**Informe tercer trabajo - Bases de Datos II – 2021-1**

***Integrantes:***

*Aristizabal Giraldo Salomé - Rendón Giraldo Sebastián - Valencia Zapata Santiago Alexis*

Para ejecutar el programa se requiere de instalar Maven 3.8.2 o superior, y un JDK de Java 11 o superior.

Al ejecutar el programa la primera vista es la de ingresar parámetros para las respectivas conexiones a las bases de datos, tanto la de Oracle como la de MongoDB, como el puerto, usuario, contraseña y nombre de la base de datos. Algunos campos tienen valores por defecto y en caso de no requerirlo, es mejor no tocarlos. Solo se puede avanzar si todos los campos tienen algún valor, y se asume que se ingresan bien los datos, por lo que no se verifican. Una vez hecho esto, se pasa al menú dónde se encuentran los botones requeridos en el trabajo.

1. Generar estadísticas

Con los datos de ventas cargados en la base de datos de Oracle, al presionar este botón primero se llama al método estático *obtenerDepartamentos* de la clase *DepartamentoOracle* que retorna un *ArrayList* de *DepartamentoOracle*, objeto que se compone del nombre y de un *ArrayList* de *CiudadesOracle*. El método realiza la consulta "SELECT nom AS nombre FROM departamento ORDER BY nom" para obtener los nombres de los departamentos, y luego itera sobre estos para obtener sus ciudades con el método de *CiudadOracle*, *obtenerCiudadesPorDepartamento*, que retorna un *ArrayList* de estos objetos que se componen del nombre, el total de ventas y el mejor vendedor, que es otro objeto *EmpleadoOracle*, que se compone de la cédula y del valor total de ventas. Los nombres de las ciudades se consultan con "SELECT c.nom AS nombre FROM ciudad c WHERE c.midep.nom= 'departamento' ORDER BY c.nom" , las ventas totales con "SELECT SUM(v.nro\_unidades\*v.miprod.precio\_unitario) AS ventas FROM empleado e, TABLE(e.ventas) v WHERE e.miciu.nom = 'ciudad' GROUP BY e.miciu.nom" y el mejor vendedor por ciudad se obtiene con la consulta "SELECT cedula, total FROM(SELECT e.cc as cedula, SUM(v.nro\_unidades\*v.miprod.precio\_unitario) AS total FROM empleado e, TABLE(e.ventas) v WHERE e.miciu.nom = 'ciudad' GROUP BY e.cc ORDER BY total DESC) WHERE rownum=1".

Una vez se retorna el *ArrayList* con los departamentos y su información, este se ingresa dentro del método de *EstadisticaMongoDB*, *addEstadisticas*, con el parámetro *deleteAllBefore* en *true*, que indica que debe borrar los registros de estadísticas anteriores presentes en la base de datos de MongoDB. El método crea una colección llamada *estadísticas* si no existe, y luego la llama, para iterar sobre la lista de departamentos para crearle un documento del tipo: {departamento: String, misventas: [{ciudad: String, totalVentas: Int, vendedor: {cedula: Int, ventas: Int}}]}, cuidando de posibles faltas de ciudades, empleados o ventas, para luego insertarlo en la colección *estadísticas*.

1. Visualizar estadísticas

Para la visualización de los datos, tanto por departamento como globales, se definió la clase:

EstadisticaMongoDB(DepartamentoMongoDB departamento, CiudadMongoDB mejorCiudad, EmpleadoMongoDB mejorVendedor, EmpleadoMongoDB peorVendedor)

Para las estadísticas por departamento, el método getEstadisticasDepartamento realiza la consulta sobre la colección “estadísticas”, dentro de la base de datos de MongoDB. La consulta es realiza con operaciones de agregación, aprovechando las *Aggregation Pipeline Stages* y *Aggregation Pipeline Operators*, que realizan operaciones sobre el resultado de la operación anterior.

El paso a paso es el siguiente:

1. { $unwind: { path: “$misventas” } }

Deconstruye el arreglo mis ventas del documento y devuelve un documento por cada elemento, cada documento es el mismo documento, con el valor del elemento reemplazando el arreglo.

1. { $sort: { “misventas.totalVentas”: -1 } }

Ordena los documentos según las ventas de cada ciudad, de mayor a menor.

1. { $group: { \_id: “$departamento”, totalVentas: { $sum: “$misventas.totalVentas” }, mejorCiudadNombre: { $first: “$misventas.ciudad” }, mejorCiudadVentas: { $first: “$misventas.nombre” }, misventas: { $push: “$misventas” } } }

Agrupa los documentos por departamento, y calcula el total de ventas del departamento, la mejor ciudad (como la primera) y el arreglo misventas por departamento.

1. { $unwind: { path: “$misventas” } }

Deconstruye el arreglo misventas generado en el paso anterior.

1. { $sort: { “misventas.vendedor.ventas” } }

Ordena los documentos según las ventas de cada vendedor, de mayor a menor, para hallar el mejor y peor vendedor.

1. { $match: { “misventas.vendedor”: { $ne: null } } }

Remueve los documentos que contengan vendedores nulos.

1. { $group: { \_id: “$\_id”, totalVentas: { $first: “$totalVentas” }, mejorCiudadNombre: { $first: “$mejorCiudadNombre” }, mejorCiudadVentas: { $first: “$mejorCiudadVentas” }, mejorVendedor: { $first: “$misventas.vendedor” }, peorVendedor: { $last: “$misventas.vendedor” } } }

Reagrupa los documentos por departamento, que desde la operación 3 se llama “\_id”, recupera totalVentas, mejorCiudadNombre y mejorCiudadVentas de las operaciones anteriores, y calcula el mejor y peor vendedor, como el primer y último vendedor respectivamente. Este es el resultado final de la consulta.

Para las estadísticas globales, el método getEstadisticasGlobales realiza la consulta sobre la misma colección, utilizando también operaciones de agregación.

1. { $unwind: { path: “$misventas” } }

Deconstruye el arreglo misventas.

1. { $sort: { “misventas.totalVentas”: -1 } }

Ordena los documentos según las ventas de cada ciudad.

1. { $group: { \_id: “$departamento”, totalDepto: { $sum: “$misventas.totalVentas” }, departamento: { $first: “$departamento” }, misventas: { $push: “$misventas” } } }

Agrupa por departamento, calcula el total de ventas de cada departamento, su nombre y el arreglo misventas por departamento.

1. { $unwind: { path: “$misventas” } }

Deconstruye el arreglo misventas generado en el paso anterior.

1. { $sort: { “totalDepto”: -1 }

Ordena según el total de ventas del departamento.

1. { $group: { \_id: “”, totalDepto: { $first: “$totalDepto” }, departamento: { $first: “$departamento” }, mejorCiudadNombre: { $first: “$misventas.ciudad” }, mejorCiudadVentas: { $first: “$misventas.totalVentas” }, misventas: { $push: “$misventas” } } }

Agrupa los documentos en un único documento, calcula el mejor departamento y la mejor ciudad y recalcula el arreglo misventas.

1. { $unwind: { path: “$misventas” } }

Deconstruye el arreglo misventas generado en el paso anterior.

1. { $sort: { “misventas.vendedor.ventas”: -1 } }

Ordena los documentos según las ventas de cada vendedor, en orden descendiente.

1. { $match: { “misventas.vendedor”: { $ne: null } } }

Remueve los documentos que contengan vendedores nulos.

1. { $group: { \_id: “”, totalDepto: { $first: “$totalDepto” }, departamento: { $first: “$departamento” }, mejorCiudadNombre: { $first: “$mejorCiudadNombre” }, mejorCiudadVentas: { $first: “$mejorCiudadVentas” }, mejorVendedor: { $first: “$misventas.vendedor” }, peorVendedor: { $last: “$misventas.vendedor” } } }

Agrupa todos los documentos en un único documento, recupera el mejor departamento, la mejor ciudad y calcula el mejor y el peor vendedor, como el primer y último elemento, respectivamente.

Por simplicidad, solo se muestran los departamentos que tengan al menos una ciudad con al menos un empleado y al menos una venta en su respectivo *varray*, cualquier otro caso, el departamento es ignorado.

3. Vaciar arreglos

Se crea una tabla auxiliar: “CREATE TABLE aux(cc NUMBER(15) PRIMARY KEY, totalacumuladoventas NUMBER(30) NOT NULL, nuevototal NUMBER(30) NOT NULL)”; Esta servirá para calcular los acumulados de las ventas de los empleados.

“INSERT INTO aux (cc, nuevototal, totalacumuladoventas) SELECT cc, 0, 0 FROM empleado;”

Esta tabla se llena con el identificador único del vendedor (cc) además las dos columnas de numeros se llenan con 0 en primera instancia dado que debemos trabajar con datos no dos columnas no nulas.

“UPDATE aux SET nuevototal = (SELECT historicoventas.totalacumuladoventas FROM historicoventas WHERE historicoventas.cc=aux.cc) WHERE EXISTS (SELECT historicoventas.totalacumuladoventas FROM historicoventas WHERE historicoventas.cc = aux.cc);” Se llena la columna nuevototal de la tabla aux con los valores de historicoventas.totalacumuladoventas si existen, en caso contrario se mantiene en 0.

“DELETE FROM historicoventas;” Dado que ya se habían guardado los datos existentes de historicoventas en la tabla aux se eliminan todos los datos de historicoventas.

“INSERT INTO historicoventas (cc, totalacumuladoventas) SELECT cedula, total FROM(SELECT e.cc as cedula, SUM(v.nro\_unidades\*v.miprod.precio\_unitario) AS total FROM empleado e, TABLE(e.ventas) v GROUP BY e.cc ORDER BY cc DESC);” Se insertan los nuevos datos de ventas por empleado a historicoventas realizando la operación nro\_unidades\*precio\_unitario a través de las referencias a productos guardadas en empleado.ventas.

“UPDATE aux SET totalacumuladoventas = (SELECT historicoventas.totalacumuladoventas FROM historicoventas WHERE historicoventas.cc=aux.cc) WHERE EXISTS (SELECT historicoventas.totalacumuladoventas FROM historicoventas WHERE historicoventas.cc = aux.cc);”. Se agregan estos nuevos valores de historicoventas.totalacumuladoventas a la tabla aux en la columna totalacumuladoventas si existen, de lo contrario se mantiene el 0.

“DELETE FROM historicoventas;”

“INSERT INTO historicoventas (cc, totalacumuladoventas) SELECT cedula, total FROM(SELECT e.cc as cedula, SUM(a.nuevototal+a.totalacumuladoventas) AS total FROM empleado e, aux a WHERE e.cc = a.cc GROUP BY e.cc ORDER BY e.cc DESC);”

Se borran los datos de historicoventas y se insertan los datos finales los cuales corresponden a la suma de aux.nuevototal+aux.acumuladoventas.

DROP TABLE aux;

UPDATE empleado SET ventas=NULL;

Finalmente se elimina la tabla auxiliar y se borran las ventas de todos los empleados.