Requêtes Avancées

Importer le fichier grades.json comme suit :

Attention : Le code ci-dessous doit être exécuté à partir d'une nouvelle fenêtre DOS (pas à partir d'un client mongo)

```
>mongoimport -d students -c grades < grades.json
connected to: 127.0.0.1
2014-04-26T14:46:35.460+0200 check 9 800
2014-04-26T14:46:35.461+0200 imported 800 objects
```

Reprendre dans un client mongo

```
> use students
switched to db students

> show collections
grades
system.indexes
```

```
> db.grades.findOne()
{
	"_id" : ObjectId("50906d7fa3c412bb040eb577"),
	"student_id" : 0,
	"type" : "exam",
	"score" : 54.6535436362647
}
```

Find et Projection

$lt et $gt

Trouver l'ensemble des students ayant un score supérieur à 95.

```
> db.grades.find({score:{$gt:95}})
{ "_id" : ObjectId("50906d7fa3c412bb040eb57c"), "student_id" : 1, "type" : "quiz", "score" : 96.76851542258362 }
{ "_id" : ObjectId("50906d7fa3c412bb040eb582"), "student_id" : 2, "type" : "homework", "score" : 97.75889721343528 }
…
Type "it" for more
>
```

Trouver l'ensemble des students ayant un score supérieur à 95 et de type « quiz ».

```
> db.grades.find({score:{$gt:95},type:"quiz"})
```

{ "_id" : ObjectId("50906d7fa3c412bb040eb57c"), "student_id" : 1, "type" : "quiz", "score" : 96.76851542258362 }
{ "_id" : ObjectId("50906d7fa3c412bb040eb5ec"), "student_id" : 29, "type" : "quiz", "score" : 97.33967728060847 }
…

Trouver l'ensemble des students ayant un score compris entre 85 et 95 et de type « quiz ».

> **db.grades.find({score:{$gt:85,$lt:95},type:"quiz"})**
{ "_id" : ObjectId("50906d7fa3c412bb040eb5c8"), "student_id" : 20, "type" : "quiz", "score" : 92.76554684090782 }
{ "_id" : ObjectId("50906d7fa3c412bb040eb5d0"), "student_id" : 22, "type" : "quiz", "score" : 86.0800081592629 }
…

Inégalité sur un string

> **db.people.insert({name:"Bob"})**
WriteResult({ "nInserted" : 1 })

> **db.people.insert({name:"Charlie"})**
WriteResult({ "nInserted" : 1 })

> **db.people.insert({name:"Dave"})**
WriteResult({ "nInserted" : 1 })

> **db.people.insert({name:"Edgar"})**
WriteResult({ "nInserted" : 1 })

> **db.people.insert({name:"Fred"})**
WriteResult({ "nInserted" : 1 })

db.people.find()
"_id" : ObjectId("535bb427a8f64ced71fb8b74"), "name" : "Bob" }
"_id" : ObjectId("535bb442a8f64ced71fb8b75"), "name" : "Charlie" }
"_id" : ObjectId("535bb44ca8f64ced71fb8b76"), "name" : "Dave" }
"_id" : ObjectId("535bb456a8f64ced71fb8b77"), "name" : "Edgar" }
"_id" : ObjectId("535bb465a8f64ced71fb8b78"), "name" : "Fred" }

db.people.find({name:{$lt:"D"}})
"_id" : ObjectId("535bb427a8f64ced71fb8b74"), "name" : "Bob" }
"_id" : ObjectId("535bb442a8f64ced71fb8b75"), "name" : "Charlie" }

db.people.find({name:{$lt:"D",$gt:"C"}})
"_id" : ObjectId("535bb442a8f64ced71fb8b75"), "name" : "Charlie" }

Expression régulière et $exists et $type

> **db.people.insert({name:"Smith",age:30, profession:"hacker"})**
WriteResult({ "nInserted" : 1 })

> **db.people.insert({name:"Jones",age:35, profession:"boulanger"})**
WriteResult({ "nInserted" : 1 })

> **db.people.insert({name:42})**
WriteResult({ "nInserted" : 1 })

> **db.people.find()**
{ "_id" : ObjectId("535bb427a8f64ced71fb8b74"), "name" : "Bob" }

```
{ "_id" : ObjectId("535bb442a8f64ced71fb8b75"), "name" : "Charlie" }
{ "_id" : ObjectId("535bb44ca8f64ced71fb8b76"), "name" : "Dave" }
{ "_id" : ObjectId("535bb456a8f64ced71fb8b77"), "name" : "Edgar" }
{ "_id" : ObjectId("535bb465a8f64ced71fb8b78"), "name" : "Fred" }
{ "_id" : ObjectId("535bb84da8f64ced71fb8b79"), "name" : "Smith", "age" : 30, "profession" : "hacker" }
{ "_id" : ObjectId("535bb883a8f64ced71fb8b7a"), "name" : "Jones", "age" : 35, "profession" : "boulanger" }
{ "_id" : ObjectId("535bb8c7a8f64ced71fb8b7b"), "name" : 42 }
```

Tous documents ayant un champ profession.

```
> db.people.find({profession:{$exists:true}})
{ "_id" : ObjectId("535bb84da8f64ced71fb8b79"), "name" : "Smith", "age" : 30, "profession" : "hacker" }
{ "_id" : ObjectId("535bb883a8f64ced71fb8b7a"), "name" : "Jones", "age" : 35, "profession" : "boulanger" }
```

Tous documents n'ayant pas un champ profession.

```
> db.people.find({profession:{$exists:false}})
{ "_id" : ObjectId("535bb427a8f64ced71fb8b74"), "name" : "Bob" }
{ "_id" : ObjectId("535bb442a8f64ced71fb8b75"), "name" : "Charlie" }
{ "_id" : ObjectId("535bb44ca8f64ced71fb8b76"), "name" : "Dave" }
{ "_id" : ObjectId("535bb456a8f64ced71fb8b77"), "name" : "Edgar" }
{ "_id" : ObjectId("535bb465a8f64ced71fb8b78"), "name" : "Fred" }
{ "_id" : ObjectId("535bb8c7a8f64ced71fb8b7b"), "name" : 42 }
```

Tous documents ayant un champ « name » de type String

Reference BSON sur les Type : http://docs.mongodb.org/manual/reference/bson-types/

Tous documents ayant un champ « name » de type String.

```
> db.people.find({name:{$type:2}})
{ "_id" : ObjectId("535bb427a8f64ced71fb8b74"), "name" : "Bob" }
{ "_id" : ObjectId("535bb442a8f64ced71fb8b75"), "name" : "Charlie" }
{ "_id" : ObjectId("535bb44ca8f64ced71fb8b76"), "name" : "Dave" }
{ "_id" : ObjectId("535bb456a8f64ced71fb8b77"), "name" : "Edgar" }
{ "_id" : ObjectId("535bb465a8f64ced71fb8b78"), "name" : "Fred" }
{ "_id" : ObjectId("535bb84da8f64ced71fb8b79"), "name" : "Smith", "age" : 30, "profession" : "hacker" }
{ "_id" : ObjectId("535bb883a8f64ced71fb8b7a"), "name" : "Jones", "age" : 35, "profession" : "boulanger" }
```

Tous documents ayant la valeur du champ « name » contenant la lettre « a ».

```
> db.people.find({name:{$regex:"a"}})
{ "_id" : ObjectId("535bb442a8f64ced71fb8b75"), "name" : "Charlie" }
{ "_id" : ObjectId("535bb44ca8f64ced71fb8b76"), "name" : "Dave" }
{ "_id" : ObjectId("535bb456a8f64ced71fb8b77"), "name" : "Edgar" }
```

Tous documents ayant la valeur du champ « name » terminant par la lettre « e ».

```
> db.people.find({name:{$regex:"e$"}})
{ "_id" : ObjectId("535bb442a8f64ced71fb8b75"), "name" : "Charlie" }
{ "_id" : ObjectId("535bb44ca8f64ced71fb8b76"), "name" : "Dave" }
```

Tous documents ayant la valeur du champ « name » commencant par la lettre « C ».

```
> db.people.find({name:{$regex:"^C"}})
{ "_id" : ObjectId("535bb442a8f64ced71fb8b75"), "name" : "Charlie" }
```

$or

Tous documents ayant la valeur du champ « name » finissant par la lettre « e » ou ayant le champ « age ».

```
> db.people.find({$or:[{name:{$regex:"e$"}},{age:{$exists:true}}]})
{ "_id" : ObjectId("535bb442a8f64ced71fb8b75"), "name" : "Charlie" }
{ "_id" : ObjectId("535bb44ca8f64ced71fb8b76"), "name" : "Dave" }
{ "_id" : ObjectId("535bb84da8f64ced71fb8b79"), "name" : "Smith", "age" : 30, "profession" : "hacker" }
{ "_id" : ObjectId("535bb883a8f64ced71fb8b7a"), "name" : "Jones", "age" : 35, "profession" : "boulanger" }
```

$and

Tous documents ayant la valeur du champ « name » supérieur à la lettre « C » et contenant la lettre « a ».

```
> db.people.find({$and:[{name:{$gt:"C"}},{name:{$regex:"a"}}]})
{ "_id" : ObjectId("535bb442a8f64ced71fb8b75"), "name" : "Charlie" }
{ "_id" : ObjectId("535bb44ca8f64ced71fb8b76"), "name" : "Dave" }
{ "_id" : ObjectId("535bb456a8f64ced71fb8b77"), "name" : "Edgar" }
```

Requête sur une liste d'éléments

```
> db.accounts.insert({name:"George", favorites:["ice cream", "pretzels"]})
WriteResult({ "nInserted" : 1 })

> db.accounts.insert({name:"Howard", favorites:["pretzels","beer"]})
WriteResult({ "nInserted" : 1 })

> db.accounts.find().pretty()
{
	"_id" : ObjectId("535bd635a8f64ced71fb8b7c"),
	"name" : "George",
	"favorites" : [
		"ice cream",
		"pretzels"
	]
}
{
	"_id" : ObjectId("535bd667a8f64ced71fb8b7d"),
	"name" : "Howard",
	"favorites" : [
		"pretzels",
		"beer"
	]
}
```

Tous documents ayant « favorites » contenant la valeur « beer ».

```
> db.accounts.find({favorites:"beer"})
{ "_id" : ObjectId("535bd667a8f64ced71fb8b7d"), "name" : "Howard", "favorites" : [ "pretzels", "beer" ] }
```

Tous documents ayant « favorites » contenant la valeur « beer » et name supérieur è « H ».

```
> db.accounts.find({favorites:"beer",name:{$gt:"H"}})
{ "_id" : ObjectId("535bd667a8f64ced71fb8b7d"), "name" : "Howard", "favorites" : [ "pretzels", "beer" ] }
```

$in et $all

```
> db.accounts.insert({name:"Irving", favorites:["pretzels","beer","cheese"]})
WriteResult({ "nInserted" : 1 })

> db.accounts.insert({name:"John", favorites:["beer","cheese"]})
WriteResult({ "nInserted" : 1 })

> db.accounts.find()
{ "_id" : ObjectId("535bd635a8f64ced71fb8b7c"), "name" : "George", "favorites" : [ "ice cream", "pretzels" ] }
{ "_id" : ObjectId("535bd667a8f64ced71fb8b7d"), "name" : "Howard", "favorites" : [ "pretzels", "beer" ] }
{ "_id" : ObjectId("535bdd60a8f64ced71fb8b7e"), "name" : "Irving", "favorites" : [ "pretzels", "beer", "cheese" ] }
{ "_id" : ObjectId("535bdd85a8f64ced71fb8b7f"), "name" : "John", "favorites" : [ "beer", "cheese" ]}
```

Tous documents ayant « favorites » contenant la valeur « beer » et « pretzels ».

```
> db.accounts.find({favorites : { $all : ["pretzels","beer"]} })
{ "_id" : ObjectId("535bd667a8f64ced71fb8b7d"), "name" : "Howard", "favorites" : [ "pretzels", "beer" ] }
{ "_id" : ObjectId("535bdd60a8f64ced71fb8b7e"), "name" : "Irving", "favorites" : [ "pretzels", "beer", "cheese" ] }
```

Tous documents ayant « favorites » contenant la valeur « beer » ou « pretzels ».

```
> db.accounts.find({favorites : { $in : ["pretzels","beer"]} })
{ "_id" : ObjectId("535bd635a8f64ced71fb8b7c"), "name" : "George", "favorites" : [ "ice cream", "pretzels" ] }
{ "_id" : ObjectId("535bd667a8f64ced71fb8b7d"), "name" : "Howard", "favorites" : [ "pretzels", "beer" ] }
{ "_id" : ObjectId("535bdd60a8f64ced71fb8b7e"), "name" : "Irving", "favorites" : [ "pretzels", "beer", "cheese" ] }
{ "_id" : ObjectId("535bdd85a8f64ced71fb8b7f"), "name" : "John", "favorites" : [ "beer", "cheese" ]
```

Requête avec la notation «.»

```
> db.users.insert({name:"richard", email: {work: "richard@yahoo.fr", personnal : "richard@gmail.com"}})
WriteResult({ "nInserted" : 1 })

> db.users.findOne()
{
	"_id" : ObjectId("535be5b1a8f64ced71fb8b81"),
	"name" : "richard",
	"email" : {
		"work" : "richard@yahoo.fr",
		"personnal" : "richard@gmail.com"
	}
}

> db.users.find({ email: {work: "richard@yahoo.fr", personnal : "richard@gmail.com"}})
```

{ "_id" : ObjectId("535be5b1a8f64ced71fb8b81"), "name" : "richard", "email" : { "work" : "richard@yahoo.fr", "personnal" : "richard@gmail.com" } }

> **db.users.find({ email: {personnal : "richard@gmail.com",work: "richard@yahoo.fr"}})**

Rien car ordre non préserver.

> **db.users.find({ email: {work: "richard@yahoo.fr"}})**

Rien car aucun document ne correspond strictement à notre critére.

> **db.users.find({"email.work" : "richard@yahoo.fr"})**

```
{ "_id" : 1, "name" : "richard", "email" : { "work" : "richard@yahoo.fr",
                                             "personnal" : "richard@gmail.com" } }
```

Compter les éléments

> **db.grades.count({type:"exam"})**

200

$unset

Remove a field from one document.

> **db.people.find()**

```
{ "_id" : ObjectId("535bb427a8f64ced71fb8b74"), "name" : "Bob" }
{ "_id" : ObjectId("535bb442a8f64ced71fb8b75"), "name" : "Charlie" }
{ "_id" : ObjectId("535bb44ca8f64ced71fb8b76"), "name" : "Dave" }
{ "_id" : ObjectId("535bb456a8f64ced71fb8b77"), "name" : "Edgar" }
{ "_id" : ObjectId("535bb465a8f64ced71fb8b78"), "name" : "Fred" }
{ "_id" : ObjectId("535bb84da8f64ced71fb8b79"), "name" : "Smith", "age" : 30, "profession" : "hacker" }
{ "_id" : ObjectId("535bb883a8f64ced71fb8b7a"), "name" : "Jones", "age" : 35, "profession" : "boulanger" }
{ "_id" : ObjectId("535bb8c7a8f64ced71fb8b7b"), "name" : 42 }
```

> **db.people.update({name: "Jones"},{$unset : {profession : 1}})**

```
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

> **db.people.find()**

```
{ "_id" : ObjectId("535bb427a8f64ced71fb8b74"), "name" : "Bob" }
{ "_id" : ObjectId("535bb442a8f64ced71fb8b75"), "name" : "Charlie" }
{ "_id" : ObjectId("535bb44ca8f64ced71fb8b76"), "name" : "Dave" }
{ "_id" : ObjectId("535bb456a8f64ced71fb8b77"), "name" : "Edgar" }
{ "_id" : ObjectId("535bb465a8f64ced71fb8b78"), "name" : "Fred" }
{ "_id" : ObjectId("535bb84da8f64ced71fb8b79"), "name" : "Smith", "age" : 30, "profession" : "hacker" }
{ "_id" : ObjectId("535bb883a8f64ced71fb8b7a"), "name" : "Jones", "age" : 35 }
{ "_id" : ObjectId("535bb8c7a8f64ced71fb8b7b"), "name" : 42 }
```

$push, $pop, $pull, $pushAll, $pullAll, $addToSet

Concerne la manipulation de tableau dans un document.

Changer un élément d'un tableau.

```
> db.arrays.insert({_id:0 , a : [1,2,3,4 ] })
WriteResult({ "nInserted" : 1 })

> db.arrays.findOne()
{ "_id" : 0, "a" : [ 1, 2, 3, 4 ] }

> db.arrays.update({_id:0 },{$set:{"a.2":5}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.arrays.findOne()
{ "_id" : 0, "a" : [ 1, 2, 5, 4 ] }
```

Ajouter un élément a un tableau.

```
> db.arrays.update({_id:0 },{$push : {a:6}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.arrays.findOne()
{ "_id" : 0, "a" : [ 1, 2, 5, 4, 6 ] }
```

Supprimer le dernier élément d'un tableau.

```
> db.arrays.update({_id:0 },{$pop : {a:1}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.arrays.findOne()
{ "_id" : 0, "a" : [ 1, 2, 5, 4 ] }
```

Supprimer le premier élément d'un tableau.

```
> db.arrays.update({_id:0 },{$pop : {a:-1}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.arrays.findOne()
{ "_id" : 0, "a" : [ 2, 5, 4 ] }
```

Ajouter plusieurs éléments

```
> db.arrays.update({_id:0 },{$push: { a:{$each : [7, 8, 9]} }})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.arrays.findOne()
{ "_id" : 0, "a" : [ 2, 5, 4, 7, 8, 9 ] }
```

Supprimer un élément d'un tableau

```
> db.arrays.update({_id:0 },{$pull : { a: 5 }})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

```
> db.arrays.findOne()
{ "_id" : 0, "a" : [ 2, 4, 7, 8, 9 ] }
```

Enlever plusieurs éléments d'un coup.

```
> db.arrays.update({_id:0 },{$pullAll : { a: [2,4,8] }})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.arrays.findOne()
{ "_id" : 0, "a" : [ 7, 9 ] }
```

Ajouter un élément que si absent dans le tableau.

```
> db.arrays.update({_id:0 },{$addToSet : { a: 5 }})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.arrays.findOne()
{ "_id" : 0, "a" : [ 7, 9, 5 ] }

> db.arrays.update({_id:0 },{$addToSet : { a: 5 }})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })

> db.arrays.findOne()
{ "_id" : 0, "a" : [ 7, 9, 5 ] }
```