Base de Dados a partir de um ficheiro genbank

Trabalho prático de Introdução aos Algoritmos, à Programação e às Bases de Dados

Mestrado em Bioinformática

Realizado por:

João José Lopes Cheixo – PG49837

Márcia Filipa Gonçalves Oliveira – PG49841

Mariana Rochinha Braguês – PG49843

INTRODUÇÃO

Este trabalho prático centra-se no desenvolvimento de ferramentas e recolha de dados de um ficheiro *genbank* de vários organismos para a elaboração e manutenção de um Sistema de Base de Dados (SBD) relacional.

Os ficheiros *genbank* desenvolvidos pelo Centro Nacional de Informações sobre Biotecnologia dos EUA (NCBI) estão associados à conservação de dados relacionados com o DNA e informação de sequenciação de proteínas, contendo também comentários, anotações e referências.

A análise dos ficheiros *genbank* escolhidos foi concentrada em organismos responsáveis por infeções associadas aos cuidados de saúde, nomeadamente bactérias. Estas são adquiridas em contexto de prestação de cuidados de saúde, como em internamentos hospitalares. Estas infeções podem ser tratadas com antibióticos adequados, sendo por isso relevante o estudo de bactérias potencialmente resistentes.

O nosso objetivo principal será verificar a qualidade do ficheiro *genbank*, a partir das informações que ele fornece acerca de organismos. Assim sendo, será desenvolvido um sistema eficiente que nos permita saber informações destes organismos.

PLANIFICAÇÃO DO SBD A REALIZAR

De acordo com o referido anteriormente, a base de dados proposta pretende facilitar a recolha de informações de organismos armazenada num ficheiro *genbank*.

Inicialmente, foram analisados vários ficheiros *genbank* de bactérias e o seu conteúdo para uma melhor perceção da base de dados a implementar.

Verificamos a existência de sete entidades: "Bactéria", "Locus", "Sequência", "Reference", "Autor", "Annotations" e "Features", e de três entidades/relacionamentos: "bac_feat", "bac_ref_ann" e "autor_has_ref". Na entidade "Bactéria" é apresentada uma informação geral de todos os organismos presentes na base de dados, nomeadamente o nome da bactéria, o seu id, uma breve descrição, o link redirecionado para o NCBI e o link de pesquisa na *PubMed*. A entidade "Locus" contém informação sobre o id do locus e o número de pares de bases. Na entidade "Sequência" temos acesso ao tamanho da sequência do microrganismo. Na entidade "Reference" podemos consultar dados sobre as referências bibliográficas, de onde foram retiradas as informações presente no ficheiro *genbank*, nomeadamente o título e o jornal de publicação. Na entidade "Autor" temos acesso à filiação das referências obtidas. Desta forma, percebemos que as entidades "Reference" e "Autor" estão relacionadas, necessitando de adicionar uma entidade/relacionamento: "autor_has_ref". Em "Annotations" conseguimos obter a taxonomia, o organismo, o tipo de molécula, as palavras-chave relevantes relacionadas com o mesmo, entre outras. Além disso, também a entidade "Annotations" se relaciona com a entidade "Reference" e com a entidade "Bactéria", criando-se uma entidade/relacionamento: "bac_ref_ann". Em "Features" conseguimos ver algumas anotações biológicas: quantas features estão presentes no ficheiro, o seu tipo e respetivas localizações. Entre esta última entidade e a "Bactéria" foi adicionada uma entidade/relacionamento: "bac_feat".

Posto isto, conseguimos perceber as informações principais e necessárias que este tipo de ficheiro pode fornecer para desenvolver o SBD.

Após o levantamento de todos os requisitos foi feito o planeamento do SBD. Assumiu-se que cada "Locus" e cada "Sequência" pertencem obrigatoriamente a uma e

única “Bactéria” e que cada “Bactéria” pertence obrigatoriamente a apenas um “Locus” e uma “Sequência” (cardinalidade 1:1). Cada “Bactéria” tem múltiplas “Features” e estas “Features” estão relacionadas com apenas uma “Bactéria” (cardinalidade 1:N). Também cada “Bactéria” tem várias “Reference” e “Annotations”, no entanto cada uma destas está relacionada com uma única “Bactéria” (cardinalidade 1:N). Cada “Reference” tem múltiplos “Autor” e cada “Autor” pode estar relacionado com várias “Reference” (cardinalidade N:M).

MODELO CONCEPTUAL

Com base no que foi descrito no capítulo anterior, foi possível desenvolver o modelo conceptual deste SBD no software *TerraER*. A cada entidade foram conferidos diversos atributos. Os atributos sublinhados correspondem a atributos chave primária, enquanto que os restantes correspondem a atributos simples.

Tabela 1 - Entidades do SBD

Entidades:	**Bactéria**	**Locus**	**Sequência**	**Features**	**Annotations**	**Reference**	**Autor**
Atributos	id_bacteria	id_locus	id_seq	id_features	id_annotations	id_reference	id_autor
	nome	id_l	len_seq	tamanho	source	journal	filiação
	ID	bp		tipo e localização	keywords	title	
	descrição				accessions		
	Link_NCBI				organism		
	Link_PubMed				taxonomy		
					letter_annotations		
					len_annotations		
					mol_type		

Relacionamentos

Após serem identificadas as entidades e os respetivos atributos, são estabelecidos os relacionamentos entre elas. Foram definidos relacionamentos 1:1 entre as entidades “Bactéria” e “Locus”, “Bactéria” e “Sequência”. Foram estabelecidos relacionamentos 1:N entre as entidades “Bactéria” e “Features”, “Bactéria” e

"Reference", "Bactéria" e "Annotations". Foi ainda definido um relacionamento N:M entre as entidades "Reference" e "Autor".

Relacionamento **"Pertencer"**: Estabelecido entre as entidades "Locus" e "Sequência" e a entidade "Bactéria". Cada "Locus" e cada "Sequência" é obrigatoriamente pertencente a uma e apenas uma "Bactéria".

Entidade/relacionamento: "autor_has_ref", "bac_feat" e "bac_ref_ann".

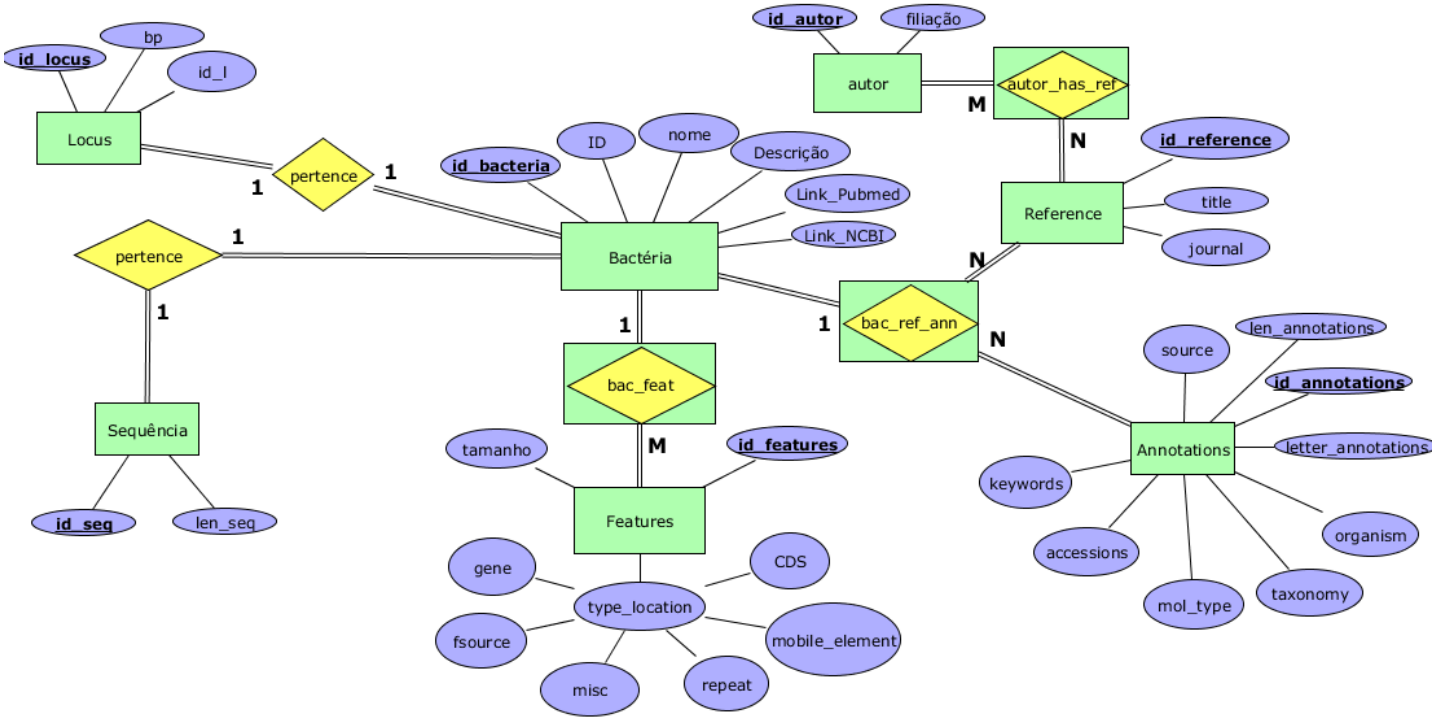

Figura 1 - Modelo conceptual

MODELO LÓGICO/MODELO RELACIONAL E MODELO FÍSICO

O modelo lógico/modelo relacional, assim como o modelo físico, do SBD foi desenvolvido no software *MySQL Workbench*, com base no modelo conceptual anteriormente apresentado. Neste modelo, cada entidade tem uma tabela associada, onde as colunas correspondem aos atributos de cada uma.

Para isso, foram aplicadas restrições aos atributos: distinguimos os atributos nulos (*NULL*) dos não nulos (*NOT NULL*) e a sua tipologia (distinguir números inteiros - *INT* - de sequências de caracteres - *VARCHAR*).

Cada entidade tem também uma chave primária associada (tabela 1). A entidade "Sequência" e a entidade "Locus" apresentam uma chave estrangeira (foreign keys) - "bacterias_idbacterias" - que é chave primária da entidade "Bactéria". A entidade/relacionamento "bac_feat" apresentam duas chaves estrangeiras - "features_id_features" e "bacterias_idbacterias" - que são chaves primárias das entidades "Features" e "Bactérias", respetivamente. A entidade "bac_ref_ann" é composta por três chaves estrangeiras - "reference_id_reference", "bacterias_idbacterias" e "annotations_id_annotations" - que são chaves primárias das entidades "Reference", "Bactérias" e "Annotations", respetivamente. Por último, a entidade "autor has reference" tem duas chaves estrangeiras - "reference_id_reference" e "autor_id_autor" - que são chaves primárias das entidades "Reference" e "Autor", respetivamente.

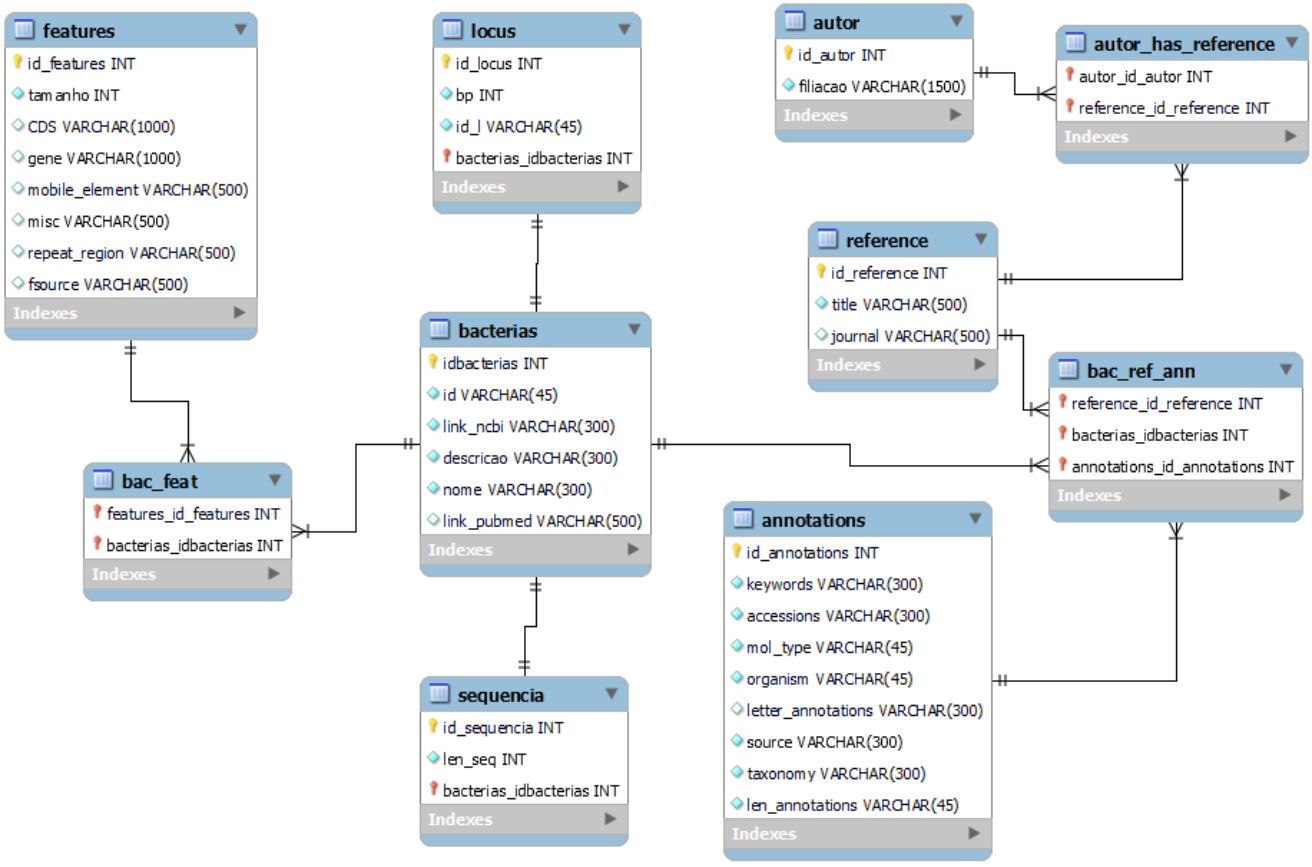

Figura 2 - Modelo no MySQL Workbench

POVOAMENTO DAS TABELAS ATRAVÉS DO MÓDULO *PYTHON* "povoarMMJ"

Para o povoamento das tabelas do modelo físico foi criado um módulo *Python* denominado "povoarMMJ". Para isso foi necessário realizar a conexão entre MySQL e *Python*, começando por importar os seguintes packages:

```
import mysql
import mysql.connector
from mysql.connector import Error
import pandas as pd
import mysql.connector as SQLC
import re
import os
from os import listdir
from os.path import isfile, join
from Bio import SeqIO
from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
from Bio import SeqFeature
```

Seguidamente, foi definida uma função utilizando *Python* denominada "create_db_connection", com o intuito de definir a sua conexão com o *MySQL Server* e com a base de dados, e a função "execute_query" que permite executar *queries* escritas no *MySQL*.

```
def create_db_connection(host_name, user_name, user_password, db_name):
    connection = None
    try:
        connection = mysql.connector.connect(
            host=host_name,
            user=user_name,
            passwd=user_password,
            database=db_name
        )
        print("MySQL Database connection successful")
    except Error as err:
        print(f"Error: '{err}'")

    return connection


def execute_query(connection, query):
    cursor = connection.cursor()
    try:
        cursor.execute(query)
        connection.commit()
        print("Query successful")
    except Error as err:
        print(f"Error: '{err}'")
```

Em seguida, foram definidas várias funções para extrair a informação dos ficheiros *genbank* e inseri-la nas respetivas tabelas do nosso SBD.

```
def multipla_auto_populacao():
    path = input("Coloque o caminho completo para a pasta que contem os ficheiro genbank:")
    lista_fich=[]
    lista_fich=demonstra_ficheiros(path)
    host_name =input("Nome de host")
    user_name = input("Nome do usuario")
    user_password = input("Password do usuario")
    db_name = input("Nome da database")
    connection = create_db_connection(host_name,user_name,user_password,db_name)
    desejo= input("Prima 1 se deseja efetuar um comando para a database ")
    if desejo=="1":
        query = input("Comando para db:")
        execute_query(connection,query)

    for file in lista_fich:
        povoar_locus(file)
        len_sequencia(file)
        features(file)
        titulo_journal(file)
        insere_jornal(file)
        filiacao_autores(file)
        annotations(file)
        nome = nome_org(file)
        valida_url_pubmed(nome)

    return 0
```

A função "multipla_auto_populacao" funciona como uma função *main* que resume todas as outras. Consiste numa função que, automaticamente, identifica todos os ficheiros *Genbank* num dado *folder* e atualiza-os para uma base de dados num *server*, usando todas as funções desenvolvidas no módulo.

FUNÇÕES, PROCEDIMENTOS E *TRIGGERS*

A função **"demonstra_ficheiros"** recebe um *path* e, foram usadas as funções do sistema operativo "*listdir*" e "*isfile*" para juntar numa lista (designada de *files)* todos os ficheiros contidos no *path* dado. De seguida, num *loop for*, em todos os ficheiros contidos na lista *files* foi armazenada numa lista "a" aqueles que são do tipo .gb (*genbank*), retorno essa lista "a".

A função **"create_db_connection"** é a que, dando um dado *host_name*, *user_name*, *user_password* e *db_name*, estabelece a conexão com a *database* e imprime uma mensagem de sucesso caso seja estabelecida conexão, senão imprime mensagem de erro; se for bem-sucedida retorna a conexão na variável *connection*.

Na função **"execute_query"**, dada uma variável *connection* e uma *query*, imprime essa *query* para a *database* em causa (já explicita pela conexão). Em caso de sucesso, é mostrada uma mensagem de sucesso, senão imprime mensagem de erro.

A função "**povoar_locus**" recebe um organismo como argumento, que essencialmente será um ficheiro *genbank*, e daí extrai-se para a variável *org* a sequência de DNA, o ID, o número de pares de base do locus e insere-se na base de dados.

A função "**len_sequencia**", similarmente recebe um ficheiro *genbank*, a qual é extraída a sequência de DNA e determina-se o tamanho desta numa variável *len_seq* que será inserida na *database*.

A função "***features***" recebe um ficheiro *genbank*, *org*, que similarmente é tratado com a função *seqIO.read*, num *loop for* até percorrer todas as *features* do *org*. Guardam-se todos os tipos de CDS numa lista chamada *CDS* e, em seguida, para cada elemento na lista, faz *print* da localização e do tamanho de CDS. Esta lógica irá repetir-se para o termo *gene, mobile_element, source, repeat_region, misc-feature*, e no final coletam-se todos estes atributos e inserem-se na *database*.

A função "**titulo_journal**" recebe um ficheiro *genbank*, coleta todos os títulos de um journal e faz *print* dos mesmos.

A função "**insere_journal**" envia todos os *journals* para a database em conjunto com os seus títulos.

A função "**filiacao_autores**" determina a filiação de autores e insere na database.

A função "**annotations**" recebe como argumento um ficheiro *genbank* e coleta esse ficheiro todas as instâncias de *source, organsim, taxonomy, molecule_type, acessions* tornando-as em *keywords* e insere-as na *database*.

A função "**nome**" recebe um ficheiro *genbank* como argumento, procura o *link NCBI* e o nome do organismo em questão, retornando-o.

A função "**url**" recebe um nome de um organismo e devolve o link da *pubmed* correspondente.

A função "**validar**" recebe um nome de um organismo e retorna se esse é válido.

A função "**valida_url_pubmed**" recebe um nome de um organismo e tenta validá-lo, através da função "validar", passando o nome do organismo como argumento. Se tal não for possível, é lançada uma exceção, caso contrário faz *print* do resultado da

função "url" com o nome do organismo passado como argumento, define-se a variável *link_pubmed* como o resultado da função "url" e insere-se na *database*.

A função "**multipla_auto_populacao**" não tem argumentos e retorna 0, um valor arbitrário que simplesmente simboliza que esta correu sem problemas. Esta função tem como objetivo fazer tudo o necessário para que vários ficheiros *genbank* possam ser tratados ao mesmo tempo sem que o usuário necessite de invocar várias vezes as mesmas funções. Assim sendo, esta função realiza as anteriores e funciona com um ou mais ficheiros *genbank* a serem tratados. Inicialmente, pede ao usuário que explicite o *path* (diretoria) para a pasta que contém os ficheiros *genbank* a serem tratados, corre a função "demonstra_ficheiros" com o *path* indicado pelo usuário e guarda o resultado numa lista chamada *lista_fich*. Em seguida, pede ao usuário que submeta valores para *o host_name, user_name, user_password, db_name*, e estabelece uma conexão correndo a função "create_db_connection" com os valores previamente pedidos como argumentos. Posteriormente, pergunta se o usuário deseja efetuar uma *query* para a *database*, caso queira o usuário prime 1 e, de seguida, explicita a *query* a ser efetuada, passando esta a ser usada como argumento para correr a função "execute_query", juntamente com a conexão previamente estabelecida. Depois, faz a parte da função que confere a multiplicidade de tratamento de ficheiros: um *loop for* que, para cada ficheiro na lista de ficheiros *genbank,* corre as seguintes funções, todas estas com o atual ficheiro como argumento: "povoarlocus", "lensequencia", "features", "titulo_journal", "insere_journal", "filiacao autores", "annotations" e, ainda, dentro do *ciclo for* guarda numa variável *nome* o resultado da função "nome_org" (sendo o argumento para esta ainda o ficheiro do *loop*), e usa a variável do nome para ser o argumento para a próxima e última função que executa a "valida_url_pubmed".

Não foram adicionados *triggers* ao SBD criado.

ANÁLISE CRÍTICA/CONCLUSÃO

Este SBD poderia ser melhorado, visto que o SBD desenvolvido apresenta informação repetida: o atributo "len_seq" da entidade "Sequência" tem o mesmo valor do atributo "bp" da entidade "Locus". Outra melhoria a considerar seria a automatização da extração dos ficheiros *GenBank* de cada organismo e o seu respetivo armazenamento.

Na execução do nosso trabalho sentimos alguma dificuldade na interpretação e na organização da informação a retirar da tipologia de ficheiro *genbank* para, posteriormente, implementar o nosso SBD.

Outra dificuldade foi sentida na conexão ao servidor "iap". Todo o SBD foi desenvolvido fora desse servidor (num servidor local) e, aquando da sua transferência para o mesmo, esta não foi realizada com êxito. O "*schema*" teve de ser novamente criado no servidor "iap".

Em suma, a base de dados desenvolvida poderá ser utilizada para otimizar o estudo de um ou vários organismos, bem como a sua relação, pertencentes a um dado projeto/investigação. Inicialmente, foi pensada para armazenar informação sobre bactérias, mas poderá ser aplicada para outro tipo de organismo.