TRABAJO PRÁCTICO MONGODB

* Materias: Base de datos
* Profesora: Verónica Pineyro
* Entrega: 24/9/21
* Integrantes: Bruno Calabrese, Martín Miranda, Santiago Nemi, Agustín Perez Rossi, Emanuel Ruiz

Tareas a desarrollar:

Hacer el diseño de las collections.

Insertar datos en las mismas.

Tomar los 3 Stored Procedures y 3 Stored Functions y definir la o las consultas necesarias a través de los métodos provistos por MongoDB para resolver la funcionalidad de dichos procedures y functions.

Diseño de las collections:

Pedido:

{

id:\_\_\_\_\_\_\_\_,

glober:

{

nombre:\_\_\_\_\_\_\_,

dni:\_\_\_\_\_\_\_\_,

telefono:\_\_\_\_\_\_\_\_\_,

mail:\_\_\_\_\_\_\_\_\_\_\_,

horasAcumuladas:\_\_\_\_\_\_\_\_

},

usuario:

{

nombre:\_\_\_\_\_\_\_,

dni:\_\_\_\_\_\_\_\_,

telefono:\_\_\_\_\_\_\_\_\_,

mail:\_\_\_\_\_\_\_\_\_\_\_,

horasAcumuladas:\_\_\_\_\_\_\_\_

},

estadoPedido:\_\_\_\_\_\_\_\_\_\_\_,

estadoGlober:\_\_\_\_\_\_\_\_\_\_\_,

medioPago:\_\_\_\_\_\_\_\_\_\_\_,

fecha\_hora\_final:\_\_\_\_\_\_\_\_\_\_,

fecha\_hora\_inicio:\_\_\_\_\_\_\_\_\_\_\_\_,

detalles:

[

{

producto:

{

id:\_\_\_\_\_\_\_,

nombre:\_\_\_\_,

},

cantidad:\_\_\_,

precio\_total:\_\_\_\_\_,

}

]

}

Producto

{

id:\_\_\_\_\_\_\_\_,

nombre:\_\_\_\_\_\_\_\_,

desc:\_\_\_\_\_\_\_\_\_\_\_\_\_,

precio:\_\_\_\_\_\_\_\_\_\_\_\_\_,

empresa:

{

razonSocial:\_\_\_\_\_\_\_\_,

comision:\_\_\_\_\_\_\_\_\_\_\_

}

}

**PROCEDIMIENTOS**:

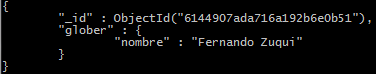
borrarProductos(IN empresa VARCHAR(30)): recibe el nombre de la empresa y borra sus productos que no estén en pedidos activos.

1. Hace un query en producto que trae todos los productos de la empresa recibida por parametro. **db.producto.find({empresa:*"empresa recibida"*},{\_id:0,id:1})**
2. Por cada id de producto de la consulta anterior se fija que el producto no esté en ningún pedido activo. **db.pedido.find({"detalles.producto.id":*id del producto del cursor*, estadoPedido:{"$ne":"activo"}})**
3. Si la consulta anterior no devuelva nada lo borra. **db.producto.deleteOne({id : *id del producto*})**



pedidosGlover(): retorna los pedidos activos de los glovers que tengan menos de 20 horas acumuladas.

1. **db.pedido.find({estadoPedido : "activo","glober.horasAcumuladas" : {$lt : 20}},{"glober.nombre" : 1}).pretty()**



incrementarComision(): Incrementa 10 a la comisión de la empresa cuyo producto esta en el pedido más caro.

1. Busca el id del producto más caro vendido. **db.pedido.find({}, {"detalles.producto.id" : 1, \_id : 0}).sort({"detalle.precio\_total" : 1}).limit(1)**
2. Incrementa 10 a la comision de la empresa del producto obtenido en la consulta anterior. **db.producto.updateOne({id : 1}, {$inc : {"empresa.comision" : 10}})**



**FUNCIONES:**

agregarProductos( idPedido INT, idProducto INT, cantidad INT):se le pasa una id de pedido, de producto y la cantidad, y hace un insert en los detalles del pedido. Después devuelve el precio total del pedido.

1. Busca el nombre y precio del producto y los guarda en dos variables. **db.producto.find({"id": *idPedido*},{id\_:0,"nombre":1,”precio”:1})**
2. Hace el insert en los detalles de la consulta. **db.pedido.updateOne({"id": *id del pedido pasada por parámetro*},{$push: {detalles:{producto:{id: *idProducto*, nombre: *nombre del producto que encontró en la consulta anterior* },cantidad: cantidad,precio\_total: *cantidad* \* *precio que se encontró en la consulta anterior* }}}).**
3. Calcula el precio total del pedido. **db.pedido.aggregate([{"$match":{id:*idPedido*}},{"$project":{"Total":{"$sum":"$detalles.precio\_total"}}])**



cambiarPrecio(idProducto INT): modifica el precio de un producto, cambia el precio total en todos los pedidos que estén activos y devuelve la cantidad de documentos actualizados.

1. Hace un update en la tabla producto. **db.producto.updateOne({id:*idRecibida*},{precio:'precio nuevo'})**
2. Busca todos los pedidos que esten activos y contengan el producto. **db.pedido.find({estadoPedido:"activo","detalles.producto.id": *idRecibida* },{\_id:0,detalles:1})**
3. Por cada fila que obtiene de la consulta anterior, multiplica la cantidad del pedido por el precio del producto y lo guarda en una variable.
4. Por cada fila que obtiene de la consulta en el paso 2, actualiza el precio total por el nuevo precio total que se calcula multiplicando la cantidad por el precio nuevo del producto. También incrementa una variable que sirve como contador de pedidos cambiados **db.pedido.updateOne({},{$set:{“detalles.precio\_total”: *precio total nuevo*}})**



plataGanada(IN idEmpresa INT,OUT monto INT): recibe un código de empresa y devuelve la plata que ganó por las comisiones.

1. Guarda en una variable la comisión de la empresa **db.producto.find({empresa:*"empresa recibida"*},{\_id:0,id:1})**
2. Hace un query que trae todos los productos de la empresa recibida. **db.producto.find({empresa:*"empresa recibida"*},{\_id:0,id:1})**
3. Por cada producto que obtiene de la consulta anterior se fija los precios totales de los productos. **db.pedido.find({detalles.producto.id: *idProducto*},{\_id:0,”detalle.precio\_total”:1})**
4. Se guarda cada precio total en una variable y luego le suma al monto el porcentaje del precio que le corresponde (comisión).