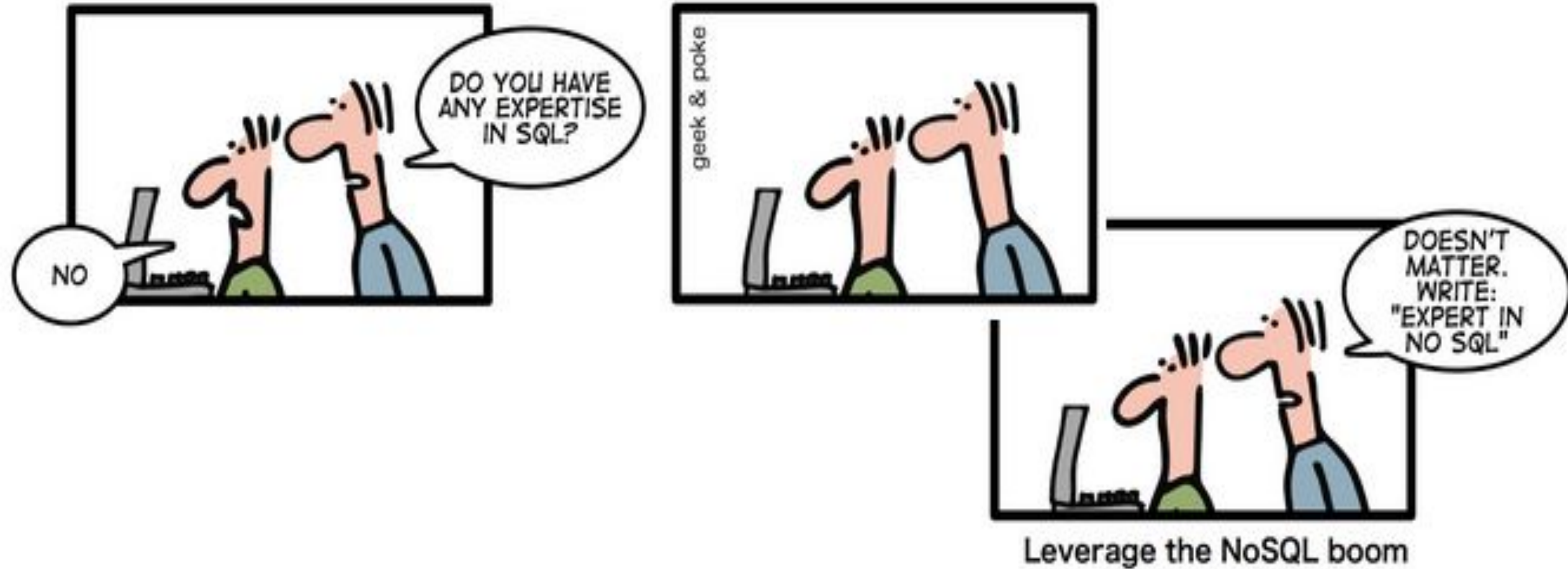




mongoDB



Temario

- Repaso-introducción MongoDB
- MongoDB Atlas
- Práctica Consultas `.find(..)`
- Pipeline de agregación
`.aggregate([..])`
- ABM

Introducción

Modelo de datos

Relacional

C1	C2	C3	C4
—	—	—	—
—	—	—	—
—	—	—	—
—	—	—	—

- Tablas estructuradas
- Formato de datos definidos y rígidos

V
S

Documentos



- Conjunto de documentos
- Formato de datos arbitrarios y anidados

SQL

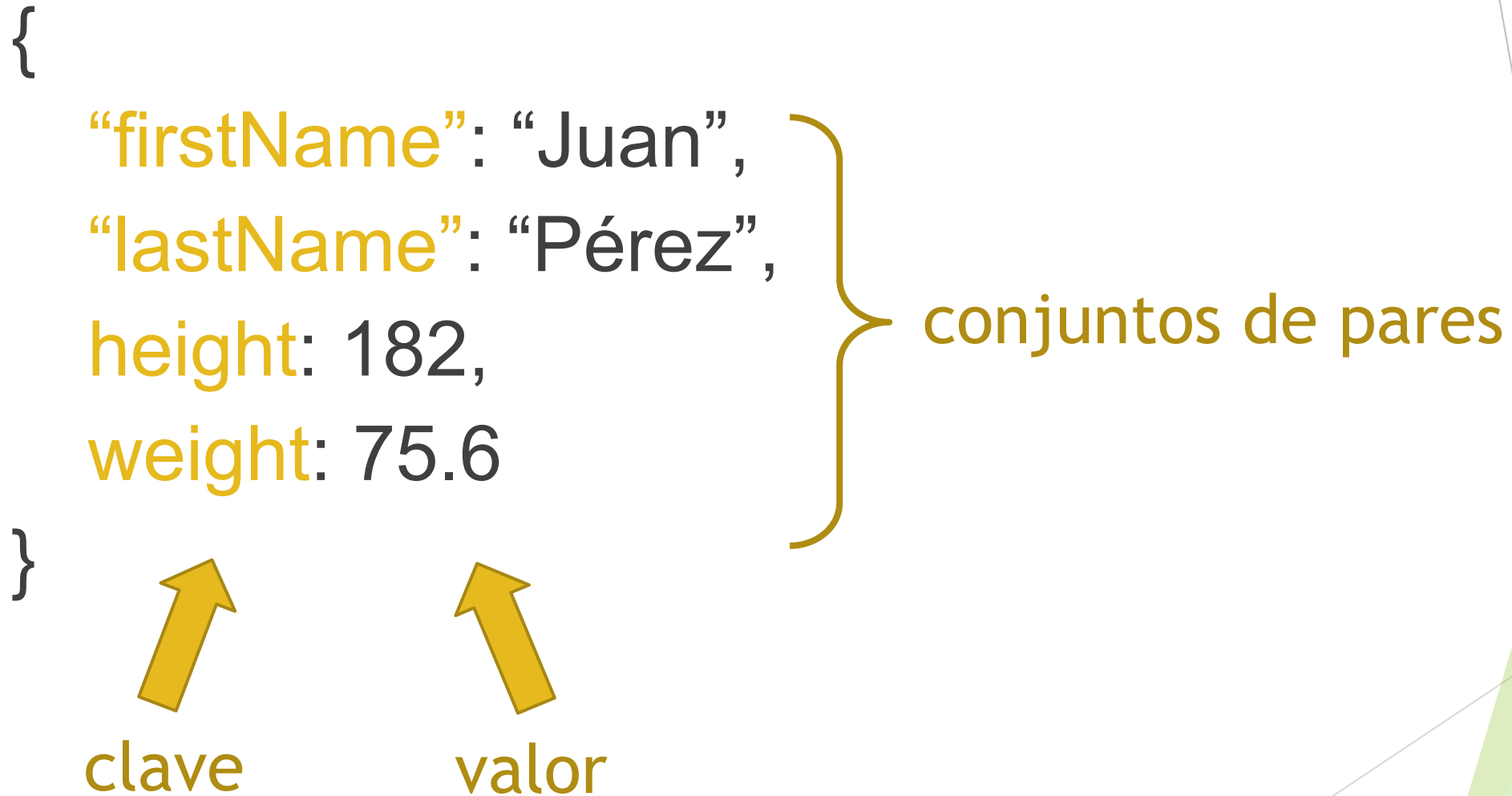
JSON (JavaScript Object Notation)

```
{  
  "firstName": "Juan",  
  "lastName": "Pérez",  
  height: 182,  
  weight: 75.6  
}
```

conjuntos de pares

↑
clave

↑
valor

A diagram illustrating the structure of a JSON object. The JSON object is shown as a set of key-value pairs enclosed in curly braces. A large curly brace on the right groups the key-value pairs and is labeled "conjuntos de pares". Two yellow arrows point to the keys "firstName" and "lastName" from the labels "clave" and "valor" respectively, located below the JSON object.

JSON

```
{
  id_: ObjectId("0123456789") ← string
  created_at: ISODate("2019-06-26T00:00:00.000Z"), ← string
  text: "Tweet que no existe.", ← string
  user_id: "102510",
  is_quote_status: false, ← bool
  retweet_count: 0, ← número (entero o decimal)
  display_text_range: [
    {...} ← array
  ],
  entities: {
    hashtags: [], ← objeto anidado
    ...
  },
  in_reply_to_status_id: null ← valores nulos (≠ undefined)
}
```

BSON (Binary JSON)

```
{
  id_: ObjectId("0123456789") ← string
  created_at: ISODate("2019-06-26T00:00:00.000Z"), ← string
  text: "Tweet que no existe.", ← string
  user_id: "102510",
  is_quote_status: false, ← bool
  retweet_count: 0, ← número (entero o decimal)
  display_text_range: [
    {...} ← array
  ],
  entities: {
    hashtags: [], ← objeto anidado
    ...
  },
  in_reply_to_status_id: null ← valores nulos
}
```


Terminología

SQL

database (schema)

table

row

column

join

MongoDB

database

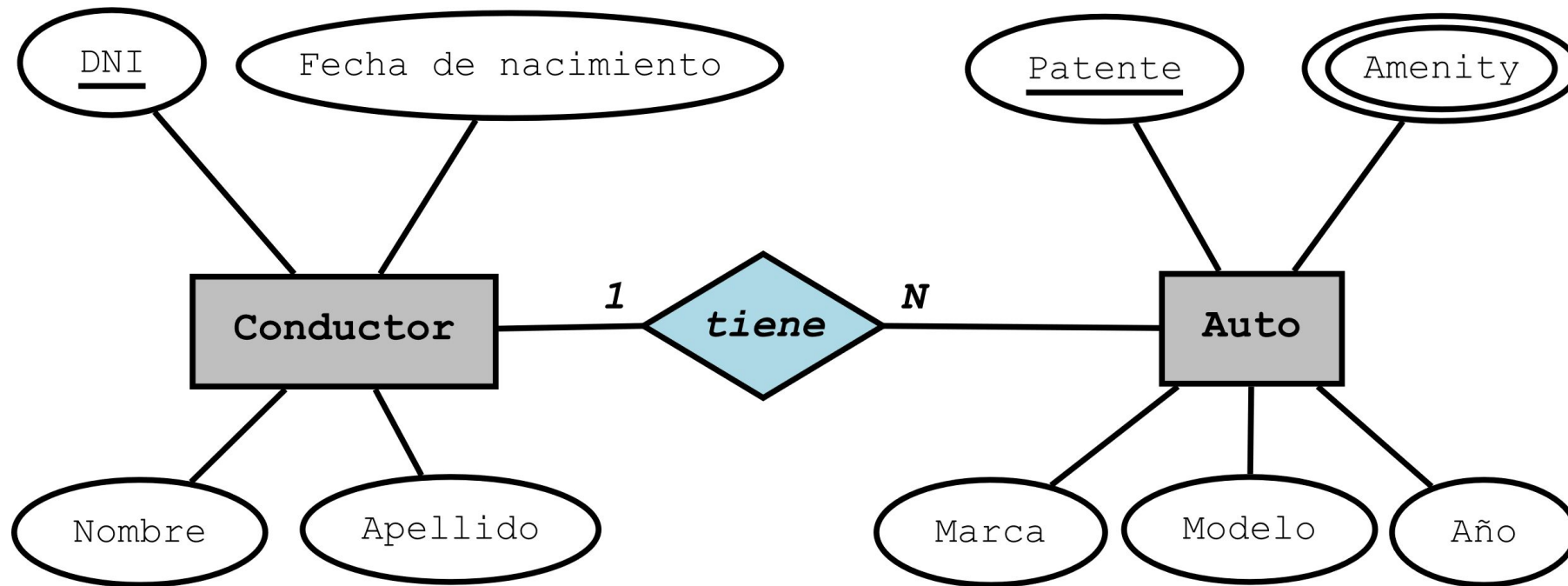
collection

document

field

embedded doc

Modelo de ejemplo



SQL: crear una tabla

```
CREATE TABLE conductor (  
    DNI integer primary key,  
    fecha_de_nacimiento date,  
    nombre varchar,  
    apellido varchar  
);
```

```
CREATE TABLE auto (  
    patente varchar primary key,  
    amenity varchar[],  
    marca varchar,  
    modelo varchar,  
    año integer,  
    DNI integer references conductor(DNI)  
);
```

MongoDB: crear una colección

```
db.createCollection("conductores");
```

```
db.createCollection("autos");
```

Listo

Schema-less

SQL: tablas

DNI	nombre	apellido	fecha_de_nacimiento
30123456	Juan	Pérez	1980-05-09
21222324	Graciela	Lopez	1964-12-01

DNI	patente	amenity	marca	modelo	año
21222324	BG 135 ZT	["airbag", "frenos ABS", "faros antiniebla"]	Toyota	Etios	2019
30123456	AA 937 BR	[]	Fiat	Uno	2008
21222324	AH 882 KS	["airbag", "frenos ABS"]	Ford	Ka	2017

MongoDB: datos

Colección **conductores**

```
{
  "_id": 30123456,
  "fecha_de_nacimiento":
1980-05-09,
  "nombre": "Juan",
  "apellido": "Pérez"
},
{
  "_id": 21222324,
  "fecha_de_nacimiento":
1964-12-01,
  "nombre": "Graciela",
  "apellido": "López"
}
```

Colección **autos**

```
{
  "_id": "BG 135 ZT",
  "amenities": ["airbag", "frenos ABS", "faros antiniebla"],
  "marca": "Toyota",
  "modelo": "Etios",
  "año": 2019,
  "dueño": 21222324
},
{
  "_id": "AA 937 BR",
  "amenities": [],
  "marca": "Fiat",
  "modelo": "Uno",
  "año": 2008,
  "dueño": 30123456
},
{
  "_id": "AH 852 KS",
  "amenities": ["airbag", "frenos ABS"],
  "marca": "Ford",
  "modelo": "Ka",
  "año": 2017,
  "dueño": 21222324
}
```

No existe
el JOIN

MongoDB: datos

Colección **conductores**

```
{
  "_id": 30123456,
  "fecha_de_nacimiento":
1980-05-09,
  "nombre": "Juan",
  "apellido": "Pérez"
},
{
  "_id": 21222324,
  "fecha_de_nacimiento":
1964-12-01,
  "nombre": "Graciela",
  "apellido": "López"
}
```

(queremos guardar esto?)

Colección **autos**

```
{
  "_id": "BG 135 ZT",
  ...
  "dueño": {
    "_id": 21222324,
    "fecha_de_nacimiento": 1964-12-01,
    "nombre": "Graciela",
    "apellido": "López"
  }
},
{
  "_id": "AA 937 BR",
  ...
  "dueño": {
    "DNI": 30123456,
    "fecha_de_nacimiento": 1980-05-09,
    "nombre": "Juan",
    "apellido": "Pérez"
  }
}
```

SQL: modificaciones

DNI	nombre	apellido	fecha_de_nacimiento
30123456	Juan	Pérez	1980-05-09
21222324	Graciela	ALTER TABLE ...;	

DNI	patente	amenity	marca	modelo	año	infracciones
21222324	BG 135 ZT	["airbag", "frenos ABS", "faros antiniebla"]	Toyota	Etios	2019	...
30123456	AA 937 BR	[]	Fiat	Uno	2008	...
21222324	AH 882 KS	["airbag", "frenos ABS"]	Ford	Ka	2017	...

MongoDB: datos

Colección **conductores**

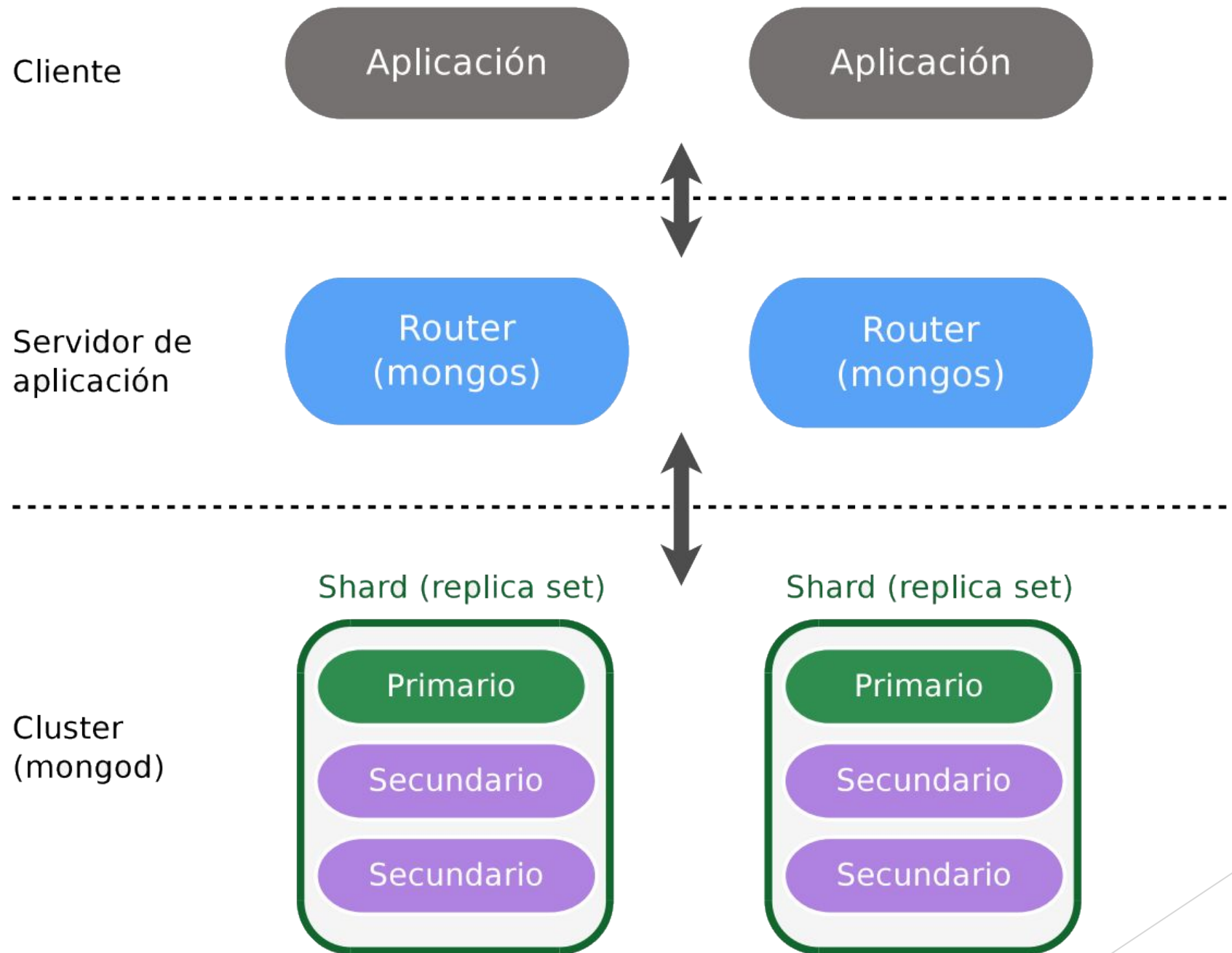
```
{
  "_id": 30123456,
  "fecha_de_nacimiento":
1980-05-09,
  "nombre": "Juan",
  "apellido": "Pérez"
},
{
  "_id": 21222324,
  "fecha_de_nacimiento":
1964-12-01,
  "nombre": "Graciela",
  "apellido": "López"
}
```

db.autos.update(...)

Colección **autos**

```
{
  "_id": "BG 135 ZT",
  ...
  "dueño": {
    ...
  },
  infracciones: [...]
},
{
  "_id": "AA 937 BR",
  ...
  "dueño": {
    ...
  },
  infracciones: [...]
}
```

MongoDB Atlas



Set de Datos: Objetos Tweet



```
{  
  ...  
  _id: str[19],  
  created_at: ISODate,  
  entities: {...},  
  user: {...},  
  user_id: str,  
  retweet_count: int,  
  ...  
}
```

Consultas

```
db.<collection>  
  .find(<query>, <projection>)  
  .sort(<order>)  
  .limit(n)
```

Limitar los documentos

□ Notación

```
db.<collection>.find().limit(<n>)
```

□ Ejemplo

```
db.tweets.find().limit(5)
```

Buscar documentos

❑ Notación

```
db.<collection>.find( <query>, <projection> )
```

❑ Ejemplos: igualdad

```
db.tweets.find(  
  { _id: ObjectId("1143670209296785408") }  
)
```

```
db.user.find(  
  { _id: "1143670209296785408" },  
  { _id: 0, text: 1 }  
)
```

Buscar documentos

❑ Ejemplo: **and**, **or**

```
db.tweets.find({ $or: [  
  { user_id: "Juan" },  
  { retweet_count: 2183 }  
] })
```

```
db.tweets.find({ $and: [  
  {user_id: "818839458"},  
  {retweet_count: 2183}  
] })
```

```
db.tweets.find({ user_id: "Juan", retweet_count: 2183 })
```

❑ Ejemplos: **eq** (**==**), **gt** (**>**), **gte** (**>=**), **lt** (**<**), **lte** (**<=**)

```
db.tweets.find({ retweet_count: { $gte: 2183 } })
```


Buscar documentos

❑ Ejemplos: documento embebido

```
db.tweets.find(  
  {"user.name": "Juan"},  
  {"name": "$user.name"}  
)
```

```
db.tweets.find(  
  {"retweet_count": { $gt: "$fa" }  
)
```

Buscar documentos

□ Ejemplos: **regex**

```
db.tweets.find( { "user.name": /Juan/ } )
```

```
db.tweets.find( { "user.name": { $regex: "Juan" } } )
```

```
db.tweets.find( { "user.screen_name": /juan/i } )
```

□ Ejemplo: **in, nin**

```
db.tweets.find( { "user.name": { $in: [ "Juan", "Laura" ] } } )
```

```
db.tweets.find( { "user.name": { $nin: [ "Pablo" ] } } )
```

Buscar documentos (arrays)

❑ Ejemplo: array: **alguno**

```
db.tweets.find( { activities: "swimming" } )
db.tweets.find( { "entities.hashtags":
  { $elemMatch:
    { "text": { $in: [ 'futbol', 'copaamerica' ] } }
  }
} )
```

❑ Ejemplo: array: **múltiples**

```
db.tweets.find( { activities:
  { $all: [ "swimming", "boxing" ] }
} )
```

❑ Ejemplo: array: **tamaño**

```
db.tweets.find( { activities: { $size: 3 } } )
```

Cursor: contar los documentos

❑ Notación

```
db.<collection>.find().count()
```

❑ Ejemplo

```
db.tweets.find().count()
```

Cursor: ordenar los documentos

❑ Notación

```
db.<collection>.find().sort( {fieldName: order} )
```

❑ Ejemplos

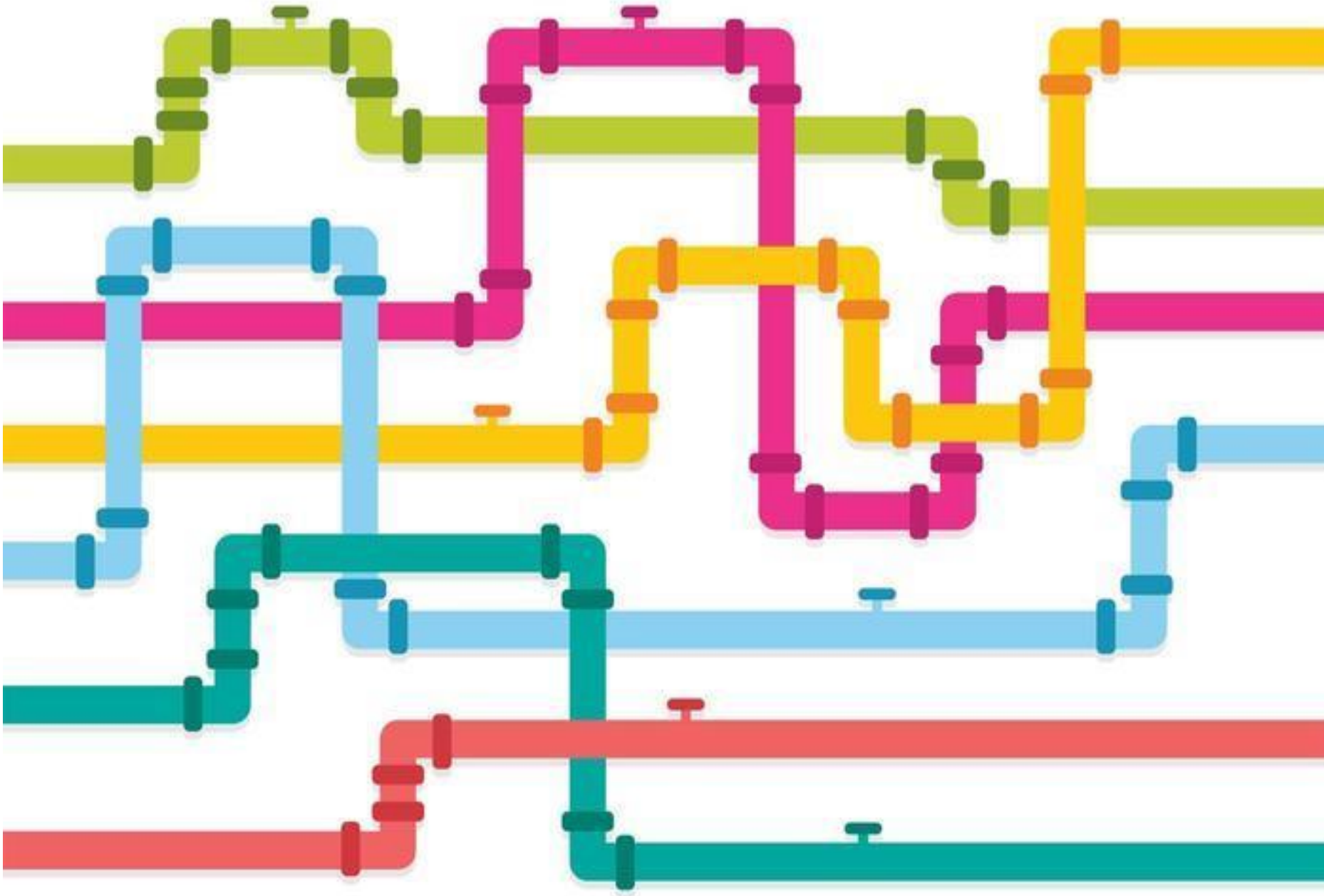
```
db.tweets.find().sort( {created_at: 1} )
```

```
db.tweets.find().sort( {retweet_count: -1} )
```

Agregación

```
db.<collection>  
  .aggregate([  
    <filter>,  
    ...  
  ])
```

Agregación (Aggregation pipeline)



Agregación

□ Notación

```
db.collection.aggregate (
```

```
[
```

```
{ $stage1 },
```

```
{ $stage2 },
```

```
...
```

```
{ $stageN },
```



agregation pipeline

```
]
```

```
)
```


Agregación - Etapa

❑ **\$match**

Permite realizar una query al igual que `find()`

❑ Notación

```
{ $match: { <query> } }
```

❑ Ejemplo

```
db.tweets.aggregate( [ { $match: { "user.name": "Juan" } } ] )
```

```
db.tweets.aggregate( [ { $match: {  
    "entities.hashtags.text": {$eq: "futbol"}  
}} ] )
```

Agregación - Etapa

❑ **\$project**

Permite elegir los campos al igual que la fase de proyección en `find(<query>, <project>)`

❑ Notación

```
{ $project: { <field>: <1, 0 or expression> ... } }
```

❑ Ejemplos

```
db.tweets.aggregate( [ { $project: { user: 1 } } ] )
```

Agregación - Etapa

❑ **\$project / \$addFields**

Además, permite crear *campos nuevos*

❑ Notación

```
{ $project: { <field>: <1, 0 or expression> ... } }  
{ $addFields: { <field>: <expression> ... } }
```

❑ Ejemplos

```
db.tweets.aggregate( [ { $project: { hashtag_count:  
    {$size: "$entities.hashtags" }  
} ] )
```

```
db.tweets.aggregate( [ { $addFields: { hashtag_count:  
    {$size: "$entities.hashtags" }  
} ] )
```

Agregación - Etapa

❑ **\$limit**

Limita la cantidad de documentos a la próxima etapa

❑ Ejemplo

```
db.tweets.aggregate(  
  [  
    { $limit: 1 }  
  ]  
)
```

Agregación - Etapa

❑ **\$skip**

Se saltea la cantidad de documentos especificada

❑ Ejemplo

```
db.tweets.aggregate(  
  [  
    { $skip: 32 }  
  ]  
)
```

Agregación - Etapa

❑ **\$sort**

Ordena por el campo indicado

❑ Ejemplo

```
db.country.aggregate(  
  [  
    { $sort: { area: -1 } }  
  ]  
)
```

Agregación - Etapa

❑ **\$group**

Agrupa documentos por el campo deseado

❑ Notación

```
db.country.aggregate([  
  { $group: {  
    _id: <expression>,  
    <field>: <expression>  
  }}  
])
```

Agregación - Operador

❏ acumuladores

```
db.country.aggregate( [ { $group: {  
  _id: "$user.name",  
  avg_retweeters: {  
    $avg: "$retweet_count"  
  }  
} ] )
```

Ídem con: **sum, max, min, first, last**

Agregación - Etapa

□ **\$unwind**

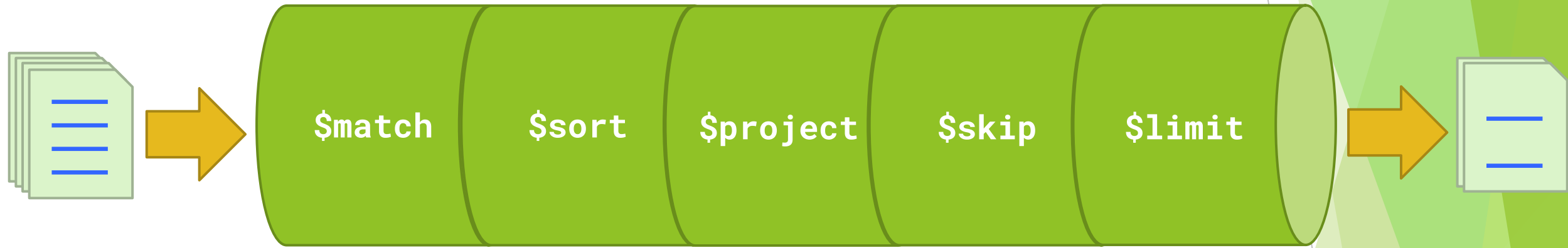
“Deconstruye” una lista de elementos en sus elementos individuales

□ Notación

```
db.country.aggregate([  
    { $unwind: <field> }  
])
```

Agregación - Etapa

□ *ejemplo*



ABM

Insertar documentos

□ Notación

```
db.<collection>.insertOne( <document> )  
db.<collection>.insertMany(  
    [ <document1>, <document2>, ... ]  
)
```

□ Ejemplo

```
db.user.insertOne(  
    {user_id: "999999", ...}  
) -> ID
```

Actualizar documentos

❑ Notación

```
db.<collection>.updateOne(  
    <filter>, <update>, {upsert: <boolean>}  
)
```

```
db.<collection>.updateMany(  
    <filter>, <update>, {upsert: <boolean>}  
)
```

❑ Ejemplos

```
db.user.updateOne(  
    {"user.name": /Juan/i}, <documento_completo>  
)
```

Actualizar documentos

❑ Ejemplos: **set**, **unset**

- ❑ Agrega o elimina un campo, sin afectar los otros del documento

```
db.tweets.updateOne(  
  {id_: ID},  
  {$set: {is_fake: true}}  
)  
db.tweets.updateMany(  
  {is_fake: true},  
  {  
    $set: {is_real: false},  
    $unset: {is_fake: 1}  
  }  
)
```

Actualizar documentos

□ Ejemplos: **inc**

- Modifica un campo numérico en la cantidad deseada

```
db.tweets.updateOne(  
  {id_: ID},  
  {$inc: {retweet_count: 10}}  
)
```

Eliminar documentos

□ Notación

```
db.<collection>.deleteOne( <filter> )
```

```
db.<collection>.deleteMany( <filter> )
```

□ Ejemplos

```
db.tweets.deleteOne(  
  {is_real: false}  
)
```