ISTEA

Bases de No Datos Relacionales

Revisión operaciones básicas

desde MongoCompassDB

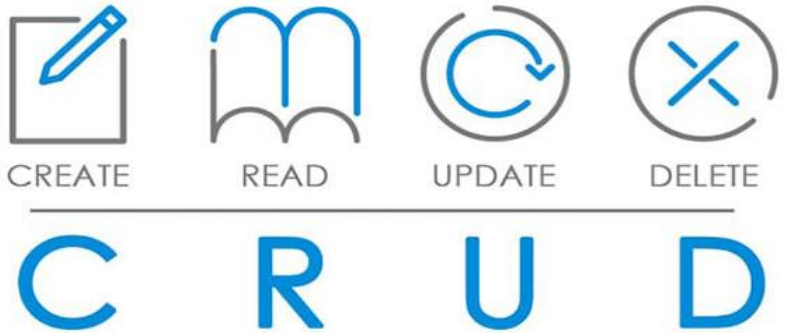
ISTEA

Bases de No Datos Relacionales

- Operaciones CRUD I

  CREATE / READ

CRUD hace referencia al acrónimo en el que se reúnen las primeras letras de las cuatro operaciones fundamentales que se utilizan en aplicaciones que persisten datos en distintos sistemas bases de datos:



Comenzamos con el CREATE

Vamos a consultar cuáles con las operaciones que nos brinda MongoDB.

Para esto en una ventana de comandos, van a ejecutar **mongod**, para iniciar el motor de base de datos

Luego van a ejecutar, en una nueva ventana de comando el **mongosh** para iniciar el shell de mongo y poder correr el siguiente comando:

db.collection.help()

CREATE

¿Cuáles son los comandos que consideran que nos permiten CREAR documentos en una base de MongoDB?

- insert
- insertMany
- insertOne
- bulkWrite

CREATE

Insertar documentos utilizando insert (comando de la base de datos)

Lo primero que debemos recordar es que en MongoDB insertamos documentos de tipo JSON, por tanto debemos pasar como parámetros a la instrucción insert **un documento o un array de documentos** que cumplan con la forma de pares:

{etiqueta1:valor1, etiqueta2:valor2}

Por ejemplo:

```
use crud1
db.PrimerCol.insert({x:1,y:2})
db.PrimerCol.insert({x:2,y:3,z:4})
```

CREATE

Insertar documentos utilizando insert

Varios documentos en el mismo insert utilizando un array:

db.PrimerCol.insert([{x:3,y:4},{x:4,y:5},{x:5,y:6}, {x:6,y:7}],{ordered:true})

La cláusula **ordered** realiza una inserción ordenada de los documentos del array. Si un insert falla no se ejecutarán los próximos. Por default es true.

Podriamos consultar la cantidad de documentos hasta aqui insertados ejecutando db.PrimerCol.countDocuments()

Pregunta corta / respuesta rápida

Pregunta corta / respuesta rápida

¿Con qué datos se conforma el ObjectId de un documento en MongoDB?

CREATE

Insertar documentos utilizando insertOne e insertMany

Estos **son métodos de inserción implementados en el shell de Mongo**, no comandos de la base de datos, y que se asocian con el comando insert de la siguiente manera:

db.coleccion.insert ({unDocumento}) → db.coleccion.insertOne({unDocumento})
db.coleccion.insert ([arrays de Documentos]) → db.coleccion.insertMany([arrays de Documentos])

CREATE

Insertar documentos utilizando insertOne e insertMany

Con lo cual podríamos ejecutar lo siguiente para obtener el mismo resultado que antes:

db.PrimerCol.**insertOne**({x:1,y:2})
db.PrimerCol.**insertOne**({x:2,y:3,z:4})
db.PrimerCol.**insertMany**(**[**{x:3,y:4},{x:4,y:5},{x:5,y:6}, {x:6,y:7}**]**,{ordered:true})

READ

¿Cuáles son los comandos que consideran nos permiten LEER documentos en una base de MongoDB?

- find
- findOne

READ

Leer documentos utilizando el método de Mongosh find

La sintaxis que debemos respetar es la siguiente:
db.collection.find({query}, {projection}, {options})
donde los tres parámetros deben ser documentos JSON, y los tres son opcionales:

- **{query}** me permite especificar un filtro a aplicar en los documentos
- **{projection}** me permite especificar qué atributos retornar
- **{options}** me permiten definir el comportamiento de la consulta y la forma de retornar los resultados

Por ejemplo:

```
use crud1
db.PrimerCol.find () ó db.PrimerCol.find ({})
```

READ

Leer documentos utilizando el método de Mongosh find

Filtrando y proyectando documentos:

```
use crud1
db.PrimerCol.find ({x:1}, {_id:0,y:1})
db.PrimerCol.find ({x:1, y:2}, {_id:0})
```

→ la “**,**” en el actúa como el operador ***and***

Qué otros operadores existen?

READ

Leer documentos utilizando el método de Mongosh **find**

Usando operadores para consultar:

Tipos de operadores:

1. De comparación
2. Lógicos
3. De elementos
4. De evaluación
5. Geoespaciales
6. De arrays
7. Bitwise

READ

Leer documentos utilizando el método de Mongosh find

Operadores:

1. **De comparación:** SINTAXIS → `{ atributo: { $operador: value } }`

$gt	buscar valores mayores que un valor especificado
$gte	buscar valores mayores o iguales que un valor especificado
$lt	buscar valores menores que un valor especificado
$lte	buscar valores menores o iguales que un valor especificado
$eq	buscar valores iguales a un valor especificado
$ne	buscar valores que no sean iguales a un valor especificado
$in	buscar cualquiera de los valores especificados en un

READ

Leer documentos utilizando el método de Mongosh find

Ejemplos operadores: De comparación

$gt
```
db.pizzas.find( { price: { $gt: 8 } } )
```

$gte
```
db.pizzas.find( { price: { $gte: 10 } } )
```

$lt
```
db.pizzas.find( { price: { $lt: 6 } } )
```

$lte
```
db.pizzas.find( { price: { $lte: 4 } } )
```

$eq *
```
db.pizzas.find( { price: { $eq: 7 } } )
```

$ne
```
db.pizzas.find( { price: { $ne: 8 } } )
```

$in *
```
db.pizzas.find( { price: { $in:[4,5,6] } } )
```

$nin *
```
db.pizzas.find( { price: { $nin:[4,5,6] } } )
```

Varios operadores ejemplos:
```
db.pizzas.find( { price: { $gt: 4 } , size: { $eq: "small" } } )
db.pizzas.find( { price: { $gt: 4 } ,  price: { $lt: 10 } } )
```

READ

Leer documentos utilizando el método de Mongosh find

Operadores:

2. Lógicos:

$and realiza la operación lógica **and** de una o varias expresiones y retorna los documentos que satisfacen **todas** las expresiones.

$or realiza la operación lógica **or** de una o varias expresiones y retorna los documentos que satisfacen **alguna** de las expresiones.

$nor realiza la operación lógica **nor** de una o varias expresiones y retorna los documentos que satisfacen **ninguna** de las expresiones.

$not realiza la operación lógica de negación sobre una expresión, y retorna los documentos que no cumplen con esa condición, incluso documentos que no contengan los atributos mencionados en la expresión

READ

Leer documentos utilizando el método de Mongosh find

Operadores:

2. Lógicos:

SINTAXIS → { $operador: [{ expresion1 }, ... , { expressionN }] }

$and
```
db.pizzas.find({$and:[{ price:{$ne: 7}},{size:"small"}]})
```
$or
```
db.pizzas.find({$or:[{ price:{$eq: 7}},{size:"small"}]})
```
$nor
```
db.pizzas.find({$nor:[{ price:{$eq: 7}},{size:"small"}]})
```

SINTAXIS → { atributo: {$operador: [{ operator-expression }] } }

$not
```
db.pizzas.find({price: {$not :{$eq:4}}})
db.pizzas.find({price: {$not: {$gt: 8}}})
```

READ

Leer documentos utilizando el método de Mongosh find

Operadores:

3. De elementos (consulta sobre atributos)

SINTAXIS → `db.coleccion.find({{ atributo: { $exists: <boolean> } }})`

$exist Cuando <boolean> es verdadera, retornará los documentos que contienen el campo, incluyendo los documentos en los que el valor del campo es nulo. Si <boolean> es falso, la consulta devuelve los documentos que no contienen el campo.


```
db.pizzas.find({spice:{$exists:true}})
```

SINTAXIS → `db.coleccion.find({{ atributo: { $type: <BSON type> } }})`

$type selecciona documentos en los que el valor del campo es del tipo BSON especificado.

Operadores

$type:

Algunos tipos BSON para pasar como parámetros

Las siguientes instrucciones son idénticas

```
db.pizzas.find({{ spice: { $type: 2}}})
db.pizzas.find({{ spice: { $type: "string"}}})
```

Type	Number	Alias
Double	1	"double"
String	2	"string"
Object	3	"object"
Array	4	"array"
Binary data	5	"binData"
ObjectId	7	"objectId"
Boolean	8	"bool"
Date	9	"date"
Null	10	"null"
32-bit integer	16	"int"
Timestamp	17	"timestamp"
64-bit integer	18	"long"
Decimal128	19	"decimal"

READ

Leer documentos utilizando el método de Mongosh find

Operadores:

4 . De evaluación

$regex Permite mediante expresiones regulares encontrar coincidencias de patrones en cadenas.

SINTAXIS → varias opciones

```
{ atributo: { $regex: /pattern/, $options: '<options>' } }
{ "atributo": { "$regex": "pattern", "$options": "<options>" } }
{ atributo: { $regex: /pattern/<options> } }
```

Ejemplos:

```
db.pizzas.find ({size:{$regex: /l$/}}) que termine en l
db.pizzas.find ({size:{$regex: /l$/,$options:"i"}})
db.pizzas.find ({"size":{"$regex":/l$/,"$options":"i"}})que termine en l (l=L)
db.pizzas.find ({size:/l$/})                que termine en l
db.pizzas.find ({size:{$regex: /l$/i}})que termine en l (l=L)
db.pizzas.find ({"size":{"$regex":"^s.","$options":"i"}}) (/^a/=/^a.*/=/^a.*$/)
db.pizzas.find ({size:/^s/i})               que comience con s,(s=S)
db.pizzaz.find ({type:{$regex: /c.*n$/}}) que empiece con c y termine con n
```

READ

Leer documentos utilizando el método de Mongosh find

Operadores:

4 . De evaluación

$expre Permite utilizar expresiones para consultar

SINTAXIS → { $expr: { <expression> } }

Ejemplo:

```
db.pizzas.find({$expr:{$gt:["$stats.dislikes", "$stats.likes"]}})
```

READ

Leer documentos utilizando el método de Mongosh findOne

Devuelve **un documento** que satisface los criterios de consulta especificados en la colección o vista.

Si varios documentos satisfacen la consulta, devuelve el primer documento de acuerdo con el orden natural de los documentos en el disco.

SINTAXIS →
```
db.pizzas.findOne( { price: { $gt: 8 } } )
```

READ

Más…

En el caso de los campos de un documento incrustado o embebido, se puede especificar el campo de consulta mediante: notación de puntos, por ejemplo "atributo.atributo_anidado": valor **(Dot Notation).**

Ejemplo:

```
{_id:1,
"cliente" : "GOMESA S.A.",
"cuit" : "30404884831",
"region" : ["CENTRO"],
"direccion":  {
              calle:"9 de julio",
              numero:1010,
              localidad:"TANDIL",
              provincia:"BUENOS AIRES"
              }
}
```

db.clientes.find({**“direccion.localidad”**:”TANDIL”})

READ

y más…

- db.coleccion.find().**limit**(x) → para limitar la cantidad de documentos devueltos al valor x
- db.coleccion.find().**skip**(x) → para saltear los primeros X documentos de resultados
- db.coleccion.find().**limit(X).skip(Y)** → devuelvo X documentos luego de saltear Y documentos
- db.coleccion.find().**sort**({ atributo: 1 }) → ordena de forma ascendente (1) o descendente (-1)

```
db.pizzas.find().sort({price:1})
```

Estas búsquedas son equivalentes

```
db.pizzas.find().sort({ "price": 1 } ).limit( 3 )
db.pizzas.find().limit( 3 ).sort( { "price": 1 } )
```

- **Es importante que al comparar se tenga en cuenta el tipo de dato:**

`db.pizzas.find({"precio" :4})` **NO es lo mismo que** `db.pizzas.find({"precio" :"4"})`

Pregunta corta / respuesta rápida

READ

¿Cómo obtengo la cantidad de documentos devueltos por una operación find?

READ

Filtrar y Contar

Utilizando dot notation agregamos un nuevo método a la instrucción find

db.pizzas.find({query}).count() // .countDocuments()

devuelven la cantidad de documentos que coinciden con el criterio de filtro pasado como parámetro.

¡¡TAREAAAA!!

Extra…

Averiguar

¿Cómo obtengo todos los valores distintos para un atributo en todos los documentos?