BASES DE DADES DOCUMENTALS

Una base de dades documental està constituïda per un conjunt de programes que emmagatzemen, recuperen i gestionen dades de documents o dades d'alguna manera estructurats.

Aquest tipus de bases de dades constitueixen una de les principals subcategories dins de les anomenades bases de dades no relacionals o NoSQL. Els sistemes NoSQL proveeixen un sistema d'emmagatzematge molt més flexible i concurrent, permeten manipular grans quantitats d'informació de manera molt més ràpida que les bases de dades relacionals.

En el següent quadre podem veure una comparativa de diferents característiques dels Sistemes Gestors de Bases de Dades entre una base de dades documental i una base de dades relacional:

Key Areas	SQL	NoSQL
Type of database	Relational Database	Non-relational Database
Schema	Pre-defined Schema	Dynamic Schema
Database Categories	Table based Databases	Document-based databases, Key-value stores, graph stores, wide column stores
Complex Queries	Good for complex queries	Not a good fit for complex queries
Hierarchical Data Storage	Not the best fit	Fits better when compared to SQL
Scalability	Vertically Scalable	Horizontally Scalable
Language	Structured Query language	Unstructured Query language
Online Processing	Used for OLTP	Used for OLAP
Base Properties	Based on ACID Properties	Based on CAP Theorem
External Support	Excellent support is provided by all SQL vendors	Rely on community support.

Differences Between SQL & NoSQL Databases – MySQL & MongoDB Comparison

MONGODB

MongoDB (de l'anglès Humongous, "enorme") és un sistema de base de dades multiplataforma, NoSQL, orientat a documents i de codi obert.

Cada entrada o registre pot tenir un esquema de dades diferent, amb atributs o "columnes" que no tenen per què repetir-se d'un registre a un altre.

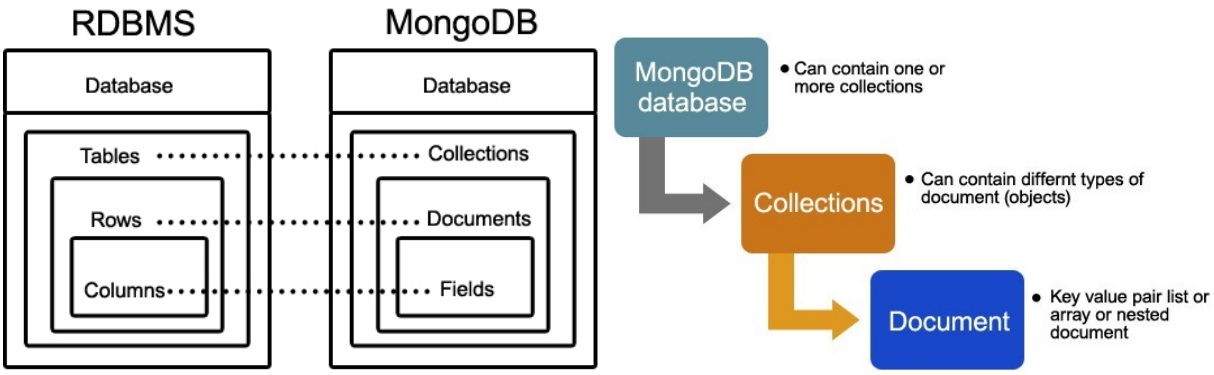
MongoDB destaca per la seva velocitat i el seu ric però senzill sistema de consulta dels continguts de la base de dades. Disposa d'un balanç perfecte entre rendiment i funcionalitat.

Cada registre o conjunt de dades s'anomena **document**. Cada document conté un conjunt d'**atributs** o camps formats per parells clau, valor.

Els documents es poden agrupar en **col·leccions**, les quals es podria dir que són l'equivalent a les taules en una base de dades relacional (només que les col·leccions poden emmagatzemar documents amb molt diferents formats, en lloc d'estar sotmesos a un esquema fix).

Es poden crear índexs per a alguns atributs dels documents, de manera que MongoDB mantindrà una estructura interna eficient per a l'accés a la informació pels continguts d'aquests atributs.

Els diferents documents s'emmagatzemen en format BSON, o Binary JSON, que és una versió modificada de JSON que permet cerques ràpides de dades.



INSTAL·LACIÓ DE MONGODB

MongoDB ens ofereix diferents opcions per treballar amb la seva base de dades:

- Instal·lació On-Premise de MongoDB Community o Enterprise Edition, en un servidor Linux, Windows, MacOS o en contenidors Docker.
    - Instal·lació MongoDB en Linux
    - Instal·lació MongoDB en Windows
- Creació d'un cluster en l'anomenada plataforma multi-cloud MongoDB Atlas.
    - Registre MongoDB Atlas

En una instal·lació On-premise en un servidor Linux es crearà un servei anomenat **mongod** que és qui gestiona l'accés a la base de dades, i que inicialment s'haurà d'habilitar i engegar.

```
campalans@srv-postgres:~$ sudo systemctl status mongod
○ mongod.service - MongoDB Database Server
     Loaded: loaded (/lib/systemd/system/mongod.service; disabled; vendor preset: enabled)
     Active: inactive (dead)
       Docs: https://docs.mongodb.org/manual

de des. 15 14:24:17 srv-postgres systemd[1]: Started MongoDB Database Server.
de des. 15 14:24:18 srv-postgres mongod[655]: {"t":{"$date":"2023-12-15T14:24:18.299Z"},"s":"I",  "c":"CONTROL",
de des. 15 14:45:12 srv-postgres systemd[1]: Stopping MongoDB Database Server...
de des. 15 14:45:12 srv-postgres systemd[1]: mongod.service: Deactivated successfully.
de des. 15 14:45:12 srv-postgres systemd[1]: Stopped MongoDB Database Server.
de des. 15 14:45:12 srv-postgres systemd[1]: mongod.service: Consumed 7.822s CPU time.
campalans@srv-postgres:~$ sudo systemctl enable mongod
Created symlink /etc/systemd/system/multi-user.target.wants/mongod.service → /lib/systemd/system/mongod.service.
campalans@srv-postgres:~$ sudo systemctl start mongod
campalans@srv-postgres:~$ sudo systemctl status mongod
● mongod.service - MongoDB Database Server
     Loaded: loaded (/lib/systemd/system/mongod.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2023-12-15 14:45:47 UTC; 5s ago
       Docs: https://docs.mongodb.org/manual
   Main PID: 1901 (mongod)
     Memory: 169.4M
        CPU: 1.044s
     CGroup: /system.slice/mongod.service
             └─1901 /usr/bin/mongod --config /etc/mongod.conf

de des. 15 14:45:47 srv-postgres systemd[1]: Started MongoDB Database Server.
de des. 15 14:45:47 srv-postgres mongod[1901]: {"t":{"$date":"2023-12-15T14:45:47.603Z"},"s":"I",  "c":"CONTROL",
campalans@srv-postgres:~$
```

A més del servei mongod trobarem un fitxer de configuració a /etc anomenat **mongod.conf**. En ell podem trobar informació com la ruta on es desen els fitxers de la base de dades, els fitxers de Log, el port i la IP per on escolta el servei mongod o la configuració regional.

```
  GNU nano 6.2                                                  /etc/mongod.conf
# mongod.conf

# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/

# Where and how to store data.
storage:
  dbPath: /var/lib/mongodb
#  engine:
#  wiredTiger:

# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log

# network interfaces
net:
  port: 27017
  bindIp: 127.0.0.1


# how the process runs
processManagement:
  timeZoneInfo: /usr/share/zoneinfo

#security:

#operationProfiling:
```

Opcions de configuració

EINES DE TREBALL AMB MONGODB

Tant si hem instal·lat MongoDB en el nostre servidor com si hem creat un cluster en el cloud MongoDB Atlas, tenim principalment dues eines per connectar i treballar directament amb la base de dades:

- **mongosh**: Entorn CLI que ens permet gestionar la base de dades mitjançant comandes.
- **MongoDBCompass**: Entorn gràfic que permet gestionar la base de dades mitjançant eines gràfiques i/o comandes. MongoDBCompass

PRIMERES COMANDES AMB MONGODB

En aquest tema ens centrarem en les comandes de l'entorn CLI Mongodb Shell.

Executem mongosh des del servidor i accedim al SGBD MongoDB. El primer que veurem és la cadena de connexió (en verd). Aquesta servirà per connectar amb la base de dades des de la línia d'ordres, o bé, des d'una aplicació externa.

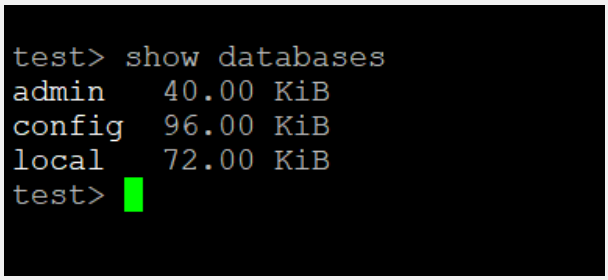
```
campalans@srv-postgres:~$ mongosh
Current Mongosh Log ID: 657c692dec89b85bba4c0953
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.1.1
Using MongoDB:          7.0.4
Using Mongosh:          2.1.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2023-12-15T14:45:47.614+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http
dnotes-filesystem
   2023-12-15T14:45:48.615+00:00: Access control is not enabled for the database. Read and write access to data and configurati
   2023-12-15T14:45:48.615+00:00: vm.max_map_count is too low
------

test>
```

Algunes comandes interessants:

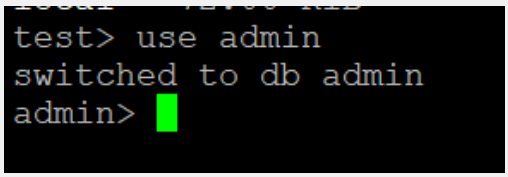
help: La comanda help ens ofereix una llista de les comandes més comunes amb la seva descripció.

> [!Important] L'entorn mongosh és 'case sensitive', per tant diferencia entre majúscules i minúscules.

show: Comanda per visualitzar diferents objectes: bases de dades, col·leccions, usuaris, rols.

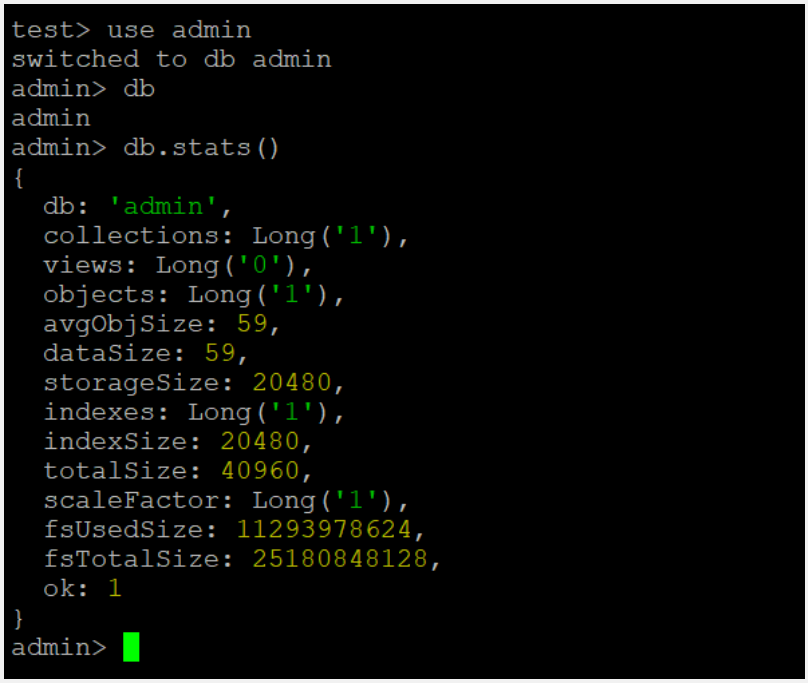
> Exemple: Visualitzar quines bases de dades tenim instal·lades.
>
> 
>

use: Per connectar a una base de dades i treballar amb ella, o fins i tot crear una nova base de dades.

> Exemple: Connectar i treballar amb la base de dades 'admin'.
>
> 
>

TREBALL AMB BASES DE DADES

db: Mostra el nom de la base de dades en ús. Per defecte test.
db.stats(): Mostra la informació d'estat de la base de dades en ús.

> Exemple: Mostrar informació de la base de dades 'admin'.
>
> 
>

use <database_name>: Crea una nova base de dades o connecta amb la base de dades si ja existeix. Amb la comanda *show dbs* o *show databases* no veurem la base de dades perquè només mostra les bases de dades

no buides.

> Exemple: Creem la base de dades 'angel' i mostrem les bases de dades.

```
test> use angel
switched to db angel
angel> db
angel
angel> show dbs
admin   40.00 KiB
config  72.00 KiB
local   72.00 KiB
angel>
```

db.dropDatabase(): Elimina la base de dades en ús.

> Exemple: Eliminem la base de dades anomenada 'angel'.

```
angel> db.dropDatabase()
{ ok: 1, dropped: 'angel' }
angel>
```

TREBALL AMB COL·LECCIONS

db.createCollection(<collection_name>): Crea una nova colecció a la base de dades en ús anomenada <collection_name>.

> Exemple: Creem una col·lecció anomenada 'test' en la base de dades 'biblioteca'.

```
admin> use biblioteca
switched to db biblioteca
biblioteca> db
biblioteca
biblioteca> db.createCollection("llibres")
{ ok: 1 }
biblioteca> db.createCollection("autors")
{ ok: 1 }
biblioteca> db.createCollection("test")
{ ok: 1 }
biblioteca> show collections
autors
llibres
test
biblioteca>
```

db.<collection_name>.drop(): Elimina la col·lecció anomenada <collection_name>.

> Exemple: Eliminem la col·lecció anomenada 'test' de la base de dades 'biblioteca'

```
biblioteca> show collections
autors
llibres
test
biblioteca> db.test.drop()
true
biblioteca> show collections
autors
llibres
biblioteca>
```

COMANDES CRUD

db.<collection_name>.insertOne: Insereix un document a la col·lecció <collection_name>

Exemple: Inserim un autor a la col·lecció 'autors' de la base de dades 'biblioteca'

```
biblioteca> db.autors.insertOne({ id: "1", cognoms: "rodoreda i gurguí", nom: "mercè" } );
{
  acknowledged: true,
  insertedId: ObjectId('657c89410e01e6c085a6d269')
}
```

db.<collection_name>.insertMany: Insereix diversos documents a la col·lecció <collections_name>

Exemple: Inserim diversos autors a la col·lecció 'autors' de la base de dades 'biblioteca'

```
biblioteca> db.autors.insertMany([
... { id: "2", cognoms: "oller y moragas", nom: "narcís"},
... { id: "3", cognoms: "carner i puig-oriol", nom: "josep"},
... { id: "4", cognoms: "català", nom: "víctor"}
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('657c89fc0e01e6c085a6d26a'),
    '1': ObjectId('657c89fc0e01e6c085a6d26b'),
    '2': ObjectId('657c89fc0e01e6c085a6d26c')
  }
}
```

**db.<collection_name>.find(): Cerca tots els documents de la col·lecció <collection_name>

Exemple: Obtenir tots els documents de la col·lecció 'autors' de la base de dades 'biblioteca'

```
biblioteca> db.autors.find()
[
  {
    _id: ObjectId('657c89410e01e6c085a6d269'),
    id: '1',
    cognoms: 'rodoreda i gurguí',
    nom: 'mercè'
  },
  {
    _id: ObjectId('657c89fc0e01e6c085a6d26a'),
    id: '2',
    cognoms: 'oller y moragas',
    nom: 'narcís'
  },
  {
    _id: ObjectId('657c89fc0e01e6c085a6d26b'),
    id: '3',
    cognoms: 'carner i puig-oriol',
    nom: 'josep'
  },
  {
    _id: ObjectId('657c89fc0e01e6c085a6d26c'),
    id: '4',
    cognoms: 'català',
    nom: 'víctor'
  }
]
```

db.<collection_name>.find(): En la funció find podem utilitzar condicions de cerca.

Exemple 1: Cercar l'autor amb identificador 3.

```
biblioteca> db.autors.find({ id: {$eq: "1"}})
[
  {
    _id: ObjectId('657c89410e01e6c085a6d269'),
    id: '1',
    cognoms: 'rodoreda i gurguí',
    nom: 'mercè'
  }
]
```

Exemple 2: Cercar els autors amb nom 'victor'.

```
biblioteca> db.autors.find({ nom: {$eq: "víctor"}})
[
  {
    _id: ObjectId('657c89fc0e01e6c085a6d26c'),
    id: '4',
    cognoms: 'català',
    nom: 'víctor'
  }
]
```

Exemple 3: Cercar els autors amb identificador menor o igual que 3.

```
biblioteca> db.autors.find({ id: {$lte: "3"}})
[
  {
    _id: ObjectId('657c89410e01e6c085a6d269'),
    id: '1',
    cognoms: 'rodoreda i gurguí',
    nom: 'mercè'
  },
  {
    _id: ObjectId('657c89fc0e01e6c085a6d26a'),
    id: '2',
    cognoms: 'oller y moragas',
    nom: 'narcís'
  },
  {
    _id: ObjectId('657c89fc0e01e6c085a6d26b'),
    id: '3',
    cognoms: 'carner i puig-oriol',
    nom: 'josep'
  }
]
```

db.<collection_name>.find().projection(<fields>): Amb la funció find també podem utilitzar *projeccions*, és a dir no mostrar el valor de totes les propietats d'un document.
Sintaxi: ***projection({propietat1: [0|1], propietat2: [0|1], ... , propietatN: [0|1]})***

- 0: No mostrar la propietat.
- 1: Mostrar la propietat.

Exemple 1: Quan utilitzem la comanda find per mostrar dades dels documents d'una col·lecció per defecte sempre ens mostra l'id intern que MongoDB assigna a tots els documents, el camp anomenat _id. En aquest exemple volem veure els documents de la col·lecció 'places' de la base de dades 'parking' però no volem que ens mostri l'id intern.

```
parking> db.places.find().projection({_id: 0})
[
  { id: '01', grandaria: 'petita', planta: '1', ocupada: 'no' },
  { id: '02', grandaria: 'petita', planta: '1', ocupada: 'no' },
  { id: '03', grandaria: 'normal', planta: '1', ocupada: 'no' },
  { id: '04', grandaria: 'normal', planta: '1', ocupada: 'no' },
  { id: '05', grandaria: 'normal', planta: '1', ocupada: 'no' },
  { id: '06', grandaria: 'normal', planta: '1', ocupada: 'no' },
  { id: '07', grandaria: 'gran', planta: '1', ocupada: 'no' },
  { id: '08', grandaria: 'gran', planta: '1', ocupada: 'no' },
  { id: '09', grandaria: 'normal', planta: '2', ocupada: 'no' },
  { id: '10', grandaria: 'normal', planta: '2', ocupada: 'no' },
  { id: '11', grandaria: 'normal', planta: '2', ocupada: 'no' },
  { id: '12', grandaria: 'normal', planta: '2', ocupada: 'no' },
  { id: '13', grandaria: 'normal', planta: '2', ocupada: 'no' },
  { id: '14', grandaria: 'normal', planta: '2', ocupada: 'no' },
  { id: '15', grandaria: 'normal', planta: '2', ocupada: 'no' },
  { id: '16', grandaria: 'normal', planta: '2', ocupada: 'no' },
  { id: '17', grandaria: 'gran', planta: '3', ocupada: 'no' },
  { id: '18', grandaria: 'gran', planta: '3', ocupada: 'no' },
  { id: '19', grandaria: 'gran', planta: '3', ocupada: 'no' },
  { id: '20', grandaria: 'gran', planta: '3', ocupada: 'no' }
]
Type "it" for more
parking>
```

Exemple 2: En aquest exemple volem veure els documents de la col·lecció 'places' de la base de dades 'parking' però només volem que ens mostri els camps id i ocupada.

```
parking> db.places.find().projection({id: 1, ocupada: 1})
[
  {
    _id: ObjectId('6581b88910681a3f6b0f1148'),
    id: '01',
    ocupada: 'no'
  },
  {
    _id: ObjectId('6581b88910681a3f6b0f1149'),
    id: '02',
    ocupada: 'no'
  },
  {
    _id: ObjectId('6581b88910681a3f6b0f114a'),
    id: '03',
    ocupada: 'no'
  },
  {
    _id: ObjectId('6581b88910681a3f6b0f114b'),
    id: '04',
    ocupada: 'no'
  },
  {
    _id: ObjectId('6581b88910681a3f6b0f114c'),
    id: '05',
    ocupada: 'no'
  },
```

db.<collection_name>.find().sort(<fields>): Amb la funció find també podem *ordenar* la sortida segons alguna propietat.
Sintaxi: **(sort(propietat: [1|-1])**

- 1: Ordre ascendent.
- -1: Ordre descendent.

Exemple: Wolem veure els documents de la col·lecció 'places' de la base de dades 'parking' però només volem que ens mostri els camps id, ocupada i planta, a més que la sortida estigui ordenada per planta en ordre descendent.

```
parking> db.places.find().projection({id: 1, ocupada: 1, planta: 1, _id: 0}).sort({planta: -1})
[
  { id: '17', planta: '3', ocupada: 'no' },
  { id: '18', planta: '3', ocupada: 'no' },
  { id: '19', planta: '3', ocupada: 'no' },
  { id: '20', planta: '3', ocupada: 'no' },
  { id: '09', planta: '2', ocupada: 'no' },
  { id: '10', planta: '2', ocupada: 'no' },
  { id: '11', planta: '2', ocupada: 'no' },
  { id: '12', planta: '2', ocupada: 'no' },
  { id: '13', planta: '2', ocupada: 'no' },
  { id: '14', planta: '2', ocupada: 'no' },
  { id: '15', planta: '2', ocupada: 'no' },
  { id: '16', planta: '2', ocupada: 'no' },
  { id: '01', planta: '1', ocupada: 'no' },
  { id: '02', planta: '1', ocupada: 'no' },
  { id: '03', planta: '1', ocupada: 'no' },
  { id: '04', planta: '1', ocupada: 'no' },
  { id: '05', planta: '1', ocupada: 'no' },
  { id: '06', planta: '1', ocupada: 'no' },
  { id: '07', planta: '1', ocupada: 'no' },
  { id: '08', planta: '1', ocupada: 'no' }
]
Type "it" for more
```

db.<collection_name>.distinct(<field>): Mostra tots els documents amb diferent valor en la propietat de la col·lecció <collection_name>. Podem utilitzar més d'una propietat separant per el caràcter coma.

Exemple: Volem veure els noms diferents que existeixen a la col·lecció 'autors' de la base de dades 'biblioteca'.

```
biblioteca> db.autors.find()
[
  {
    _id: ObjectId('6581ace98a28b7c141dba943'),
    id: '2',
    cognoms: 'oller y moragas',
    nom: 'narcís'
  },
  {
    _id: ObjectId('6581ace98a28b7c141dba944'),
    id: '3',
    cognoms: 'carner i puig-oriol',
    nom: 'josep'
  },
  {
    _id: ObjectId('6581ace98a28b7c141dba945'),
    id: '4',
    cognoms: 'català',
    nom: 'josep'
  }
]
biblioteca> db.autors.distinct("nom")
[ 'josep', 'narcís' ]
biblioteca>
```

db.<collection_name>.countDocuments(): Conta el nombre de documents que conté la colecció <collection_name>.

Exemple: Volem saber quants documents té la col·lecció 'places' de la base de dades 'parking'.

```
parking> db.places.countDocuments()
20
parking>
```

Podem afegir una condició, llavors contará quants documents compleixen la condició.

Exemple: Volem saber quants documents on el camp 'grandaria' tingui el valor 'gran' que té la col·lecció 'places' de la base de dades 'parking'.

```
parking> db.places.countDocuments()
20
parking> db.places.countDocuments({"grandaria": "gran"})
6
parking>
```

db.<collection_name>.updateOne(<condition>, <set statement>): Actualitza un document de la col·lecció <collection_name> segons la condició . Si la condició retorna més d'un document només actualitza el primer.

Exemple: Volem actualitzar el nom de l'autor amb id 4 de la col·lecció 'autors' de la base de dades 'biblioteca'

```
biblioteca> db.autors.updateOne({"id": "4"},{$set : {"nom": "Caterina"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
```

db.<collection_name>.updateMany(<condition>, <set statement>): Actualitza diversos documents de la colecció <collection_name> segons la condició . Per actualitzar tots els documents de la colecció deixem la condició en blanc amb {}

Exemple: Consultem tots els autors de la col·lecció 'autors' de la base de dades 'biblioteca'

```
biblioteca> db.autors.find()
[
  {
    _id: ObjectId('657c89410e01e6c085a6d269'),
    id: '1',
    cognoms: 'rodoreda i gurguí',
    nom: 'mercè'
  },
  {
    _id: ObjectId('657c89fc0e01e6c085a6d26a'),
    id: '2',
    cognoms: 'oller y moragas',
    nom: 'narcís'
  },
  {
    _id: ObjectId('657c89fc0e01e6c085a6d26b'),
    id: '3',
    cognoms: 'carner i puig-oriol',
    nom: 'josep'
  },
  {
    _id: ObjectId('657c89fc0e01e6c085a6d26c'),
    id: '4',
    cognoms: 'català',
    nom: 'Caterina'
  }
]
```

A continuació, actualitzem el nom de tots els autors amb id igual o superior a 2

```
biblioteca> db.autors.updateOne({"id": {$gte: "2"}},{$set : {"nom": "josep"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
biblioteca> db.autors.find()
[
  {
    _id: ObjectId('657c89410e01e6c085a6d269'),
    id: '1',
    cognoms: 'rodoreda i gurguí',
    nom: 'mercè'
  },
  {
    _id: ObjectId('657c89fc0e01e6c085a6d26a'),
    id: '2',
    cognoms: 'oller y moragas',
    nom: 'josep'
  },
  {
    _id: ObjectId('657c89fc0e01e6c085a6d26b'),
    id: '3',
    cognoms: 'carner i puig-oriol',
    nom: 'angel'
  },
  {
    _id: ObjectId('657c89fc0e01e6c085a6d26c'),
    id: '4',
    cognoms: 'català',
    nom: 'angel'
  }
]
```

db.<collection_name>.deleteOne(): Elimina un document de la col·lecció <collection_name>. Si la condició retorna més d'un document només elimina el primer.

Exemple: Eliminem l'autor amb nom 'Josep' de la col·lecció 'autors' de la base de dades 'biblioteca'

```
biblioteca> db.autors.find()
[
  {
    _id: ObjectId('657c89410e01e6c085a6d269'),
    id: '1',
    cognoms: 'rodoreda i gurguí',
    nom: 'mercè'
  },
  {
    _id: ObjectId('657c89fc0e01e6c085a6d26a'),
    id: '2',
    cognoms: 'oller y moragas',
    nom: 'josep ma'
  },
  {
    _id: ObjectId('657c89fc0e01e6c085a6d26b'),
    id: '3',
    cognoms: 'carner i puig-oriol',
    nom: 'josep'
  },
  {
    _id: ObjectId('657c89fc0e01e6c085a6d26c'),
    id: '4',
    cognoms: 'català',
    nom: 'víctor'
  }
]
biblioteca> db.autors.deleteOne({"nom": "josep"})
{ acknowledged: true, deletedCount: 1 }
biblioteca>
```

db.<collection_name>.deleteMany(): Elimina tots els documents de la col·lecció <collection_name> que compleixin la condició . Per eliminar tots els documents de la colecció deixem la condició en blanc amb {}

Exemple: Eliminem tots els documents de la col·lecció 'autors' de la base de dades 'biblioteca'

```
biblioteca> db.autors.find()
[
  {
    _id: ObjectId('657c89410e01e6c085a6d269'),
    id: '1',
    cognoms: 'rodoreda i gurguí',
    nom: 'mercè'
  },
  {
    _id: ObjectId('657c89fc0e01e6c085a6d26a'),
    id: '2',
    cognoms: 'oller y moragas',
    nom: 'josep ma'
  },
  {
    _id: ObjectId('657c89fc0e01e6c085a6d26c'),
    id: '4',
    cognoms: 'català',
    nom: 'víctor'
  }
]
biblioteca> db.autors.deleteMany({})
{ acknowledged: true, deletedCount: 3 }
biblioteca> db.autors.find()

biblioteca>
```