UNIVERSIDAD AUTÓNOMA METROPOLITANA
Casa abierta al tiempo

EDUCACIÓN IT

MongoDB Fundamentos

En la primer clase descubrimos el potencial de MongoDB para manejar grandes volúmenes de datos de manera flexible y con capacidad de escalar fácilmente para soportar una demanda creciente. Comparamos los motores de base de datos NoSql con los Relacionales. También vimos que MongoDB almacena datos en forma de documentos BSON (JSON Binarios) dentro de colecciones. Ahora pasemos a comprender cómo se crean, insertan, actualizan y consultan esos datos.

Mongo Shell (Línea de comandos)

Mongo Shell viene con la distribución estándar de MongoDB. Ofrece un entorno JavaScript con acceso completo al lenguaje y las funciones estándar. Proporciona una interfaz completa para la base de datos MongoDB.

Para ejecutarlo simplemente escribir "mongo" en una terminal.

Comandos básicos

```
show dbs //Muestra todas las bases de datos
use <dbname> //Creo una base nueva, si existe la usa
db.help() //Todos los comandos del objecto db
db.stats(); //Estadisticas de la base actual
db.hostInfo(); //Datos del host
```

Exportar e Importar datos

Importar y exportar datos son operaciones muy comunes en motores de base de datos. Su principal uso es el resguardo o backup de la información pero también se utilizan para generar entornos de trabajo asilados de los productivos obteniendo datos reales para que el desarrollador pueda trabajar sin comprometer la operación del sitio o app. Estos comandos se deben ejecutar directo en la terminal, fuera del cliente de mongo.

```
mongoimport --db dbName --collection collectionName --file fileName.json
--jsonArray //Importar desde JSON
```

```
mongoexport -d dbName -c collName --out fileName //Exporta un JSON
mongodump --db dbName //Backup en dir dump BSON
mongorestore //Restore desde dump
```

Comandos CRUD (ABM)

Los comandos CRUD (Create, Read, Update, Delete) nos permiten manipular los datos de una colección. Estos comandos se conocen en castellano como Altas, Bajas, Modificaciones.

Crear Bases de datos

```
use db //Creo una base nueva o si existe la uso
```

Crear Colecciones

```
db.createCollection(collName) //Crear colecciones.
show collections //Muestra las colecciones de una base.
```

Crear Documentos en colecciones

Las operaciones de crear o insertar agregan nuevos documentos a una colección. 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.col.insert() //Inserto un documento
db.col.insert([]) //Inserto multiples documentos
db.col.save() //Otra sintaxis, más usada desde código
```

Consultar Documentos

```
db.col.find({criterio}) //Resultados
db.col.find({}).pretty() //Resultados con formato
db.col.find({$or:[{"userId":6},{"userId":7}]})
db.col.find({$and:[{"userId":6},{"id":53}]}) //AND
db.col.find({$and:[{"userId":6},{"title":/error/}]}) //AND-LIKE
db.col.find({"id":{$gte:60},"userId":6}) //AND Mayor o igual que
```

Proyección

```
db.col.find({},{"website":1,"_id":0}) //Selecciono qué atributos mostrar
```

Límite

```
db.col.find({}).limit(n)
```

Orden

```
db.col.find({}).sort({"website":-1}); //Orden DESC
```

Editar documentos

```
db.col.update({criterio},{data}) //Actualiza un doc
db.col.update({criterio},{data},{multi:true}). //Actualiza multiples doc
```

Reemplazar documentos

```
db.col.save({ObjectId},{nuevo doc}) //Reemplaza todo el doc
```

Borrar documentos

```
db.col.remove({criterio}) //Elimina un elemento
```

Agregación

Las operaciones de agregación procesan los registros de datos y devuelven los resultados calculados. Las operaciones de agregación agrupan los valores de varios documentos y pueden realizar una variedad de operaciones en los datos agrupados para devolver un solo resultado. MongoDB proporciona tres formas de realizar la agregación: el canal de agregación, la función de reducción de mapas y los métodos de agregación de propósito único.

Función aggregate()

Nos permite filtrar un conjunto de datos y luego efectuar operaciones de agregación como $sum, $avg, $min, $max.

```
db.col.aggregate([
      {$match:{key:value}},
      {$group:{_id, operator}}
]);
```

MapReduce

MongoDB también proporciona operaciones de mapReduce para realizar la agregación. En general, las operaciones de reducción de mapa tienen dos fases: una etapa de mapa que procesa cada documento y emite uno o más objetos para cada documento de entrada, y la fase de reducción que combina la salida de la operación de mapa. Opcionalmente, map-reduce puede tener una

etapa final para realizar modificaciones finales en el resultado. Al igual que otras operaciones de agregación, map-reduce puede especificar una condición de consulta para seleccionar los documentos de entrada, así como clasificar y limitar los resultados. No es tan eficiente como el canal de agregación, esta pensada para un procesamiento batch.

Agregación de propósito único

Todas estas operaciones agregan documentos de una sola colección. Si bien estas operaciones proporcionan un acceso simple a los procesos de agregación comunes, carecen de la flexibilidad y las capacidades de la canalización de agregación y reducen el mapa.

```
db.collection.count()
db.collection.distinct()
```

Indexación

Los índices soportan la resolución eficiente de consultas. Sin índices, MongoDB debe escanear cada documento de una colección para seleccionar aquellos documentos que coincidan con la declaración de consulta. Este escaneo es altamente ineficiente y requiere que MongoDB procese un gran volumen de datos.

Los índices son estructuras de datos especiales, que almacenan una pequeña parte del conjunto de datos en una forma fácil de recorrer. El índice almacena el valor de un campo específico o conjunto de campos, ordenado por el valor del campo como se especifica en el índice.

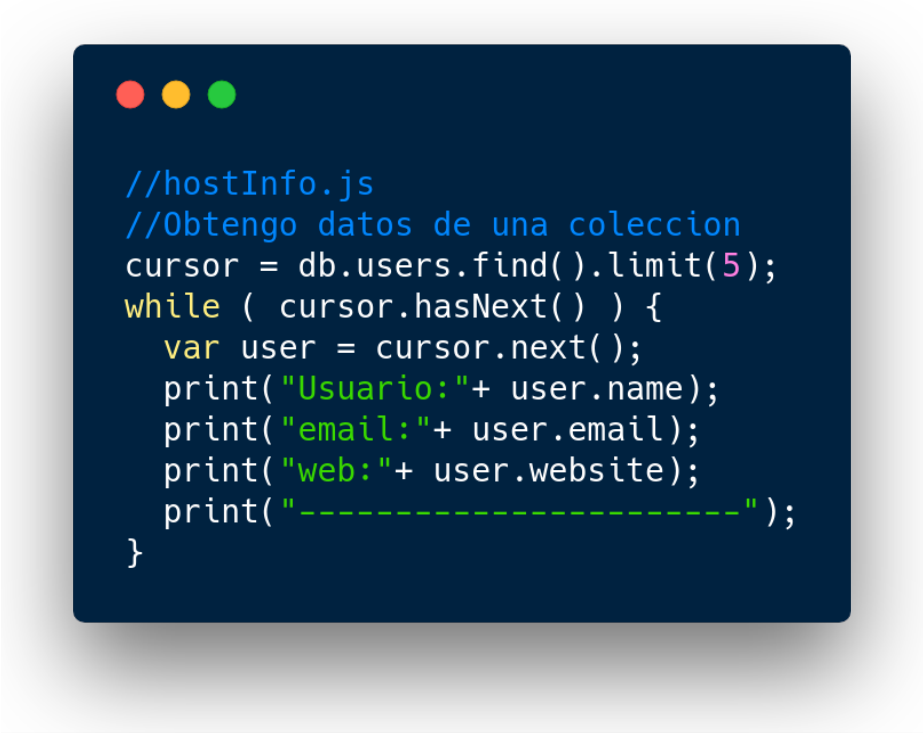
```
db.col.ensureIndex({ campo: 1});
//Crea un índice en el campo con orden creciente
```

Ejecutar Javascript desde el motor

MongoDB incluye un motor de Javascript el cual nos permite ejecutar código JS directo en MongoDB o podemos incluir un archivo externo.

Ejecutar código JS desde un archivo externo

```
load(path) //Ejecuto un archivo externo
```



```
//hostInfo.js
//Obtengo datos de una coleccion
cursor = db.users.find().limit(5);
while ( cursor.hasNext() ) {
  var user = cursor.next();
  print("Usuario:"+ user.name);
  print("email:"+ user.email);
  print("web:"+ user.website);
  print("-----------------------");
}
```

.

Fuentes

https://docs.mongodb.com/manual/mongo/

Apress - Practical MongoDB