MongoDB

Adrien Vossough

Le format JSON

Le format JSON

Le format JSON :

- Format de données
- Basé sur l'écriture des objets littéraux JavaScript
- Très lisible pour un format de données (peu verbeux)
- Souvent utilisé en Ajax en remplacement du format XML qui est beaucoup plus lourd

Types de données dans le format JSON :

- Les objets notés : **{ }**
- Les tableaux notés : **[ ]**
- Des valeurs génériques de type tableau, objet, booléen, nombre, chaîne ou null.

Il existe de nombreux validateurs (google : Validator json)

Le format JSON

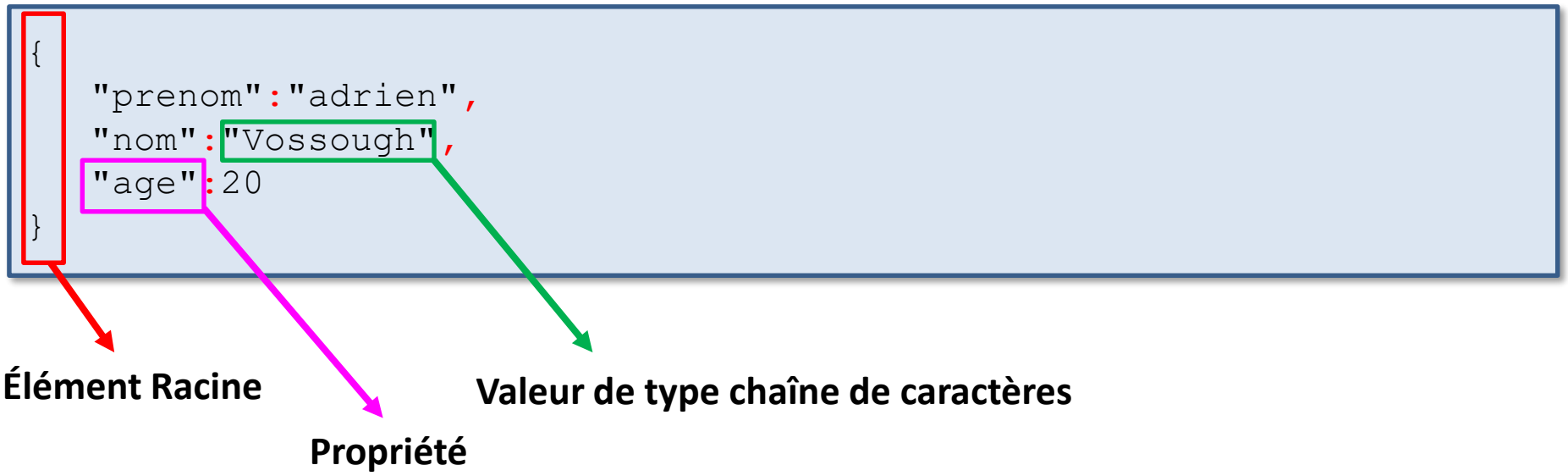
Les données au format JSON ont toujours pour racine un objet.

Il n'est pas possible de placer des commentaires directement en JSON.

Le nom d'une propriété est une chaîne de caractère, elle est séparée de sa valeur par ":"

La propriété/valeur est séparée de la suivante par une ",", la dernière valeur n'est pas suivie d'une virgule.

Exemple simple :



Le format JSON

Les types :

```
{
    "null": null,
    "booleen": true,
    "entier" : 20,
    "flottant": 12.5,
    "chaîne": "salut",
    "tableau1":["un", "deux", "trois"],
    "tableau2":[1, 2, 3],
    "tableau3":[1, "deux", 3.0],
    "objet1": { "animal": "chien", "age": 12, "sexe":"mâle"},
    "objet2": {
        "metier": "développeur",
        "tab_dans_objet": ["java", "javascript"]
    },
    "tableau_objets": [
        {"eleve": "Jean"},
        {"eleve":"Alexandre"},
        {"formateur": "Adrien"}
    ]
}
```

MongoDB

Adrien Vossough

3V : Volume, Velocity, Variety

NoSQL propose de retirer des contraintes du SQL : la structure, le langage d'interrogation ou la cohérence des données

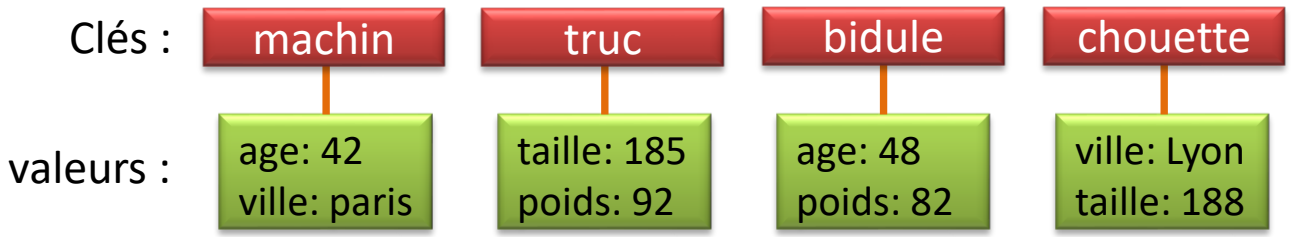
NoSQL propose une autre manière d'interroger les données et de les stocker

NoSQL est divisé en plusieurs familles de BDD :

NoSQL pour "Not only SQL" sont des SGBDs qui proposent de diminuer les contraintes du SQL pour favoriser la distribution des données.

Les clés-valeurs

Base de données sans schéma où des clés uniques identifient des valeurs de n'importe quel type.



Opérations CRUD possible :
Create (key,value)
Read (key)
Update (key,value)
Delete (key)

Base de données "clés-valeurs" : Redis, Memcached, Azure Cosmos DB, SimpleDB

En colonnes :

Les données sont souvent représentées en ligne pour sélectionner des entités

id	Age	Taille	poids
Truc	45	184	75
bidule	32	172	71
Machin	28	179	85

En colonnes, la base se focalise sur les attributs plutôt que les entités ceci pour s'orienter sur des calculs analytiques (statistiques, etc…)

id	Age
Truc	45
bidule	32
Machin	28

id	Taille
Truc	184
bidule	172
Machin	179

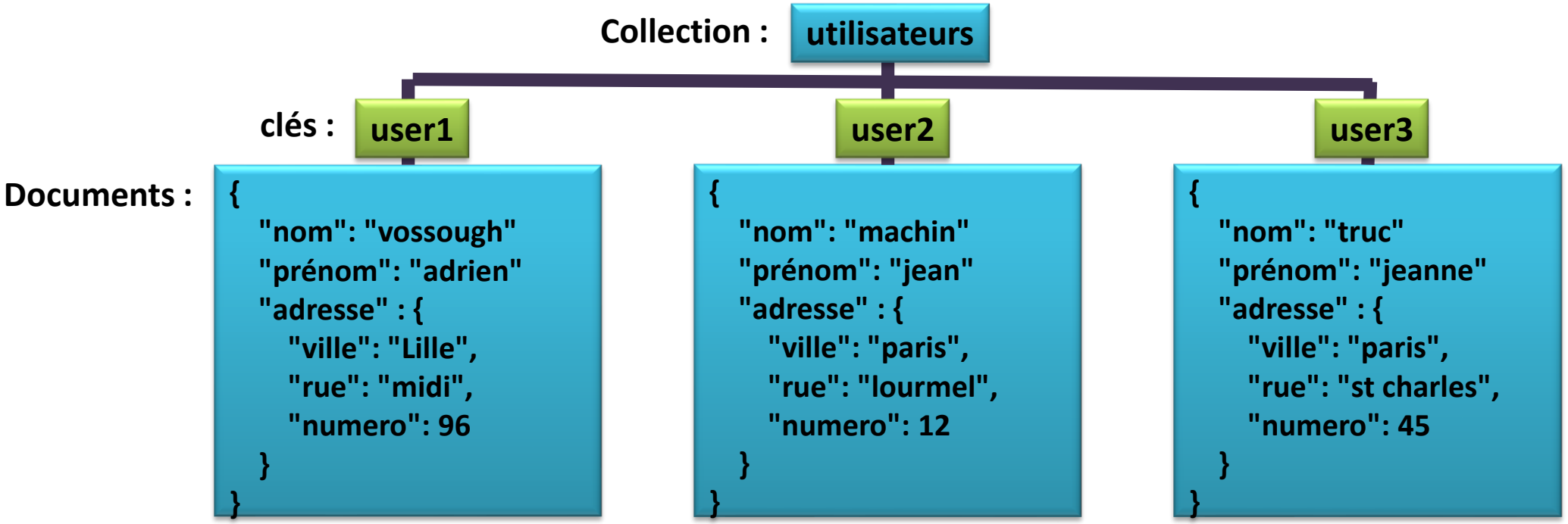
id	poids
Truc	75
bidule	71
Machin	85

Base de données "colonnes" : BigTable (Google), HBase (Apache, Hadoop), Spark SQL (Apache), Elasticsearch (elastic)

Les bases orientées documents rappelle une base de données classique pour des requêtes complexes.

Repose sur le principe du clé/valeur mais avec une extension sur les champs qui composent ce document.

Les documents peuvent être vu comme des fichiers textes contenant des informations rangés dans des répertoires appelés collections.



Base de données "documents" : MongoDB (MongoDB), CouchBase (Apache, Hadoop), DynamoDB (Amazon), Cassandra (Facebook -> Apache)

Les bases orientées graphes permettent des recherches de liens entre deux entités non reliées directement.

En SQL :

pneu		
id	**taille**	**id_vehicule**
45	45	14
46	32	14
47	28	15

vehicule		
id	**marque**	**id_personne**
14	Volvo	142
15	Fiat	143

personne	
id	**nom**
142	Machin
143	Truc
144	bidule

personne_entreprise	
id_personne	**id_entreprise**
142	35
142	36
144	36

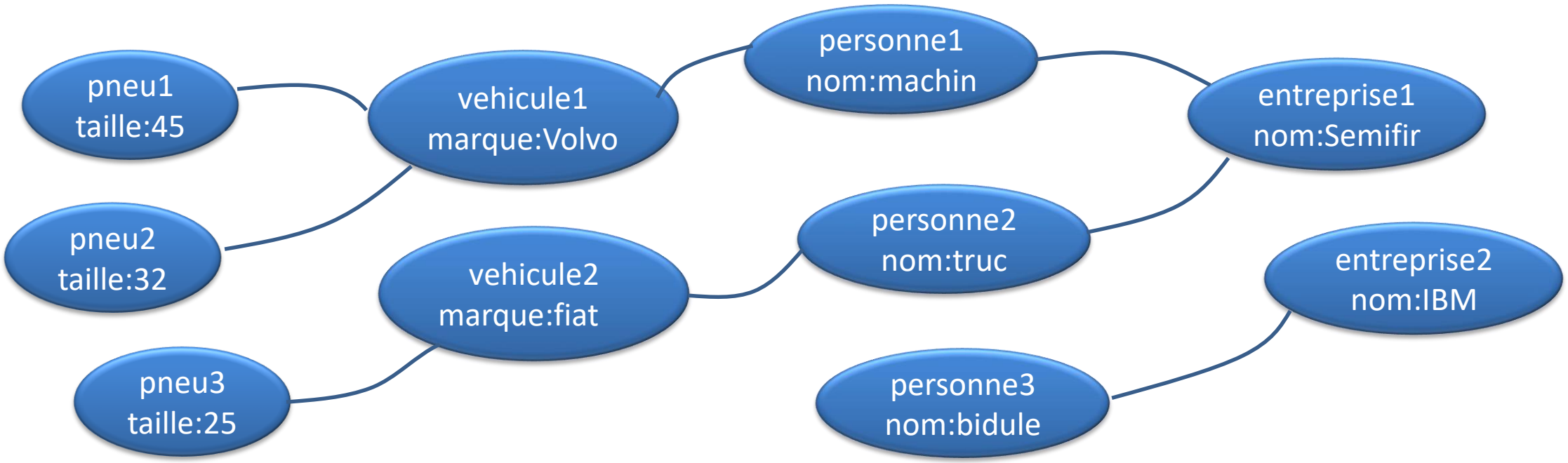
entreprise	
id	**Nom**
35	Semifir
36	IBM

Pour avoir le lien entre un pneu et une entreprise, nous devons passer par l'ensemble des tables ci-dessus.

```
SELECT pneu.id, entreprise.id, entreprise.nom FROM pneu
INNER JOIN vehicule ON pneu.id_vehicule = vehicule.id
INNER JOIN personne ON vehicule.id_personne = personne.id
INNER JOIN personne_entreprise ON personne.id = personne_entreprise.id_personne
INNER JOIN entreprise ON entreprise.id_entreprise = entreprise.id
LIMIT 1;
```

Le traitement par jointure est lourd et complexe car nous devons passer par chacun des tables pour passer d'un point A vers un point B.

Une base de donnée graphe utilise des nœuds (une entité) relié entre eux par des pointeurs physiques (lien) appelés arcs.



Il n'y a plus besoin de regarder dans les tables pour connaître les liens mais suivre les arcs.

Base de données "graphes" : Neo4j, OrientDB (Apache), FlockDB (Twitter)

SQL ou NoSQL ?

Le SQL:

- Permet des requêtes complexes avec jointures
- Permet de gérer parfaitement l'intégrité des données.
- Permet des transactions ACID

Le NoSQL

- Latences diminuées au maximum
- le moins de contrainte possible
- Suit généralement les propriétés **BASE**.
  - Basically Available : Garantie la disponibilité des données quelle que soit la charge
  - Soft-state : Autorise les incohérences temporaires (durant certaines modifications)
  - Eventually consistent : À terme, la base atteindra un état cohérent

Site pour télécharger MongoDB : https://www.mongodb.com/download-center?jmp=nav#community

Sous Mac : décompresser l'archive et déplacer le répertoire dans vos applications "/Applications"

Sous Debian/Ubuntu : > sudo apt install mongodb

Le serveur sera installé dans le répertoire : $MONGO.

Répertoires d'installation :

Windows : C:\Program Files\MongoDB\Server\3.x\

Linux : /opt/local/bin/

Mac : /Applications

Ajouter le répertoire C:\data\db

Lancer la commande **mongod** qui se trouve dans le sous-répertoire **/bin** de mongodb.

Il est possible ensuite d'utiliser **MongoDB Compass** pour se connecter au serveur.

Import d'une base de données depuis un fichier JSON

```
/bin/mongoimport --db nom_base_donnees--collection nom_collection fichier.json
```

L'application mongo dans le répertoire /bin/ de MongoDB permet de lancer un **shell** avec une api JavaScript pour se connecter au SGBD.

ATTENTION A LA CASSE ET AUX TYPES !
"10" n'est pas 10
Mettre les collections et les bases de données en minuscule

Les touches "flèche haut" et "flèche bas" permettent de revenir sur des anciennes commandes

La touche "tab" permet l'auto-complétion

ctrl+c ou la commande quit() ou exit permettent de quitter le shell

Commande du Shell Mongo

```
# commande pour afficher : équivalent à console.log() en JavaScript
print()
#affiche au format JSON
print(tojson(<obj>))
#affiche au format JSON
printjson()
```

Pour créer une base de données :

commande

```
# se connecte à une base de donnée existante ou non
db = connect("ma_base");

# affiche les bases de données
show dbs

#ma base ne s'affiche pas car ne sera réellement crée qu'avec au moins un document dans une collection

# création d'une collection
db.nomCollection

#affiche les collections de la base courante
show collections

# la collection ne s'affiche pas car vite, il faut la remplir d'un docume

# création d'un document
db.nomCollection.insert({ "nom": "adrien" })

# tester show collections et show dbs
```

.

quelques commandes

```
# affiche la liste des bases de données avec leur taille en mémoire
show dbs
# idem
show databases
# idem que le précédent mais au format JSON
db.adminCommand('listDatabases')

# utiliser la base "primer"
use primer

# afficher les collections
show collections

# renommer une collection
db.oldname.renameCollection('newname')

# effacer une collection
db.contacts.drop()

# effacer la base dans laquelle on est
db.dropDatabase()
```

Se placer dans une base de données

sélection des documents avec une requête

```
# retourne tous les documents d'une collection
db.ma_collection.find()

# retourne le premier document trouvé
db.ma_collection.findOne()

# retourne le premier document répondant à la requête
db.ma_collection.findOne( { "borough": "Queens" } )

# retourne tous les documents répondant à la requête
db.ma_collection.find( { "borough": "Queens" } )
db.ma_collection.find( {"borough": "Queens", "cuisine": "American" } )

# recherche d'une valeur dans un sous-document
db.ma_collection.find( {"address.street": "Broadway", "cuisine": "American" } )

# selection en utilisant une expression régulière
db.ma_collection.find({"borough": /Queens/i})

# retourne le nombre de documents trouvés
db.ma_collection.find({ "borough": "Queens" } ).count()
```

Exemple de document


```
{
  "_id": ObjectId("5a69158c831924b5a4115960"),
  "address": {
    "building": "45-15",
    "coord": [-73.91427200000001, 40.7569379],
    "street": "Broadway",
    "zipcode": "11103"
  },
  "borough": "Queens",
  "cuisine": "American",
  "grades": [{
    "date": ISODate("2013-12-04T00:00:00Z"),
    "grade": "A",
    "score": 9
   }, {
    "date": ISODate("2013-05-02T00:00:00Z"),
    "grade": "C",
    "score": 8
  }],
  "name": "Lavelle'S Admiral'S Club",
  "restaurant_id": "40365844"
}
```

Se placer dans une base de données

sélection des documents avec une requête

```
# tri des résultats par name par ordre croissant
db.ma_collection.find().sort({ "name":  1})

# tri des résultats par name par ordre décroissant
db.ma_collection.find().sort({ "name":  -1})

# tri des résultats par name puis par borough
db.ma_collection.find().sort({name : 1, borough : 1 });
```

Exemple de document


```
{
   "_id": ObjectId("5a69158c831924b5a4115960"),
   "address": {
      "building": "45-15",
      "coord": [-73.91427200000001, 40.7569379],
      "street": "Broadway",
      "zipcode": "11103"
   },
   "borough": "Queens",
   "cuisine": "American",
   "grades": [{
      "date": ISODate("2013-12-04T00:00:00Z"),
      "grade": "A",
      "score": 9
    }, {
     "date": ISODate("2013-05-02T00:00:00Z"),
      "grade": "C",
      "score": 8
   }],
   "name": "Lavelle'S Admiral'S Club",
   "restaurant_id": "40365844"
}
```

Se placer dans une base de données

sélection des documents

```
# operateur AND
db.ma_collection.find({ $and: [{"borough": "Queens"}, {"cuisine": "American" }] })
# identique au précédent
db.ma_collection.find({"borough": "Queens", "cuisine":"American"})

# operateur OR pour sélectionner correspondant à l'une des valeurs fournies
db.ma_collection.find({ $or: [{"borough": "Queens"}, {"cuisine": "American" }] })


# $gt (greater than) pour sélectionner les valeurs supérieurs à …
db.ma_collection.find( {"restaurant_id": {$gt : "40365844"} } )
```

Exemple de document


```
{
   "_id": ObjectId("5a69158c831924b5a4115960"),
   "address": {
      "building": "45-15",
      "coord": [-73.91427200000001, 40.7569379],
      "street": "Broadway",
      "zipcode": "11103"
   },
   "borough": "Queens",
   "cuisine": "American",
   "grades": [{
      "date": ISODate("2013-12-04T00:00:00Z"),
      "grade": "A",
      "score": 9
    }, {
      "date": ISODate("2013-05-02T00:00:00Z"),
      "grade": "C",
      "score": 8
   }],
   "name": "Lavelle'S Admiral'S Club",
   "restaurant_id": "40365844"
}
```

Opérateur	Signification
$gt	Supérieur à. Par exemple, "clé":{ $gt : 5 }
$lt	Inférieur à. Par exemple, "clé":{ $lt : 5 }
$in	Inclus dans le tableau de valeurs. Par exemple, "clé":{ $in : [1, 2, 3] }
$gte	Supérieur ou égal à. Par exemple, "clé":{ $gte : 5 }
$lte	Inférieur ou égal à. Par exemple, "clé":{ $lte : 5 }
$ne	Non égal à. Par exemple, "clé":{ $ne : 5 }
$nin	Non inclus dans le tableau de valeurs. Par exemple, "clé":{ $nin : [1, 2, 3] }
$not	Négation. Par exemple, "clé": {$not :{ $gt : 5 } }
$size	Taille de la liste. Par exemple, "clé_tableau": {$**size** :3}
$exists	Les documents ne respectent pas tous le même schéma et peuvent avoir des clés que d'autres n'ont pas. $exists teste la présence d'une clé. Par exemple, "clé" : {"$exists" : true}

Recherche par type de la valeur : { "clé" : { $type : 2 } }

type	valeur
Double 1	1
String	2
Object	3
Array	4
Binaire	5
Undefined	6 (déprécié)
Object id	7
Booleen	8
Date	9
Null	10
Timestamp	17

Recherche dans une liste de sous-document

sélection des documents

```
# recherche tous les documents dont un sous-document de la liste grades correspond
db.ma_collection.find({"grades.score": 9 })

# recherche tous les documents dont un sous-document de grades a pour valeur 9 et
# le même sous-document ou un autre a pour valeur C.
# Le document à droite en exemple est selectionné
db.ma_collection.find({"grades.score": 9, "grades.grade": "C"})


# recherche tous les documents dont un sous-document grades correspond
# Le document à droite en exemple N'est PAS selectionné
db.ma_collection.find({"grades":{ $elemMatch : {"score": 9, "grade":"C"}} })
```

Exemple de document

```
{
   "_id": ObjectId("5a69158c831924b5a4115960"),
   "address": {
      "building": "45-15",
      "coord": [-73.91427200000001, 40.7569379],
      "street": "Broadway",
      "zipcode": "11103"
   },
   "borough": "Queens",
   "cuisine": "American",
   "grades": [{
      "date": ISODate("2013-12-04T00:00:00Z"),
      "grade": "A",
      "score": 9
    }, {
     "date": ISODate("2013-05-02T00:00:00Z"),
      "grade": "C",
      "score": 8
   }],
   "name": "Lavelle'S Admiral'S Club",
   "restaurant_id": "40365844"
}
```

sélection des documents

```
# Distinct : retourne toutes les valeurs d'une clé
db.ma_collection.distinct("cuisine")
```

Exemple de document


```
{
   "_id": ObjectId("5a69158c831924b5a4115960"),
   "address": {
      "building": "45-15",
      "coord": [-73.91427200000001, 40.7569379],
      "street": "Broadway",
      "zipcode": "11103"
   },
   "borough": "Queens",
   "cuisine": "American",
   "grades": [{
      "date": ISODate("2013-12-04T00:00:00Z"),
      "grade": "A",
      "score": 9
    }, {
     "date": ISODate("2013-05-02T00:00:00Z"),
      "grade": "C",
      "score": 8
   }],
   "name": "Lavelle'S Admiral'S Club",
   "restaurant_id": "40365844"
}
```

La projection est le 2éme paramètre de find.
Il permet de sélectionner les champs à retourner

sélection des documents avec projection

```
# la valeur null permet de sélectionner tous les champs en fournissant une projection
# retourne les documents en affichant que le champ "name" et l'id
db.ma_collection.find(null, { name : 1 });

# permet de ne pas afficher l'id
db.ma_collection.find(null, { _id:0, name : 1 });
```

Exemple de document


```
{
   "_id": ObjectId("5a69158c831924b5a4115960"),
   "address": {
      "building": "45-15",
      "coord": [-73.91427200000001, 40.7569379],
      "street": "Broadway",
      "zipcode": "11103"
   },
   "borough": "Queens",
   "cuisine": "American",
   "grades": [{
      "date": ISODate("2013-12-04T00:00:00Z"),
      "grade": "A",
      "score": 9
    }, {
      "date": ISODate("2013-05-02T00:00:00Z"),
      "grade": "C",
      "score": 8
   }],
   "name": "Lavelle'S Admiral'S Club",
   "restaurant_id": "40365844"
}
```

L' agrégation permet de sélectionner des documents, les filtrer et les regrouper par attributs

db.ma_collection.**aggregate**(requete, projection, tri, groupement, detacher_liste)

sélection des documents avec projection

```
# $match  identique au filtre de find({ "borough" :"Queens"} )
db.ma_collection.aggregate( { $match : { "cuisine" :"American" } } )

 # $project  séléction des champs
db.ma_collection.aggregate(
    { $match : { "cuisine" :"American" } } ,
   { $project : {"cuisine":1, "borough":1, "_id":0} }
)

 # $sort  tri par les champs séléctionnés
db.ma_collection.aggregate(
   { $match : { "cuisine" :"American" } } ,
   { $project : {"cuisine":1, "borough":1, "_id":0} },
   { $sort : {"cuisine":1} }
)
```

Exemple de document


```
{
   "_id": ObjectId("5a69158c831924b5a4115960"),
   "address": {
      "building": "45-15",
      "coord": [-73.91427200000001, 40.7569379],
      "street": "Broadway",
      "zipcode": "11103"
   },
   "borough": "Queens",
   "cuisine": "American",
   "grades": [{
      "date": ISODate("2013-12-04T00:00:00Z"),
      "grade": "A",
      "score": 9
    }, {
     "date": ISODate("2013-05-02T00:00:00Z"),
      "grade": "C",
      "score": 8
   }],
   "name": "Lavelle'S Admiral'S Club",
   "restaurant_id": "40365844"
}
```

L' agrégation permet de sélectionner des documents, les filtrer et les regrouper par attributs

db.ma_collection.**aggregate**(requete, projection, tri, groupement, detacher_liste)

sélection des documents avec projection

```
# $group  : groupe les résultats avec une clé sélectionné
# le paramètre _id est obligatoire et un attribut total contenant une fonction
# d'aggrégation
db.ma_collection.aggregate(
   { $match : { "cuisine" :"American" } } ,
   { $project : {"cuisine":1, "borough":1, "_id":0} },
   { $sort : {"cuisine":1} },
   { $group : {"_id": "$borough", "total": { $sum: 1 } } }
)

# le résultat est le nombre de restaurant faisant de la cuisine américaine par quartier

# l'ordre des opérations est importants, pour trier le résultat :
db.ma_collection.aggregate(
   { $match : { "cuisine" :"American" } } ,
   { $project : {"cuisine":1, "borough":1, "_id":0} },
   { $group : {"_id": "$borough", "total": { $sum: 1 } } },
   { $sort : {"total":1} }
)
```

Exemple de document


```
{
   "_id": ObjectId("5a69158c831924b5a4115960"),
   "address": {
      "building": "45-15",
      "coord": [-73.91427200000001, 40.7569379],
      "street": "Broadway",
      "zipcode": "11103"
   },
   "borough": "Queens",
   "cuisine": "American",
   "grades": [{
      "date": ISODate("2013-12-04T00:00:00Z"),
      "grade": "A",
      "score": 9
    }, {
     "date": ISODate("2013-05-02T00:00:00Z"),
      "grade": "C",
      "score": 8
   }],
   "name": "Lavelle'S Admiral'S Club",
   "restaurant_id": "40365844"
}
```

Modification des documents

```
# Nous sélectionnons un document par son id et indiquons la nouvelle valeur du champs
db.ma_collection.update (
   {"_id" : ObjectId("5a69158c831924b5a41158f6")},
   {$set : {"borough" : "five"}}
);

# retourne si un document a été trouvé et modifié
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

# ajout d'une clé
db.ma_collection.update (
   {"_id" : ObjectId("5a69158c831924b5a41158f6")},
   {$set : {"ville" : "new-york"}}
);

# mise à jour de plusieurs document
db.ma_collection.update (
   {"borough": "Queens"},
   {$set : {"ville" : "new-york"}},
   {"multi" : true}
);
```

Exemple de document


```
{
   "_id": ObjectId("5a69158c831924b5a4115960"),
   "address": {
      "building": "45-15",
      "coord": [-73.91427200000001, 40.7569379],
      "street": "Broadway",
      "zipcode": "11103"
   },
   "borough": "Queens",
   "cuisine": "American",
   "grades": [{
      "date": ISODate("2013-12-04T00:00:00Z"),
      "grade": "A",
      "score": 9
    }, {
      "date": ISODate("2013-05-02T00:00:00Z"),
      "grade": "C",
      "score": 8
   }],
   "name": "Lavelle'S Admiral'S Club",
   "restaurant_id": "40365844"
}
```

Modification des documents

```
# Suppression d'un document
db.ma_collection. remove (
   {"_id" : ObjectId("5a69158c831924b5a41158f6")}
);

# Suppression de plusieurs document
db.ma_collection.update (
   {"borough": "Queens"}
   {"multi" : true}
);
```

Exemple de document


```
{
   "_id": ObjectId("5a69158c831924b5a4115960"),
   "address": {
      "building": "45-15",
      "coord": [-73.91427200000001, 40.7569379],
      "street": "Broadway",
      "zipcode": "11103"
   },
   "borough": "Queens",
   "cuisine": "American",
   "grades": [{
      "date": ISODate("2013-12-04T00:00:00Z"),
      "grade": "A",
      "score": 9
    }, {
      "date": ISODate("2013-05-02T00:00:00Z"),
      "grade": "C",
      "score": 8
   }],
   "name": "Lavelle'S Admiral'S Club",
   "restaurant_id": "40365844"
}
```