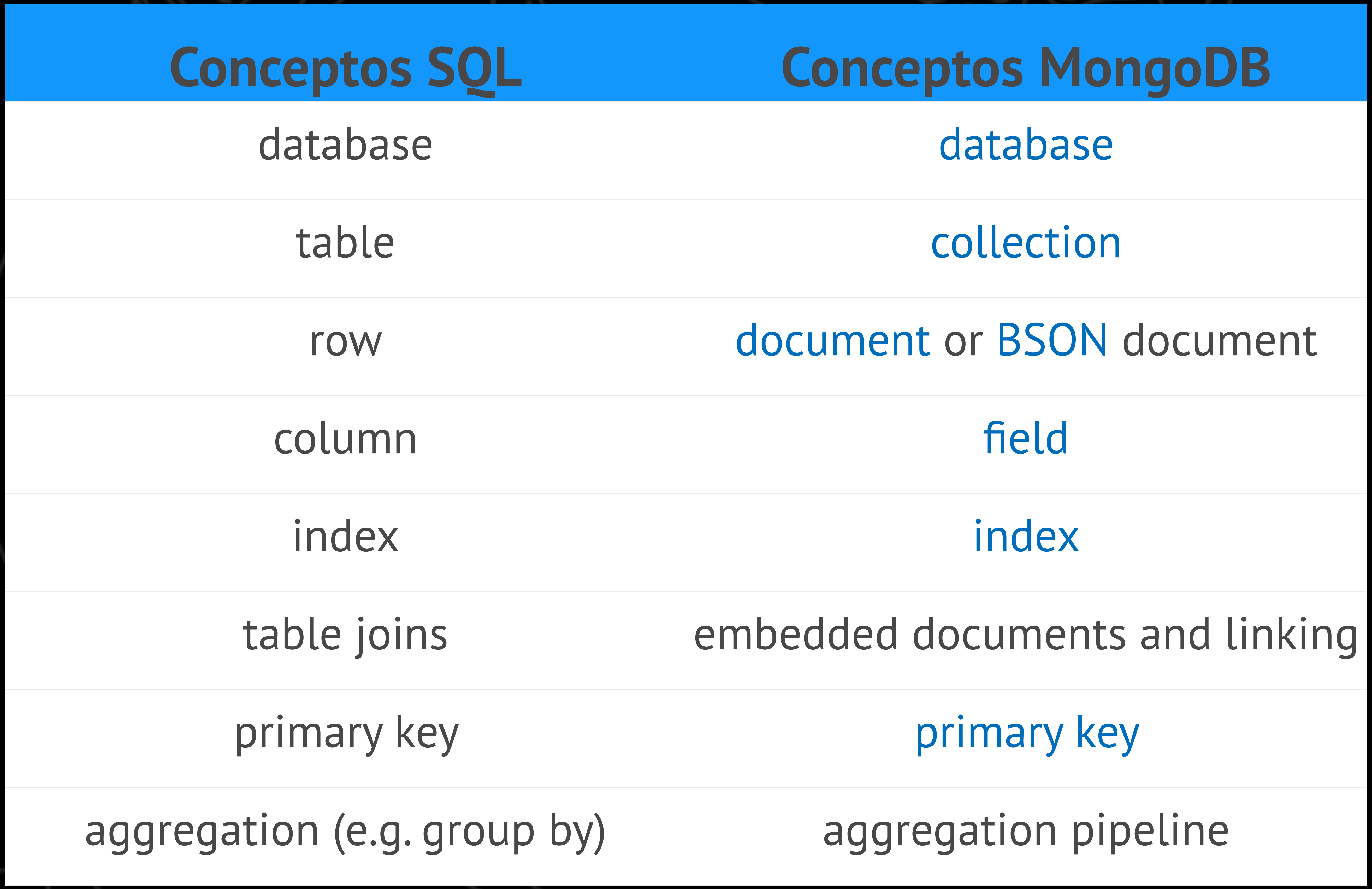
Curso de MongoDB

Unidad 03: Introducción al Modelo CRUD

Índice de contenidos

- Mapeo de consultas SQL a MongoDB
- Consultas básicas y búsquedas de información
- Operaciones de lectura
- Operaciones de escritura
- Ejemplo en Java

Mapeo de consultas SQL a MongoDB



Conceptos SQL	Conceptos MongoDB
database	database
table	collection
row	document or BSON document
column	field
index	index
table joins	embedded documents and linking
primary key	primary key
aggregation (e.g. group by)	aggregation pipeline

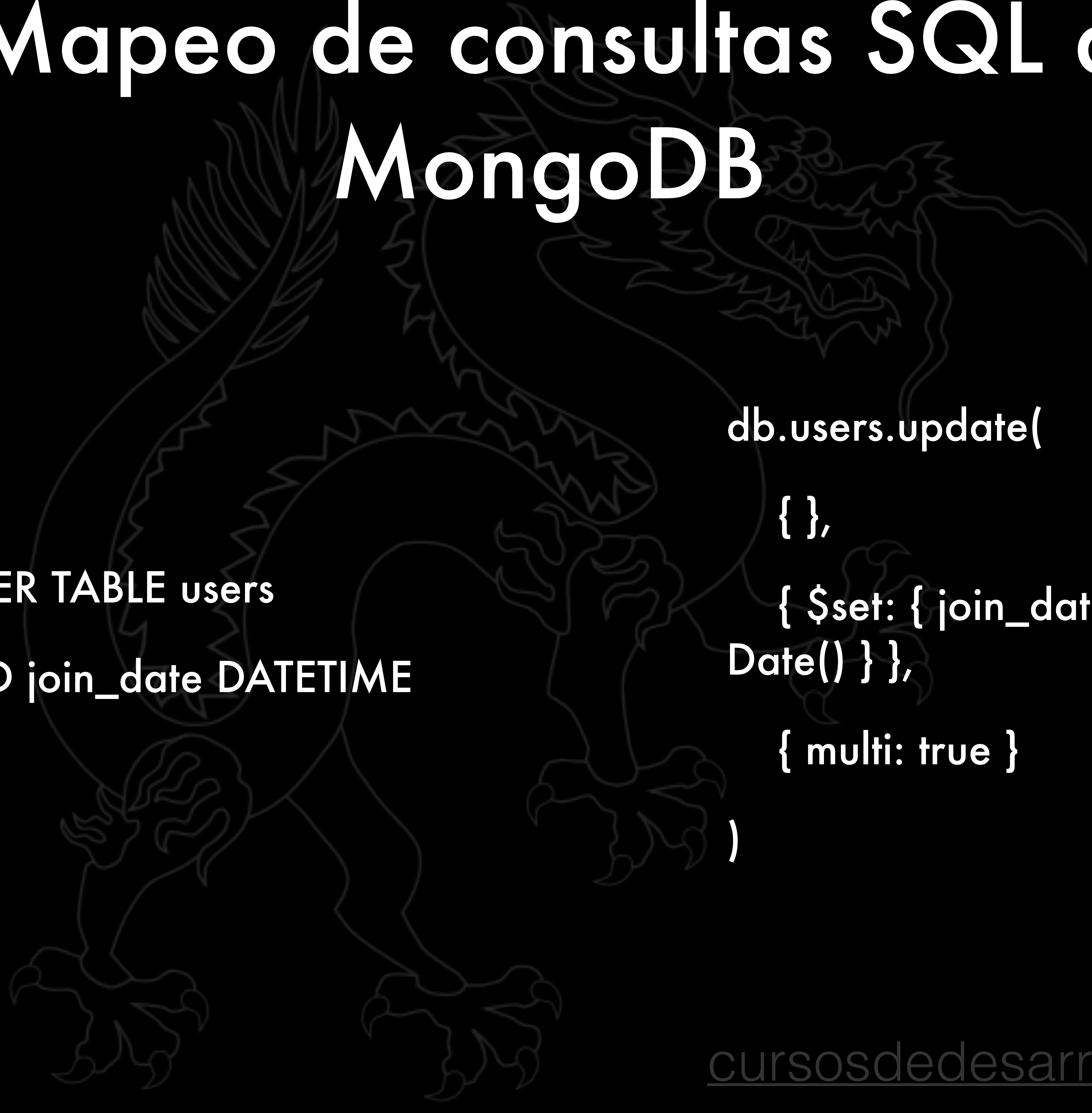
Mapeo de consultas SQL a MongoDB

```
CREATE TABLE users (
    id MEDIUMINT NOT NULL
        AUTO_INCREMENT,
    user_id Varchar(30),
    age Number,
    status char(1),
    PRIMARY KEY (id)
)
```

```
db.users.insert( {
    user_id: "abc123",
    age: 55,
    status: "A"
 } )
```

```
db.createCollection("users")
```

Mapeo de consultas SQL a MongoDB



```
ALTER TABLE users

ADD join_date DATETIME
```

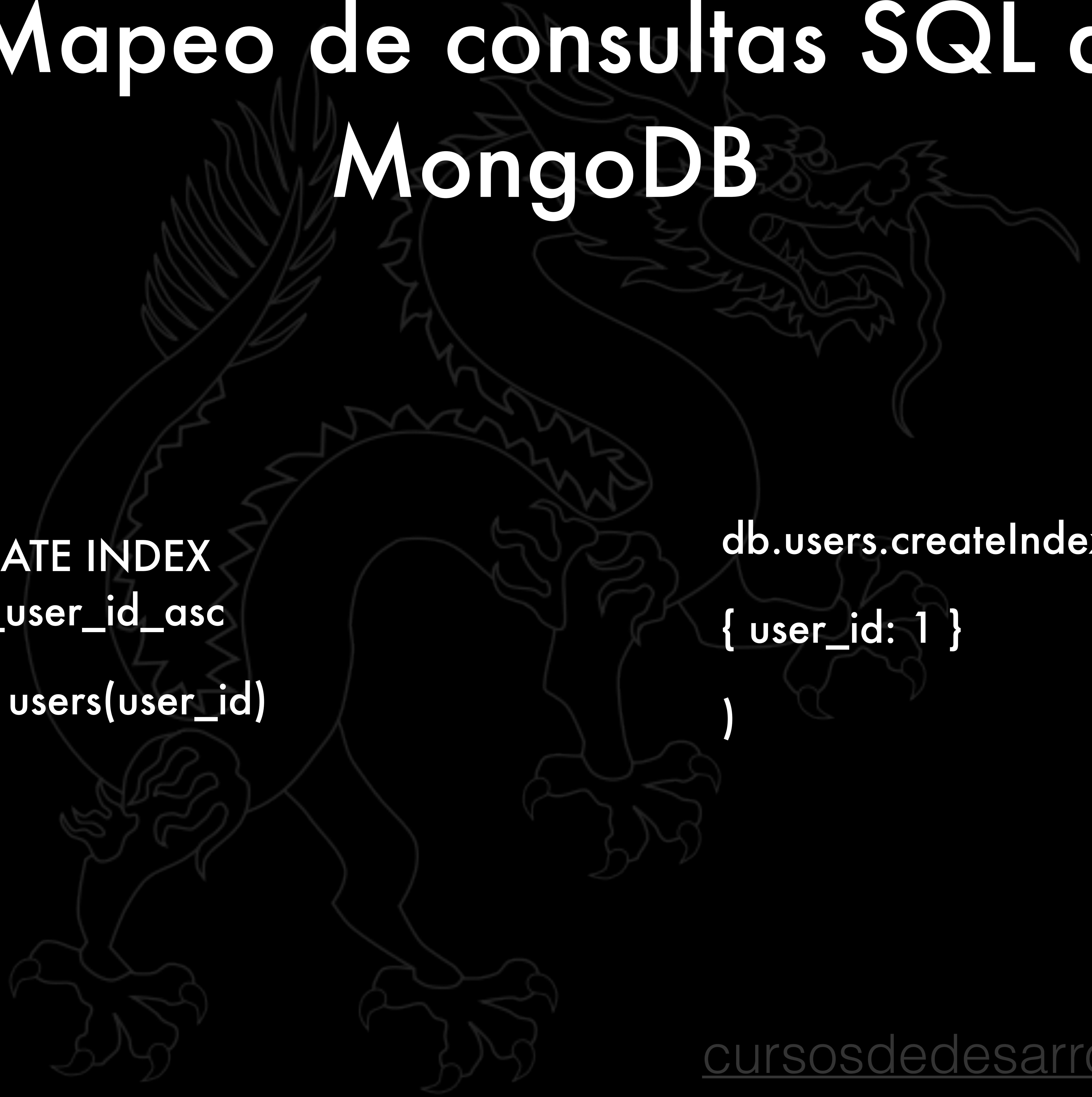
```
db.users.update(

    { },

    { $set: { join_date: new Date() } },

    { multi: true }

)
```

Mapeo de consultas SQL a MongoDB

```
ALTER TABLE users

DROP COLUMN join_date
```

```
db.users.update(

    { },

    { $unset: { join_date: "" } },

    { multi: true }

)
```

Mapeo de consultas SQL a MongoDB



```
CREATE INDEX
idx_user_id_asc

ON users(user_id)
```

```
db.users.createIndex(

{ user_id: 1 }

)
```

Mapeo de consultas SQL a MongoDB

```
CREATE INDEX

idx_user_id_asc_age_desc
ON users(user_id, age DESC)
```

```
db.users.createIndex(
{ user_id: 1, age: -1 }
)
```

Mapeo de consultas SQL a MongoDB

DROP TABLE users	db.users.drop()

Mapeo de consultas SQL a MongoDB

DROP TABLE users	db.users.drop()

Mapeo de consultas SQL a MongoDB

```
INSERT INTO users(user_id,
              age,
              status)
VALUES ("bcd001",
       45,
       "A")
```

```
db.users.insert(
  { user_id: "bcd001", age:
45, status: "A" }
)
```

Mapeo de consultas SQL a MongoDB

```
SELECT *

FROM users
```

```
db.users.find()
```

Mapeo de consultas SQL a MongoDB

```
SELECT id,
       user_id,
       status
FROM users
```

```
db.users.find(
    { },
    { user_id: 1, status: 1 }
)
```

Mapeo de consultas SQL a MongoDB

```
SELECT user_id, status

FROM users
```

```
db.users.find(

    { },

    { user_id: 1, status: 1,

      _id: 0 }

)
```

Mapeo de consultas SQL a MongoDB

```
SELECT *
FROM users
WHERE status = "A"
```

```
db.users.find(
    { status: "A" }
)
```

Mapeo de consultas SQL a MongoDB

```
SELECT *

FROM users

WHERE status != "A"
```

```
db.users.find(

    { status: { $ne: "A" } }

)
```

Mapeo de consultas SQL a MongoDB

```
SELECT *
FROM users
WHERE status = "A"
AND age = 50
```

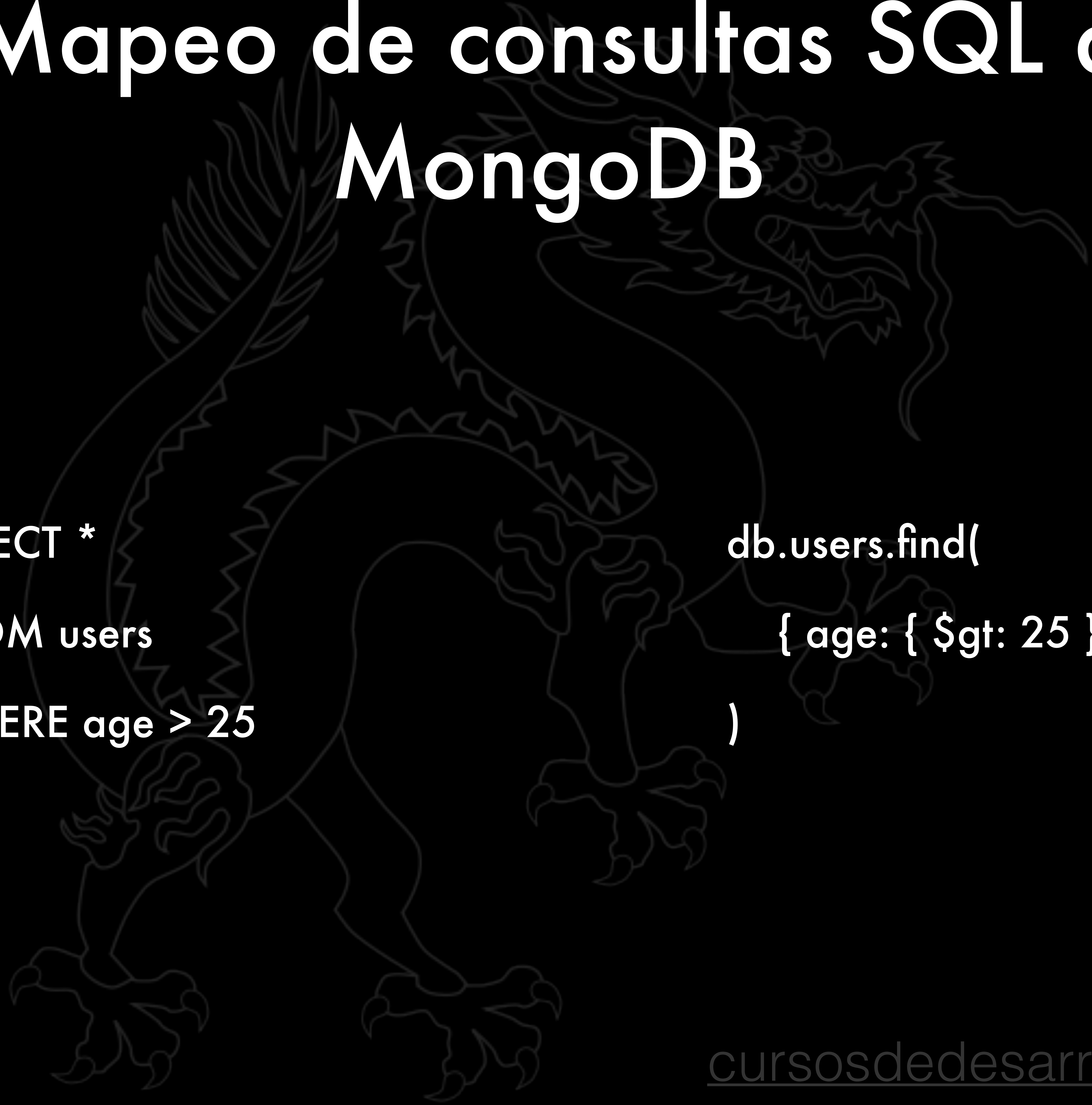
```
db.users.find(
    { status: "A",
      age: 50 }
)
```

Mapeo de consultas SQL a MongoDB

```
SELECT *
FROM users
WHERE status = "A"
OR age = 50
```

```
db.users.find(
    { $or: [ { status: "A" } ,
             { age: 50 } ] }
)
```

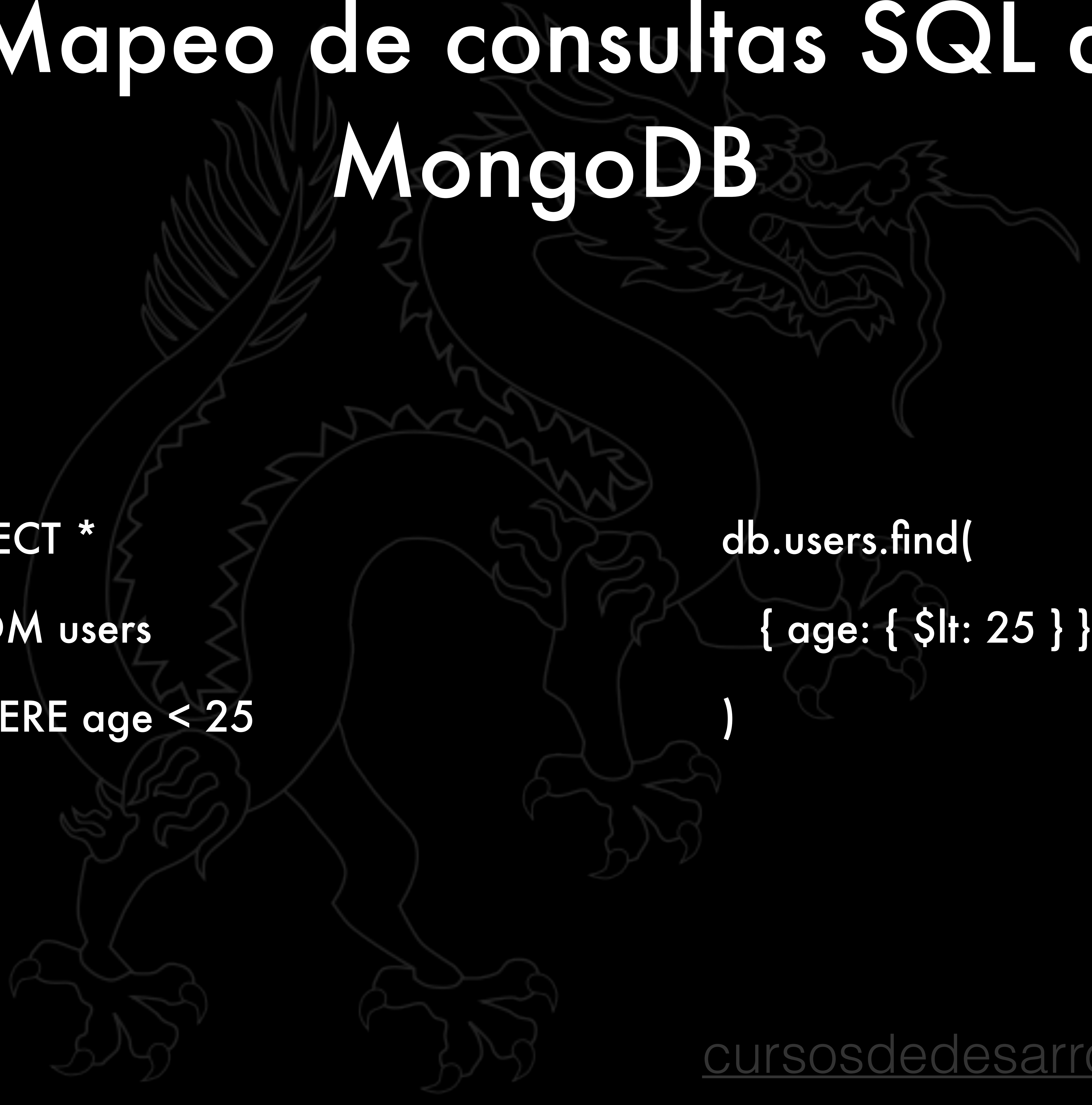
Mapeo de consultas SQL a MongoDB



```
SELECT *
FROM users
WHERE age > 25
```

```
db.users.find(
    { age: { $gt: 25 } }
)
```

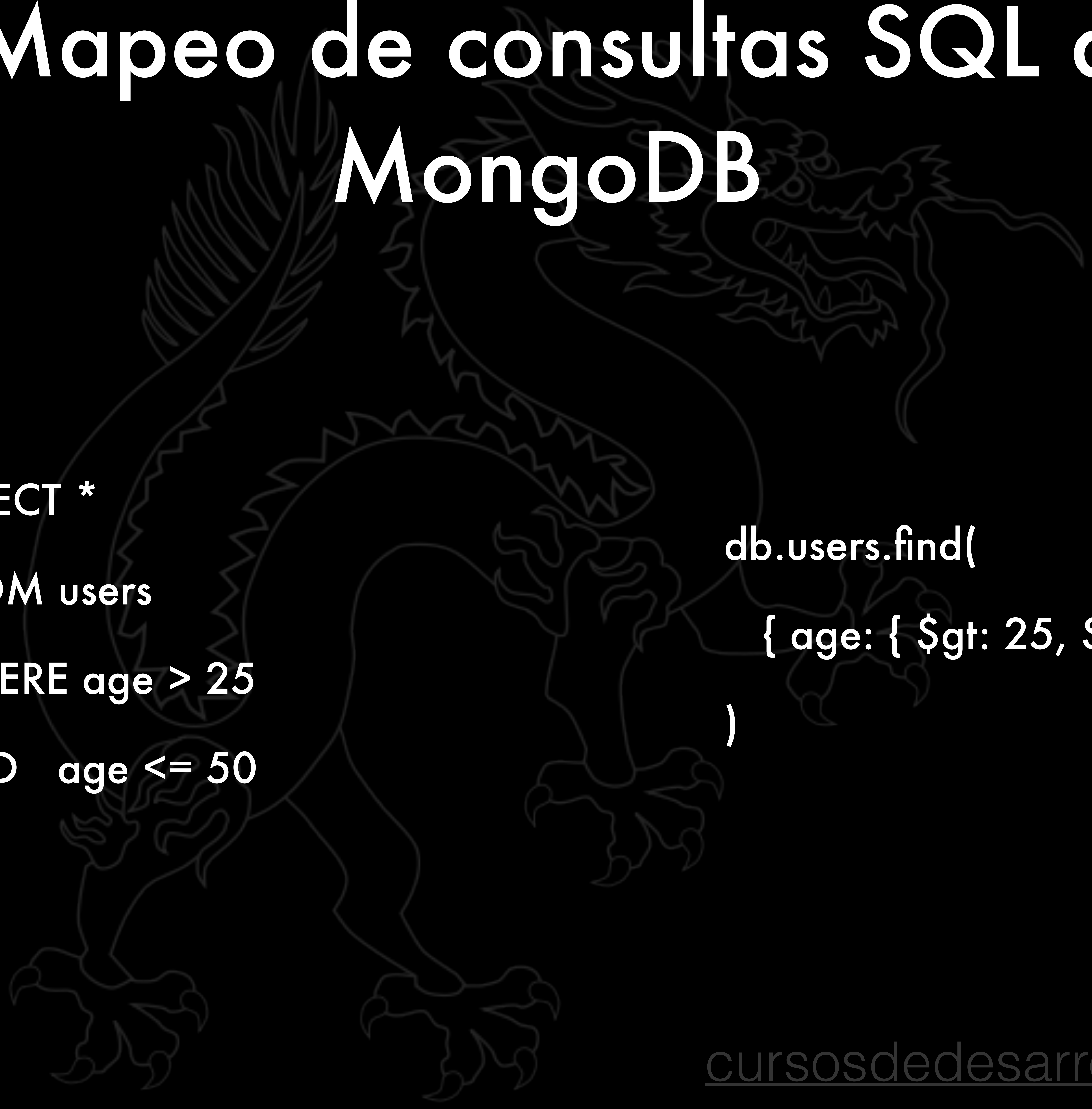
Mapeo de consultas SQL a MongoDB



```
SELECT *
FROM users
WHERE age < 25
```

```
db.users.find(
   { age: { $lt: 25 } }
)
```

Mapeo de consultas SQL a MongoDB



```
SELECT *

FROM users

WHERE age > 25

AND   age <= 50
```

```
db.users.find(

   { age: { $gt: 25, $lte: 50 } }

)
```

Mapeo de consultas SQL a MongoDB

```
SELECT *

FROM users

WHERE user_id like "%bc%"
```

```
db.users.find( { user_id:  /bc/ }
)
```

Mapeo de consultas SQL a MongoDB

```
SELECT *
FROM users
WHERE status = "A"
ORDER BY user_id ASC
```

```
db.users.find( { status: "A" } )
.sort( { user_id: 1 } )
```

Mapeo de consultas SQL a MongoDB

```
SELECT *
FROM users
WHERE status = "A"
ORDER BY user_id DESC
```

```
db.users.find( { status: "A" } )
.sort( { user_id: -1 } )
```

Mapeo de consultas SQL a MongoDB

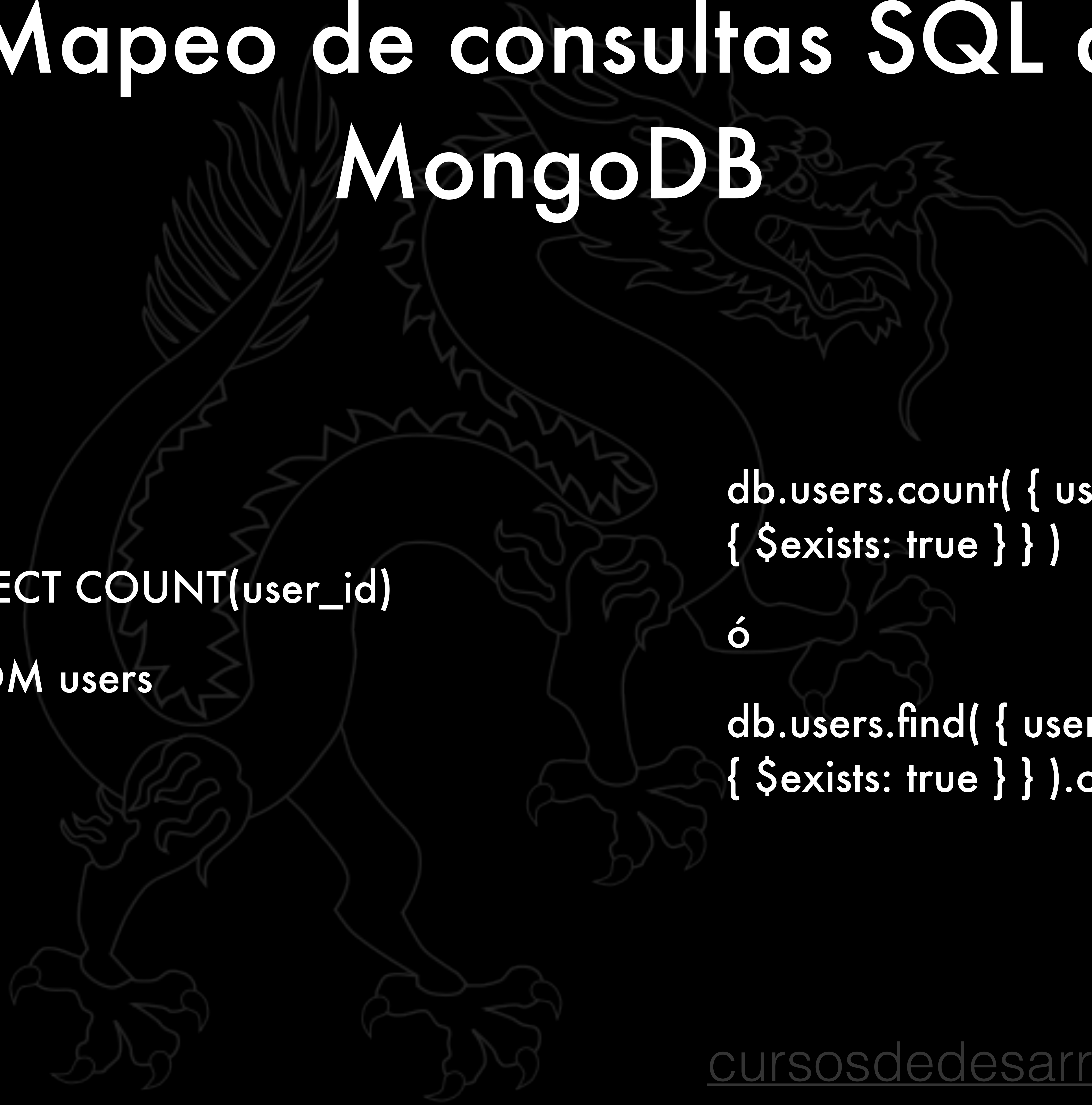
```
SELECT COUNT(*)

FROM users
```

```
db.users.count()
```

ó

```
db.users.find().count()
```

Mapeo de consultas SQL a MongoDB



```
SELECT COUNT(user_id)

FROM users
```

```
db.users.count( { user_id:
{ $exists: true } } )
```

ó

```
db.users.find( { user_id:
{ $exists: true } } ).count()
```

Mapeo de consultas SQL a MongoDB

```
SELECT DISTINCT(status)
FROM users
```

```
db.users.distinct( "status" )
```

Mapeo de consultas SQL a MongoDB

```
SELECT *
FROM users
LIMIT 1
```

```
db.users.findOne()
```

ó

```
db.users.find().limit(1)
```

Mapeo de consultas SQL a MongoDB

```
SELECT *

FROM users

LIMIT 5

SKIP 10
```

```
db.users.find().limit(5).skip(10)
```

Mapeo de consultas SQL a MongoDB

```
EXPLAIN SELECT *

FROM users

WHERE status = "A"
```

```
db.users.find( { status: "A" } )

.explain()
```

Mapeo de consultas SQL a MongoDB

```
UPDATE users
SET status = "C"
WHERE age > 25
```

```
db.users.update(
   { age: { $gt: 25 } },
   { $set: { status: "C" } },
   { multi: true }
)
```

Mapeo de consultas SQL a MongoDB

```
UPDATE users

SET age = age + 3

WHERE status = "A"
```

```
db.users.update(

   { status: "A" } ,

   { $inc: { age: 3 } },

   { multi: true }

)
```

Mapeo de consultas SQL a MongoDB

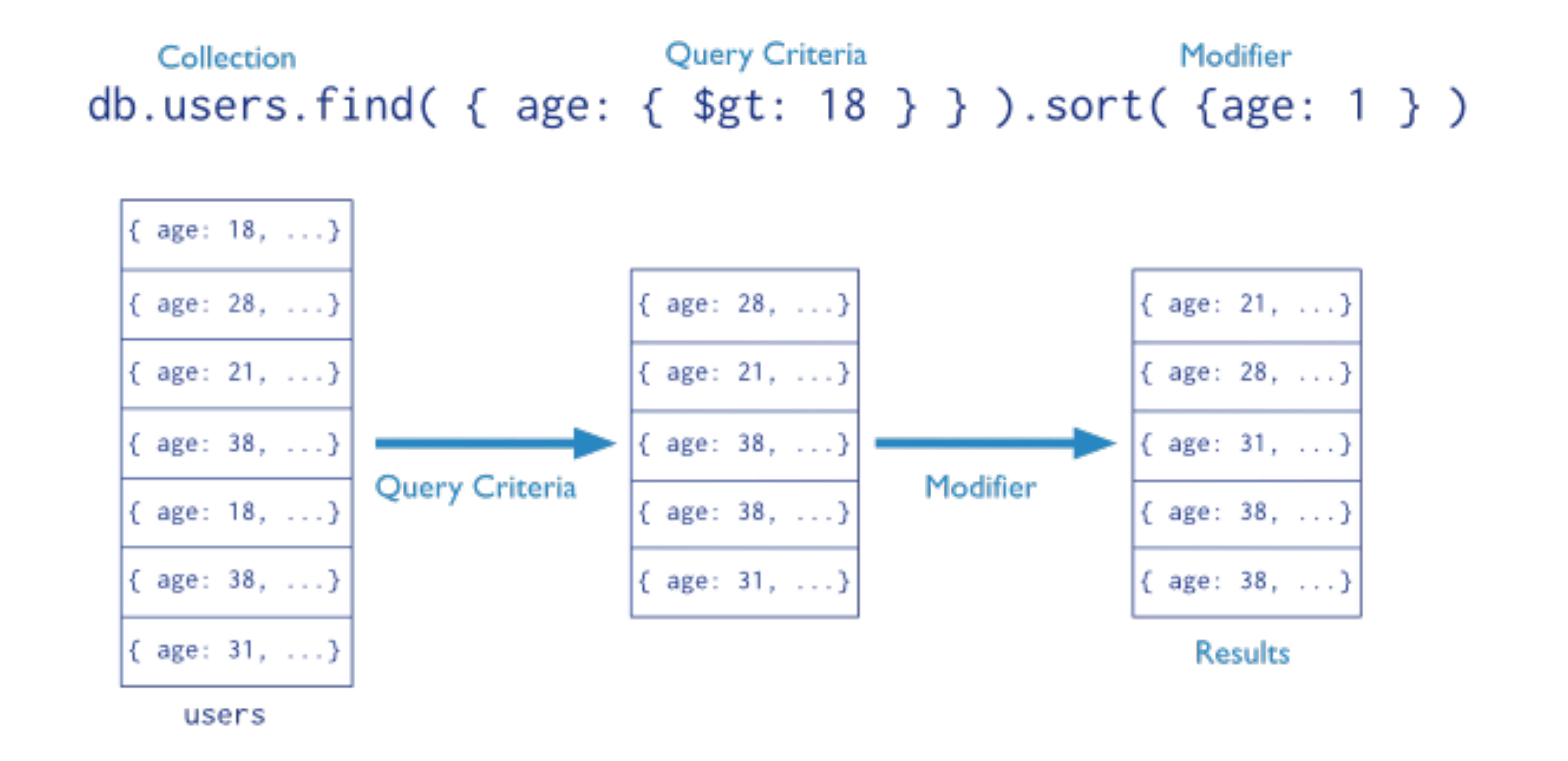
```
DELETE FROM users

WHERE status = "D"
```

```
db.users.remove(

{ status: "D" }

)
```

Mapeo de consultas SQL a MongoDB

DELETE FROM users	db.users.remove({})

Consultas básicas y búsquedas de información



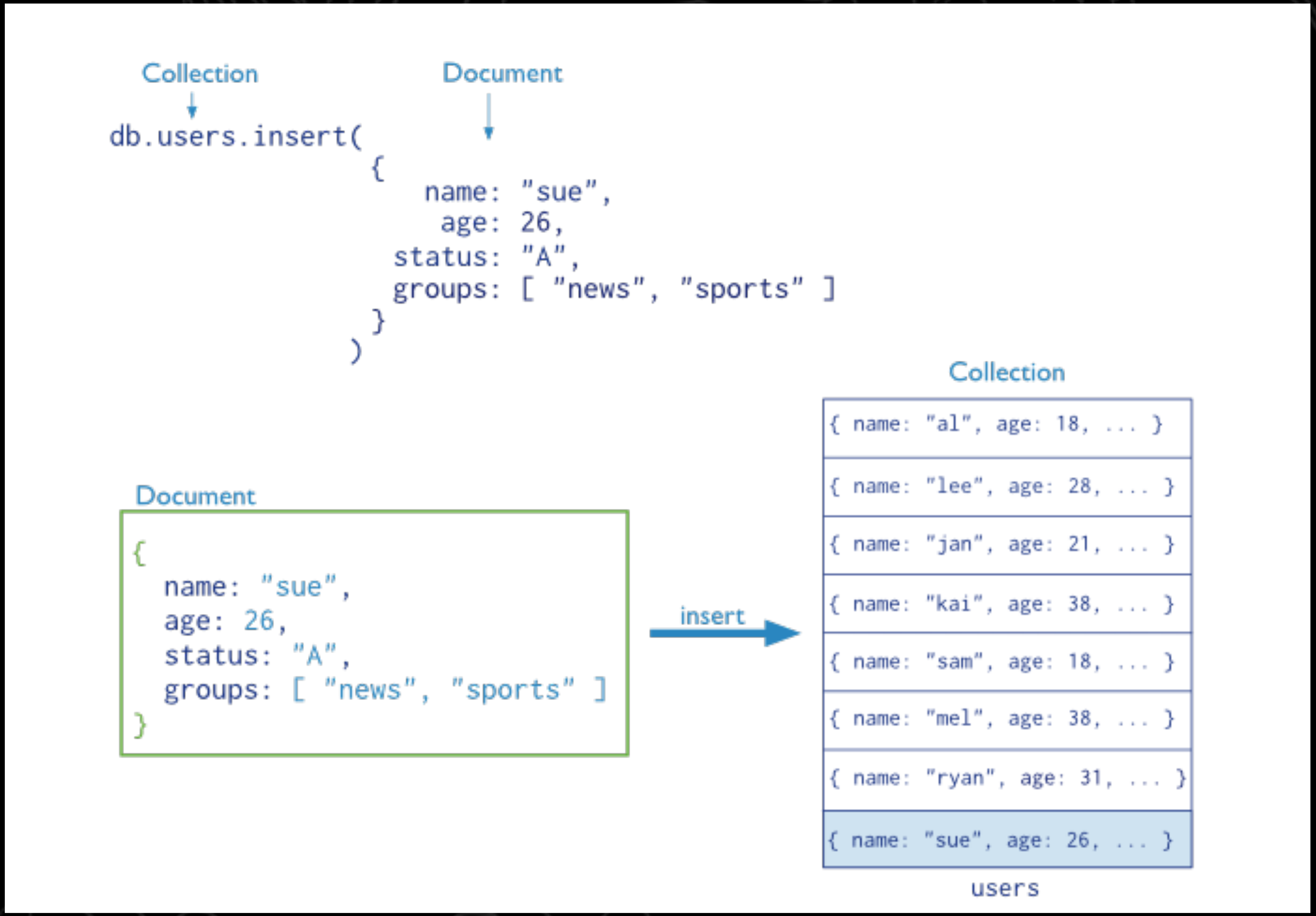
```
db.users.find( { age: { $gt: 18 } } ).sort( {age: 1 } )
```

```
{ age: 18, ...}
{ age: 28, ...}
{ age: 21, ...}
{ age: 38, ...}
{ age: 18, ...}
{ age: 38, ...}
{ age: 31, ...}
```

```
{ age: 28, ...}
{ age: 21, ...}
{ age: 38, ...}
{ age: 38, ...}
{ age: 31, ...}
```

```
{ age: 21, ...}
{ age: 28, ...}
{ age: 31, ...}
{ age: 38, ...}
{ age: 38, ...}
```

Consultas básicas y búsquedas de información

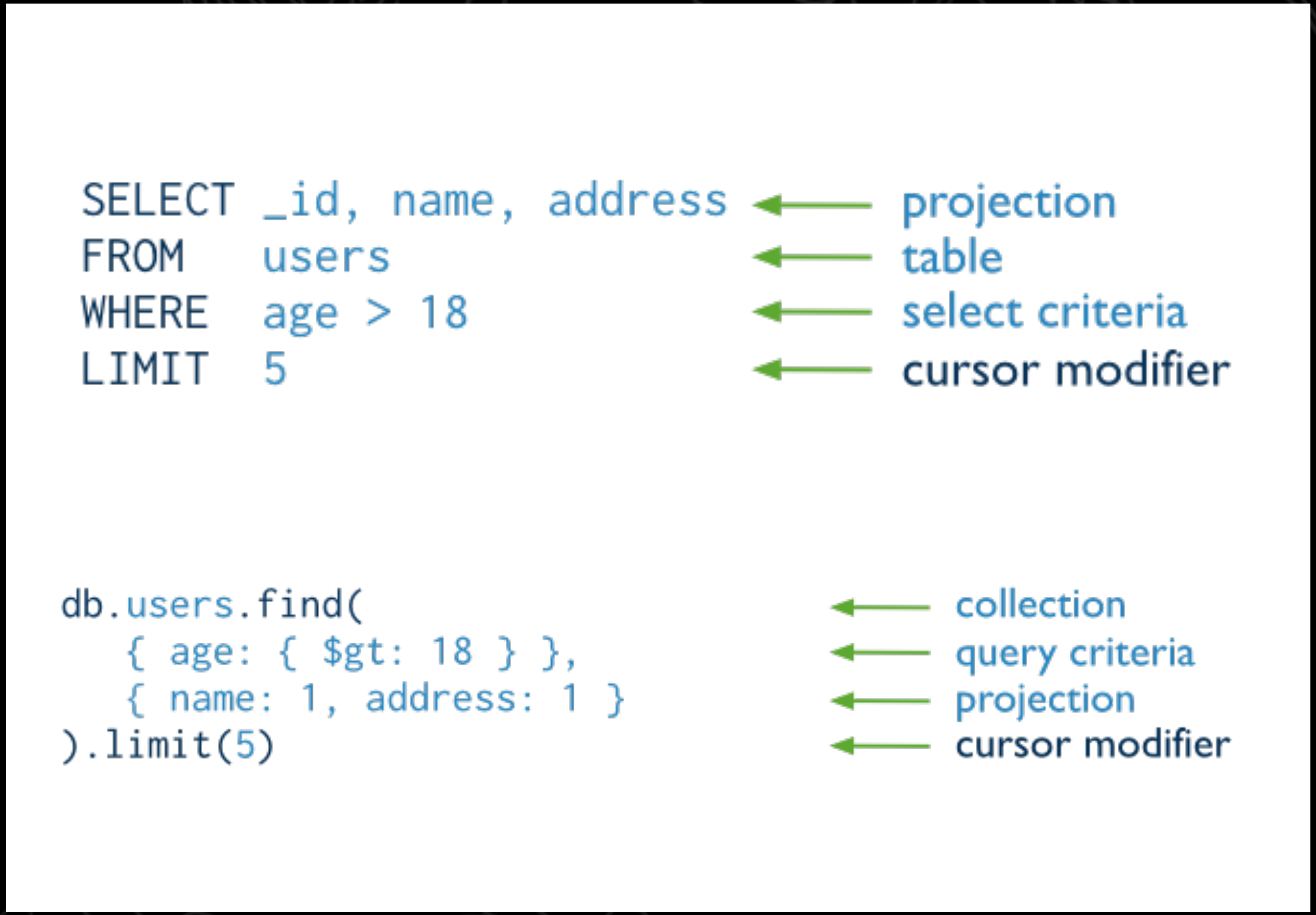


```
db.users.insert(
                {
                   name: "sue",
                    age: 26,
                 status: "A",
                 groups: [ "news", "sports" ]
                }
              )
```

```
{
  name: "sue",
  age: 26,
  status: "A",
  groups: [ "news", "sports" ]
}
```

```
{ name: "al", age: 18, ... }
{ name: "lee", age: 28, ... }
{ name: "jan", age: 21, ... }
{ name: "kai", age: 38, ... }
{ name: "sam", age: 18, ... }
{ name: "mel", age: 38, ... }
{ name: "ryan", age: 31, ... }
{ name: "sue", age: 26, ... }
```

Consultas básicas y búsquedas de información



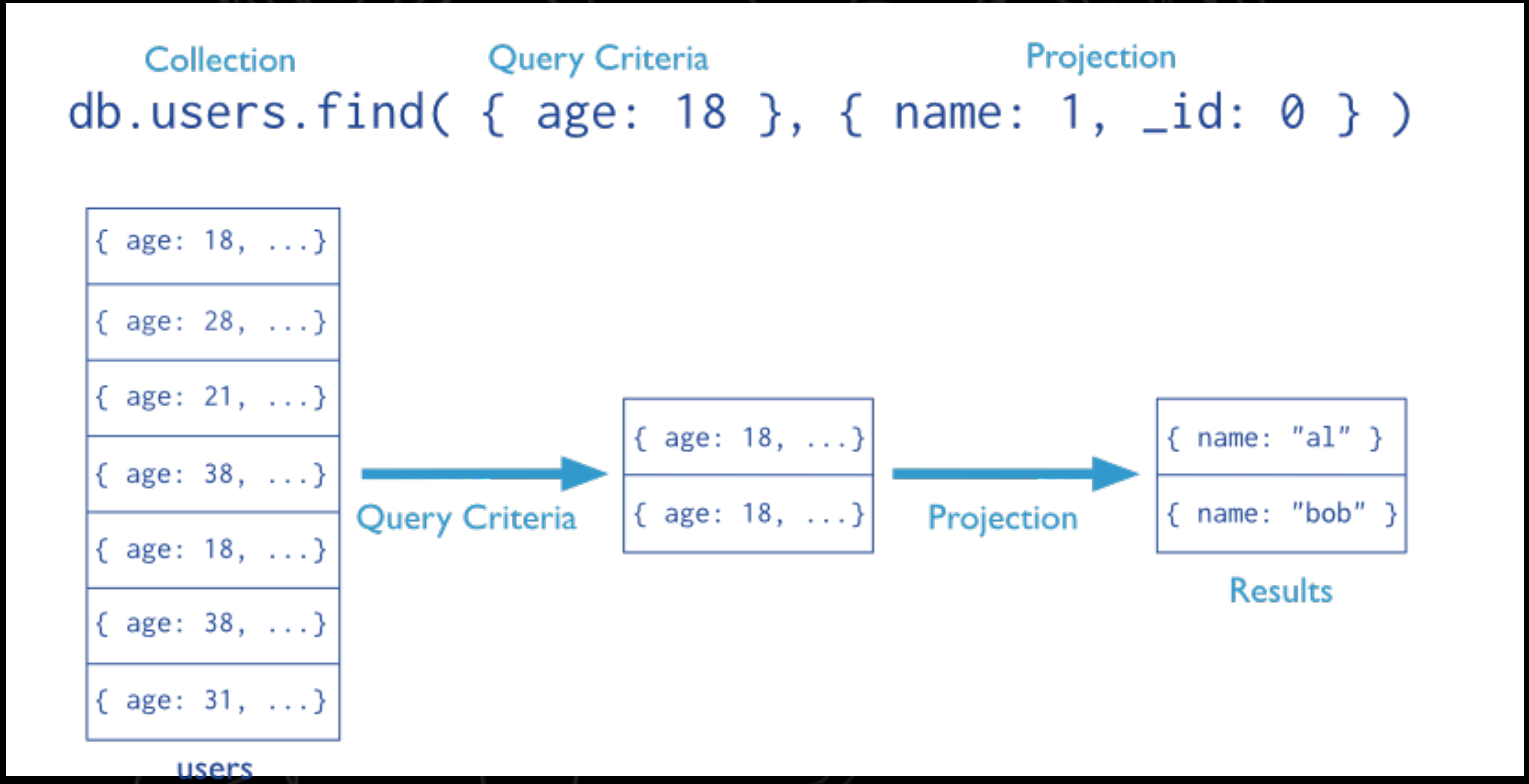
```
SELECT _id, name, address   ←  projection
FROM   users                ←  table
WHERE  age > 18             ←  select criteria
LIMIT  5                    ←  cursor modifier
```

```
db.users.find(                          ←  collection
   { age: { $gt: 18 } },                ←  query criteria
   { name: 1, address: 1 }              ←  projection
).limit(5)                              ←  cursor modifier
```

Consultas básicas y búsquedas de información

- Todas las consultas se hacen sobre una colección
- Se pueden modificar para imponer límites (limit), saltos (skips) y ordenaciones (sort)
- Las sentencias de actualización (update) utilizan la misma sintaxis que las consultas (find)
- En las proyecciones se incluirá siempre el identificativo de Objeto a no ser que se especifique lo contrario en la consulta

Consultas básicas y búsquedas de información



```
db.users.find( { age: 18 }, { name: 1, _id: 0 } )
```

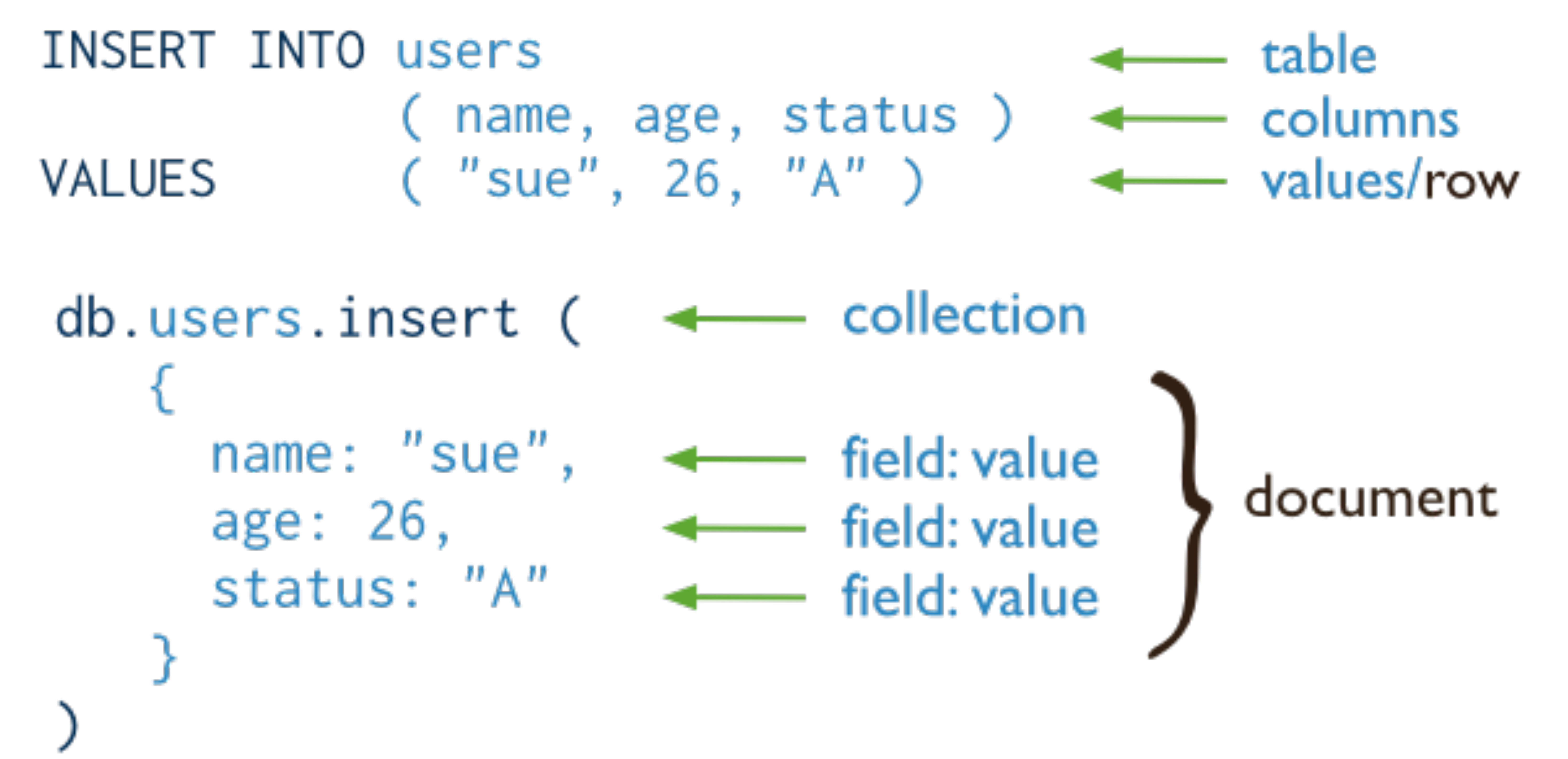
Consultas básicas y búsquedas de información

- Para mejorar el rendimiento de las consultas se pueden establecer índices
- Si hacemos una consulta del estilo a:
  - db.inventory.find({ type: typeValue });
- Podemos crear un índice en ese campo
  - db.inventory.createIndex({ type: 1 })

Operaciones de Lectura

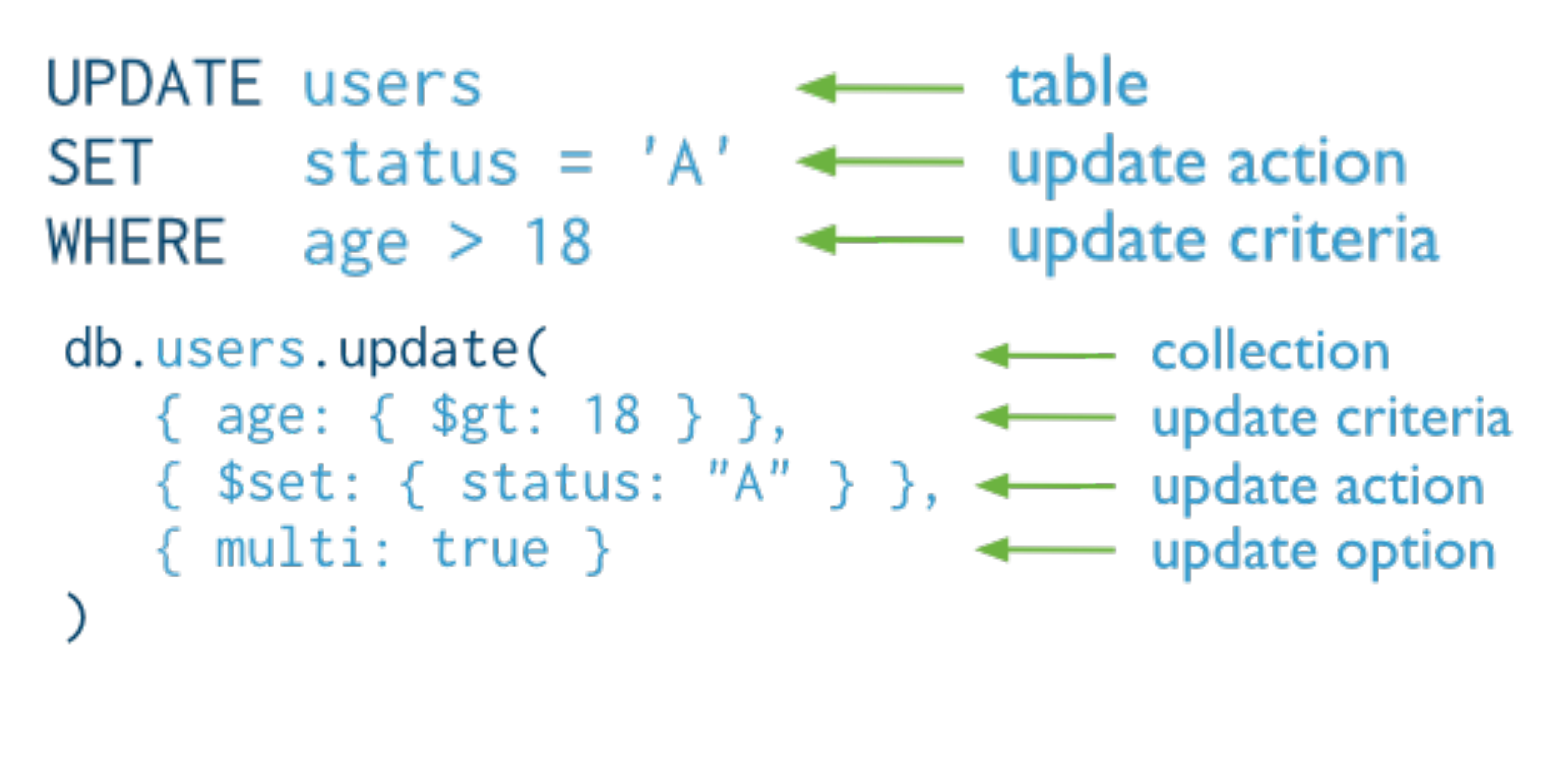
- MongoClient client = new MongoClient();
- MongoDatabase database = client.getDatabase("mydb");
- MongoCollection<Document> collection = database.getCollection("mycoll");

Operaciones de Escritura



```
INSERT INTO users
            ( name, age, status )
VALUES      ( "sue", 26, "A" )

db.users.insert (
   {
     name: "sue",
     age: 26,
     status: "A"
   }
)
```

Operaciones de Escritura



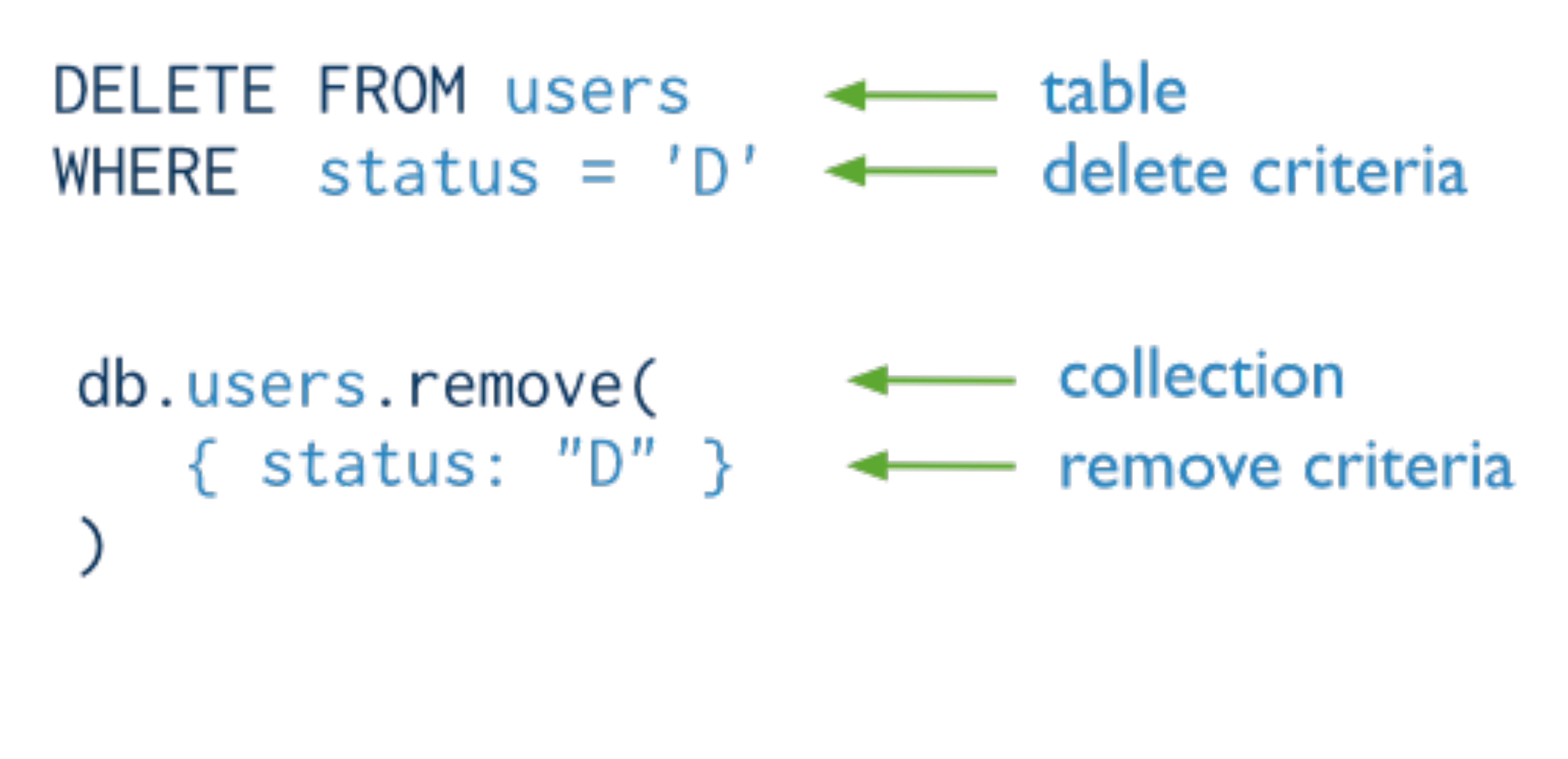
```
UPDATE users          ←  table
SET    status = 'A'   ←  update action
WHERE  age > 18       ←  update criteria
```

```
db.users.update(                       ←  collection
   { age: { $gt: 18 } },               ←  update criteria
   { $set: { status: "A" } },          ←  update action
   { multi: true }                     ←  update option
)
```

Operaciones de Escritura

```
MongoCollection<Document> collection = database.getCollection("mycoll");

// insert a document

Document document = new Document("x", 1)

collection.insertOne(document);

document.append("x", 2).append("y", 3);

// replace a document

collection.replaceOne(Filters.eq("_id", document.get("_id")), document);

// find documents

List<Document> foundDocument = collection.find().into(new
ArrayList<Document>());
```

Operaciones de Escritura



```
DELETE FROM users
WHERE  status = 'D'
```

```
db.users.remove(
   { status: "D" }
)
```

Ejemplo en Java

Ejemplo de Uso Maven de Mongo DB con el Java Driver

Autoría: David Vaquero Santiago
http://cursosdedesarrollo.com

cursosdedesarrollo.com