Escola de Engenharia

Universidade do Minho

Implementação de uma base de dados para ficheiros GenBank

Introdução aos Algoritmos, à Programação e às Bases de Dados

Mestrado em Bioinformática

Joana Araújo PG49836, Mariana Silva PG45966, Tiago Silva PG49849

2022/2023

ÍNDICE

1. INTRODUÇÃO 2

1.1. CONTEXTUALIZAÇÃO E OBJETIVOS 2

1.2. ANÁLISE E JUSTIFICAÇÃO DA VIABILIDADE DO PROJETO 3

1.3. REQUISITOS E CARACTERIZAÇÃO DOS PERFIS DE UTILIZAÇÃO . 3

2. IMPLEMENTAÇÃO 3

2.1. CONEXÃO DO PYTHON À BASE DE DADOS SQL 4

2.2. IDENTIFICAÇÃO DAS ENTIDADES, ATRIBUTOS E RELACIONAMENTOS 4

2.3. MODELO CONCEPTUAL 7

2.4. MODELO LÓGICO 8

2.5. MODELO FÍSICO 9

2.6. POVOAÇÃO A PARTIR DAS SCRIPTS PYTHON 10

3. ANÁLISE CRÍTICA E PERSPETIVAS FUTURAS 11

REFERÊNCIAS BIBLIOGRÁFICAS 12

1. INTRODUÇÃO

1.1. CONTEXTUALIZAÇÃO E OBJETIVOS

O nosso mundo evoluiu para um ponto ótimo de avanços. Estes avanços têm ajudado no desenvolvimento de tecnologias, normas industriais, *gadgets* e dispositivos que necessitam de um sistema crucial de gestão, manipulação e produção de grandes quantidades de dados (Patil et al., 2017).

Quantidades maciças de dados, ou "Big data", são extremamente úteis no mundo moderno onde somos constantemente bombardeados com informações sobre atividades sociais, ciência, emprego, saúde e outros tópicos (Dash et al., 2019). Big data apresenta desafios significativos para muitas áreas científicas (Hey & Trefethen, 2002). É necessário criar novos métodos para organizar estes dados, de modo a podermos extrair informação útil a fim de satisfazer as nossas necessidades sociais atuais e futuras (Dash et al., 2019).

A informação recolhida a partir das muitas fontes de entrada e saída é utilizada para construir uma certa infraestrutura, mas também é vulnerável a danos se não for tratada adequadamente. Isto levou ao desenvolvimento de um método mais simples e mais eficaz para gerir dados enormes, onde a ideia do MySQL foi inicialmente estabelecida. A ferramenta MySQL oferece um modelo de dados mais flexível, maior escalabilidade, e também um desempenho superior ao mesmo tempo que incorpora várias características essenciais de bases de dados relacionais (Patil et al., 2017).

O GenBank é uma vasta biblioteca pública de sequências de proteínas e nucleótidos que é anotada com informação biológica e bibliográfica. O Centro Nacional de Informação em Biotecnologia (NCBI) é responsável pela criação e divulgação desta base de dados (Benson et al., 2003, 2010).

Esta base dados está dividida pela breve descrição da sequência, o nome científico e a taxonomia do organismo que forneceu a sequência, citações bibliográficas, e uma tabela de características que enumera regiões biologicamente significativas, tais como regiões de codificação e a proteína que traduzem, unidades de transcrição, regiões repetitivas, e os *sites* de mutação ou modificação (Benson et al., 2003).

Para este projeto, temos como objetivo a análise de um ou vários ficheiros Genbank, encontrados online, na biblioteca NCBI e, a partir de *Webscrapping*, inserir os dados obtidos numa base de dados SQL, seguindo o modelo relacional. Deste modo, pretende-se obter uma base

de dados que garanta a atualização das informações e mantenha a integridade dos dados, bem como facilitar o seu acesso e reduzir a redundância dos mesmos.

1.2. ANÁLISE E JUSTIFICAÇÃO DA VIABILIDADE DO PROJETO

Para permitir a viabilidade deste projeto, é necessário ter atenção a diversos fatores na sua implementação, sendo um dos objetivos tentar minimizar o número de ocorrências de erros na base de dados.

Além disso, teve-se em consideração que grandes volumes de dados inseridos podem provocar transtornos na sua acessibilidade e retorno de informação, no entanto, procuramos implementar uma base de dados onde, independentemente do volume dos materiais, o bom funcionamento da mesma não seja afetada. Eventualmente, num contexto real, ser-se-ia necessário desenhar uma base de dados mais robusta, com capacidade de armazenamento de maiores quantidades de dados.

1.3. REQUISITOS E CARACTERIZAÇÃO DOS PERFIS DE UTILIZAÇÃO

O objetivo visa permitir que o utilizador escolha que termo quer analisar, numa pesquisa máxima de 20 IDs, na base de dados do NCBI. É necessário validar que os termos introduzidos são inseridos de forma correta. Além disso, devemos certificar que não há mudança de locus para um mesmo ficheiro GenBank, pois este não muda quando há alterações na versão. Por outro lado, queremos que as Features, a Origin e o PubMed sejam atualizados, ou seja, modificados conforme a versão.

Por último, é essencial capacitar a inserção, atualização e deleção de dados de qualquer uma das entidades, atributos e respetivos dados, assim como evitar a redundância dos mesmos.

2. IMPLEMENTAÇÃO

Para realizar a implementação da base de dados é necessário definir um conjunto de entidades e respetivos atributos, assim como os relacionamentos entre as primeiras. Cada entidade é representada por uma tabela, em que as suas colunas correspondem aos atributos e as linhas aos dados de cada registo inserido.

Os relacionamentos podem apresentar diferentes tipos de cardinalidade, que correspondem ao número de tuplos que este pode conter, nomeadamente um para um (1:1), um para muitos (1:N) ou muitos para muitos (N:N).

2.1. CONEXÃO DO PYTHON À BASE DE DADOS SQL

O objetivo deste projeto passa por procurar páginas Genbank e, a partir de *Webscrapping*, selecionar a informação pretendida. Para isto, foram desenvolvidas scripts usando expressões regulares e package BioPython.

De seguida, de forma a inserir e obter estes dados na base de dados de trabalho, foi realizada a conexão, a partir do mysql.connector, preenchendo os campos *standard* "host", "user", "password" e "database". Desta forma, foram desenvolvidas scripts de modo a inserir e obter os dados na base de dados, respeitando os relacionamentos descritos abaixo.

2.2. IDENTIFICAÇÃO DAS ENTIDADES, ATRIBUTOS E RELACIONAMENTOS

A normalização da base de dados consiste no processo de organização de dados, de modo a melhorar o seu desempenho. Isto inclui a criação de tabelas e o estabelecimento das relações entre estas, de acordo com as regras concebidas de modo a proteger os dados e tornar a base de dados mais flexível, eliminando redundância e dependência inconsistente.

Foram definidas as entidades envolvidas no problema, respetivos atributos e relacionamentos.

A entidade "**GENBANK**" representa o formato do ficheiro com as informações do gene dos organismos em estudo. Esta entidade apresenta um conjunto de atributos, nomeadamente "**Id_GB**" que, por auto incremento, identifica o índice do ficheiro GenBank, "**GB_Id**" que identifica o ficheiro Genbank pelo seu nome, "**Definition**", que corresponde à descrição sucinta do gene, a "**Modification_date**" que diz qual foi a última vez que o campo LOCUS foi alterado no NCBI e, por último, a "**Version**" que é o identificador da versão mais atualizada. Foi definida como chave primária (PK) o atributo "**Id_GB**", uma vez que este é único para cada ficheiro.

A entidade "**LOCUS**" suporta as informações da última versão do ficheiro GenBank obtido. O "**Id_locus**", que identifica o índice do atributo por auto incremento, o "**Locus_Id**" que diz o nome identificador do locus, a "**Sequence_length**", que indica o número de pares de bases ou aminoácidos da sequência, "**Molecule_type**", que diz qual é o tipo de molécula presente, "**GenBank_division**" indica em qual divisão a sequência se integra, "**Accession**" é o identificador de um registo, "**Source**", que é o nome da espécie em estudo, seguido do seu nome comum e "**Organism**" que indica o nome da espécie. O atributo "**Id_locus**" foi definido como chave primária.

A entidade "**ORIGIN**" mostra a sequência nucleotídica ou aminoacídica do gene escolhido. A esta entidade foram dados seis atributos, dos quais "**Adenine**", "**Citosine**", "**Timine**" e "**Guanine**" indicam o número das bases nucleotídicas e aminoacídicas. O "**Length**" dá-nos o tamanho da sequência, que corresponde à soma de todas as bases identificadas

anteriormente e "**Sequence**" mostra a sequência respetiva. O "**Id_origin**" identifica o índice da *origin*, por auto incremento, e é definido como chave primária.

A entidade "**FEATURES**" possui informações sobre os genes, assim como as regiões com significado biológico. Possui 3 atributos, nomeadamente "gene", que é a região identificada como sendo de interesse biológico, "**Source_location**" que mostra a localização da sequência e **Id_F** que identifica a feature, por auto incremento na base de dados, sendo definida como chave primária.

A entidade "**CDS**" representa a região codificante que corresponde à sequência de aminoácidos numa proteína, que inclui codões de iniciação e de "**STOP**". Esta entidade compreende seis atributos como a "**Translation_seq**" que mostra a sequência traduzida, a "**Location**" que mostra a sua localização no genoma, "**Product**" que detém o significado biológico da proteína, "**Id_protein**" que referencia o identificador da proteína no NCBI e , por fim, o "**Id_CDS**" que identifica a ordem da sequência traduzida a analisar na base de dados. Este último atributo foi definido como chave primária.

A entidade "**PUBMED**" é uma ferramenta de literatura, que permite obter artigos relacionados com os ficheiros Genbank em causa. Os atributos desta entidade são "**Journal**", que designa o nome da revista onde o artigo está integrado, "**Title**" que mostra o nome do referido artigo, "**Abstract**" que corresponde a um breve resumo do artigo publicado, "**PM_Id**" que detém o identificador da publicação online do artigo no arquivo PubMed, e "**Id_PM**" que é auto incrementável, sendo considerada a chave primária desta entidade.

A entidade "**AUTHORS**" descreve o nome dos autores que participaram nos artigos em questão, e está relacionada com a entidade "**PUBMED**" numa relação de muitos para muitos (N:N), uma vez que vários autores podem ter várias publicações no PubMed, assim como o PubMed pode ter diversos autores inscritos na plataforma. Para esta entidade, contribuem os atributos "**Name**" que refere o nome do autor que participou no desenvolvimento do artigo e "**Id_Author**" que é um identificador do mesmo, por auto incremento e, por isso, foi considerado como chave primária desta entidade. Contribui ainda a chave estrangeira "**AFFILIATION_id_affiliation"** que corresponde a chave primária da entidade "**AFFILIATION**"

Por último, foi definida a entidade "**AFFILIATION**", referente à afiliação do autor. Esta entidade relaciona-se com "**AUTHORS**" num relacionamento de um (ou zero) para muitos (1:N), uma vez que uma afiliação pode ter vários autores, no entanto, um autor apenas pode ter uma única afiliação, ou até mesmo nenhuma. Os atributos desta entidade são "**Description**", que descreve a localização da afiliação, e "**Id_affiliation**" que identifica a ordem das afiliações. Este último atributo foi definido como chave primária devido ao auto incremento.

As tabelas **Tabela 1** e **Tabela 2** resumem, de forma visual, a caracterização das entidades, atributos e relacionamentos, e respetivas cardinalidades definidos para a criação da base de dados.

Tabela 1: Caracterização das entidades, atributos e relacionamentos correspondentes à base de dados criada aplicada à pesquisa de informações num ficheiro Genbank.

Entidade/ relacionamento	Nome	Tipo/domínio	Nulo	Auto incremento	Chave primária
GenBank	Id_GB	INT	N	S	S
	GB_id	VARCHAR(45)	N	N	N
	Modification_date	VARCHAR(12)	S	N	N
	Definition	VARCHAR(200)	S	N	N
	Version	VARCHAR(20)	S	N	N
	LOCUS_id_locus	INT	N	N	N
	ORIGIN_id_origin	INT	S	N	N
LOCUS	Id_Locus	INT	N	S	S
	Accession	VARCHAR(100)	N	N	N
	Sequence_length	INT	S	N	N
	Molecule_type	VARCHAR(20)	S	N	N
	GB_division	VARCHAR(10)	S	N	N
	Source	VARCHAR(200)	S	N	N
	Organism	VARCHAR(200)	S	N	N
	Locus_name	VARCHAR(45)	S	N	N
FEATURES	Id_F	INT	N	N	S
	Gene	VARCHAR(45)	S	N	N
	Source_location	VARCHAR(45)	S	N	N
ORIGIN	Id_origin	INT	N	S	S
	Length	INT	S	N	N
	Adenine	INT	S	N	N
	Citosine	INT	S	N	N
	Timine	INT	S	N	N
	Guanine	INT	S	N	N
	Sequence	LONGTEXT	S	N	N
CDS	Id_CDS	INT	S	N	S
	Translation_seq	LONGTEXT	N	N	N
	Location	VARCHAR(45)	N	S	N
	Product	VARCHAR(45)	N	S	N
	Protein_ID	VARCHAR(45)	N	N	N
PubMed	Id_PM	INT	N	S	S
	PM_ID	INT	N	N	N
	Jornal	VARCHAR(100)	S	N	N
	Title	VARCHAR(500)	S	N	N
	Abstract	LONGTEXT	S	N	N
Authors	Id_Author	INT	N	S	S
	Name	VARCHAR(200)	S	N	N
	AFFILIATION_id_affiliation	INT	S	N	N
Affiliation	Id_affiliation	INT	N	S	S
	Description	LONGTEXT	S	N	N

Tabela 2: Cardinalidade das entidades e relacionamentos correspondentes à base de dados criada aplicada à pesquisa de informações num ficheiro Genbank.

Cardinalidade	Entidade	Relacionamento	Entidade	Cardinalidade
1…1	GenBank	Contains	LOCUS	N…1
N…1	GenBank	GB_FT_relationship	FEATURES	N…1
1…1	GenBank	Contains	ORIGIN	N…1
N…1	GenBank	PM_GB_relationship	PubMed	N…1
N…1	GenBank	CDS_GB_relationship	CDS	N…1
N…1	PubMed	PM_AUTHORS_relationship	Authors	N…1
1…1	Authors	Affiliated	Affiliation	N…1

2.3. MODELO CONCEPTUAL

Após definir a organização dos dados e estabelecimento de entidades, atributos e relacionamentos, realizou-se o modelo conceptual, recorrendo ao software TerraER, representado na **Figura 1**.

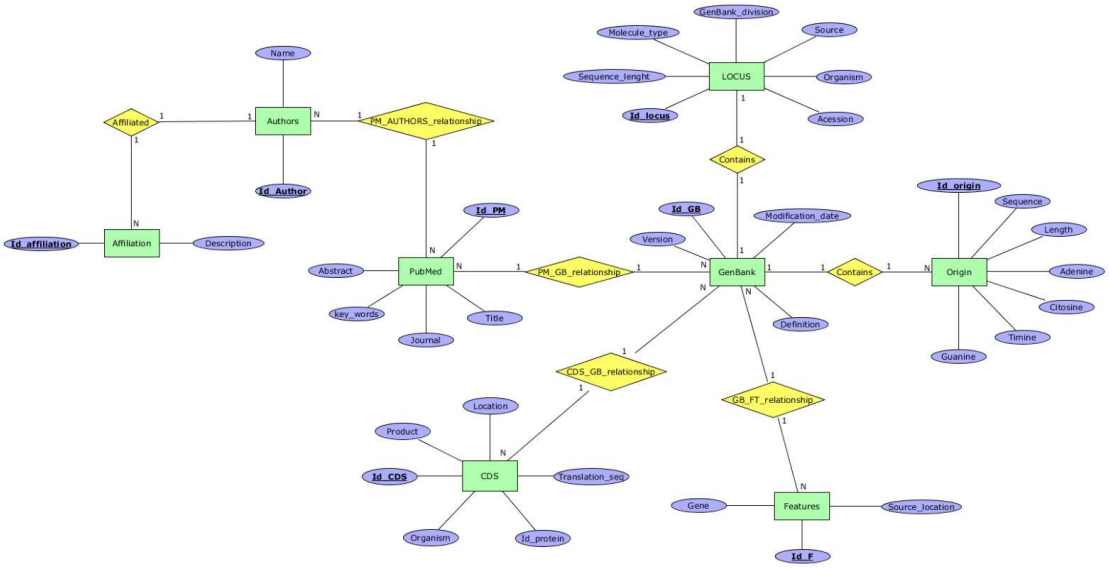

Figura 1. Modelo conceptual de uma base de dados aplicada à pesquisa de informações num ficheiro Genbank, realizado no software TerraER.

2.4. MODELO LÓGICO

A fase seguinte foi transformar o modelo conceptual anteriormente desenhado, na **Figura 1**, num modelo lógico, representado na **Figura 2**, em que se teve em consideração a cardinalidade dos relacionamentos. Para tal, recorreu-se ao MySQL Workbench.

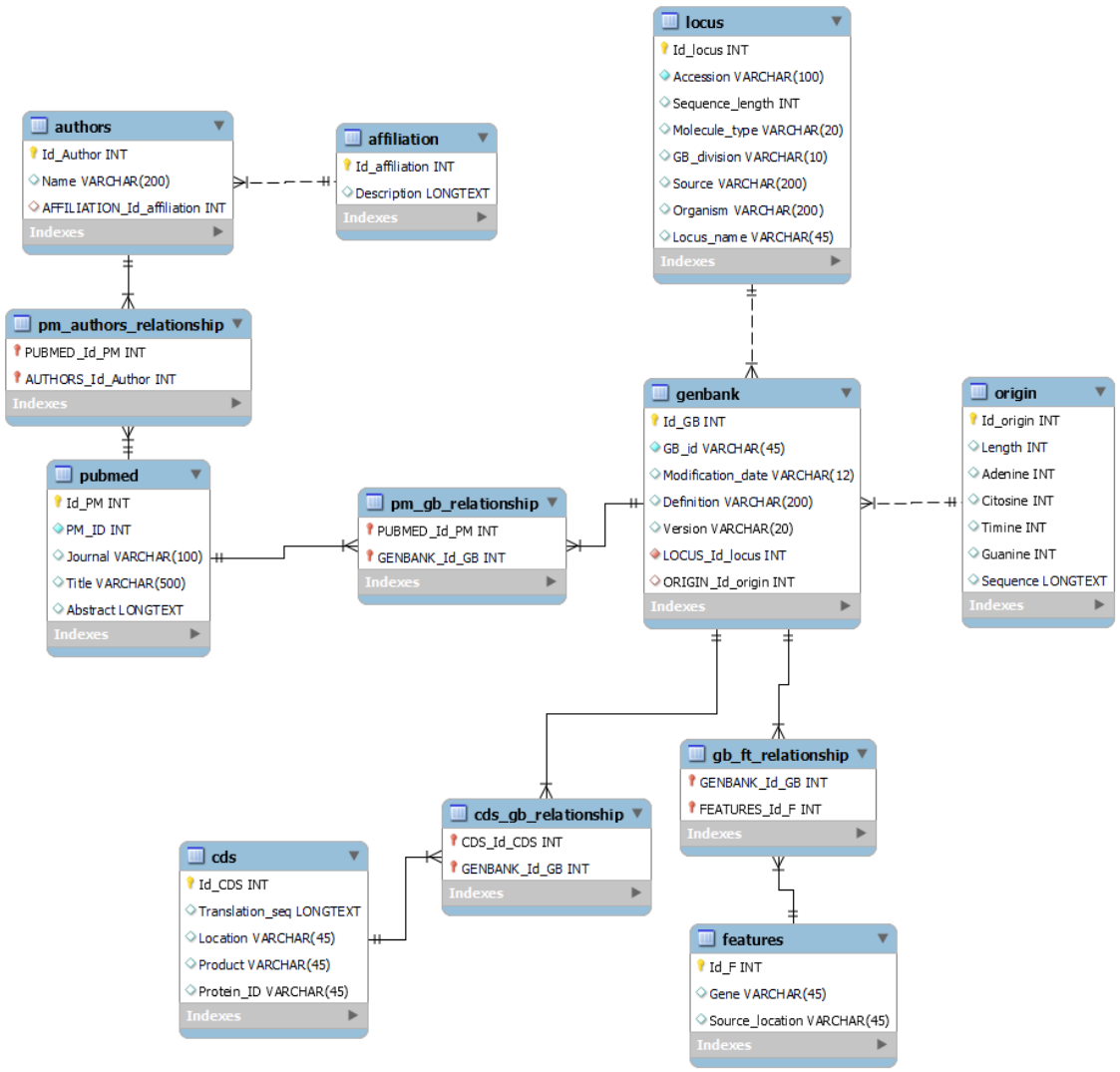

Figura 2. Modelo lógico da base de dados aplicada à pesquisa de informações num ficheiro Genbank, realizado no MySQL Workbench.

2.5. MODELO FÍSICO

O modelo físico da base de dados é obtido recorrendo ao *Forward Engineer*, criando uma script que possibilita a criação da base de dados e respetivos elementos. Na **Figura 3** é possível ler que a script cria a tabela '**LOCUS**', se esta não existir, e os respetivos atributos e sua caracterização.

```
-- -----------------------------------------------------
-- Table `teste`.`locus`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `teste`.`locus` (
  `Id_locus` INT NOT NULL AUTO_INCREMENT,
  `Accession` VARCHAR(100) NOT NULL,
  `Sequence_length` INT NULL DEFAULT NULL,
  `Molecule_type` VARCHAR(20) NULL DEFAULT NULL,
  `GB_division` VARCHAR(10) NULL DEFAULT NULL,
  `Source` VARCHAR(200) NULL DEFAULT NULL,
  `Organism` VARCHAR(200) NULL DEFAULT NULL,
  `Locus_name` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`Id_locus`))
ENGINE = InnoDB
AUTO_INCREMENT = 10
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `teste`.`origin`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `teste`.`origin` (
  `Id_origin` INT NOT NULL AUTO_INCREMENT,
  `Length` INT NULL DEFAULT NULL,
  `Adenine` INT NULL DEFAULT NULL,
  `Citosine` INT NULL DEFAULT NULL,
  `Timine` INT NULL DEFAULT NULL,
```

Figura 3. Excerto de script que originou o modelo físico da base de dados aplicada à pesquisa de informações num ficheiro Genbank, realizado no MySQL Workbench.

2.6. POVOAÇÃO A PARTIR DAS SCRIPTS PYTHON

Com as scripts desenvolvidas inicialmente, a partir do Python, foram inseridos os dados de forma automatizada na base de dados (**Figura 4**), respeitando todos os relacionamentos entre entidades.

Para evitar a duplicação da entrada de dados, fazemos um SELECT ao ID da tabela, passando como condição o identificador único de cada entrada. Como exemplo, para inserir informação na tabela LOCUS, sabendo que o Locus_name é único, usamo-lo para verificar se esta informação já foi inserida da BD.

Para obter resultado do SELECT, usamos o *fetchall*, que vai buscar toda a informação do SELECT. Esta função retorna uma lista de tuplos. Por esta razão, somos forçados a fazer um ciclo "for" para iterar esta lista. No fim do ciclo, vamos ter o último ID para o qual o Locus_name seja igual ao valor que foi inserido no SELECT.

De seguida, se a variável id_loc estiver preenchida, significa que a informação já foi préviamente inserida na base de dados. Se for None, significa que a temos que inserir.

De maneira a evitar a múltipla chamada de INSERTS à base de dados, a informação que tiver que ser inserida será adicionada à sql_loc (string que irá fazer o INSERT), após ter sido percorrida toda a informação que queremos adicionar, neste caso, a nossa loc_name_list.

A variável sql_loc é a string a executar no SQL para inserir os dados na BD, se esta tiver valores. Como tal, verificamos se esta variável tem um valor diferente do valor com que foi inicializada. Caso positivo, iremos enviar esta string para ser executada no MySQL.

Para evitar problemas de syntax no SQL, usamos a função "endswith(',')" para verificar se o último elemento é uma ",". Se afirmativo, esta é removida.

Neste momento, é possível relacionar tabelas e procurar informações de forma conveniente para o utilizador.

```
Cursor.execute(TableName)
sql_loc = "INSERT INTO LOCUS (Locus_name, Sequence_length, Molecule_type, GB_division, Source, Organism, Accession) VALUES"
loc_name_list = list(dict.fromkeys(loc_name_list))
for index in range(len(loc_name_list)):

    select_ids_loc = f"SELECT Id_Locus FROM LOCUS WHERE Locus_name = \"{loc_name_list[index]}\""
    Cursor.execute(select_ids_loc)
    myresult_locus = Cursor.fetchall()
    id_loc = None
    for res_loc, in myresult_locus:
        id_loc = res_loc
    if id_loc is None:
        if index == len(loc_name_list)-1:
            sql_loc += f" (\"{loc_name_list[index]}\", \"{length[index]}\", \"{mol_type[index]}\", \"{gb_div[index]}\", \"{source_list[index]}\"
        else:
            sql_loc += f" (\"{loc_name_list[index]}\", \"{length[index]}\", \"{mol_type[index]}\", \"{gb_div[index]}\", \"{source_list[index]}\"
print(sql_loc)
if sql_loc != "INSERT INTO LOCUS (Locus_name, Sequence_length, Molecule_type, GB_division, Source, Organism, Accession) VALUES":
    if sql_loc.endswith(","):
        sql_loc = sql_loc[:-1]
    Cursor.execute(sql_loc)

DataBase.commit()

print(Cursor.rowcount, "record inserted.")
```

Figura 4. Excerto de script Python que originou permitiu a povoação da base de dados aplicada à pesquisa de informações num ficheiro Genbank.

3. ANÁLISE CRÍTICA E PERSPETIVAS FUTURAS

Ao longo do desenvolvimento da base de dados fomos confrontados com dificuldades principalmente no estabelecimento de atributos e relacionamentos.

Por outro lado, ocorreram problemas no acesso ao NCBI em alguns IDs, provavelmente porque o site terá sido alterado, sendo algumas sequências não retornadas. Este ponto poderia ser explorado para uma melhor coerência entre a base de dados criada e a base de dados do NCBI.

Além disso, pode-se explorar com maior profundidade a entidade das afiliações dos autores, uma vez que estas, ao longo do tempo, podem mudar se o autor alterar o seu centro de investigação e, por isso, deveria ser possível atualizar esta informação, adicionando uma *deletion table*, que atualiza o *status* atual.

REFERÊNCIAS BIBLIOGRÁFICAS

Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., & Sayers, E. W. (2010). GenBank. *Nucleic Acids Research*, *39*(suppl_1), D32–D37.

Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., & Wheeler, D. L. (2003). GenBank. *Nucleic Acids Research*, *31*(1), 23.

Dash, S., Shakyawar, S. K., Sharma, M., & Kaushik, S. (2019). Big data in healthcare: management, analysis and future prospects. *Journal of Big Data*, *6*(1), 1–25.

Hey, T., & Trefethen, A. E. (2002). The UK e-science core programme and the grid. *International Conference on Computational Science*, 3–21.

Patil, M. M., Hanni, A., Tejeshwar, C. H., & Patil, P. (2017). A qualitative analysis of the performance of MongoDB vs MySQL database based on insertion and retriewal operations using a web/android application to explore load balancing — Sharding in MongoDB and its advantages. *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 325–330. https://doi.org/10.1109/I-SMAC.2017.8058365