



Universidade do Minho

Escola de Engenharia

Mestrado em Bioinformática

**Base de Dados para o armazenamento e
informação contida em ficheiro de formato
Genbank**

Autores:

- José Lemos
- Paulo Seixal
- Rúben Fernandes

Introdução aos Algoritmos, à Programação e às
Bases de Dados

Janeiro de 2023

Índice

| | |
|---|----|
| Introdução | 3 |
| Objetivos e Motivação | 3 |
| Contextualização | 3 |
| Entidades e atributos | 4 |
| Modelo conceptual | 5 |
| Modelo lógico e modelo físico | 5 |
| <i>Functions, procedures e triggers</i> | 7 |
| Povoamento das tabelas | 7 |
| Discussão e análise | 10 |
| Bibliografia | 10 |

Introdução

Objetivos e Motivação

Foi proposto como elemento de avaliação a elaboração de um trabalho que inclui a resolução de um problema existente nos ficheiros GenBank. De forma a solucionar este problema, foi proposta a criação de uma base de dados para transferir, organizar e armazenar informações referentes a ficheiros GenBank. Também foi pretendido a possibilidade de procurar e armazenar artigos relacionados com as sequências em causa.

Os ficheiros em formato GenBank armazenam muita informação associada a várias sequencias biológicas de vários organismos sendo por vezes difícil a procura, organização e comparação da informação. Este modelo poderá ser direcionado para locais onde manipulam e armazenam este tipo de ficheiros, de forma a facilitar a procura e a transferência da informação por parte de utilizadores (bioinformáticos, investigadores, académicos, etc.)(1,2)

Contextualização

Para a inicialização deste trabalho, é necessário entender e descrever toda a informação e características que se podem retirar de um ficheiro GenBank :

- **LOCUS** – Armazena informação básica do registo.
 - Nome do Locus (ex: SCU49845)
 - N.º pares de bases
 - Tipo de sequência
 - Direção da sequência
 - Divisão do GenBank
 - Data de submissão
- **DEFINITION** – Breve descrição da sequência onde menciona o organismo, nomes dos genes/proteínas.
- **ACCESSION** – Identificador único para o registo da sequência
- **VERSION** – Representado por um complexo de duas partes onde o primário é composto pelo accession e um número de versão representativo de uma alteração da sequência (ex.: U49845.1), a segunda chave é designada pela palavra-chave GI seguida de um número que é alterado quando é efetuada uma alteração na sequência(1,2).
- **KEYWORDS** – Palavra ou frase que descreve a sequência. Registo sem keywords são representados por um ponto final.
- **SOURCE** - Indica o nome mais comum usado para o organismo de onde a sequência foi retirada.
 - ORGANISM – Indica o nome científico e a sua taxonomia.
- **REFERENCE** – Campo onde se referencia citações e publicações de origem da informação presente no ficheiro. Dentre deste campo também se pode consultar:
 - AUTHORS
 - TITLE
 - JOURNAL
 - PUBMED
- **FEATURES** – Contém informação de interesse para os investigadores, localização da porção de uma sequência e informação relativa a essa porção.
 - source
 - CDS

- Gene
- **ORIGIN** – Contém a sequência em estudo.

Com a informação retirada das características do ficheiro anteriormente mencionadas, é possível então perceber as entidades e atributos que podem constituir a base de dados. Neste trabalho iremos construir um modelo conceptual utilizando a ferramenta TerraER3.14, aplicação desse modelo conceptual para um modelo lógico através da criação de um diagrama no *MySQL Workbench*, passando este para um modelo físico através do *forward engineering*. A aplicação de *functions, triggers and procedures* apenas serão usados em caso de necessidade. Por fim, será utilizado módulos Python para o povoamento das tabeladas da base de dados(3).

Entidades e atributos

No que se refere a entidades, iniciou-se pela entidade-relacionamento “**LOCUS**” que representa o cabeçalho do ficheiro GenBank (fundamental para a identificação do registo) associados a esta entidade temos como atributos a sua chave primária (**accession**), identificativo único retirado do campo “*VERSION*” do registo, o nome do locus (**locus_name**), é usado para distinguir entradas de genes com sequências semelhantes, o tipo da sequência (**sequence_type**) em que pode ser, por exemplo, linear, circular etc., o tipo de molécula (**molecule**) que foi sequenciada podendo este campo ser DNA, RNA, cDNA, tRNA, etc. Temos também a data de introdução no GenBank (**genbank_date**), que nos diz quando é que foi submetido um ficheiro GenBank e a divisão GenBank (**genbank_division**) que é o grupo denominado apenas por 3 letras a qual a entrada pertence(3).

Ligadas diretamente ao “**LOCUS**” vamos ter 9 entidades:

- “**DEFINITION**”, que vai ter o seu identificador como chave primária (**def_ID**), o texto com a definição correspondente (**definition**).
- “**SEQUENCES**” onde temos associado a sequência (**seq**) e o tamanho da sequência (**seq_lenght**).
- Temos as features do GenBank com as entidades “**FEAT_GENE**” com uma chave primária (**feat_gene_ID**), que nos disponibiliza, o nome do gene (**name**) e a sua localização (**location**). A entidade “**FEAT_CDS**” onde armazena as sequências codificantes, com uma chave primária (**feat_cds_id**) a proteína resultante (**product**), o identificador da proteína (**protein_id**) a localização (**cds_location**) o codão START (**codon_start**) e a sequência traduzida (**translation**). A “**FEAT_SOURCE**” contem uma chave primária (**feat_source_id**), a sua localização (**location**), o organismo (**organism**) e a referência taxonómica (**db_xref**).
- Em “**SOURCE**” (nome do organismo) temos uma chave primária (**source_ID**) e o nome científico (**scientific_name**).
- Em “**TAXONOMY**” temos uma chave primária (**taxonomy_ID**) e a denominação (**taxonomy**).
- “**LOCUS_KEYWORDS**” vamos ter também uma chave primária (**locus_keyword_ID**), nome da keyword (**keyword_name**) e a ordem da keyword no ficheiro (**key_order**).
- “**REFERENCE**” é onde vamos ter armazenadas as referências, com uma chave primária (**reference_id**), a localização (**ref_location**), o título (**title**), o seu registo (**journal**) e o id do artigo no pubmed (**pubmed_id**). Temos também a entidade relacionamento “**PUBMED**” que vai permitir conectar a entidade “**REFERENCE**” com a entidade “**AUTHORS**”, com uma chave primária (**pubmed_id**), um título (**title**), o número do

artigo (**issue**), o número de páginas (**pages**), a fonte (**source**), e a data de publicação (**date**). Por último, temos a entidade “**AUTHORS**”, onde armazenamos os nomes dos autores, com uma chave primária (**author_ID**) e o nome do autor (**author_name**).

Modelo conceptual

A partir das entidades e atributos definidos no ponto anterior, e recorrendo ao software TerraER, procedeu-se à realização do esquema conceptual para a base de dados. Este esquema (Figura 1) representa as entidades, os atributos e as respectivas relações entre entidades indicando ainda a cardinalidade de cada relacionamento.

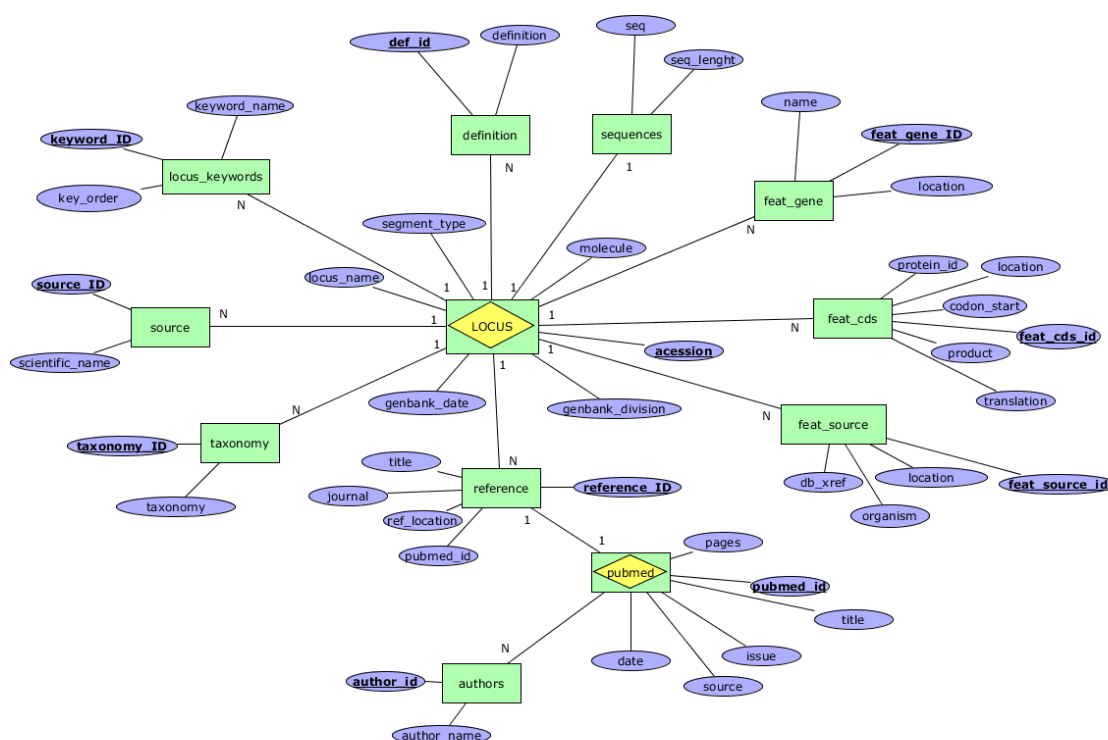


Figura 1 - Modelo conceptual da base de dados

Modelo lógico e modelo físico

Os modelos lógicos e físicos da base de dados foram criados no *MySQL Workbench* com base no modelo conceptual apresentado no ponto anterior do presente relatório.

Começando pelo modelo lógico, cada entidade do modelo conceptual corresponde a uma tabela no modelo lógico, nas quais as respetivas colunas correspondem aos atributos. Nesta fase de criação de tabelas para o modelo lógico, são estabelecidas algumas restrições a atribuir a cada atributo, como por exemplo: não nulos, chaves primárias, chaves secundárias, etc., bem como o tipo de dados a inserir.

Ao longo da criação dos modelos, a criação de chaves primárias e secundárias é de extrema importância, uma vez que irão permitir identificar inequivocamente, em cada tabela, uma determinada entrada da base de dados, para além de estabelecer relações entre cada entidade.

Em alguns casos, a chave primária pode constituir uma chave estrangeira numa outra tabela, estabelecendo assim a relação entre as duas entidades.

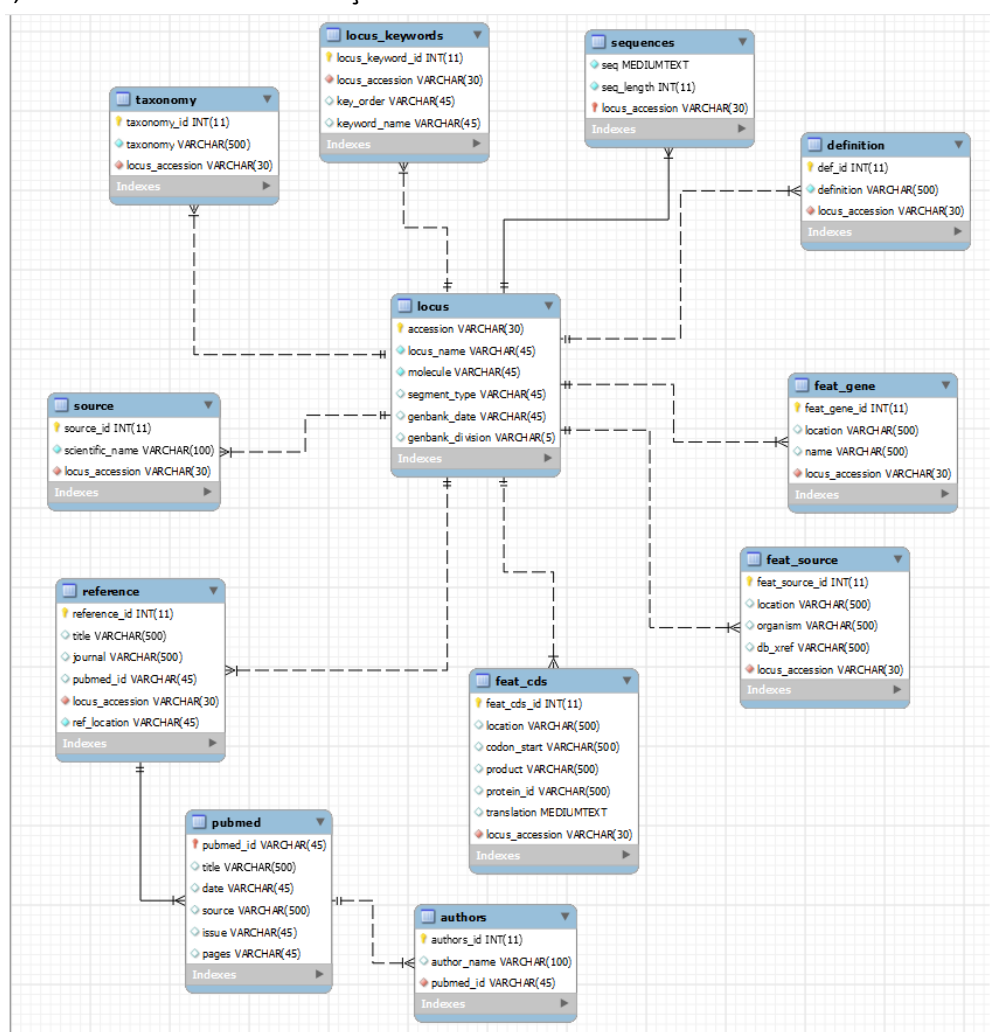


Figura 2 - Imagem representativa do modelo lógico criado a partir do programa MySQL WorkBench

No modelo lógico da base de dados é possível serem observados vários casos onde esta situação se verifica. A título de exemplo, podemos utilizar o caso da tabela ‘locus’, onde a sua chave primária ‘accession’ servirá de chave estrangeira na tabela ‘definition’. Desta forma ao introduzirmos dados na tabela ‘definition’ estaremos a descrever como estes se relacionam com os dados da tabela ‘locus’. Por exemplo, a definição ‘Human immunodeficiency virus type 1’ (tabela ‘definition’) está associada ao um locus com o accession ‘L42022.1’, que é uma molécula DNA e foi inserido no genbank a 24 de março de 1997 (tabela ‘locus’). Este é apenas um exemplo entre muitos que podem ser observados por todo o modelo, sendo bastante comum.

Existe ainda uma outra particularidade, neste caso, relacionada com a tabela ‘pubmed’, que considerou-se ser uma entidade-relacionamento de forma a receber a informação obtida na tabela reference, sendo populada e correlacionada com a tabela ‘authors’. Alguns ficheiros genbank possuem várias referências associadas, no entanto, nem todas as referências possuem um id do pubmed. Assim sendo, a tabela ‘pubmed’ está dependente da presença de pelo menos uma referência com o ID do pubmed (na tabela ‘reference’). Só a partir deste ID é que se torna possível a introdução de dados na tabela ‘pubmed’ relacionados com o artigo em questão.

Relativamente ao modelo físico, o mesmo foi gerado a partir do modelo lógico. Ou seja, uma vez criado o modelo lógico no MySQL Workbench, é possível aplicar o *forward engineer*

que nos permite obter o modelo físico a partir do modelo lógico em causa, tal como permite criar a base de dados propriamente dita.

```
1  -- MySQL Workbench Forward Engineering
2
3  SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
4  SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
5  SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
6
7  --
8  -- Schema mydb
9  --
10 --
11 -- Schema grupo_PRJ
12 --
13 --
14 --
15 -- Schema grupo_PRJ
16 --
17 CREATE SCHEMA IF NOT EXISTS `grupo_PRJ` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_bin ;
18 USE `grupo_PRJ` ;
19
20 --
21 -- Table `grupo_PRJ`.`locus`
22 --
23 CREATE TABLE IF NOT EXISTS `grupo_PRJ`.`locus` (
24   `accession` VARCHAR(30) NOT NULL,
25   `locus_name` VARCHAR(45) NOT NULL,
26   `molecule` VARCHAR(45) NOT NULL,
27   `segment_type` VARCHAR(45) NULL DEFAULT NULL,
28   `genbank_date` VARCHAR(45) NULL DEFAULT NULL,
29   `genbank_division` VARCHAR(2) NULL DEFAULT NULL,
30   PRIMARY KEY (`accession`),
31   UNIQUE INDEX `accession_UNIQUE` (`accession` ASC) )
32 ENGINE = InnoDB
33 DEFAULT CHARACTER SET = utf8mb4
34 COLLATE = utf8mb4_bin;
35
36
```

Figura 3 - Execerto inicial do modelo físico criado a partir do forward engineering do modelo lógico

Functions, procedures e triggers

Na elaboração deste trabalho não se considerou extremamente necessário a implementação de *functions*, *procedures* e *triggers*. No entanto, decidiu-se criar cinco funções com o objetivo de efetuar a contagem de nucleótidos (A, T, C, G, U) encontrados como elementos da sequência representada pelo atributo **seq**, presente na tabela **sequences**.

```
1  CREATE DEFINER=`root`@`localhost` FUNCTION `count_A`(seq MEDIUMTEXT) RETURNS int
2    DETERMINISTIC
3  BEGIN
4    DECLARE res INT;
5    SET res = (length(seq) - length(replace(seq, 'A', '')));
6    RETURN(res) ;
7  END
```

Figura 4 - Exemplo de uma das cinco funções criadas. Nesta encontra-se representada a contagem das adeninas.

Povoamento das tabelas

Para o povoamento das tabelas, foi-nos proposta a sua realização com recurso a scripts em python. No entanto, esta linguagem de programação, por si só, não é capaz de se conectar a uma base de dados em SQL. De modo a contornar este obstáculo, foi usado o módulo

mysql.connector que permite fazer a conexão entre o script e uma base de dados em concreto, ao especificar o host, o utilizador, a password do servidor e o nome da base de dados a ser povoada.

Relativamente ao povoamento propriamente dito, o mesmo foi realizado com o auxílio de duas estratégias diferentes: o uso de expressões regulares e o package Biopython.

As expressões regulares são padrões utilizados para seleccionar combinações de caracteres, como letras, dígitos, palavras, padrões de caracteres, entre outros. Este método torna-se especialmente útil na análise de ficheiros genbank uma vez que podemos procurar padrões que se repetem em cada ficheiro, podendo assim realizar uma busca padronizada por informação específica no ficheiro, independentemente do organismo que queremos analisar.

Para além das expressões regulares, foi também usado o package Biopython que possui uma série de funcionalidades aplicadas à Bioinformática. Para este trabalho, o Biopython tornou-se especialmente útil uma vez que permite ler e retirar as várias informações armazenadas num ficheiro genbank com recurso ao módulo SeqIO.

Relativamente aos scripts propriamente ditos, foram criados dois scripts ligeiramente diferentes, cada um com objetivos diferentes: um deles será utilizado para retirar informação de um ficheiro genbank armazenado na própria máquina, e o segundo será utilizado para o caso de retirar informações de um ficheiro genbank diretamente da web, sem ser necessário o download do próprio ficheiro, indicando apenas o email e o *accession number* ou a *version* do ficheiro a analisar.

Para retirar as informações de um ficheiro genbank diretamente da web com o package Biopython, é necessária a utilização do módulo Entrez que permite criar um *handle* com as informações do ficheiro na web a serem lidas pelo módulo SeqIO.

A partir deste ponto, os dois scripts foram desenvolvidos de forma semelhante, sendo que para o script de leitura de ficheiros armazenados foram utilizadas expressões regulares e Biopython, e para o script de leitura de ficheiros da web foi apenas utilizado o Biopython.

Durante o desenvolvimento dos scripts, foi necessário encontrar soluções para algumas especificidades encontradas para o povoamento de algumas tabelas. A título de exemplo, iremos abordar as tabelas **'reference'** e **'pubmed'**.

Relativamente à tabela **'reference'**, notou-se que nem todos os ficheiros apresentavam referências com todos os atributos que queríamos estudar. Por exemplo, algumas referências poderiam não ter associado um ID do pubmed e/ou a localização da referência. Assim sendo, no desenvolvimento do código, foi necessário ter esta questão em conta com várias condições **'if'**, de modo a garantir que todos os dados seriam introduzidos de maneira correta.

```
14
15 for i in zip(title, journal, pubmed_id, ref_location):
16     if i[2] == '' and i[3] == '':
17         sql_references = f"INSERT INTO reference (title, journal, locus_accession) VALUES ('{i[0]}', '{i[1]}', '{locus_accession}')"
18         print(sql_references)
19         print('Esta referência não possui ID do pubmed, nem descreve a sua localização')
20         print()
21         Cursor.execute(sql_references)
22         DataBase.commit()
23
24     elif i[2] == '' and i[3] != '':
25         sql_references = f"INSERT INTO reference (title, journal, locus_accession, ref_location) VALUES ('{i[0]}', '{i[1]}', '{locus_accession}', '{i[3]}')"
26         print(sql_references)
27         print('Esta referência não possui ID do pubmed')
28         print()
29         Cursor.execute(sql_references)
30         DataBase.commit()
31
```

Figura 5 - Excerto de um exemplo retirado do script para o povoamento de tabelas neste caso da tabela **reference**.

Relativamente à tabela **'pubmed'**, para o seu povoamento optou-se por retirar a informação de cada artigo diretamente da web. Para isso, recorreu-se novamente ao módulo

Entrez que, a partir do ID do pubmed de uma referência em específico, permite criar e ler um *handle* com informações do artigo associado.

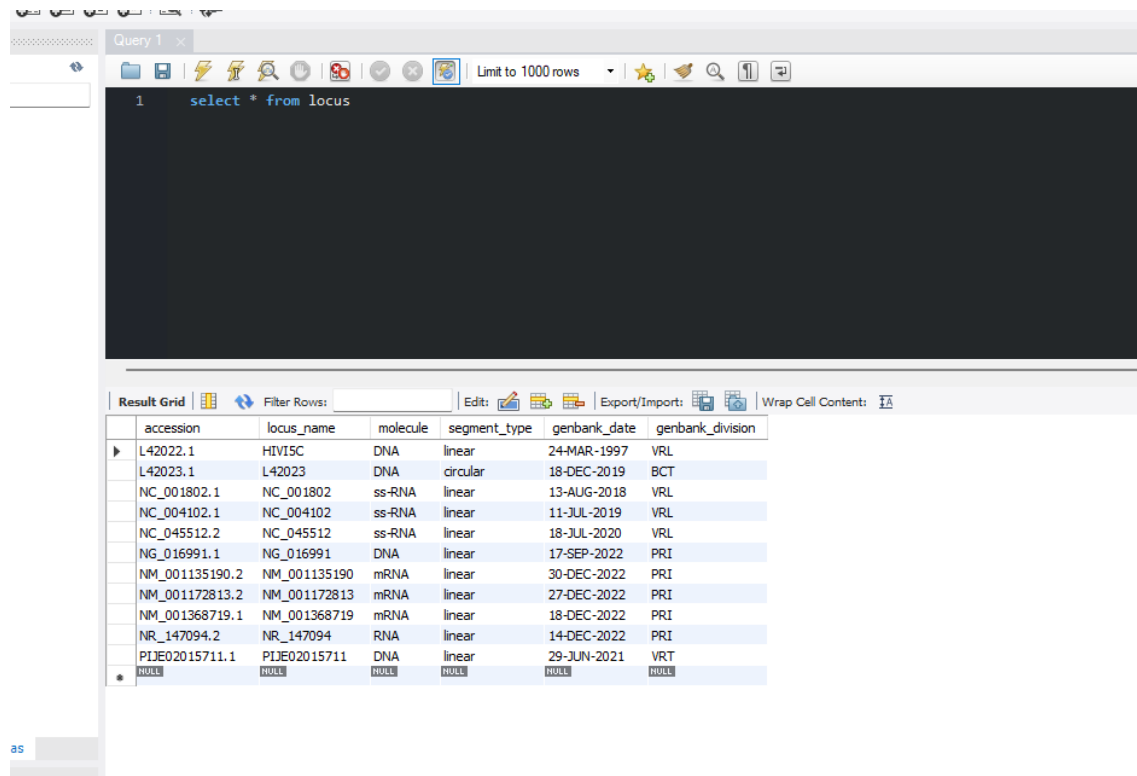
```

1 for id in pubmed_id:
2     if id != '':
3         pubmed_handle = Entrez.esummary(db="pubmed", id=id, retmode="xml")
4         pubmed_record = Entrez.read(pubmed_handle)
5         pubmed_handle.close()

```

Figura 6 - Exemplo de como retirar informações de um artigo a partir da web com o auxílio do módulo "Entrez".

Neste caso, o povoamento foi efetuado a partir da web através do script em cima mencionado com o nome '*povoamento_tabelas_from_web.ipynb*'. Na tabela 4, pode ser consultado uma tabela com o povoamento efetuado.



The screenshot shows a Jupyter Notebook interface. At the top, there's a toolbar with various icons. Below it, a text area contains the SQL query: `select * from locus`. Below the query, a 'Result Grid' is displayed, showing a table with 6 columns: *accession*, *locus_name*, *molecule*, *segment_type*, *genbank_date*, and *genbank_division*. The table contains 13 rows of data, including entries like L42022.1, L42023.1, NC_001802.1, etc. The last row shows 'NULL' values for all columns.

| accession | locus_name | molecule | segment_type | genbank_date | genbank_division |
|----------------|--------------|----------|--------------|--------------|------------------|
| L42022.1 | HIVISC | DNA | linear | 24-MAR-1997 | VRL |
| L42023.1 | L42023 | DNA | circular | 18-DEC-2019 | BCT |
| NC_001802.1 | NC_001802 | ss-RNA | linear | 13-AUG-2018 | VRL |
| NC_004102.1 | NC_004102 | ss-RNA | linear | 11-JUL-2019 | VRL |
| NC_045512.2 | NC_045512 | ss-RNA | linear | 18-JUL-2020 | VRL |
| NG_016991.1 | NG_016991 | DNA | linear | 17-SEP-2022 | PRI |
| NM_001135190.2 | NM_001135190 | mRNA | linear | 30-DEC-2022 | PRI |
| NM_001172813.2 | NM_001172813 | mRNA | linear | 27-DEC-2022 | PRI |
| NM_001368719.1 | NM_001368719 | mRNA | linear | 18-DEC-2022 | PRI |
| NR_147094.2 | NR_147094 | RNA | linear | 14-DEC-2022 | PRI |
| PIJE02015711.1 | PIJE02015711 | DNA | linear | 29-JUN-2021 | VRT |
| NULL | NULL | NULL | NULL | NULL | NULL |

Figura 7 - Demonstração da invocação da tabela *locus*, representando o povoamento efetuado.

Discussão e análise

O trabalho desenvolvido foi considerado um sucesso pois foi possível atingir todos os objetivos propostos, desde a criação do modelo conceptual à implementação do modelo lógico e físico, bem como a utilização de módulos Python para o povoamento das tabelas da base de dados criada. Considera-se que esta base de dados está funcional, com o armazenamento de toda a informação objetivada, apresentando-se como uma solução ao problema proposto.

Durante a realização deste trabalho sentiram-se dificuldades em correlacionar os modelos criados com os scripts de povoamento, tendo sido necessário proceder a várias iterações dos modelos para um melhor relacionamento da informação. Outra dificuldade sentida foi a idealização e aplicação de *procedures*, *functions* e *triggers*, pois, certamente existirão outras opções de implementação destas ferramentas na nossa base de dados, para além das criadas para este trabalho, de forma a melhorar e otimizar a sua utilização.

No decorrer da criação dos scripts foi necessário proceder a várias iterações do código para que este se correlacione com o maior número de ficheiros. Um exemplo deste aspeto foi a forma como se contornou o problema dos apóstrofes presentes em alguns títulos dos artigos das referências, sendo que, neste caso, foi apenas necessário trocar o símbolo ['] pelos [""], como pode ser observado no código da tabela “references”, por exemplo.

Numa análise feita durante a testagem do povoamento das tabelas foi possível verificar que existem ficheiros GenBank disponibilizados na internet cuja informação encontra-se por defeito omitida devido ao seu elevado tamanho, estando assim inacessível para a importação para a base de dados.

Bibliografia

1. GenBank Sample Record [Internet]. [cited 2023 Jan 18]. Available from: <https://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html>
2. Sayers EW, Cavanaugh M, Clark K, Pruitt KD, Schoch CL, Sherry ST, et al. GenBank. Nucleic Acids Res [Internet]. 2021 Jan 8 [cited 2023 Jan 18];49(D1):D92–6. Available from: <https://academic.oup.com/nar/article/49/D1/D92/5983623>
3. Mann N. Developing a database for Genbank information. Developing a database for Genbank information. 2004 [cited 2023 Jan 18]; Available from: <https://doi.org/10.18297/etd/902>