Bootcamp Analista de Dados

Relacionamentos e PyMongo

SOUL CODE

Relacionamentos com MongoDB

Relacionamentos com MongoDB

- Embed (Incorporar documentos) significa aninhar documentos dentro de outros documentos. Isso é útil quando os dados são frequentemente acessados juntos e a relação entre eles é um para um ou um para poucos.
- Utiliza-se a incorporação de documento quando um documento pertence exclusivamente a outro documento ou quando muitos documentos pertencem a um documento e são frequentemente acessados juntos.

SOUL CODE

Relacionamentos com MongoDB

SOUL CODE

- Considere um cenário com documentos de usuários e endereços:

```
{
  "_id": 1,
  "name": "Jonas",
  "endereco": {
    "cidade": "Franca",
    "estado": "São Paulo",
  }
}
```

Relacionamentos com MongoDB

- Reference (Referenciar documentos) significa armazenar referências (normalmente IDs) de outros documentos. Isso é útil quando os dados são acessados separadamente.
- Utiliza-se a referência de documento quando muitos documentos podem estar relacionados a muitos outros documentos.

SOUL CODE

Relacionamentos com MongoDB

SOUL CODE

- Considere um cenário com documentos de usuários e pedidos:

```
{
  "_id": 1,
  "name": "Jonas",
  "email": "jonas@mail.com",
  "pedidos": [1001, 1002, 1003]
}
```

```
{
  "_id": 1001,
  "produto": "Laptop",
  "quantidade": "Laptop",
  "preco": 1300.9,
  "usuario_id": 1
}
```


SOUL CODE
Python com MongoDB

Python com MongoDB

SOUL CODE

- Bancos de dados são usados para armazenar grandes volumes de dados de forma organizada. Linguagens de programação como Python podem recuperar, manipular e atualizar esses dados conforme necessário para várias aplicações.
- A maioria das aplicações web e móveis modernas dependem de bancos de dados para armazenar informações dos usuários, conteúdo, e configurações.
- Python é amplamente utilizado em ciência de dados e análise de dados. Conectar-se a bancos de dados permite que os cientistas de dados acessem grandes conjuntos de dados para realizar análises.

Python com MongoDB

- Pymongo é uma biblioteca para Python que permite a conexão com o banco de dados MongoDB. Ele fornece uma interface para realizar operações CRUD (Create, Read, Update, Delete) e outras operações no MongoDB.
- Para instalar a biblioteca pymongo no Google Colaboratory, você pode usar o comando pip:

```
%pip install pymongo
```

Python com MongoDB

SOUL CODE

- Para estabelecer uma conexão ao MongoDB usando pymongo é necessário primeiro importar a biblioteca pymongo:

```
import pymongo
```

- MongoClient é usado para se conectar ao servidor MongoDB. No exemplo, estamos conectando ao MongoDB que está rodando localmente:

```
cliente = pymongo.MongoClient("mongodb://localhost:27017")
```

Python com MongoDB

SOUL CODE

- A seguir o comando retorna um banco de dados com nome especificado:

```
db = cliente.mercado
print(db.name)
print(db.list_collection_names())
```

- O comando retorna uma coleção do banco de dados com nome especificado:

```
collection = db.produtos
print(collection.name)
```

Python com MongoDB

SOUL CODE

- Dados no MongoDB são representados usando documentos no estilo JSON. No PyMongo, usa-se dicionários para representar documentos:

```
produto = {
    "nome": "Abacaxi",
    "preco": 14.6,
    "categorias": ["alimentos", "frutas"],
    "quantidade": 95
}
```

Python com MongoDB

SOUL CODE

- Para inserir um documento em uma coleção podemos usar o método insert_one():

```
produto_inserido = db.produtos.insert_one({
    "nome": "Abacaxi",
    "preco": 14.6,
    "categorias": ["alimentos", "frutas"],
    "quantidade": 95
})
print(produto_inserido.inserted_id)
```

Python com MongoDB

- O tipo mais básico de consulta que pode ser executado no MongoDB é find_one(). Este método retorna um único documento que corresponde a uma consulta:

```
produto = db.produtos.find_one({ "nome": "Abacaxi" })
print(produto)
```

- O resultado é um dicionário correspondente ao que inserimos anteriormente.

Python com MongoDB

- Para obter mais de um documento como resultado de uma consulta, usamos o método find().
- Ele retorna uma estrutura iterável, o que permite acessar todos os documentos correspondentes:

```
produtos = db.produtos.find()

for produto in produtos:

    print(produto)
```

SOUL CODE

Python com MongoDB

- O comando a seguir usa o método update_one() para atualizar o valor de preço de documento na coleção produtos que corresponde ao filtro:

```
db.produtos.update_one(
    { "nome": "Abacaxi" },
    { "$set": { "preco": 15.4 }
)
```

SOUL CODE

Python com MongoDB

- O comando a seguir usa o método delete_one() para remover um documento na coleção produtos que corresponde ao filtro:

```
db.produtos.delete_one({ "nome": "Abacaxi" })
```

- O comando a seguir usa o método delete_many() para remover todos os documentos na coleção produtos que corresponde ao filtro:

```
db.produtos.delete_many({ "nome": "Abacaxi" })
```

SOUL CODE

Python com MongoDB

- O comando a seguir usa o método delete_one() para remover um documento na coleção produtos pelo id:

```
from bson.objectid import ObjectId

produto_object_id = ObjectId("763d9e...")

db.produtos.delete_one({ "_id": produto_object_id })
```

SOUL CODE