Trabajo tema 4

Autores: Andrés García, Andrés Millán, Carlota Valdivia, Adrián Rodríguez, José Mª Poblador.

Objetivos

- Descarga e instalación del SGBD (1-2 páginas).
- Descripción del DDL y DML utilizado (no se necesita sintaxis completa, solo nombrar los comandos utilizados y para qué sirven) (1-2 páginas).
- Sentencia empleada para la creación de estructuras, inserción, modificación, borrado de datos, consultas (1-2 páginas).
- Breve descripción del mecanismo de conexión al SGBD desde una aplicación (1-2 páginas).
- Breve discusión sobre si sería adecuado para implementar el SI de la práctica (1/2 - 1 página).

Motivación del trabajo

Hemos elegido MongoDB por su sencillez de uso y por su gran potencial, ya que es un modelo de base de datos hoy día ampliamente utilizado, sobre todo en el mundo de la programación Web. Además, varios miembros del equipo tienen experiencia en el uso de esta aplicación, con lo que esperamos que ello aporte un valor añadido a este trabajo.

¿Qué es MongoDB?

MongoDB es un sistema de gestión de bases de datos NoSQL, de código abierto y orientado a documentos. El término "NoSQL" significa que la base de datos no utiliza tablas para almecenar entradas o relacionar entre si las distintas entradas almacenadas. Las entradas se almacenan como objetos independientes en formato BSON, que significa *binary JSON* . Esta estructura codifica tipo e información de la longitud de la información, lo que permite recuperar los datos mucho más rápido.

Equivalencia de términos en MongoDB

MongoDB utiliza unos términos algo distintos a los tradicionales de SQL. Sin embargo, existe una clara equivalencia entre unos y otros. Estos son:

Base de datos relacional	MongoDB
Base de datos	Base de datos (*database*)
Tabla	Colección (*collection*)
Tupla/fila	Documento (*document*)
Columna	Campo (*field*)
Clave primaria	Clave primaria (proporcionada por defecto como `_id` por Mongo)

Ventajas y desventajas

La diferencia principal de MongoDB frente a otros sistemas de gestión de bases de datos es su sistema de documentos y colecciones.

Ventajas

Como ventaja en este sistema, vemos que aporta una enorme flexibilidad a diferencia de otros sistemas, pudiendo hacer modificaciones en la estructura sin detener o incluso tener dos tuplas del mismo tipo y con un formato totalmente distinto, esto puede ser muy interesante en casos donde los datos no siguen una estructura definida o si el esquema va a sufrir muchos cambios.

Otra ventaja, sobre todo comprarandola con las bases de datos estructurados, es su facilidad a la hora de escalar los datos, al no seguir estructuras es mucho más fácil escalar horizontalmente, es decir, dividir la base de datos en diferentes servidores, lo que se conoce como estructuras distribuidas y de caracter descentralizado. Además este tipo de bases de datos no suelen requerir de grandes recursos a diferencia de las SQL.

Desventajas

Esta flexibilidad a su vez presenta también varias desventajas, ya que el exceso de flexibilidad junto a malas prácticas pueden crear inconsistencias importantes en la base de datos.
Además en el caso de que tengamos modelos con muchas relaciones y dependencias, se hará mucho más dificil utilizar este tipo de bases de datos sin que haya inconsistencias y requerirán de un mayor control del desarrollador del sistema.
Las NoSQL no están tan estandarizadas como SQL, haciendo que pueda ser complicado realizar ciertas peticiones más complejas, aunque en el caso de Mongo, al ser uno de los más populares se puede encontrar prácticamente cualquier duda resuelta.

Descarga e instalación del SGBD

Para utlizar Mongo, hemos decidido que uno de los integrantes instale en su máquina el SGBD mientras que el resto nos conectamos a esta. Para ello, hemos usado la función de Live Share de VSCode sobre la máquina de José Mª Poblador, la cual utiliza Manjaro, distribución basada en Arch Linux.

Para instalarlo en esta distro, hemos seguido el siguiente proceso:

```
$ pamac build mongodb-bin           # Instala el binario de mongodb.
Utilizamos esta versión y no el paquete
                                    # `mongodb` puesto que esta requiere 180GB
de espacio en disco y
                                    # un largo tiempo de compilación.
`mongodb-bin` se salta parte de este proceso.

$ systemctl start mongodb.service   # Iniciamos el servicio
$ systemctl enable mongodb.service  # Lo activamos
```

Con esto, el SGBD queda instalado. Para iniciarlo, escribimos `mongo`. La base de datos corre por defecto en la IP 127.0.0.1 sobre el puerto 27017.

MongoDB es multiplataforma, pudiéndose instalar en cualquier Windows, Mac o Linux, aunque hay que remarcar que el proceso de instalación e inicio difiere en cada sistema operativo.

GUI oficial

Mongo Compass es la GUI oficial de Mongo. Permite conectarse a una base de datos, hacer peticiones, visualizar datos, así como ver y optimizar las peticiones. El cliente está disponible para Windows, Linux y Mac. Las características que proporciona son muy variadas y potentes.

Conexiones al SGBD desde una aplicación

Una vez se ha instalado el cliente, necesitaremos conectarnos a MongoDB. Para ello, necesitaremos un string URI (*Uniform Resource Identifier*). Es similar a una URL, y es un parámetro de la shell de mongo, Mongo Compass y el driver de MongoDB.
El string URI usado asume que se ha configurado el sistema de autenticación; es decir, se ha creado un usuario y contraseña con los permisos correspondientes para la conexión.

Conexión desde terminal

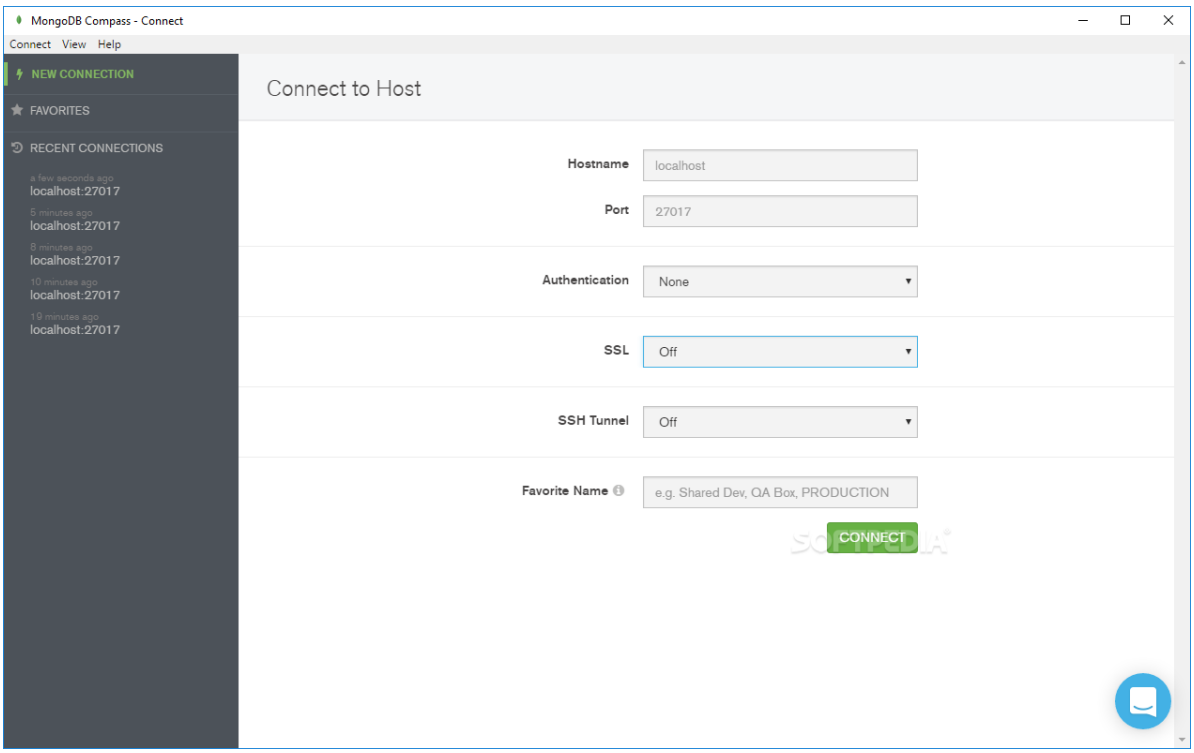
Para conectarse desde la shell, se sigue el siguiente formato:

```
mongo mongodb://$[hostlist]/$[database]?authSource=$[authSource] --username
$[username]
```

Por ejemplo, la URI del servidor local que estamos usando nosotros es `mongodb://127.0.0.1:27017/` . Corresponde a la IP junto al puerto de conexión.

Conexión desde el cliente Compass

El funcionamiento es análogo al de la shell. La pantalla de conexión a host contiene la siguiente información:



Como queremos conectarnos de forma local, en `Hostname` debemos poner `localhost` y en `port` `27017` .

Conexión desde un servidor Node.js

Normalmente, cuando se realiza una aplicación, la conexión suele venir desde otros clientes; no desde Mongo Shell ni Compass.

Mongoose es un cliente para mongodb de modelado de objetos para **Node.js** muy conocido. Permite hacer todo tipo de operaciones sobre la base de datos. El funcionamiento es similar a la Mongo Shell.

Para instalarlo, escribimos `npm install mongoose --save`. En nuestro fichero, importamos el paquete (`const mongoose = require('mongoose'`). Finalmente, para conectarnos:

```
const url = 'mongodb://127.0.0.1:27017/[base_de_datos]'
mongoose.connect(${url})
```

Descripción del DDL y DML utilizado

Lenguaje de Definición de Datos (DDL)

El DDL (*Data Definition Language*), es un lenguaje de programación que define estructuras de datos, proporcionado por los sistemas gestores de base de datos. Los programadores de un sistema gestor de base de datos son los encargados de definir y modificar las estructuras que almacenarán los datos, los procedimientos o las funciones que permitan consultarlos.

Repasemos las sentencias que se utilizan para definir las estructuras:

Creación

Para la creación de una tabla en MongoDB podemos usar los métodos implícitos `insertOne(<document>)` e `insertMany([<document1>, <document2>])`, que crean la colección si no existe previamente e insertan tantos documentos como especifiquen. Se emplean de la siguiente forma:

```
db.nombreColeccion.insertOne(
    {
        ...
    }
)
```

También existe un método de creación que crea explícitamente una colección: `createCollection("nombreColeccion")`

Alteración

Para modificar la estructura en MongoDB, como añadir o borrar una columna, se utiliza el método `updateMany(<filter>,<update>)`.

Como argumento recibe un filtro en que se basa la actualización, y recibe un operador `$set` o `$unset` que se encarga de añadir o borrar `fields`.

En MongoDB la forma de alterar la información no se hace a nivel de colecciones puesto que no se trata de una modificación estructural, sino de documentos.

Borrado (Drop)

Para borrar los documentos de una colección en MongoDB, la cual no está asociada ni depende de ninguna otra se usa, `db.nombreColeccion.drop()`.

Lenguaje de Manipulación de Datos (DML)

El DML (*Data Manipulation Language*) es el lenguaje proporcionado por los sistemas gestores de bases de datos que permite a los usuarios recuperar, almacenar, eliminar, modificar e insertar datos que se encuentran contenidos en la base de datos.

Las sentencias que se utilizan para manipular los datos son

Inserciones

La inserción en MongoDB se puede usar también para definir los atributos de la colección, aunque a diferencia de SQL, no se exige que todos los documentos tengan los mismos atributos.

Se usa `insertOne(<document>)` e `insertMany([<document1>, <document2>])`.

Consultas

La selección se vale del comando `find()`. Se aceptan diversos parámetros para ajustar la selección a aquello que precisemos.

`db.<collection>.find()` mostraría todo el contenido de la colección `<collection>`. Al igual que en SQL, podemos especificar parámetros para hacer la búsqueda más precisa.

```
db.<collection>.find(
    {<Where>},
    {<Select items>}
).sort({<Sort by>})
```

Modificaciones

En Mongo, las modificaciones usan el comando `updateMany()`. La sintaxis es muy similar a la selección, y como hemos visto, pueden usarse también para alterar los fields de la tabla.

Borrados (Delete)

En Mongo, los documents se borran mediante `deleteMany()`. La sintaxis es similar a los comandos antes vistos.

Conclusión

En el caso de un sistema de gestión de una tienda online, como podemos ver, hay demasiadas relaciones y no va a ser muy necesario escalar todo el sistema, por lo que en este caso sería mucho más útil usar una base de datos SQL.

Quizás ciertas tablas podrían ser interesantes de llevar a un sistema más flexible como MongoDB, como es el caso de las analíticas que tienen un formato totalmente dependiente del tipo de analítica que sea, pero en el resto de casos, es necesario tener una base de datos estructurada para evitar posibles inconsistencias a la hora de borrar o modificar datos.

Ejemplo de uso de sentencias

Creación

SQL	Mongo

`CREATE TABLE Inventario (`
`  codigo_alm INT(13) NOT NULL,`
`  EAN_producto INT(13) NOT NULL,`
`  cantidad INT(4),`
`  FOREIGN KEY (codigo_alm) REFERENCES Almacen(codigo),`
`  FOREIGN KEY (EAN_producto) REFERENCES Producto(EAN_prod`
`  PRIMARY KEY (codigo_alm, EAN_producto)`
`);`
	`db.createCollection("Inventario")`

`CREATE TABLE Producto (`
`  EAN_producto INT(13) NOT NULL PRIMARY KEY,`
`  nombre VARCHAR(50),`
`  fabricante VARCHAR(255),`
`  precio DECIMAL(10, 2) NOT NULL CHECK (precio > 0)`
`);`
	`db.createCollection("Producto")`

`CREATE TABLE Almacen (`
`  codigo INT(13) PRIMARY KEY,`
`  direccion varchar(255)`
`);`
	`db.createCollection("Almacen")`

Inserción

SQL	Mongo

`insert into Inventario`
`  (codigo_alm, EAN_producto, cantidad)`
`values`
`  (1, 13012, 20),`
`  (2, 56847, 5),`
`  (3, 66391, 30),`
`  (1, 67961, 50);`
	`db.Inventario.insertMany([`
	`  {`
	`  codigo_alm_id: 1,`
	`  EAN_producto_id: 13012,`
	`  cantidad: 20`
	`  },`
	`  {`
	`  codigo_alm_id: 2,`
	`  EAN_producto_id: 56847,`
	`  cantidad: 5`
	`  },`
	`  {`
	`  codigo_alm_id: 3,`
	`  EAN_producto_id: 66391,`
	`  cantidad: 30`
	`  },`
	`  {`
	`  codigo_alm_id: 1,`
	`  EAN_producto_id: 67961,`
	`  cantidad: 50`
	`  }`
	`])`

Inserción

```sql
insert into Producto
    (EAN_producto, nombre, fabricante, precio)
values
    (13012, "XPS 13",   "Dell",     1200.00),
    (56847, "3060 Ti",  "Nvidia",   1099.00),
    (66391, "3080",     "Nvidia",   1500.00),
    (67961, "3070",     "Nvidia",   500.00);
```

```mongo
db.Producto.insertMany([
    {
        EAN_producto_id: 13012,
        nombre: "XPS 13",
        fabricante: "Dell",
        precio: 1200.00
    },
    {
        EAN_producto_id: 56847,
        nombre: "3060 Ti",
        fabricante: "Nvidia",
        precio: 1099.00
    },
    {
        EAN_producto_id: 66391,
        nombre: "3080",
        fabricante: "Nvidia",
        precio: 1500.00
    },
    {
        EAN_producto_id: 67961,
        nombre: "3070",
        fabricante: "Nvidia",
        precio: 500.00
    }
])
```

```sql
insert into Almacen
    (codigo, direccion)
values
    (1, "Calle Apetecán 3"),
    (2, "Camino DDSI 10.0"),
    (3, "Avenida de los horrores navideños");
```

```mongo
db.Almacen.insertMany([
    {
        codigo_id: 1,
        direccion: "Calle Apetecán 3"
    },
    {
        codigo_id: 2,
        direccion: "Camino DDSI 10.0"
    },
    {
        codigo_id: 3,
        direccion: "Avenida de los horrores navideños"
    }
])
```

Búsqueda

SQL

```sql
select * from Inventario;

select * from Inventario where EAN_producto_id = 56847;

select * from Inventario where cantidad > 10;
```

Mongo

```mongo
db.Inventario.find()

db.Inventario.find(
    {EAN_producto_id: 56847}
)

db.Inventario.find(
    {cantidad: {$gt: 10} }
)
```

Búsqueda

```sql
select nombre from Producto;

select * from Producto
    where fabricante = "Nvidia"
    order by EAN_Producto asc;

select Ean_Producto, nombre from Producto where precio < 1050;
```

```mongo
db.Producto.find(
    {},
    {nombre: 1}
)

db.Producto.find(
    { fabricante: "Nvidia" }
).sort({ EAN_producto_id: 1})

db.Producto.find(
    {precio: {$lt 1050}},
    {EAN_producto:1, nombre:1}
)
```

```sql
select * from Almacen where codigo_id = 2;

select * from Almacen where codigo_id != 2;
```

```mongo
db.Almacen.find(
    {codigo_id: 2}
)

db.Almacen.find(
    {codigo_id: { $ne: 2 }}
)
```

Actualización

SQL	Mongo
`update Inventario set Cantidad = 50 where codigo_alm = 1 and EAN_producto = 82151;`	`db.Inventario.updateMany( {codigo_alm_id: 1, EAN_producto_id:67961}, {$set: {cantidad: 70} } )`
`update Producto set precio = 420 where nombre = "3060 Ti";`	`db.Producto.updateMany( {nombre: "3060 Ti"}, {$set: {precio: 420} } )`
`update Almacen set direccion = "Camino DDSI 1, 18000 Graná" where codigo_alm = 1;`	`db.Almacen.updateMany( {codigo_id: 1}, {$set: {direccion: "Camino DDSI 1, 18000 Graná"}} )`
`delete from Almacen where codigo = 1;`	`db.Almacen.deleteMany( {codigo_id: 1} )`