Licenciatura Engenharia Informática

Ciência de Dados

Base de Dados MongoDB
2º Semestre

Executado por
45206 Bruno Martins
30156 Pedro Rodrigues
40204 Pedro Pinto

Orientado por
Ricardo Ferreira

Entregue em
05/05/2022

1. Resumo

Dependendo da complexidade do problema em ambiente real, a utilização de uma base de dados não relacional pode ser ideal para requisitos onde não envolve muita lógica ou regras de negócio. Este tipo de base de dados podem ser configurados fácilmente e são adaptativos em relação á estrutura dos mesmos.

Por este motivo, estas base de dados não relacionais, por norma, têm mais têndencia a escalar com mais facilidade.

Não tendo um complexo sistema de gestão de base de dados por trás, aumenta a performance da aplicação, contudo, não garante a segurança extra esperada numa base de dados relacional.

Posto isto, a base de dados mongoDB será utilizada para o presente projeto, de forma a adaptar o trabalho anterior, realizado a partir de um esquema relacional.

2. Índice

1. Resumo **2**

2. Índice **3**

3. Introdução (ou Objectivos) **4**

4. Desenvolvimento **5**

4.1. Estrutura Inicial **5**

4.1.1. Tabela relacional vs Documento mongoDB **5**

4.2. Metodologia Adotada - Normalização ou Referência **7**

4.3. Identificação dos Documentos e Atributos **8**

4.4. Identificação dos Documentos - **document_name** **15**

4.5. Carregamento de Dados - Python **16**

4.6. Casos de Uso Gerados - Python **18**

5. Conclusão **20**

3. Introdução (ou Objectivos)

Conforme indicado acima, será utilizado a ferramenta base de dados mongoDB para criar todos os documentos e inserções de todos os dados.

Foi utilizado a ferramenta python para complementar a resolução do problema, criando um script que faz a conexão á base de dados e insere mais alguns dados adicionais à base de dados.

Será explicado, no desenvolvimento, a estrutura inicial da base de dados, apresentação das entidades e o modo de criação e de relações de cada documento, que estarão presentes na collection principal.

Existirá, uma componente prática, incluida ao projeto com relação à criação de um script, realizado em Python, que fará todo o carregamento de dados para o servidor local da base de dados mongoDB, **utilizando a ferramenta pymongo**.

4. Desenvolvimento

4.1 Estrutura inicial

A estrutura utilizada para a criação da base dados foi feita pela seguinte lógica:

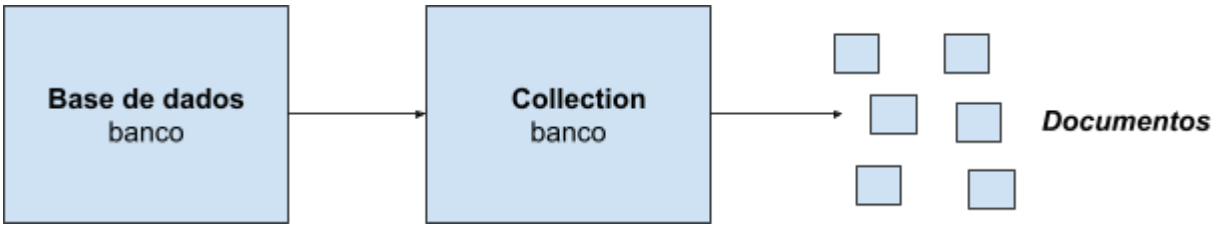

Figura 1: Estrutura lógica da configuração da base de dados

A base de dados, **denominada de banco**, terá apenas uma **collection**, com o mesmo nome para evitar a complexidade de comunicação entre os mesmos.

Dentro da coleção, está presente vários documentos em que cada documento representara cada registo, de uma forma simplificada.

4.1.1 Tabela relacional vs Documento mongoDB

Para representar rapidamente a diferente entre uma tabela e um documento, de forma prática:

- Uma tabela é uma estrutura previamente criada em que contém registos. Portanto, existe vários registos ligados apenas a uma tabela.

- Um documento tem a mesma estrutura da tabela repetida várias vezes, contudo, já se faz presente com os valores já previament definidos.

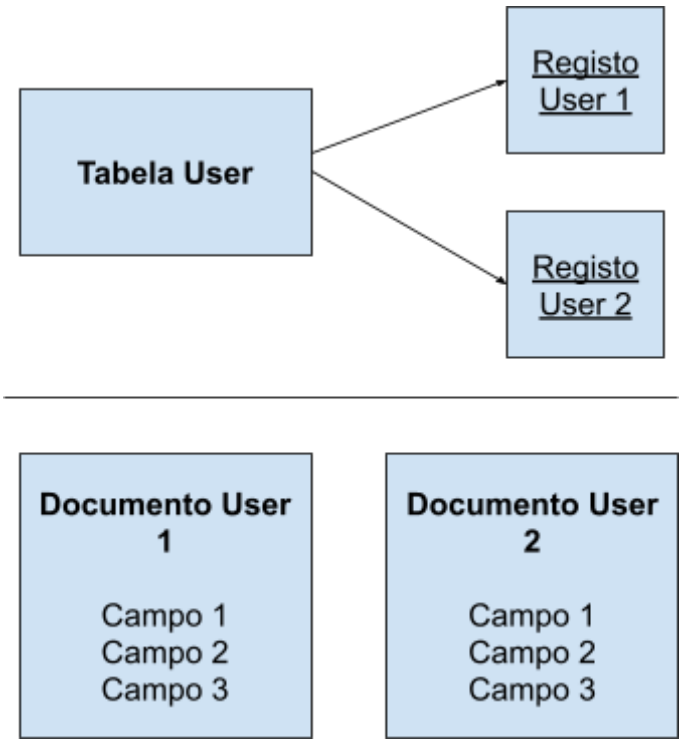

Figura 2: Tabela versus Documentos - estrutura lógica dos dados

4.2 Metodologia Adotada - Normalização ou referência

Para construir todas as relações entre os demais documentos, utilizou-se a metodologia - **Modelo de Dados Normalizados** ou **Base de dados por referência em mongoDB**.

Apesar de, em base de dados não relacionais, a prática de documentos inseridos ser um dos mais populares, acredita-se que possa ser um bocado perigoso de manter uma organização consistente se a base de dados escalar de forma exponêncial.

Quantos mais documentos e relações téoricas, (teóricas porque não ser o padrão normalizado de base de dados), mais atenção se deve ter na consistência de todos os documentos, verificando se todos os “registos” -> documentos possuem o mesmo número de campos. Construir um documento com documentos inseridos não é favorável, segundo a opinião do grupo deste trabalho, para a organização.

Decidindo trazer um bocado da parte teórica da base de dados relacional, torna-se mais simples perceber, mesmo em mongoDB, com uma técnica legal, a ligação/referência entre os documentos. Se vários documentos utilizarem o mesmo documento, evita-se ter que atualizar este mesmo diversas vezes.

4.3 Identificação dos documentos e atributos

No passado trabalhou, a base de dados relacional SQL Server, foi explicado toda a lógica necessária para as ligações. Em mongoDB, a explicação acaba por ser mais facilitada, dado que apenas é necessário indicar a estrutura dos os documentos a serem criados.

Base de dados: **banco**, que contém **uma collection banco**, com documentos em que as seguintes definições são:

1. Documento **User**
    - _id - campo orientado pelo método ObjectId(), gerado automáticamente
    - uuid_user
    - address
    - city
    - phone_number
    - is_active
    - first_name
    - last_name
    - cc_identification
    - nif_identification
    - personal_email
    - dob
    - created_at
    - updated_at
    - **document_name**

2. Documento **Account**
    - **document_name**
    - id_account_type - referência para os documentos de account_type.
    - created_at
    - updated_at
    - is_active
    - balance
    - files- um array ou vetor que guardará todas as referências para os files de “file”. **Estas referências são chaves em strings que representam o _id do documento *File*.**
    - movements - um array ou vetor que guardará todas as referências para os movements de “Movement”. **Estas referências são chaves em strings que representam o _id do documento *Movement*.**

3. Documento **Card**
    - **document_name**
    - id_card_type - referência para os documentos de CardType
    - card_number
    - card_pin_number
    - created_at
    - updated_at
    - valid_until
    - cvv
    - card_holder_name
    - is_active
    - id_account - referência para os documentos de Account.

4. Documento **Holder**
    - **document_name**
    - id_account - referência para os documentos de Account.
    - id_customer - referência para os documentos de Customer.
    - created_at
    - updated_at
    - register_date
    - is_active

5. Documento **HolderPermission**
    - **document_name**
    - id_permission - referência para os documentos de Permission.
    - id_holder - referência para os documentos de Holder.
    - is_permision_active
    - created_at
    - updated_at

6. Documento **Permission**
    - **document_name**
    - permission_designation
    - created_at
    - updated_at

7. Documento **Customer**
    - **document_name**
    - id_user - referência para os documentos de User
    - contract_number
    - pin_code
    - created_at
    - updated_at

8. Documento **AccountType**
    - **document name**
    - id_account_type
    - account_type_designation

9. Documento **File**
    - **document name** - *este document_name é o que identifica o Documento (mongodb).*
    - url
    - created_at
    - updated_at

10. Documento **Movement**
    - **document_name**
    - start_balance
    - end_balance
    - reserved_balance
    - id_account
    - id_operation_type
    - uuid_movement

- created_at

11. Documento **Collaborator**
    - **document_name**
    - id_user
    - work_email_address
    - work_phone_number
    - created_at
    - updated_at

12. Documento **OperationType**
    - **document_name**
    - operation_name
    - created_at

13. Documento **CreditProcessStatus**
    - **document_name**
    - status
    - created_at

14. Documento **Credit**
    - **document_name**
    - amount
    - end_duration
    - initial_duration
    - is_paid
    - created_at
    - updated

15. Documento **CardType**
    - **document_name**
    - card_type_designation
    - created_at

16. Documento **CreditProcess**
    - **document_name**
    - id_credit_process_status - referência para os documentos de credit_process_status
    - id_customer- referência para os documentos de customer
    - id_collaborator - referência para os documentos de collaborator
    - id_account - referência para os documentos de account
    - id_credit - referência para os documentos de credit
    - created_at
    - updated

4.4 Identificação dos documentos - document_name

Como demonstrado, em todas as collections existe um campo desginado por **document_name**.

Este campo poderá suscitar alguma confusão com o contexto do documento **Documents**, documento responsável por guardar todas as informações dos ficheiros.

De forma a identificar unicamente as collections, a estratégia utilizada foi a criação deste campo em todos os documentos.

Vantagens do campo **document_name** para identificação pessoal dos mesmos:

1. Ajuda em possíveis queries, filtrando o documento em questão que se quer procurar, evitando a busca desnecessária de dados.

2. Melhor performance devido á filtragem adicional.

3. Contextualização explicita no código, sabe-se que documento é necessário e permite a divisão de código ao programador, na vertente de poder criar estruturas lógicas e camadas de abstração com mais facilidade, nesse mesmo código.

4.5 Carregamento de dados - Python

O python é uma das linguagens de programação mais simples devido à quantidade de bibliotecas, criadas pela comunidade, que estão disponibilizadas na linguagem. Usada normalmente em construção de gráficos, referentes à análise de dados e de *data science*.

O script em questão tem como objetivo inserir os dados, em formato de dicionários. Em outras palavras, é um array de objetos, **em que cada um terá o padrão chave-valor**, que será enviado como payload para o mongoDB, de modo a serem criados os documentos.

Para este script correr de forma correta, pressupõe-se que a collection banco esteja vazia ou ainda não esteja criada.

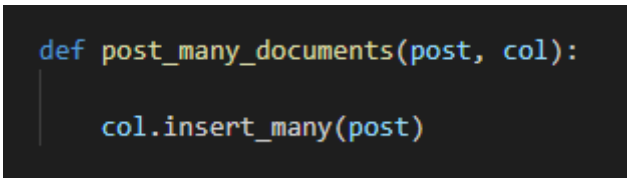

Figura 3: Função *post_many_documents* relativamente à inserção de dados

Após a inserção **dos documentos principais, ou seja, aqueles que não têm documentos com referências**, o código entra na próxima fase onde vai buscar os identificadores únicos de cada tabela.

```
def get_id_from_data(data, arrid, col):

    for tabela in range(len(data)):

        cursor = col.find({"document_name": data[tabela]["document_name"]})
        for document in cursor:

            if (get_documents_id(document) in arrid):
                continue
            else:
                arrid.append(get_documents_id(document))
    return arrid
```

Figura 4: Função *get_id_from_data* que carrega os identificadores

A função consiste em percorrer, pelo array de dicionário, guardando o identificador do documento num único array. Se este identificador já se encontrar no array, não será inserido novamente.

```
def get_documents_id(document):

    return document["_id"]
```

Figura 5: Função *get_documents_id* que retorna o valor do campo "_id"

Após guardado os identificadores únicos no array, os documentos subsequentes, **aqueles que têm campos com valores por referência** serão inseridos, completando assim o carregamento de dados.

4.6 Casos de Uso Gerados - Python

Nos casos de se querer **associar os vários documentos a outro documento** como por exemplo uma conta de utilizador ter vários *files* associados:

- Cria-se uma coluna que vai guardar um **array de identificadores únicos** que estão associados aos documentos necessários que a conta precisa, tendo, contido os ids referentes aos documentos *Files* de modo a que, mediante a eventuais alterações a estes documentos *Files*, serem apenas realizadas uma única vez, ao invés de alterar documento por documento que utilizasse direto a estrutura, se fosse em modo incorporado.

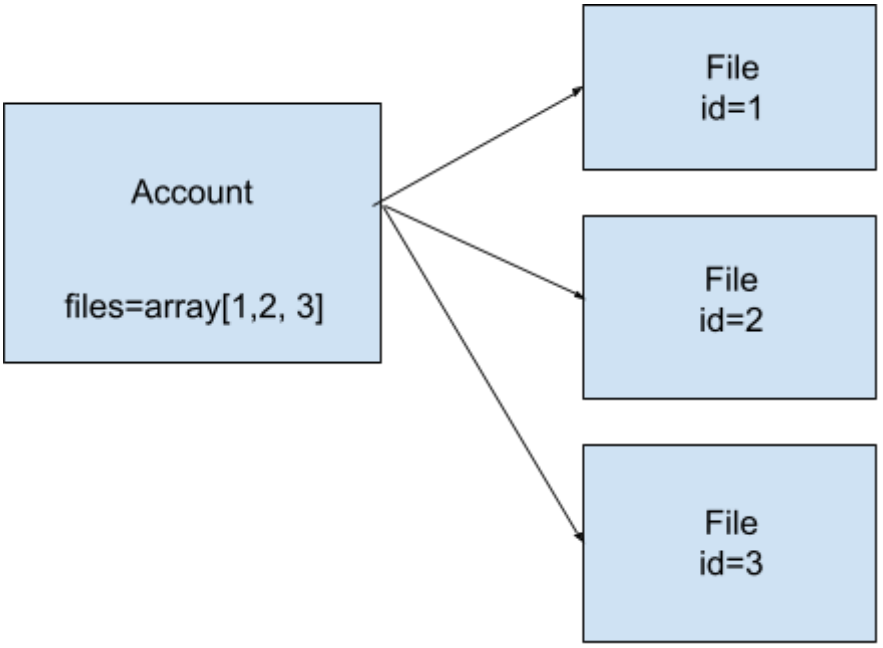

Figura 6: Exemplo de associação entre *Account* e *File*

A mesma lógica se aplica aos documentos ***Movement*** e ***Credit*** onde se guarda os identificadores únicos em vez do documento inteiro.

O script pode ser alterado para níveis empresariais, onde no caso de uma empresa ter que mudar de plataforma, onde se encontra a bases de dados, converter para json a informação, no mongoDB e usar o script, já numa versão onde aceitaria este ficheiro de json, para migrar os dados de forma automática.

5. Conclusão

Com o esforço conjunto conseguiu-se aprender, não só, uns com os outros, como funciona melhor a idealização e construção de uma base de dados mas também, o aperfeiçoamento da habilidade de se conseguir dividir um problema em pequenos pedaços para no final chegar ao objetivo final.

Aprendeu-se, não só a decidir entre duas formas de se arquitetar o modelo da base de dados, **modo de referência/normalizado - modo embeded/incorporado** mas também a criar mecanismos de conexão e gestão ao mongodb usando Python.

Sentiu-se, contudo, alguma dificuldade com a modelação do script Python e na concretização inicial da solução devido à não simplificação do pensamento, posteriormente resolvido. Continuar a pensar primeiro no problema e dividir em pequenas tarefas é um hábito a continuar a ser seguido.