

INFORME TRABAJO 3 BASE DE DATOS

JOHAN ALEXANDER PANIAGUA BEDOYA

ALEJANDRO QUIROZ VELEZ

JHON JAIRO SERNA CORDOBA

BASES DE DATOS 2

FRANCISCO MORENO

FACULTAD DE MINAS

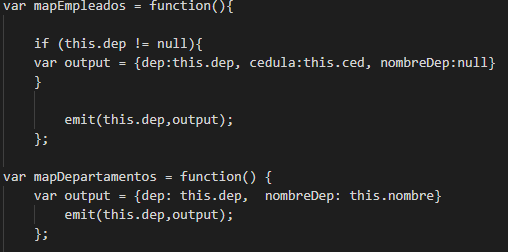
2019

INFORME GENERAL DEL TRABAJO

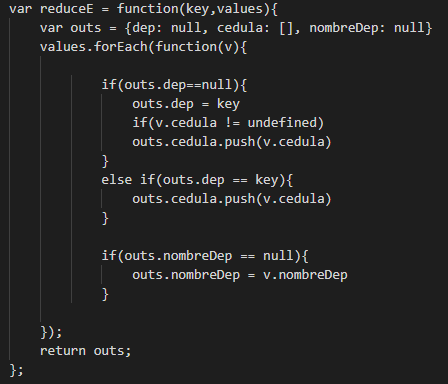
El objetivo de la tercera entrega fue poner en práctica todos los conocimientos adquiridos durante las clases teórico-prácticas con relación a Bases de Datos Documentales impartidas por el profesor Francisco Moreno, con el fin de profundizar aún más en nuestros conocimientos.

Este trabajo se dividió en dos puntos, los cuales se desarrollaron de la siguiente manera:

1. El objetivo de este punto es imitar el Join de las bases de datos relacionales con bases de datos no relacionales. Este punto se dividió en dos partes:
   1. Para esta parte se utilizó la función mapReduce(). La función mapReduce recibe tres parámetros, el primero es una función de tipo map, el segundo una función del tipo reduce y el tercero es cómo queremos hacer la salida. Se elaboraron dos funciones map, una por cada colección con la cuál simularíamos el Join relacional, esto con el fin de capturar los datos correctos. Las funciones map fueron las siguientes:



En este caso, solamente se realizó una función reduce, ya que ésta era suficiente para obtener los datos necesarios. La función es la siguiente y se explica el código a continuación.



La función recibe dos parámetros, una key, la cuál es única (por la cual hacemos el join) y unos values (en este caso una lista). “outs” es nuestra variable de retorno, la cual contiene todos los datos que se deben mostrar en la simulación del Join. Los datos que se encuentran en “values” se recorren por medio de un for each, y nuestro iterador será la variable “v”. Procedemos a hacer las verificaciones para llenar nuestra variable de retorno con los datos correspondientes, como la llave es única, los datos almacenados son los correctos. Una vez asignados todos los valores retornamos nuestra variable con todos los datos.

Ahora procedemos a realizar el llamado de la función desde las respectivas colecciones. Es recomendable hacer el llamado primeramente para la colección empleado y luego para departamento. Los datos obtenidos en el Join se almacenan en la colección “emp\_dept”.

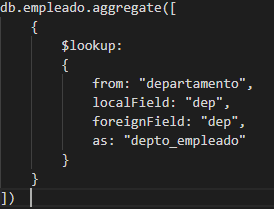


Nota: el out de la colección departamento lleva la opción “reduce”. Todo esto con el fin de que los datos no sean sobrescritos, sino que se carguen juntamente con los de empleado,

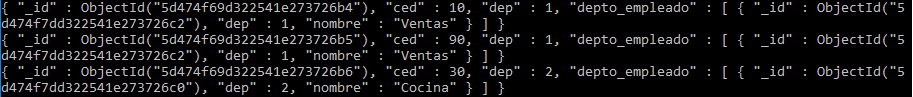
Los resultados obtenidos fueron los siguientes:



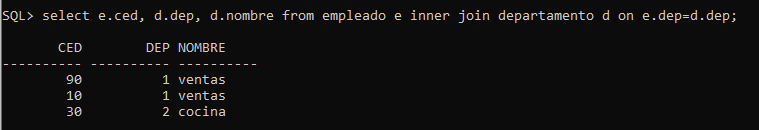
* 1. Se utilizó en comando $lookup para imitar el “Join” relacional. Este comando se emplea mediante la función “aggregate”, la cual se emplea sobre una colección. Para nuestro caso, la función fue empleada en la colección empleado, ya que ésta nos arrojó los resultados que eran necesarios. El código empleado fue el siguiente:



Los resultados obtenidos fueron los siguientes:

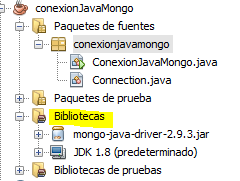


Para comprobar que los resultados eran los deseados, se realizó la consulta en SQL y los datos arrojados fueron los siguientes:

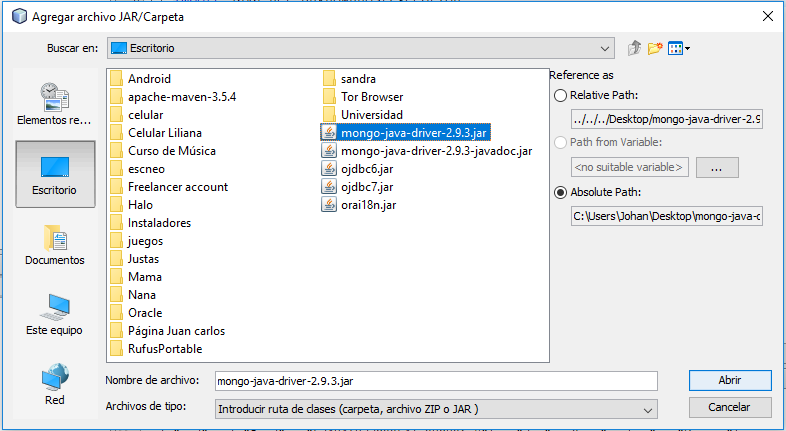


1. Este punto se dividió en dos partes:
   1. En este literal se nos pide establecer una conexión de Java con MongoDB, la cual se hizo de la siguiente manera:

* En primer lugar, se descargó el driver “mongo-java-driver-2.9.3.jar” del siguiente link <http://central.maven.org/maven2/org/mongodb/mongo-java-driver/2.9.3/>
* Luego de descargar el controlador, procedemos a abrir nuestro IDE para compilar código escrito en Java, en nuestro caso, usamos NetBeans.
* Una vez abierto nuestro IDE, procedemos a crear un nuevo proyecto del tipo “Java Application”. El nombre puede ser elegido a gusto, en nuestro caso se llamó “conexionJavaMongo”.
* Una vez creada la aplicación, importamos el controlador y lo conectamos con la aplicación de la siguiente manera:
* Hacemos clic derecho en “Bibliotecas”



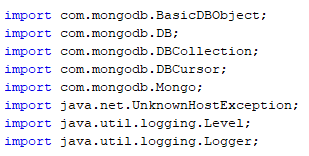
* Hacemos clic en “Agregar archivo JAR”
* Seleccionamos el controlador descargado previamente y hacemos clic en “abrir”



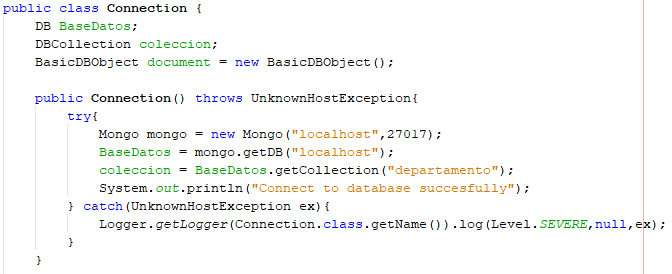
* Hecho esto, verificamos que el controlador haya cargado correctamente



* Creamos un archivo del tipo Java Class, el nombre puede ser elegido a gusto, en nuestro caso, para efectos prácticos se llamó “Connection”. Una vez creado el archivo, el siguiente paso es abrirlo e importar las siguientes librerías:



* Luego de importar las librerías, procedemos a escribir el siguiente código:



A continuación, se explica el código:

La variable “BaseDatos” es del tipo BD, la cual nos sirve para obtener la base de datos con la cual nos queremos conectar.

La variable “colección” del tipo DBCollection es utilizada para obtener la colección que deseamos de la correspondiente base de datos.

La variable document del tipo BasicDBObject nos sirve para crear un nuevo documento, el cual se usará para insertar o actualizar datos en la base de datos.

Cabe aclarar que la base de datos a la cual nos queremos conectar y la colección con la cual queremos trabajar debe haberse creado previamente.

El constructor Connection realiza la conexión de la siguiente manera:

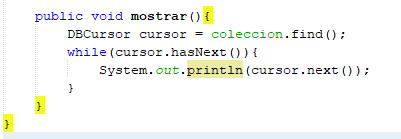
La variable “mongo” del tipo Mongo establece la conexión con la url y el puerto en el cual se encuentra nuestra base de datos. En este caso, se usó el localhost y el puerto 27017.

La variable BaseDatos nos sirve para almacenar la base de datos de interés, en nuestro caso, la base de datos se llama “localhost”.

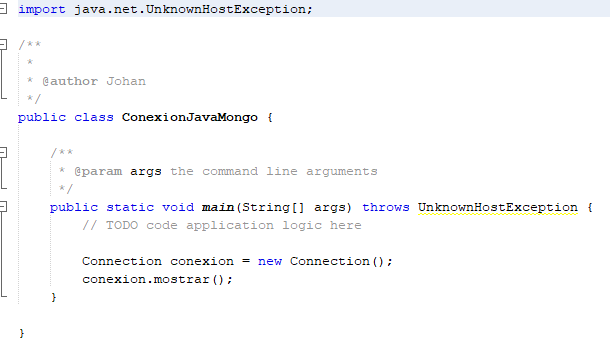
La variable colección obtiene la colección de datos de interés, en nuestro caso, usamos la colección “departamento”.

Luego, se mostrará en consola el mensaje “Connect to database succesfully” indicando que se conectó correctamente a la base de datos. En caso de que haya algún error se ejecutará la sentencia “catch”, la cual capturará el error y mostrará en pantalla cuál fue el error que impidió la conexión con la base de datos.

* Para efectos prácticos, se creó una funcionalidad más para mostrar todos los datos que hay en una colección en la base de datos, todo esto para demostrar que efectivamente se hizo la conexión a la base de datos:

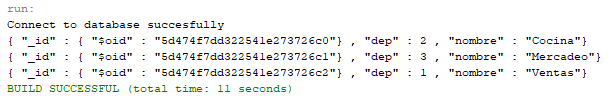


* Hecho esto, lo siguiente es irnos al archivo principal de nuestra aplicación (el cual se crea por defecto y generalmente se llama igual al nombre de nuestra aplicación) y escribimos el siguiente código dentro del main:



La variable “conexion” establece la conexión a la base de datos y la sentencia “conexión.mostrar();” nos permite probar que efectivamente se conectó a la base de datos y que los datos son reales.

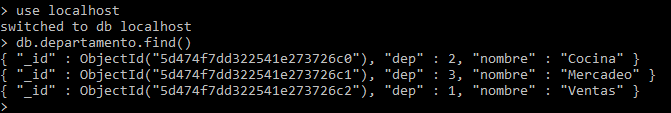
* Ahora podemos ejecutar la aplicación, la cual nos debe arrojar los siguientes resultados:



Podemos comparar los datos obtenidos directamente en Mongo Compass o por consola siguiendo estos pasos:

* Ejecutar la sentencia “mongod” para montar el servidor.
* Ejecutar la sentencia “mongo” para crear un cliente.
* Ejecutar la sentencia “use (nombre de la base de datos)” para conectarnos con la base de datos deseada
* Ejecutar la sentencia db.(nombre de la colección).find()” para obtener todos los datos que se encuentran en la base de datos y la colección deseada.

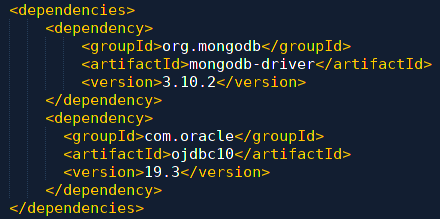
Se obtienen los siguientes resultados:



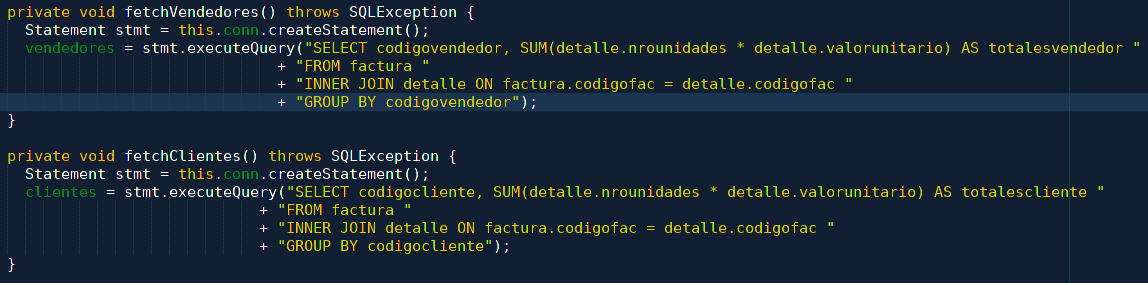
Como se puede evidenciar, los datos obtenidos son los mismos, lo cual indica que la conexión con la base de datos fue creada exitosamente.

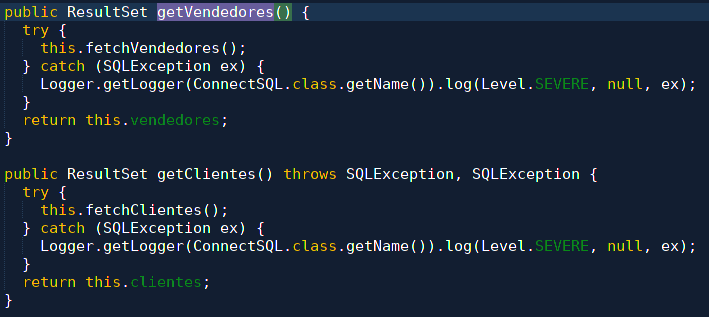
* 1. Aquí se nos pide tomar hacer dos querys a una base de datos en ORACLE SQL y enviar estos datos a unas colecciones de MongoDB.

Iniciamos un nuevo proyecto en Netbeans con control de paquetes de Maven y agregamos nuestros drivers para manejar las conexiones a ambas bases de datos.



Ahora creamos una clase para manejar la conexión a SQL y creamos dos métodos para hacer las peticiones que se nos pide guardar y un par de métodos accesores con el fin de obtener el ResultSet desde la interfaz.

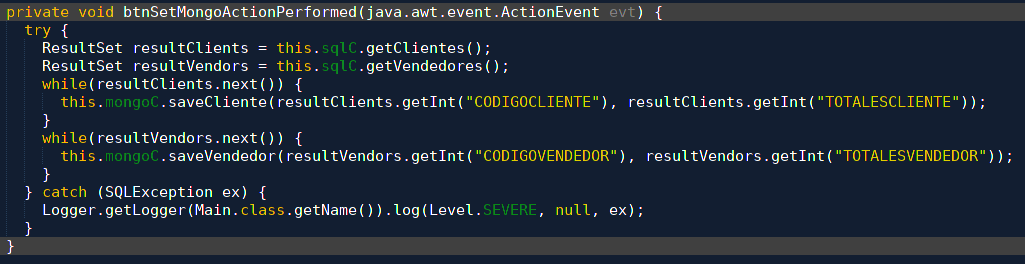




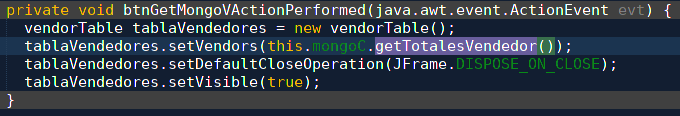
Con el manejo de datos de SQL listos creamos una clase para manejar el envío y visualización de los datos en Mongo así:



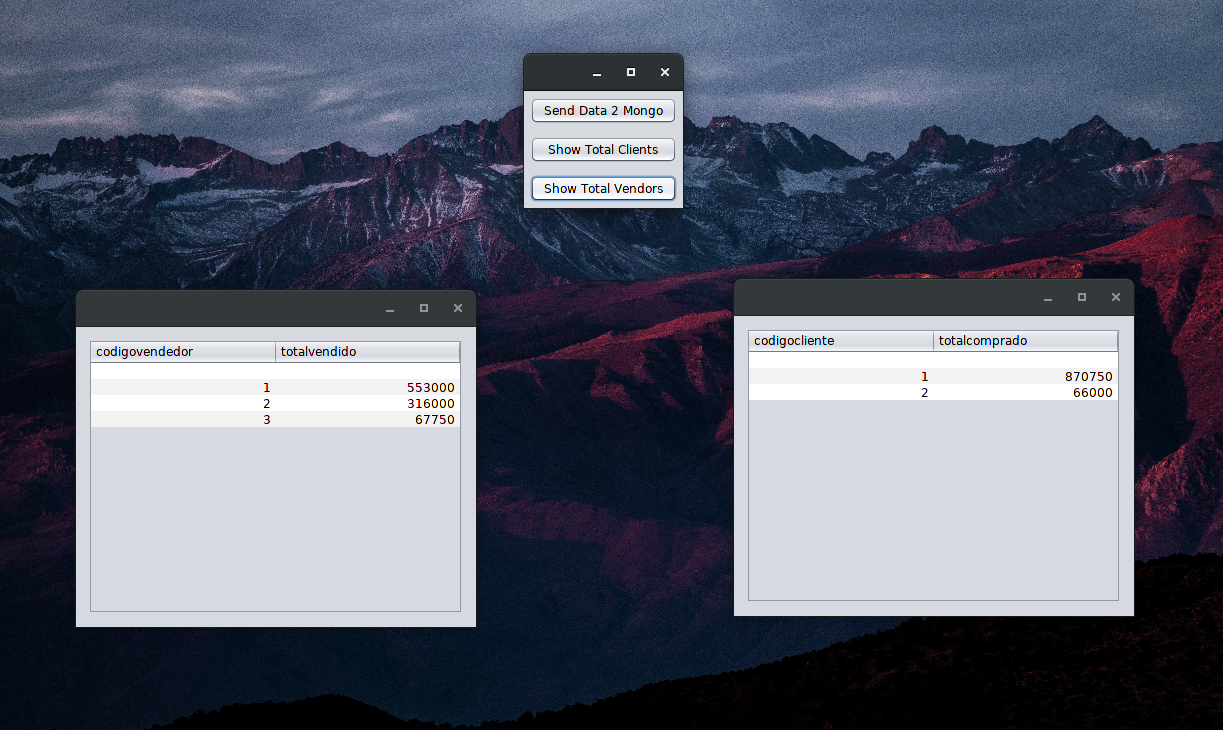
Ya con estas dos construcciones listas y configuradas podemos crear una interfaz y usar estas clases ya listas para poblar las colecciones de Mongo.



También se simplifica mucho el poblar un JFrame con los datos desde Mongo.



**Resultados:**



**Notas:**

* Se omiten más detalles del manejo en la UI por simplicidad y porque se sale del alcance de este curso.
* Se usaron las tablas definidas en el trabajo sin modificaciones y las convenciones de nombres dadas para mongo
* La única adición a la configuración de mongo fue asignar codigocliente y codigovendedor como índices a las colecciones para garantizar que no se agreguen repetidos y poderlos actualizar con el método upsert (Clase ConnectMongo)