



Universidade do Minho
Escola de Engenharia

Relatório do Trabalho Prático

Introdução aos Algoritmos, à Programação e às Bases de Dados

Professores: António Abelha e Jorge Rocha

Mestrado em Bioinformática

Joana Oliveira Gonçalves, pg49835

Gonçalo Apolinário Cardoso, pg49834

Ano letivo: 2022/2023

1. Introdução

1.1- Definição da contextualização, motivação e objetivos, perante a sua escolha para o novo SBD relacional a implementar

Com o inúmero aumento da quantidade de dados biológicos disponíveis tornou-se evidente a urgente necessidade de armazenamento e agrupamento desses dados biológicos visando uma padronização desses mesmos dados.

O GenBank é um banco de dados de Nucleotídeos do NCBI, localizado no National Institutes of Health (NIH), armazenando informação sobre sequências nucleodídicas de aproximadamente 260.000 espécies (Benson et al, 2013). O Genbank começou a acumular sequências em 1982 já apresentando 606 sequências nucleotídicas, este número cresceu muito rápido e atualmente contém 171 milhões de sequências depositadas. O PubMed, de forma geral, é um base de dados de consulta bibliográfica com milhões de citações e resumos de artigos científicos.

No nosso dia a dia, como estudantes de bioinformática precisamos muitas vezes de armazenar dados e informações existentes nos ficheiros do banco de dados Genbank e consultar ainda literatura relacionada com as informações encontradas nestes ficheiros na base de dados Pubmed.

Desta forma, pretende-se com este trabalho aliar a consolidação da aprendizagem dos conteúdos lecionados na unidade curricular de Introdução aos Algoritmos, à Programação e às Bases de Dados ao nível do planeamento e execução de Sistemas de Bases de Dados para a automatização dos processos descritos anteriormente.

Por tudo isto, a base de dados a ser criada e implementada seria principalmente aplicada para armazenar e organizar as informações existentes nos ficheiros Genbank de forma lógica, eficiente e estruturada. Sendo também possível, neste SBD consultar o PubMed e retirar artigos relacionados com as sequências em causa e armazenar essa informação de uma forma organizada e seletiva.

2. Implementação do SBD

2.1. Identificação das Entidades, Atributos e Relacionamentos

Com base no objetivo do nosso trabalho achou-se pertinente um planeamento prévio da base de dados a ser implementada. Para isso, identificaram-se e caracterizaram-se potenciais entidades envolvidas no GenBank, bem como os vários atributos que as constituem e posteriormente identificaram-se e caracterizaram-se os diversos relacionamentos que existiam entre as entidades estabelecidas, tudo isto evidenciado nas tabelas seguintes.

Tabela 1- Identificação e caracterização de potenciais entidades envolvidas no caso de estudo apresentado, bem como os vários atributos que as constituem.

Entidade/ Relacionamento	Nome	Tipo e domínio	key
Gene Bank	Version(ID)	Texto (15)	S
	organism	Texto(45)	
	definition	Texto(400)	
	accession	Texto(10)	
	keywords	Texto(200)	
	taxonomy	Texto(500)	
References	Title(ID)	Texto(200)	S
	Pubmed_id	Texto(15)	
	journal	Texto(45)	
	consortium	Texto(100)	
	comment	Texto(1000)	
Pubmed Information	Pubmed_id	Texto(15)	S
	abstract	Texto(2000)	
	title	Texto(100)	
	Authors	Texto(1000)	
	Jornal	Texto(500)	
	affiliation	Texto(200)	
Locus	Locus_name	Texto(15)	S
	Modification_date	Texto(11)	
	Sequence_length	Inteiro	
	Genbank_division	Texto(7)	
	Molecule_type	Texto(20)	
	Topology	Texto(15)	
Features	Source	Texto(20)	S
	taxon	Texto(15)	
	gene	Inteiro	
	mRNA	Inteiro	
	regulatory	Inteiro	
	protein_bind	Inteiro	
	sig_peptide	Inteiro	
	misc_feature	Inteiro	
	misc_difference	Inteiro	
CDS	Protein_id	Texto(15)	S
	version	Texto(15)	
	product	Texto(45)	
	base_span	Texto(20)	
	gene	Texto(45)	
	translation	Inteiro	
Authors	Author_name	Texto(45)	S
Sequence	sequence	Texto(250)	S

Tabela 2- Identificação e caracterizaram-se os diversos relacionamentos que existiam entre as entidades estabelecidas

Entidade	#	Relacionamento	Entidade	#
Gene Bank	N..N	Genbank_References(tabela)	References	N..N
	1..1	Possui	Locus	1..1
	1..1	Possui	Features	1..1
	1..1	Possui	Sequence	1..1
Features	1..N	Contêm	CDS	N..1
References	1..N	Possui	PubMed Information	N..1
References	N..N	reference_authors(Tabela)	Authors	N..N

A partir destas tabelas foi então possível a idealização do passo seguinte: a construção dos modelos conceptual, lógico e, por fim físico.

Há três tipos de modelos de dados: o conceptual, o lógico e o físico. Todos concordam que uma estrutura de dados com qualidade é imprescindível para garantir a legitimidade do banco de dados facilitando a manutenção do sistema de aplicação.

2.2. Modelo Conceptual

Uma base de dados serve para gerir conjuntos de informação de modo a facilitar a organização, manutenção e pesquisa de dados.

O Modelo conceptual de base de dados é a representação física do modelo, e permite a qualquer pessoa compreender o raciocínio por detrás da base de dados criada. Para tal recorre-se ao Diagrama De Entidade e Relacionamento, onde deverão ser identificadas todas as entidades e os relacionamentos entre elas. O software utilizado para a realização do modelo conceptual da nossa base de dados foi o *TerraER*.

De seguida, apresenta-se o Diagrama De Entidade e Relacionamento (**Fig.1**)

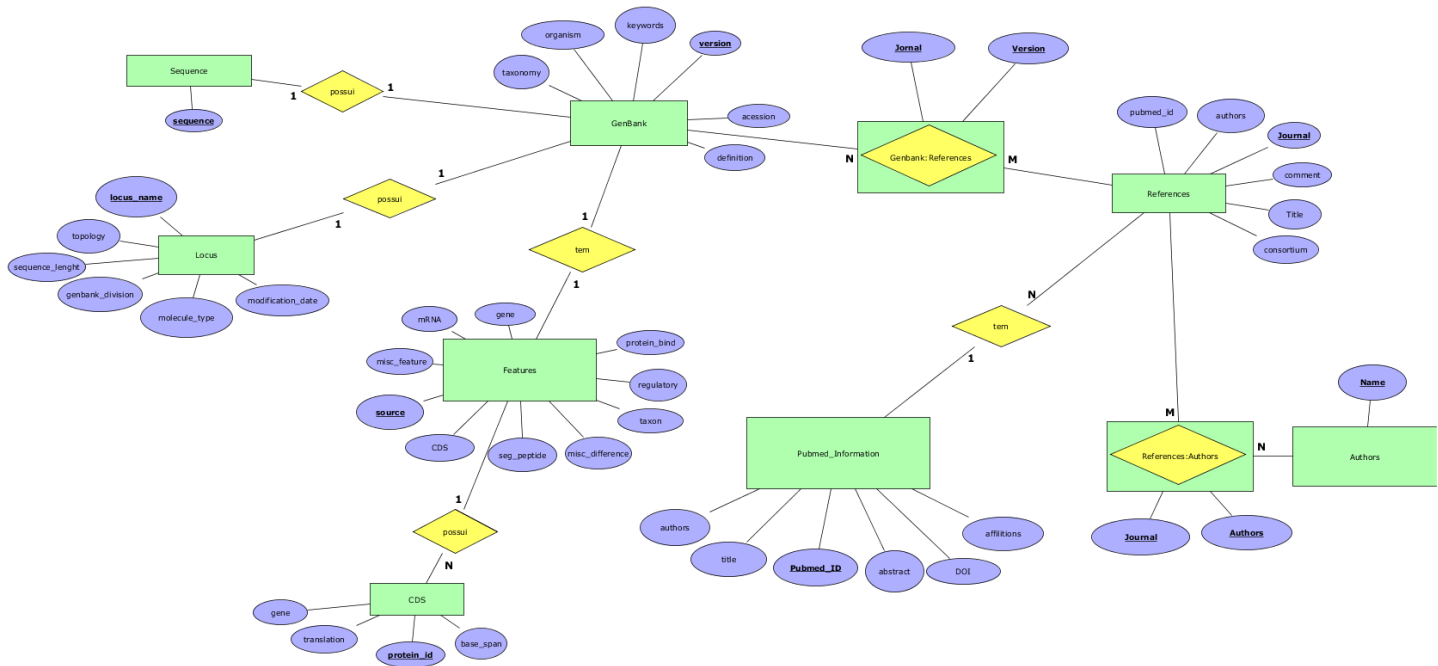


Fig.1- Esquema conceptual da base de dados aplicada ao armazenamento e organização das informações existentes nos ficheiros Genbank., realizado a partir do software *TerraER*.

2.3. Modelo Lógico

Após a realização do modelo conceptual desenvolvido (Fig.1), procedeu-se ao desenvolvimento do modelo lógico recorrendo ao MySQL Workbench.

A realização deste modelo é importante porque este estabelece a estrutura dos elementos de dados e os relacionamentos entre eles. O modelo de dados lógico leva os elementos de modelagem de dados conceituais um passo adiante, adicionando-lhes mais detalhes como chaves nas colunas, restrições, chaves primárias e estrangeiras. As colunas incluirão os atributos e seu respetivo tipo.

De seguida, apresenta-se o modelo lógico da nossa base de dados (**Fig.2**)

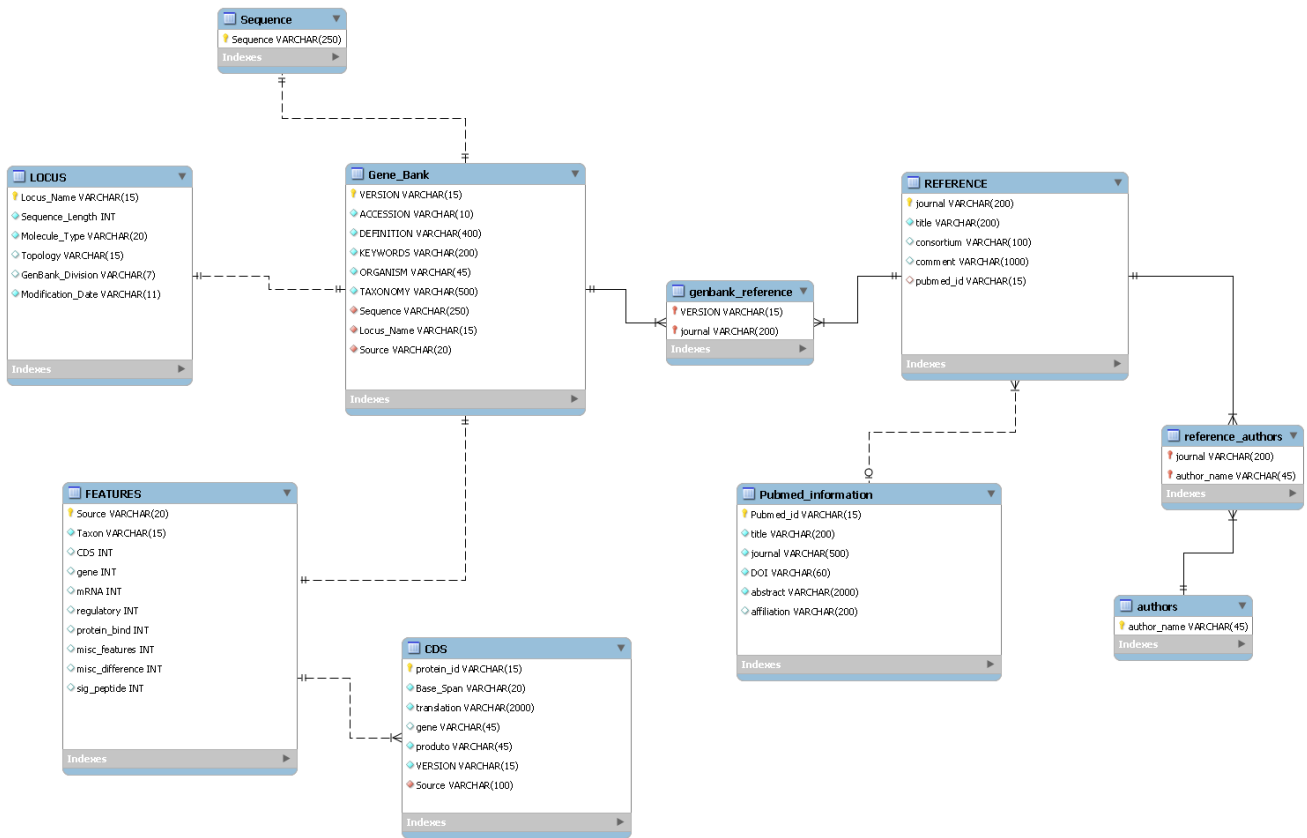


Fig.2- Esquema lógico da base de dados aplicada ao armazenamento e organização das informações existentes nos ficheiros Genbank., realizado a partir do MySQL Workbench.

2.4. Modelo Físico

Após a realização do modelo lógico precisávamos de um script para possibilitar a criação da base de dados e os seus respetivos elementos, por isso fizemos *Export* do ficheiro com o modelo lógico (ex: model.mwb), e depois o *Foward Engineer* para a criação da *script*. O respetivo *script* irá criar a base de dados, as tabelas com as respetivas colunas, relações e a partir deste implementar funções, procedimentos e eventuais triggers para proceder à execução de determinados processos associados ao SBD.



The screenshot shows the 'Review SQL Script' window of the 'Forward Engineer to Database' tool. The script is as follows:

```

44
45
46 -- Table 'trabalho'. 'FEATURES'
47
48 DROP TABLE IF EXISTS 'trabalho'. 'FEATURES' ;
49
50 CREATE TABLE IF NOT EXISTS 'trabalho'. 'FEATURES' (
51   'Source' VARCHAR(20) NOT NULL,
52   'Taxon' VARCHAR(15) NOT NULL,
53   'CDS' INT NULL,
54   'gene' INT NULL,
55   'mRNA' INT NULL,
56   'regulatory' INT NULL,
57   'protein_bind' INT NULL,
58   'misc_features' INT NULL,
59   'misc_difference' INT NULL,
60   'sig_peptide' INT NULL,
61   PRIMARY KEY ('Source'))
62 ENGINE = MyISAM ROW_FORMAT=COMPRESSED;
63
64
65
66 -- Table 'trabalho'. 'Gene_Bank'
67
68 DROP TABLE IF EXISTS 'trabalho'. 'Gene_Bank' ;
  
```

At the bottom of the window, there are buttons for 'Save to File...' and 'Copy to Clipboard'.

Fig 3: Excerto do *script* que originou o modelo físico da base de dados. É ainda possível ver o código que deu origem à tabela **'Features'**.

3- Desenvolvimento e implementação de um módulo Python para povoamento do Modelo Físico

Com o intuito de proceder à povoação automática do nosso recém gerado SBD, implementamos um módulo python. Este consiste num jupyter notebook baseado, maioritariamente, em BioPython e expressões regulares (ER), e através destes módulos foi-nos possível extrair toda a informação que consideramos pertinente, isto é, todos os atributos do modelo lógico. Adicionalmente, o módulo `mysql.connector` permitiu estabelecer a conexão entre o modelo e o notebook e assim proceder ao povoamento das tabelas.


3.1. Extração da informação de páginas do GenBank/PubMed e construção das tabelas

O sub módulo Entrez pertence ao BioPython permitiu-nos a extração rápida e eficiente da informação contida nos ficheiros GenBank e PubMed, demonstrado nas figuras seguintes:

```
handle = Entrez.efetch(db='nucleotide', id=insira, rettype='gb', retmode='text')
seq_record = SeqIO.read(handle, "gb")
print(seq_record)
```

```
handle = Entrez.efetch(db="pubmed", id=z, retmode="xml")
record = Entrez.read(handle)
```

Cada ficheiro GenBank contém uma estrutura única como demonstrado a seguir:



Sample GenBank Record

PubMed

Entrez

GenBank Flat File Format

Click on any link in this sample record to see a detailed description of that data element also return to the [Alphabetical Quicklinks Table](#) or [Resource Guide](#)

LOCUS SCU49845 5028 bp DNA PLN 21-JUN-1999
 DEFINITION Saccharomyces cerevisiae TCP1-beta gene, partial cds, and Axl2p (AXL2) and Rev7p (REV7) genes, complete cds.
 ACCESSION U49845
 VERSION U49845.1 GI:1293613
 KEYWORDS .
 SOURCE Saccharomyces cerevisiae (baker's yeast)
 ORGANISM Saccharomyces cerevisiae
 Eukaryota; Fungi; Ascomycota; Saccharomycotina; Saccharomycetes; Saccharomycetales; Saccharomycetaceae; Saccharomyces.
 REFERENCE 1 (bases 1 to 5028)
 AUTHORS Torpey,L.E., Gibbs,P.E., Nelson,J. and Lawrence,C.W.
 TITLE Cloning and sequence of REV7, a gene whose function is required for DNA damage-induced mutagenesis in Saccharomyces cerevisiae
 JOURNAL Yeast 10 (11), 1503-1509 (1994)
 PUBMED 7871890
 REFERENCE 2 (bases 1 to 5028)
 AUTHORS Roemer,T., Madden,K., Chang,J. and Snyder,M.
 TITLE Selection of axial growth sites in yeast requires Axl2p, a novel plasma membrane glycoprotein
 JOURNAL Genes Dev. 10 (7), 777-793 (1996)
 PUBMED 8846915
 REFERENCE 3 (bases 1 to 5028)
 AUTHORS Roemer,T.
 TITLE [Direct Submission](#)
 JOURNAL Submitted (22-FEB-1996) Terry Roemer, Biology, Yale University, New Haven, CT, USA
 FEATURES Location/Qualifiers

Fig.4: Exemplo da estrutura de um ficheiro GenBank.

Considerando este exemplo, decidimos subdividir a informação em 10 tabelas distintas “Sequence”, “LOCUS”, “Gene_Bank”, “FEATURES”, “CDS”, “Pubmed_information”, “REFERENCE”, “genbank_reference”, “author”s e “reference”. authors

Gene_Bank

Para a tabela Gene_Bank definimos como chave primária a “VERSION”, uma vez que esta está presente em todas as entradas GenBank, identificando cada uma delas de forma única. Adicionámos a esta tabela ainda 5 atributos (“ACCESSION”, “DEFINITION”, “KEYWORDS”, “ORGANISM” E “TAXONOMY”) e 3 chaves estrangeiras (“Sequence”, “Locus_Name” e “Source”). Estas chaves estrangeiras associam as tabelas “Sequence”, “LOCUS”, “FEATURES” à tabela Gene_Bank numa relação de 1:1.

Sequence

Para a tabela “Sequence” e considerando que a sequência de cada ficheiro é única, decidimos que esta seria apenas constituída pela sequência de nucleótidos em questão (“Sequence”), sendo assim a chave primária que estabelece conexão com a tabela Gen_Bank. Posteriormente esta mesma tabela poderá ser usada no trabalho de outra UC nomeadamente a UC de Algoritmos de Análise de Sequências Biológicas.

LOCUS

A tabela “LOCUS”, possui como chave primária o nome do Locus (“Locus_Name”), e 5 atributos (“Topology”, “sequence_length”, “genbank_division”, “molecule_type”, e “modification_date”).

FEATURES

Para a tabela “FEATURES” optamos por seleccionar a source do organismo como chave primária (“Source”) uma vez que esta é um atributo único das Features, e adicionamos vários atributos que correspondem às features que mais frequentemente surgem em ficheiros GenBank (“CDS”, “mRNA”, “gene”, “regulatory”, etc) e ainda o taxon (“taxon”) que identifica o tipo de organismo.

CDS

A tabela “CDS” associa-se com a tabela “FEATURES” na medida em que o campo de FEATURES pode conter várias regiões codificantes (“CDS”) numa relação de 1:N. A chave primária é o id da proteína (“protein_id”) codificada pela CDS correspondente. Para além disso, temos mais 3 atributos (“gene”, “translation” e “base_span”) que fornecem informações adicionais para cada CDS.

REFERENCE

A tabela “REFERENCE” estabelece uma relação de N:N com a tabela Gen_Bank, uma vez que cada ficheiro GenBank pode ter várias referências e uma determinada referência pode estar presente em vários ficheiros GenBank. A chave primária para esta tabela revelou-se uma escolha difícil, dado que a informação referente a este campo existente em cada ficheiro é algo variável. Tendo isto em conta, escolhemos o journal como a nossa chave primária. Para atributos adicionamos mais informação relativa à referência (“title”, “comment”, “consortium”, “authors”). Dos referidos anteriormente, apenas o título e o journal são atributos não nulos. Por fim, utilizamos como chave estrangeira o id do artigo do pubmed (“pubmed_id”) que poderá ou não estar presente em cada referência e que estabelece a relação com o artigo na base de dados do pubmed e com cada autor implicado na respetiva referência.

Pubmed_Information

A tabela “Pubmed_Information” contém a informação relativa ao artigo retirada da base de dados PubMed. Esta provém diretamente do id presente (não obrigatoriamente) em cada referência. Em função disto a chave primária é o id do artigo (“pubmed_id”), na medida em que cada referência só pode conter no máximo 1 id mas um determinado id pode estar presente em mais que uma referência (relação 1:N).

Authors

A tabela “Authors” surgiu da necessidade de permitir queries mais avançadas, *e.g.*, determinar que autores é que estão associados a certos artigos. Esta é apenas constituída pelo nome dos autores sendo este a chave primária que estabelece uma relação de N:N com a tabela das referências.

Genebank_references

Esta tabela representa o relacionamento entre as tabelas Gene_Bank e Reference e possui como atributos as chaves estrangeiras/primárias “journal” e “VERSION”.

Reference_authors

Esta tabela representa o relacionamento entre as tabelas authors e Reference e possui como atributos as chaves estrangeiras/primárias “authors_name” e “journal”.

3.2. Povoamento

Como referido anteriormente, o povoamento do nosso modelo foi efetuado através do módulo mysql.connector. Numa primeira instância, estabelecemos a conexão entre o nosso notebook e o sql Workbench, tal como se segue:

```
import mysql.connector
from mysql.connector import Error

db = mysql.connector.connect(host="127.0.0.1",
                             user="root",
                             passwd="PASSWORD",
                             db="trabalho",
                             auth_plugin = "mysql_native_password"
                             )
```

De seguida inserimos nas diferentes tabelas as informações recolhidas na forma de variáveis python. A ordem pela qual as tabelas são preenchidas não resulta de uma escolha arbitrária, uma vez que esta é dependente do planeamento feito previamente, isto é, uma tabela com chave estrangeira só poderá ser preenchida quando a sua chave primária corresponde já se encontrar na base de dados. Daí a seguinte ordem de inserção:

Inserção na tabela Sequence

```
cur = db.cursor()
cur.execute("""
    INSERT INTO sequence (sequence)
    VALUES (%s)
    """,
    (a))

db.autocommit = True
```

Inserção na tabela LOCUS

```
cur = db.cursor()
cur.execute("""
    INSERT INTO Locus (Locus_Name, Sequence_Length, Molecule_Type, Topology, GenBank_Division, Modification_Date)
    VALUES (%s,%s,%s,%s,%s,%s)
    """,
    (nome, tamanho, molecule_type, topologia, data_fyle_division,data_modificacao))

db.autocommit = True
```

Inserção na tabela FEATURES

```
cur = db.cursor()
cur.execute("""
    INSERT INTO Features (Source, Taxon, CDS, gene, mRNA, regulatory, protein_bind, misc_features, misc_difference, sig_peptide)
    VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)
    """,
    (source, taxon[0], CDS, GENES, mRNA, Regulatory, ProteinBind, misc_feature, misc_difference, sig_peptide))

db.autocommit = True
```

Inserção na tabela Gene_Bank

```
cur = db.cursor()
cur.execute("""
    INSERT INTO Gene_Bank (ACCESSION, VERSION, DEFINITION, KEYWORDS, ORGANISM, TAXONOMY, Sequence, Locus_Name, Source)
    VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s)
    """,
    (accession, id, definicao, keywords[0], organismo, taxonomia, a[0], nome, source))
db.autocommit = True
```

Inserção na tabela Pubmed_Information

```
cur = db.cursor()
y = 0
for x in ids_pubmed:
    if len(ids_pubmed[y]) == 0:
        pass
    else:
        cur.execute("""
            INSERT INTO Pubmed_information (Pubmed_id, title, journal, DOI, abstract, affiliation)
            VALUES (%s,%s,%s,%s,%s,%s)
            """,
            (repr(ids_pubmed[y]), repr(title[y]), repr(journal[y]), repr(DOI[y]), repr(abstract[y]), repr(Afiliacao[y])))
        y += 1
db.autocommit = True
```

Inserção na tabela authors

```
cur = db.cursor()
for x in auu:
    cur.execute("""
        INSERT INTO authors (author_name)
        VALUES (%s)
        """,
        (x,))
db.autocommit = True
```

Inserção na tabela reference

```
cur = db.cursor()
y = 0
for x in ids_pubmed:
    cur.execute("""
        INSERT INTO reference (journal, title, pubmed_id, consortium, comment)
        VALUES (%s,%s,%s,%s,%s)
        """,
        (journal[y], title[y], ids_pubmed[y], consortium[y], comment[y]))
    y += 1
db.autocommit = True
```

Inserção na tabela CDS

```
cur = db.cursor()
y = 0
for x in CDS_id:
    cur.execute("""
        INSERT INTO CDS (protein_id, Base_Span, translation, gene, produto, VERSION, Source)
        VALUES (%s,%s,%s,%s,%s,%s,%s)
        """,
        (x[0], CDS_loc[y], CDS_aa[y][y], CDS_gene[y][y], CDS_prod[y][y], id, source))
    y += 1
db.autocommit = True
```

Inserção na tabela `reference_authors`

```
cur = db.cursor()
for i, c in enumerate(journal):
    for p in lista_final[i]:
        cur.execute("""
            INSERT INTO reference_authors(journal, author_name)
            VALUES (%s,%s)
            """,
            (journal[i], p))

db.autocommit = True
```

Inserção da tabela `genbank_reference`

```
cur = db.cursor()
for x in range(len(journal)):
    cur.execute("""
        INSERT INTO genbank_reference (VERSION, journal)
        VALUES (%s,%s)
        """,
        (id, journal[x]))

db.autocommit = True
```

4. Desenvolvimento e Implementação de Funções e *Triggers*

Os *triggers* são objetos que fazem parte do banco de dados que, relacionados a certa tabela, permitem a execução de processamentos em consequência de uma determinada ação. Podemos por isso programar que ações sejam executadas antes (BEFORE) ou depois (AFTER) de fazer uma inserção (INSERT), alteração (UPDATE) ou remoção (DELETE) de registos de uma determinada tabela (ON TABLE *nome_da_tabela*).

Na nossa ótica, o desenvolvimento e implementação de triggers não constitui um papel necessário no nosso modelo, uma vez que a inserção dos dados foi efetuada com auxílio das funcionalidades dos jupyter notebooks e consequente não inviabilizando a base de dados

5. Análise crítica do trabalho realizado. Eventuais futuras ações corretivas ou melhorias

Durante a realização deste trabalho, enfrentamos alguns contratempos nomeadamente dificuldade em definir, identificar e caracterizar potenciais entidades, relacionamentos e chaves primárias.

O problema anterior não se deveu propriamente à falta de conhecimentos sobre o funcionamento do ficheiro Genbank porque a sua estrutura é bem definida, no entanto a sua informação é muito heterogénea e inconstante dificultado a parte de planeamento da base de dados e posterior povoamento da mesma, por exemplo numa primeira fase, na tabela “Reference” definimos o título como chave primária o que se revelou impraticável já que mais do que uma referência possuía como título “Direct submission” o que resultaria numa impossibilidade já que a chave primária seria um atributo único.

Após a realização e implementação da base de dados, uma possível melhoria seria a introdução de um maior número de ficheiros Genbank de forma a ver se tanto a script como a base de dados funcionam sem qualquer tipo de erros, isto porque neste trabalho inserimos um número pequeno de ficheiros Genbank.

Ao realizar a inserção de mais ficheiros Genbank também, abria possibilidade da introdução de mais alguns atributos que fossem considerados importantes e que não foram necessários para os ficheiros Genbank adicionados na nossa base de dados. Isto porque, mais informação nas tabelas tornaria esta base de dados numa ferramenta mais multifuncional, flexível e robusta.

Apesar da nossa base de dados ser funcional, existem algumas possíveis otimizações, nomeadamente a implementação de triggers e o desenvolvimento de uma função capaz de ler um ficheiro GenBank criando automaticamente colunas correspondentes às suas eventuais features na tabela “FEATURES”. Outra possível melhoria será a substituição de campos vazios por algo mais intuitivo.

Por fim, , a criação de mais tabelas que são de momento atributos de outras tabelas poderá aprofundar a robustez de queries a realizar devido à inerente criação de mais relacionamentos como é o caso da coluna “affiliation” na tabela “Pubmed_information”.