EA 1.4.2: Mongo DB

UF4 – Bases de dades NoSQL

Administració de Sistemes Informàtics en Xarxa
M02 Bases de dades Versió 1.0
Laura Villalba Guerrero

Que veurem?

1. Introducció
2. Què és MongoDB?
3. Possible escenaris d'aplicació de MongoDB
4. Instal·lació i configuració
   i. Iniciar el servidor
   ii. Iniciar el client
   iii. Altres paràmetres
5. Com començar
   i. Crear una base de dades
   ii. Crear una collection
6. Operacions de consulta
   i. Tipus de dades utilitzades en MongoDB
   ii. Comandes útils a la consola de MongoDB
   iii. Consultes

Que veurem?

7. Operacions de consulta avançades
   i. Operacions de comparació
   ii. Operacions lògiques
   iii. Operacions segons l'existència o el tipus dels elements
8. Operacions de consulta avançades II
   i. Consultar arrays
   ii. Dot Notation
   iii. Consultes en subdocuments
   iv. Cerques en camps de text amb expressions regulars
   v. Cursors
   vi. Count
   vii. Sort
   viii. Limit
   ix. Skip
   x. To array

Que veurem?

9. Operacions d'actualització de dades
   i. Inserció de dades
   ii. Eliminació de dades
   iii. Actualització de dades

1. Introducció

- MySQL (1995), és sense dubte el SGBD més utilitzat per projectes web, però les noves aplicacions web es caracteritzen per un **immens volum i gran quantitat de dades**.
- L'anterior fet, han augmentat la popularitat de les bases de dades no relacionals, convertint-les en el principal competidor de les bases de dades tradicionals.
- Molts de les bases de dades distribuïdes, com MongoDB, publicada a l'any 2009, estan orientades a documents i es coneixen com bases de dades NoSQL, ja que no depenen únicament de SQL (*Structured Query Language).*
- No fer servir un llenguatge de Querys, implica major exigència a la capa d'aplicació.
- Aquest tipus de base de dades permet la distribució de processos de treball i conjunts de dades en múltiples servers. Això permet que l'**escalabilitat** d'aquestes bases de dades sigui pràcticament **il·limitada**.

2. Què és MongoDB?

- En 2007, la companyia de software 10gen va començar amb el desenvolupament de MongoDB (que prové de l'anglès *humongous=""gegant, enorme").*
- Una base de dades de codi obert per a documents que, gairebé dos anys després de ser publicada, es convertiria en la base de dades NoSQL més popular.
- Fins avui, 10gen, ara sota el nom de MongoDB Inc., ha estat la responsable del desenvolupament del programari i de la distribució comercial de solucions empresarials.
- MongoDB està desenvolupada en C++ i emmagatzema les seves dades en formato BSON (*Binary JSON)*, que es basa , a la seva vegada, en JSON (*JavaScript Object Notation).*
- Gràcies a fer servir JSON, suporta tots el tipus de dades de JavaScript.

2. Què és MongoDB? (II)

- El que pugui fer servir JavaScript, converteix a MongoDB en una BD ideal per plataformes NodeJS.
- Les bases de dades MongoDB contenen una o més col·leccions de dades i administren diversos documents amb varis camps de dades.
- La connexió amb el servidor MongDB pot establir-se de diferents maneres.
- La ***Shell de Mongo***, ve instal·lat per defecte en la majoria de distribucions i serveis per facilitar ***l'accés mitjançant la línia de comandes***.
- És possible activar una interfície administrativa basada en HTTP per accedir al navegador.
- Existeixen diferents interfícies d'usuari com ***MongoChef***, ***Robomongo*** o ***Mongoclient***, que permeten l'edició i representació gràfica de les dades.

3. Possible escenaris d'aplicació de MongoDB

- MongoDB és una excel·lent opció quan es requereixen realitzar projectes web recolzat en grans conjunts de dades sense estructurar.
- Quan no tenim cap esquema, el treball basat en documents és el mètode ideal per gestionar un gran nombre de dades diferents que han de ser ***emmagatzemats i processats ràpidament.***
- L'escalabilitat horitzontal d'aquest sistema és casi il·limitada, doncs les bases de dades es poden distribuir fàcilment en diferents servidors sense comprometre la funcionalitat.
- Per garantir la seguretat i la disponibilitat de les dades a llarg termini, MongoDB facilita la creació de còpies de la totalitat de les dades i les posa a disposició entre els diferents servers.

3. Possible escenaris d'aplicació de MongoDB (II)

- La implementació de MongoDB resultat apropiada per qualsevol tipus de projecte web que es caracteritzin per:
  - **Escalabilitat** → si el projecte web està creixent, probablement augmentarà el nombre de visites i sol·licituds, el que demanda una major capacitat de resposta per part de les bases de dades.
  - **Disponibilitat** → una de les disponibilitats de tot projecte web es la disponibilitat en tot moment, inclòs en el ca de errades en el server.
  - **Flexibilitat** → un projecte ha de poder ajustar-se en qualsevol moment de forma dinàmica.

3. Possible escenaris d'aplicació de MongoDB (III)

- Quan emmagatzemen les dades en una base MongoDB, es presenta un petit interval de temps en el que l'accés de lectura només està permet per les dades més antigues, això és conegut com ***Consistència eventual.ç***

- Aquest model de consistència permet ***l'accés d'escriptura sobre les aplicacions***, degut a que el servidor de la base de dades està separat dels clústers de bases de dades més grans mitjançant les particions de xarxa.

4. Instal·lació i configuració

- A l'àrea de descarregas d ela pàgina oficinal de MongoDB (https://www.mongodb.com/download-center/community?jmp=nav), pots trobar l'edició de codi obert '*Community Server*'.
- Per començar has de descarregar els arxius binaris d'instal·lació per el nostre sistema (*Ubuntu*) i descarregar-los.
- MongoDB és un programa multiplataforma, pel que s'adapta a una gran varietat de sistemes, com Windows, Linux, OSX i Solaris.
- Per Línux i altres sistemes UNIX és necessari descarregar l'arxiu, descomprimir i instal.lar-ho amb ajuda del gestor de paquets corresponent.
- Depenen de la distribució, hauràs d'importar la clau pública GPG per MongoDB.

4. Instal·lació i configuració (II)

GPG (*Gnu Privacy Guard)*, es fa servir per xifrar i signar digitalment, sent a més multiplataforma. Ve incorporat en alguns sistemes Linux, com en Ubuntu.

- Ubuntu requereix aquesta clau d'autenticació. S'ha d'implementar la següent comanda:

```
1 | sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927
```

- A continuació, actualitza la llista del gestor de paquets:

```
1 | sudo apt-get update
```

- Finalment, instal.la MongoDB, així com altres eines útils de gestió:

```
1 | sudo apt-get install -y mongodb-org
```

i. Iniciar el servidor

- Si ho desitges, en el arxiu de configuració */etc/mongod.conf* pots canviar el directori d'instal·lació assignat per defecte */var/lib/mongodb*, així com el registre */var/log/mongodb*.

- Per inicialitzar la base de dades:

```
1 | sudo service mongod start
```

- Si vols parar la base de dades, fes servir el paràmetre *stop*, es ***finalitza l'execució de la base de dades***; amb el paràmetre *restart* podràs reiniciar-la. Per comprovar si MongoDB s'ha iniciat amb èxit, pots buscar la següent línia en l'arxiu de registre */log/mongodb/mongod.log*

```
1 | [initandlisten] waiting for connections on port <port>
```

- El port per defecte per aquest servei és 27017

ii. Iniciar el client

- Una vegada iniciat el servidor, es pot iniciar el client.
- En aquest punt es fa servir ***mongo Shell*** *(*basada en JavaScript), que es fa servir per **l'administració de la base de dades**, per **l'accés** i **l'actualització del conjunt de dades.**
- Per iniciar el client:

```
1 mongo
```

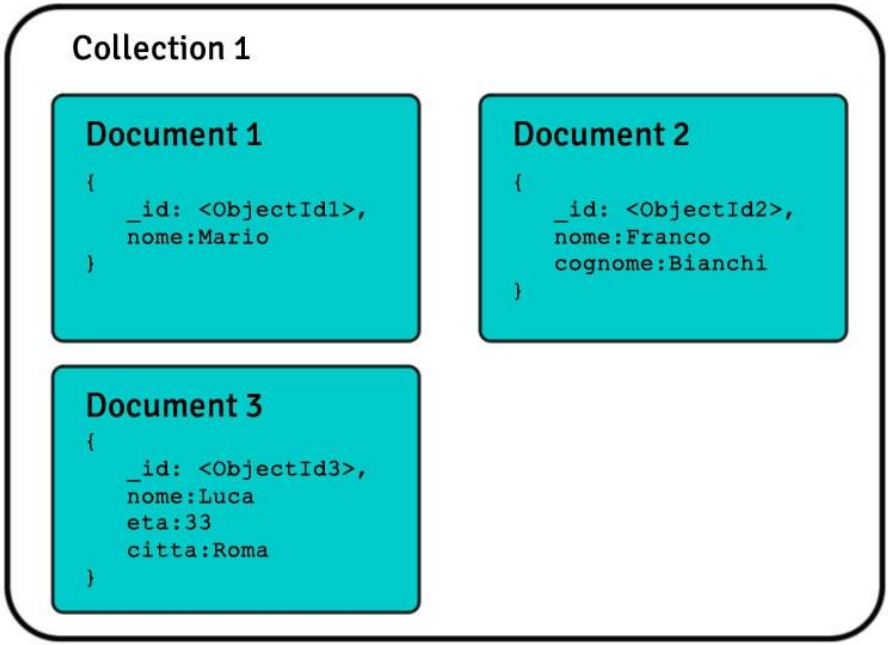
- Mongo Shell es connecta amb la instancia de mongo DB executada en el host i en el port local 27017. Es possible modificar els ajustos estàndards i adaptar-ho a les pròpies necessitats.

iii. Altres paràmetres

Paràmetre	Descripció
--Shell	Activa la interfície del Shell i la presenta després d'executar una comanda.
--nodb	Evita que Mongo Shell es connecti amb una base de dades
--port <port>	Defineix el port per la connexió
--host <hostname>	Defineix el host per la connexió
--username <username> o -u <username>	Una vegada definits els permisos d'accés, pot accedir amb el teu respectiu nom d'usuari (<*username*>).
--password <password> o -p <password>	Quan s'han definit els permisos d'accés, pots accedir utilitzant la contranseya (<*password*>) corresponent, </password>

5. Com començar

- MongoDB és un SGBD orientat a documents. Això vol dir que el que guardem a la base de dades són documents.
- MongoDB guarda els documents en BSON, que és una implementació binaria de JSON.
- Tots els documents guardats en la base de dades es poden tractar com farien en Jscript.
- Els documents es guarden en **col·leccions** (taules en SGBDR).
- Els documents no tenen perquè tenir els mateixos camps, ja que no tenint un esquema definit.



i. Crear una base de dades

- Abans de començar, es necessari crear una base de dades, de lo contrari, les *collections* i els documents s'emmagatzemarà en la base de dades de prova que es genera automàticament.
- La comada ***use*** facilita la creació d'una base de dades.

```
use mibasededatos
```

- Aquesta comanda (*use)* també permet seleccionar una base de dades ja existent.

ii. Crear una col·lecció

- Per crear una col·lecció o carpeta d'arxius per els diferents documents BSON, on es guardarà tota la informació. La sintaxi és:

```
db.createCollection(<name>, { options } )
```

ii. Crear una col·lecció

- La comanda té dos paràmetres: *name* (nom de la carpeta dels arxius) i *options (*opcions de configuració per la col·lecció).
- A les opcions s'especifiquen, per exemple, si s'ha de limitar la mida dels arxius (*capped: true)* o si s'han de limitar la quantitat de bytes (*size:<number>)* o el nombre de documents (*max:<number>)* a una col.lecció.
- Exemple: Una col·lecció esmentada micarpetadearchivos, amb un límit de bytes de 6142800 i un màxims de 10000 documents:

```
1  db.createCollection ("micarpetadearchivos", { capped: true,
2                       size: 6142800,
3                       max: 10000 } )
```

6. Operacions de consulta

i. Tipus de dades

- MongoDB, mitjançant JSON, pot fer servir els següent tipus de dades:
  - **String** → guardats en UTF-8. Sempre amb cometes dobles “
  - **Number**→ nombres. Al guardar-se en BSON pot ser de tipus byte, int32, in64 o double.
  - **Boolean** → amb valor true o false.
  - **Array** → es representa entre [] i pot contenir de 1 a N elements, que poden ser de qualsevol dels tipus anteriors.
  - **Documents** → un document en format JSON pot contenir altres documents embeguts que incloguin més documents o qualsevol dels tipus anteriorment descrits.
  - **Null**

I. Tipus de dades

- Un document complert pot tenir el següent format:
- Com es pot veure en l'exemple hem utilitzat pràcticament tots els tipus que hem descrit.
- Podeu veure també el camp especial _id, creat amb ObjectId. Aquest objecte, que és en realitat tractat com un string, es crea automàticament en tots els documents que inserim en la nostra base de dades i té un valor únic. Podem especificar nosaltres el valor que vulguem, però haurem de tenir en compte que no pot haver-hi dos documents amb el mateix _id.

```
{
    "People":
        [
            {
                "_id": ObjectId("51c420ba77edcdc3ec709218"),
                "nombre": "Manuel",
                "apellidos": "Pérez",
                "fecha_nacimiento": "1982-03-03",
                "altura": 1.80,
                "activo": true,
                "intereses":["fútbol","tenis"],
                "tarjeta_credito": null,
                "dni":
                    {
                        "numero":"465464646J",
                        "caducidad":"2013-10-21"
                    }
            },
            {
                "_id": ObjectId("51c420ba77ed1dc3ec705289"),
                "nombre": "Sara",
                "apellidos": "Ruano",
                "fecha_nacimiento": "1985-12-03",
                "altura": 1.65,
                "activo": false,
                "intereses":["moda","libros","fotografía","política"],
                "tarjeta_credito": null
            }
        ]
}
```

ii. Comandes a la consola de MogoDB

- ***Show dbs*** → la consola mostrarà les bases de dades existents i l'espai consumit.
- ***Use nomdb*** → per exemple: use hr → estarem connectats a la base de dades amb nom hr. Totes les consultes realitzades a partir d'aquí seran per aquesta base de dades.
- ***Show collections*** → ens mostrarà les col·leccions existents en la base de dades que estem fent servir.
- ***Help*** → ens mostrarà una llista de comandes generals que podem utilitzar en la consola.
- ***db.help ()*** → veurem l'ajuda de les comandes aplicables a la base de dades.
- ***db.nocollection.help(***) → veurem les comandes aplicables a una col·lecció en concret.

iii. Consultes

- Per poder realitzar consultes, el primer és **importar dades**. Ja que la nostra base de dades està buida.
- Per importar dades, farem servir la comanda **mongoimport**
- Amb aquesta comanda podem importar dades en diferents formats. Farem servir JSON.
- Per importar una arxiu:

```
./mongoimport --host localhost --port 27666 --db test --collection people --file data.json --jsonArray
```

- Amb la comanda anterior, el que fem es incorporar les dades al nostre server, que està en el port 27666, concretament a una base de dades esmentada ***test*** i a una col·lecció anomenada ***people***.
- Una vegada que tenim importats les dades, ja podem començar a fer consultes. Ens connectarem al servidor amb la comanda ***mongo***, i ens situarem a la base de dades esmentada ***test***.

iii. Consultes

```
PS C:> mongo localhost:27666
MongoDB shell version: 2.4.4
connecting to: localhost:27666/test
> use test
switched to db test
>
```

- **d*b.nombre_de_coleccion.find()*** → Aquest comando pot rebre dos paràmetres: una consulta i una projecció. Tots dos comandos són opcionals pel que si executem la comanda

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

- Obtindrem una llarga llista amb els primers 20 elements de la col.lecció.

iii. Consultes

```
PS C:> mongo localhost:27666
MongoDB shell version: 2.4.4
connecting to: localhost:27666/test
> use test
switched to db test
>
```

- d***b.nombre_de_coleccion.find()*** → Aquest comando pot rebre dos paràmetres: una consulta i una projecció. Tots dos comandos són opcionals pel que si executem la comanda. Obtindrem una llarga llista amb els primers 20 elements de la col·lecció. Amb aquesta consulta no veurem els resultats amb un bon format. ***db.people.find()***
- ***db.nombre_de_coleccion.find().pretty()*** → dona el mateix resultat que l'apartat anterior, però amb un format millor. ***db.people.find().pretty()***
- ***db.people.find({age:34}).pretty()*** → Amb aquesta comanda obtindrem les persones amb l'edat igual a 34 anys. Podem afegir tants filtres com vulguem.

iii. Consultes

- ***db.people.find({age:34,isActive:true}).pretty()*** → Fem servir dos filtres que l'edat sigui 34 i que estigui actiu.
- ***db.people.find({age:34,isActive:true},{name:1,age:1,isActive:1}).pretty()*** → Si volem seleccionar només alguns camps, hem de fer servir el segon paràmetre de la consulta ***find*** per ***definir una projecció***.
- ***db.people.find({age:34,isActive:true},{name:1,age:1,isActive:1,_id:0}).pretty()*** → L' id es mostra sempre per defecte. Si no volem que es mostri, hi ha que especificar-ho.
- ***db.people.find({age:34,isActive:true},{name:0,age:0,isActive:0,_id:0}).pretty()*** → Si volem mostrar tots el camps, però traient només alguns, el que farem serà desactivar els nos desitjats a la projecció.
- ***db.people.findOne({age:34,isActive:true},{name:0,age:0,isActive:0,_id:0})*** → la comanda ***findone*** té el mateix funcionament que la comanda ***find***, amb la diferència que si el comando troba més d'un resultat que compleixi les condicions de la consulta, tan sols ens retornarà el primer. ***FindOne no acepta pretty***

7. Operacions de consulta avançades

- Durant aquestes consultes farem servir molts operadors existents en MongoDB. La llista detallada la podeu trobar a https://docs.mongodb.com/manual/reference/operator/

i. Operacions de comparació

Operation	Syntax	Example
Equality	{<key>:<value>}	db.people.find({nombre:"Laura"}).pretty()
Less Than	{<key>:{$lt:<value>}}	db.people.find({altura:{$lt:1.5}}).pretty()
Less Than Equals	{<key>:{$lte:<value>}}	db.people.find({altura:{$lte:1.5}}).pretty()
Greater Than	{<key>:{$gt:<value>}}	db.people.find({altura:{$gt:1.5}}).pretty()
Greater Than Equals	{<key>:{$gte:<value>}	db.people.find({altura:{$gte:1.5}}).pretty()
Not Equals	{<key>:{$ne:<value>}}	db.people.find({altura:{$ne:1.5}}).pretty()

ii. Operacions lògiques

- **Operador lògic AND** → per fer servir un operador lògic AND només tenim que fer les consultes com en apartats anteriors. Els camps separats per comes.
  - La sintaxis:

```
>db.mycol.find({key1:value1, key2:value2}).pretty()
```

  - Exemple:
    - ***db.people.find({nombre:"laura", altura:1.64}).pretty()***

- **Operador lògic OR** → per realitzar cerques amb condicions OR, hem de fer servir l'operador lògic **$or**.
  - La sintaxis:

```
>db.mycol.find({$or:[{key1:value1}, {key2:value2}]}).pretty()
```

  - El següent exemple mostra les persones que s'anomenen Laura o que el camp actiu és fals.
    - ***db.people.find({$or:[{nombre:"laura"}, {activo:false}]}).pretty()***

ii. Operacions lògiques

- **Operador lògic AND i OR**
  - El següent exemple mostra els documents que tenen persones que la seva mida sigui més gran de 1,80, que tinguin el camp activo en true i que s'anomenin Laura o Pepe
    - ***db.people.find(altura:{$gt:1.80},activo:true,$or:[{nombre:"laura"}, {nombre:"pepe"}]}).pretty()***

iii. Operacions segons l'existència o el tipus dels elements

- MongoDB és una base de dades sense esquema, això vol dir que els documents, encara que siguin de la mateixa col·lecció, poden tenir diferents camps, inclòs aquests camps poden tenir diferents tipus a cada document.
- Es poden realitzar consultes que retornin els documents on existeixi un determinat camp.
  - El següent exemple mostra els documents en els quals existeix un camp company.
    - ***db.people.find({company:{$exists:true}},{name:1, age:1, company:1})***
- Si volem buscar els documents que no tenen el camp company, nomes hem de canviar el true por el fals en el $exists.

iii. Operacions segons l'existència o el tipus dels elements

- MongoDB pot guardar documents amb distints tipus de dades en el mateix camp.
- Si el camp *age* és un *number* per a tots els documents, podem inserir un document nou amb un *string* en aquest camp.
- Podem necessitar filtrar per documents en els quals un camp serà d'un determinat tipus. Per fer això, es realitza amb l'operador ***$type.***
- Al següent exemple cerquem els documents on el camp *company* sigui del tipus 2, que és tipus *string*
  - ***db.people.find({company:{$type:2}},{name:1, age:1, company:1})***
- El següent enllaç podeu trobar la relació entre el nombre i el tipus de dades: https://docs.mongodb.com/manual/reference/operator/query/type/#op._S_type

iii. Operacions segons l'existència o el tipus dels elements

- MongoDB pot guardar documents amb distints tipus de dades en el mateix camp.
- Si el camp *age* és un *number* per a tots els documents, podem inserir un document nou amb un *string* en aquest camp.
- Podem necessitar filtrar per documents en els quals un camp serà d'un determinat tipus. Per fer això, es realitza amb l'operador ***$type.***
- Al següent exemple cerquem els documents on el camp *company* sigui del tipus 2, que és tipus *string*
  - ***db.people.find({company:{$type:2}},{name:1, age:1, company:1})***
- El següent enllaç podeu trobar la relació entre el nombre i el tipus de dades: https://docs.mongodb.com/manual/reference/operator/query/type/#op._S_type

8. Operacions consultes avançades II

i. Consultar arrays

- MongoDB pot guardar arrays d'elements.
- Els elements que es guarden en una *array* pot ser de qualsevol tipus (*string, number,* altres arrays o subdocuments)
- A la col·lecció *people*, cada persona té associat un camp _*tags*_, que és un *array* de *strings.*
- Si volem trobar totes les persones que tenen en el camp _tags_ varis valors, la consulta seria la següent:
  - ***db.people.find({tags:{$all:["laborum","sunt"]}},{name:1,tags:1}***

- Amb l'operador $all, busquem varis elements dins un array. En l'exemple anterior estem cercant les persones que continguin *laborum* i *sunt* en el camp _tag_
- Per que la cerca retorni aquells documents que només continguin almenys un valor, farem servir l'operador **$in**
  - ***db.people.find({tags:{$in:["laborum","sunt", "nisi"]}},{name:1,tags:1}***

i. Consultar arrays

- Podem fer servir l'operador ***$slice***, que és un tipus [***skip, límit***]
  - ***db.people.find({tags:{$in:["laborum","sunt","nisi"]}}, {tags:{$slice:[2,3]},name:1})***

- S'ignoren els dos primers elements de l'array i s'agafen només els tres següents.
- Si volem començar a buscar pel final de l'array, tenim que utilitzar un nombre negatiu.
  - ***db.people.find({tags:{$in:["laborum","sunt","nisi"]}}, {tags:{$slice:[-2,3]},name:1})***

ii. Dot Notation

- Es fan servir en MongoDB per realitzar consultes en *array* i en subdocuments.
- Afegirem un punt després de l'identificador de l'array o subdocument per realitzar consultes sobre un índex en concret de l'array o sobre un camp en concret del subdocument.
  - ***db.people.find({"tags.1":"enim"})***

- A l'exemple busquem tots el documents que compleixen la condició de que el valor 1 de l'array sigui "enim".
- Els arrays comencen amb l'índex 0 i és necessari que "tags.1" estigui entre cometes.

iii. Consulta a subdocuments

- En el nostres dades d'exemple existeix un camp esmentat *friends,* que conté un array de subdocuments.
- Si en aquest camp, volem trobar els elements que contenen en subdocument compost por l'id 1 i el nom "Trinity Ford" utilitzarem la consulta:

  - ***db.people.find({friends: {id;1, name:"Trinity Ford"}})***

- En aquest cas només retorna els documents que en l'array del camp friends té el subdocument _{id;1, name:"Trinity Ford"}_.

- Si volem cercar per un camp del subdocument en concret, haurem d'usar dot notation.
  - ***db.people.find({friend.name":"Trinity Ford"})***

- Si fem servir ***dot notation***, podem fer consultes més precises i complexes:
  - ***db.people.find({"friends.2.name":{gte:"T"}},{friends:{$slice:-1},name:1})***

  - *A la cerca anterior trobarem en l'arraya _friends_ els elements que estiguin a la posició 2 i que el seu noms sigui major o igual que T. A més a la projecció mostrem l'últim element, que es pel que estem filtrant.*

iv. Expressions regulars

- Per realitzar cerques amb patrons dins un camp de text, hem de fer servir *regular expressions.*
- Per fer servir *regular expressions* en MongoDB farem servir l'operador ***$regex.***
- Podeu trobar informació més completa sobre aquest tema a https://docs.mongodb.com/manual/reference/operator/query/regex/index.html
- Exemples
  - ***db.people.find({"name":{$regex:".fis"}}, {name:1})***
  - ***db.people.find({"name": {$regex:".Fis"}}, {name:1})***

- A les consultes anteriors cerquem elements que continguin _"fis"_ o "Fis". Per defecte, les expressions regulars són ***case sensitive***. Per ignorar les majúscules hem de fer servir l'operador **$options**

  - ***db.people.find({"name": {$regex:".fis", $options:"i"}}, {name:1})***

- Si fem servir ***regex*** les cerques són més lentes.

v. Cursors

- Un cursor és un iterador sobre els resultats d'una consulta.
- El cursor està en el servidor, mentre que en el client només tenim l'identificador d'aquest.
- Amb aquest sistema s'eviten moure dades innecessàries del servidor al client, ja que el cursor per defecte només retorna 20 resultats. Si volem mostrar més, hem d'escriure "***it***" en la consola, el què ens retornarà els següents 20 resultats.

vi. Count

- Retorna el nombre de documents retornats per una consulta.
  - ***db.people.find ({"friends.2.name": {$gte:"t"}}).count()***

vii. Sort

- Ordena els resultats pel camp especificat. La següent consulta ordena de forma ascendent
  - ***db.people.find({"friends.2.name":{$gte:"T"}},{_aneu:0,name:1}).sort({name:1})***
- I la següent consulta ordena de forma descendent.
  - ***db.people.find({"friends.2.name":{$gte:"T"}},{_aneu:0,name:1}).sort({name: -1})***
- Podem especificar més d'un camp separant-lo per comes.
  - ***db.people.find({"friends.2.name":{$gte:"T"}},{_aneu:0,name:1,email:1}).sort({name: 1,email:1})***

viii. Limit

- Limita el nombre de resultats retornats. El següent exemple retorna els cinc primers documents.
  - ***db.people.find({"friends.2.name":{$gte:"T"}},{name:1}).limit(5)***

ix. Skip

- Ignora els N primers documents especificats. El següent exemple salta els cinc primers documents i retorna els següents.
  - ***db.people.find({"friends.2.name":{$gte:"T"}},{name:1}).skip(5)***

x. To array

- Guarda els resultats en un array que podem assignar a una variable
  - ***var myArray = db.people.find({"friends.2.name":{$gte:"T"}},{name:1}).toArray()***
- El que té de bo tots aquests comandos, és que es poden concatenar. Per exemple per a saltar els 10 primers documents retornats i agafar els 5 següents podem usar la següent consulta:
  - ***db.people.find({"friends.2.name":{$gte:"T"}},{name:1}).skip(10).limit(5)***
- També podem ordenar els resultats per ordre ascendent i agafar només el primer, retornant el valor més baix.
  - ***db.people.find({"friends.2.name":{$gte:"T"}}, {name:1}).sort({name:1}).limit(1)***

9. Operacions consultes avançades II

- Una base de dades és capaç de realitzar qualsevol operació CRUD (*Create, Read, Update i Delate)*.
- A partir d'aquí veurem com inserta, modificar i esborrar dades

i. Inserció de dades

Insert

- La sintaxis per fer un insert és la següent:

```
>db.COLLECTION_NAME.insert(document)
```

- Com es pot veure, la inserció és molt senzilla. Ens n'hi ha prou amb passar com a paràmetre el document JSON que volem inserir.
- Podem inserir manualment el camp _id. Això és una cosa totalment vàlida, sempre que tinguem en compte que el _id és un camp únic.
- Si tractem d'executar una altra inserció, però amb el mateix _id, la shell de MongoDB ens mostrarà un error.

i. Inserció de dades

Insert

- MongoDB crea sempre un índex únic per al camp _id que existeix en totes les col·leccions i en tots els documents.
- Si no tenim necessitat que el camp _id tingui un format especial, el més senzill és deixar que MongoDB el generi automàticament per a cada inserció.
- Per a això haurem d'ignorar el camp _id i no introduir-lo juntament amb el document JSON que estem inserint.

```
db.test.insert({
    nombre: "Ramón Ramírez",
    telefonos: [
        "465465456"
    ],
    direccion: {
        calle: "C\ Este",
        numero: 26,
        extra: "Portal 8, 4º izquierda",
        ciudad: "Madrid"
    }
})
```

i. Inserció de dades

Save

- Per a inserir documents també podem utilitzar el comando ***save***, que s'utilitza de la mateixa manera que ***insert***.

```
db.products.save({
    _id: 890,
    nombre: "Portátil Lenovo",
    cantidad: 7,
    precio: 800
})
```

- Igual que amb ***insert***, si no especifiquem el camp _id, MongoDB l'inserirà automàticament.
- La diferència entre un ***insert*** i un ***save***, és que si quan fem un ***save*** _id que especifiquem ja existeix, l'actualitzat o modifica. Si fem servir un ***insert*** amb el mateix _id, ens donarà un error.

i. Inserció de dades

Save

- Per a inserir documents també podem utilitzar el comando ***save***, que s'utilitza de la mateixa manera que ***insert***.

```
db.products.save({
    _id: 890,
    nombre: "Portátil Lenovo",
    cantidad: 7,
    precio: 800
})
```

- Igual que amb ***insert***, si no especifiquem el camp _id, MongoDB l'inserirà automàticament.
- La diferència entre un ***insert*** i un ***save***, és que si quan fem un ***save*** _id que especifiquem ja existeix, l'actualitzat o modifica. Si fem servir un ***insert*** amb el mateix _id, ens donarà un error.

ii. Eliminació de dades

- Per a eliminar dades de la nostra col·lecció, utilitzarem la comanda ***remove***.

```
db.products.remove({_id:890})
```

- Aquesta comanda rep com a paràmetre la consulta que s'utilitzarà per a filtrar els documents que s'esborraran. Si no especifiquem cap consulta, s'eliminaran totes les dades de la col·lecció.
- La comanda ***remove*** accepta un segon paràmetre de tipus booleà anomenat ***justOne***. Per defecte és ***false***, però si ho establim com true, només s'esborrarà un document de la col·lecció, encara que existeixin diversos que compleixin les condicions especificades en la consulta.

```
db.collection.remove(
   <query>,
   <justOne>
)
```

iii. Actualització de dades

- Per a actualitzar dades, utilitzarem la comanda ***update***.
- Aquesta comanda, en la seva forma bàsica, rep dos paràmetres: un amb la consulta per a filtrar els documents a modificar i un altre amb els elements que es modificaran.
- La sintaxi:

```
>db.COLLECTION_NAME.update(SELECTIOIN_CRITERIA, UPDATED_DATA)
```

```
db.products.update({nombre:"HDD Maxtor"},{cantidad:30})
```

- Quan només volem actualitzar un camp, hem de fer servir l'operador **$set**.
- Aquest operador serveix per indicar que només ha d'actualitzar el camp especificat, sinó ho fem, ens actualitzarà el registre amb només un camp.

```
db.products.update({tipo:"HDD"},{$set:{cantidad:0}})
```

iii. Actualització de dades

Opció multi

- MongoDB només actualitza un document tret que li indiquem el contrari.
- Per actualitzar més un document que compleixi amb la mateixa condició, hem d'afegir l'opció **multi**.
- El següent exemple cerca els productes de tipus HDD i actualitza la seva quantitat a 0. Però només d'un registre.

```
db.products.update({tipo:"HDD"},{$set:{cantidad:0}})
```

- Per actualitzar tots els registres que compleixen la mateixa condició, hem de fer servir l'opció multi:

```
> db.products.update({tipo:"HDD"},{$set:{cantidad:10}},{multi:true})
```

iii. Actualització de dades

Opció upsert

- L'opció upsert, ens insereix els document si aquest no existeix.
- Similar a la comanda *save* que hem vist a les operacions d'inserció.
- El següent exemple cercarà elements de tipus "RAM", però al no existir cap inserirà un nou document.

```
db.products.update({
    tipo: "RAM"
},
{
    nombre: "Kingston 2Gb",
    cantidad: 50,
    precio: 26.50,
    tipo: "RAM"
},
{
    upsert: true
})
```

- Si tornem a executar la consulta, canviant 2GB per 4GB, el document s'actualitzarà ja que de la consultar anterior existeix un document de tipus "RAM"

```
db.products.update({
    tipo: "RAM"
},
{
    nombre: "Kingston 4Gb",
    cantidad: 50,
    precio: 26.50,
    tipo: "RAM"
},
{
    upsert: true
})
```

Preguntes!!!!!