**Nome**: Marcos Souza

**BD noSQL -** classe valor

**Prof:** BRAZ IZAIAS DA SILVA JUNIOR **– Ciência de dados – 5ºSemestre.**

**Modelagem utilizando Dicionário ou mapa:**

**Coleção de Personalidades**:

Nesta coleção, estamos armazenando informações gerais sobre cada personalidade, como nome, data de nascimento, nacionalidade, descrição e características principais. O \_id é um campo único que serve como identificador exclusivo para cada documento. Aqui está a explicação dos campos:

* \_id: Identificador único gerado automaticamente para cada documento.
* nome: Nome da personalidade.
* data\_nascimento: Data de nascimento da personalidade.
* nacionalidade: Nacionalidade da personalidade.
* descricao: Breve descrição sobre a personalidade.
* caracteristicas\_principais: Uma lista de características principais da personalidade, como genialidade, curiosidade, inovação, etc.

**Coleção de Bibliografias:**

Nesta coleção, estamos armazenando informações sobre as bibliografias associadas a cada personalidade. Cada documento nesta coleção contém informações sobre uma bibliografia específica, incluindo o \_id, um campo para identificação exclusiva, e personalidade\_id, que faz referência ao \_id da personalidade associada. Aqui está a explicação dos campos:

* \_id: Identificador único gerado automaticamente para cada documento.
* personalidade\_id: Referência ao \_id da personalidade associada a esta bibliografia.
* titulo: Título da bibliografia.
* autor: Autor da bibliografia.
* ano\_publicacao: Ano de publicação da bibliografia.
* genero: Gênero da bibliografia, como "Física", "Literatura", etc.
* descricao: Breve descrição da bibliografia.

**Conexão com o MongoDB em Python:**

A parte de código Python apresentada serve para conectar-se ao servidor MongoDB, criar ou acessar um banco de dados chamado bibliografias\_personalidades, acessar as coleções personalidades e bibliografias, e inserir dados nos documentos dessas coleções. Aqui estão os principais passos do código:

1. Importamos a classe MongoClient da biblioteca pymongo para estabelecer a conexão com o servidor MongoDB.
2. Criamos uma instância do cliente MongoDB apontando para o endereço do servidor (no exemplo, o servidor é local).
3. Acessamos o banco de dados bibliografias\_personalidades (se não existir, o MongoDB o criará automaticamente).
4. Acessamos as coleções personalidades e bibliografias dentro desse banco de dados.
5. Inserimos dados na coleção personalidades usando o método insert\_one(). O retorno deste método é o \_id gerado automaticamente para o documento inserido.
6. Usamos esse \_id para inserir dados na coleção bibliografias, relacionando a bibliografia com a personalidade correspondente.

Por fim, fechamos a conexão com o MongoDB para garantir que todos os recursos sejam liberados adequadamente.

Este é um exemplo inicial e simples de como vamos estruturar e armazenar informações usando o MongoDB para uma aplicação de bibliografias de personalidades. À medida que nossa aplicação crescer, vamos poder adicionar mais campos, desenvolver consultas mais avançadas e melhorar a estrutura conforme necessário.

Fazendo a aplicação no Jupyter Notebook com o mongo db:

from pymongo import MongoClient

# Conectar ao servidor MongoDB

client = MongoClient(nossa conexão)

# Criar ou acessar um banco de dados

db = client["bibliografias\_personalidades"]

# Acessar coleções

colecao\_personalidades = db["personalidades"]

colecao\_bibliografias = db["bibliografias"]

# Inserir dados na coleção de personalidades

personalidade = {

"nome": "Albert Einstein",

"data\_nascimento": "14/03/1879",

"nacionalidade": "Alemão",

"descricao": "Físico teórico famoso por sua teoria da relatividade.",

"caracteristicas\_principais": [

"Genialidade",

"Curiosidade",

"Inovação"

]

}

personalidade\_id = colecao\_personalidades.insert\_one(personalidade).inserted\_id

# Inserir dados na coleção de bibliografias

bibliografia = {

"personalidade\_id": personalidade\_id,

"titulo": "Relatividade: A Teoria Especial e Geral",

"autor": "Albert Einstein",

"ano\_publicacao": "1915",

"genero": "Física",

"descricao": "Livro que apresenta a teoria da relatividade."

}

colecao\_bibliografias.insert\_one(bibliografia)

# Fechar a conexão

client.close()

**Modelagem utilizando Redis:**

O Redis é um banco de dados em memória que armazena dados como pares chave-valor. Para esse caso, vamos criar chaves para cada personalidade e usar estruturas de dados do Redis, como hashes e listas, para armazenar as informações relevantes.

**Chave-Valor para Personalidades:**

Vamos usar um hash do Redis para armazenar informações sobre cada personalidade. A chave da hash será o nome da personalidade.

HSET "**personalidade**:Albert Einstein" "**nome**" "Albert Einstein" "**data**\_nascimento" "1879-03-14" "**nacionalidade"** "Alemão" "**descricao**" "Físico teórico conhecido por sua teoria da relatividade."

**Lista de Características:**

Vamos usar uma lista para armazenar as principais características de cada personalidade.

RPUSH "**caracteristicas**:Albert Einstein" "Genialidade" "Criatividade" "Curiosidade"

**Chave-Valor para Bibliografias:**

Vamos usar um hash do Redis para armazenar informações sobre cada bibliografia. A chave da hash será um identificador único.

HSET "bibliografia:1" "personalidade" "Albert Einstein" "titulo" "A Teoria da Relatividade" "autor" "Albert Einstein" "ano\_publicacao" "1915" "genero" "Física" "descricao" "Um dos trabalhos mais famosos de Einstein sobre a teoria da relatividade."

**Conexão com Redis em Python:**

Aqui vamos conectar ao Redis e inserir os dados usando Python com a biblioteca redis-py:

import redis

# Conectando ao Redis

redis\_client = redis.Redis(host='localhost', port=6379, db=0)

Dados da personalidade personalidade = { "nome": "Albert Einstein", "data\_nascimento": "1879-03-14", "nacionalidade": "Alemão", "descricao": "Físico teórico conhecido por sua teoria da relatividade." }

# Inserindo informações da personalidade

redis\_client.hmset("personalidade:Albert Einstein", personalidade)

Principais características caracteristicas = ["Genialidade", "Criatividade", "Curiosidade"]

# Inserindo características na lista

redis\_client.rpush("caracteristicas:Albert Einstein", \*caracteristicas)

Dados da bibliografia bibliografia = { "personalidade": "Albert Einstein", "titulo": "A Teoria da Relatividade", "autor": "Albert Einstein", "ano\_publicacao": "1915", "genero": "Física", "descricao": "Um dos trabalhos mais famosos de Einstein sobre a teoria da relatividade." }

# Inserindo informações da bibliografia

redis\_client.hmset("bibliografia:1", bibliografia)

Neste exemplo, usamos a biblioteca redis-py para se conectar ao Redis, armazenar informações sobre personalidades usando hashes, armazenar características usando listas e armazenar informações sobre bibliografias usando hashes.

O Redis é altamente eficiente para armazenamento em memória e é muito rápido para recuperação de dados. No entanto, é importante lembrar de que ele é otimizado para casos em que os dados cabem na memória, e os dados são perdidos após um reboot ou encerramento do servidor.