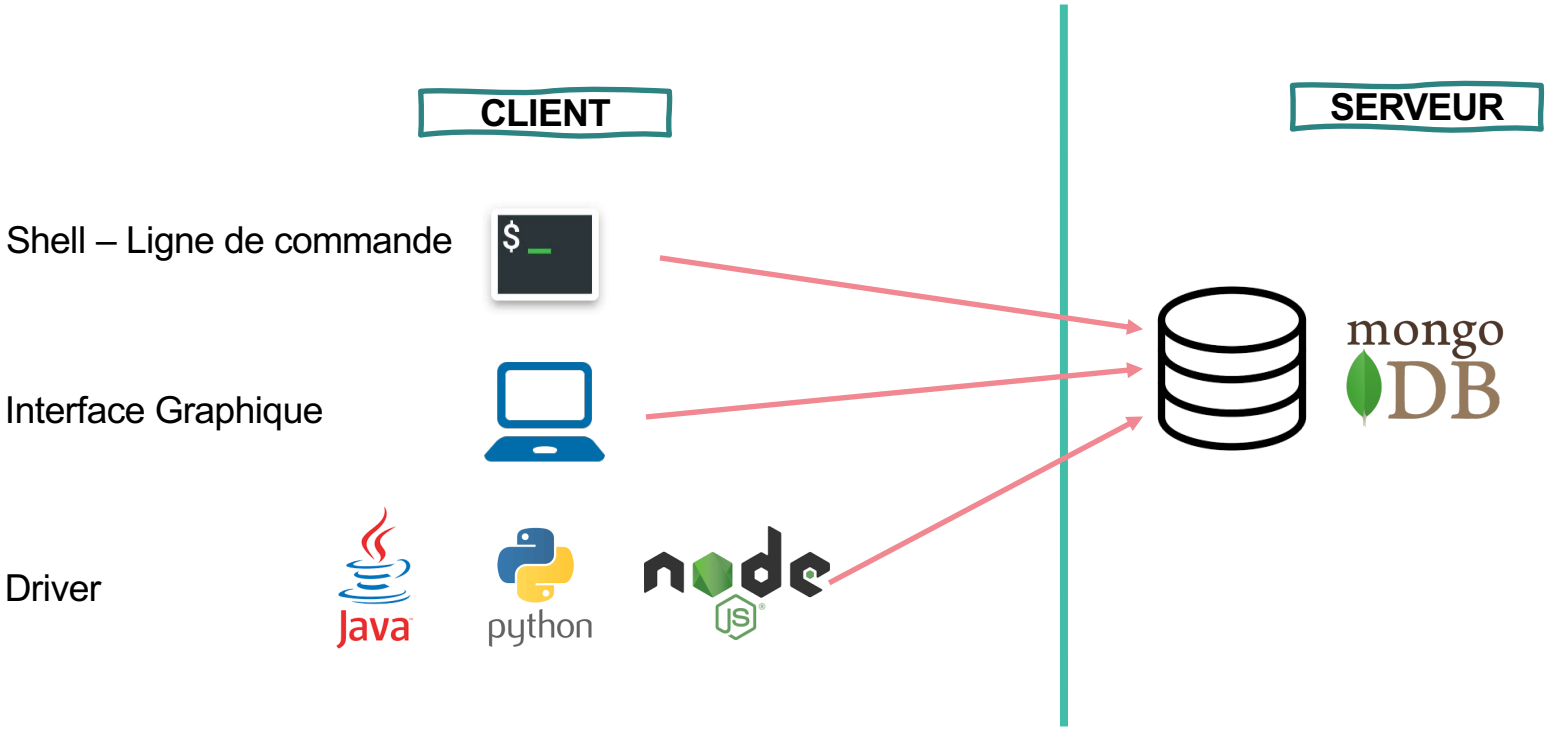


MongoDB

Organisation de la données

MongoDB

Client/Serveur



MongoDB

Concepts clés

- **Pas de jointure**
  - Les données sont regroupées dans un même document
  - Le document contient les données nécessaires
  - **Concepts SQL à oublier** : INNER JOIN, LEFT JOIN, …
  - **Concept nouveau** : Duplication de données

MongoDB

Concepts clés

- **Pas de schéma de base de données**
  - Le modèle ne permet pas de contraindre la donnée
  - On ne peut pas typer les données attendues
  - Un schéma peut contenir des données différentes (ex: une fiche utilisateur et une fiche de film)
  - **Concept SQL à oublier** : Foreign key, not null, …
  - **Concept nouveau** : C'est l'application qui connait le format des données et non plus la base de données.

```
CREATE TABLE clients(
   Nom char(30) NOT NULL,
   Prenom char(30) NOT NULL,
   Age integer, check (age < 100),
   Email  char(50) NOT NULL, check (Email LIKE "%@%")
)
```

MongoDB

Concepts clés

- **Pas de transaction**
  - Il n’est pas possible d’ouvrir une transaction, modifier et lire des documents puis fermer ou rollbacker la transaction
  - Repose sur le principe d’atomicité des requêtes : insérer, supprimer, modifier, ….
  - Les requêtes sont atomiques au niveau du document.
  - **Concept à oublier** : Ouvrir une transaction, lire les données, modifier les données, commiter la transaction.
  - **Concept Nouveau** : Rechercher et modifier les données en une seule requête (Atomicité).

MongoDB

Concepts clés

- **MAIS …**

  Maintenant, MongoDB propose des transactions, des jointures entre documents et des schémas de données

  Cependant, ces notions ne doivent pas être mises en avant pour rester dans la philosophie NoSQL

MongoDB

JSON et BSON

- MongoDB repose grandement sur le JSON pour le format de ces données et des requêtes
- Il utilise le BSON (Binary) pour le stockage des données.
- Le BSON est un JSON étendu
- Il contient des types de données supplémentaires, voici les principales :

Type	
Objectid	L'identifiant unique d'un document (_id)
Date	Les dates
Timestamp	Les timestamps
Binary Data	Données binaires

https://docs.mongodb.com/manual/reference/bson-types/

MongoDB

L'objectID

- Equivalent de la clé primaire et des séquences en SQL
- Se nomme toujours _id
- Permet d'identifier un document de façon unique
- Il est ordonné
- Il peut être fourni lors de l'insertion d'un document
- S'il n'est pas présent, MongoDB le génère automatiquement
- Prends la forme d'un UUID (longueur de 12 bytes en hexadécimal) :
  - 4-byte *timestamp*, Représente la date de création de l'objet
  - 5-byte *généré aléatoirement par le process*. Unique par serveur et processus.
  - 3-byte valeur incrémentale, Initialisé avec une valeur aléatoire

"_id" : ObjectId("5197c6b453cce2ec3a743811")

MongoDB

Les dates

- Date() : Retourne la date courante en String
- new Date() : Retourne la a date courante dans un objet ISODate (<YYYY-mm-ddTHH:MM:ssZ>)

- Attention : Les dates (notamment les heures) sont stockés en UTC. Attention au décalage horaire et au manipulation.

```
db.insertDocumentWithDateDemo.insertOne(
{"UserName":"Larry","UserMessage":"Hi","UserMessagePostDate":new Date("2012-09-24")}
);
```

```
{
"_id" : ObjectId("5c9cca58a629b87623db1b16"),
"UserName" : "Larry",
"UserMessage" : "Hi",
"UserMessagePostDate" : ISODate("2012-09-24T00:00:00Z")
}
```

MongoDB

Les dates

Insertion d'une date en UTC

```
db.date.insertOne(
{"UserName":"Larry","UserMessage":"Hi",
"UserMessagePostDate":new Date("2012-09-24T12:00:00Z")}
);
```

```
{
    "_id" : ObjectId("618a445e90b59b2dba28cb5c"),
    "UserName" : "Larry",
    "UserMessage" : "Hi",
    "UserMessagePostDate" : ISODate("2012-09-24T12:00:00.000Z")
}
```

Insertion d'une date en timezone locale

```
db.date.insertOne(
{"UserName":"Larry","UserMessage":"Hi",
"UserMessagePostDate":new Date("2012-09-24T12:00:00")}
);
```

```
{
    "_id" : ObjectId("618a44a590b59b2dba28cb5d"),
    "UserName" : "Larry",
    "UserMessage" : "Hi",
    "UserMessagePostDate" : ISODate("2012-09-24T10:00:00.000Z")
}
```

Modélisation de données

Modélisation de données

Présentation

- Le challenge est de trouver un équilibre entre les besoins de l'application, les performances et l'organisation des données.
- Il faut prioriser l'utilisation de la donnée et sa manipulation par rapport à la structuration naturel de la données.

- Schéma Flexible :
  - Une collection peut stocker des documents avec des structurations complétement différentes : Stocker un document correspondant à une fiche d'une personne avec une document correspondant à une fiche de d'une voiture.

- Cette flexibilité facilite le mapping vers des objets et facilite les évolutions de la structure de données
- **Ce n'est pas le modèle qui connaît la structure de la donnée mais l'application**

- **En pratique : Une collection ne contient que des documents avec des structures similaires ou de légères différences (ex: héritage objet)**

Modélisation de données

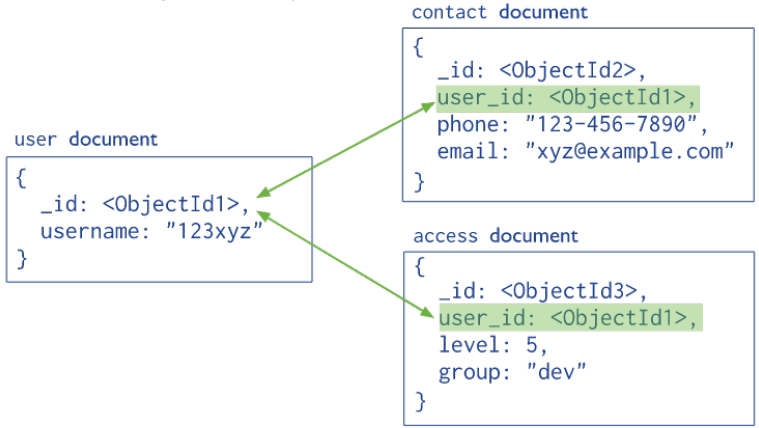
Concept de modélisation – Sous document

- Une modélisation possible est l'utilisation de « Embedded sub-document » : Notion de sous-document
- Corresponds aux objets JSON
- On parle de modèle dénormalisé

Modélisation de données

Concept de modélisation – Référence

- Utilisation de référence entre Collection pour faire référence à des documents.
- Corresponds au « Foreign Key » du SQL.
- On parle de modèle normalisé
- Attention, il s'agit uniquement de lien logique. Il n'y a pas de contrôle de cohérence de la base de données.



- D'un point de vue conception, il est préférable d'utiliser une clé métier et non technique

Modélisation de données

Concept de modélisation – One to One with embedded document

- Représentation d'une relation One to One
- Performant si l'adresse doit être manipulée avec le nom

```
{
   _id: "joe",
   name: "Joe Bookreader",
   address: {
              street: "123 Fake Street",
              city: "Faketon",
              state: "MA",
              zip: "12345"
            }
}
```

Modélisation de données

Concept de modélisation – One to One avec référence

- Le regroupement des informations dans un document peut donner un document lourd, le chargement d'information inutile.
- Une relation « One to One » peut être modélisée avec une référence. Regrouper les données les plus utilisées.

```
{
  "_id": 1,
  "title": "The Arrival of a Train",
  "year": 1896,
  "runtime": 1,
  "released": ISODate("1896-01-25"),
  "type": "movie",
  "directors": [ "Auguste Lumière", "Louis Lumière" ],
  "countries": [ "France" ],
  "genres": [ "Documentary", "Short" ],
}
```

```
{
  "_id": 156,
  "movie_id": 1, // reference to the movie collection
  "poster": "http://ia.media-imdb.com/images/image.jpg",
  "plot": "A group of people, ...",
  "fullplot": "A group of people ..",
  "imdb": {
    "rating": 7.3,
    "votes": 5043,
    "id": 12
  },
  "tomatoes": {
    "viewer": {
      "rating": 3.7,
      "numReviews": 59
    },
    "lastUpdated": ISODate("2020-01-29T00:02:53")
  }
}
```

Modélisation de données

Concept de modélisation – One to many

- Modélisation avec des sous-documents
- L'ensemble des informations est dans le document

- Ex : une personne possède plusieurs adresses

```
{
  "_id": "joe",
  "name": "Joe Bookreader",
  "addresses": [
                {
                  "street": "123 Fake Street",
                  "city": "Faketon",
                  "state": "MA",
                  "zip": "12345"
                },
                {
                  "street": "1 Some Other Street",
                  "city": "Boston",
                  "state": "MA",
                  "zip": "12345"
                }
              ]
}
```

Modélisation de données

Concept de modélisation – One to many

- Modélisation avec des sous-documents et référence
- Le problème peut être la taille de la liste. Ex : une liste d'avis sur un produit peut contenir des milliers de ligne
- Une possibilité est de conserver une sous-partie des avis les plus récents dans le produit et d'avoir une seconde collection avec la liste des avis : Duplication de données

Collection product

```
{
  "_id": 1,
  "name": "Super Widget",
  "description": "This is the most useful item in your toolbox.",
  "price": { "value": NumberDecimal("119.99"), "currency": "USD" },
  "reviews": [
    {
      "review_id": 786,
      "review_author": "Kristina",
      "review_text": "This is indeed an amazing widget.",
      "published_date": ISODate("2019-02-18")
    }
    ...
    {
      "review_id": 777,
      "review_author": "Pablo",
      "review_text": "Amazing!",
      "published_date": ISODate("2019-02-16")
    }
  ]
}
```

Collection review

```
{
  "review_id": 786,
  "product_id": 1,
  "review_author": "Kristina",
  "review_text": "This is indeed an amazing widget.",
  "published_date": ISODate("2019-02-18")
}
…
{
  "review_id": 1,
  "product_id": 1,
  "review_author": "Trina",
  "review_text": "Nice product. Slow shipping.",
  "published_date": ISODate("2017-02-17")
}
```

Modélisation de données

Concept de modélisation – One to many

- Modélisation par référence.
- La duplication de la donnée n'est pas intéressante ou non conforme à l'application

```
{
   title: "MongoDB: The Definitive Guide",
   author: [ "Kristina Chodorow", "Mike Dirolf" ],
   published_date: ISODate("2010-09-24"),
   pages: 216,
   language: "English",
   publisher: {
            name: "O'Reilly Media",
            founded: 1980,
            location: "CA"
          }
}
```

Nous voulons utiliser une référence pour le publisher.

Modélisation de données

Concept de modélisation – One to many – Solution 1

Collection Publisher

```
{
   name: "O'Reilly Media",
   founded: 1980,
   location: "CA",
   books: [123456789, 234567890, ...]
}
```

Collection book

```
{
   _id: 123456789,
   title: "MongoDB: The Definitive Guide",
   author: [ "Kristina Chodorow", "Mike Dirolf" ],
   published_date: ISODate("2010-09-24"),
   pages: 216,
   language: "English"
}
{
   _id: 234567890,
   title: "50 Tips and Tricks for MongoDB Developer",
   author: "Kristina Chodorow",
   published_date: ISODate("2011-05-06"),
   pages: 68,
   language: "English"
}
```

Modélisation de données

Concept de modélisation – One to many – Solution 2

Collection Publisher

```
{
  _id: "oreilly",
  name: "O'Reilly Media",
  founded: 1980,
  location: "CA"
}
```

Collection book

```
{
  _id: 123456789,
  title: "MongoDB: The Definitive Guide",
  author: [ "Kristina Chodorow", "Mike Dirolf" ],
  published_date: ISODate("2010-09-24"),
  pages: 216,
  language: "English",
  publisher_id: "oreilly"
}
{
  _id: 234567890,
  title: "50 Tips and Tricks for MongoDB Developer",
  author: "Kristina Chodorow",
  published_date: ISODate("2011-05-06"),
  pages: 68,
  language: "English",
  publisher_id: "oreilly"
}
```

Modélisation de données

Schéma de validation

- Possibilité de préciser un JSON schéma lors de la création d'une collection

```
db.createCollection("students", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: [ "name", "address" ],
      properties: {
        name: {
          bsonType: "string",
          description: "must be a string and is required"
        },
        address: {
          bsonType: "object",
          required: [ "city" ],
          properties: {
            street: {
              bsonType: "string",
              description: "must be a string if the field exists"
            },
            city: {
              bsonType: "string",
              description: "must be a string and is required"
            }}}}}})
```

TD

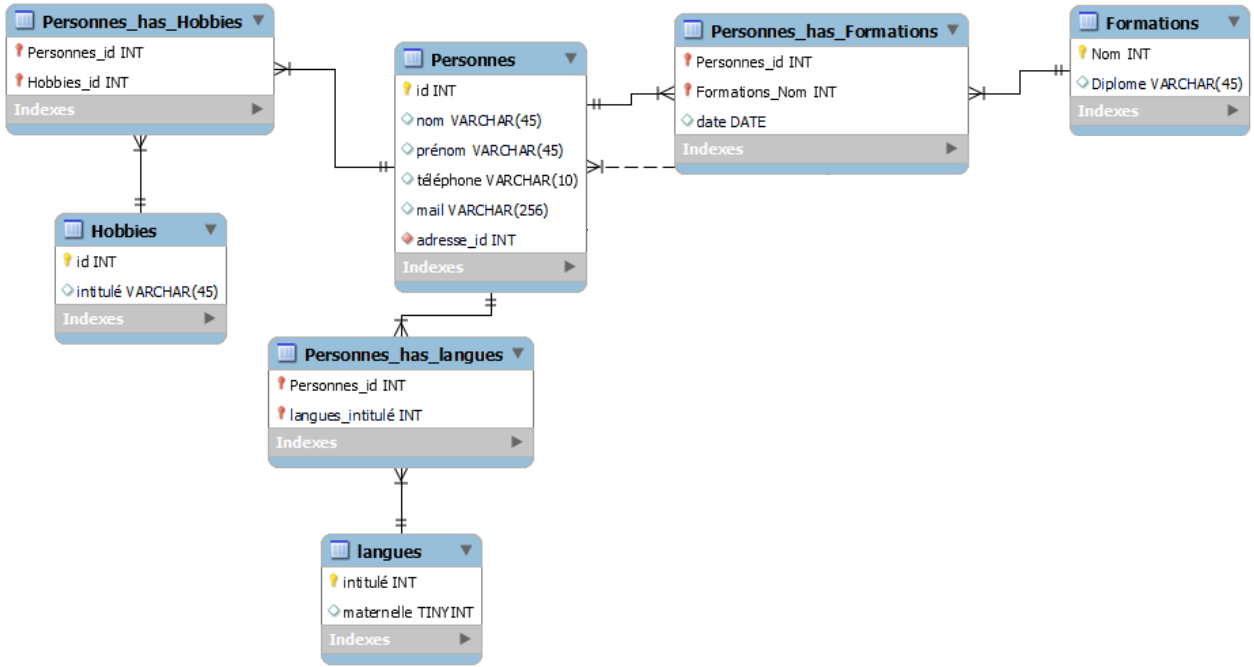
MongoDB

TD1 : Conception

- TD1.1 : Ecrire le MCD relationnel représentant une lise de CV avec les informations suivantes : nom, prénom, téléphone, mail, adresse, liste de formations, liste des hobbies, langues pratiquées.
- TD1.2 : Ecrire la représentation MongoDB correspondante
- TD1.3 : Ajouter au MCD et à la représentation MongoDB les informations suivantes : Une liste d'entreprise caractérisées par nom, SIRET, Adresse, téléphone, mail. Ajouter aux CV les expériences professionnelles en les liant aux entreprises.

MongoDB

TD1.1 : Correction



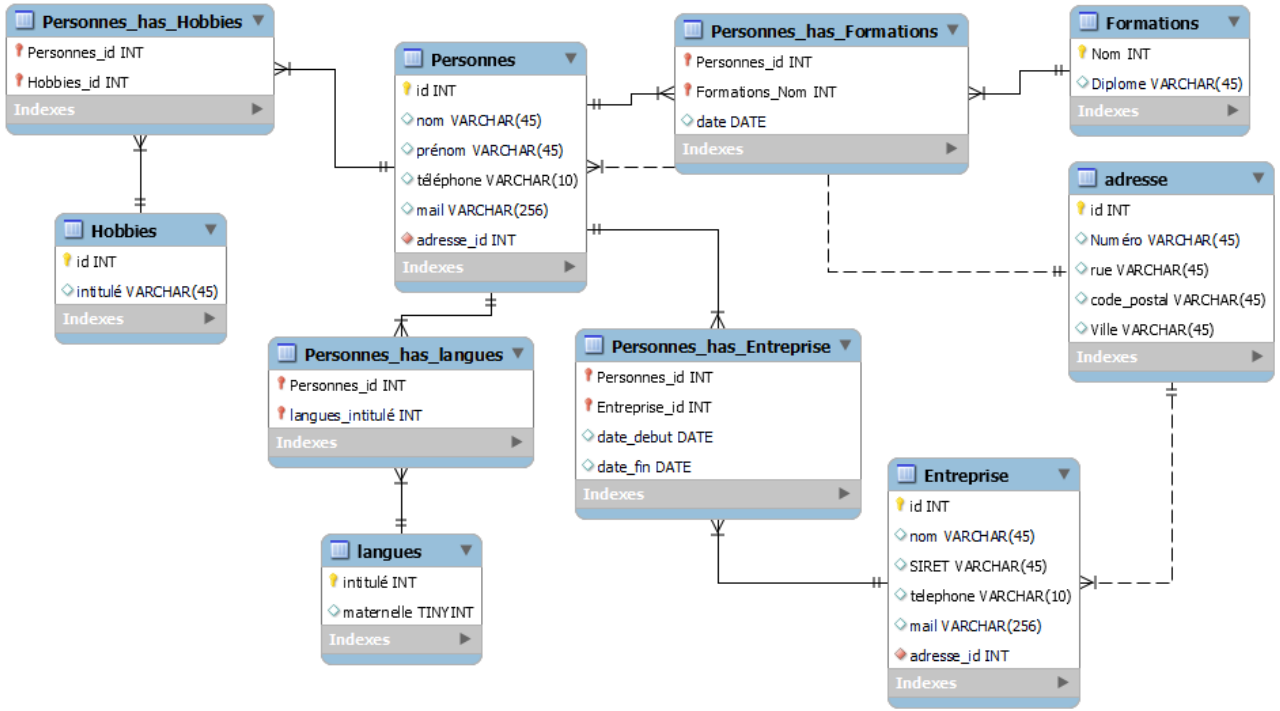
MongoDB

TD1.2 : Correction

- _id
- Nom
- Prénom
- Mail
- Téléphone
- Adresse:
  - Numéro
  - Rue
  - CodePostal
  - Ville
- Formations :
  - Intitulé
  - Date
- Langues :
  - Intitulé
  - Maternelle
- Hobbies
  - Intitulé

MongoDB

TD1.3 : Correction



MongoDB

TD1.3 : Correction

- _id
- Nom
- Prénom
- Mail
- Téléphone
- Adresse:
  - Numéro
  - Rue
  - CodePostal
  - Ville
- Formations :
  - Intitulé
  - Date
- Entreprises:
  - Entreprise_id
  - Date début
  - Date Fin
- …

- _id
- Nom
- Mail
- SIRET
- Téléphone
- Adresse:
  - Numéro
  - Rue
  - CodePostal
  - Ville

Les operations CRUD

Les opérations CRUD

Création (Create)

- Permet d'insérer des documents dans une collection

```
•db.collection.insertOne()
•db.collection.insertMany()
```

- Lors de l'insertion un champs _id sera généré par MongoDB si il n'est pas présent. C'est l'équivalent d'une clé primaire pour identifier le document.

Les opérations CRUD

Recherche (Read)

- Rechercher les documents dans une collection

```
•db.collection.find(query, projection)
```

paramètres	
query	Requête au format JSON pour restreindre la recherche
projection	Selection des champs du document à retourner par la requête (Optionnel)

```
db.users.find(
   { age: { $gt: 18 } },
   { name: 1, address: 1 }
).limit(5)
```

← collection
← query criteria
← projection
← cursor modifier

Les opérations CRUD

Recherche (Read)

requêtes	
db.users.find()	Recherche tous les documents de la collection
db.users.find({ _id: 5 })	Recherche le document dont l'identifiant est 5
db.users.find({ "name": "sue" })	Recherche les documents dont le champs « name » est égale à « sue »
db.users.find({ }, { name: 1 })	Recherche tous les documents mais ne retourne que le champs « name »
db.users.find({ birth: { $gt: new Date('1950-01-01') } })	Recherche tous les documents donc le champs « birth » est supérieur à la date.

Les opérations CRUD

Modification (Update)

- Permet de modifier un ou plusieurs document dans une collection

```
•db.collection.updateOne()
•db.collection.updateMany()
•db.collection.replaceOne()
```

```
db.users.updateMany(                    <-- collection
   { age: { $lt: 18 } },                <-- update filter
   { $set: { status: "reject" } }       <-- update action
)
```

Les opérations CRUD

Suppression (Delete)

- Permet de supprimer un ou plusieurs document dans une collection

- db.collection.deleteOne()
- db.collection.deleteMany()

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

collection

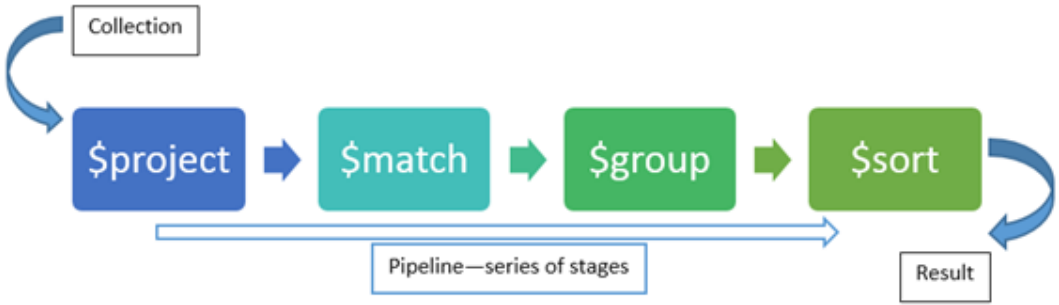
delete filter

Les pipelines d’agrégation

Les pipelines d’agrégations

Définition

- Correspond à une suite d’opérations sur des données et retourne le résultat
- Un pipeline contient une ou plusieurs opérations qui s’enchaînent dans l’ordre
- Chaque opération s’effectue sur les documents en entrée et fournit un ou des documents en sortie
- Les documents en sortie deviennent l’entrée de l’opération suivante.



Les pipelines d'agrégations

Exemple – 1/3

```
db.mycollection.aggregate([
   {$match:{“phone_type”:”smart”}},
   {$group:{“_id”:”$brand_name”,”total”:{$sum:”$price”}}}
])
```

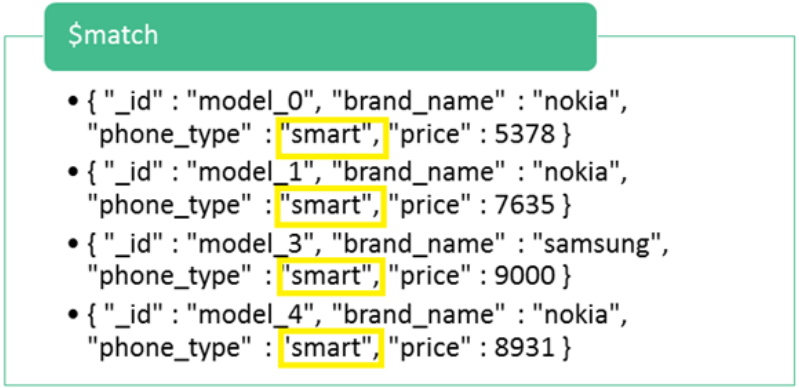
collection

- { "_id" : "model_0", "brand_name" : "nokia", "phone_type" : "smart", "price" : 5378 }
- { "_id" : "model_1", "brand_name" : "nokia", "phone_type" : "smart", "price" : 7635 }
- { "_id" : "model_2", "brand_name" : "samsung", "phone_type" : "feature", "price" : 1305 }
- { "_id" : "model_3", "brand_name" : "samsung", "phone_type" : "smart", "price" : 9000 }
- { "_id" : "model_4", "brand_name" : "nokia", "phone_type" : "smart", "price" : 8931 }

Les pipelines d'agrégations

Exemple – 2/3

```
db.mycollection.aggregate([
   {$match:{“phone_type”:”smart”}},
   {$group:{“_id”:”$brand_name”,”total”:{$sum:”$price”}}}
])
```



Les pipelines d’agrégations

Exemple – 3/3

```
db.mycollection.aggregate([
   {$match:{“phone_type”:”smart”}},
   {$group:{“_id”:”$brand_name”,”total”:{$sum:”$price”}}}
])
```

$group - result set

- { "_id" : "nokia", "total" : 21944 }
- { "_id" : "samsung", "total" : 9000 }

Les pipelines d'agrégations

La syntaxe

```
db.mycollection.aggregate([
   {$match:{“phone_type”:”smart”}},
   {$group:{“_id”:”$brand_name”,”total”:{$sum:”$price”}}}
])
```

- Conforme au JSON mais introduit un nouveau caractère : $
- `$match` : Un champs sans double quote commençant par $: Correspondant à un stage ou un opérateur prédéfini par MongoDB = une fonction
- `”$price” :` Un champs avec double quote et commençant par $: Fait référence à une propriété du document JSON

Les pipelines d'agrégations

Les Stages

Opérateurs	Description	Exemple
$project	Sélection (ajout ou suppression) des champs des documents	{ $project: { **"<field1>"**: 1, **"<field2>"**: 0, ... } }
$match	Filtre les documents qui correspondent à la condition	db.articles.aggregate([{ $match : { author : "dave" } }]);
$group	Regroupe les documents suivant un critère	db.date.aggregate([{$group:{"_id":"$UserName", total:{$sum:1 }}}])
$sort	Tri des documents	{ $sort: { date: 1, prix: -1} }
$limit	Limite le nombre de documents renvoyés	{ $limit: 10}
$unwind	Déconstruction d'un tableau	db.inventory.aggregate([{ $unwind : **"$sizes"** }])

Les pipelines d’agrégations

Les opérateurs

- Les opérateurs permettent l’exécution d’opération sur les documents
- Permet de créer un nouveau champs résultat de l’opération
- Il en existent plus de 100 …

- Exemple d’opération :
  - Arithmétique : Compter le nombre de documents, additionner des prix, faire une moyenne, multiplier, tronquer, …
  - Tableau : retourner le premier/dernier élément, trier le tableau, filtrer le tableau, …
  - Comparaison de valeur : supérieur, égale , …
  - Conditionnelle : If/else, switch, …
  - Date : ajouter des jours, le jour du mois, conversion, …
  - String : Concaténation, Upper Case, recherche par regex, tronquer, …
  - Trigonométrie : cosinus, tangente, …
  - …..

https://docs.mongodb.com/manual/reference/operator/aggregation/

Les pipelines d’agrégations

Les opérateurs - exemple

```
{ "_id" : 1, "item" : "abc", "price" : 10, "quantity" : 2, "date" : ISODate("2014-01-01T08:00:00Z") }
{ "_id" : 2, "item" : "jkl", "price" : 20, "quantity" : 1, "date" : ISODate("2014-02-03T09:00:00Z") }
{ "_id" : 3, "item" : "xyz", "price" : 5, "quantity" : 5, "date" : ISODate("2014-02-03T09:05:00Z") }
{ "_id" : 4, "item" : "abc", "price" : 10, "quantity" : 10, "date" : ISODate("2014-02-15T08:00:00Z") }
{ "_id" : 5, "item" : "xyz", "price" : 5, "quantity" : 10, "date" : ISODate("2014-02-15T09:05:00Z") }
```

```
db.sales.aggregate(
  [
    {
      $group:
        {
          _id: { day: { $dayOfYear: "$date"}, year: { $year: "$date" } },
          totalAmount: { $sum: { $multiply: [ "$price", "$quantity" ] } },
          count: { $sum: 1 }
        }
    }
  ]
)
```

```
{ "_id" : { "day" : 46, "year" : 2014 }, "totalAmount" : 150, "count" : 2 }
{ "_id" : { "day" : 34, "year" : 2014 }, "totalAmount" : 45, "count" : 2 }
{ "_id" : { "day" : 1, "year" : 2014 }, "totalAmount" : 20, "count" : 1 }
```

TD

Les pipelines d'agrégation

TD 1 : Quel est le résultat de ces agrégations ?

Données

```
db.orders.insertMany( [
{ _id: 0, productName: "Steel beam", status: "new", quantity: 10 },
{ _id: 1, productName: "Steel beam", status: "urgent", quantity: 20 },
{ _id: 2, productName: "Steel beam", status: "urgent", quantity: 30 },
{ _id: 3, productName: "Iron rod", status: "new", quantity: 15 },
{ _id: 4, productName: "Iron rod", status: "urgent", quantity: 50 },
{ _id: 5, productName: "Iron rod", status: "urgent", quantity: 10 }
] )
```

Agrégation 1

```
db.orders.aggregate( [
{ $match: { status: "urgent" } },
{ $group: { _id: "$productName", sumQuantity: { $sum: "$quantity" } } }
] )
```

Agrégation 2

```
db.orders.aggregate( [
   { $sort: { "quantity": -1 } },
   { $set : {"priorité": "$status"}},
   { $project: {"_id":1, "productName":1, "priorité":1 }},
   { $limit:3}
] )
```

Agrégation 3

```
db.orders.aggregate( [
   { $limit:3},
   { $match: {"status":"urgent"}},
   { $sort: { "quantity": 1 } },
] )
```

Les pipelines d’agrégation

TD 1 : Quel est le résultat de ces agrégations ? - Correction

Résultat 1

```
{"_id" : "Steel beam","sumQuantity" : 50.0}
{"_id" : "Iron rod","sumQuantity" : 60.0}
```

Les pipelines d’agrégation

TD 1 : Quel est le résultat de ces agrégations ? - Correction

Résultat 2

```
{"_id" : 4.0,"productName" : "Iron rod","priorité" : "urgent"}
{"_id" : 2.0,"productName" : "Steel beam","priorité" : "urgent"}
{"_id" : 1.0,"productName" : "Steel beam","priorité" : "urgent"}
```

Les pipelines d’agrégation

TD 1 : Quel est le résultat de ces agrégations ? - Correction

Résultat 3

```
{"_id" : 1.0, "productName" : "Steel beam", "status" : "urgent", "quantity" : 20.0}
{"_id" : 2.0, "productName" : "Steel beam", "status" : "urgent", "quantity" : 30.0}
```

Les indexes

Les indexes

Les indexes simple

- Comme toute base de données, l'indexation est **très importante** pour la performance
- Les indexes se posent au niveau des collections
- Sans index, MongoDB effectue un « Collection Scan » = Scan de chaque document dans la collection
- Un index consiste à demander au SGBD d'indexer les documents suivant une clé du document
- Avec un index, MongoDB ne parcourra qu'une sous partie des documents correspondant à l'index
- Les indexes peuvent être posés sur tous les champs d'un document ou d'un sous-document.
- Un index tri les documents de façon ascendante (1) ou descendante (-1).

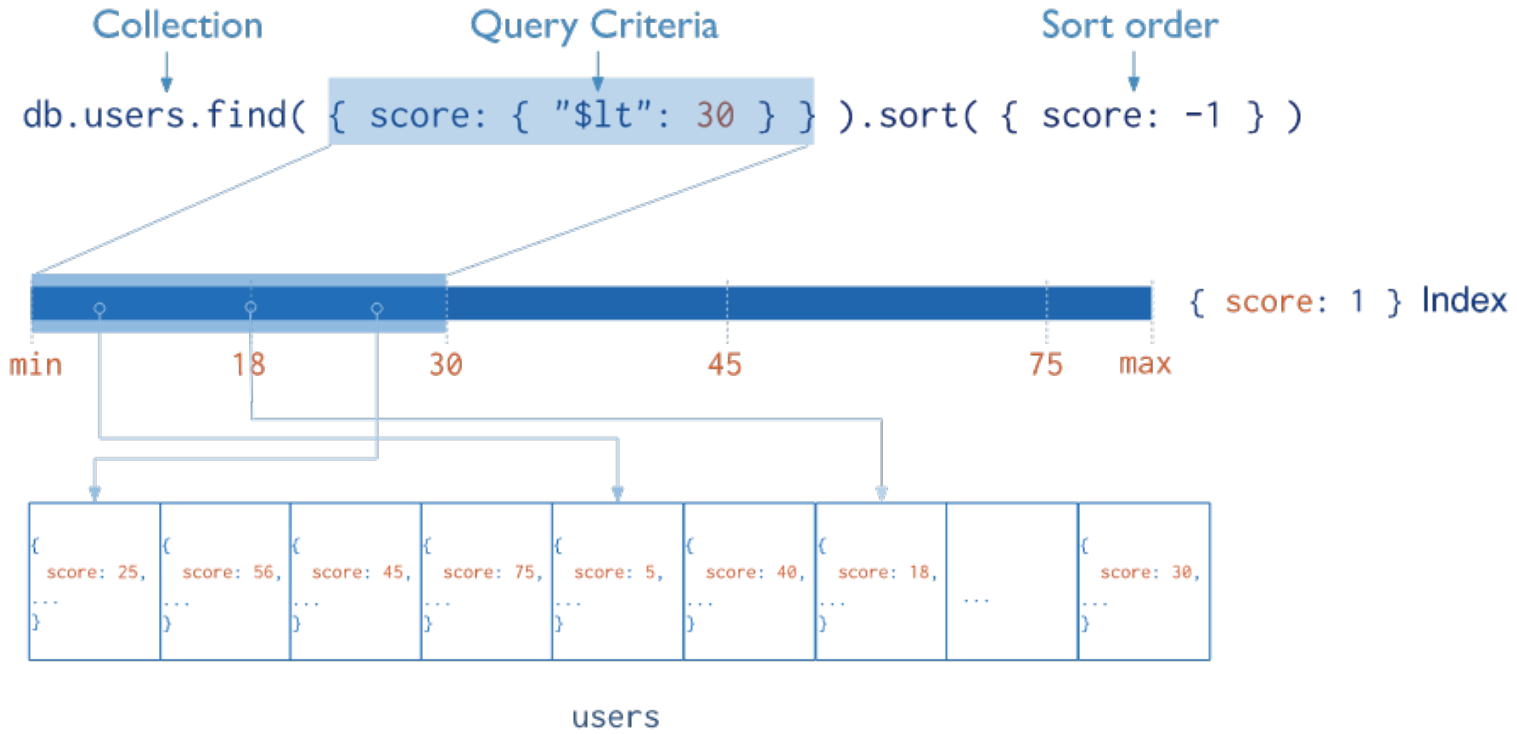
```
{
"_id": ObjectId("570c04a4ad233577f97dc459"),
"score": 1034,
"location": { state: "NY", city: "New York" }
}
```

```
db.records.createIndex( { score: 1 } )
```

- Par défaut, il existe un index unique sur l'identifiant du document : _id

Les indexes

Les indexes simple



Les indexes

Les indexes composés

- Possibilité de combiné un index sur plusieurs champs d'un document

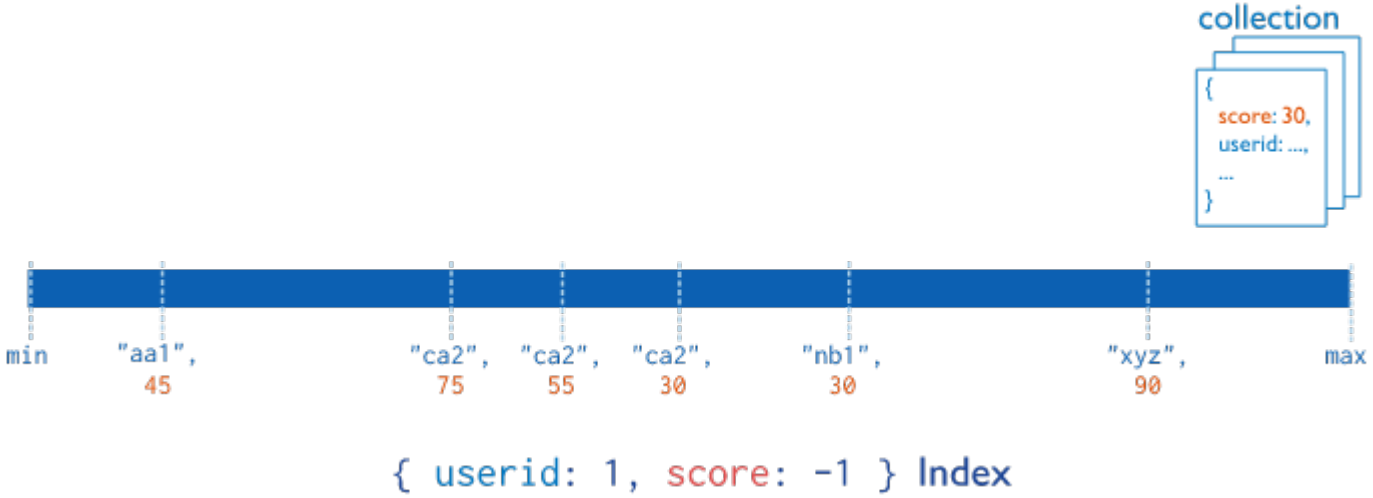
```
{
"_id": ObjectId("570c04a4ad233577f97dc459"),
"score": 35,
“userid”: “bb2”,
"location": { state: "NY", city: "New York" }
}
```

```
db.users.createIndex( { “userid": 1, “score": 1 } )
```

- Les documents seront d'abord indexés par le champ « item » puis par le champ « stock »

Les indexes

Les indexes composés




```
db.users.find( { userid: “bb2" } )
db.users.find( { userid: “bb2", score: { $gt: 55 } } )
```

Les index

Les propriétés des index

Type		
unique	Unicité des valeurs dans l'index	db.members.createIndex({ **"user_id"**: 1 }, { unique: true })
Partial index	L'index porte sur un sous ensemble	db.restaurants.createIndex({ cuisine: 1 }, { partialFilterExpression: { rating: { $gt: 5 } } })
TTL index	Un index sur un champs de type date pour spécifier l'expiration du document.	db.eventlog.createIndex({ "lastModifiedDate": 1 }, { expireAfterSeconds: 3600 })

Les index

Stratégie et performance

- Indexer les champs utilisés pour les recherches et pour les tris
- Lors des agrégations, uniquement le premier stage utilise les index.

- Pour étudier la performance, il existe la méthode **explain()**.
- Les mots clé importants :
  - COLLSCAN : Parcours de toutes la collection = Pas d'index
  - IXSCAN : Utilisation d'un index

```
db.orders.explain().aggregate( [
{ $match: { status: "urgent" } },
{ $group: { _id: "$productName", sumQuantity: { $sum: "$quantity" } } }
] )
```


```
{"stages" : [
    {
      "$cursor" : {
        "queryPlanner" : {
          "plannerVersion" : 1,
          "namespace" : "HEI.orders",
          "indexFilterSet" : false,
          "parsedQuery" : {
            "status" : {"$eq" : "urgent"}
          },
          "queryHash" : "52D5C93F",
          "planCacheKey" : "52D5C93F",
          "winningPlan" : {
            "stage" : "PROJECTION_SIMPLE",
            "transformBy" : {
              "productName" : 1,
              "quantity" : 1,
              "_id" : 0
            },
            "inputStage" : {
              "stage" : "COLLSCAN",
              "filter" : {
                "status" : { "$eq" : "urgent"}
              },
              "direction" : "forward"
            }
          },
          "rejectedPlans" : []
        }
      }
    },
    {
      "$group" : {
        "_id" : "$productName",
        "sumQuantity" : { "$sum" : "$quantity"}
      }
    }
]}
```

Les indexes

Les indexes de type texte

- Indexation de champs textuel : Titre, commentaire, résumé, ….
- Limitation : un seul index de type texte pour une collection
- Indexation de un, plusieurs ou tous les champs texte

```
db.reviews.createIndex( { comments: "text" } )
```

- Recherche de type texte:
  - Case insensitive
  - Charactère accentué
  - Tokenization
  - Prise en compte des langues
  - Stop Words
  - …

```
db.reviews.createIndex(
  {
    subject: "text",
    comments: "text"
  }
 )
```

- Notion de recherche Fulltext : Recherche Google

Les indexes

Les indexes de type géospatial

- Indexation des coordonnées spatiales (latitude, longitude, …)
- Utilisation des objets GeoJson (point, line, sphere, polygone, …)

```
db.collection.createIndex( { loc: "2dsphere" } )
```

```
db.places.insertMany( [
  {
    loc : { type: "Point", coordinates: [ -73.97, 40.77 ] },
    name: "Central Park",
    category : "Parks"
  },
  {
    loc : { type: "Point", coordinates: [ -73.88, 40.78 ] },
    name: "La Guardia Airport",
    category : "Airport"
  }
] )
```

Les Collections

Les collections

Les vues

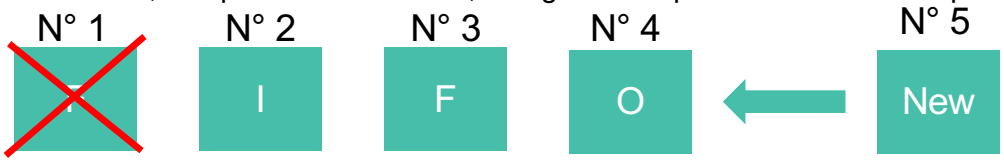
- Les vues permettent de présenter des données résultant d'une agrégation
- Les données ne sont pas stockées sur disque, l'agrégation est exécutée à la demande
- Uniquement des opérations de lecture, pas d'écriture
- La vue utilise les index de la collection source. Pas de possibilité d'en ajouter.

- Cas d'utilisation :
  - Présenter une collection sans les données personnelles
  - Calcule de somme ou de moyenne
  - Jointure entre deux collections (ex: Jointures entre la collection inventaire et la collection d'historique des commandes)

Les collections

Les Collections Capped

- Les collections « Capped » sont des collections avec une taille fixe (volume ou nombre d'enregistrements) qui conserve l'ordre d'insertion.
- Principe FIFO (First In First Out)
- Quand la taille allouée est atteinte, à la prochaine insertion, MongoDB remplace le document le plus ancien par le nouveau



- **Cas d'utilisation :**
  - Cache de données (ex : Information sur la session utilisateur)
  - Informations de journalisation/d'historique (ex : Toutes les personnes qui ont visité le site)

- **Attention aux restrictions :**
  - Ces documents ne sont pas optimisés pour être mis à jour -> Attention aux performances
  - Pas possible de supprimer un document manuellement

Les collections

Les Collections TimeSeries

- Se comporte comme une collection classique
- Efficaces pour stocker des mesures suivant une période de temps.
- Stockage optimisé pour ce type de données
- Ordonnées par défaut suivant le champs « timeField »
- Exemples de données :
  - Données météorologique : Mesure de la température
  - Données de courses : Mesure des points de passage d'un coureur

```
{
"metadata": {"sensorId": 5578, "type": "temperature"},
"timestamp": ISODate("2021-05-18T16:00:00.000Z"),
"temp": 12
}
```

```
db.createCollection(
   "weather24h",
   {
      timeseries: {
         timeField: "timestamp",
         metaField: "metadata",
         granularity: "hours"
      },
      expireAfterSeconds: 86400
   }
)
```

Réplication et le sharding

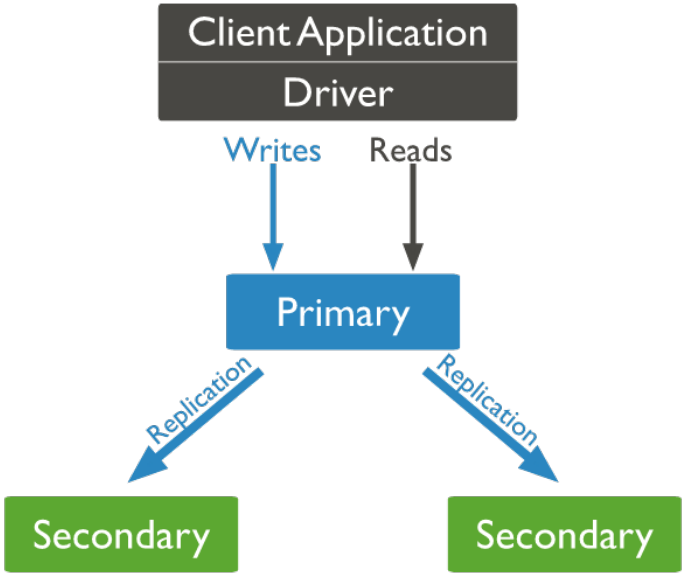
Réplication et Sharding

Les principes

- Il existe 3 modes de déploiement pour mongoDB:
  - Standalone : Correspond à une seul instance du serveur mongoDB. Un déploiement simple pour le développement par exemple.
  - Replicat Set : Introduit plusieurs nœuds pour répliquer la données. Toujours un nombre impaire.
  - Sharding : Plusieurs nœuds pour répliquer la donnée mais chaque nœud ne contient par l'ensemble des données. Distribution de la donnée.

Réplication et Sharding

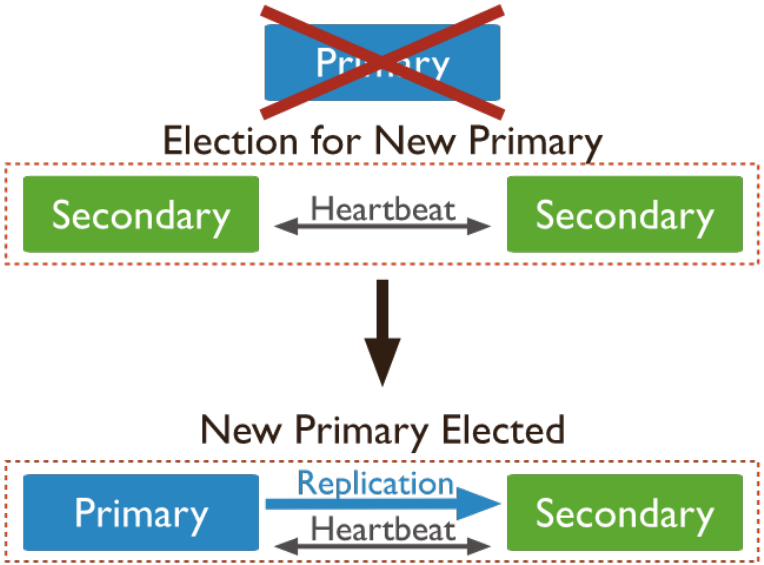
Le Replicaset



- 3 nœuds :
  - 1 primaire pour les opérations d'écriture et lecture
  - 2 secondaires pour une réplication asynchrone par défaut
- Possibilité de faire des lectures sur les secondaires
- Notion de WriteConcern : Le client décide du mode de synchronisation de la donnée

Réplication et Sharding

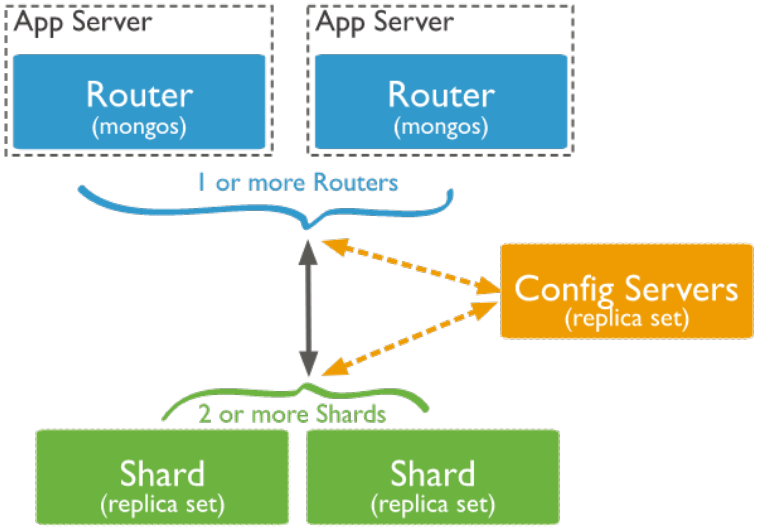
Le Replicaset



- La haute disponibilité
- Cas de perte d'un noeud
  - Une nouvelle élection a lieu pour élire un nouveau primaire
- Utilisé pour les opérations de maintenance également :
  - Changement de configuration
  - Changement de version

Réplication et Sharding

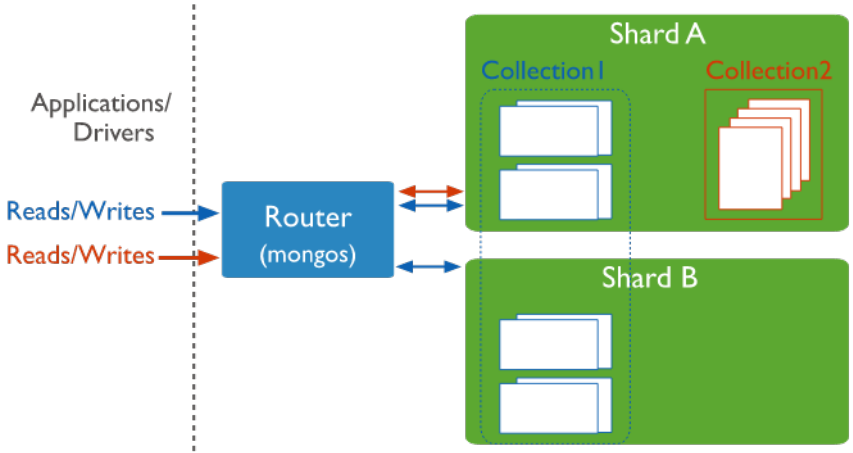
Le Sharding



- **Router** : Il est le router des requêtes entre le client et les shard. Ils ne contiennent pas de données mais savent où la trouver.
- **Config Server** : Contient les configurations des serveurs
- **Shard** : Chaque Shard contient un sous ensemble des documents. Un shard peut être déployé comme un replicat set

Réplication et Sharding

Le Sharding



- La collection 1 est répartie sur les deux shards.
- La collection 2 n'utilise pas de sharding

- Les documents sont réparties sur les shard suivant une clé de sharding

Bibliographie

Bibliographie

- https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql/4462471-maitrisez-le-theoreme-de-cap
- https://www.ambient-it.net/top-meilleures-db-nosql-2021/
- https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql/4462433-choisissez-votre-famille-nosql
- https://www.lebigdata.fr/time-series-database-definition
- https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql/4474601-decouvrez-le-fonctionnement-de-mongodb