Curso Intro Big Data– MongoDB

MongoDB – Consultas Parte 2 – Inserción de Documentos – Borrado de Documentos – Actualizaciones

DBlandIT®

Consultando una Colección – Expresiones Regulares

Expresiones Regulares

$regex

El operador $regex provee capacidades de expresiones regulares para consultas de strings basadas en "pattern matching".
MongoDB usa expresiones de tipo "Perl compatible regular expressions" ("PCRE.")
Se pueden especificar expresiones regulares usando Objetos de expresiones regulares ó el operador $regex.

Opciones asociadas al operador $regex

- **i la búsqueda No es Case Sensitive. (se puede usar tbién en expresiones regulares de Javascript)**
- **m chequea la expresion contra multilíneas. (se puede usar tbién en expresiones regulares de Javascript)**
- **x "extended" capability. En este caso $regex ignora todos los espacios en blanco entre caracteres.(sólo se puede utilizar con $regex)**
- **s habilita el character "." Para matchear todos los caracteres incluso el character de newline.**

DBlandIT®

Consultando una Colección – Expresiones Regulares

Listar todas las facturas donde el apellido del cliente contiene los textos "Ma" y "ni" en cualquier parte del texto, siempre y cuando "Ma" está antes de "ni", sin tener el cuenta el Case Sensitive (n=N).

```
db.facturas.find( { "cliente.apellido" : /Ma.*ni/i },{nroFactura:1,"cliente.apellido":1,_id:0} );
```

```
> db.facturas.find( {"cliente.apellido": /Ma.*ni/i },{nroFactura:1,"cliente.apel
lido":1,_id:0} );
{ "nroFactura" : 1457, "cliente" : { "apellido" : "Manoni" } }
{ "nroFactura" : 1459, "cliente" : { "apellido" : "Manoni" } }
{ "nroFactura" : 1460, "cliente" : { "apellido" : "Manoni" } }
{ "nroFactura" : 1464, "cliente" : { "apellido" : "Manoni" } }
```

```
db.facturas.find({"cliente.apllido":{$regex:"Ma.*ni",$options:"i"}}
,{nroFactura:1,"cliente.apellido":1,_id:0} );
```

```
> db.facturas.find({"cliente.apellido":{$regex:"Ma.*ni",$options:"i"}},
        {nroFactura:1,"cliente.apellido":1,_id:0} );
{ "nroFactura" : 1457, "cliente" : { "apellido" : "Manoni" } }
{ "nroFactura" : 1459, "cliente" : { "apellido" : "Manoni" } }
{ "nroFactura" : 1460, "cliente" : { "apellido" : "Manoni" } }
{ "nroFactura" : 1464, "cliente" : { "apellido" : "Manoni" } }
{ "nroFactura" : 1466, "cliente" : { "apellido" : "Manoni" } }
```


DBlandIT®

Consultando una Colección – Expresiones Regulares

Consulta con el Case Sensitive activo

```
db.facturas.find({"cliente.apellido":{$regex:'ma.*ni'}} );
```

```
> db.facturas.find({"cliente.apellido":{$regex:"ma.*ni"}} );
>
>
>
>
```

Consulta con el Case Sensitive inactivo

```
db.facturas.find({"cliente.apellido":{$regex:'ma.*ni',$options:"i"}} );
```

```
> db.facturas.find({"cliente.apellido":{$regex:"ma.*ni",$options:"i"}} );
{ "_id" : ObjectId("544fbeca9984ce2ac8036684"), "nroFactura" : 1457, "fechaEmisi
on" : ISODate("2014-02-24T00:00:00Z"), "fechaVencimiento" : ISODate("2014-02-24T
00:00:00Z"), "condPago" : "CONTADO", "cliente" : { "nombre" : "Juan Manuel", "ap
ellido" : "Manoni", "cuit" : 2029889382, "region" : "NEA" }, "item" : [ { "produ
cto" : "TUERCA 2mm", "cantidad" : 2, "precio" : 60 }, { "producto" : "TALADRO 12
mm", "cantidad" : 1, "precio" : 490 }, { "producto" : "TUERCA 5mm", "cantidad" :
 15, "precio" : 90 } ] }
{ "_id" : ObjectId("544fbeca9984ce2ac8036686"), "nroFactura" : 1459, "fechaEmisi
on" : ISODate("2014-02-25T00:00:00Z"), "fechaVencimiento" : ISODate("2014-04-26T
00:00:00Z"), "condPago" : "60 Ds FF", "cliente" : { "nombre" : "Juan Manuel", "a
pellido" : "Manoni", "cuit" : 2029889382, "region" : "NEA" }, "item" : [ { "prod
ucto" : "SET HERRAMIENTAS", "cantidad" : 1, "precio" : 700 }, { "producto" : "TA
LADRO 12mm", "cantidad" : 1, "precio" : 490 } ] }
```


DBlandIT®

Consultando una Colección – Expresiones Regulares

Para combinar una expresión regular de matcheo con otro operador, se deber usar el operador "$regex".

```
db.facturas.find({"cliente.apellido":{$regex:'Ma.*',
                                      $nin:['Manoni'],
                                      $options:'i'}},{"cliente.apellido":1,_id:0} );
```

```
> db.facturas.find({'cliente.apellido':{$regex:'Ma.*',$nin:['Manoni'],$options:'
i'}},{'cliente.apellido':1,_id:0} )
{ "cliente" : { "apellido" : "Malinez" } }
{ "cliente" : { "apellido" : "Malinez" } }
{ "cliente" : { "apellido" : "Malinez" } }
{ "cliente" : { "apellido" : "Malinez" } }
```


DBlandIT®

Consultando una Colección – Cursores

Cursores.

Si queremos hacer un find de todos los documentos de una colección, el motor traerá los primeros 20, y luego Ud. podrá a través de la opción **it** (iteration) solicitarle los próximos 20.
El motor abre un cursor y permite manejarlo con diferentes métodos.

`db.facturas.find()` luego **`it`**

```
> db.facturas.find({},{_id:0,nroFactura:1,"cliente.nombre":1})
{ "nroFactura" : 1448, "cliente" : { "nombre" : "Martin" } }
{ "nroFactura" : 1449, "cliente" : { "nombre" : "Martín" } }
{ "nroFactura" : 1447, "cliente" : { "nombre" : "Marina" } }
{ "nroFactura" : 1448, "cliente" : { "nombre" : "Martin" } }
{ "nroFactura" : 1449, "cliente" : { "nombre" : "Martin" } }
{ "nroFactura" : 1450, "cliente" : { "nombre" : "Juan Manuel" } }
{ "nroFactura" : 1451, "cliente" : { "nombre" : "Soledad" } }
{ "nroFactura" : 1452, "cliente" : { "nombre" : "Juan Manuel" } }
{ "nroFactura" : 1453, "cliente" : { "nombre" : "Juan Manuel" } }
{ "nroFactura" : 1454, "cliente" : { "nombre" : "Marina" } }
{ "nroFactura" : 1455, "cliente" : { "nombre" : "Martin" } }
{ "nroFactura" : 1456, "cliente" : { "nombre" : "Martin" } }
{ "nroFactura" : 1457, "cliente" : { "nombre" : "Juan Manuel" } }
{ "nroFactura" : 1458, "cliente" : { "nombre" : "Soledad" } }
{ "nroFactura" : 1459, "cliente" : { "nombre" : "Juan Manuel" } }
{ "nroFactura" : 1460, "cliente" : { "nombre" : "Juan Manuel" } }
{ "nroFactura" : 1461, "cliente" : { "nombre" : "Marina" } }
{ "nroFactura" : 1462, "cliente" : { "nombre" : "Martin" } }
{ "nroFactura" : 1463, "cliente" : { "nombre" : "Martin" } }
{ "nroFactura" : 1464, "cliente" : { "nombre" : "Juan Manuel" } }
Type "it" for more
```

```
> it
{ "nroFactura" : 1465, "cliente" : { "nombre" : "Soledad" } }
{ "nroFactura" : 1466, "cliente" : { "nombre" : "Juan Manuel" } }
{ "nroFactura" : 1467, "cliente" : { "nombre" : "Juan Manuel" } }
{ "nroFactura" : 1468, "cliente" : { "nombre" : "Marina" } }
{ "nroFactura" : 1469, "cliente" : { "nombre" : "Martin" } }
{ "nroFactura" : 1470, "cliente" : { "nombre" : "Martin" } }
{ "nroFactura" : 1471, "cliente" : { "nombre" : "Juan Manuel" } }
{ "nroFactura" : 1472, "cliente" : { "nombre" : "Soledad" } }
{ "nroFactura" : 1473, "cliente" : { "nombre" : "Juan Manuel" } }
{ "nroFactura" : 1474, "cliente" : { "nombre" : "Juan Manuel" } }
{ "nroFactura" : 1475, "cliente" : { "nombre" : "Marina" } }
{ "nroFactura" : 1476, "cliente" : { "nombre" : "Martin" } }
{ "nroFactura" : 1477, "cliente" : { "nombre" : "Martin" } }
{ "nroFactura" : 1478, "cliente" : { "nombre" : "Juan Manuel" } }
{ "nroFactura" : 1479, "cliente" : { "nombre" : "Soledad" } }
{ "nroFactura" : 1480, "cliente" : { "nombre" : "Juan Manuel" } }
{ "nroFactura" : 1481, "cliente" : { "nombre" : "Juan Manuel" } }
{ "nroFactura" : 1482, "cliente" : { "nombre" : "Marina" } }
{ "nroFactura" : 1483, "cliente" : { "nombre" : "Martin" } }
{ "nroFactura" : 1484, "cliente" : { "nombre" : "Martin" } }
Type "it" for more
```


DBlandIT®

Consultando una Colección – Cursores

.hasNext()
Evalúa si existe un próximo documento. Devuelve True o False.

.next()
Devuelve el próximo documento mostrándolo en formato JSON.

Ejemplo de uso con una función en javascript

```
{
var cursor= db.facturas.find().limit(100);
while (  cursor.hasNext()  ) {

          print("Factura Nro: "+cursor.next().nroFactura);
                                                    }
}
```

```
> { var cursor= db.facturas.find().limit(15);  while (  cursor.hasNext()  ) {
          print("Factura Nro: "+cursor.next().nroFactura);
                                    } }
Factura Nro: 1448
Factura Nro: 1449
Factura Nro: 1447
Factura Nro: 1448
Factura Nro: 1449
Factura Nro: 1450
Factura Nro: 1451
Factura Nro: 1452
Factura Nro: 1453
Factura Nro: 1454
Factura Nro: 1455
Factura Nro: 1456
Factura Nro: 1457
Factura Nro: 1458
Factura Nro: 1459
```


DBlandIT®

Insertando un Documento

El método insert tiene la siguiente sintaxis:

Evalúa si existe un próximo documento. Devuelve True o False.

.

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

writeconcern	Es opcional, lo veremos en la parte de consistencia.
Ordered	lo vemos en un par de slides

Ejemplo, inserción de un documento sin _id:

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

_id: Document Id único autogenerado

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


DBlandIT®

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 duplicat
e key error index: finanzas.facturas.$_id_  dup key: { : 23094776.0 }"
        }
})
```


DBlandIT®

Insertando múltiples Documentos

Ejemplo, inserción de varios documentos:

```
db.facturas.insert([{_id:110,nroFactura:10011},
                    {_id:150,nroFactura:10012},
                    {_id:160,nroFactura:10013},
                    {_id:130,nroFactura:10014}],
                    {ordered:true})
```

La cláusula **ordered** realiza una inserción ordenada de los documentos del array, por default está en false.
Si un insert falla no se ejecutarán los próximos inserts.
Por default ordered está en false.

El método insert() retorna un objeto que contiene el estado de la operación completa.

```
> db.facturas.find({_id:$in(121,125,122,130)})
2014-11-05T06:38:31.326-0300 ReferenceError: $in is not defined
> db.facturas.find({_id:{$in:[121,125,122,130]}})
{ "_id" : 121, "nroFactura" : 10000 }
{ "_id" : 122, "nroFactura" : 10021 }
{ "_id" : 125, "nroFactura" : 10001 }
{ "_id" : 130, "nroFactura" : 10023 }
> db.facturas.insert([{_id:134,nroFactura:10000},{_id:133,nroFactura:10001},{_id
:132,nroFactura:10021},{_id:131,nroFactura:10023}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 4,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
```


DBlandIT®

Bulk Writes

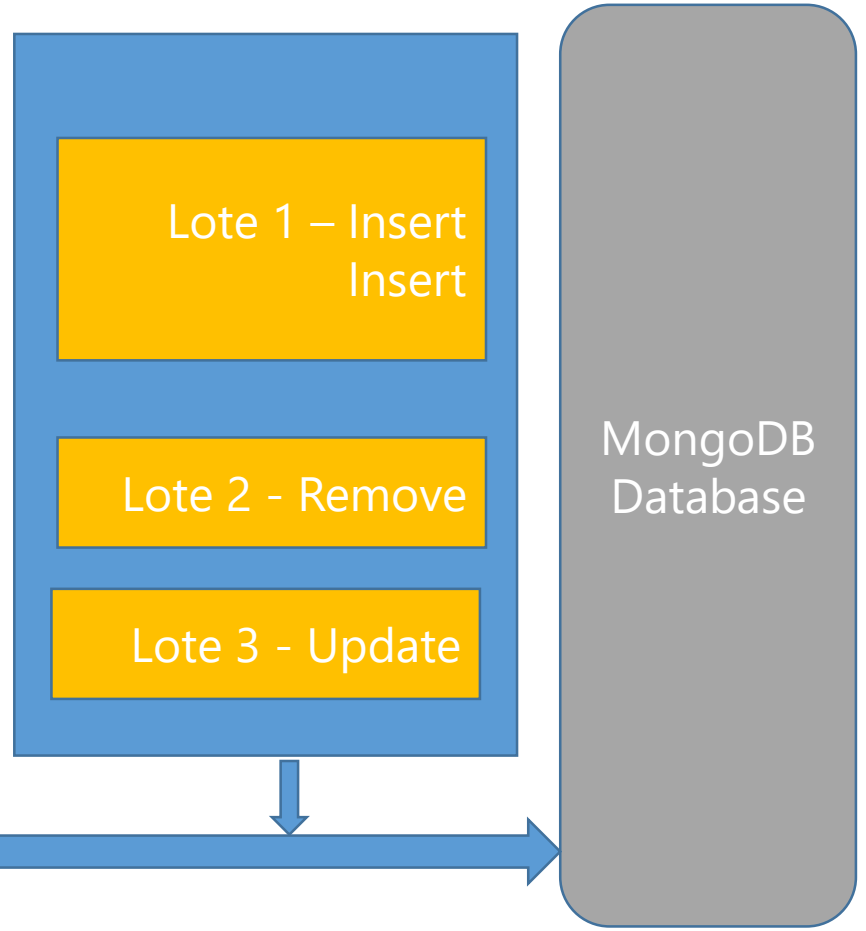
```
var bulk = db.facturas.initializeOrderedBulkOp();

bulk.insert( { nroFactura: 20003, fecEmision:
ISODate("2014-10-21T00:00:00Z"), condPago: "CONTADO",
estado:"Pend"} );
bulk.insert( { nroFactura: 20004, fecEmision:
ISODate("2014-10-21T00:00:00Z"), condPago: "CONTADO",
estado:"Pend" } );

bulk.find( { estado: "Pend" } ).remove();

bulk.find( { estado: "Pend" } ).update( { $set: {
condPago: "30dsFF" } } );

bulk.execute();
```



DBlandIT®

Bulk Writes

```
> var bulk = db.facturas.initializeOrderedBulkOp();
> bulk.insert( { nroFactura: 20001, fecEmision: ISODate("2014-10-21T00:00:00Z"),
 condPago: "CONTADO", estado:"Pend"} );
> bulk.insert( { nroFactura: 20002, fecEmision: ISODate("2014-10-21T00:00:00Z"),
 condPago: "CONTADO", estado:"Pend" } );
> bulk.find( { estado: "Pend" } ).removeOne();
> bulk.find( { estado: "Pend" } ).update( { $set: { condPago: "30dsFF" } } );
> bulk.execute();
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 1,
        "nModified" : 1,
        "nRemoved" : 1,
        "upserted" : [ ]
})
```

El **BulkWrite** retorna un objeto que contiene el estado de la operación completa.
La operación se puede realizar de forma ordered serializando cada operación y asegurando que si una falla, no se ejecutará ninguna de las próximas operaciones. Ó puede realizarse de forma Unordered en la que cada operación se realizará en paralelo, si una operación falla, MongoDB continuará con la ejecución del resto de las operaciones.

DBlandIT®

Método getLastError()

La función getLastError es una función javascript, a cualquier función si la ejecuto sin sus paréntesis, el código de la misma es mostrado en pantalla.

```
> db.getLastError
function ( w , wtimeout ){
    var res = this.getLastErrorObj( w , wtimeout );
    if ( ! res.ok )
        throw "getlasterror failed: " + tojson( res );
    return res.err;
}
```

Si ejecutamos la función db.getLastError luego de realizar una inserción y no hubo error la misma devolverá null.

```
> db.getLastError()
null
> db.getLastErrorObj()
{ "n" : 0, "connectionId" : 18, "err" : null, "ok" : 1 }
```

En versiones anteriores a 2.6 era necesario ejecutar este método después de cada operación de actualización (insert,update o remove) para saber el resultado de la misma.
A partir de la 2.6 los métodos de esctritura retornan el estado de la operación de escritura, incluyendo la información del error.

DBlandIT®

Borrando Documentos

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()
```


DBlandIT®

Borrando Documentos

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 – Remove por un criterio con múltiples documentos que aplican

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

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


DBlandIT®

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 multiples 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, lo veremos en la parte de consistencia.

DBlandIT®

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.
$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)***

DBlandIT®

Modificando Documentos Completos

Update Totales/Completos

```
mydoc=db.facturas.findOne({nroFactura:1449})
```

```
> mydoc=db.facturas.findOne({nroFactura:1449})
{
        "_id" : ObjectId("53685dbb2baf7b93f61df566"),
        "nroFactura" : 1449,
        "fechaEmision" : ISODate("2014-02-20T00:00:00Z"),
        "fechaVencimiento" : ISODate("2014-02-20T00:00:00Z"),
        "condPago" : "CONTADO",
        "cliente" : {
                "nombre" : "Martin",
                "apellido" : "Zavasi",
                "cuit" : 2038373771,
                "region" : "CABA"
        },
        "item" : [
                {
                        "producto" : "TUERCA 2mm",
                        "cantidad" : 6,
                        "precio" : 60
                },
                {
                        "producto" : "CORREA 10mm",
                        "cantidad" : 12,
                        "precio" : 134
                }
        ]
}
```

```
mydoc.condPago="XXX"
db.facturas.update(   { _id: mydoc._id },  mydoc )
```

Sólo actualizamos el documento cuyo _id fue el recuperado con el findOne()

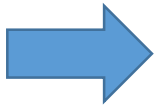
DBandIT®

Modificando Documentos Completos

Update Totales/Completos

```
mydoc=db.facturas.findOne({nroFactura:1449})
```

```
> mydoc=db.facturas.findOne({nroFactura:1449})
{
        "_id" : ObjectId("53685dbb2baf7b93f61df566"),
        "nroFactura" : 1449,
        "fechaEmision" : ISODate("2014-02-20T00:00:00Z"),
        "fechaVencimiento" : ISODate("2014-02-20T00:00:00Z"),
        "condPago" : "CONTADO",
        "cliente" : {
                "nombre" : "Martin",
                "apellido" : "Zavasi",
                "cuit" : 2038373771,
                "region" : "CABA"
        },
        "item" : [
                {
                        "producto" : "TUERCA 2mm",
                        "cantidad" : 6,
                        "precio" : 60
                },
                {
                        "producto" : "CORREA 10mm",
                        "cantidad" : 12,
                        "precio" : 134
                }
        ]
}
```



```
> db.facturas.findOne({nroFactura:1449})
{
        "_id" : ObjectId("53685dbb2baf7b93f61df566"),
        "nroFactura" : 1449,
        "fechaEmision" : ISODate("2014-02-20T00:00:00Z"),
        "fechaVencimiento" : ISODate("2014-02-20T00:00:00Z"),
        "condPago" : "XXX",
        "cliente" : {
                "nombre" : "Martin",
                "apellido" : "Zavasi",
                "cuit" : 2038373771,
                "region" : "CABA"
        },
        "item" : [
                {
                        "producto" : "TUERCA 2mm",
                        "cantidad" : 6,
                        "precio" : 60
                },
                {
                        "producto" : "CORREA 10mm",
                        "cantidad" : 12,
                        "precio" : 134
                }
        ]
}
```

```
mydoc.condPago="XXX"
db.facturas.update(    { _id: mydoc._id },   mydoc )
```

Sólo actualizamos el documento cuyo _id fue el recuperado con el findOne()

DBlandIT®

Modificando Documentos Completos

Update Totales/Completos – Otro Ejemplo

Dada la siguiente colección

```
> db.updtst.find()
{ "_id" : ObjectId("536a8240793253ebed598065"), "x" : 1, "y" : 100 }
{ "_id" : ObjectId("536a8245793253ebed598066"), "x" : 2, "y" : 200 }
{ "_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 }
```

Update de un elemento completo

```
db.updtst.update({x:1},{ "_id" : ObjectId("536a8240793253ebed598065"), "x" : 1, "y" : 999 })
```

Este comando reemplaza el documento que contiene x:1 por otro documento con los valores x:1 e y: 999

```
> db.updtst.update({x:1},{ "_id" : ObjectId("536a8240793253ebed598065"), "x" : 1
, "y" : 999 })
> db.updtst.find()
{ "_id" : ObjectId("536a8240793253ebed598065"), "x" : 1, "y" : 999 }
{ "_id" : ObjectId("536a8245793253ebed598066"), "x" : 2, "y" : 200 }
{ "_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 }
```


DBlandIT®

Modificando Documentos Completos

Update Totales/Completos – Otro Ejemplo

```
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 }
```


DBlandIT®

Modificando Documentos Parciales

Update 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 }
```


DBlandIT®

Modificando Documentos Parciales

Update Parciales

Ejemplo 2 – Operador $set – Agregar un nuevo atributo a un documento existente.

Dado el siguiente documento:

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

```
db.updtst.update({_id:100},{$set : {z:1000}})
```

Realizará una modificación del documento agregando un nuevo atributo z con valor 1000.

```
> db.updtst.find({_id:100})
{ "_id" : 100, "x" : 100, "y" : 100, "z" : 1000 }
```


DBlandIT®

Modificando Documentos Parciales

Update Parciales

Ejemplo 3 – 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
" }
```


DBlandIT®

Modificando Documentos Parciales

Update Parciales

Ejemplo 4 – Operador $set – Opción multi – Cambiar el valor de un atributo en todos los docs.

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

Este comando reemplaza en TODOS los documentos encontrados con valor x:2, reemplazando el valor del atributo z:"NUEVO" por "NW"

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


DBlandIT®

Modificando Documentos Parciales

Update Parciales

Operador $unset

El operador $unset nos permite eliminar un atributo de un documento determinado.

Ejemplos:
Dado el siguiente conjunto de documentos:

```
> db.updtst.find()
{ "_id" : ObjectId("536a93d5793253ebed598070"), "x" : 3, "y" : 200 }
{ "_id" : ObjectId("536a93d8793253ebed598071"), "x" : 3, "y" : 300 }
{ "_id" : ObjectId("536a93e8793253ebed598072"), "x" : 1, "y" : 999 }
{ "_id" : ObjectId("536a93ed793253ebed598073"), "x" : 1, "y" : 200 }
{ "_id" : 300, "items" : [  3,  10 ], "x" : 1, "y" : 200 }
>
```

Ejemplo 1 - Eliminación del atributo "y"del documento con {_id:300}.

```
db.updtst.update({ _id:300}, { $unset: { y: 1})
```

```
>
> db.updtst.update({ _id:300}, { $unset: { y: 1}})
> db.updtst.find()
{ "_id" : ObjectId("536a93d5793253ebed598070"), "x" : 3, "y" : 200 }
{ "_id" : ObjectId("536a93d8793253ebed598071"), "x" : 3, "y" : 300 }
{ "_id" : ObjectId("536a93e8793253ebed598072"), "x" : 1, "y" : 999 }
{ "_id" : ObjectId("536a93ed793253ebed598073"), "x" : 1, "y" : 200 }
{ "_id" : 300, "items" : [  3,  10 ], "x" : 1 }
>
```


DBlandIT®

Modificando Documentos Parciales

Update Parciales

Ejemplo 2 - Eliminación del atributo "y" de todos los documentos de la colección. Uso de la cláusula multi.

```
db.updtst.update({}, { $unset: { y: 1}},{multi:true})
```

```
> db.updtst.update({}, { $unset: { y: 1}},{multi:true})
> db.updtst.find()
{ "_id" : ObjectId("536a93d5793253ebed598070"), "x" : 3 }
{ "_id" : ObjectId("536a93d8793253ebed598071"), "x" : 3 }
{ "_id" : ObjectId("536a93e8793253ebed598072"), "x" : 1 }
{ "_id" : ObjectId("536a93ed793253ebed598073"), "x" : 1 }
{ "_id" : 300, "items" : [  3,  10 ], "x" : 1 }
>
```


DBlandIT®

Modificando Documentos Parciales

Update Parciales

Operador $inc

El operador $inc nos permite incrementar o decrementar el valor de un atributo de un Documento dado.

Dado el siguiente documento:

```
> db.updtst.find({_id:100})
{ "_id" : 100, "x" : 100, "y" : 100, "z" : 1000 }
```

Ejemplo 1 – Operador $inc - incremenal

```
db.updtst.update({_id:100},{$inc : {z:1}})
```

Incrementa en uno el valor del atributo z

```
> db.updtst.update({_id:100},{$inc : {z:1}})
> db.updtst.find({_id:100})
{ "_id" : 100, "x" : 100, "y" : 100, "z" : 1001 }
> db.updtst.update({_id:100},{$inc : {z:1}})
> db.updtst.find({_id:100})
{ "_id" : 100, "x" : 100, "y" : 100, "z" : 1002 }
```


DBlandIT®

Modificando Documentos Parciales

Update Parciales

Operador $inc (Cont.)

El operador $inc nos permite incrementar o decrementar el valor de un atributo de un Documento dado.

Dado el siguiente documento:

```
> db.updtst.find({_id:100})
{ "_id" : 100, "x" : 100, "y" : 100, "z" : 1000 }
```

Ejemplo 2 – Operador $inc - decremenal

```
db.updtst.update({_id:100},{$inc : {z:-1}})
```

decrementa en uno el valor del atributo z

```
> db.updtst.update({_id:100},{$inc : {z:-1}})
> db.updtst.find({_id:100})
{ "_id" : 100, "x" : 100, "y" : 100, "z" : 1001 }
> db.updtst.update({_id:100},{$inc : {z:-1}})
> db.updtst.find({_id:100})
{ "_id" : 100, "x" : 100, "y" : 100, "z" : 1000 }
```


DBlandIT®

Modificando Documentos Parciales

Update Parciales

Cláusula upsert

Esta cláusula hace que ante update, el motor realice la inserción de un nuevo documento si no existe el documento a modificar, o actualice el documento existente.

Ejemplo:

```
db.accesos.update({_id:"/sitioA/login"},{$inc : {visitas:1}}, {upsert:true})
```

ó

```
db.accesos.update({_id:"/sitioA/login"},{ $inc : {visitas:1} }, true)
```

Este update realizará la inserción de un nuevo documento, si no existe, o incrementará en uno las visitas si existe.

Realizamos nuestro primer update sobre un documento inexistente

```
> db.accesos.update({_id:"/sitioA/login"},{$inc : {visitas:1}}, {upsert:true})
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : "/sitioA/login"
})
```

Vemos que se insertó un documento con visitas 1

```
> db.accesos.find()
{ "_id" : "/sitioA/login", "visitas" : 1 }
```


DBlandIT®

Modificando Documentos Parciales

Update Parciales

Cláusula upsert (Cont.)

Si volvemos a ejecutar el mismo comando:

```
> db.accesos.update({_id:"/sitioA/login"},{$inc : {visitas:1}}, {upsert:true})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
> db.accesos.find()
{ "_id" : "/sitioA/login", "visitas" : 2 }
```

Observamos que se ha actualizado en 1 las visitas del documento con el _id indicado.

Ahora, nos viene un nuevo update sobre otra página:

```
> db.accesos.update({_id:"/sitioB/login"},{$inc : {visitas:1}}, {upsert:true})
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : "/sitioB/login"
})
> db.accesos.update({_id:"/sitioB/signup"},{$inc : {visitas:1}}, {upsert:true})
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : "/sitioB/signup"
})
> db.accesos.find()
{ "_id" : "/sitioA/login", "visitas" : 2 }
{ "_id" : "/sitioB/login", "visitas" : 1 }
{ "_id" : "/sitioB/signup", "visitas" : 1 }
```

Observamos que se han insertado dos nuevos documentos con 1 visita cada uno.

DBlandIT®

Modificando Documentos Parciales

Update Parciales

Cláusula upsert (Cont.)

Agregar un nuevo datos a partir de ahora que es la cantidad de "likes".

```
> db.accesos.update({_id:"/sitioA/login"},{$inc : {likes:1}}, {upsert:true})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
> db.accesos.find()
{ "_id" : "/sitioB/login", "visitas" : 1 }
{ "_id" : "/sitioB/signup", "visitas" : 1 }
{ "_id" : "/sitioA/login", "visitas" : 2, "likes" : 1 }
```

Vemos que en el documento indicado ahora hay un nuevo atributo likes con valor 1

A partir de ahora nuestro update puede ser así:

```
> db.accesos.update({_id:"/sitioA/login"},{$inc : {likes:1, visitas:1}}, {upsert
:true})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
> db.accesos.find()
{ "_id" : "/sitioB/login", "visitas" : 1 }
{ "_id" : "/sitioB/signup", "visitas" : 1 }
{ "_id" : "/sitioA/login", "visitas" : 3, "likes" : 2 }
```

En este caso vemos que a partir de ahora incrementó los atributos like y visitas en 1.

DBlandIT®