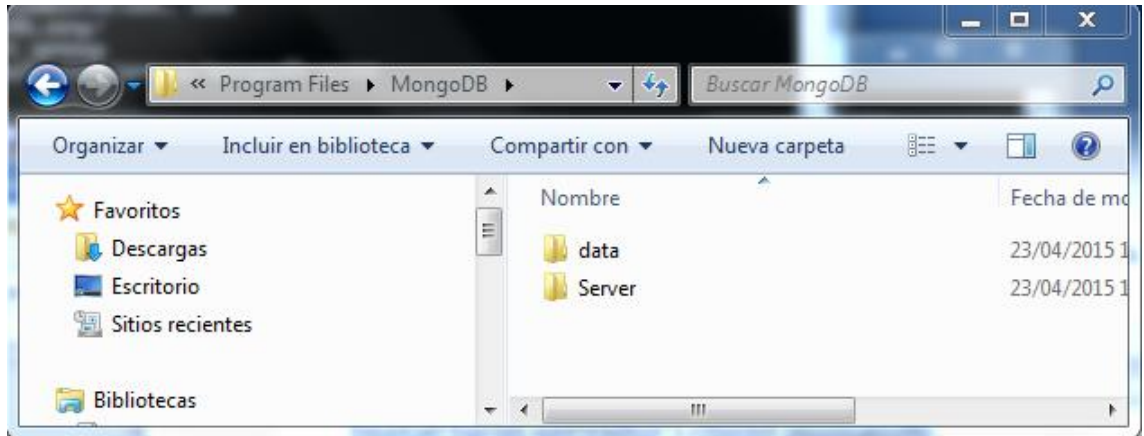


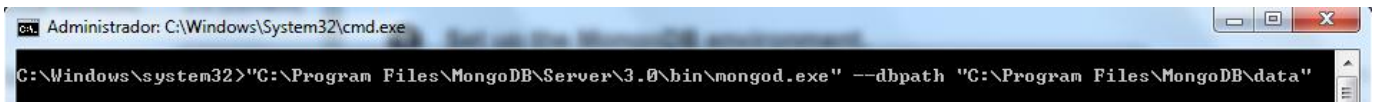
## Instal·lació servidor i client mongodb

1. Descarregar l'arxiu [mongodb-win32-x86\\_64-2008plus-ssl-3.0.2-signed](#) des d'aquest enllaç.
2. Executa la instal·lació completa del programa descarregat al punt 1. Si tot a funcionat de forma correcta haurà aparegut la carpeta "[c:\Program files\MongoDB](#)".
3. Dintre de la ubicació anterior heu de crear una carpeta "data" que serà el lloc on es guardaran les dades.

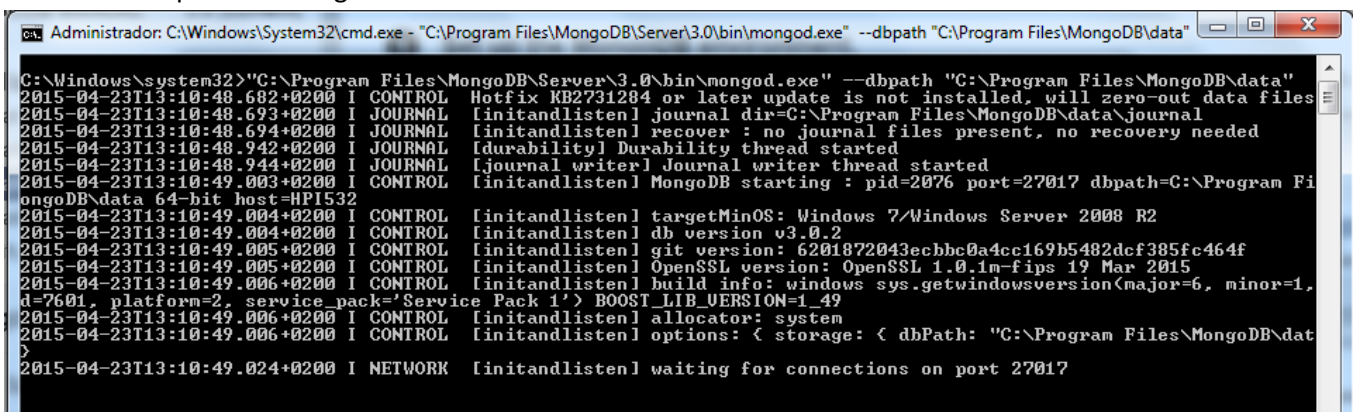


## Posada en marxa del servidor mongoDB

4. Obriu un terminal com a administradors i escriviu la següent comanda que ens servirà per posar en marxa el servidor mongoDB utilitzant la ubicació creada al punt anterior com a ubicació per les dades.



Veureu que el servidor es posarà en marxa. No podeu continuar fins que no surti per pantalla lo següent:



En aquest moment, el servidor mongoDB està posat en marxa i espera connexions.

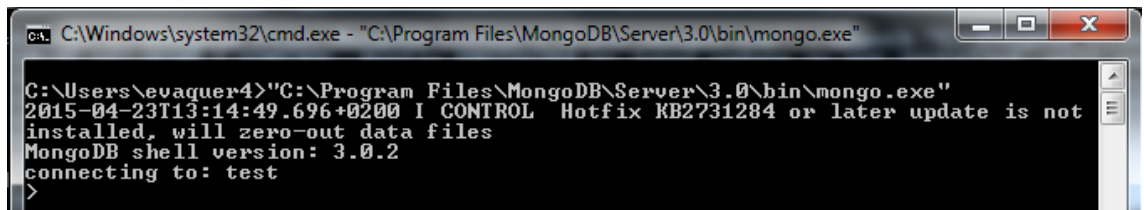
## Posada en marxa de la shell mongoDB

- Obrim un altre cmd (en aquest cas no es necessari que s'obri com a administrador) i escriuiu la següent comanda:



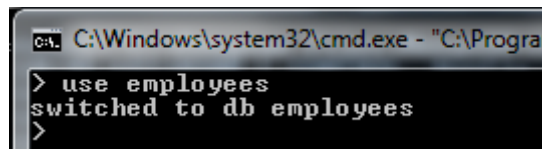
```
C:\Windows\system32\cmd.exe
C:\Users\evaquer4>"C:\Program Files\MongoDB\Server\3.0\bin\mongo.exe"
```

Si tot ha funcionat de forma correcta us ha de sortir per pantalla la següent informació:



```
C:\Windows\system32\cmd.exe - "C:\Program Files\MongoDB\Server\3.0\bin\mongo.exe"
C:\Users\evaquer4>"C:\Program Files\MongoDB\Server\3.0\bin\mongo.exe"
2015-04-23T13:14:49.696+0200 I CONTROL Hotfix KB2731284 or later update is not
installed, will zero-out data files
MongoDB shell version: 3.0.2
connecting to: test
>
```

En aquest punt, ja estem connectats a mongoDB, i estem utilitzant l'esquema test. Per canviar a un altre esquema podeu utilitzar la comanda `use <nom_esquema>`.



```
C:\Windows\system32\cmd.exe - "C:\Program
> use employees
switched to db employees
>
```

## Importació de dades

- Descarregueu-vos el [students.csv](#) de marte.
- Executeu la següent comanda substituint el path pel lloc de teu sistema de fitxers on hagi guardat el fitxer:

```
C:\Program Files\MongoDB\Server\3.0\bin\mongoimport.exe" --db
students --type csv --headerline --file <path>\students.csv
```

## Introduction

### Nomenclatura

Model relacional	MongoDB
Database	Database
Collection	table
Document o BSON document	Register
Field	columna
index	index
table joins	embedded documents and linking
primary key	primary key
Specify any unique column or column combination as primary key	the primary key is automatically set to the _id field
aggregation	aggregation pipeline

## Com inserir informació a una BBDD MongoDB

La comanda que utilitzarem per inserir informació en la BBDD és insert. A continuació teniu un exemple d'ús:

```
use test
db.people.insert( { "name" : "Smith", "age": 30 } ) ;
```

Per comprovar que la inserció s'ha realitzat correctament escriviu:

```
db.people.find()
```

Qualsevol comanda MongoDB és pot utilitzar dintre d'un bloc javascript. Teclegeu l'exemple que teniu a continuació i comproveu després quants registres s'han inserit a la BBDD:

```
use test
for(i=0; i<1000; i++) {
db.sample.insert( { "i" : i, "x" : Math.random() } );
}
db.sample.count() ;
```

La comanda db.sample.count us mostrarà el número total de documents inserits.

## Com llegir informació a una BBDD MongoDB

Per llegir informació de una BBDD utilitzarem la comanda find(). A continuació teniu un exemple d'ús:

```
use test
db.people.find( { "name" : "Smith" } ) ;
```

La anterior comanda retorna tots els documents de la col·lecció people que tinguin valor "Smith" al camp "name".

Les consultes poden ser més complicades si afegim operadors especials:

```
use test
db.people.find( { "age" : { "$gte" : 20 } } ) ;
db.people.find( { "name" : "Smith" , "age" : { "$gte" : 20 } } ) ;
```

La primera ens retornarà els documents de la col·lecció people els quals tinguin un valor major o igual a 20 al camp "age".

A la segona, ens tindrà en compte la condició anterior i a més a més el valor del camp "name" ha de ser igual a "Smith".

A continuació tens una llista de tots els operadors especials que pots utilitzar:

```
$gt, $gte, $lt, $lte, $ne, $in, $nin, $mod, $regex/$options,  
$all, $size, $exists, $type, $not, $or, $nor, $elemMatch,  
and $where (try not to use $where !)
```

### Consultes a vectors

Les consultes que examinen vectors comparan tots els elements del vector amb l'argument que coincideixi amb el predicat. Per exemple:

```
db.collection.find( { "x" : 2 } ) ;  
  
and  
  
db.collection.find( { "x" : { "$gt" : 4 } } ) ;
```

retornarà els documents on x es el vector [1,3,5] (en el primer cas , no trobarà cap document i al segon cas mostrarà el 5).

### Com podem triar quins camps mostrarem?

Podem passar un segon argument a comanda find() per limitar els camps que es visualitzaran com a resultat de la consulta. Per exemple:

```
db.scores.find( {} , { "score" : 1 } ) ;  
db.scores.find( {} , { "score" : 0 } ) ;
```

En el primer cas només inclourem els camps score i \_id i en el segon inclourem tots els camps excepte score.

### Cursors

Al fer una consulta a MongoDB inicialitzem el cursor. Els cursors són iterables, i accedeixen als següents resultats de forma transparent. Per exemple:

```
db.students.find()  
  
it  
  
it  
  
it...
```

Si executeu la anterior consulta podeu comprovar que només es mostren els 20 primers resultats. Podeu mostrar els següents resultats de forma recurrent utilitzant la comanda it

Una altra forma de moure's per un cursor és amb les comandes next(), hasNext(), forEach():

```
var cursor = db.students.find() ;  
while ( cursor.hasNext() ) printjson( cursor.next() ) ;  
db.students.find().forEach( function(x){ printjson( x ) } ) ;  
db.students.find().forEach( printjson ) ;  
cursor.forEach // without () prints source
```

Podeu ordenar els resultats de la consulta utilitzant els modificadors `sort()`, `skip()` i `limit()`. Per exemple:

```
db.students.find().sort( { "birth_date" : -1 } ).limit( 10 ) ;
```

La comanda anterior retornarà els 10 primers documents de la col·lecció `students` ordenats descendentment pel camp `birth_date`.

## Actualitzar informació a una BBDD MongoDB

La comanda que utilitzarem per actualitzar dades a una BBDD MongoDB és `Update()`. A continuació teniu un exemple d'ús:

```
use test  
db.stuff.insert( { "_id" : 123, "foo" : "bar" } ) ;  
db.stuff.find( { "_id": 123 } ) ;
```

A l'exemple anterior hem creat una col·lecció dintre la BBDD `test` que es diu `stuff`.

Quan realitzem l'insert inicial, es crea la BBDD i s'afegeix un document de tipus `stuff` amb només 2 camps: `_id` i `foo`. `_id` és sempre la clau primària de les col·leccions.

Amb la comanda `find()` comprovem que el document s'ha creat de forma correcta. Realitzem les actualitzacions:

```
db.stuff.update( { "_id" : 123 } , { "hello" : "world" } ) ;  
db.stuff.update( { "_id" : 123 } , { "$set" : { "hello" : "world" } } ) ;  
db.people.update( { "name" : "Smith" } , { "$set" : { "children" : 2.5 } } ) ;  
db.people.update( { "name" : "Smith" } ,  
{ "$push" : { "interests" : "chess" } } ) ;
```

En la primera línia estic afegint al document identificat com "123" el camp "hello" amb el valor "world".

En la segona línia l'operador `$set` reemplaça el valor del camp "hello" per "world".

A la tercera línia estic modificant el camp "children" del document amb "name" "Smith" a 2.5

A la quarta línia estic afegint al vector "interest" al valor `chess` dintre del document amb "name" "Smith".

Els operadors de la comanda `Update` són els següents:

```
$set, $unset, $inc, $push, $pushAll, $pull, $pullAll, $pop,  
$addToSet, $rename, $bit, $ positional operator
```

## Upserts i multi-update

El concepte upsert fa referència al fet que la comanda Update() en cas que l'element no existeix, pot inserir-lo (hem d'afegir el paràmetre true al final de la comanda).

A més a més de fer actualitzacions simples MongoDB pot realitzar actualitzacions múltiples. A continuació teniu un parell d'exemples:

```
db.people.update( { "name" : "Jones" } , { "$set" : { "age" : 50 } } ,  
{ "upsert" : true } ) ;  
  
db.people.update( { } , { "$set" : { "city" : "NYC" } } ,  
{ "multi" : true } ) ;
```

## Esborrar informació amb MongoDB

Per esborrar documents d'una col·lecció utilitzarem la comanda remove(), que treballa exactament igual que la comanda find().

```
db.collection.remove( { "x" : 2 } ) ;
```

La anterior comanda esborrarà tots els documents que tinguin valor 2 al camp "x".

## Esborrar Col·leccions de MongoDB

Podem esborrar tots els documents d'una col·lecció utilitzant la comanda drop(). Per exemple:

```
db.foo.drop() ;
```

La anterior comanda esborrarà tots els documents de la col·lecció foo.

El resultat és el mateix que esborrar tots els documents amb la comanda remove, però la operació es realitza de forma molt més eficient.

## Esborrar Database de MongoDB

Per esborrar tota una BBDD de MongoDB utilitzarem la comanda db.dropDatabase(). Per exemple:

```
Use employees;  
db.dropDatabase() ;
```

Aquesta comanda esborra la base de dades actual, en aquest cas "employees".

## Expressions regulars

Les expressions regulars són utilitzades freqüentment a tots els llenguatges de programació per buscar patrons o paraules a una cadena de caràcters. MongoDB també proveeix aquesta funcionalitat utilitzant la comanda \$regex. MongoDB utilitza PCRE (Perl Compatible Regular Expression) com a llenguatge per a expressions regulars.

A diferència de la recerca de text, no necessitem fer cap configuració per utilitzar les expressions regulars.

Considereu la següent estructura de document a la col·lecció post:

```
{
  "post_text": "enjoy the mongodb articles on tutorialspoint",
  "tags": [
    "mongodb",
    "tutorialspoint"
  ]
}
```

### Utilitzant expressions regex

- Buscar un text dintre d'un camp:

```
db.posts.find({post_text:{$regex:"tutorialspoint"}})
o
db.posts.find({post_text:/tutorialspoint/})
```

- Buscar un text dintre d'un camp (case insensitive):

```
db.posts.find({post_text:{$regex:"tutorialspoint",$options:"$i"}})
```

- Buscar un text dintre d'un vector:

```
db.posts.find({tags:{$regex:"tutorial"}})
```

- Buscar un text al principi d'un camp (a l'exemple busquem que comenci per a):

```
db.posts.find({"post_text ":/^a/})
```

- Buscar un text al final d'un camp (a l'exemple busquem que acabi per a):

```
db.posts.find({"post_text ":/a$/})
```

- Buscar un text a un camp tenint en compte la longitud de la cadena

```
db.Posts.find( {$where: "(7 <= this.post_text.length) && (this.post_text.length)" } )
```

## Disseny de BBDD

Mongodb no utilitza els join. ¿Cóm podem implementar les relacions 1:N o N:N i les consultes que faria amb Joins?

### Veiem un cas 1:N.

Cada empleat te un supervisor. Afegim al fitxer empleat un camp que correspon a l'id d'empleat.

Insereixo el "jefe":

```
> db.empleados.insert({_id: ObjectId("11111111111111111111"), nombre:"cap mantenimiento"});
WriteResult({ "nInserted" : 1 })
```

Insereixo els empleats que tingui al anterior com a "jefe":

```
> db.empleado.insert({_id:ObjectId("22222222222222222222"), nombre:"pepe", jefe: ObjectId("11111111111111111111")});
WriteResult({ "nInserted" : 1 })
> db.empleado.insert({_id:ObjectId("33333333333333333333"), nombre:"juan", jefe: ObjectId("11111111111111111111")});
WriteResult({ "nInserted" : 1 })
>
```

Busquem tots els empleats que tenen el "jefe" "cap de manteniment"

```
> db.empleado.find({jefe: ObjectId("11111111111111111111")});
{ "_id" : ObjectId("22222222222222222222"), "nombre" : "pepe", "jefe" : ObjectId("11111111111111111111") }
{ "_id" : ObjectId("33333333333333333333"), "nombre" : "juan", "jefe" : ObjectId("11111111111111111111") }
>
```

a mode de repàs, per veure només els noms:

```
> db.empleado.find({jefe: ObjectId("11111111111111111111")}, {nombre:1, _id:0});
{ "nombre" : "pepe" }
{ "nombre" : "juan" }
```

El problema està en cercar per nom i no per id. Això requerirà una consulta extra i treballar amb el llenguatge de programació que estigui utilitzant.

### Veamos ahora un caso N:N.

Ara resulta que un empleat pot tener més d'un encarregat o "jefe". La solució es afegir un vector de ids en lloc d'un id escalar.

Insereixo un empleat que serà un altre "jefe" i un segon empleat que tindrà dos "jefes":

```
> db.empleado.insert({_id: ObjectId("44444444444444444444444444444444"), nombre:"cap mantenimiento2"});
WriteResult({ "nInserted" : 1 })
```

```
> db.empleado.insert({_id: ObjectId("55555555555555555555555555555555"), nombre:"Empleado con dos jefes", "jefe": [ ObjectId("44444444444444444444444444444444"), ObjectId("11111111111111111111111111111111") ] });
```

Si cerquem els empleats per sota del primer "jefe", la recerca es la mateixa:



```
> db.empleado.find({jefe: ObjectId("1111111111111111111")});
{ "_id" : ObjectId("2222222222222222222"), "nombre" : "pepe", "jefe" : ObjectId("1111111111111111111") }
{ "_id" : ObjectId("3333333333333333333"), "nombre" : "juan", "jefe" : ObjectId("1111111111111111111") }
{ "_id" : ObjectId("5555555555555555555"), "nombre" : "Empleado con dos jefes", "jefe" : [ ObjectId("4444444444444444444"), ObjectId("1111111111111111111") ] }
> db.empleado.find({jefe: ObjectId("1111111111111111111"), {nombre:1, _id:0}});
{ "nombre" : "pepe" }
{ "nombre" : "juan" }
{ "nombre" : "Empleado con dos jefes" }
>
```

## DOCUMENTOS EMBEBITS

Consisteix en introduir un document empleat, per exemple, com un camp d'un altre documento empleat.

```
> db.empleado.insert({_id:ObjectId("77777777777777777777777777777777"), nombre:"raul", jefe: {_id:ObjectId("11111111111111111111111111111111"), nombre:"cap mantenimiento"}});
WriteResult({ "ninserted" : 1 })
```

Aquesta és una manera per poder fer una cerca còm : empleats sota supervisió de l'empleat de nom "cap mantenimiento".

## ¿Quina és la conseqüència dels documents embebits?

La desnormalizació. Hem vist que podem introduir l'ídi el nom d'un empleat com "jefe" d'un altre. El caso es que el nom del "jefe" està repetit per cada empleat al que supervisa.

## Altres comandes

- `db.help()` -> Show help for database methods
- `db.<collection>.help()` -> Show help on collection methods. The `<collection>` can be the name of an existing collection or a non-existing collection
- `show dbs` -> print a list of all dbs on the system
- `use db` -> Switch current database to `<db>`. The mongo shell variable `db` is set to the current database.
- `show collections` -> Print a list of all collections for current database
- `show users` -> Print a list of users for current database.
- `show databases` -> New in version 2.4: Print a list of all available databases
- `db.<collection>.find()` -> Show
- `db.<collection>.find().pretty()` -> will in all the places where produce formatted JSON structure which is more readable

## Exercicis

Utilitzant la col·lecció “Students” que hem importat al principi de la pràctica, realitza les següents consultes:

1. Busqueu els estudiants de gènere masculí
2. Busqueu el estudiants de gènere femení
3. Busqueu els estudiants nascuts l'any 1993
4. Busqueu els estudiants de gènere masculí nascuts a l'any 1993
5. Busqueu els estudiant nascuts a la dècada dels 90
6. Busqueu els estudiants de gènere masculí nascuts abans del l'any 90
7. Busqueu els estudiants de gènere femení nascuts abans del l'any 90
8. Busqueu els estudiants nascuts a la dècada dels 90
9. Busqueu els estudiants de gènere masculí nascuts a la dècada dels 80
10. Busqueu els estudiants de gènere femení nascuts a la dècada dels 80
11. Busqueu els estudiants que no han nascut a l'han 1985
12. Busqueu els estudiants nascuts als anys 1970, 1980 o 1990
13. Busqueu els estudiants no nascuts als anys 1970, 1980 o 1990
14. Busqueu el s estudiants nascuts en any parell
15. Busqueu el s estudiants nascuts en any múltiple de 10
16. Busqueu els estudiants que tinguin telèfon auxiliar
17. Busqueu els estudiants que no tinguin segon cognom
18. Busqueu els estudiants que tinguin telèfon auxiliar i un sol cognom
19. Busqueu els estudiants que tinguin un email que acabi en .net
20. Busqueu els estudiants que tinguin un nom que comenci per vocal
21. Busqueu els estudiants que tinguin un nom més llarg de 13 caràcters
22. Busqueu els estudiants que tinguin un nom amb més de 3 vocals
23. Busqueu els estudiants que tinguin un dni que comenci per lletra
24. Busqueu els estudiants que tinguin un dni que comenci i acabi per lletra
25. Busqueu els estudiants que tinguin telèfon que comenci per 622