Índices.

Por defecto los índices en MongoDB operan de la misma manera que en la mayoría de los sistemas SQL, es decir, nos permiten optimizar el propiedades (fields) para ser usados como criterio de búsqueda o para ordenar resultados.
Como bien hemos visto en el capítulo anterior, el índice por defecto que se crea en cada documento es el _id.
*Los índices en MongoDB usan una estructura de datos B-Tree

Creando índices.

Cuando creamos un índice dentro de una collection debemos indicar el field y el tipo de sort (si es ascendiente o descendiente).

```
> db.dispatches.createIndex( {Service:1 })
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

En el ejemplo anterior creamos un nuevo índice del field Service en forma ascendente.

```
> db.dispatches.createIndex( {Service:-1 })
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "ok" : 1
}
```

En el ejemplo anterior creamos un índice sobre el field "Service" en forma descendiente.

Índices combinados.

Podemos crear índices combinados (con mas de un field), solo basta con indicar los campos que deseamos combinar.

```
> db.dispatches.createIndex( { Lat: 1, Long:1  } );
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 3,
        "numIndexesAfter" : 4,
        "ok" : 1
}
```

Índices de fields con documentos embebidos.

No es necesario que los índices que deseamos crear sean de fields con valores fijos, ya que podemos crear índices con field con documentos embebidos.

```
> db.dispatches.createIndex( { Candidates:1   } );
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 4,
        "numIndexesAfter" : 5,
        "ok" : 1
}
```

Índices multikey.

MongoDB utiliza índices multikey para los valores almacenados en los fields con valores del tipo Array.
Por cada uno de los elementos almacenados en los Array MongoDB generará índices por separado, sean o no documentos lo que están almacenados en los Arrays.
También es posible crear un índice sobre un field de un documento almacenado en un Array.

```
> db.dispatches.createIndex( { 'Candidates.Movil':1   } );
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 5,
        "numIndexesAfter" : 6,
        "ok" : 1
}
```

Índices Text.

MongoDB nos permite crear índices del tipo Text sobre cualquier field para optimizar las búsquedas de textos. Podemos tener mas de un índice Text en una collection.
Veamos cómo crear un índice del tipo Text sobre el field "zonaName".

```
> db.dispatches.createIndex( {ZonaName: 'text'} );
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 5,
        "numIndexesAfter" : 6,
        "ok" : 1
}
```

Operadores.

A continuación seguiremos viendo operadores con el fin de poder realizar query más complejas de las que veníamos desarrollando.
Para este grupo nuevo de operadores seguiremos utilizando la collection "dispatches".

$exists

El operador $exists se utiliza para indicar como condición si una propiedad debe o no existir.

```
> db.dispatches.find( {lastTry: {$exists: true} }, {lastTry:1, _id:0})
{ "lastTry" : "2013-09-04T13:15:27.324Z" }
{ "lastTry" : "2013-09-04T13:12:52.280Z" }
{ "lastTry" : "2013-09-04T13:16:37.378Z" }
{ "lastTry" : "2013-09-04T13:27:54.082Z" }
{ "lastTry" : "2013-09-04T13:16:12.345Z" }
{ "lastTry" : "2013-09-04T13:17:47.384Z" }
…
```

$type

El operador $type se utiliza para establecer como condición que una propiedad debe ser de un tipo de datos BSON específico.
La forma de indicar el tipo de propiedad (el valor válido para $type) debe ser por el nombre del tipo de datos BSON o su número (código de tipo de dato).
Observemos la siguiente tabla:

Type	Number	Alias
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"
JavaScript	13	"javascript"
Symbol	14	"symbol"
JavaScript (with scope)	15	“javascriptWithScope”
32-bit integer	16	"int"
Timestamp	17	"timestamp"
64-bit integer	18	"long"
Min key	19	"minKey"
Max key	20	"maxKey"

Cuando deseamos hacer match con una propiedad del tipo Array, debemos tener en cuenta que el operador intentará hacer match con cada uno de los elementos del Array y no con la propiedad en sí.

```
> db.dispatches.find( { Service: { $type: 16} }, {Service: 1, _id:0});
{ "Service" : 70208 }
{ "Service" : 70209 }
>
```

Operadores de evaluación.

Los operadores de evaluación son aquellos que reciben como valor una expresión a evaluar al fin de obtener una condición más compleja.

$mod

El operador $mod se utiliza para establecer como condición que el módulo de una propiedad X debe dar como resto N.

```
> db.dispatches.find( { Service: { $mod: [2,0]} }, {Service: 1, _id:0});
{ "Service" : 70208 }
> db.dispatches.find( { Service: { $mod: [3,0]} }, {Service: 1, _id:0});
{ "Service" : 70209 }
>
```

$regex

El operador $regex se utiliza para indicar como condición que la propiedad debe cumplicar con una expresion regular. MongoDB usa expresiones regulares compatibles con PERL, es decir, PCRE.

Las formas de uso válidas son:
{ <field>: { $regex: /pattern/, $options: '<options>' } }
{ <field>: { $regex: 'pattern', $options: '<options>' } }
{ <field>: { $regex: /pattern/<options> } }

$options.

El operador $options se utiliza para pasar opciones a la expresión regular.

Opción	Descripción.
i	Indicar que no debe ser case-sensitive

m	Para expresiones que contiene ^ para el inicio y $ para el fin, haciendo match el inicio y fin de cada una de las líneas.
x	Ignora los espacios en blanco.

Veamos el siguiente ejemplo en el cual buscamos todos los documentos en los cuales la propiedad calleName hace match con la expresión "/juan/" sin importar el uso de mayúsculas/minúsculas.

```
> db.dispatches.find( { calleName: {$regex: /juan/, $options: 'i'} }, {calleName:1, _id: 0} );
{ "calleName" : "JUAN MARIA GUTIERREZ" }
{ "calleName" : "JUAN SANCHEZ" }
>
```

$text

El Operador $text se utiliza para realizar búsqueda de texto sobre índices del tipo Text.
Modelo de uso:

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

Propiedad.	Descripción.
$search	El término a buscar.
$language	Opcional: indicamos el idioma en el que vamos a buscar, de esa forma sabe que stopwords debe usar. Leer mas...
$caseSensitive	Opcional (boolean): indica si vamos o no a buscar en forma sensitiva (distinguir entre mayúsculas y minúsculas). Por defecto la propiedad es false.

$diacriticSensitive	Opcional (boolean): por default es false.

Ejemplo:

```
> db.dispatches.find( { $text: { $search: "\"Base\"" }  }, {ZonaName:1, _id: 0} );
{ "ZonaName" : "BASE 9 (SEGUI Y DORREGO - 21Hs a 06Hs SEGUI E ITALIA)" }
{ "ZonaName" : "BASE 9 (SEGUI Y DORREGO - 21Hs a 06Hs SEGUI E ITALIA)" }
{ "ZonaName" : "BASE 9 (SEGUI Y DORREGO - 21Hs a 06Hs SEGUI E ITALIA)" }
{ "ZonaName" : "BASE 9 (SEGUI Y DORREGO - 21Hs a 06Hs SEGUI E ITALIA)" }
{ "ZonaName" : "BASE 9 (SEGUI Y DORREGO - 21Hs a 06Hs SEGUI E ITALIA)" }
{ "ZonaName" : "BASE 9 (SEGUI Y DORREGO - 21Hs a 06Hs SEGUI E ITALIA)" }
{ "ZonaName" : "BASE 9 (SEGUI Y DORREGO - 21Hs a 06Hs SEGUI E ITALIA)" }
{ "ZonaName" : "BASE 10 (SAN NICOLAS Y 27 DE FEBRERO)" }
{ "ZonaName" : "BASE 14 (27 FE FEBRERO Y NECOCHEA)" }
{ "ZonaName" : "BASE 19 (SAN MARTIN Y MUÑOZ)" }
{ "ZonaName" : "BASE 19 (SAN MARTIN Y MUÑOZ)" }
{ "ZonaName" : "BASE 19 (SAN MARTIN Y MUÑOZ)" }
{ "ZonaName" : "BASE 51 (JUNIN Y ALBERDI)" }
{ "ZonaName" : "BASE 2 (GRANDOLI Y GUTIERREZ)" }
{ "ZonaName" : "BASE 5 (ESPAÑA Y URIBURU)" }
{ "ZonaName" : "BASE 3 (HOSPITAL SAENZ PEÑA)" }
{ "ZonaName" : "BASE 4 (OROÑO Y LAMADRID)" }
{ "ZonaName" : "BASE 4 (OROÑO Y LAMADRID)" }
{ "ZonaName" : "BASE 4 (OROÑO Y LAMADRID)" }
{ "ZonaName" : "BASE 5 (ESPAÑA Y URIBURU)" }
```

$where

El operador $where se utiliza para establecer como condición una expresión JavaScript.
La condición será verdadera si la expresión JS devuelve true.
El operador "this" representa a cada uno de los documentos.

```
> db.dispatches.find( {$where: "this.createdBy==976" } , {createdBy:1, calleName:1 ,_id:0});
{ "calleName" : "MORENO", "createdBy" : 976 }
{ "calleName" : "AVELLANEDA", "createdBy" : 976 }
{ "calleName" : "PARAGUAY", "createdBy" : 976 }
{ "calleName" : "ITUZAINGO", "createdBy" : 976 }
{ "calleName" : "CORRIENTES", "createdBy" : 976 }
{ "calleName" : "COLON", "createdBy" : 976 }
{ "calleName" : "BATLLE Y ORDONEZ", "createdBy" : 976 }
{ "calleName" : "ESPAÑA", "createdBy" : 976 }
{ "calleName" : "CORRIENTES", "createdBy" : 976 }
```

```
{ "calleName" : "ARIJON", "createdBy" : 976 }
{ "calleName" : "CABILDO", "createdBy" : 976 }
{ "calleName" : "DARACT", "createdBy" : 976 }
{ "calleName" : "GARAY", "createdBy" : 976 }
...
```

O usando una function:

```
> db.dispatches.find( { $where: function(){ return this.createdBy==976  } } ,
{createdBy:1,_id:0 } );
{ "createdBy" : 976 }
{ "createdBy" : 976 }
{ "createdBy" : 976 }
{ "createdBy" : 976 }
{ "createdBy" : 976 }
{ "createdBy" : 976 }
{ "createdBy" : 976 }
```

Métodos de collections.

insertMany

Recibe un Array como argumento e inserta cada uno de los documentos contenidos en el Array como documento separados.

```
db.alumnos.insertMany([{added: new Date(), name: 'Alejandro', age:23},
{added: new Date(), name: 'Martin', age:22},
{added: new Date(), name: 'Carlos', age:18},
{added: new Date(), name: 'Juan', age:31},
{added: new Date(), name: 'Pedro', age:28}]);

db.alumnos.find( {}, {});
```

dropIndex

El método dropIndex elimina un índice (podemos indicar el tipo).

```
> db.dispatches.dropIndex( { Candidates:1 } );
{ "nIndexesWas" : 6, "ok" : 1 }
>
```

getIndexes

Devuelve todos los índices de una collection.

```
> db.alumnos.getIndexes()
[
        {
                "v" : 1,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "cursodb.alumnos"
        }
]
>
```