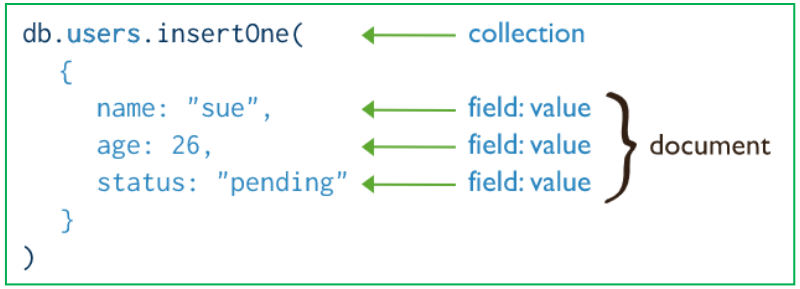
CRUD EN MONGODB

CRUD - Create

Las operaciones CRUD son Create, Read, Update y Delete documentos.

- La operación de creación o inserción añade nuevos documentos a una colección.
- Importante: si la colección no existe actualmente, las operaciones de inserción crearán la colección.
- MongoDB proporciona los siguientes métodos para insertar documentos en una colección:

db.<collection>.insert()
db.<collection>.insertOne()
db.<collection>.insertMany()



- En MongoDB, las operaciones de inserción se aplican sobre una única colección.
- Todas las operaciones de escritura en MongoDB **son atómicas** en el nivel de un solo documento.

CRUD - Create

- insert() inserta uno o varios documentos en una colección. Devuelve la descripción de la operación y el número de registros insertados.

```
> db.inventory.insert( { item: "papas", qty: 120, tags: ["verdu"], size: { h: 30, w: 35.5, uom: "cm" } } )
WriteResult({ "nInserted" : 1 })
```

- insertOne() inserta un único documento en una colección y devuelve un documento que incluye el valor de campo _id del documento recién insertado.

```
> db.inventory.insertOne(
...    { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5ec04285534ad19d22230259")
}
```

CRUD - Create

- insertMany() inserta varios documentos en una colección y devuelve un documento que incluye el valor del campo _id de los documentos recién insertados.

```
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5ec044cd534ad19d2223025b"),
                ObjectId("5ec044cd534ad19d2223025c"),
                ObjectId("5ec044cd534ad19d2223025d")
        ]
}
```

- Las operaciones de inserción crearán la colección si esta no existe.
- Cada documento almacenado en una colección requiere de un campo _id único que actúa como clave principal.
- Si el documento nuevo omite dicho campo, el controlador de MongoDB generará automáticamente un **ObjectId** para el campo _id.

CRUD - Read

- Se acceden a datos de una colección, No hay nada similar al JOIN de SQL
- El método find(buscar): devuelve un cursor para recorrer el resultado.

```
> db.<coleccion>.find( <filtros>, <proyecciones> )
```

```
> db.products.find( )
```

```
> db.products.find
```

```
> db.products.find( {} )
```

```
> db.products.find().pretty()
```

- El método findOne (buscar un único resultado): devuelve el primer resultado de la consulta.

```
> db.<coleccion>.findOne( <filtros>, <proyecciones> )
```

CRUD - Read

```
db.users.find(
   { age: { $gt: 18 } },
   { name: 1, address: 1 }
).limit(5)
```

← collection
← query criteria
← projection
← cursor modifier

- Las consultas en MongoDB devuelven una colección de documentos que cumplan una condición determinada.
- Para consultar o leer los documentos, MongoDB proporciona los siguientes métodos:

```
> db.products.find()
```

```
SELECT * FROM products
```

```
> db.products.find( { status: { $in: [ "A", "D" ] } } )
```

```
SELECT * FROM products WHERE status in ("A", "D")
```

```
> db.products.find( { status: "A", qty: { $lt: 30 } } )
```

```
SELECT * FROM products WHERE status = "A" AND qty < 30
```

Datos de ejemplo (Pág. 3)

https://docs.google.com/document/d/1SAXybtB8xCdM7ifrd_oMVkft664BJ32ZH5k9ZsPrcuY/edit?usp=sharing

CRUD - Read

```
db.users.find(
   { age: { $gt: 18 } },
   { name: 1, address: 1 }
).limit(5)
```

← collection
← query criteria
← projection
← cursor modifier

- El método db.collection.findOne() también realiza una operación de lectura para devolver un solo documento. Internamente, el método db.collection.findOne() es el método db.collection.find() con un límite de 1:

```
> db.products.find( {
   status: "A",
   $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ]
  } )
```

```
SELECT * FROM products WHERE status = "A" AND ( qty < 30 OR item LIKE "p%")
```

Datos de ejemplo (Pág. 3)

https://docs.google.com/document/d/1SAXybtB8xCdM7ifrd_oMVkft664BJ32ZH5k9ZsPrcuY/edit?usp=sharing

CRUD - Read

```
db.users.find(
   { age: { $gt: 18 } },
   { name: 1, address: 1 }
).limit(5)
```

← collection
← query criteria
← projection
← cursor modifier

```
> db.<coleccion>.find( {"field1": "value", "field2": "value" } )
```

```
> use fwd

> db.product.find( {"is_active": true, "weight": 1 } )

> db.product.find(…).limit(n).pretty()       //n = número registros
```

```
> db.product.findOne( {"is_active": true, "weight": 1 } )

> db.product.findOne(…).pretty()
```

CRUD - Read

```
...
"price": 332.20,
"variants" : {
    "1" : { "name" : "Blue, Small", "color": "blue" },
    "2" : { "name" : "Black, Small", "color": "orange },
    "3" : { "name" : "Pink, Small", "color": "yellow" }
},
"is_active" : true,
...
```

- Para especificar una condición de igualdad en un campo que es un documento incrustado o anidado, use el documento de filtro de consulta {<campo>: <valor>} y asigne a <valor> el documento que quiere buscar o consultar.

```
> db.products.find( {"variants.1": { "name": "Blue, Small", "color": "red" } } )
```

- Las coincidencias de igualdad en todo el documento incrustado requieren una coincidencia exacta del documento <valor> especificado, incluido el orden de los campos. Por ejemplo, la siguiente consulta no coincide con ningún documento en la colección de inventario:

```
> db.products.find( { "variants.1": { "color": "red", "name": "Blue, Small" } } )
```

- Cuando utilice la notación de punto, el campo y el campo anidado deben estar entre comillas:

CRUD - Read

Aportan una nueva dimensión los datos

- Permiten almacenar "varios registros" dentro de uno.
  - En bases de datos relacionales necesitaríamos utilizar varias tablas la mayoría de las veces.
- Esta disposición, complican las operaciones.
  - Siempre se debe recordar que la unidad básica de trabajo son los documentos.

Notación de puntos (dot notation)

- Permite acceder a valores que estén dentro de listas o subdocumentos:
  - Listas: Índice del elemento

```
"category_ids.0"
```

- Subdocumentos:
  - Clave del elemento

```
"variants.1.name"
```

CRUD - Read

```
"variants.1.name"
```

```
"category_ids.0"
```

```
{
        "_id": ObjectId("53f8659a957cfd6a0a8b4595"),
        "name": "The North Face McMurdo II Boot ...",
        "primary_category_id": 8,
        "price": 135.95,
        "variants": {
        "1": {
                "name": "Brown, 7"
        },
        "2": {
                "name": "Black, 7"
        }
        },
        "category_ids": [ 8, 15 ],
        "date_created": ISODate("2014-08-23T09:57:46Z")
}
```

CRUD - Read

Búsquedas en listas

- Es posible realizar búsquedas exactas o de elementos puntuales
  - Coincidencia exacta:

```
{ "category_ids": [8, 15] }
```

  - Coincidencia exacta:

```
{ "category_ids": 8 }
```

```
{
        "_id": ObjectId("53f8659a957cfd6a0a8b4595"),
        "name": "The North Face McMurdo II Boot ...",
        "primary_category_id": 8,
        "price": 135.95,
        "variants": {
        "1": {
                "name": "Brown, 7"
        },
        "2": {
                "name": "Black, 7"
        }
        },
        "category_ids": [ 8, 15 ],
        "date_created": ISODate("2014-08-23T09:57:46Z")
}
```

CRUD - Read

Búsquedas en subdocumentos

- Es posible realizar búsquedas exactas o de elementos puntuales
  - Coincidencia exacta:

```
{ "variants": {
      "1" : { "name": "Brown, 7" },
      "2" : { "name": "Black, 7" }
  }
}
```

  - Coincidencia exacta:

```
{ "variants.2.name": "Black, 7" }
```

```
{
        "_id": ObjectId("53f8659a957cfd6a0a8b4595"),
        "name": "The North Face McMurdo II Boot ...",
        "primary_category_id": 8,
        "price": 135.95,
        "variants": {
        "1": {
                "name": "Brown, 7"
        },
        "2": {
                "name": "Black, 7"
        }
        },
        "category_ids": [ 8, 15 ],
        "date_created": ISODate("2014-08-23T09:57:46Z")
}
```

CRUD - Read

Obtener subconjunto de resultados

Métodos asociados al cursor

limit (limitar)

- Obtener como máximo X resultados de una consulta
- Mejora en rendimiento, evita recorrer resultados extra

skip (saltar)

- Ignorar los primeros X resultados de una consulta
- 

No envía o muestra los resultados ignorados, pero sí los recorre.

```
> db.products.find().skip( 5 ).limit( 5 )
```

```
db.library.insertMany([
          { item: "journal", qty: 25, tags: ["blank", "red"], dim_cm: [ 14, 21 ] },
          { item: "notebook", qty: 50, tags: ["red", "blank"], dim_cm: [ 14, 21 ] },
          { item: "paper", qty: 100, tags: ["red", "blank", "plain"], dim_cm: [ 14, 21
          { item: "planner", qty: 75, tags: ["blank", "red"], dim_cm: [ 22.85, 30 ] },
          { item: "postcard", qty: 45, tags: ["blue"], dim_cm: [ 10, 15.25 ] }
       ]);
```

CRUD - Read

- Para especificar la condición de igualdad en un array, se debe usar como <valor> un array exacto al que se quiere buscar, teniendo en cuenta el orden de los elementos.

```
> db.library.find( { tags: ["red", "blank"] } )
```

- Si lo que desea es consultar un array que contenga los elementos "rojo" y "en blanco", sin tener en cuenta el orden u otros elementos del array, use el operador $all:

```
> db.library.find( { tags: { $all: ["red", "blank"] } } )
```

- Para consultar si el campo de array contiene al menos un elemento con el valor especificado, use el filtro {<campo>: <valor>} donde <valor> es el valor del elemento:

```
> db.library.find( { tags: "red" } );
```

```
db.library.insertMany([
          { item: "journal", qty: 25, tags: ["blank", "red"], dim_cm: [ 14, 21 ] },
          { item: "notebook", qty: 50, tags: ["red", "blank"], dim_cm: [ 14, 21 ] },
          { item: "paper", qty: 100, tags: ["red", "blank", "plain"], dim_cm: [ 14, 21
          { item: "planner", qty: 75, tags: ["blank", "red"], dim_cm: [ 22.85, 30 ] },
          { item: "postcard", qty: 45, tags: ["blue"], dim_cm: [ 10, 15.25 ] }
       ]);
```

CRUD - Read

- Para especificar condiciones en los elementos en el campo del array, puede usar operadores de consulta en el documento de filtro de consulta.
- El ejemplo anterior, permite consultar todos los documentos donde el array dim_cm contiene al menos un elemento cuyo valor es mayor que 23.

```
> db.library.find( { dim_cm: { $gt: 23 } } )
```

- Al especificar condiciones compuestas en los array, es posible indicar en la consulta que un solo elemento cumpla con la condición o cualquier combinación de elementos del array cumpla con las condiciones:

```
> db.library.find( { dim_cm: { $gt: 15, $lt: 20 } } )
```

```
db.library.insertMany([
          { item: "journal", qty: 25, tags: ["blank", "red"], dim_cm: [ 14, 21 ] },
          { item: "notebook", qty: 50, tags: ["red", "blank"], dim_cm: [ 14, 21 ] },
          { item: "paper", qty: 100, tags: ["red", "blank", "plain"], dim_cm: [ 14, 21
          { item: "planner", qty: 75, tags: ["blank", "red"], dim_cm: [ 22.85, 30 ] },
          { item: "postcard", qty: 45, tags: ["blue"], dim_cm: [ 10, 15.25 ] }
       ]);
```

CRUD - Read

- Con ayuda del operador $elemMatch se pueden especificar varios criterios en los elementos de una array de modo que al menos uno de ellos cumpla todos los criterios especificados.

```
> db.library.find( { dim_cm: { $elemMatch: { $gt: 20, $lt: 29 } } } )
```

- Utilice la notación de punto para indicar condiciones de consulta sobre un elemento concreto del array. Para ello utilice el índice o posición particular del array. Los arrays usa indexación basada en cero, igual que Java.
- La consulta también puede hacerse a través del tamaño del array. Para ello, utilice el operador $size para consultar array de un número de elementos concreto.

```
> db.library.find( { "dim_cm.1": { $gt: 25 } } )
```

```
> db.library.find( { "tags": { $size: 3 } } )
```


https://docs.google.com/document/d/1SAXybtB8xCdM7ifrd_oMVkft664BJ32ZH5k9ZsPrcuY/edit?usp=sharing

Datos de ejemplo (Pág. 3)

```
db.library.insertMany( [
   { item: "journal", instock: [ { warehouse: "A", qty: 5 }, { warehouse: "C", qty: 15 }
   { item: "notebook", instock: [ { warehouse: "C", qty: 5 } ] },
   { item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 15 }
   { item: "planner", instock: [ { warehouse: "A", qty: 40 }, { warehouse: "B", qty: 5 }
   { item: "postcard", instock: [ { warehouse: "B", qty: 15 }, { warehouse: "C", qty: 35
]);
```

CRUD - Read

- Compruebe las siguientes consultas:

```
> db.library.find( { "instock": { warehouse: "A", qty: 5 } } )
> db.library.find( { "instock": { qty: 5, warehouse: "A" } } )
> db.library.find( { 'instock.qty': { $lte: 20 } } )
> db.library.find( { 'instock.0.qty': { $lte: 20 } } )
> db.library.find( { "instock": { $elemMatch: { qty: 5, warehouse: "A" } } } )
> db.library.find( { "instock": { $elemMatch: { qty: { $gt: 10, $lte: 20 } } } } )
> db.library.find( { "instock.qty": { $gt: 10,  $lte: 20 } } )
> db.library.find( { "instock.qty": 5, "instock.warehouse": "A" } )
```


https://docs.google.com/document/d/1SAXybtB8xCdM7ifrd_oMVkft664BJ32ZH5k9ZsPrcuY/edit?usp=sharing

Datos de ejemplo (Pág. 3)

```
db.library.insertMany( [
  {item: "journal", status: "A", size: { h: 14, w: 21, uom: "cm" }, instock: [{ warehouse: "A", qty: 5 }]},
  {item: "notebook", status: "A",  size: { h: 8.5, w: 11, uom: "in" }, instock: [{ warehouse: "C", qty: 5 }]},
  {item: "paper", status: "D", size: { h: 8.5, w: 11, uom: "in" }, instock: [{ warehouse: "A", qty: 60 }]},
  {item: "planner", status: "D", size: { h: 22.85, w: 30, uom: "cm" }, instock: [{ warehouse: "A", qty: 40 }]},
  {item: "postcard", status: "A", size: { h: 10, w: 15.25, uom: "cm" }, instock: [{ warehouse: "B", qty: 15 },
        { warehouse: "C", qty: 35 } ] }
]);
```

CRUD - Read

- Compruebe las siguientes consultas:

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

```
> db.library.find( { status: "A" }, { item: 1, status: 1 } )
```

```
> db.library.find( { status: "A" }, { item: 1, status: 1, _id: 0 } )
```

```
> db.library.find( { status: "A" }, { item: 1, status: 0, _id: 0 } )
```

```
> db.library.find( { status: "A" }, { status: 0, instock: 0 } ) //excluir solo estos
```

```
> db.library.find(
      { status: "A" },
      { item: 1, status: 1, "size.uom": 1 }
  )
```

CRUD - Read

Proyecciones

- Permite definir qué campos se incluirán en los resultados.
  - Por defecto, se traen todos los campos.
- Objeto con campos a incluir o excluir del resultado:
  - La clave indica el nombre campo.
  - Valores a 0: se trae todo menos dichos campos.
  - Valores a 1: sólo se traen esos campos.
  - El _id siempre viene, salvo que se excluya explícitamente.
  - No se pueden mezclar valores a 0 y a 1, excepto el _id.

```
> db.products.find( )
```

```
> db.products.find( {} )
```

CRUD - Read

Proyecciones

- Excluir el campo "description":

```
> db.products.find( {}, { "description": 0 } )
```

- Obtener solo el campo "Price":

```
> db.products.find( {}, { "price": 1, "_id": 0 } )
```

```
> db.products.findOne( {}, { "description": 0 } )
```

```
> db.products.findOne( {}, { "price": 1, "_id": 0 } )
```

CRUD - Read

Ordenación

- Método asociado al cursor ¿qué significa esto?
- Por defecto, los datos se obtienen en el orden natural.
  - Orden en que están almacenados en disco
- Objeto con campos a utilizar
  - Es relevante el orden de los campos.
  - Valor 1 para orden ascendente, -1 para descendente.

```
> db.products.find().sort( { "price": -1, "name": 1 } )
```

CRUD – Read - Operadores

- Cadenas que empiezan con **$** con semántica predefinida.
- Aplicable a filtros, proyecciones, modificaciones y otros.

Ejemplos

- Obtener todos los productos que cuestan más de €50

```
> db.products.find({ "price": { "$gt": 50 } })
```

- Obtener los productos que están en la categoría 8 o que no están activos

```
> db.products.find( { "$or": [ { "category_ids": 8 }, { "is_active": false } ] } )
```

CRUD - Read - Operadores

Comparaciones

`$gt, $gte, $lt, $lte, $ne, $in, $nin`

```
> db.products.find( { "primary_category_id": { "$nin": [ 4, 6, 12 ] } } )
```

Lógicos

`$or, $nor, $and, $not`

```
> db.products.find( { price: { $not: { $gt: 1.99 } } } )
```

Campos

`$exists, $type`

Consulta la diapositiva "Tipos de datos"

```
> db.products.find( { "price": { "$type": 2 } } )
```

Tipos de datos ($type)

A partir de MongoDB 3.2, el operador $type acepta alias de cadena para los tipos BSON además de los números correspondientes a los tipos BSON.

Las versiones anteriores solo aceptaban los números correspondientes al tipo BSON.

Importante:

Al comparar valores de diferentes tipos de BSON, MongoDB utiliza el siguiente orden de comparación, de menor a mayor:

Tipo	Número	Alias	Cuidado
Double	1	“double”	
String	2	“string”	
Object	3	“object”	
Array	4	“array”	
Binary data	5	“binData”	
Undefined	6	“undefined”	Deprecated.
ObjectId	7	“objectId”	
Boolean	8	“bool”	
Date	9	“date”	
Null	10	“null”	
Regular Expression	11	“regex”	
DBPointer	12	“dbPointer”	Deprecated.
JavaScript	13	“javascript”	
Symbol	14	“symbol”	Deprecated.
JavaScript (with scope)	15	“javascriptWithScope”	
32-bit integer	16	“int”	
Timestamp	17	“timestamp”	
64-bit integer	18	“long”	
Decimal128	19	“decimal”	New in version 3.4.
Min key	-1	“minKey”	Tipo interno
Max key	127	“maxKey”	Tipo interno

CRUD - Read - Operadores

Listas

$all, $elemMatch, $size

```
> db.products.find( { "category_ids": { "$all": [ 8, 15 ] } } )
```

Avanzados

$mod, $regex, $where

```
> db.products.find( { "sku": { "$not": { "$regex": "\w{3}\d{4}" } } } )
```

Índices especiales

$geoWithin, $geoIntersects, $near, $nearSphere, $text

```
>db.products.find({"$where":"this.category_ids.indexOf(this.primary_category_id)== -1"})
```

CRUD - Read - Operadores

Comparación

Operador	Descripción
$eq	Compara valores que son iguales a un valor especifico.
$gt	Compara valores que son mayores que un valor específico.
$gte	Compara valores que son mayor o igual que un valor específico.
$in	Coincide con cualquiera de los valores especificados en una matriz.
$lt	Compara valores que son menores que un valor específico.
$lte	Compara valores que son menores o igual que un valor específico.
$ne	Coincide con todos los valores que no son iguales a un valor concreto.
$nin	No coincide con ninguno de los valores especificados en una matriz.

Lógicos y de Elementos

Operador	Descripción
$and	Une cláusulas de consulta con un AND lógico y devuelve todos los documentos que coinciden con las condiciones de ambas cláusulas.
$or	Une las cláusulas de consulta con un OR lógico y devuelve todos los documentos que coinciden con las condiciones de cualquiera de las cláusulas.
$not	Invierte el efecto de una expresión de consulta y devuelve documentos que no coinciden con la expresión de consulta.
$nor	Unir las cláusulas de consulta con un NOR lógico devuelve todos los documentos que no coinciden con ambas cláusulas.
$exists	Coincide con documentos que tienen el campo especificado.
$type	Selecciona documentos si un campo es del tipo especificado.

CRUD - Read - Operadores

Evaluación

Operador	Descripción
$expr	Permite el uso de expresiones de agregación dentro del lenguaje de consulta.
$jsonSchema	Validar documentos contra el esquema JSON dado.
$mod	Realiza una operación de módulo sobre el valor de un campo y selecciona documentos con un resultado específico.
$regex	Selecciona documentos donde los valores coinciden con una expresión regular específica.
$text	Realiza búsqueda de texto.
$where	Coincide con documentos que satisfacen una expresión de JavaScript.

Array y Proyección

Operador	Descripción
$all	Coincide con matrices que contienen todos los elementos especificados en la consulta.
$elemMatch	Selecciona documentos si el elemento en el campo de matriz coincide con todas las condiciones especificadas de $elemMatch.
$size	Selecciona documentos si el campo de matriz tiene un tamaño especificado.
$	Proyecta el primer elemento en una matriz que coincide con la condición de consulta.
$meta	Proyecta la puntuación del documento asignada durante la operación $ text.
$slice	Limita el número de elementos proyectados desde una matriz. Admite saltos y límite.

CRUD - Read - Operadores

Comparación: $eq

```
{ <field>: { $eq: <value> } }
```

```
{ field: <value> }
```

```
> db.products.find( { price: { $eq: 339.95 } } )
```

```
> db.products.find( { price: 339.95 } )
```

```
> db.products.find( { "variants.1.name": { $eq: "Blue, Small" } } )
```

```
> db.products.find( { "variants.1.name": "Blue, Small" } )
```

```
> db.products.find( { "category_ids.0": { $eq: 7 } } )
```

```
> db.products.find( { "category_ids.0": 7 } )
```

CRUD - Read - Operadores

Comparación: $gt, $gte, $lt, $lte, $ne

```
{ <field>: { $eq: <value> } }
```

```
> db.products.find( { price: { $gt: 339.95 } } )
```

```
> db.products.find( { "variants.1.name": { $eq: "Blue, Small" } } )
```

```
> db.products.find( { "category_ids.0": { $gt: 5 } } )
```

CRUD - Read - Operadores

Comparación: $in, $nin

```
{ field: { $in: [<value1>, <value2>, ... <valueN> ] } }
```

```
> db.products.find( { price: { $in: [100, 200]} } )
```

```
> db.products.find( { "category_ids.0": { $in: [1, 9] } } )
```

```
> db.products.find( { "name": { $in: [ /^Bu/, /^Be/ ] } } ) //$regex
```

CRUD - Read - Operadores

Lógicos: $and, $or

```
{ $and: [ { <expression1> }, { <expression2> } , ... , { <expressionN> } ] }
```

```
{ $or: [ { <expression1> }, { <expression2> } , ... , { <expressionN> } ] }
```

```
> db.products.find({ $and: [ { price: { $ne: 1.99 } }, { price: { $exists: true } } ] } )
```

```
> db.products.find({ price: { $ne: 1.99, $exists: true } })
```

```
> db.inventory.find( {
    $and : [
        { $or : [ { price : 0.99 }, { price : 1.99 } ] },
        { $or : [ { is_active : true }, { weight : { $lt : 2 } } ] }
    ]
} )
```

CRUD - Read - Operadores

Lógicos: $not, $nor

```
{ field: { $not: { <operator-expression> } } }
```

```
{ $nor: [ { <expression1> }, { <expression2> }, ...  { <expressionN> } ] }
```

```
> db.products.find( { price: { $not: { $gt: 1.99 } } } )
```

```
> db.products.find( { "name": { $not: { $regex: /^p.*/ } } } ) //versión: {$lte: 4.0}
```

```
> db.products.find( { "name": { $not: /^p.*/ } } )
```

```
> db.products.find( { $nor: [ { price: 1.99 }, { sale: true } ] } )
```

CRUD - Read - Operadores

Elementos: $type, $exists

```
{ field: { $type: <BSON type> } }
```

```
{ field: { $type: [ <BSON type1> , <BSON type2>, ... ] } }
```

```
> db.products.find( { "weight" : { $type : "number" } } )
```

```
> db.products.find( { "name" : { $type : "string" } } )
```

```
> db.products.find( { "category_ids" : { $type : "array" } } ) //versión: {$lte: 3.6}
```

CRUD - Read - Operadores

Evaluación: $expr

```
{ $expr: {<expression> } } //versión: {$lte: 3.6}
```

```
> db.products.find( { $expr : { $gt : ["$weight", "$price"] } } )
```

```
> db.products.find( { "name" : { $type : "string" } } )
```

CRUD - Read - Operadores

Evaluación: $mod, $text

```
{ field: { $mod: [ divisor, remainder ] } }
```

```
{
  $text:
    {
      $search: <string>,
      $language: <string>,
      $caseSensitive: <boolean>,
      $diacriticSensitive: <boolean>
    }
}
```

```
> db.products.find( { weight: { $mod: [ 4, 0 ] } } )
```

```
> db.products.find( { $text: { $search: "bueno" } } )
```

```
> db.products.find( { $text: { $search: "\"buen amigo\"" } } )
```

```
> db.products.find( { $text: { $search: "buen", $language: "es" } } ) //no stopword
```

```
> db.products.find( { $text: { $search: "buen", $caseSensitive: true } } )
```

CRUD - Read - Operadores

Evaluación: $where

El operador $where permite pasar una expresión o función JavaScript al sistema de consulta.
$where proporciona una mayor flexibilidad, pero requiere que la base de datos procese la expresión o función de JavaScript para cada documento de la colección.
Es recomendable que haga referencia al documento en la expresión o función JavaScript utilizando this u obj

```
> db.products.find( { $where: function() {
   return ( this.weight >= 1 )
  } } );
```

- $expr es más rápido que $where porque no ejecuta JavaScript. Use $expr siempre que sea posible.
- $where no puede utilizar lo índices creados, la consula va mucho mejor si utiliza $gt, $in, etc.
- Si necesita manipular sus datos con un operador que no exista, recurra a $where.

CRUD - Read - Shema

**Evaluación: $jsonSchema** //versión: {$lte: 3.6}

```
{ $jsonSchema: <json schema object> }
```

```
> db.products.find( { $jsonSchema: <schema> } )
```

```
{
  $jsonSchema: {
     required: [ "name", "major", "gpa", "address" ],
     properties: {
        name: {
           bsonType: "string",
           description: "must be a string and is required"
        },
        address: {
           bsonType: "object",
           required: [ "zipcode" ],
           properties: {
               "street": { bsonType: "string" },
               "zipcode": { bsonType: "string" }
           }
        }
     }
  }
}
```

```
> db.createCollection( <collection>, { validator: { $jsonSchema: <schema> } } )
```

Muchas gracias por tu atención