MongoDB

BASE DE DATOS NOSQL

Contenidos

- Actualización y Eliminación de Documentos
- Indexación
- Replicación
- Sharding
- Relaciones
- MapReduce
- GridFS

Actualización de documentos

- Se usan los métodos **update()** y **save():**
  - update() actualiza valores en un documento existente
  - save() reemplaza el documento existente con el documento que se pasa
- SINTAXIS:

```
db.collection.update(
   <query>,
   <update>,
   {
     upsert: <boolean>,
     multi: <boolean>,
     writeConcern: <document>,
     collation: <document>,
     arrayFilters: [ <filterdocument1>, ... ]
   }
)
```

condiciones where

parámetros de actualización

convierte updates en insert si no coinciden los parámetros de la query

Actualización de documentos

- Se usan los métodos **update()** y **save():**
  - update() actualiza valores en un documento existente
  - save() reemplaza el documento existente con el documento que se pasa
- SINTAXIS:

```
db.inventory.updateMany(
   { "qty": { $lt: 50 } },
   {
     $set: { "size.uom": "in", status: "P" },
     $currentDate: { lastModified: true }
   }
)
```

Operadores de actualización

- En mongodb no hace falta crear colecciones, se crean al insertar un documento

Name	Descripción
$currentDate	asigna un valor de fecha actual a un campo fecha (Date o timestamp)
$inc	incrementa el valor de unm campo la cantidad indicada
$min	Sólo actualiza el campo si el valor especificado es menor que el existente en el campo
$max	Sólo actualiza el campo si el valor especificado es mayor que el existente en el campo
$mul	Multiplica el valor del campo por la cantidad especificada
$rename	Renombra un campo
$set	Añade el valor a un campo
$setOnInsert	Añade el valor a un campo si la actualización acaba insert
$unset	Elimina el campo especificado de un documento

Operadores de actualización Arrays

- En mongodb no hace falta crear colecciones, se crean al insertar un documento

Name	Descripción
`$`	Marcador para actualizer el primer elemento que cumpla las condiciones de la query
`$[]`	Marcador para actualizar todos los elementos en un array para todos los documentos que cumplan las condiciones de la query
`$[<identifier>]`	Marcador para actualizar todos los elementos de un array que cumplen la condición de la opción `arrayFilters` para todos los documentos que cumplan una condición
`$addToSet`	Añade elementos a un array si no existen en el set
`$pop`	Elimina el primero o último elemento de un array
`$pull`	Elimina todos los elementos de un array
`$push`	Añade ítems a un Aray
`$pullAll`	Elimina todos los valores de un array que cumplen la condición

Eliminación de documentos

- Método **remove()** que acepta dos parámetros:
- **deletion criteria** – (Optional) criterios para eliminar documentos.
- **justOne** – (Optional) a true o 1 elimina un documento.
- Sin parámetros se eliminan todos los documentos:

```
db.inventory.remove()
```

Buscar y modificar

- Busca documentos por unos criterios y realiza alguna operación previo a mostrarlos

- Método **db.collection.findAndModify({**
  query: <document>, #Criterios de búsqueda
  sort: <document>, #Criterios de ordenación
  remove: <boolean>, #Borra los documentos encontrados
  update: <document>, #Actualiza los campos indicados
  new: <boolean>, #true: devuelve el documento modificado, #false: mismo documento
  fields: <document>, #devuelve los campos indicados
  ...
});

EJ:

```
db.people.findAndModify({
    query: { name: "Andy" },
    sort: { rating: 1 },
    update: { $inc: { score: 1 } },
    upsert: true
})
```

EJERCICIO 4

1. Insertar los documentos de la colección media.json en una base de datos llamada «media» en una única operación.
2. Actualizar el documento que hace referencia a la película «Matrix», de manera que se cambia su estructura a:

```
{“tipo”: “DVD”,
“Titulo”: “Matrix”,
“estreno”: 1999,
“genero”:”accion”
}
```

3. Considerar un nuevo documento para la colección media:

```
{“tipo”: “Libro”,
“Titulo”: “Constantinopla”,
“capitulos”:12,
“leidos”:3
}
```

Añadir el documento a la colección media y a continuación incrementar en 5 unidades el valor de la clave «leídos».

EJERCICIO 4(CONT)

4. Actualizar el documento referido a la película «Matrix» cambiando el valor de la clave «género» a «ciencia ficción».
5. Actualizar el documento referido al libro «Java para todos» de manera que se elimine la clave «editorial».
6. Actualizar el documento referido al libro «Java para todos» añadiendo el autor «María Sancho» al array de autores.
7. Actualizar el documento referido a la película «Matrix» añadiendo al array «actores» los valores de «Antonio Banderas» y «Brad Pitt» en una única operación.
8. Actualizar el documento referido a la película «Matrix» añadiendo al array «actores» los valores «Joe Pantoliano», «Brad Pitt» y «Natalie Portman» en caso de que no se encuentren, y si se encuentran no se hace nada.
9. Actualizar el documento referido a la película «Matrix» eliminando del array el primer y último actor.
10. Actualizar el documento referido a la película «Matrix» añadiendo al array actores los valores «Joe Pantoliano» y «Antonio Banderas».

EJERCICIO 4(CONT)

11. Actualizar el documento referido a la película «Matrix» eliminado todas las apariciones en el array «actores» de los valores «Joe Pantoliano» y «Antonio Banderas».
12. Actualizar el documento referido al disco «Recuerdos» y añadir una nueva canción al array «canciones»:

```
{“cancion”:5,
“titulo”: “El atardecer”,
“longitud”: “6:50”
}
```

13. Actualizar el documento referido al disco «Recuerdos» de manera que la canción «El atardecer» tenga asignado el número 3 en vez de 5.
14. Actualizar el documento referido al disco «Recuerdos» de manera que en una sola operación se cambia el nombre del artista a «Los piratillas» y se muestre el documento resultante.
15. Renombrar el nombre de la colección «media» a «multimedia».

Indexación

- Estructura de datos que mantiene información acerca de los valores de campos específicos en los documentos de una colección
- Se utiliza para ordenar y clasificar rápidamente los documentos de una colección
- Se asegura una búsqueda y recuperación rápida de datos de los documentos
- Cuando se crea un índice, aumenta la velocidad de las consultas, pero se reduce la velocidad de las inserciones y las eliminaciones
- Deben mantenerse (borrarse o reconstruirse) por:
  - Limpiar algunas irregularidades que aparecen en los índices.
  - Aumento del tamaño de la base de datos.
  - Espacio excesivo ocupado por los índices. Solo se pueden definir como máximo cuarenta índices por colección.

Índice simple

- Función **createIndex(clave:orden)**
  - clave: que se usará para crear el índice
  - orden: dirección de ordenación del índice: 1 ascendente y -1 descendente
- El índice se creará para todos los valores de la clave indicada para todos los documentos de la colección.

```
db.posts.createIndex(“Etiquetas”:1)
```

- Para indexar un documento embebido:

```
db.posts.createIndex(“comentarios.contador”:1)
```

- Si se indexa un array, se incluyen todos los elementos del array en el índice, pero no se puede indicar el orden dentro del array.
- Para consultar los índices: **db.collection.getIndexes()**

Índice compuesto

- Mantienen bajo el número de índices de una colección. Se deben utilizar siempre que sean posible
- Tipos de índices compuestos:
  - índices multiclave: { titulo:”One Post”, autor:{nombre:”John”, email:”john@email.com”}}

```
db.posts.createIndex(“autor”:1)
```

  - índices compuestos definidos por el usuario

```
db.posts.createIndex(“nombre”:1,”email”:1)
```

  - En la ordenación no es bueno usar índices compuestos, salvo que la lista de términos y las direcciones de ordenación encajen exactamente con la estructura del índice. En estos casos, una elección mejor es usar índices simples individuales sobre cada campo.

Índice de texto

- Se pueden crear **índices de texto** para realizar búsquedas sobre campos de texto (string):
- Búsquedas: db.collection.find({$text:{$search:”cadena o cadenas de texto a buscar”}})
- Para crear el índice:

```
db.reviews.createIndex( { comments: "text" } )
```

- Si se incluyen varios campos de texto, se puede especificar la relevancia (peso) de la búsqueda de un campo respecto a los otros:

```
db.blog.createIndex(
  { content: "text",  keywords: "text",   about: "text"  },
  { weights: { content: 10,   keywords: 5  },
    name: "TextIndex"   } )
```

Otros índices

- Índices Hashed:
  - Permiten reducir el tamaño de los índices almacenando sólo la clave hashed. Se utiliza en particiones-SHARDS
- Índice Geospatial:
  - Permiten agilizar la búsqueda de geometrías en una superficie esférica (las búsquedas pueden ser de inclusión, intersección y proximidad)
- Índice TTL (Time-To-Live)
  - Índices de campo único que permiten eliminar documentos de una colección después de un intervalo de tiempo o de una hora especificada
  - Útil para eliminar datos ligados a eventos, logs o información de session que necesitan que persistan en la bd durante un tiempo finito.

EJERCICIO 5

1. Cargar el fichero movieDetail.json en una bd movies
2. Listar los índices de la colección creada
3. Crear un índice sobre el campo title, orden ascendente
4. Crear un índice sobre título y director

EJERCICIO 5 (CONT)

1. Crear la siguiente consulta: Buscar todos los directores de películas que hayan estado nominadas y no hayan conseguido un oscar
2. Crear un índice para optimizar esta consulta