

# NoSQL

**Carrera  
Programador  
full-stack**

# MongoDB - Operaciones sobre una colección

- **Buscar documento para cliente con determinado apellido**
  - `db.facturas.find({"cliente.apellido":"Malinez"})`
- **Consultar los primeros dos documentos**
  - `db.facturas.find().limit(2)`
- **Consultar documentos saltando los primeros dos documentos**
  - `db.facturas.find().skip(2)`
- **Visualización mejorada (solo en cmd )**
  - `db.facturas.find().pretty()`
- **Consultar dos documentos, saltando los dos primeros documentos de una colección.**
  - `db.facturas.find().limit(2).skip(2)`
- **Consultar documentos sólo mostrando algunos datos**
  - `db.facturas.find({"cliente.apellido":"Malinez"}, {"cliente.cuit":1, "cliente.region":1})`
- **Buscar documento para cliente con dos criterios**
  - Para Zavasi teníamos dos documentos: `db.facturas.find({"cliente.apellido":"Zavasi"})`
  - Agregamos un criterio: `db.facturas.find({"cliente.apellido":"Zavasi", "nroFactura":1005})`
  - **Ordenamiento en forma ascendente**
  - `db.facturas.find().sort({"nroFactura":1})`
- **Ordenamiento en forma descendente**
  - `db.facturas.find().sort({"nroFactura":-1})`

- 1. Buscar todas las facturas emitidas a clientes cuyo apellido sea "Zavasi".  
`Consulta:db.facturas.find({"cliente.apellido": "Zavasi"}).pretty()`
- 2. Mostrar las dos primeras facturas registradas.  
`Consulta:db.facturas.find().limit(2).pretty()`
- 3. Saltar las dos primeras facturas y mostrar las siguientes dos.  
`Consulta:db.facturas.find().skip(2).limit(2).pretty()`
- 4. Mostrar solo el CUIT y la región del cliente cuyo apellido sea "Malinez", sin mostrar el `_id`.  
`Consulta:db.facturas.find( {"cliente.apellido": "Malinez"}, {"cliente.cuit": 1, "cliente.region": 1, _id: 0})`
- 5. Buscar una factura emitida al cliente "Zavasi" con número de factura 1001.  
`Consulta:db.facturas.find({ "cliente.apellido": "Zavasi", "nroFactura": 1001}).pretty()`
- 6. Ordenar las facturas por número de factura de menor a mayor.  
`Consulta:db.facturas.find().sort({nroFactura: 1})`
- 7. Ordenar las facturas por número de factura de mayor a menor.  
`Consulta:db.facturas.find().sort({nroFactura: -1})`
- 8. Buscar todas las facturas y mostrar solo los nombres de los clientes y el total facturado.  
`Consulta:db.facturas.find({}, { "cliente.nombre": 1, "total": 1, "_id": 0})`
- 9. Buscar facturas con un total mayor a 50000.  
`Consulta:db.facturas.find({ "total": { "$gt": 50000 }}).pretty()`
- 10. Contar cuántas facturas hay en total en la colección.  
`Consulta:db.facturas.countDocuments()`

# MongoDB - Consultando una colección - Criterios

- Busca la cantidad de facturas cuyo Nro. de Factura sea mayor que 1465

```
>
> db.facturas.find( { nroFactura : { $gt: 1465 } } ).count()
30512
>
```

- Busca las facturas cuya fecha de emisión sea mayor o igual al 24/02/2014

```
db.facturas.find({ fechaEmision: { $gte: ISODate("2014-02-24T00:00:00Z") } } )
```

```
> db.facturas.find( { fechaEmision: { $gte: ISODate("2014-02-24T00:00:00Z") } } )
{ "_id" : ObjectId("53685dbb2baf7b93f61df567"), "nroFactura" : 1450, "fechaEmision" : ISODate("2014-02-24T00:00:00Z"), "fechaVencimiento" : ISODate("2014-02-24T00:00:00Z"), "condPago" : "CONTADO", "cliente" : { "nombre" : "Juan Manuel", "apellido" : "Manoni", "cuit" : 2029889382, "region" : "NEA" }, "item" : [ { "producto" : "TUERCA 2mm", "cantidad" : 2, "precio" : 60 }, { "producto" : "TALADRO 12mm", "cantidad" : 1, "precio" : 490 }, { "producto" : "TUERCA 5mm", "cantidad" : 15, "precio" : 90 } ] }
{ "_id" : ObjectId("53685dbb2baf7b93f61df568"), "nroFactura" : 1451, "fechaEmision" : ISODate("2014-02-24T00:00:00Z"), "fechaVencimiento" : ISODate("2014-03-26T00:00:00Z"), "condPago" : "30 Ds FF", "cliente" : { "nombre" : "Soledad", "apellido" : "Lavagno", "cuit" : 2729887543, "region" : "NOA" }, "item" : [ { "produ
```

Operadores \$

\$gt

\$gte

\$lt

\$lte

\$not

\$or

\$in

\$nin

\$exist

\$regex

1. \$gt, \$lt – Mayores o menores que

2. // Sueldos mayores a 250000

```
db.empleados.find({ sueldo: { $gt: 250000 } })
```

// Sueldos menores a 200000

```
db.empleados.find({ sueldo: { $lt: 200000 } })
```

2. \$gte, \$lte – Mayores o iguales, menores o iguales-----

// Sueldos mayores o iguales a 300000 db.empleados.find({ sueldo: { \$gte: 300000 } })

// Sueldos menores o iguales a 150000 db.empleados.find({ sueldo: { \$lte: 150000 } })

3. \$eq, \$ne – Igual o distinto-----

// Empleados que ganan exactamente 220000

```
db.empleados.find({ sueldo: { $eq: 220000 } })
```

// Empleados que NO son del área de Finanzas

```
db.empleados.find({ area: { $ne: "Finanzas" } })
```

4. \$in, \$nin – Coincidencias dentro de un conjunto-----

// Empleados del área de Marketing o Ventas

```
db.empleados.find({ area: { $in: ["Marketing", "Ventas"] } })
```

// Empleados cuyo sueldo NO sea 180000 ni 200000

```
db.empleados.find({ sueldo: { $nin: [180000, 200000] } })
```

5. \$and, \$or – Condiciones combinadas-----

// Empleados del área de IT y con sueldo mayor a 250000

```
db.empleados.find({ $and: [ { area: "IT" }, { sueldo: { $gt: 250000 } } ] })
```

// Empleados del área de Recursos Humanos o que ganan menos de 200000

```
db.empleados.find({ $or: [ { area: "Recursos Humanos" }, { sueldo: { $lt: 200000 } } ] })
```

6. Anidado: \$and con \$in-----

// Empleados de Marketing o Ventas, con sueldo mayor a 200000

```
db.empleados.find({ $and: [ { area: { $in: ["Marketing", "Ventas"] } }, { sueldo: { $gt: 200000 } } ] })
```

# MongoDB - Insertando un documento


- **Sintaxis del método insert:**

```
db.collection.insert  
( <document or array of documents>,  
  { writeConcern: <document>,  
    ordered: <boolean> } )
```

- **Ejemplo, inserción de un documento sin \_id:**

```
db.facturas.insert({nroFactura:30003,codPago:"CONTADO"})
```

```
> db.facturas.insert(<nroFactura:30003,codPago:"CONTADO">)  
WriteResult(< "nInserted" : 1 >)  
>  
>  
> db.facturas.find(<nroFactura:30003>)  
{ "_id" : ObjectId("5459a129cc19250561ad5f82"), "nroFactura" : 30003, "codPago"  
: "CONTADO" }
```



**\_id: Document Id único autogenerado**

# MongoDB - Insertando un documento

- Ejemplo, inserción de un documento con `_id`:

```
db.facturas.insert({_id:23094776, nroFactura:30004,codPago:"CONTADO"})
```

```
> db.facturas.insert(<{_id:23094776,nroFactura:30004,codPago:"CONTADO"}>)
WriteResult<{ "nInserted" : 1 }>
>
>
> db.facturas.find(<nroFactura:30004>)
{ "_id" : 23094776, "nroFactura" : 30004, "codPago" : "CONTADO" }
```

- Al crear una colección, el motor de BD crea un índice único sobre el atributo `_id`

```
> db.facturas.insert(<{_id:23094776,nroFactura:30004,codPago:"30dsFF"}>)
WriteResult<{
  "nInserted" : 0,
  "writeError" : {
    "code" : 11000,
    "errmsg" : "insertDocument :: caused by :: 11000 E11000 duplicate key error index: finanzas.facturas.$_id_ dup key: { : 23094776.0 }"
  }
}>
```



# MongoDB - Borrando un documento

- Operación REMOVE. Sintaxis:

```
db.<collection_name>.remove({criterio_de_eliminación})
```

- Esta operación eliminará los documentos que cumplan con el criterio definido.
  - **Warning:** Remove es una operación de tipo multi-documento!!
  - **Recomendación:** Es conveniente antes de borrar hacer un find o un count para asegurarse lo que quiero borrar.
- **Ejemplo 1 – Borrado de TODOS LOS DOCUMENTOS de una colección**

```
db.accesos.remove({})
```

**Elimina TODOS LOS ELEMENTOS de una colección**

```
> db.accesos.remove({})
WriteResult<{ "nRemoved" : 3 }>
>
> db.accesos.find()
```

# MongoDB - Borrando un documento

- Ejemplo 2 – Remove por clave primaria

```
db.updtst.remove({_id:100})
```

- Elimina el documento cuyo \_id sea 100 de la colección **updtst**

```
> db.updtst.remove(<{_id:100}>)
WriteResult(< "nRemoved" : 1 >)
>
> db.updtst.find()
{ "_id" : 300, "items" : [ 88, 99, 97 ] }
{ "_id" : 200 }
```

- Ejemplo 3 – Borrado por un criterio con múltiples documentos que aplican

```
db.updtst.remove({items:88})
```

```
> db.updtst.remove(<{_id:100}>)
WriteResult(< "nRemoved" : 1 >)
>
> db.updtst.find()
{ "_id" : 300, "items" : [ 88, 99, 97 ] }
{ "_id" : 200 }
```

# MongoDB - Modificando documentos

- Permite modificar uno o más documentos de una colección. Por default modifica sólo un documento

```
db.coleccion.update ( {clausula_where},  
                      {documento_o_expresión_a_modificar},  
                      { upsert, multi, writeconcern}  
                      )
```

- **upsert** (true o false) Si está configurado en “True” significa que realizará un update si existe un documento que concuerda con el criterio, o un insert si no existe algún documento que concuerde con el criterio. El valor default es “false”, en este caso no realiza un insert cuando no existe documento que concuerde con el criterio.
- **multi** (true o false) Es opcional. Si es configurado en true, el update realiza la actualización de múltiples documentos que concuerdan con el criterio cláusula\_where. Si es configurado en false, modifica solo un documento. El valor default es false. Sólo actúa en updates parciales con operadores \$.
- **writeconcern** Es opcional, visto anteriormente

# MongoDB - Modificando documentos

- **Update Totales/Completos**

Se realiza el update del documento completo, reemplazando el mismo.

- **Update Parciales**

## Operadores

### Operadores sobre cualquier atributo

- |         |  |
|---------|--|
| \$set   | • Permite modificar el valor de un atributo, o agregar un nuevo atributo al documento. |
| \$unset | • Permite eliminar un atributo de un documento.  |
| \$inc   | • Incrementa o decrementa el valor de un atributo ( n ó -n)                            |

### Operadores sobre Arrays

- |            |   |
|------------|---|
| \$push     | • Agrega un elemento a un Array o crea un Array con un elemento.  |
| \$addToSet | • Agrega un elemento al Array solo si no existe en el Array.  |
| \$pushAll  | • Agrega varios elementos a un Array con los valores indicados o crea un Array con esos elementos. (Operación Múltiple) |
| \$pop      | • Elimina un elemento de un Array por sus extremos, permitiendo eliminar el primer elemento (-1) o el último (1).       |
| \$pull     | • Elimina todos los elementos de un Array que contengan el valor indicado.  |
| \$pullAll  | • Elimina todos los elementos de un Array que contengan alguno de los valores indicados. (Operación Múltiple)           |

# MongoDB - Modificando documentos completos

- Update Totales/Completos

```
db.updtst.update({x:2},{ "x" : 2, "y" : 999 })
```

Este comando reemplaza **el primer documento encontrado** por con valor x:2 por este otro en donde el elemento y:999, no tengo el control de cuál estoy modificando, lo correcto era modificar poniendo en el criterio el `_id`.

```
> db.updtst.update({x:2},{ "x" : 2, "y" : 999 })
> db.updtst.find()
{ "_id" : ObjectId<"536a8240793253ebed598065">, "x" : 1, "y" : 999 }
{ "_id" : ObjectId<"536a8245793253ebed598066">, "x" : 2, "y" : 999 }
{ "_id" : ObjectId<"536a8248793253ebed598067">, "x" : 2, "y" : 100 }
{ "_id" : ObjectId<"536a824b793253ebed598068">, "x" : 2, "y" : 300 }
{ "_id" : ObjectId<"536a8250793253ebed598069">, "x" : 3, "y" : 100 }
{ "_id" : ObjectId<"536a8254793253ebed59806a">, "x" : 3, "y" : 200 }
{ "_id" : ObjectId<"536a8257793253ebed59806b">, "x" : 3, "y" : 300 }
```

# MongoDB - Modificando documentos parciales

- Ejemplo 1 – Operador \$set – Modificación de un valor de un atributo existente

Dado el siguiente documento:

```
> db.updtst.insert({_id:100,x:10,y:100})
```

```
db.updtst.update({_id:100},{ $set : {x:100}})
```

Realizará una modificación del valor de atributo x a 100

```
> db.updtst.find({_id:100})
{ "_id" : 100, "x" : 100, "y" : 100 }
```

- Ejemplo 2 – Operador \$set – Opción multi - Agregar un atributo en todos los documentos

```
db.updtst.update({x:2},{ $set : {z:"NUEVO"}},{multi:true})
```

Este reemplaza en TODOS los documentos encontrados con valor x:2 agregando el atributo z:"NUEVO"

```
> db.updtst.update({x:2},{ $set : {z:"NUEVO"}},{multi:true})
> db.updtst.find({x:2})
{ "_id" : ObjectId("536a8245793253ebed598066"), "x" : 2, "y" : 999, "z" : "NUEVO"
" }
{ "_id" : ObjectId("536a8248793253ebed598067"), "x" : 2, "y" : 100, "z" : "NUEVO"
" }
{ "_id" : ObjectId("536a824b793253ebed598068"), "x" : 2, "y" : 300, "z" : "NUEVO"
" }
```