
mongoDB®

Bases de dades relacionals

- Eren la forma més estesa per emmagatzemar informació fins que varen arribar altres bases de dades (distribuïdes, jeràrquiques…)
- La informació s'organitza en entitats anomenades *taules*. Cada taula s'organitza en atributs (columnes) i es van insertant registres (files).
- La informació de les taules solen estar relacionades -> BBDD relacionals
- Principis **ACID** (Atomicidad, Consistència, Aïllament, Durabilitat).
- Se segueix un esquema: conjunt de regles que contenen les taules que han de seguir la informació d'aquestes.
- En les BBDD relacionals s'emmagatzemen dades estructurades.

BBDD no relacionals

- Amb l'arribada d'Internet, suposen una alternativa al model clàssic de bases de dades SQL relacionals.

  Empreses com Google, Facebook, Amazon o Twitter comencen a utilitzar-les pel fet que no podien solucionar problemes de l'enfocament relacional (desestructuració de la informació, poca flexibilitat a causa dels esquemes relacionals,…).

- Són idònies quan les dades a emmagatzemar no segueixen una estructura fixa.

  Les BBDD no relacionals són una alternativa a les BBDD relacionals.

BBDD no relacionals

- Inconvenient: no garanteixen els principis ACID.

  (MongoDB des de la versió 4.2 sí que incorpora l'ús de les transaccions)

BBDD no relacionals

El terme noSQL es refereix a “no només SQL” per indicar que també poden suportar SQL. MongoDB suporta SQL.

Dins d’aquest tipus de bases de dades es troben:

- Els **documentals**: com es MongoDB, ja que es guarden documents JSON.
- Les **orientades a grafs**: es solen trobar en problemes relacionats amb l’estadística i, darrerament, amb Big Data.
- etc.

Com no segueixen cap esquema, milloren la flexibilitat i escalabilitat respecte a les bases de dades SQL.

MongoDB

- Orientat a documents (JSON)
- No se segueix cap esquema (shchemaless).
- Disposa d'una consola (shell) construïda sobre Javascript que permet executar múltiples funcions.
- Àmpliament utilitzada en aplicacions per Internet i Big Data.

SQL i noSQL - Avantatges

SQL	noSQL
Maduresa Bases de dades àmpliament utilitzades. Es manté la relació entre taules i les transaccions. Es considera el model clàssic. **ACID** **Estàndards ben definits** L'estàndard SQL ha anat madurant amb els anys i ha passat per diverses revisions. **Escriptura fàcil**	**Versatilitat o flexibilitat** **Baixa necessitat de recursos** pel que fa a servidors i aplicacions. **Optimització** L'usuari disposa d'una consola o shell per escriure les seves consultes i, en el cas de mongoDB, internament es fa una optimització prèvia quan s'executen aquestes consultes. Això fa que es millori la velocitat de les consultes.

SQL i noSQL - Inconvenients

SQL	noSQL
Manteniment difícil i costós **Poca flexibilitat davant a canvis en el disseny inicial.** És molt important dedicar-li molt de temps a la fase de disseny de la BD. Per no haver de canviar-lo en futures fases del projecte. **Complexitat i instal·lació** La seva complexitat en general i els seus processos d'instal·lació i configuració perquè siguin totalment funcionals.	**ACID** **Documentació del software** No sol ser massa bona. **No hi ha estàndard consensuat.** Les versions del JSON no solen ser estàndard. Això impedeix l'expansió més ràpida de l'ús de BDD noSQL.

Instal·lació de MongoDB

- Encara que MongoDB s'executi típicament en un clúster de màquines. Aquestes es troben al Cloud, per exemple: AmazonWS, Azure o Google Cloud.
- Per a la instal·lació en local, es descarrega gratuïtament a mongodb.com.

Col·leccions i Documents.

- En MongoDB la informació **no s'emmagatzema en registres dintre de taules**.
- En lloc de taules, s'utilitzen **col·leccions**. A diferència de les taules, les col·leccions no tenen uns camps predefinits.
- Així, les col·leccions no són més que un conjunt de documents sense una estructura predefinida, però que guarden informació sobre objectes similars.
- D'aquesta forma, els documents d'una col·lecció no tenen que contenir sempre els mateixos camps, sinó que existeix un alt grau de flexibilitat en la informació que emmagatzemen.
- Els documents s'escriuen en format JSON. Internament, s'emmagatzemen en format BSON.

Col·leccions i Documents

SQL	Taules	Files, tuples i registres	Columnes o Camps
noSQL	Col·leccions	Documents JSON	Propietats o Camps

¿Què és el format JSON?

- JSON: JavaScript Object Notation
- Format de text senzill per l'intercanvi de dades
- Alternativa a XML com llenguatge d'intercanvi de dades. JSON és molt més senzill de llegir i d'escriure.
- Suportat per multitud de llenguatges de programació.
- Un objecte JSON està format per un o varis parells string:value (cadena: valor). Exemple:

nom: pere

¿Què és el format JSON?

- Tipus de dades:

Simples	Complexes
Nombres (enters, decimals,...)	Array
Cadena de text	Objecte
Data	Binary data
Hora	objectId
Boolean (vertader o fals)	Expressions regulars

  - Subdocuments (Embedding)
  - Referències a altres Documents (Linking)

Exemple

```
{
    “títol” : “Game of thrones”,
    “autor” : “George R.R. Martin”,
    “any”   : 1997,
    “premis” : [“Premi Locus”, “Premi World Fantasy”, “Premi Hugo”, “Premi Nebula”, “Premi Ignotus”]
}
```

Comandes per treballar des de la consola

Esborrar contingut de la consola	**cls**
Sortir de la consola	**Ctrl+C**
Mostrar la base de dades	**show dbs**
Canvia de base de dades	**use <db_name>**
Llistar les col·leccions d'una base de dades	**show collections** / **show tables**
Mostrar el nom de la base de dades	**db.getName()** / **db**
Llistar les metadades d'una base de dades	**db.stats()**
Sol·licitar ajuda sobre les comandes	**db.help()**
Mostrar informació sobre el servidor	**db.hostInfo()**
Mostrar data i hora del sistema	**Date()**
Mostrar les dades d'una col·lecció i donar format JSON	**db.<collectionName>.find().pretty()**

Creació i gestió de bases de dades

Utilitzar o crear (si no existeix) una base de dades Nota: fins que no tingui col·leccions no es veurà reflectida a la llista de BBDD	**use <db_name>**
Crear una col·lecció i inserir un registre	**db.<collectionName>.insertOne(---)** Exemple: db.curs.insertOne({nom: “informatica”})
Eliminar base de dades Nota: Prèviament, hem de comprovar que ens trobem a la bd que volem eliminar amb la comanda **db**	**db.dropDatabase()**

Operacions amb dades (CRUD)

Insertar	Només un document **db.<collectionName>.insertOne(<json>);** Múltiples documents: **db.<collectionName>.insertMany(<json>); (array de documents [])**
Eliminar	Només un document: **db.<collectionName>.deleteOne(<json>);** *Per exemple*: db.alumnes.deleteOne({“id”:”1452365”}); Múltiples documents: **db.<collectionName>.deleteMany(<json>); (array de documents [])** *Per exemple*: db.alumnes.deleteMany({“es_titular”:false}); Si volem eliminar totes les dades: **db.<collectionName>.deleteMany({}); Alerta!**

Operacions amb dades (CRUD)

Actualizar	Només un document
	db.<collectionName>.updateOne(<filter>,<update>);
	
	Exemple:
	db.cursos.updateOne(
	{ referencia: “C0001”},
	{
	$set: {
	nom: “Administració de Bases de Dades”,
	duracio: 9
	}
	}
	);
	
	Múltiples documents:
	db.<collectionName>.updateMany(<filter>,<update>);
	
	Reemplaçar un document:
	db.Collection.replaceOne(<filter>,<update>):

Operacions amb dades (CRUD)

findOne()
 	
Llegir
	db.<collectionName>.find(<filter>);
	
	Exemples:
	
	db.cursos.find({ _id: 5});
	
	db.alumnes.find({data_naixement: { $gt: new Date('1995-01-01') } });

Exercicis

1) Crea la base de dades “Tenda”
2) Crea la col·lecció Usuaris i introdueix tres elements amb les següents característiques:

Nom d’usuari	Josep11	Claudia_	Pepe345
Nom	Josep	Claudia	Pere
Telèfon 1	685478596	641588412	970785465
Telèfon 2	68547563	656365852	
Domicili	Av. Patata 25	Carrer Tomate 26 Bj	Carrer Seta 35 A
CP	07658	78596	48626
Any de naixement	1990	1995	1973

Exercicis

3) Com estructuraries millor el document?

4) Afegeix un nou “usuari”:

Nom d’usuari	Mari
Nom	Marina
Telèfon 1	62228596
Domicili	Av. Lechuga 123
CP	07777
Any de naixement	1953

5) Retorna l’usuari amb el nom d’usuari: “Josep11”

6) Retorna els usuaris que varen néixer a l’any 1990.

Exercicis

7) Crea un document amb les següents característiques (pots posar els valors que trobis):

- DNI
- Nom
- Data de naixement
- Validat (boolean)
- Salari (nombre real)
- Propietats (casa, cotxe…)
- Mascotes (array d'objectes amb nom, tipus de mascota i edat)

Consultes: condicions - Operadors lògics

AND: Ambdues condicions s'han de complir.

db.socis.find({$and: [{nom: "Juan"}, {llinatge: "Martín"}]})

Mostra els documents on el nom sigui "Juan" i el llinatge sigui "Martín".

OR: Alguna de les condicions s'ha de complir.

db.socis.find({$or: [{nom: "Juan"}, {llinatge: "Martín"}]})

Mostra els documents on el nom sigui Juan o el llinatge sigui "Martín" (o les dues condicions).

NOT: No ha de cumplir la condició.

el mateix faríem amb $ne

db.socis.find({numSoci: { $not: {$eq: 300} }})

Mostra els documents on el nombre de soci no sigui 300.

Consultes: condicions - Operadors lògics

NOR: Que no es compleixi cap condició.

db.socis.find({$nor: [{nom: “Juan”}, {llinatge: “Martín”}]})

Mostra els documents on el nom no sigui “Juan” i el llinatge no sigui “Martín”.

OPERADORS LÒGICS SOBRE CAMPS

EXISTS: existeix el camp.

db.socis.find({ llinatge2: { $exists: true } })

Mostra els documents que tenguin el camp “llinatge2”.

TYPE: comprovació del tipus de camp.

db.socis.find({ llinatge2: { $type: “string”} })

Mostra els documents que on llinatge2 sigui del tipus String.

Consultes: condicions - Operadors lògics

OPERADORS LÒGICS SOBRE ARRAYS

- **ALL:** l'array conté tots els valors
- **ElemMatch:** algun valor de l'array compleix les condicions
- **SIZE:** l'array és d'un tamany concret

```
db.series.find({ tags: {$all: ["historia", "aventura"] } })
```

```
db.series.find({puntuacion: {$elemMatch: {$gte: 2, $lt: 5 }}})
```

```
db.series.find({ tags: {$size: 3} })
```

Consultes: condicions i projeccions

- **Valor 1:** si volem mostrar un camp
- **Valor 0**: si no volem mostrar-ho
- Per defecte, si no es declara res a la projecció, es mostren tots els camps.
  - El camp **_id** sempre es mostra per defecte encara que hi hagi altres camps en la projecció amb valor 1.
  - Si no es vol mostrar el camp **_id**, s'ha de posar a 0 explícitament en la projecció.

Consultes: condicions i projeccions

Exemples:

db.socis.find({numSoci: {$eq: 300} }, { nom: 1, llinatge1: 1 })

Mostra el nom, el llinatge1 i _id

db.socis.find({numSoci: {$eq: 300} }, { nom: 1, llinatge1: 1, _id: 0 })

Mostra el nom i el llinatge1

db.socis.find({numSoci: {$eq: 300} }, { numSoci: 0 })

Mostra tots els camps excepte numSoci

Consultes: modificadors

- **Sort:** ordena els resultats segons els camps donats. Un valor de 1 indica ordre ascendent, i un valor de -1 ordre descendent:

```
db.socis.find({puntuacion: {$eq: 1000} }, { nom: 1, llinatge1: 1 }).sort({nom:1})
```

- **Limit:** limita el nombre de documents retornats. El 0 indica que no hi ha límit.

```
db.socis.find({nom: {$eq:”Juan} }, { nom: 1, llinatge1: 1 }).limit(3)
```

Consultes: agrupament

- **Count:** conta el nombre de vegades que apareix en els documents d'una col·lecció un valor en un camp.

Suposem que a la col·lecció "series" tenim un camp anomenat puntuació i els següents documents:

{puntuacio: 5}, {puntuacio: 3}, {puntuacio: 5}, {puntuacio: 5}, {puntuacio: 2}

Tinguent en compte això:

db.series.count({puntuacio: 5}) retornaria 3.

Consultes: agrupament

- **Distinct:** retorna un vector amb els diferents valors (sense repetició) que té un camp concret en els documents d'una col·lecció.

  Exemple: suposant que en una col·lecció anomenada

Suposem que a la col·lecció "series" tenim un camp anomenat puntuació i els següents documents:

{puntuacio: 5}, {puntuacio: 3}, {puntuacio: 5}, {puntuacio: 5}, {puntuacio: 2}

Tinguent en compte això:

db.series.distinct("puntuacio") retornaria l'array [5,3,2].

Consultes: agrupament

https://docs.mongodb.com/manual/reference/operator/aggregation/group/

Per realitzar agrupaments, també podem utilitzar **aggregate()**. Aquest mètode processa múltiples documents i es pot fer servir per agrupar per camps, executar operacions segons els agrupaments i analitzar els canvis de les dades.

Aquí tens alguns exemples d'ús d'aquest mètode:

```
db.series.aggregate({
        $group: {
                _id: '$tipus',
                nombre: { $count: {} }
        }
}):
```

Agrupa els documents segons el camp 'tipus' i compte el nombre de documents per grup, aquest valor el posarà al camp 'tipus'

Consultes: agrupament

```
db.series.aggregate({
        $group: {
                _id: ‘$tipus’,
                puntuacio_mitjana: { $avg: ‘$puntuacio’ },
        }
}):
```

Agrupa els documents segons el camp ‘tipus’ i mostra la puntuació mitjana de cada grup al camp puntuacio_mitjana.

Consultes: agrupament

```
db.series.aggregate([
        {
                $sort: { puntuacio: -1 }
        },
        {
                $group: {
                        _id: ‘$tipus’,
                        nom: { $first: ‘$nom’ },
                        puntuacio: { $first: ‘$puntuacio’ }
                }
        }
]):
```

Volem saber el nom i puntuació del document amb millor puntuació de cada tipus.

Exercicis

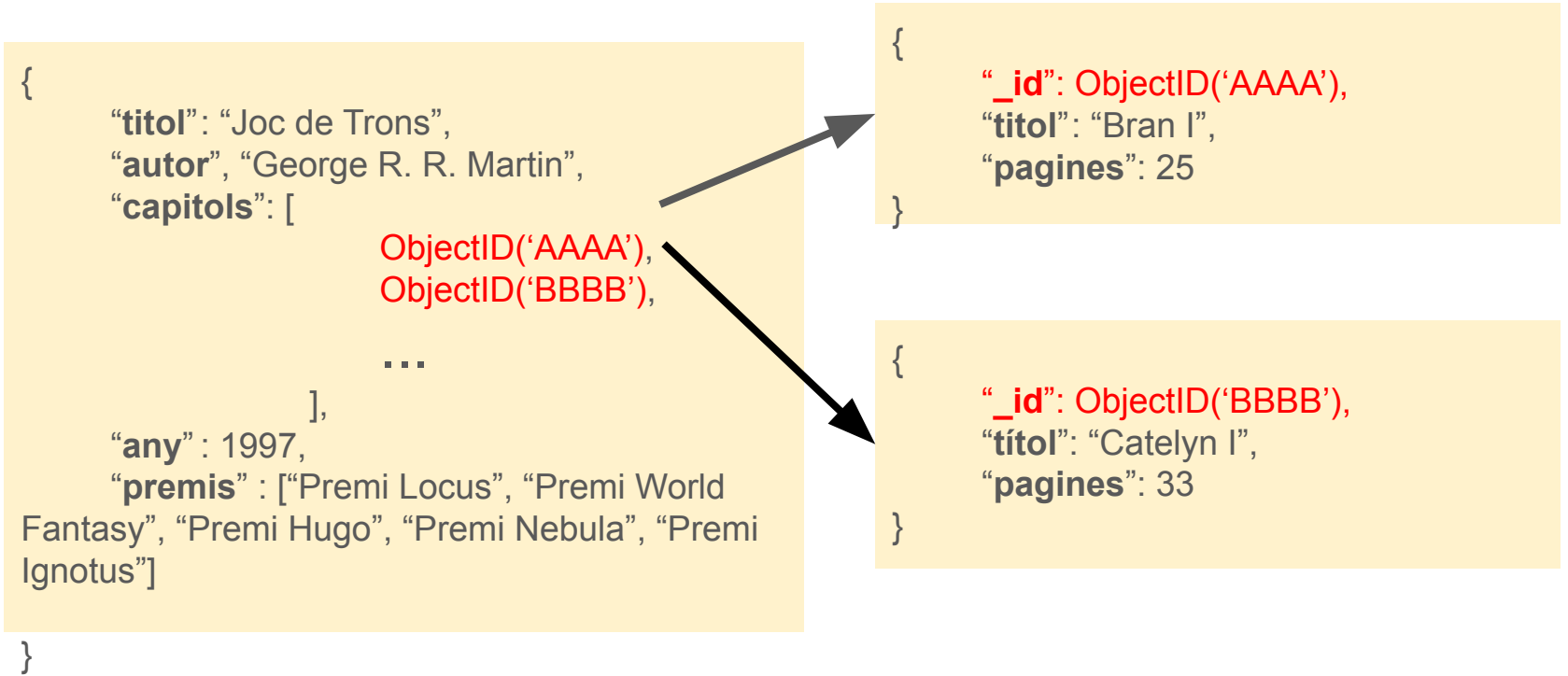
Documents incrustats (Embedding)

```
{
        “titol”: “Joc de Trons”,
        “autor”, “George R. R. Martin”,
        “capitols”: [
                        {
                                “titol”: “Bran I”,
                                “pagines”: 25
                        },
                        {
                                “titol” “Catelyn I”,
                                “pagines”: 33
                        }
                ],
        “any” : 1997,
        “premis” : [“Premi Locus”, “Premi World Fantasy”, “Premi Hugo”, “Premi Nebula”, “Premi Ignotus”]

}
```

Documents incrustats (Embedding)

AVANTATGES	INCONVENIENTS
- Obtenim totes les dades en una única consulta. - Evita JOINs costosos. - Actualitza totes les dades amb una única operació atòmica (llegeix i escriu a la mateixa operació).	- Documents de mida més gran. - Límit de 16 MB per document. Quan més gran més costoses seran les operacions. Hem d'intentar que els documents tinguin la mida més petita possible.

Documents referenciats



Documents referenciats

AVANTATGES	INCONVENIENTS
- Els documents són més petits. - És més difícil arribar al límit de 16 MB. - No hi ha duplicació de dades. - Les dades amb menys accessos no s'han de mostrar a cada consulta.	- Fan falta internament més consultes per obtenir les dades desitjades.

Embedding or Refferencing?

1) Sempre afavoreix els documents incrustats a menys que existeixi una bona raó per no fer-ho.
2) L'accés a la informació de documents (no dels seus incrustats) és una raó per no utilitzar els documents incrustats.
3) Evita JOINs, però els pots utilitzar si tens un bon disseny de l'esquema.

Tipus de relacions: One to One

```
{
        “_id”: ObjectId(“AA”),
        “titol”: “Joc de Trons”,
        “any” : 1997,
        “autor”: “George R.R. Martin”,
        “premis” : [“Premi Locus”, “Premi World Fantasy”, “Premi Hugo”, “Premi Nebula”, “Premi Ignotus”]
}
```

Tipus de relacions: One to Few

```
{
        "_id": ObjectId("AA"),
        "titol": "Joc de Trons",
        "autor", "George R. R. Martin",
        "llibres": [
                        {
                                "titol": "Joc de Trons",
                                "any": 1996
                        },
                        {
                                "titol" "Xoc de reis",
                                "any": 1998,
                        }, ...
                ]
}
```

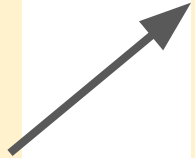
Tipus de relacions: One to Many

```
{
        “_id”: ObjectId(“AA”),
        “titol” : “Joc de Trons”,
        “autor”, “George R. R. Martin”,
        “capitols”: [
                        {
                                “titol”: “Bran I”,
                                “pagines”: 25,
                                “contingut”, “...”
                        },
                        {
                                “titol” “Catelyn I”,
                                “pagines”: 33,
                                “contingut”: “...”
                        }
                ]
}
```

Hem de tenir en compte que si volem afegir els continguts de cada capítol, el document serà molt gran. En aquest cas, és interessant utilitzar els documents referenciats.

Tipus de relacions: One to Many

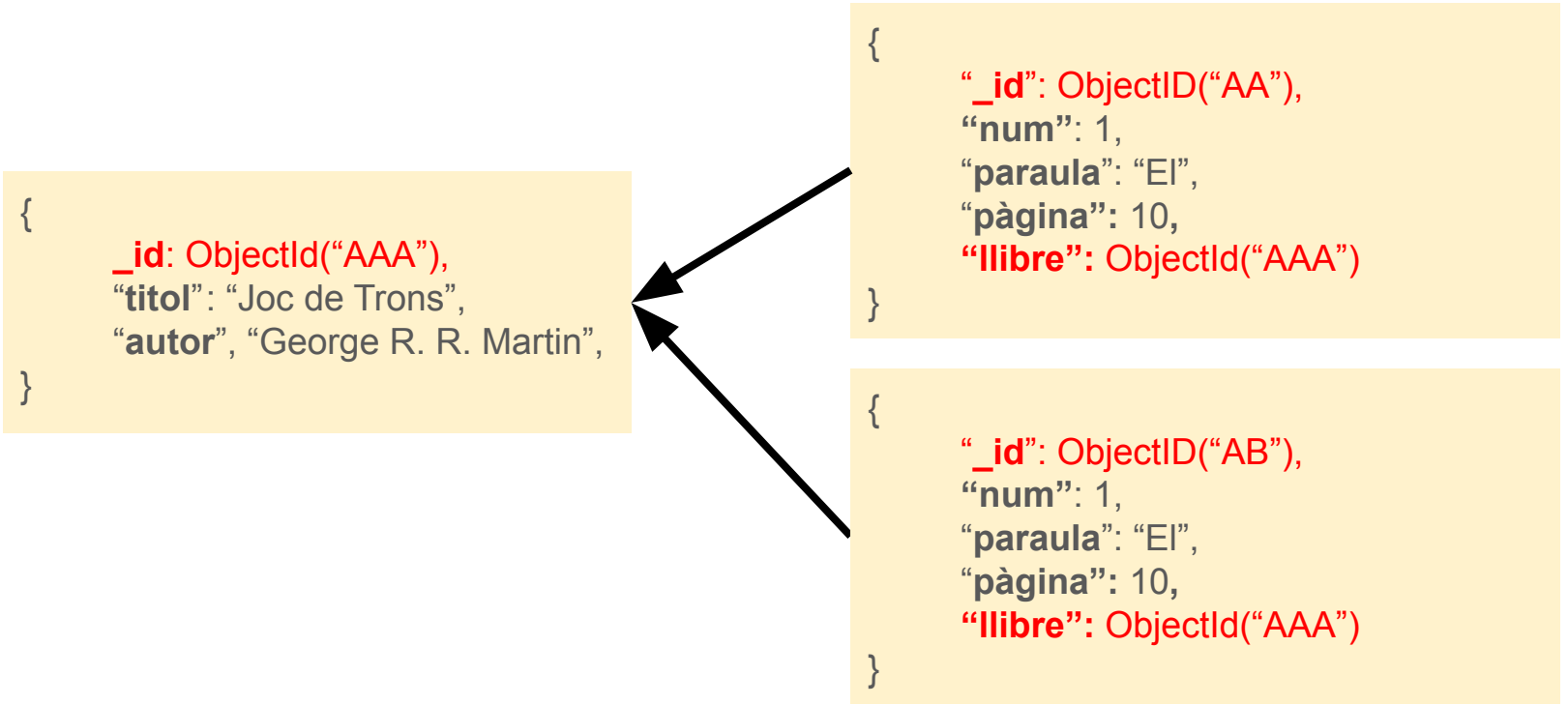
```
{
        “_id”: ObjectId(“AAA”),
        “titol”: “Joc de Trons”,
        “autor”, “George R. R. Martin”,
        “capitols”: [
                        ObjectID(‘AAAA’),
                        ObjectID(‘BBBB’),

                        …
                ],
}
```



```
{
        “_id”: ObjectID(‘AAAA’),
        “titol”: “Bran I”,
        “pagines”: 25,
        “contingut”: “...”
}
```

```
{
        “_id”: ObjectID(‘BBBB’),
        “títol”: “Catelyn I”,
        “pagines”: 33
        “contingut”: “...”
}
```

Tipus de relacions: One to Nombre MOLT elevat

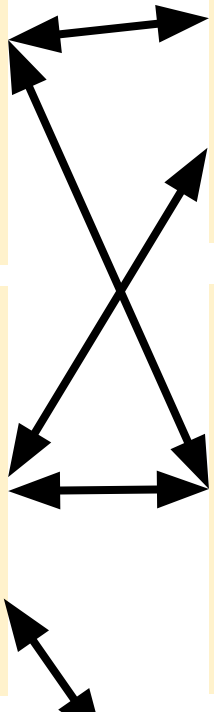


Tipus de relacions: Many to Many

```
{
        _id: ObjectId(“AAA”),
        “nom”: “Pep Lluís”,
        “tasques”: [
                ObjectId(“AB1”),
                ObjectId(“AB2”),
        }
}
```

```
{
        “_id”: ObjectID(‘AB1’),
        “descripcio”: “Aprendre MongoDB”,,
        “data”: ISODate(“2021-02-28T09:42:41.382Z”),
        “persones”: [
                ObjectId(“AAA”),
                OjectId(“BBB”)
        ]
}
```

```
{
        _id: ObjectId(“AAA”),
        “nom”: “Junior”,
        “tasques”: [
                ObjectId(“AB1”),
                ObjectId(“AB2”),
                ObjectId(“AB3”)
        }
}
```



```
{
        “_id”: ObjectID(‘AB2’),
        “descripcio”: “Entregar el projecte”,,
        “data”: ISODate(“2021-02-20T09:42:41.382Z”),
        “persones”: [
                ObjectId(“AAA”),
                OjectId(“BBB”)
        ]
}
```

Tipus de relacions: Preferència

One to One: Parells Clau-Valor

One to Few: Embedding

One to Many: Referencing

One to Nombre Molt Elevat: Referencing (de “documents fills” a “pare”)

Many to Many: Referencing (recíproc)