//TRABALHO FEITO EM DUPLA

// MATEUS SCHEFFER RODRIGUES E LEONARDO MATOS DE SOUZA

Atividade Semestral

Banco de Dados - 2023-1

Pode ser feito em duplas ou individual.

Título do Projeto: Sistema de Gerenciamento de Pedidos para uma Pizzaria

Objetivos

Desenvolver um sistema que permita o gerenciamento de pedidos em uma pizzaria utilizando MongoDB. O sistema deve permitir a inserção, consulta, atualização e exclusão de dados (operações CRUD).

Descrição

Nesta atividade, vocês irão aplicar os conceitos aprendidos sobre banco de dados NoSQL, especificamente utilizando MongoDB, para implementar um sistema de gerenciamento de pedidos para uma pizzaria.

A pizzaria oferece vários tipos de pizzas, com tamanhos e preços diferentes, além de bebidas. Os clientes podem fazer pedidos para entrega ou para comer no local. Cada pedido pode conter múltiplas pizzas e bebidas, e deve-se manter um registro dos detalhes do cliente (nome, endereço para entrega, telefone), data e hora do pedido, itens do pedido e o total do pedido.

Regras de negócio:

* Cada cliente pode ter múltiplos pedidos, mas cada pedido pertence a um único cliente.
* Cada pedido pode ter várias pizzas e/ou bebidas, com quantidades variáveis.
* As pizzas têm tamanhos (pequena, média, grande) que afetam o preço. O preço e as características de cada pizza devem ser armazenados.
* Os pedidos para entrega devem incluir informações de endereço. Os pedidos para comer no local devem ter a informação do número da mesa.
* O total do pedido deve ser calculado com base nos itens do pedido.

**Cliente:**

nome (String)

endereco\_entrega (json)

telefone (String)

Json>

**Endereco\_entrega**

data\_hora (datetime)

taxa\_entrega (double)

rua (Varchar)

numero (String)

complemento (Varchar)

cidade (Varchar)

estado (Varchar)

cep (Varchar)

**Pedido:**

cliente\_id (Referência ao ID do cliente)

data\_hora (Data e hora do pedido)

tipo\_entrega (String: entrega ou local)

numero\_mesa (Número da mesa para pedidos locais)

itens (Array de objetos contendo os itens do pedido)

**Item:**

tipo (String: pizza ou bebida)

nome (String)

quantidade (int)

preco\_unitario (Número decimal)

**Pizza:**

\_id

nome (String)

tamanho (String: pequena, média, grande)

preco\_unitario (double)

quantidade (Número inteiro)

**Bebida:**

\_id

nome (String)

preco\_unitario (double)

quantidade (Número inteiro)

db.createCollection("clientes")

db.createCollection("pedidos")

db.createCollection("pizzas")

db.createCollection("bebidas")

Entregáveis:

* Diagrama de entidade e relacionamento (DER) para o banco de dados NoSQL. (pode fazer no figma a modelagem)



* **Estruturas JSON para todas as entidades identificadas na modelagem.**

**CLIENTE**

{

"nome": "Nome do Cliente",

"endereco\_entrega": {

"rua": "Rua do Cliente",

"numero": "Número do Cliente",

"complemento": "Complemento do Cliente",

"cidade": "Cidade do Cliente",

"estado": "Estado do Cliente",

"cep": "CEP do Cliente"

},

"telefone": "Telefone do Cliente"

}

**PEDIDO**

{

"cliente\_id": "ID do Cliente",

"data\_hora": "Data e Hora do Pedido",

"tipo\_entrega": "Tipo de Entrega (entrega ou local)",

"numero\_mesa": "Número da Mesa (para pedidos locais)",

"itens": [

{

"tipo": "Tipo do Item (pizza ou bebida)",

"nome": "Nome do Item",

"quantidade": "Quantidade do Item",

"preco\_unitario": "Preço Unitário do Item"

},

{

"tipo": "Tipo do Item (pizza ou bebida)",

"nome": "Nome do Item",

"quantidade": "Quantidade do Item",

"preco\_unitario": "Preço Unitário do Item"

},

...

]

}

**PIZZA**

{

"nome": "Nome da Pizza",

"tamanho": "Tamanho da Pizza (pequena, média, grande)",

"preco\_unitario": "Preço Unitário da Pizza",

"quantidade": "Quantidade de Pizzas"

}

**BEBIDA**

{

"nome": "Nome da Bebida",

"preco\_unitario": "Preço Unitário da Bebida",

"quantidade": "Quantidade de Bebidas"

}

* **Comandos MongoDB para criar as coleções e inserir os dados.**

CRIAÇÃO E POPULAÇÃO DAS TABELAS

db.clientes.insertMany([

{

"nome": "João Silva",

"endereco\_entrega": "Rua A, 123",

"telefone": "(11) 987654321"

},

{

"nome": "Maria Souza",

"endereco\_entrega": "Avenida B, 456",

"telefone": "(22) 123456789"

}

])

db.pedidos.insertOne({

"data\_hora": new Date(),

"tipo\_entrega": "entrega",

"numero\_mesa": null,

"itens": [

{

"tipo": "pizza",

"nome": "Calabresa",

"quantidade": 2,

"preco\_unitario": 25.99

},

{

"tipo": "bebida",

"nome": "Coca-Cola",

"quantidade": 3,

"preco\_unitario": 5.99

}

]

})

db.pizzas.insertMany([

{

"nome": "Calabresa",

"tamanho": "grande",

"preco\_unitario": 25.99,

"quantidade": 10

},

{

"nome": "Mussarela",

"tamanho": "média",

"preco\_unitario": 20.99,

"quantidade": 5

}

])

db.bebidas.insertMany([

{

"nome": "Coca-Cola",

"preco\_unitario": 5.99,

"quantidade": 20

},

{

"nome": "Guaraná",

"preco\_unitario": 4.99,

"quantidade": 15

}

])

* **Comandos MongoDB para realizar atualizações e exclusões de dados.**

**Atualização de um documento na coleção "clientes":**

mudando o telefone da cliente Maria souza

db.clientes.updateOne(

{ "\_id": ObjectId("64930f901e3a4390f382613a") },

{ $set: { "telefone": "(48) 977777779" } }

)

Mudando de tipo de entrega local para entrega

**Atualização de um documento na coleção "pedidos":**

db.pedidos.updateOne(

{ "\_id": ObjectId("64934b3a6d45a362c908cb24") },

{ $set: { "tipo\_entrega": "local" } }

)

adicionando item num array

**Adição de um item em um array de documentos na coleção "pedidos":**

db.pedidos.updateOne(

{ "\_id": ObjectId("64934b3a6d45a362c908cb24") },

{ $push: { "itens": { "tipo": "pizza", "nome": "Mussarela", "quantidade": 2, "tamanho": "media", "preco\_unitario": 22.99 } } }

)

**Remoção de um documento na coleção "clientes":**

db.clientes.deleteOne({ "\_id": ObjectId("64930f901e3a4390f3826139") })

* **Comandos MongoDB para as consultas que devem incluir pelo menos uma de cada: consulta simples, consulta com uma condição.**

Consulta simples para retornar todos os documentos da coleção "clientes":

**db.clientes.find()**

Consulta com uma condição para retornar os pedidos com tipo de entrega igual a "entrega":

**db.pedidos.find({ "tipo\_entrega": "local" })**

Consulta para mostrar os clientes que pediram a bebida Coca-Cola:

**db.pedidos.find({ "itens.nome": "Coca-Cola", "itens.tipo": "bebida" })**

* **Comando MongoDB com 2 consultas usando Aggregate.   
   uma que utilize join e outra que utilize group by.**

**agregate para saber o valor total dos itens, o nome de quem pediu e a mesa em que sentou**

db.pedidos.aggregate([

{

$match: {

tipo\_entrega: "local"

} },

{

$lookup: {

from: "clientes",

localField: "cliente\_id",

foreignField: "\_id",

as: "cliente"

}

},

{

$unwind: "$cliente"

},

{

$project: {

valor\_total: {

$reduce: {

input: "$itens",

initialValue: 0,

in: {

$add: ["$$value", { $multiply: ["$$this.quantidade", "$$this.preco\_unitario"] }]

}

}

},

nome: "$cliente.nome",

mesa: "$numero\_mesa"

}

}

]);

**.GROUP BY para mostrar a quantidade de pedidos entrega e local e a quantidade de itens**

db.pedidos.aggregate([

{

$group: {

\_id: "$tipo\_entrega",

total\_pedidos: { $sum: 1 },

total\_itens: { $sum: { $size: "$itens" } }

}

}

]);

**AQUI usamos o group by de novo mas para contar o cliente que mais consumiu itens**

db.pedidos.aggregate([

{

$unwind: "$itens"

},

{

$group: {

\_id: "$cliente\_id",

total\_itens: { $sum: "$itens.quantidade" }

}

},

{

$sort: {

total\_itens: -1

}

},

{

$limit: 1

},

{

$lookup: {

from: "clientes",

localField: "\_id",

foreignField: "\_id",

as: "cliente"

}

},

{

$unwind: "$cliente"

},

{

$project: {

"cliente.nome": 1,

total\_itens: 1

}

}

]);

**aqui usamos o group by para ver qual foi o item mais vendido**

db.pedidos.aggregate([

{

$unwind: "$itens"

},

{

$group: {

\_id: "$itens.nome",

quantidade\_total: { $sum: "$itens.quantidade" }

}

},

{

$sort: {

quantidade\_total: -1

}

},

{

$limit: 5

}

]);

* **Índices apropriados para melhorar o desempenho das consultas. Explicar por que desses índices.**

Índice para busca por nome do cliente:

db.clientes.createIndex({ "nome": 1 })

Esse índice ajudará a otimizar as consultas que buscam clientes com base no nome.

Índice para busca por pizzas mais consumidas:

db.pizzas.createIndex({ "quantidade": -1 })

Esse índice ordena as pizzas com base na quantidade, permitindo que você recupere as pizzas mais consumidas com mais eficiência.

Índice para busca por bebidas mais consumidas:

db.bebidas.createIndex({ "quantidade": -1 })

Esse índice ordena as bebidas com base na quantidade, permitindo que você recupere as bebidas mais consumidas com mais eficiência.

Índice para busca por valores:

db.pizzas.createIndex({ "preco\_unitario": 1 })

db.bebidas.createIndex({ "preco\_unitario": 1 })

Esses índices ajudam a otimizar consultas que envolvem a busca por pizzas ou bebidas com base no preço unitário.

* Deve ser entregue o link do github com a modelagem, os JSONs e os scripts MongoDB.