Fechas en MongoDB

MongoDB es una base de datos muy potente a la hora de utilizar valores de fecha. En este artículo analizaremos cómo gestionar correctamente las fechas y cómo encontrar los datos en base a fechas.

Agregar fechas

La forma más sencilla de insertar una fecha es la siguiente:

```
> db.coleccion.insert({"fecha" : new Date()})
```

El comando anterior agrega un nuevo documento a la colección, con un campo fecha que contendrá la fecha y hora actuales. Si listamos el contenido del documento, obtendremos un resultado como el siguiente:

```
> db.coleccion.find()

{ "_id" : ObjectId("5443e87e5894b090804c822e"), "fecha" : ISODate("2014-10-19T16:36:14.197Z") }
```

El tipo de dato `ISODate` contendrá la fecha en formato internacional (año-mes-día), seguido de la hora con precisión de milésimas de segundo.

Para agregar una fecha específica sin especificar la hora, se puede utilizar la función `Date()` con el siguiente formato:

```
> db.coleccion.insert({fecha: new Date("2014, 10, 21")})
```

Si queremos agregar una fecha y hora concretas, podemos utilizar el tipo `ISODate()` como función para especificar el valor:

```
> db.coleccion.insert({fecha: ISODate("2014-10-19T18:56:20.197Z")})
```

Búsqueda de fechas

Para hacer una búsqueda exacta (por fecha y hora), podemos utilizar el siguiente formato:

```
> db.coleccion.find({"fecha" : ISODate("2014-10-20T22:00:00Z")})

{ "_id" : ObjectId("5443f41b5894b090804c8234"), "fecha" : ISODate("2014-10-20T22:00:00Z") }
```

Si deseamos buscar todos los documentos a partir de una fecha específica:

```
> db.coleccion.find({"fecha" : {"$gt" : ISODate("2014-10-18T00:00:00")}})
```

También podemos utilizar la siguiente sintaxis:

```
> db.coleccion.find({"fecha" : {"$gt" : new Date("2014-10-18")}})
```

En ambos casos indicamos a la función `find()` (*encontrar*) que compare el campo `"fecha"` con el valor a buscar, que es un nuevo documento que contiene el operador `"$gt"` (*greater than* ó *mayor que*) y la fecha a partir de la cual debe hacer coincidir el resultado.

Los operadores condicionales utilizados por **MongoDB** para evaluar las condiciones de los valores a encontrar son los siguientes:

- `$gt:` *greater than* (*mayor que*)
- `$gte:` *greater than or equal* (*mayor o igual que*)
- `$lt:` *less than* (*menor que*)
- `$lte:` *less than or equal* (*menor o igual que*)
- `$ne:` *not equal* (*distinto de*)
- `$in:` *in* (*dentro de*(un array)
- `$nin:` *not in* (*no dentro de* (un array))

Por ejemplo, si deseamos encontrar aquellos documentos que se encuentren entre dos fechas dadas:

```
> start = ISODate("2014-10-18T00:00:00Z")
ISODate("2014-10-18T00:00:00Z")
> end = ISODate("2014-10-20T00:00:00Z")
ISODate("2014-10-20T00:00:00Z")
> db.coleccion.find({"fecha" : {"$gte" : start, "$lte" : end}})
```

Al encadenar una lista de criterios a evaluar, por defecto asumirá que se aplica una operación lógica `"$and"`. Lo anterior sería lo mismo que este comando:

```
> db.coleccion.find({"$and" : [{"fecha" : {"$gte" : start}}, {"fecha" : {"$lte" :
end}}]})
```

Si queremos omitir el rango anterior utilizaríamos el operador lógico `"$or"`:

```
> db.coleccion.find({"$or" : [{"fecha" : {"$lt" : start}}, {"fecha" : {"$gt" : end}}]})
```

Los operadores lógicos utilizados por **MongoDB** son los siguientes:

- `$and:` Todas las condiciones del array se deben cumplir
- `$or:` Se debe cumplir, al menos, una condición del array
- `$not:` Invierte la condición.

Puede ocurrir que deseemos obtener documentos que se encuentren entre una lista de posibles coincidencias. Para ello, utilizaremos el operador condicional `"$in"`, seguido de un array con la lista de valores a evaluar:

```
> db.coleccion.find({"fecha" : {"$in" : [ISODate("2014-10-20T22:00:00Z"),
ISODate("2014-10-19T18:56:20.197Z"), ISODate("2014-10-15T18:56:20.197Z")]}})
```

En el caso de que quisiéramos cualquier documento excepto los que se encuentran en la lista, utilizaríamos el operador condicional `"$nin"`.

Optimización de las búsquedas

Para que nuestras búsquedas sean mucho más rápidas y eficientes, necesitaríamos indexar la colección por el campo fecha:

```
> db.coleccion.ensureIndex({"fecha" : 1})
{
   "createdCollectionAutomatically" : false,
   "numIndexesBefore" : 1,
   "numIndexesAfter" : 2,
   "ok" : 1
}
```

El comando `ensureIndex` permite crear un índice en base a la lista de campos especificados. Con un valor 1, indicamos que el campo esté ordenado de alfabéticamente o de menor a mayor. Si el valor fuese -1, el orden sería a la inversa.