**Taller MongoDB y Javascript + Node**

**Elaborado: 25-07-2024**

**Diego López Zapata**

**MongoDB**

**El taller de Mongo se revisa el lunes 29 de julio.**

**Punto 1**

**En mongo y visual**

Una empresa de arrendamiento de inmuebles requiere de una base de datos que permita la gestión de la información de la inmobiliaria

De los inmuebles se requiere conocer:

Su codigo, ciudad, dirección, tipo inmueble, area, servicios, propietarios, valor arriendo

El codigo debe ser de tipo numérico.

La ciudad debe ser de tipo String.

La dirección deber ser de tipo String.

El tipo inmueble de tipo String.

El área de tipo numérica.

Los servicios manejarlos como un array

Los propietarios manejarlos como un array

El valor arriendo es numérico.

Crear una base de datos con una sola colección con un diseño embebido y luego:

Registrar 3 documentos con el insertOne

db.inmuebles.insertOne({

  "codigo": 1,

  "ciudad": "Medellin",

  "direccion": "Calle 123",

  "tipo\_inmueble": "Apartamento",

  "area": 80,

  "servicios": ["Agua", "Luz", "Internet"],

  "propietarios": ["Propietario 1", "Propietario 2"],

  "valor\_arriendo": 500000

});

db.inmuebles.insertOne({

  "codigo": 2,

  "ciudad": "Bucaramanga",

  "direccion": "Avenida 456",

  "tipo\_inmueble": "Casa",

  "area": 150,

  "servicios": ["Agua", "Luz"],

  "propietarios": ["Propietario 3"],

  "valor\_arriendo": 800000

});

db.inmuebles.insertOne({

  "codigo": 3,

  "ciudad": "Cartagena",

  "direccion": "Carrera 789",

  "tipo\_inmueble": "Local",

  "area": 100,

  "servicios": ["Agua", "Internet"],

  "propietarios": ["Propietario 4", "Propietario 5"],

  "valor\_arriendo": 600000

});

Investigue cómo consultar la cantidad de documentos de una colección y luego crear la consulta que muestre la cantidad de documentos de la colección inmuebles.

db.inmuebles.countDocuments();

Listar todos los documentos con todos los atributos

db.inmuebles.find().pretty();

Listar todos los documentos pero solo con los atributos tipo inmueble, ciudad, area y valor arriendo.

db.inmuebles.find({}, { tipo\_inmueble: 1, ciudad: 1, area: 1, valor\_arriendo: 1 }).pretty();

Editar la ciudad de un inmueble. El mueble a editar debe ser filtrado por codigo.

db.inmuebles.updateOne(

  { codigo: 1 },

  { $set: { ciudad: "Cali" } }

);

Incrementar todos los arriendos en un 8%

db.inmuebles.updateMany(

  {},

  { $mul: { valor\_arriendo: 1.08 } }

);

Eliminar uno de los arriendos filtrándolo por el idObject.

db.inmuebles.deleteOne({ \_id: ObjectId("60d5ec49f1a2c42b1c0a1d2e") });

Editar el valor de ciudad, dirección y valor arriendo para un documento

db.inmuebles.updateOne(

  { codigo: 2 },

  { $set: { ciudad: "Bogota", direccion: "Nueva Avenida 456", valor\_arriendo: 850000 } }

);

**Punto 2**

GreenFood es una empresa que se dedica a la venta de frutas y verduras orgánicas en el área metropolitana de Medellín.

Los clientes pueden realizar sus pedidos desde un sitio web. De los clientes interesa conocer su documento, tipo documento, nombres, celular y dirección.

De los productos su código, nombre, categoria, precio y proveedores.

De los pedidos su codigo, cliente, fecha, estado del pedido, productos, cantidad del producto, precio unitario

De los proveedores interesa conocer su documento, nombres y celular.

Diseñe la base de datos en MongoDb que incluya diseño embebido y referenciado.

Es decir que en MongoDB no toda la base de datos tiene que manejar todas sus colecciones embebidas o referenciadas sino que puede combinar ambas.Cómo mínimo la base de datos debe contener las colecciones: productos, clientes y pedidos.

En cada colección inserte 3 documentos.

db.clientes.insertMany([

  {

    "documento": "123456789",

    "tipo\_documento": "CC",

    "nombres": "Juan Pérez",

    "celular": "3001234567",

    "direccion": "Carrera 10 # 20-30"

  },

  {

    "documento": "987654321",

    "tipo\_documento": "CC",

    "nombres": "Ana Gómez",

    "celular": "3107654321",

    "direccion": "Calle 5 # 10-20"

  },

  {

    "documento": "456789123",

    "tipo\_documento": "CE",

    "nombres": "Luis Martínez",

    "celular": "3201234567",

    "direccion": "Avenida 15 # 30-40"

  }

]);

*// Insertar productos*

db.productos.insertMany([

  {

    "codigo": 1,

    "nombre": "Manzana",

    "categoria": "Fruta",

    "precio": 2000,

    "proveedores": [

      { "documento": "987654321", "nombres": "Proveedor A", "celular": "3101234567" },

      { "documento": "123123123", "nombres": "Proveedor B", "celular": "3201234567" }

    ]

  },

  {

    "codigo": 2,

    "nombre": "Zanahoria",

    "categoria": "Verdura",

    "precio": 1500,

    "proveedores": [

      { "documento": "456789123", "nombres": "Proveedor C", "celular": "3301234567" }

    ]

  },

  {

    "codigo": 3,

    "nombre": "Banano",

    "categoria": "Fruta",

    "precio": 1200,

    "proveedores": [

      { "documento": "987654321", "nombres": "Proveedor A", "celular": "3101234567" }

    ]

  }

]);

*// Insertar pedidos*

db.pedidos.insertMany([

  {

    "codigo": 1,

    "cliente": { "documento": "123456789", "tipo\_documento": "CC", "nombres": "Juan Pérez" },

    "fecha": "2024-07-29",

    "estado": "Entregado",

    "productos": [

      { "codigo": 1, "cantidad": 3, "precio\_unitario": 2000 },

      { "codigo": 2, "cantidad": 2, "precio\_unitario": 1500 }

    ]

  },

  {

    "codigo": 2,

    "cliente": { "documento": "987654321", "tipo\_documento": "CC", "nombres": "Ana Gómez" },

    "fecha": "2024-07-30",

    "estado": "Pendiente",

    "productos": [

      { "codigo": 2, "cantidad": 1, "precio\_unitario": 1500 },

      { "codigo": 3, "cantidad": 5, "precio\_unitario": 1200 }

    ]

  },

  {

    "codigo": 3,

    "cliente": { "documento": "456789123", "tipo\_documento": "CE", "nombres": "Luis Martínez" },

    "fecha": "2024-07-31",

    "estado": "Cancelado",

    "productos": [

      { "codigo": 1, "cantidad": 4, "precio\_unitario": 2000 }

    ]

  }

]);

Listar de la colección pedidos todos sus documentos empleando el find.

db.pedidos.find().pretty();

Listar de la colección pedidos todos sus documentos empleando el aggregate con el fin de detallar las colecciones referenciadas.

db.pedidos.aggregate([

  {

    $lookup: {

      from: "clientes",

      localField: "cliente.documento",

      foreignField: "documento",

      as: "cliente\_info"

    }

  },

  {

    $unwind: "$cliente\_info"

  },

  {

    $project: {

      "codigo": 1,

      "fecha": 1,

      "estado": 1,

      "productos": 1,

      "cliente": {

        "nombres": "$cliente\_info.nombres",

        "tipo\_documento": "$cliente\_info.tipo\_documento"

      }

    }

  }

]);

Investigar como sumar valores en una columna en mongodb y realice un ejemplo con la columna cantidad del producto de la colección productos.

db.pedidos.aggregate([

  { $unwind: "$productos" },

  {

    $group: {

      \_id: "$productos.codigo",

      total\_cantidad: { $sum: "$productos.cantidad" }

    }

  }

]);

**Punto 3**

**Responda falso o verdadero.**

* MongoDB es un motor de base de datos Relacional. **Falso**
* MongoDB es un motor de base de datos poco flexible. **Falso**
* MongoDB es un motor de base de datos que no permite almacenar grandes volúmenes de información. **Falso**
* MongoDB es un motor de base de datos que permite almacenar grandes volúmenes de información. **Verdadero**
* En MongoDB es obligatorio aplicar las reglas de normalización aplicadas en una base de datos relacional. **Falso**
* MongoDBCompass permite la administración de bases de datos MongoDB de manera local. **Verdadero**
* MongoDB Atlas permite crear base de datos en la nube **Verdadero**
* JSON es un formato para intercambio de información **Verdadero**
* JSON es un lenguaje de programación para crear APIS **Falso**
* Mongo almacena los datos en formato BSON **Verdadero**
* En MongoDB se pueden almacenar arrays de objetos. **Verdadero**
* Para listar todos los datos de una colección en mongodb se emplea **SELECT \* FROM collection Falso**
* Para listar todos los datos de una colección de manera organizada en mongodb se emplea db.collecction.find().pretty()**Verdadero**
* Para listar todos los datos de una colección de manera organizada en mongodb se emplea db.collecction.find().uggly()**Falso**
* En ninguna base de datos de mongodb se permite eliminar documentos. **Falso**
* En ninguna base de datos de mongodb se permite modificar documentos. **Falso**

**Seleccione la opción correcta:**

1. **¿Cuál es el comando para insertar un documento en una colección en MongoDB?**

a) db.collection.updateOne()

b) db.collection.find()

**c) db.collection.insertOne()**

d) db.collection.deleteOne()

1. ¿Cómo se realiza una consulta en MongoDB?

a) SELECT \* FROM collection

**b) db.collection.find()**

c) db.collection.query()

d) FIND \* FROM collection

1. **¿Qué es un índice en MongoDB y para qué se utiliza?**

a) Un mecanismo para crear copias de seguridad

b) Un método para ordenar los datos

**c) Una estructura que mejora la velocidad de las consultas**

d) Una herramienta para administrar usuarios

1. **¿Qué es el \_id en un documento de MongoDB?**

a) Un campo opcional que se puede usar para comentarios

**b) Un identificador único para cada documento**

c) Una referencia a otro documento

d) Un índice que mejora la velocidad de las consultas

1. **¿Cómo se actualiza un documento en MongoDB?**

**a) db.collection.updateOne()**

b) db.collection.insertOne()

c) db.collection.removeOne()

d) db.collection.findOne()

1. **¿Qué comando se utiliza para eliminar un documento de una colección en MongoDB?**

a) db.collection.updateOne()

b) db.collection.insertOne()

**c) db.collection.deleteOne()**

d) db.collection.removeOne()

1. **¿Cuál es la estructura básica de un documento BSON?**

**a) Clave-valor**

b) Clave-columna

c) Fila-columna

d) Etiqueta-dato

1. **¿Qué significa que MongoDB sea "schemaless"?**

a) Que los datos deben seguir un esquema rígido

**b) Que los datos no tienen un esquema fijo**

c) Que solo se pueden almacenar datos estructurados

d) Que los datos están siempre ordenados

1. **¿Cuál es una ventaja de que MongoDB sea schemaless?**

a) Los datos siempre están ordenados alfabéticamente

**b) Permite flexibilidad en la estructura de los documentos**

c) Requiere menos espacio en disco

d) Facilita la creación de índices

**Analice los siguientes enunciados y justifique su respuesta.**

1. Le contactan para el desarrollo de una red social destinada a gestionar y almacenar grandes volúmenes de información de líderes políticos.La empresa que le contacta tiene dentro de sus políticas, no incluir en sus proyectos la bases de datos NO SQL porque sus proyectos anteriores no lo han requerido y nadie en la empresa tiene conocimientos de MongoDB. ¿Cómo justificarías que para el proyecto debes emplear MongoDB y no una base de datos SQL?

* Crecimiento fácil: MongoDB permite manejar grandes cantidades de información sin complicaciones. A medida que la red social crezca, MongoDB puede adaptarse sin problemas.
* Flexibilidad: No necesita seguir un esquema rígido. Esto significa que puedes almacenar diferentes tipos de datos sin tener que hacer cambios complicados.
* Rápido y eficiente: Es ideal para situaciones donde hay muchas interacciones, como publicaciones y comentarios, ya que puede manejar muchas acciones al mismo tiempo.
* Datos variados: Las redes sociales tienen fotos, videos y textos. MongoDB permite guardar todo esto de manera sencilla.
* Ayuda disponible: Aunque la empresa no tenga experiencia con MongoDB, hay muchos recursos y comunidades en línea que pueden ayudar.

1. Suponga que en su proyecto formativo se decide emplear MongoDB aplicando estrictamente las reglas de normalización empleadas en bases de datos SQL. ¿Es acertada la decisión por qué sí o por qué no?

No es una buena idea.

* Diseño diferente: MongoDB no está hecho para seguir reglas estrictas como las bases de datos tradicionales. Intentar hacerlo puede complicar las cosas y hacer que funcione más lento.
* Unir datos: En MongoDB, a menudo se agrupan datos relacionados en un solo lugar, lo que facilita su acceso. Seguir reglas rígidas podría hacer que esto sea más complicado.
* Adaptabilidad: La flexibilidad de MongoDB permite que los datos cambien sin necesidad de hacer ajustes grandes y difíciles.

1. Supuesto. El SENA requiere mantener un control total de toda la información de sus programas de formación y tiene como proyecto cambiar de motor de base de datos ORACLE que es relacional, seguro y estable a MONGOdb. El líder del proyecto argumenta que la migración de toda la información de Oracle a MongoDB se hará en 2 meses. El líder de proyecto es arriesgado, le gusta quedar bien ante sus jefes, es terco en sus decisiones, le gustan las nuevas tecnologías y tiene 3 años de experiencia en bases de datos sql y 6 meses en NO SQL. La base de datos actual maneja aproximadamente 500 tablas ¿Qué opinión tienen sobre el proyecto?

* Riesgo alto: Cambiar de una base de datos estable como Oracle a MongoDB, especialmente con alguien que tiene poca experiencia, puede ser arriesgado. Esto podría llevar a problemas inesperados.
* Complejidad: Con 500 tablas en Oracle, la migración no será sencilla. Se necesita tiempo y cuidado para asegurarse de que todo se transfiera correctamente.
* Plazo poco realista: Decir que se puede hacer en 2 meses parece optimista. La migración lleva tiempo, y hay que probar que todo funciona bien después.
* Evaluar necesidades: Es importante preguntarse si realmente se necesita cambiar. Si Oracle ya está funcionando bien, puede que no valga la pena hacer el cambio.

**Ejercicio en Node, Jscript, HTML**

**La actividad de Javascript se revisa el martes 30 de julio.**

Crear un proyecto en node con un servidor

Crear un formulario con los campos:

Codigo Reserva,

Nombre Cabaña,

Fecha inicio ,

Fecha fin,

Cantidad Dias,

Alimentacion,

Precio por Dia,

Precio Total

El código de reserva es un campo donde solo se pueden digitar números.

El nombre de la cabaña se debe digitar

La fecha de inicio no puede ser superior a la fecha fin. Ambas fechas deben ser ingresadas.

La cantidad de días es de sólo lectura y se calcula entre la diferencia entre la fecha inicial y final.

La alimentación es un campo de tipo radio donde se selecion si ó no.

El precio por día es de $100.000 y debe ser un campo de solo lectura.

El precio total es la multiplicación entre la cantidad de días y el precio por día.

Al dar clic en el botón enviar, realizar las validaciones respectivas y si todo está diligenciado, conformar un objeto con los datos e imprimirlo por consola.

El formulario debe estar incorporado al proyecto que creó en node y funcionar correctamente.

**Éxitos**