UNIVERSIDADE FEDERAL DO PARANÁ

DIEVAL GUIZELINI

BANCO DE DADOS BIOLÓGICO NO MODELO RELACIONAL PARA MINERAÇÃO DE DADOS EM GENOMAS COMPLETOS DE PROCARIOTOS DISPONIBILIZADOS PELO NCBI GENBANK

CURITIBA
2010

DIEVAL GUIZELINI

BANCO DE DADOS BIOLÓGICO NO MODELO RELACIONAL PARA MINERAÇÃO DE DADOS EM GENOMAS COMPLETOS DE PROCARIOTOS DISPONIBILIZADOS PELO NCBI GENBANK

Dissertação apresentada ao Curso de Pós-Graduação em Bioinformática, Área de Concentração em Bioinformática, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná, como parte dos requisitos para obtenção do título de Mestre em Bioinformática.

Orientador: Prof. Dr. Roberto Tadeu Raittz
Co-orientador: Prof. Dr. Fabio de Oliveira Pedrosa

CURITIBA
2010

TERMO DE APROVAÇÃO

DIEVAL GUIZELINI

BANCO DE DADOS BIOLÓGICO NO MODELO RELACIONAL PARA MINERAÇÃO DE DADOS EM GENOMAS COMPLETOS DE PROCARIOTOS DISPONIBILIZADOS PELO NCBI GENBANK

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre em Bioinformática, pelo Programa de Pós-Graduação em Bioinformática, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná, pela seguinte banca examinadora:

Orientador: Prof. Dr. Roberto Tadeu Raittz

Co-orientador: Prof. Dr. Fabio de Oliveira Pedrosa

Prof. Dr. Júlio Cesar Nievola
Pontifícia Universidade Católica do Paraná

Profª Drª Rose Adele Monteiro
Universidade Federal do Paraná

Profª Drª Jeroniza Nunes Marchaukoski
Universidade Federal do Paraná

Curitiba, 29 de outubro de 2010

À minha mãe, Alice.
À minha esposa, Juliana.
Aos meus filhos, Lucas e João.
Aos meus irmãos, Dayane e Diges.

AGRADECIMENTOS

À minha mãe, Alice Feltrin, pela vida e pela educação que me propiciou.

À minha esposa Juliana, companheira na longa jornada da vida; mulher que amo mais do que sou capaz de expressar.

Ao professor doutor Roberto Tadeu Raittz, meu amigo e orientador, pelo incentivo e tempo dedicado à realização dessa pesquisa.

Ao professor doutor Fabio de Oliveira Pedrosa, meu co-orientador, pelas dicas, discussões e, principalmente, por ter me ajudado a manter o foco na pesquisa e no desafio de definir o escopo desse trabalho.

À minha amiga Jeroniza Nunes Marchaukoski, pela paciência e pelas inúmeras discussões sobre o tema de banco e correlatos.

Ao companheiro de luta, Mario de Paula Soares Filho, pela prontidão em momentos críticos, mesmo nas horas mais impróprias, e ainda pelo tempo para discutir dúvidas que nos perseguem no dia-a-dia.

Aos professores Geraldo Picheth e Leda Satie Chubatsu, em especial, por terem ajudado a melhorar minha didática e por terem contribuído para que eu mudasse minha visão sobre a pesquisa científica.

Aos professores Emanuel Maltempi de Souza e Maria Berenice Reynaud Steffens, pelas sugestões e considerações feitas ao longo do desenvolvimento deste trabalho.

Aos professores e colegas Giselle Munhoz Alves, Leonardo Magalhães Cruz, Liu Un Rigo e Paulo Afonso Bracarense Costa pelo convívio e contribuições.

Aos meus amigos Michelly Gehlen, Eduardo Tieppo, Rodrigo Cardoso e Waldir Silva pela amizade, pelas conversas e parcerias que certamente irão aumentar e dar bons frutos para todos.

Aos colegas Ademir Prado, Daniela Tapety, Guilherme Willemann, Leandro Steim, Luiz Fernando Soares, Rosa Cordélli, Terumi Kamada, Waldemar Volanski pelas contribuições e pelo bom convívio nesses últimos dois anos.

Aos amigos que conheci na Escola Técnica da UFPR, Mauro José Belli e Renato R. C. Dutra, e que hoje estão em outros lugares, pelo incentivo à docência e à continuidade nos estudos.

Ao professor Dartagnan Baggio Emerenciano, pela amizade e pelos mais de doze anos de luta juntos e porque “a luta continua companheiros” por muitos e longos anos.

Aos meus amigos Jaime, Sandramara, Luiz Antônio, Rafaela, Maria Valéria, Cícero, João Negrão, Marlene, Marly, Maura, Adão, Marinalva e Noeli, pelos anos de convívio e companheirismo, mesmos nos dias mais difíceis de nossas vidas.

Aos colegas Lucas, Neves, Paulo Radtke, Pedro, Cátia, Alessandro, Rafael e Paulo Moraes, que chegaram mais recentemente ao nosso grupo, mas que direta e indiretamente são colaboradores.

Aos meus amigos do Setor de Educação Profissional e Tecnológica, por me apoiarem nesta trajetória.

À doutora Maria Júlia Camina Bugalo, neuropediatra, que tem cuidado do Lucas desde os quatro anos de idade e nos ajudado e muito nesses longos anos.

Ao Programa de Pós-Graduação em Bioinformática.

Ao Instituto Nacional em Ciência e Tecnologia em Fixação Biológica de Nitrogênio.

Ao National Center for Biotechnology Information.

A todos que, direta ou indiretamente, contribuíram para eu concretizar este trabalho, sou, sinceramente,

Muito grato!

Ando devagar
porque já tive pressa
levo esse sorriso
porque já chorei demais

Hoje me sinto mais forte,
mais feliz, quem sabe,
só levo a certeza
de que muito pouco sei,
ou nada sei
(...)
É preciso amor
pra poder pulsar
é preciso paz pra poder sorrir
é preciso a chuva para florir
(...)
Cada um de nós compõe a sua história
cada ser em si
carrega o dom de ser capaz
e ser feliz.

Tocando em Frente
Almir Sater e Renato Teixeira

RESUMO

O NCBI GenBank, um dos três principais bancos de dados primários, tem centralizado as informações obtidas pelos processos de sequenciamento de DNA e/ou RNA e as tem distribuído no formato de arquivos textos. Nos servidores de arquivos do GenBank, para o Domínio Bactéria e Domínio Archea, existe um arquivo em formato específico para cada organismo, cromossomo ou plasmídeo completamente sequenciado, com seus genomas e respectivas anotações. Detectou-se a ausência de um modelo de banco de dados para armazenar todas as informações, bem como se observou a necessidade de redistribuir essas informações no formato de banco de dados relacional. Este trabalho propõe um modelo de banco de dados relacional e um conjunto de ferramentas para análise, transposição dos dados no formato texto para o modelo de banco de dados relacional desenvolvido e estratégias de atualização. O modelo foi desenvolvido a partir da análise da especificação do GenBank e da observação das informações de organismos espalhados em mais de 2000 arquivos. Para o desenvolvimento das ferramentas, adotou-se a metodologia da prototipação, padrões de projetos, testes e análises de desempenho. Os resultados obtidos demonstram a possibilidade de armazenar todos os dados nos principais SGBD, com redução significativa da redundância nos dados e obtenção de alto desempenho nas quatro etapas do processo: 1) sincronização dos arquivos de texto em um repositório local a partir do servidor de arquivos do NCBI; 2) análise dos arquivos e interpretação dos campos; 3) carga dos dados analisados no banco de dados e; 4) aderência do modelo desenvolvido com a especificação e desempenho observado nas consultas feitas. Esta dissertação contribui para um novo modelo de organização, acesso e distribuição das informações do NCBI GenBank.

Palavras-chave: GenBank. Esquema de dados. Genomas sequenciados, anotação e ferramentas de ETL.

ABSTRACT

The NCBI GenBank is the main primary database of sequences and annotations of genomes, it has centralized the research results of DNA sequencing and RNA. This information is provided in several ways, especially in text files in GenBank format. Each chromosome or plasmid of organisms completely sequenced the Domain Bacteria and Archaea have a file available in the NCBI file servers. These files contain the genomic sequence and its annotations of organism. We observed the absence of a model database to store all the information and the need to redistribute this information in the form of relational database. This paper proposes a model of relational database and a set of tools for analysis and load textual data file for the model of relational database developed and upgrade strategies. The model was developed from the analysis of specification of the GenBank and the observation of information organism organized in more than 2000 files. The development of the tools used the methodology of prototyping, design patterns, testing and performance analysis. The results demonstrate the ability to store all data in the main DBMS, with significant reduction of redundancy in the data and to obtain higher performance in the four stages of the process: 1) synchronization of text files into a local repository from the file server of NCBI, 2) analysis and interpretation of the archives of the fields, 3) load the data analyzed in the database and 4) adherence of the model developed with the specification and observed performance in consultations. This paper contributes to a new model of organization, access and distribution of information from NCBI GenBank.

Keyword: GenBank. Database model, genomic sequence, annotations, ETL tools

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

GB	Gigabytes
pb	pares de bases
mpb	mega pares de bases (1mpb = 1.000.000 pb)
®	marca registrada

LISTA DE SIGLAS

ACID - Acrônimo para Atomicidade, Consistência, Isolamento e Durabilidade. Propriedades que devem ser observadas em uma transação de banco de dados.

ANSI - American National Standards Institute

API - Interface de Programação de Aplicações (Application Programming Interface, do termo inglês).

ASCII - acrônimo para American Standard Code for Information Interchange

CEP - Código de Enderessamento Postal

DDBJ - Data Bank of Japan

DER - Diagrama Entidade Relacionamento

DNA - ácido desoxirribonucléico

DSD/DED - Diagrama de Estrutura de Dados (DSD – do equivalente em inglês – Data Structure Diagram), poucos autores realizaram a adequação da sigla para DED.

EBI - Instituto de Bioinformática Europeu (European Bioinformatics Institute)

EMBL - European Molecular Biology Laboratory

ETL - Extração, Transformação e Carga (Extract, Transform and Load, em inglês)

FASTA - Formato utilizado para armazenar sequências de bases e de aminoácidos em arquivo texto

FN - Forma Normal

FTP - Protocolo de Transferência de Arquivo (File Transfer Protocol, em inglês)

GenBank - Banco de dados público do National Center for Biological Information, do Instituto de Saúde dos Estados Unidos da America.

GBFF GenBank Flat File – sigla adotada em grande parte das publicações consultadas e empregada nesse trabalho para diferenciar os arquivos textos do banco de dados.

INSDC - International Nucleotide Sequence Database Collaboration

IO ou I/O - Entrada e Saída (Input/Output, em inglês), pode ser empregado para dispositivos ou processos executados pelos programas de

computador.

ISS - Organização Internacional para Padronização (acrônimo para a versão em inglês)

JCP - Java Community Process

JDBC - Java Database Connectivity

JSR - Java Specification Requests

KDD - *Knowledge Discovery in Databases*, termo em inglês para Descoberta de Conhecimento em Banco de Dados.

NCBI - National Center for Biotechnology Information

NIH - National Institutes of Health

NLM - National Library of Medicine

ODBC - Open Data Base Connectivity

RAM - Memória de acesso aleatório (Random Access Memory, em inglês)

RNA - ácido ribonucleico

SGBD - Sistemas de Gerenciamento de Banco de Dados (Database Manager Systems – DBMS)

SQL - Structured Query Language

UF - Unidade Federativa

UFPR - Universidade Federal do Paraná

UML - Unified Modeling Language

SUMÁRIO

1 INTRODUÇÃO

As publicações do *National Center for Biotechnology Information* (NCBI) referentes aos bancos de dados de genomas revelam que, na última década, houve um crescimento exponencial na quantidade de genomas completamente sequenciados. Em 1999, havia disponíveis apenas 22 genomas completos de organismos (BENSON et al., 2000); em 2003, já eram 130 (BENSON et al., 2004); em 2007, somavam 570 (BENSON et al., 2007). Em maio de 2010, o NCBI comemora os mil genomas de procariotos completamente sequenciados disponíveis (FIGURA 1).

FIGURA 1 - FIGURA COMEMORATIVA DO SÍTIO DO NCBI
FONTE: NCBI (http://www.ncbi.nlm.nih.gov/, 2010)

As informações dos genomas disponíveis nos principais bancos de dados públicos são em geral postas à disposição em arquivos textos que seguem formatos especializados. Existem vários formatos de arquivos para os dados biológicos, mas os mais populares são o formato fasta para sequências de nucleotídeos e aminoácidos e o formato GenBank (GBFF) para sequências e as respectivas anotações.

Apesar da simplicidade dos formatos, a pesquisa e a localização das informações nos arquivos demandam o uso de programas especializados como o Artemis, BioEdit, Clustal e outros. Essas aplicações oferecem diversas ferramentas que auxiliam o usuário na análise das informações de um organismo, desde que ele esteja contido em um único arquivo de dados.

Ainda que muito úteis, as aplicações apresentam limitações no que se refere ao processo de comparação de informações contidas em arquivos diferentes. Já em análises cuja comparação de uma determinada informação gênica em relação ao conjunto de genomas disponíveis é necessária, ou de grupos específicos de genomas, não são aplicáveis.

Entre outras, as buscas que estávamos realizando manualmente e que motivaram a reunião de todos os arquivos foram: identificação dos organismos que possuem um ou vários dos genes nif; montagem da tabela dinâmica contendo nas colunas o nomes dos possíveis genes *housekeeping* e nas linhas os respectivos organismos; recuperação das sequências utilizando as anotações como referências entre outras.

O uso de Sistemas Gerenciadores de Banco de Dados (SGBD) na área da bioinformática possibilita aos pesquisadores ter acesso a recursos desenvolvidos, sobretudo para gerenciar grandes volumes de dados, abstraír a representação e organização da informação, usar técnicas de ordenação e pesquisa, assegurar a integridade da informação e reduzir o tempo na análise e cruzamento dos dados. Os requisitos para uso dos SGBD na área da bioinformática são:

- um modelo simples e completo para representar informações disponíveis nos formatos já existentes;
- um programa que faça conversão dos formatos textos para o modelo do banco de dados;
- uma aplicação para realizar as pesquisas;
- um processo de manutenção e atualização dos dados.

Na literatura, encontram-se diversos trabalhos com propostas para cada um dos componentes citados. No entanto, verifica-se a ausência de algum que proponha a reprodução das informações constantes em GBFF de genomas completos de Bactérias e Acheae em uma estrutura normalizada de um Banco de Dados Relacional.

Este trabalho propõe uma modelagem de banco de dados para as informações disponíveis em arquivos textos no formato GenBank e um conjunto de ferramentas para facilitar seu uso, a fim de contribuir para a sistematização da informação genômica.

O desenvolvimento das ferramentas em estudo possibilita explorar novas soluções e estratégias, com destaque a contribuições nas áreas de:

- banco de dados, com o desenvolvimento de um modelo relacional com baixa redundância e aderente a especificação do GenBank;
- analisadores (parser) de arquivos textos no formato Genbank;
- carga de dados em banco de dados;

- atualização e sincronização das informações.

Ainda que os desenvolvimentos propostos sejam aplicáveis de forma genérica a qualquer tipo de dado depositado no NCBI no formato *Genbank*, o presente trabalho tem como escopo apenas os genomas completos de Bactérias e Archea.

Vale mencionar que este trabalho apresenta um conjunto de recursos e metodologias que possibilita à comunidade científica ter um banco relacional com todos os dados disponíveis no NCBI GenBank dos genomas completamente sequenciados dos domínios Bactéria e Archaea em aproximadamente quarenta minutos.

1.1 OBJETIVO GERAL

Estruturar um banco de dados relacional para informações dos genomas completamente sequenciados de organismos procariotos, postos em disponibilidade pelo NCBI no servidor de arquivos do GenBank, para permitir a mineração dos dados biológicos.

1.2 OBJETIVOS ESPECÍFICOS

Para que o objetivo geral proposto seja atingido, identificam-se os seguintes objetivos específicos:

- analisar a especificação e a estrutura dos arquivos GenBank;
- avaliar a obtenção e a sincronização dos arquivos disponíveis no GenBank com a réplica local;
- desenvolver um modelo relacional de banco de dados para as informações do GenBank;
- desenvolver uma ferramenta de análise e interpretação do formato de arquivo do GenBank;
- desenvolver uma ferramenta de carga (transposição) das informações em formato texto para um Sistema de Gerenciamento de Banco de Dados, e
- propor estratégias de distribuição das informações do NCBI GenBank.

1.3 JUSTIFICATIVA E RELEVÂNCIA DO TRABALHO

Os dados dos genomas estão distribuídos em milhares de arquivos. Entretanto, as ferramentas de busca fornecidas pelo NCBI, apesar de muito boas, limitam a busca por um motivo (sequência, taxonomia etc). O cruzamento desses motivos na pesquisa é extremamente desejável e necessário para possibilitar esse tipo de estudo.

Atualmente, pesquisadores precisam desenvolver bancos locais e toda a tecnologia de carga para, então, realizar as buscas. Essas limitações justificam os mais de 1400 sistemas de bancos de dados biológicos existentes, sendo que, em boa parte, há dados provenientes do NCBI GenBank. Desenvolver um banco de dados que possibilite aos pesquisadores esse recurso em menor tempo possibilitará que se reduza o desenvolvimento de novos bancos com informações parciais do GenBank e que se amplie a possibilidade de integração com outras bases de dados. O desenvolvimento de um sistema de banco de dados biológico pode ser estimado em seis meses, enquanto o processo de carga de todos os dados do GenBank pode ser estimado em dias. Possibilitar o acesso, a integração dos dados e reduzir o tempo para obtenção do banco de dados é essencial para a comunidade científica.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

O presente trabalho está organizado conforme as recomendações da Universidade Federal do Paraná (UFPR) para dissertações e teses, em cinco seções:

- Introdução, contendo os objetivos, organização e notações adotadas nessa dissertação (capítulo 1);
- revisão dos conceitos aplicados e levantamento do conhecimento sobre o tema (capítulo 2);
- descrição da metodologia para desenvolvimento do trabalho (capítulo 3);
- resultados obtidos (capítulo 4),
- discussão (capítulo 5) e
- conclusão (capítulo 6);
- recomendações de futuros trabalhos (capítulo 7).

Neste trabalho, optou-se em acrescentar a seção “Considerações Iniciais” em cada capítulo, com uma breve descrição dos assuntos e objetos do capítulo.

1.5 NOTAÇÃO ADOTADA PARA TERMOS TÉCNICOS

Na redação e editoração desta dissertação, adotou-se a padronização recomendada pelas normas e recomendações da UFPR, conforme transcrita a seguir:

> 2.10 USO DE ASPAS, ITÁLICO E NEGRITO
>
> O uso de aspas, itálico e negrito deve ser estabelecido antes de iniciar a digitação ou a datilografia do documento e deve ser coerente e uniforme, evitando-se o uso alternado de diferentes tipos de destaque para o mesmo tipo de expressão.
>
> Nos casos seguintes, apenas o itálico pode ser utilizado:
> a) palavras e frases em língua estrangeira e expressões em latim;
> b) nomes de espécies, em botânica, zoologia e paleontologia;
> c) títulos de documentos (livros, revistas, artigos e outros) citados nos texto.
>
> Recomenda-se apenas o uso do negrito para letras ou palavras que mereçam ênfase, quando não for possível dar esse realce pela redação.
> Tanto o itálico como o negrito podem ser utilizados em títulos de documentos na lista de referências.
>
> As aspas são sinais de pontuação empregados:
> a) no início e no final de uma citação que não exceda cinco linhas;
> b) em citações textuais de rodapé;
> c) em expressões do idioma vernáculo usuais apenas no meio profissional;
> d) em termos relativizados, isto é, utilizados com significado diferente, como apelidos e gíria, ou ainda com sentido irônico;
> e) em definições conceituais de termos.*(IPARDES, 2001, p. 52-53).*

Adotou-se o itálico para termos em língua estrangeira, incluindo jargões da área da informática que não possuem traduções conhecidas ou termos não constantes do Dicionário de Informática. Optou-se por negrito para ênfases e realces, principalmente, em textos citados. Os termos técnicos de informática previstos no referido dicionário estão com as respectivas citações.

2 REVISÃO DA LITERATURA

2.1 CONSIDERAÇÕES INICIAIS

Por meio deste capítulo, apresenta-se uma visão geral do contexto biológico, das informações gênicas disponíveis no NCBI *GenBank*, dos aspectos gerais da área de banco de dados aplicados à sistematização das informações, da abordagem empregada pelos bancos de dados biológicos, dos processos de “Extração, Transformação e Carga de Dados” (ETL – *Extract, Transform and Load,* do inglês), como parte do processo de mineração de dados e dos trabalhos correlatos em que se desenvolveram propostas semelhantes às abordadas nesta dissertação.

Inicia-se a revisão pela visão da área de bioinformática em relação aos bancos de dados.

2.2 BANCO DE DADOS

2.2.1 Conceitos preliminares

Atualmente, o trabalho de conceituação de banco de dados aponta algumas limitações em função dos autores descreverem o conceito com base em modelos matemáticos ou pelo viés da aplicação de banco de dados em sistemas de informação.

Antes de apresentar o conceito de banco de dados, serão revisados os conceitos essenciais da área de informática, relacionados a banco de dados, de acordo com a abordagem de Yong (1983):

a) BIT – é a menor parte da informação, manipulável pelo computador. Em geral, associam-se os valores de zero e 1 com os conceitos de falso e verdadeiro, respectivamente. Neste ponto, surge a primeira questão importante: o computador não consegue armazenar ou realizar operações em apenas um bit isolado. Todas as operações computacionais são realizadas no nível de *byte;*
b) *Byte* – é o conjunto de oito bits e que permite a representação de 256 (zero a 255) possibilidades de informações distintas. Observa-se que, para o computador, um *byte* sempre será um *byte*; ou seja, um número que não tem

nenhum significado próprio associado a si. Os programadores recorrem a recursos das linguagens de programação para associar simbologia e algum sentido aos valores armazenados no *byte*; por exemplo, o valor 65 pode ser uma informação numérica ou o símbolo ‘A’ na tabela ASCII (acrônimo para *American Standard Code for Information Interchange*);

c) Tipo de dado – em termos de computação eletrônica, o conceito foi desenvolvido para que as linguagens de programação pudessem “prever” o conjunto dos estados de um dado, ou seja, valores que o dado pode ter em um determinado momento do processamento, domínio do dado ou tipo do dado. A definição de tipo permitiu definir o comportamento e o tamanho do conjunto de operações possíveis a cada tipo de dado. Este recurso resolve o problema de ambiguidade no uso de alguns símbolos. Por exemplo, o símbolo “+” quando presente em uma expressão, cujos operadores são numéricos, representa a operação matemática de adição. No entanto, se uma das variáveis for do tipo texto (String), então a operação deve ser de concatenação. Ao mesmo tempo, o tipo de dado permitiu que as linguagens gerenciassem melhor o uso de memória, uma vez que estabelecia o domínio – o conjunto de valores possíveis de serem representados pela variável – e, consequentemente, evitava a necessidade de realocar as operações de atribuição. Observa-se que este conceito foi estendido e é amplamente usado nos gerenciadores de banco de dados, planilhas etc.;
d) Variável – é um nome associado a um endereço de memória do computador, cuja área indicada por esse endereço serve para preservar um dado. Uma variável possui apenas um estado – um valor ou um dado – em um determinado momento. As variáveis ou os respectivos dados estão sempre associados a um tipo de dado (VILLAS e VILLASBOAS, 1988);
e) Arquivo – é um depósito de informações codificadas ou não armazenadas no sistema de arquivos de um computador ou de uma unidade externa (mais sobre arquivos e formato de arquivos podem ser vistos no item 2.3);
f) Item de dado ou campo – Para Yong (1983), corresponde a um conjunto de *bytes*. Constitui uma unidade básica representativa de informação, identificável e possível de definir tamanho e formato. Um campo é uma variável, uma parte de uma estrutura maior, que consegue sozinha armazenar um dado. Também pode ser denominado de atributo ou coluna;

g) Registro (tupla) – um registro consiste em um conjunto de itens de dados, reunidos de forma a caracterizar uma ocorrência de um conjunto de atributos, de uma determinada entidade (YONG, 1983). Registro também é visto como uma estrutura de dados heterogênea – possibilidade de campos de tipos de dados diferentes – capaz de representar um conjunto de dados de uma mesma entidade;
h) Tabela (relações ou entidates) – inicialmente, pode-se afirmar que tabela é um conjunto de registros. Na prática, deve-se ver a tabela como um elemento bidimensional, em cujas colunas tem-se os campos e nas linhas os registros;
i) Entidade – é um modelo computacional que representa uma pessoa, um objeto ou algo real que precise ser representado ou armazenado e processado nos sistemas computacionais (SILBERSCHATZ et al., 2006). O termo entidade pode ter muitos conceitos envolvidos, dependendo da literatura e do contexto em que seja empregado. Neste trabalho, adotou-se o conceito de Entidade obtido no contexto da “modelagem de entidade/relacionamento”;
j) Relacionamento – Chama-se de relacionamento a associação que existe entre as informações de duas ou mais tabelas. A construção da ligação entre as tabelas pressupõe a existência de um atributo comum;
k) Índice – Os índices são empregados para manter a ordem dos registros de uma tabela segundo um ou mais de seus atributos. Os atributos que fazem parte de um índice são chamados de chave. Geralmente, os índices são mantidos em arquivos separados das tabelas, em estruturas projetadas para o armazenamento de uma chave e o respectivo endereço físico do registro. Existem diversos tipos de índices. Os mais comuns são B-tree (árvores B), hash e bitmaps;
l) Chave – é um atributo (campo) de uma tabela que faz parte de um índice. Quando uma chave é formada por dois ou mais atributos, denomina-se de chave-composta. Alguns atributos possuem a característica natural de identificar unicamente o registro, como o número do Cadastro de Pessoa Física (CPF) em uma tabela de pessoas. Nesses casos, esses atributos são chamados de chaves candidatas. Por diversas razões, as tabelas precisam ter um elemento que identifique e preserve a identidade e unicidade de um registro; esse elemento comumente é chamado de id ou código e é usado para ser a chave primária dessa tabela. Quando o id ou o código de uma tabela é usado em uma segunda

tabela para fins de relacionamento, chama-se o atributo da segunda tabela de chave estrangeira (*foreign key*);

m) Cardinalidade – descreve o grau de relacionamento entre duas tabelas, basicamente podem ser 1x1, 1xN ou NxN, onde 1x1 significa que para cada registro da primeira tabela pode ter zero ou um registro na segunda tabela. A segunda relação estabelece que para cada registro da primeira tabela pode existir zero, um ou muitos registros na segunda tabela. A terceira relação NxN pode aparecer na fase de análise; porém, no modelo, é convertida para duas relações do tipo 1xN com uma terceira tabela de ligação;

n) Modelos de dados – Para Silberschatz et al. (2006) a estrutura de banco de dados é apoiada pelo modelo de dados: uma coleção de ferramentas conceituais com o objetivo descrever dados e suas relações, semântica de dados e restrições de consistência. O autor acrescenta:

> Os modelos de dados podem ser agrupados em quatro categorias:
> 1. Modelo relacional – o modelo relacional usa uma coleção de tabelas para representar os dados e as relações entre eles. Cada tabela possui diversas colunas e cada coluna possui um nome único.
>
> 2. Modelo entidade/relacionamento – o modelo de entidade/relacionamento (E-R) é baseado em uma percepção de um mundo real que consiste em uma coleção de objetos básicos, chamados entidades e as relações entre esses objetos.
>
> 3. Modelo de dados baseado em objeto – Pode ser visto como uma extensão ao modelo E-R com noções de encapsulamento, métodos (funções) e identidade de objeto.
>
> 4. Modelo de dados semi-estruturado – permite a especificação dos dados em que itens de dados individuais do mesmo tipo possam ter diferentes conjuntos de atributos. Isso é oposto dos modelos de dados mencionados anteriormente, em que todos os itens de dados de um determinado tipo precisam ter o mesmo conjunto de atributos. (SILBERSCHATZ et al., 2006, p. 5).

O modelo relacional é uma teoria matemática desenvolvida por Edgard Frank Codd, matemático e pesquisador da IBM. Atualmente, quando se expressa o termo “banco de dados”, praticamente equivale a dizer banco de dados relacional.

Sistemas de Gerenciamento de Banco de Dados – Correspondem a um conjunto de programas para administrar o banco de dados (YONG, 1983). Já para Silberschatz et al. (2006), consiste em uma coleção de dados inter-relacionados e um conjunto de programas para acessá-los.

De acordo com Silberschatz et al. (2006, p. 6),

> [...] um sistema de banco de dados fornece uma linguagem de definição de dados (DDL) para especificar o esquema de banco de dados e uma linguagem de manipulação de dados (DML) para expressar as consultas e atualizações do banco de dados.

Na prática, a maioria dos SGBD atualmente adota a linguagem SQL (acrônimo de "*Structured Query Language*"), baseada na padronização realizada pelo ANSI e em um conjunto de extensões do próprio fabricante.

A maioria dos SGBD disponíveis no mercado e amplamente empregados são Sistemas de Banco de Dados Relacionais, ou seja, organizam os dados e as relações em tabelas. Por essa razão, nesse trabalho, utilizar-se-á somente SGBD – como é usual –, porém referir-se-á sempre aos SGBD Relacionais.

Transação – Silberschatz et al. (2006, p. 71) introduzem o conceito de transação como "uma sequência de instruções de consulta e/ou atualizações". Posteriormente, os autores refinam o conceito para "é uma unidade de execução do programa que acessa e possivelmente atualiza vários itens de dados" (SILBERSCHATZ et al., 2006, p. 409) e acrescentam:

> Para garantir a integridade dos dados, é necessário que o sistema de banco de dados mantenha as seguintes propriedades das transações:
>
> **Atomicidade:** Todas as operações da transação são refletidas corretamente no banco de dados, ou nenhuma delas.
>
> **Consistência:** A execução de uma transação isolada (ou seja, sem qualquer outra transação executando simultaneamente) preserva a consistência do banco de dados.
>
> **Isolamento:** Embora várias transações possam ser executadas simultaneamente, o sistema garante que para cada par de transações Ti e Tj, parece para Ti que ou Tj terminou a execução antes que Ti começasse ou Tj iniciou a execução depois que Tj terminou. Assim, cada transação não está ciente das outras transações executando simultaneamente no sistema.
>
> **Durabilidade:** Depois que uma transação for completada com sucesso, as mudanças que ela fez ao banco de dados persistem, mesmo que existam falhas no sistema.
>
> Essas propriedades normalmente são conhecidas como propriedades ACID; o acrônimo é derivado da primeira letra de cada uma das quatro propriedades. (SILBERSCHATZ et al., 2006, p. 409).

As propriedades transacionais são requisitos essenciais para a maior parte dos sistemas de informação da atualidade, praticamente assumindo como requisito para qualquer banco de dados.

2.2.2 Conceito de banco de dados

Date afirma que “um banco de dados é uma coleção de dados persistentes utilizada pelos sistemas de aplicação de uma empresa.”. A afirmação de Date não discrimina “um banco de dados” do conjunto de arquivos presentes em sistema de arquivos. Ainda, conceitua banco de dados pelo aspecto de “durabilidade” da informação. Conforme o autor,

> [...] dizemos que os dados do banco de dados ‘persistem’ porque, uma vez aceitos pelo SGBD para entrada inicial no banco de dados, eles podem ser removidos apenas subsequentemente do banco de dados por alguma solicitação explicita ao SGBD, não como um efeito colateral de (por exemplo) conclusão da execução de um programa. (DATE, 2000, p. 9).

Yong (1983, p.18) descreve o conceito de banco de dados como “Uma Base de Dados é constituída de um conjunto de arquivos relacionados entre si”. Yong, cinco anos antes de Date, acrescentou o conceito de “relação” ao propor a definição de banco de dados, porém não resolveu o problema de definir um banco de dados como o conjunto de arquivos de um mesmo tipo (como imagens ou documentos do Word), independentenebte da informação em si.

Para Martin (1975, p. 39), banco de dados é definido sendo uma “Coleção de dados inter-relacionados, armazenados juntos e com redundância controlada para servir a uma ou mais aplicações”. Nesse contexto, Martin amplia um pouco sua definição e possibilita, porém, interpretações inadequadas para o conceito de banco de dados; afinal, “armazenados juntos” pode ser na mesma pasta, em um mesmo dispositivo (CD-ROM, HD etc) ou em um mesmo arquivo. Dados inter-relacionados podem ser compreendidos como uma relação interna maior que quando se afirma apenas relação, porém não exclui que um conjunto de arquivos que possui uma mesma estrutura não seja compreendido como um banco de dados. Já o conceito de redundância controlada é, a nosso ver, um dos aspectos que diferencia um banco de dados das demais organizações de informação.

Para Medeiros (2007, p. 14),

> Date, afirma [...] Grassmann e Tremblay dizem que os atributos ou itens que descrevem entidades do mundo real, tal como uma pessoa, coisa ou evento, são estruturadas em registros que, por sua vez, compõem os arquivos. Se o conjunto destes é aproveitado em programas ou aplicações em certa área de uma empresa, então, a esse conjunto denomina-se de

> banco de dados. [...[O'Brien afirma que banco de dados é um conjunto integrado de elementos de dados relacionados logicamente.

A discussão proposta por Medeiros para compreender o conceito de banco de dados associa os conceitos de entidade, registro, aplicações que acessam os dados entre outros. Esses conceitos são de fato muito abordados em banco de dados e encontram-se na "moda", porém, não definem banco de dados.

Butzen e Forbes (1997, p.8-15) propõem que o conceito de banco de dados deva ser apresentado como resposta à pergunta "o que é banco de dados" e a respectiva resposta "*a database is an orderly body of data, and the software that maintains*", o que, de acordo com os autores, leva às questões: o que são dados e o que significa manter a consistência dos dados? Em resposta à primeira questão, os autores descrevem que "*the data is difficult to define exactly. We offer this definition: a datum is a symbol that describes an aspect of an entity or event in the real world*.", e a resposta para a segunda questão foi descrita como um conjunto de ações: organização dos dados, inserção, recuperação, exclusão, atualização e preservação da integridade. O conceito de dados se aproxima bastante dos aspectos apresentados por Medeiros e acrescenta alguns dos aspectos definido originalmente por Codd, entre eles a manutenção da informação e integridade das relações.

Para Silberschatz et al. (2006, p. 1), o banco de dados, na realidade, é o conjunto de dados; por sua vez, "um sistema de gerenciamento de banco de dados (DBMS) é uma coleção de dados inter-relacionados e um conjunto de programas para acessar esses dados". Nesse contexto, retorna-se ao princípio de que qualquer conjunto de dados pode ser denominado "banco de dados".

Butzen e Forbes (1997, p. 56-59) acrescentam elementos na discussão sobre a implementação de banco de dados relacionais:

> *The relational model gives us a proven, complete model for our databases. However, it is only a model - a theoretical construct.*
>
> *Translating this model into bits that you can install and run on your linux system is challenging.*
>
> *Some complications arise from the fact that aspects of the relational model are dificult to implement. Other complications arise from the fact that some early implementations of the relational model cut corners in some ways. These features have affected the development of relational databases, and may well have stunted them. Some person argue that SQL itself falls into this category; certainly, some*

> *features of SQL (such as the fact that it permits a table to hold duplicate rows) are definitely nom-relational.*
>
> *Other complications arise from the fact that people have an enormous investiment in existing technology, and they simply cannot throw it away, regardless of how good the alternative is: if you have bought a lemon and can't afford to junk it, you just have to drive it.*
>
> *Given that there is a gap between the ideal of the relational model and the reality of the implemented database, how can we judge whether a database that claims to be relational is, in fact, a relational database? In 1985, E. J. Codd published 12 criteria with which you can judge whether a given database system is truly relational. No database management system fulfills all 12 criteria; but the criteria do given us a rule of thumb with wich we can judge the degree to wich a database management fulfills the relational model. (BUTZEN & FORBES, 1997, p.55-56).*

Como se observou, os autores descrevem a existência de um abismo entre o modelo relacional e a implementação do modelo nos atuais bancos de dados e consideram que as regras definidas por Codd são elementos a serem considerados na avaliação do grau de implementação do modelo relacional por um sistema de gerenciamento de banco de dados. As 12 regras definidas por Codd, segundo os autores, são:

> *1. the database stores all data within tuples;*
> *2. every datum can be accessed through the combination of the name of its relation, the name of its attribute, and the value of its tuple's primary key;*
> *3. NULL values are implemented systematically;*
> *4. the database's catalogue is itself stored within one or more relations that can be read by authorized users;*
> *5. the system implements a query language with which a user can perform the following tasks:*
> – *define a relation*
> – *define views*
> – *manipulate data*
> – *set constraints to enforce database integrity*
> – *grant and delete permission to view or manipulate a relation*
> – *bundle manipulations of the database as a transaction*
> *6. the system must be able to update through a view;*
> *7. the system must be able to insert, update, and delete sets of tuples;*
> *8. the programs with which the database is manipulated are independent of how the database is physically stored or accessed;*
> *9. the programs with which the database is manipulated are independent of how the database is logically organized internally;*
> *10. rules that articulate semantic-level integrity should be describable within the database management system's query*

> *language, be storable within the database system's catalogues, and be enforceable by the database management system itself;*
> *11. a relational database should operate exactly the same whether housed on a single machine or distributed over a network;*
> *12. no one can use a low-level language to subvert database's integrity rules. (BUTZEN & FORBES, 1997, p.57-59).*

Em função do objeto desta dissertação, o conceito de banco de dados será considerado como **o conjunto de informações relacionadas a um tema, organizado e estruturado conforme o modelo relacional e, preferencialmente, gerenciado por um SGBD que assegure as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade) durante as transações**.

2.3 MODELAGEM DE BANCO DE DADOS

2.3.1 Conceito e paradigmas

O modelo de banco de dados ou modelo de dados é uma descrição dos tipos de informações que estão armazenadas em um banco de dados e, normalmente, o modelo está associado a um sistema de banco de dados.

A linguagem de modelagem de dados pode ser gráfica ou textual. Observa-se que esses recursos estão gradativamente caindo em desuso, em consequência das ferramentas disponíveis para modelagem de sistemas.

> [...] praticamente todo o uso dos bancos de dados ocorre de dentro dos programas de aplicação. De modo correspondente, quase toda a interação do usuário com o banco de dados é indireta, por meio de programas de aplicações. (SILBERSCHATZ et. al., 2006, p.207).

Essa consideração coloca o banco de dados como um componente a serviço de uma ou mais aplicações de modo que, historicamente, a origem dos bancos de dados se encontra associada ao desenvolvimento de um sistema. Portanto, deve-se considerar que o projeto do modelo de banco de dados terá como referencial a análise e especificação do sistema de informações que o banco de dados estará associado.

Existem diversas metodologias para análise de sistemas e diversas ferramentas para modelagem do sistema de informação. Basicamente, estas metodologias são dividas em dois paradigmas: estruturado e orientado a objetos.

No paradigma estruturado,

> [...] à medida que se monta o diagrama de fluxo de dados, identifica-se locais onde os dados foram mantidos de uma transação para a seguinte ou armazenamento permanentemente, porque descreviam um aspecto do mundo fora do sistema. (GANE E SARSON, 1983, p.19).

Pode-se considerar, portanto, que o projeto do modelo de banco de dados pode ser derivado das informações descritas no dicionário de dados e no fluxo de informação. Para mais detalhes sobre a metodologia estruturada de análise de sistemas, recomenda-se a leitura do livro *Análise estruturada de sistemas*. (GANE e SARSON, 1983).

No paradigma orientado a objetos, de acordo com Pender (2004, p. 83), “o diagrama de classe (UML) se encontra no centro do processo de modelagem de objetos. Ele é o diagrama principal para capturar todas as regras que governam a definição e o uso de objetos”, consequentemente, o modelo de dados deve ser capaz de manter todos os objetos, com seus atributos e relações, dando viabilidade à restauração de um novo objeto contendo os estados originais. Para outras informações sobre a metodologia de análise orientada a objetos, recomenda-se a leitura dos trabalhos de Pender (2004), *UML Guia do usuário,* entre outros.

2.3.2 Metodologia para desenvolvimento da modelagem de banco de dados

Para Silberschatz et al.,

> [...] o projetista de banco de dados precisa interagir com os usuários da aplicação para entender as necessidades da aplicação, representá-las de uma maneira de alto nível que possa ser entendida pelos usuários, e depois, traduzir as necessidades para níveis mais baixos do projeto (SILBERSCHATZ et al., 2006, p. 133).

Ainda, conforme entende o autor, são fases do projeto de banco de dados:

> A fase inicial do projeto de banco de dados é caracterizar completamente as necessidades de dados dos prováveis usuários do banco de dados. O projetista de banco de dados precisa interagir extensivamente com especialistas de domínio e usuários para realizar essa tarefa.[...]
>
> Em seguida, o projetista escolhe um modelo de dados e, aplicando os conceitos do modelo de dados escolhido, traduz essas necessidades para um esquema conceitual do banco de dados. [...] O **modelo entidade-**

> **relacionamento normalmente é usado para representar o projeto conceitual**.[grifo nosso] [...] Em geral, a **fase do projeto conceitual resulta na criação de um diagrama entidade-relacionamento** que fornece uma representação gráfica do esquema.
>
> Um esquema conceitual completamente desenvolvido também indica as necessidades funcionais da empresa. [...] Nessa fase do projeto conceitual, **o projetista pode revisar o esquema para garantir que ele atenda às necessidades funcionais** [grifo nosso].
>
> O processo de transcrição de um modelo de dados abstrato para a implementação do banco de dados ocorre nas duas fases finais do projeto.
>
> Na fase de **projeto lógico** [grifo do autor], o projetista mapeia o esquema conceitual de alto nível para o modelo de dados de implementação do sistema de banco de dados que será usado. **O modelo de dados de implementação normalmente é o modelo de dados relacional** [grifo nosso] [...].
>
> Finalmente, o projetista usa o esquema de banco de dados específico do sistema resultante na fase de **projeto físico** [grifo do autor], em que os recursos físicos do banco de dados são especificados. Esses recursos incluem a forma de organização de arquivos e as estruturas de armazenamentos internas(...). (SILBERSCHATZ et al., 2006, p. 133-134).

O modelo entidade-relacionamento (MER) citado, foi proposto por Peter Chen e é composto de entidades, atributos e relacionamentos. No texto original, Chen propõe uma representação gráfica em que os retângulos representam as entidades, os losangos para os relacionamentos, os círculos para os atributos e as linhas para conexões.

Baseado no modelo de Chen, vários autores propuseram modelos gráficos semelhantes, que aperfeiçoavam algum aspecto da notação original.

Nas décadas de 1970 e 1980, após a elaboração do modelo entidade relacionamento, desenvolvia-se o diagrama entidade-relacionamento (DER) e, depois, ampliava-se o detalhamento das informações com o uso do Diagrama de Estrutura de Dados (DSD, do correspondente em inglês). A lógica era aperfeiçoar o modelo de forma incremental e cada fase preservar o foco em um aspecto do processo. A transcrição do Diagrama de Entidade Relacionamento para o Diagrama de Estrutura de Dados denotava a evolução do modelo lógico do banco de dados para o modelo físico.

No final da década de 1980 e início da década de 1990, com a popularização dos ambientes gráficos, o emprego das ferramentas Case (do inglês *Computer-Aided Software Engineering*) e a busca por processos mais produtivos – menor custo e maior produtividade – começaram a surgir aplicações para automatizar os

processos. Concomitantemente, os sistemas de banco de dados começaram a crescer, tornando a adoção dos modelos gráficos da época impróprios para representar o tamanho e a complexidade.

Naquela época, surgiram diversas propostas, e entre elas se destacaram os trabalhos de Richard Barker[1] e James Martin[2]. Esses trabalhos, na prática, fundiram o modelo lógico e físico em um único diagrama, sendo a origem do atual diagrama de entidade-relacionamento, que nasceu da fusão do modelo anterior com o diagrama de estrutura de dados, acrescido de um conjunto de aperfeiçoamentos. Na ocasião, o modelo foi popularizado com as ferramentas ErWin®, Sybase PowerDesigner®, Data Architect®, entre outros.

Essencialmente, o diagrama entidade-relacionamento é formado por retângulos e linhas. Os retângulos representam as entidades (tuplas, relações ou tabelas) e as linhas as relações. Nos retângulos são descritos o nome da entidade, os atributos (incluindo o tipo e tamanho/precisão da informação) e as chaves primárias, estrangeiras e/ou outras. Existem variações nas representações gráficas dos elementos, na forma de identificação dos índices ou na representação dos símbolos que denotam as relações, porém os elementos, geralmente são os mesmos.

A FIGURA 2 apresenta um exemplo dos elementos gráficos presentes no DER, conforme Martin.

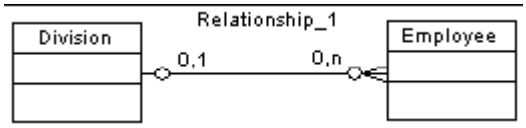

FIGURA 2 - EXEMPLO DOS ELEMENTOS DO DIAGRAMA ENTIDADE-RELACIONAMENTO

FONTE: SYBASE (2001, p. 76)

Uma etapa importante na elaboração da modelagem relacional é o processo de normalização. Para Yong (1983),

> De uma maneira genérica, uma relação pode conter atributos cujos domínios sejam também relações; dizemos então que a relação está na forma NÃO NORMALIZADA e os domínios são do tipo “não simples”.

[1] BARKER, Richard. ***CASE Method:*** *Tasks and Deliverables.* Wokingham, Inglaterra: Addison-Wesley, 1990.
[2] MARTIN, James. ***Information engineering.*** v. 1, 2 e 3. Prentice Hall, 1990.

> Para a devida utilização dos operadores da teoria de conjuntos é necessário que a relação não normalizada seja transformada numa forma onde os atributos só contenham domínios simples. A relação resultante, contendo atributos com domínios simples, ainda pode conter irregularidades devido a dependências funcionais não desejáveis. Tais irregularidades se manifestam principalmente no processo de manutenção (inclusão, atualização e exclusão de tuplas) das relações.
>
> O processo de transformação de relações não normalizadas às formas mais ótimas de expressão é conhecido como NORMALIZAÇÃO. (YONG, 1983, p.125).

Existem cinco níveis de formas normais (denotadas pelos símbolos 1FN, 2FN, 3FN, 4FN e 5FN), porém as três primeiras são as mais conhecidas e aplicadas, praticamente, de forma automática pelos profissionais de informática. Cabe observar que cada nível depende do nível anterior e é uma especialização do caso anterior.

As formas normais são “Uma relação R está na Primeira Forma Normal (1FN) se e somente se todos os domínios considerados contenham somente valores atômicos (isto é, valores que não são relações)” (YONG, 1983, p.127).

Ao adaptar-se o enunciado acima à linguagem adotada neste trabalho, tem-se que uma tabela R está na Primeira Forma Normal (1FN) se e somente se todos os campos – ou atributos – considerados contenham somente valores atômicos, ou seja, o conteúdo previsto para o campo não pode ser mais dividido em dois ou mais campos. Por exemplo, se projetarmos uma tabela para armazenar informações de alunos, então essa tabela poderá ser denominada de aluno e pode ter campos, tais como: matrícula, nome e endereço. A princípio, os campos *matrícula* e *nome* estão na forma atômica, não podem ou não precisam ser divididos em mais campos. O campo “endereço”, contudo, pode e deve ser divido em logradouro, número, complemento, bairro, CEP, cidade e UF. Afirma-se, portanto, que o campo é composto ou que possui um domínio composto, ou que é uma relação ou ainda, não se encontra na forma atômica.

> Uma relação R está na Segunda Forma Normal (2FN) se ela está na 1FN e cada atributo dependente é funcionalmente dependente do atributo chave e sem ser funcionamento dependente de qualquer subconjunto do atributo-chave (no caso de ser composto de diversos atributos). (YONG, 1983, p.129)

Para melhor compreensão da 2FN, pode-se considerar a tabela de pedidos apresentados a seguir:

TABELA 1 - EXEMPLO DE UMA TABELA DE PEDIDOS

num_pedido	cod_produto	Nome_produto	qdade	Valor_unit	Total
1010	23A	Bananas	12	0,05	0,60
1011	25	Maças	12	0,05	0,60
1013	23A	Bananas	24	0,05	1,20
1015	26	Morango	20	0,05	1,00

FONTE: o autor (2010). Os dados são fictícios.

O nome do produto (nome_produto) depende funcionalmente do código do produto (cod_produto) e não depende do número do pedido (num_pedido) que é a chave primária da tabela de *pedidos*. Não se encontra na 2FN, portanto. A consequência de não atender essa regra fica evidente nas operações de manutenção. Sempre que o nome do produto sofrer alterações, essas modificações terão de ser propagadas para muitos registros e, em caso de falha, poderá ocorrer a existência de um mesmo código de produto com duas ou mais descrições (nome do produto) diferentes. “Uma relação R está na Terceira Forma Normal (3FN) se está em 2FN e os atributos dependentes, não são dependentes transitivos do atributo-chave.” (YONG, 1983, p.130).

Após a correção da tabela anterior para atender a 2FN, obtém-se a próxima tabela, em que se pode verificar no mesmo conjunto se atende à regra da 3FN.

TABELA 2 - EXEMPLO DA TABELA DE PEDIDOS NA 2FN

num_pedido	cod_produto	qdade	valor_unit	total
1010	23ª	12	0,05	0,60
1011	25	12	0,05	0,60
1013	23ª	24	0,05	1,20
1015	26	20	0,05	1,00

FONTE: o autor (2010). Os dados são fictícios.

A Tabela 3 não atende a 3FN porque o campo total é obtido pelo produto do conteúdo do campo *qdade* com o conteúdo do campo *valor_unit*, ou seja, a coluna “total” depende dos atributos “qdade” e “valor_unit”.

> Uma relação normalizada R está na Quarta Forma Normal, se e somente se, quando existe uma dependência do tipo “múltiplo valores” (digamos de B

> para A) então todos os demais atributos serão dependentes funcionais em A. (YONG, 1983, p.132).

Em geral, as tabelas de ligações das relações NxN ternárias não atendem a 4FN, de modo que devem ser dividias em duas ou três tabelas de ligação. Por exemplo, considere uma tabela que descreve a associação de projeto com equipe e funcionário. Essa tabela deverá ter os atributos: cod_projeto, cod_equipe e cod_funcionario. Para atender a 4FN, essa relação deve ser expressa com duas outras relações, a primeira contendo cod_projeto e cod_equipe e a segunda contendo cod_projeto e cod_funcionario. Observe que a 4FN assegura as relações projeto-equipe e projeto-funcionário, porém não resolve a relação equipe-funcionário.

Conforme Date, (2000), “Uma tabela está na 5FN se e somente se estiver na 4FN e um relacionamento triplo não puder ser decomposto em relacionamentos binários sem geração de informação incorreta.”

A 5FN é um caso especial de relacionamento de chaves primárias de três tabelas, em que nem sempre é possível a decomposição correta. Ou seja, a divisão em duas ou três de tabelas de associação não consegue representar todas as combinações possíveis.

Existe uma variação das formas normais, conhecida como BCNF, que segue seu enunciado: “Uma relação R está na BCNF se e somente se R está na 3FN e nenhum atributo possua dependência transitiva em relação à chave primária (DATE, 2000).

Para Silberschatz et al. (2006, p. 181), “Uma das formas normais mais desejáveis que se pode obter é a Boyce-Codd Normal Form (BCNF). Ela elimina toda redundância que pode ser descoberta com base nas dependências funcionais”. Os autores afirmam, porém, que o método pode acarretar alguns problemas. Conforme entendem, “veremos que a decomposição em BCNF pode impedir o teste eficiente de certas dependências funcionais”. Ainda de acordo com os autores, a BCNF é um caso particular da 3FN, em que se pode observar:

> A BCNF exige que todas as dependências não tiviais seja da forma $\alpha \rightarrow \beta$, onde α é uma superchave. A terceira forma normal (3FN) atenua essa restrição permitindo dependências funcionais não triviais cujo lado esquerdo não é uma *superchave.* (SILBERSCHATZ et al., 2006, p. 183).

Para Silberschatz et al. (2006, p. 29), ainda, “uma **superchave** é um conjunto de um ou mais atributos que, tomados coletivamente, nos permite identificar unicamente uma tupla na relação.”.

2.4 INFORMAÇÕES GÊNICAS E ORIGEM DOS DADOS

Nesta seção, apresentamos a origem dos dados que foram coletados e analisados para a elaboração do banco de dados desenvolvido neste trabalho e os conceitos elementares que subsidiam a análise e interpretação em linhas gerais das informações constantes nos arquivos do GenBank.

2.4.1 Contexto Biológico

Segundo PROSDOCIMI (2002), a Biologia Molecular surgiu como ciência no século XX, com as descobertas: da molécula do DNA (ácido desoxirribonucléico) e seu papel no armazenamento das informações gênicas, da estrutura química identificada por Watson e Crick, do código genético e do fluxo da informação biológica, dos ácidos nucléicos para as proteínas, dos métodos de sequenciamento, entre outros.

As células são as unidades estruturais e funcionais de todos os organismos vivos, podem ser unicelulares como as bactérias ou pluricelulares (nos seres humanos são estimados estima 100.000.000.000.000 de células). Em cada célula existe um sistema complexo, que permite a realização de diversas funções especializadas, tais como, a conversão de nutrientes em energia, reprodução, transporte de substâncias, produção de proteínas, entre outras. Cada célula de um organismo vivo contém cromossomos, que são compostos de uma sequência de pares de bases de DNA. O conjunto das sequências do DNA forma o genoma, código que fornece as instruções de controle da replicação e do funcionamento das células e do organismo.

As células são classificadas em procariotos ou eucariotos. Nos eucariotos, os cromossomos são localizados no núcleo, estrutura formada pela membrana carioteca ou nuclear e que separa os cromossomos do citoplasma. Nos procariotos, os cromossomos estão imersos no citoplasma e formam uma estrutura denominada

nuclóide. Ainda, nos eucariotos o cromossomo é formado pelo DNA associado à molécula de histona (proteína básica).

O DNA é um polímero longo e linear, sem ramificações, que contém milhões de nucleotídeos. Os nucleotídeos são constituídos por três tipos de moléculas: um açúcar (desoxirribose), um fosfato e uma base nitrogenada (adenina, guanina, timina e citosina). As bases nitrogenadas podem ser vistas na figura 3.

Segundo o NCBI (2010, adaptado), as sequências de ácidos nucléicos representam o início de uma jornada que busca descrever e compreender a estrutura, a função e o desenvolvimento genético dos organismos.

As atividades celulares são resultantes de dois processos: expressão gênica e replicação (duplicação). No processo de replicação a fita do DNA é copiada para uma nova fita de DNA, enquanto que na expressão gênica, o DNA é transcrito para uma sequência de RNA (ácido ribonucléico). O RNA difere-se do DNA pela presença da base Uracila (representada na Figura 3) no lugar da Tinina.

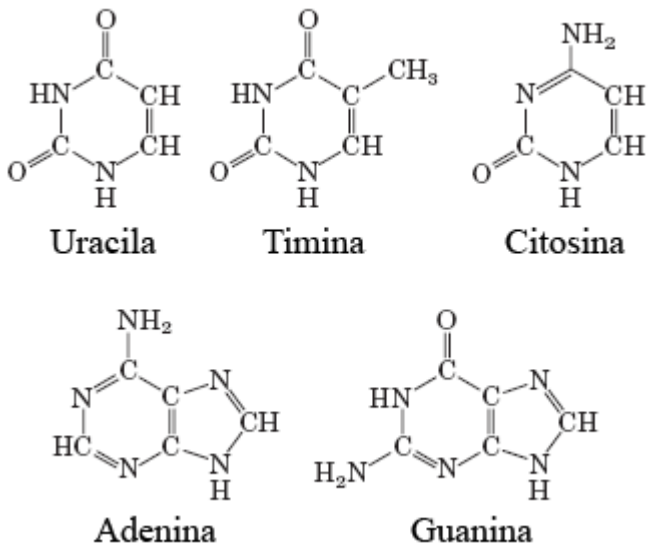

FIGURA 3 - BASES NITROGENADAS

FONTE: Traduzida de NELSON e COX (2004, p. 10)

Os RNAs possuem diversas funções no processo de sintetização das proteínas e são classificados nos tipos: mRNA ou RNA mensageiro, tRNA ou RNA transportador, rRNA ou RNA ribossomal e sRNA ou RNA pequeno (do inglês, small).

Nos procariotos as subunidades (pequenas moléculas de RNA ribossomal) ribossomais se unem a um mRNA formando o ribossomo. Uma tríplice de nucleotídeos (denominados códon de início) presente na sequencia do RNA

mensageiro indica o início do processo de tradução. À medida que a síntese da proteína ocorre, os RNA transportadores vão levando os aminoácidos para o interior do ribossomo. Cada três nucleotídeos (normalmente chamados de códon) presentes no RNA mensageiro são o código equivalente para um aminoácido. No final do processo de tradução, que normalmente é identificado pelo códon de parada, o ribossomo se desprende do RNA mensageiro, o ribossomo é dividido em suas subunidades e a proteína encontra-se concluída.

Os dois processos descritos anteriormente podem ser visualizados na Figura 4, que representa o dogma central da Biologia Molecular.

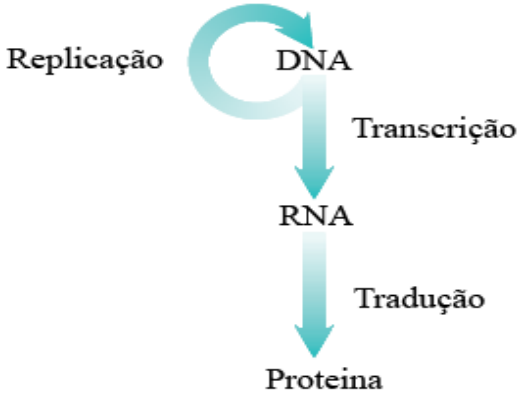

FIGURA 4 - DOGMA CENTRAL DA BIOLOGIA MOLECULAR

FONTE: Traduzida de NELSON e COX (2004, p. 922)

As proteínas são macromoléculas biológicas abundantes, de maneira que ocorrem em todas as células. Existem milhares de tipos de proteínas, de tamanhos muito diferentes e com um enorme número de funções biológicas. As proteínas são os instrumentos moleculares por meio do qual a informação genética é expressa. Todas as proteínas são construídas a partir do mesmo conjunto de vinte aminoácidos, ligados covalentemente – C-N resultante é chamada de ligação peptídica – em uma característica sequência linear. Para cada um desses aminoácidos, há uma cadeia lateral com propriedades químicas distintas, a combinação de um conjunto de aminoácidos é que codifica as proteínas. A partir dessas sequências de aminoácidos pode-se produzir uma diversidade de produtos, como enzimas, hormônios, anticorpos, transportadores, fibras moleculares e outras substâncias (NELSON e COX, 2004, p. 75).

Os 20 aminoácidos são: Glicina, Alanina, Leucina, Valina, Isoleucina, Prolina, Fenilalanina, Serina, Treonina, Cisteina, Tirosina, Asparagina, Glutamina, Aspartato, Glutamato, Arginina, Lisina, Histidina, Triptofano e Metionina. (NELSON e COX, 2004, p.78)

O processo bioquímico para descobrir as sequências das bases nitrogenadas presentes no DNA e no RNA de um organismo é chamado de sequenciamento. Existem vários métodos de seqüenciamento; porém, todos eles fornecem pequenas (entre 30 pb a 800 pb) sequências de nucleotídeos ou aminoácidos.

Os genomas – conjuntos de genes de um organismo – são descritos como sequências gigantescas (nas bactérias atualmente seqüenciadas, variam de 1 a 10 mpb) de nucleotídeos.

2.4.2 Seqüenciamento

Denomina-se “seqüenciamento” as técnicas ou métodos utilizados para identificar uma sequência de DNA ou RNA.

Frederick Sanger da Universidade de Cambridge desenvolveu o primeiro método em 1972 e conclui o seqüenciamento do genoma de um vírus em 1977, o método de Sanger era muito caro e inviável para projetos maiores. No mesmo ano, Maxam e Gilbert publicam o método “seqüenciamento do DNA por degradação química” (MAXAM E GILBERT, 1977). O projeto “Genoma Humano” iniciado no final dos anos 90 organizou e incentivou o desenvolvimento de novas técnicas. Segundo Elaine R Mardis (2008), a conclusão do seqüenciamento do genoma humano marca:

> *Since the completion of the human genome project (HGP) (26, 51), substantive changes have occurred in the approach to genome sequencing that have moved away from BACbased approaches and toward whole-genome sequencing (WGS), with changes in the accompanying assembly algorithms. In the WGS approach, the genomic DNA is sheared directly into several distinct size classes and placed into plasmid and fosmid subclones. Oversampling the ends of these subclones to generate paired-end sequencing reads provides the necessary linking information to fuel wholegenome assembly algorithms. The net result is that genomes can be sequenced more rapidly and more readily, but highly polymorphic or highly repetitive genomes remain quite fragmented after assembly.*
> (MARDIS, 2008)

Segundo a autora, os métodos de seqüenciamento apresentavam erros menores que 1 base errada a cada 40.000 pb. A partir de 2004, uma nova geração de seqüenciadores apareceu no mercado, reduzindo o custo e o tempo.

Ao descrever o processo de seqüenciamento Melissa Lemos (2004) acrescenta o aspecto que nem todos os projetos visam o seqüenciamento de genomas completos, diferencia as sequências e faz as considerações a seguir:

> Diferente do projeto genoma humano, nem todos os projetos têm o objetivo de sequenciar o genoma inteiro do organismo. Muitos projetos estão focados em obter apenas os genes expressos, chamados de ESTs, o que simplifica muito a tarefa e consequentemente, reduz os custos.
>
> EST (expressed sequence tag, em inglês) é uma sequência obtida de forma aleatória, geralmente incompleta, de DNA, que representa um gene expresso e que pode ser utilizado para identificar (tag) o gene. Um gene expresso é aquele cujo produto, seja uma proteína ou um RNA, está sendo produzido em um dado momento em uma célula.
>
> Estes projetos tiram vantagem do fato de ser relativamente simples fazer cópias de DNAs a partir de mRNAs durante o processo de tradução. Estas cópias, chamadas de cDNAs, quando sequenciadas, são nomeadas de ESTs.
>
> Considerando que os custos dos projetos genoma são elevados e que a quantidade de sequências não codificantes em eucariotos é muito maior que em procariotos, os projetos genoma do tipo ESTs são mais comuns para eucariotos, e os projetos completos para procariotos. Para se ter uma ideia, em um genoma de eucariotos, 90% do DNA não codifica para proteínas, tornando a obtenção das sequências dos genes que as codificam muito mais direta.
>
> Apesar dos genomas variarem de tamanho de milhões de nucleotídeos, em uma bactéria, para bilhões de nucleotídeos, em humanos e em grande parte dos animais e plantas, as reações químicas que os pesquisadores utilizam para decodificar os pares de bases do DNA durante o processo de sequenciamento são precisas para se obter apenas 600 a 700 nucleotídeos por vez. Sendo assim, um dos processos de sequenciamento mais utilizados, conhecido como shotgun, se inicia com a quebra do DNA em milhões de fragmentos aleatórios, que são então “lidos” por uma máquina de sequenciamento de DNA. A análise computacional é então usada para construir cromatogramas (consistindo de quatro curvas de cores diferentes, cada curva representando um sinal para uma das quatro bases), e para converter cada cromatograma em uma sequência de bases (chamadas de reads) por um programa específico.
> (LEMOS, 2004, p.25-26)

Para MARDIS (2008), os métodos e equipamentos disponíveis no mercado atualmente são:

> *Three platforms for massively parallel DNA sequencing read production are in reasonably widespread use at present: the*

> *Roche/454 FLX (30) (http://www.454.com/enablingtechnology/ the-system.asp), the Illumina/Solexa Genome Analyzer (7) (http://www.illumina.com/pages.ilmn?ID=203), and the Applied Biosystems SOLiDTM System (http://marketing.appliedbiosystems.com/images/ Product/Solid Knowledge/flash/102207/solid.html). Recently, another two massively parallel systems were announced: the Helicos HeliscopeTM (www.helicosbio.com)*
> *and Pacific Biosciences SMRT (www.pacificbiosciences.com) instruments. The Helicos system only recently became commercially available, and the Pacific Biosciences instrument will likely launch commercially in early 2010. Each platform embodies a complex interplay of enzymology, chemistry, high-resolution optics, hardware, and software engineering. These instruments allow highly streamlined sample preparation steps prior to DNA sequencing, which provides a significant time savings and a minimal requirement for associated equipment in comparison to the highly automated, multistep pipelines necessary for clone-based high-throughput sequencing.*
> (MARDIS, 2008)

O artigo contém uma descrição do funcionamento das técnicas e outras considerações, para maiores detalhes consulte-o em: http://www.annualreviews.org/doi/pdf/10.1146/annurev.genom.9.081307.164359.

Um segundo processo é necessário após o seqüenciamento para obter os genomas, esse processo é denominado montagem.

2.4.3 Montagem de sequências

O processo de montagem consiste em organizar os reads (sequências obtidas no processo de seqüenciamento) de tal forma que se recupere a ordem original da sequência do DNA do organismo seqüenciado. Segundo LEMOS (2004), após o seqüenciamento:

> (...) um programa chamado assembler, ou montador (por exemplo,os programas CAP3 [Huang, 1999] e Phrap [Green, 2004]), é incumbido da tarefa de combinar os reads para reconstruir a sequência original [Lemos, 2003a; Pop,2002]. Infelizmente, este processo não é simples. Os dados podem conter erros, sejam por causa de limitações na tecnologia de sequenciamento ou por falha humana durante o trabalho de laboratório, e mesmo na ausência de erros há características das sequências de DNA que complicam esse trabalho (um exemplo são as regiões repetitivas, chamadas de repetições).
>
> Na prática, a cobertura imperfeita, repetições e erros de sequenciamento fazem com que a montagem una apenas partes dos reads. Cada conjunto de reads que se une é chamado de contig. O objetivo final é unir todos os contigs gerados e formar um único contig, que representará um

> cromossomo completo. A tarefa de fechar os buracos entre os contigs e obter a molécula completa é chamada de finishing. (LEMOS, 2004, p.25-27)

A evolução observada nas tecnologias de seqüenciamento não se verificou nos programas de montagem de genomas, entre as novidades, atualmente destacam a abordam do Velvet e De Novo. Porém essas aplicações apresentam problemas quando se tem baixa cobertura (menor que 15) e área repetidas. Na prática, o Phrap continua apresentando os bons resultados e ajudando no processo de montagem.

A última etapa do processo de montagem é conhecida por acabamento (do inglês, Finishing). O acabamento visa superar duas limitações importantes do processo de seqüenciamento: a saída fragmentada do montador de genomas, causado principalmente pela presença de regiões repetidas e a presença de erros nos contigs, conseqüências de falhas de cobertura, seqüenciamento ou montagem de áreas repetidas. O processo de acabamento pode ser divido em fechamento das ausências existentes entre os contigs e a validação da montagem. Ainda segundo Nagarajan et al (2010):

> *In gap closure, pairs of adjacent contigs are identified, then the genomic sequence spanning the gap between them is determined, traditionally through directed-PCR and primer-walking approaches. When mate-pair libraries are available, the adjacency of contigs can often be inferred from the mate-pair data and the gaps spanned by paired reads (sequencing gaps) can be closed relatively easily. Contigs whose adjacency cannot be inferred from mate-pair data, however, require expensive (and error-prone) combinatorial PCR experiments [4]. New experimental technologies alleviate these difficulties in two ways. First of all, in nextgen sequencing projects performed to a high depth of coverage (>20-fold is common in 454 projects) sequence gaps between contigs are rare due to the relatively unbiased libraries generated by these new technologies; fragmentation into contigs is largely due to the presence of repeats. Therefore, often, once the adjacency between two contigs is determined (e.g. through PCR experiments), the contigs can be simply "glued" together without the need for additional sequencing. Second of all, the adjacency of contigs can be easily determined either through recently developed nextgen mate-pair protocols or through the use of new mapping technologies, such as the optical mapping approach from Opgen Inc. http://www.opgen.com.*
>
> *The validation and refinement finishing stage aims to correct errors in the assembled sequence - both single-base errors (such as mis-called bases due to sequencing errors) as well as large-scale errors (such as mis-assemblies due to repeats). Both problems are somewhat alleviated in nextgen sequencing data. Due to a high level of coverage, most single base errors can be automatically corrected.*

> *This is true even in the case of 454 pyrosequencing where errors within homo-polymer tracts are common. Furthermore, assembly software designed specifically for high-coverage nextgen data (e.g. the Newbler assembler from 454) use conservative algorithms specifically designed to avoid mis-assemblies. The resulting assemblies are usually more fragmented; contigs end at repeat boundaries where the reconstruction of the genome is ambiguous. The high depth of coverage and conservative assembly strategy also enable a better estimation of the number of repeat copies contained within a contig. Repeat-induced ambiguities can be resolved through targeted PCR experiments aimed at uncovering the correct adjacency of the assembled contigs. As in the case of gap closure, once two contigs have been determined to be adjacent in the assembly, they can be simply glued together. In the following section we describe the results from our experience in putting these principles into practice*
> (Nagarajan et al, 2010)

Destes processos de seqüenciamento, saem a primeira parte do conjunto de informações que é depositada e redistribuída pelo NCBI GenBank. As sequências de nucleotídeos representam aproximadamente 45% dos dados existentes nos arquivos textos do GenBank.

Após o processo de sequenciamento e montagem, os pesquisadores desenvolvem a anotação do genoma, como descrito na próxima seção.

2.4.4 Anotação

De acordo com Weiss (2010), uma vez obtida a sequência genômica é necessário agregar informação biológica a ela. Esta caracterização é chamada de anotação, em que são identificadas regiões codificadoras com base na parte estrutural (códon de início, códon de término, quantidade de GC) e regulatória (região promotora, sítio de ligação de ribossomo).

Para Melissa Lemos (2004),

> Como projetos de genoma geram dados brutos, ou seja, sem significado biológico, há outra fase, chamada de fase de anotação, cujo objetivo consiste em converter esses dados em informações biologicamente relevantes (sequências anotadas).
>
> Uma anotação é uma meta-informação ou uma descrição de características em mais alto nível da sequência biológica, ou biossequência. Anotações úteis incluem vários tipos de informações, por exemplo, se um trecho de DNA contém um gene e qual a função dele.
>
> Os pesquisadores usam diversas ferramentas e programas de computador, combinados com interpretação humana, para realizar esse processo de

> anotação. Porém, considerando a velocidade com que pesquisadores geram sequências de DNA atualmente, anotar os dados automaticamente torna-se um desafio computacional, e a análise humana cuidadosa está se tornando cada vez mais difícil. (LEMOS, 2004, p.25-27).

Ferro (2008) conceitua que a anotação

> [...] pode ser compreendida como características que descrevem e qualificam uma sequência biológica. A anotação de sequências, segundo Stein (2001), consiste em um processo múltiplo, pelo qual uma ou mais sequências brutas de DNA ou de aminoácidos são analisados com a finalidade de se atribuir características, contextualizando-se estas sequências do ponto de vista biológico conforme suas funções. Assim, na anotação de sequências acrescentam-se informações à sequência nucleotídica, tais como localização de genes, presença de regiões repetitivas, identificação de genes de tRNA, rRNA e codificadores de proteínas, etc. Algumas dessas descrições podem ser deduzidas a partir de dados de similaridade com sequências biológicas cuja função já seja conhecida. Embora a anotação geralmente esteja relacionada a sequências genômicas, qualquer sequência biológica pode ser anotada, incluindo sequências de cDNAs e proteínas (Eibeck et al., 2005; Berriman e Harris, 2004). (FERRO, 2008, p. 28-29).

Os problemas mais comuns relatados no processo de anotação são:

- similaridade x homologia;
- genes depositados em bancos de dados sem função;
- genes hipotéticos;
- genes hipotéticos conservados;
- falta de padronização dos vocabulários de anotação, e
- erros/falta de anotação nos bancos de dados públicos.

Conforme Guimarães (2006),

> Homologia, como definição geral, designa um relacionamento de descendência comum entre quaisquer entidades, em um cenário evolutivo, e se refere a duas estruturas ou sequências que se desenvolveram de uma única estrutura ou sequência ancestral. Assim, as entidades relacionadas por homologia, em particular genes, são chamadas homólogas. (GUIMARÃES, 2006, p. 24).

Altschul et al. (1990) descrevem a similaridade entre sequências, como uma unidade de medida, em

> *The discovery of sequence homology to a known protein or family of proteins often provides the first clues about the function of a newly sequenced gene. As the DNA and amino acid sequence databases continue to grow in size they become increasingly useful in the analysis of newly sequenced genes and proteins because of the greater chance of finding*

> *such homologies. There are a number of software tools for searching sequence databases but all use some measure of similarity between sequences. To distinguish biologically significant relationships from chance similarities. Perhaps the best studied measures are those d in conjunction with variations .of the dynamic programming algorithm (X'eedleman 8c Wunsch, 1970; Sellers, 1974; Sankoff 8: Kruskal, 1983; Waterman, 1984).*
>
> *These methods assign scores to insertions, deletions and replacements, and compute an alignment of two sequences that corresponds to the least costly set of such mutations. Such an alignment may be thought of as minimizing the evolutionary distance or maximizing the similarity between the two sequences compared. In either case, the cost of this alignment is a measure of similarity (ALTSCHUL, 1990).*

Um protocolo para detecção de erros foi proposto por Schnoes et al. (2009):

> *Due to the rapid release of new data from genome sequencing projects, the majority of protein sequences in public databases have not been experimentally characterized; rather, sequences are annotated using computational analysis. The level of misannotation and the types of misannotation in large public databases are currently unknown and have not been analyzed in depth. We have investigated the misannotation levels for molecular function in four public protein sequence databases (UniProtKB/Swiss-Prot, GenBank NR, UniProtKB/TrEMBL, and KEGG) for a model set of 37 enzyme families for which extensive experimental information is available. The manually curated database Swiss-Prot shows the lowest annotation error levels (close to 0% for most families); the two other protein sequence databases (GenBank NR and TrEMBL) and the protein sequences in the KEGG pathways database exhibit similar and surprisingly high levels of misannotation that average 5%–63% across the six superfamilies studied. For 10 of the 37 families examined, the level of misannotation in one or more of these databases is .80%. Examination of the NR database over time shows that misannotation has increased from 1993 to 2005. The types of misannotation that were found fall into several categories, most associated with "overprediction" of molecular function.*
>
> *These results suggest that misannotation in enzyme superfamilies containing multiple families that catalyze different reactions is a larger problem than has been recognized. Strategies are suggested for addressing some of the systematic problems contributing to these high levels of misannotation (SCHNOES et al., 2009).*

Pelos percentuais de erros indicados, deve-se considerar a possibilidade de existir pelo menos um erro de anotação presente em cada genoma sequenciado disponível no GenBank.

2.4.5 As sequências e Anotações são depositadas no GenBank

Depois que os genomas são sequenciados, montados e anotados, em geral, os pesquisadores depositam essas informações em um dos três grandes bancos de

dados públicos que formam o *International Nucleotide Sequence Database Collaboration* (INSDC).

De acordo com Benson et al. (2010), o NCBI oferece esses dados coletados, gratuitamente, de diversas formas, conforme segue:

> *[…]]hich exchanges data daily to ensure that a uniform and comprehensive collection of sequence information is available worldwide. NCBI makes the GenBank data available at no cost over the Internet, through FTP and a wide range of web-based retrieval and analysis services (4).*
>
> ***ORGANIZATION OF THE DATABASE***
>
> *From its inception, GenBank has grown exponentially, and continues to do so with the number of sequence records doubling approximately every 35 months. The traditional GenBank divisions contain 106 billion nucleotide bases from 108 million individual sequences, with 11 million new sequences added in the past year. Contributions from whole genome shotgun (WGS) projects supplement the data in the traditional divisions to bring the total to 255 billion bases. Complete genomes (www.ncbi.nlm.nih.gov/Genomes/) continue to represent a rapidly growing segment of the database. GenBank now contains more than 1000 complete genomes from bacteria and archaea, and 30% of these were deposited during the past year. The number of eukaryote genomes with significant coverage and assembly continues to increase as well, with over 380 WGS assemblies now available. […]*
>
> ***BUILDING THE DATABASE***
>
> *The data in GenBank and the collaborating databases, EMBL and DDBJ, are submitted primarily by individual authors to one of the three databases, or by sequencing centers as batches of EST, STS, GSS, HTC, WGS or HTG sequences. Data are exchanged daily with DDBJ and EMBL so that the daily updates from NCBI servers incorporate the most recently available sequence data from all sources.*
>
> *[…]**RETRIEVING GenBank DATA***
>
> *[…]*
>
> *Obtaining GenBank by FTP NCBI distributes GenBank releases in the traditional flat file format as well as in the ASN.1 format used for internal maintenance. The full bi-monthly GenBank release along with the daily updates, which incorporate sequence data from EMBL and DDBJ, is available by anonymous FTP from NCBI at ftp.ncbi.nih.gov/genbank. The full release in flat file format is available as a set of compressed files with a non-cumulative set of updates at ftp.ncbi.nih.gov/daily-nc/. Uncompressed, a complete copy of release 173 occupies 437 GB. A script is provided in ftp.ncbi.nih .gov/tools/ to convert a set of daily updates into a cumulative update.(BENSON et al., 2010).*

Neste trabalho, optou-se por obter os dados no servidor de arquivos (FTP) e atualizar os dados a cada revisão oficial, o que ocorre em média a cada sessenta dias. Na área da bioinformática, surgiram diversos formatos de arquivos; entretanto,

o mais usado para sequências é o formato Fasta e para sequências e anotações é o formato do GenBank. No servidor de arquivos do NCBI, foram encontradas informações em diversos formatos. Neste trabalho, porém, usa-se apenas o formato GenBank.

2.5 BANCO DE DADOS BIOLÓGICOS

2.5.1 GenBank

O banco de dados GenBank (MIZRACHI, 2010) é um banco de dados público de sequências de nucleotídeos e anotação de apoio bibliográfico e biológico, criado, distribuído e mantido pelo *National Center for Biotechnology Information* (NCBI), uma divisão da *National Library of Medicine* (NLM), localizada no campus do US *National Institutes of Health* (NIH) em Bethesda, MD (BENSON, 2010).

O NCBI juntamente com o European Bioinformatics Institute (EBI) do European Molecular Biology Laboratory (EMBL) e o DNA Data Bank of Japan (DDBJ) constituem o International Nucleotide Sequence Database Collaboration (INSDC), por meio da qual as informações são permutadas diariamente.

As informações estão disponíveis em vários programas do NCBI e também em arquivos no formato texto em seu servidor de arquivos (FTP). Esses arquivos seguem o padrão definido em conjunto pelo EMBL e recebe o nome de GenBank; portanto, o nome GenBank se refere ao mesmo tempo a um banco de dados do NCBI e a um formato de arquivo. Este formato de arquivo, de acordo com LEE et al. (2008), é um dos mais populares, devido ao detalhamento das características (anotação) das sequências e da legibilidade.

Há mais informações sobre o GenBank no Manual do NCBI, bem como o resumo no GenBank em http://www.ncbi.nlm.nih.gov/Genbank/index.html.

A especificação do “NCBI-GenBank Flat File Release 178.0”, empregada para o desenvolvimento desse trabalho, está nos sites do NCBI, no endereço: ftp://ftp.ncbi.nih.gov/genbank/gbrel.txt. Detalhes são apresentados no documento http://www.ncbi.nlm.nih.gov/projects/collab/FT/index.html.

A nomenclatura dos arquivos encontrados no servidor de arquivos do NCBI é padronizada, conforme descrito em “NCBI Reference Sequence (RefSeq)”:

> *RefSeq accession numbers can be distinguished from GenBank accessions by their distinct prefix format of 2 characters followed by an underscore character ('_'). For example, a RefSeq protein accession is NP_015325.*
> *Accession Molecule Method@*
> *Note*
> *[…]*
> *AP_123456 Protein Mixed*
> *Protein products; alternate protein record. This prefix is used for records that are provided to reflect an alternate assembly or annotation. The AP_ prefix was originally designated for bacterial proteins but this usage was changed.*
>
> ***NC**_123456 Genomic Mixed*
> ***Complete genomic molecules including genomes, chromosomes, organelles, plasmids.** [negrito nosso]*
> *[…]*
> *NS_123456 Genomic Automated*
> *Genomic records that represent an assembly which does not reflect the structure of a real biological molecule. The assembly may represent an unordered assembly of unplaced scaffolds, or it may represent an assembly of DNA sequences generated from a biological sample that may not represent a single organism. (NCBI, 2010, key.html).*

Do padrão proposto pelo NCBI para a nomenclatura, o primeiro símbolo no nome do arquivo pode ser: A – registro de alternativo; N – nucleotídeos, X, Y e Z para proteínas. Na segunda posição, os valores possíveis são: C completo; P proteína; G – genoma; M, P, R, T, W, S – referem-se a proteínas.

Dos três bancos de dados públicos, o NCBI GenBank é o mais popular, sendo esse o que recebe a maior quantidade de submissões e que reporta o maior crescimento de pares de bases sequenciadas e disponíveis, conforme pode ser demonstrada pela FIGURAS 5 e 6.

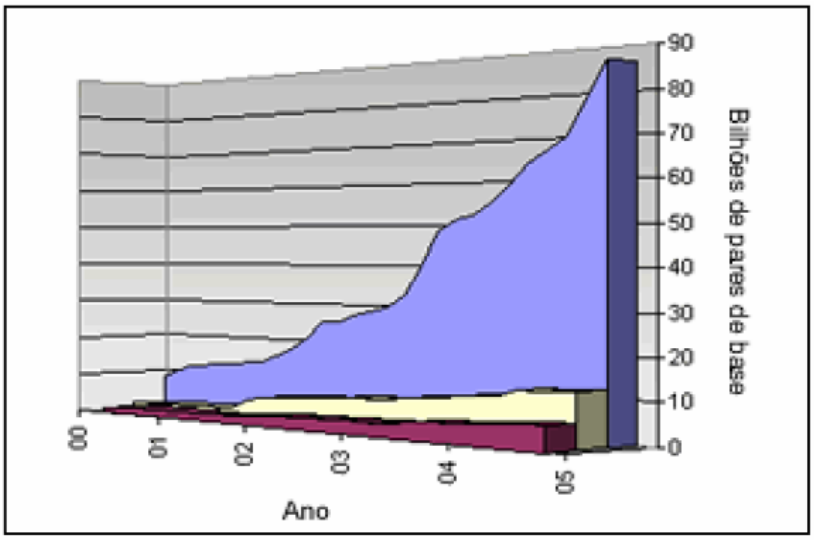

FIGURA 5 - COMPARAÇÃO DO CRESCIMENTO EM NÚMEROS DE PARES DE BASES: GENBANK (AZUL), EMBL (AMARELO) E DDBJ (ROXO)

FONTE: GUIMARÃES (2006, p.8)

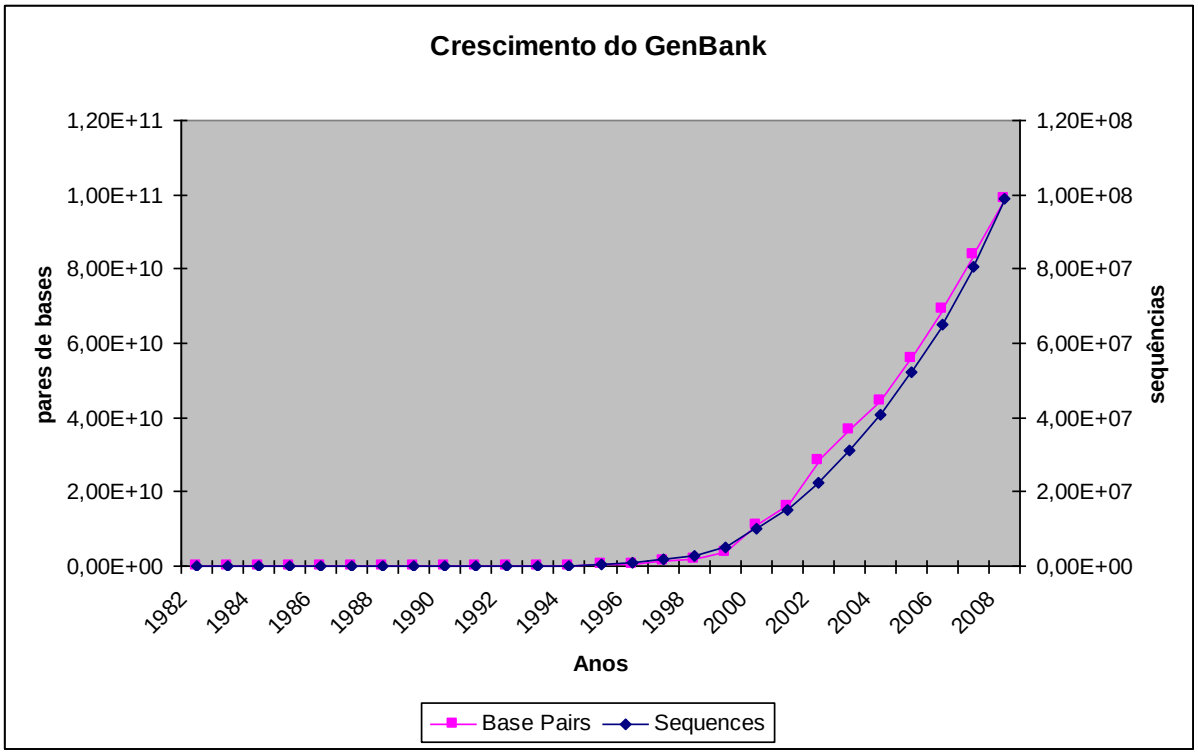

FIGURA 6 - CRESCIMENTO EM NÚMEROS DE PARES DE BASES E SEQUÊNCIAS NO GENBANK[3]

FONTE: imagem produzida pelo autor com base em página da internet (GENBAK STATISTICS, NCBI, 2010).

2.5.2 European Molecular Biology Laboratory (EMBL)

O Banco de Sequências de Nucleotídeos EMBL (ou banco EMBL), é o banco de dados europeu primário de sequências. Os pesquisadores submetem diretamente suas sequências nesse banco, de forma semelhante ao NCBI GenBank. O EMBL juntamente com o GenBank e o DDBJ compõem o INSDC. Os dados entre os três bancos são permutados diariamente (COCHRANE et al., 1998). Os dados do EMBL deram origem ao projeto *European Nucleotide Archive* (ENA) (COCHRANE et al., 2008).

O banco de dados é um projeto liderado por Guy Cochrane, e integra o Grupo PANDA (acrônimo de The **P**rotein **a**nd **N**ucleotide **Da**tabase Group - Banco de Proteínas e Nucleotídeos) coordenador por Ewan Birney (EBI, 2010).

[3] http://www.ncbi.nlm.nih.gov/genbank/genbankstats.html

2.5.3 DNA Data Bank of Japan (DDBJ)

O “DNA Data Bank of Japan” (DDBJ) tem funcionado como base de dados internacional de sequências nucleotídeos, sediado no Japão, faz parte do National Institute of Genetics (NIG – http://www.ddbj.nig.ac.gp) e compõe o INSDC (International Nucleotide Sequence Database Collaboration).

Conforme o DDBJ, seu principal objetivo é melhorar a qualidade do INSDC como domínio público. Quando os pesquisadores enviam dados para os bancos de dados, o DDBJ realiza esforços para enriquecer as informações descritas, seguindo um conjunto de normas unificadas e, de preferência, automatizado (DNA DATA BANK OF JAPAN, 2010).

2.5.4 Outros bancos específicos

A diversidade de bancos de dados biológicos é gigantesca. Geralmente, os bancos de dados nascem de uma necessidade específica para um projeto e acabam alcançando o domínio público, em função da relevância das descobertas ou pelas exigências das revistas especializadas que muitas vezes condicionam o acesso a esses dados para aceitarem e publicarem os artigos submetidos.

Segundo COCHRANE & GALPERIN (2009), atualmente existem 1230 bancos de dados de algum aspecto biológico. A lista completa pode ser encontrada no endereço da internet: http://www.oxfordjournals.org/**nar**/**database**/a/.

Nesta pesquisa, complementou-se essa lista e identificou-se 1.450 bancos de dados. Ao acompanhar a estrutura de organização e classificação da *Nucleic Acids Research Database* (NAR *Database*), tem-se o resumo apresentado na TABELA 3.

TABELA 3 - AGRUPAMENTO DOS BANCOS DE DADOS BIOLÓGICOS, BASEADOS NO MODELO DA NAR DATABASE

Grupo / Categoria dos Trabalhos	Quantidade
Doenças e Gene Humanos	129
Estruturas	126
Expressão Gênica e Dados de Microarray	69
Genoma Humano e de outros vertebrados	114
Genomas (não vertebrados)	274
Imunológicos	27
Organelas	26
Outros Bancos de Dados de Biologia Molecular	44
Plantas	107
Recursos de Proteômica	20
Sequências de Nucleotídeos	134
Sequências de Proteínas	189
Sequências de RNA	72
Vias metabólicas	119
Total geral	**1450**

FONTE: o autor (2010). Com base nos dados do artigo da NAR Database.

Outro trabalho que tem o propósito de catalogar os bancos de dados é o MetaBase (BIODATABASE, 2010), que contabiliza 1.800 bancos de dados cadastrados.

2.6 COMO SÃO VISTOS OS BANCOS DE DADOS NA ÁREA DA BIOINFORMÁTICA

Russ B. Altman (1998) abre o editorial da Bioinformatics com o título "*A Curriculum for Bioinformatics: The Time is Ripe*" e logo no primeiro parágrafo traduz algumas questões que permeavam a discussão da área em 1998, conforme segue:

> *There seems to be wide agreement within both industry and academia that there are not enough scientists adequately trained in bioinformatics or computational biology. This sentiment stems principally from the difficulties in finding employees, graduate students and post-docs with appropriate*

> *skills for joining research and/or development teams in this field. The recent drain of academics into industry threatens to reduce our ability to provide the training needed to meet the demand of the job markets. An obvious question is 'What is the proper curriculum for bioinformatics professionals?' At first, the idea of defining a curriculum for bioinformatics may seem premature. The very definition of bioinformatics is still the matter of some debate.*
>
> *Although some interpret it narrowly as the information science techniques needed to support genome analysis, many have begun to use it synonymously with 'computational molecular biology' or even all of 'computational biology'. For this discussion of curriculum,* ***bioinformatics addresses problems related to the storage, retrieval and analysis of information about biological structure, sequence and function****.[negrito nosso] (ALTMAN, 1998, p. 549).*

Problemas de armazenamento, recuperação e análise de informação são comumente abordados por aplicações de banco de dados na área da informática.

No mesmo texto, o autor descreve que as competências exigidas para os cursos de bioinformática podem ser divididas em cinco áreas: biologia, ciência da computação, estatística, ética e bioinformática pura. Ele apresenta a necessidade da modelagem e implementação de bancos de dados biológicos, a preocupação em técnicas de otimização, algoritmos de buscas entre outros. Segue a descrição do autor para a bioinformática pura:

> *5. The final and most critical portion of the curriculum would be a set of core bioinformatics courses that build upon the contributing disciplines to present the basic intellectual structure of the field. The precise list of core areas remains a topic of debate, but the recent release of a number of textbooks for bioinformatics (Baldi and Brunak, 1998; Bishop and Rawlings, 1997; Durbin et al., 1998; Gusfield, 1997; Salzberg et al., 1998; Setubal and Medianis, 1997) indicates that there is general agreement about the importance of certain fundamental concepts.These include (in no particular order):*
> *(…)*
> *- Design and implementation of biological databases/knowledge bases.*
>
> *In addition, the computer science and statistics sections of the curriculum should introduce the student to certain key technologies that are commonly used within bioinformatics, including:*
> - *Optimization (Expectation Maximization, Monte Carlo, Simulated Annealing, gradient-based methods)*
> - *Dynamic programming*
> - *Bounded search algorithms*
> - *Cluster analysis*
> - *Classification*
> - *Neural Networks*
> - *Genetic Algorithms*
> - *Bayesian Inference*
> - *Stochastic Context Free Grammars.*
>
> (ALTMAN, 1998, p. 549).

As questões de modelagem de banco de dados e otimização são amplamente aplicadas nesse trabalho, pois organização e desempenho foram problemas observados por Altmann em 1998 e, ainda hoje, há grandes demandas nesse seguimento.

Prosdocimi et al. (2002) relatam a inexistência de cursos formais na área da bioinformática no Brasil e a criação dos primeiros programas de pós-graduação em nível de Doutorado na USP e na UFMG, bem como os principais conceitos e recursos de informática aplicados. Sobre banco de dados, os autores afirmam:

> Em consequência da grande quantidade de informações de sequências de nucleotídeos e de aminoácidos que são produzidas atualmente, principalmente em projetos Genoma, Transcriptoma e Proteoma, o uso dos bancos de dados vem assumindo uma importância crescente na bioinformática.
>
> Um banco de dados pode ser considerado uma coleção de dados inter-relacionados, projetado para suprir as necessidades de um grupo específico de aplicações e usuários. Um banco de dados organiza e estrutura as informações de modo a facilitar consultas, atualizações e deleções de dados. (PROSDOCIMI et al., 2002, p.14).

Neste trabalho, concorda-se com os autores, e quase oito anos após esse artigo, os dados do GenBank ainda não são fornecidos conforme um modelo relacional, ou para uma tecnologia de SGBD.

De acordo com o NCBI[4] (2010), “Essa avalanche de informações genômicas, por sua vez, levou a uma exigência absoluta para bases de dados informatizadas para armazenar, organizar e indexar os dados e ferramentas especializadas para visualizar e analisar os dados.” Esses autores contribuem, também, com a definição de banco de dados biológicos:

> ***What Is a Biological Database?***
>
> *A **biological database** is a large, organized body of persistent data, usually associated with computerized software designed to update, query, and retrieve components of the data stored within the system. A simple database might be a single file containing many records, each of which includes the same set of information. For example, a record associated with a nucleotide sequence database typically contains information such as contact name, the input sequence with a description of the type of molecule, the scientific name of the source organism from which it was isolated, and often, literature citations associated with the sequence.*

[4] http://www.ncbi.nlm.nih.gov/About/primer/bioinformatics.html

> *For researchers to benefit from the data stored in a database, two additional requirements must be met:*
> - *easy access to the information;*
> - *a method for extracting only that information needed to answer a specific biological question.*
>
> *At NCBI, many of our databases are linked through a unique search and retrieval system, called Entrez. Entrez (pronounced ahn' tray) allows a user to not only access and retrieve specific information from a single database but to access integrated information from many NCBI databases. For example, the Entrez Protein database is cross-linked to the Entrez Taxonomy database. This allows a researcher to find taxonomic information (taxonomy is a division of the natural sciences that deals with the classification of animals and plants) for the species from which a protein sequence was derived.*
> (NCBI, 2010, http://www.ncbi.nlm.nih.gov/About/primer/bioinformatics.html)

De acordo com Elmasri e Navathe (2005), as principais características dos dados biológicos são:

- Por serem altamente complexos comparados a outras aplicações, as definições dos dados biológicos devem ser capazes de representar uma subestrutura de dados complexa e deverá garantir que nenhuma informação seja perdida durante a modelagem dos dados.

- Os sistemas biológicos devem ser flexíveis ao lidar com tipos e valores de dados. A colocação de restrições deve ser limitada, uma vez que isso pode excluir valores inesperados. A exclusão desses valores resulta em perda de informação.

- Os esquemas nos bancos de dados biológicos mudam muito rápido. Para um maior fluxo de informações entre gerações ou versões de bancos de dados, a evolução do esquema e a migração de objetos de dados devem ser possíveis. Um banco de dados evolutivo fornece um mecanismo oportuno e ordenado para acompanhar as modificações em entidades de dados individuais nos bancos de dados biológicos ao longo do tempo.

- Mesmo que se utilize o mesmo sistema, as representações dos mesmos dados por diferentes biólogos provavelmente serão diferentes. Desta maneira, devem ser suportados mecanismos para “alinhar” diferentes esquemas biológicos ou diferentes versões de esquema. Devido à complexidade dos dados biológicos, existem diversas maneiras para modelar qualquer entidade fornecida com resultados que refletem o foco particular do cientista. Ainda que dois biólogos produzam modelos de dados diferentes, se lhes for solicitado que interprete a mesma entidade, esses modelos provavelmente terão inúmeros pontos em comum. Nessas circunstâncias, é necessário que os pesquisadores sejam capazes de executar consultas através desses pontos comuns.

- A maioria dos usuários de dados biológicos não necessita de acesso de escrita no banco de dados, apenas acesso para leitura. Os usuários geram uma variedade de padrões de acesso de leitura no banco de dados que são diferentes dos padrões dos bancos de dados tradicionais.

- A maioria dos biólogos provavelmente não possui conhecimento da estrutura interna do banco de dados, ou seja, eles sabem de quais dados necessitam, mas não possuem conhecimentos técnicos sobre como um sistema de banco de dados representa os dados. Neste caso, as

> interfaces do banco de dados biológicos devem exibir para os usuários informações de maneira que seja aplicável para o problema que eles estejam tentando tratar e reflita a estrutura dos dados de bases.
>
> - Os sistemas biológicos precisam dar suporte a consultas complexas, pois a definição e a representação destas consultas são extremamente importantes para o biólogo. Sem conhecimento da estrutura de dados, os usuários comuns não podem construir por conta própria uma consulta complexa através dos dados. Sendo assim, os sistemas devem fornecer ferramentas para que se construam essas consultas.
>
> - Frequentemente, os pesquisadores desejam consultar os dados mais atualizados, mas devem também ser capazes de reconstruir trabalhos anteriores e reavaliar informações anteriores e atuais. Desta maneira, os valores que estão para ser atualizados em um banco de dados biológicos não devem ser descartados. (ELMASRI E NAVATHE, 2005).

Ana Carolina Ramos Guimarães (2006) considera os problemas com os bancos de dados e as informações depositadas, pois

> Por outro lado, alguns problemas ainda não foram totalmente solucionados, como a redundância (vários registros contendo a mesma sequência), contaminação (presença de sequências dos vetores de clonagem nas sequências submetidas ao banco), erros nas anotações, distintos conceitos e formatações. (GUIMARÃES, 2006, p.5-6).

Em nenhum dos trabalhos consultados citam detalhes da organização interna dos três bancos de dados primários, e/ou mecanismos de obtenção de mecanismos de atualização para qualquer um dos SGBD comerciais, gratuitos ou livres disponíveis no mercado.

2.6.1 Como os bancos de dados biológicos são classificados

Para Prosdocimi et al. (2002, p.14), os bancos de dados biológicos são classificados em primários e secundários:

> Os primeiros são formados pela deposição direta de sequências de nucleotídeos, aminoácidos ou estruturas protéicas, sem qualquer processamento ou análise. Os principais bancos de dados primários são o GenBank,o EBI (European Bioinformatics Institute), o DDBJ (DNA Data Bank of Japan) e o PDB (Protein Data Bank).
> [...]
> Os bancos de dados secundários, como o PIR (Protein Information Resource)ouoSWISS-PROT, são aqueles que derivam dos primários, ou seja, foram formados usando as informações depositadas nos bancos primários. (PROSDOCIMI et al., 2002, p.14).

Os autores descrevem que os bancos de dados podem ser classificados como estruturais ou funcionais: os bancos estruturais mantêm dados relacionados à estrutura de proteínas (PROSDOCIMI et al., 2002) e os funcionais armazenam informações em forma de mapas metabólicos e busca por palavras-chave.

Ana Carolina Ramos Guimarães (2006) observa que os bancos de dados biológicos não costumam ser classificados pelas tecnologias empregadas e sim pelo conteúdo das informações. Para a autora,

> Disso resulta uma série de diferentes classificações, todas arbitrárias, onde não há uma "melhor" ou "pior", e sim a mais adequada a um determinado propósito (na maioria das vezes didáticas). Por exemplo, podemos dividi-los em i) banco de dados de sequências (nucleotídicas e protéicas); ii) banco de dados de estruturas e iii) outros. Outra abordagem de classificação sugere sua divisão em i) banco de dados primários, onde ocorre a deposição direta de sequências sem nenhum processamento interpretativo, e ii) banco de dados secundários, formados com informação processada derivada dos bancos primários. No primeiro grupo, situam-se o GenBank, o EMBL (mantido pelo EBI – European Bioinformatics Institute), o DDBJ (DNA Data Bank of Japan) e o PDB (Protein Data Bank). Do segundo grupo podemos citar o PIR (Protein Information Resource), o Swiss-Prot, o REBASE, o PROSITE, o Pfam, entre outros (Tsoka & Ouzounis, 2000).
>
> Além disso, os bancos podem ser classificados também como estruturais ou funcionais. O primeiro grupo corresponde aos bancos contendo dados relativos à estruturas de proteínas, como visto nos bancos SCOP (Structural Classification of Protein), CATH (Class/Architecture/Topology/Homology) e FSSP (Structure-Structure Aligmment of Proteins). Já os bancos funcionais armazenam informações relativas à função dos genes e proteínas em termos de mecanismos de reação enzimática, participação em vias bioquímicas e localização celular, como será mostrado mais adiante (Ouzounis et al.,2003). (GUIMARÃES, 2006, p.5-6).

A Revista Nucleic Acids Research Database classifica em seu site os trabalhos na área de banco de dados, com as seguintes categorias:

> – *RNA sequence databases*
> – *Protein sequence databases*
> – *Structure Databases*
> – *Genomics Databases (non-vertebrate)*
> – *Metabolic and Signaling Pathways*
> – *Human and other Vertebrate Genomes*
> – *Human Genes and Diseases*
> – *Microarray Data and other Gene Expression Databases*
> – *Proteomics Resources*
> – *Other Molecular Biology Databases*
> – *Organelle databases*
> – *Plant databases*
> – *Immunological databases*
> (OXFORD JOURNALS, 2010)

2.6.2 Considerações gerais e recomendações para projetos de banco de dados biológicos

Conforme Birney et al. (2002), o processo de criação, gerenciamento e exibição de informações sobre o genoma requerem *software* não trivial. De acordo com esses autores, a complexidade está associada à escala de armazenamento, aos recursos computacionais necessários e à complexidade dos problemas. Assim, as soluções adotadas requerem um mosaico de componentes de *software* que apresentam mais semelhanças que diferenças.

Catanho e Miranda (2007) entendem que a criação e manutenção de bases de dados biológicos é um desafio por si, não somente porque quase sempre envolve um grande número de dados, mas, sobretudo, porque requer a concepção de esquemas e estruturas que representem com precisão a complexidade dos sistemas biológicos; ou seja, trata-se de uma tarefa frequentemente difícil de ser cumprida.

2.7 FORMATO DE ARQUIVOS

Um arquivo é uma unidade lógica (ou entrada) nos sistemas de arquivos. É função do sistema operacional apresentar ao usuário um amigável e limpo modelo abstrato de arquivos independentemente de dispositivo (TANENBAUM, 2000). Existem vários sistemas de arquivos universais que permitem a leitura e escrita em dispositivos externos (pen-drives, CDs, DVDs), por diversos sistemas operacionais.

Um arquivo é uma sequência finita de *bytes*. Pode ser classificado em binário ou texto. São chamados de arquivos textos os arquivos em que os conteúdos são expressos por símbolos geralmente reconhecíveis pelas pessoas, sem a necessidade de ocorrer à decodificação da informação. Nos arquivos textos, normalmente não se tem os símbolos de controle – valores inferiores a 31 da tabela ASCII –, exceto os de final de linha e tabulação. Os arquivos binários são opostos aos arquivos textos, quase sempre apresentam codificação inelegível e ocorrência de todos os valores possíveis do conjunto de caracteres em uso. A diferença essencial entre arquivo texto e binário está em seu tamanho. Como os arquivos textos possuem um conjunto menor de símbolos do que o conjunto dos arquivos binários, estes precisam de mais combinações dos símbolos disponíveis para representar a mesma informação. Eis um exemplo prático: o número 200 precisa de

três símbolos para ser representado em um arquivo texto e apenas um byte (ou símbolo) para ser representado em formato binário.

Os arquivos podem apresentar organizações rígidas ou estruturadas; neste caso, diz-se que o arquivo está no formato X ou que ele é do formato X. Durante muito tempo, as estruturas dos arquivos foram usadas como segredo industrial e serviram para manter os clientes presos a tecnologias dos fabricantes de *softwares*. Muitos dos formatos amplamente usados são patenteados ou foram criados por empresas que restringem o uso do formato; neste caso, são chamados de formatos proprietários. Sobretudo com o advento do *software* livre e dos princípios defendidos pelo Creative Commons (2010 – *Copyleft*), no entanto, as empresas começaram a definir modelos universais ou formato aberto (ODF ALLIANCE, 2010, *Open Document Format*). A comunidade open-source desenvolve versões de programas, em geral, capazes de “ler” um formato proprietário ou expor para a sociedade a estrutura desses arquivos. O Wotsit (2010) é um desses depósitos que armazenam a informação de mais de mil contribuições sobre diversos formatos.

2.8 FORMATO DE ARQUIVOS DE INFORMAÇÕES BIOLÓGICAS

Os dois formatos de arquivos mais empregados para armazenar sequências e anotações são, respectivamente, o formato Fasta e o formato GenBank.

2.8.1 Fasta

De acordo com Guimarães (2006),

> [...] o algoritmo Fasta é um outro método heurístico para analisar o alinhamento local de sequências (Pearson, 1985). Anterior ao Blast, ainda é usado por muitos cientistas, principalmente devido à sua precisão, superior ao Blast. Por outro lado, sua execução é mais lenta quando comparada ao Blast. (GUIMARÃES, 2006, p. 15).

A aplicação Fasta contribui para o formato de arquivos mais usado para armazenamento de sequências. Um arquivo Fasta é formado por um conjunto de sequências, de maneira que para cada sequência existe um cabeçalho e a sequência propriamente dita. As informações são organizadas em um arquivo texto, normalmente com até sessenta colunas. Cada sequência pode ter uma linha de

cabeçalho e várias linhas compondo as sequências. A linha de cabeçalho é indicada pela presença do símbolo de maior (>) na primeira coluna. A FIGURA 7 apresenta um exemplo do conteúdo de um arquivo Fasta, com várias sequências.

```
>id_1_ct_1_Ponta: b
gatattgggttcgctggcgttgagcacggcattgcccgaagcgtcggcggtcaccttgggcagggtgacg
atgatctcggtgccgtggatgtccctgccgatgatggtgaccactacctgggcgaagggagggacgggaa
gacggtctggttcacgccattggcctccaccaaggcgcgcagtggcgtggcgcccaggttggtgatggtc
tgcgtggcttcatcgatgtagttgtcggccacttcgggcgtattgtcgcgcaccaccagcagcatcgggc
ccttgtagaagggacgggtg
>id_2_ct_1_Ponta: g
ccagcgcaaagctgggacctgtacgcctccatgtttgggggcattccacaaagcgagctggatgcactgg
ccgatttctggaccgccttccccgctctgcgcacagctctgtttatcgacaatggcacgccctatgtggc
ggcaaaaatcatccaggtcgccgaatggttcgccgaagagaaaaatctgaaagcagccatcaagaaagag
gccgaagctctgcacctcaagaccaagaaaacgattgaaggactgactgatgcgcaggtccacattctgc
tagaacgcaagtggattactcctttcattgatgaactccacggcctgcccgaccagcagattgatacctt
agtccgcaagctcgaagtcc
>id_3_ct_2_Ponta: b
gttgcggctacatcgccatcgtcggtcgtcccaacgttggcaagtcgaccctgactgaacgtgctggtcg
gcgccaaggtcagcatcacctcgcgcaaggcccagaccacgcgccaccgcatcaccggcatccagaccat
>id_4_ct_2_Ponta: g
caccgccgcccagcgtcgcgactgccggccttgctgctgctggaccgccttggccacggcttcgggcgcg
ggcatgggagacgccggcgcatgggcgtccacgtaggtcttcagttcatcgacggtgagcacccgctcgg
>id_5_ct_3_Ponta: b
acacgccggtgacgttggtgccggaagccttccagtccttcaccagctgggcggcgaccggatcggtcac
accggagtagaccaccgggatgctcttggtggcggccaccagggcctgggccgaggggagtggcgatggc
cacgatggcgtcgggcttgtcacccacgaacttgcgggcgatctgggcggcggtgccagtattgccctgg
gcgctctggtattcccacttcaggttcttgccggcttcgaagccttcttccttcaactcgtccttgacgc
```

FIGURA 7 - EXEMPLO DO CONTEÚDO DE UM ARQUIVO FASTA

FONTE: o autor (2010).

2.8.2 Estrutura de um arquivo texto no formato GenBank

As informações apresentadas a seguir foram compiladas do arquivo gbrel.txt[5].

Os arquivos com extensão gbk é a segunda parte de um conjunto de informações. A primeira parte, denominada de cabeçalho, é armazenada nos arquivos com extensão gff. Mas é na segunda parte que se encontram as informações que a comunidade científica procura, ou seja, as sequências genômicas e as respectivas anotações. Esse segundo arquivo é conhecido por "*entry*" e representa um registro ou "entrada" no banco de dados *GenBank*.

O arquivo GBFF (acrônimo adotado para *GenBank Flat File,* em inglês) possui uma divisão interna, identificada pela posição dos textos nas respectivas colunas, onde cada linha é composta por duas partes: a primeira parte se encontra nas posições de um a dez e pode conter:

[5] NCBI-GenBank Flat File Release 178.0 [ftp://ftp.ncbi.nih.gov/genbank/gbrel.txt]

a) a palavra-chave, começando na coluna um, identifica as grandes seções do arquivo. Exemplos: Locus, Reference, Features;
b) uma palavra-chave secundária, que inicia na terceira – logo após dois espaços em branco – indica uma subdivisão de uma palavra-chave maior, descrita no item anterior. Exemplo: author é uma subdivisão presente na seção Reference. Podem ocorrer exceções, como no caso de Pubmed (subdivisão de Reference), que inicia na terceira posição;
c) os dez primeiros caracteres em branco identificam que a informação contida a partir da décima-segunda coluna é continuação da informação contida na linha anterior;
d) um código, iniciado na coluna seis, indica a natureza de uma entrada – função ou característica – de uma região anotada;
e) um número, terminado na coluna nove, identifica a posição inicial da sequência de nucleotídeos apresentados a seguir;
f) duas barras (//) nas posições um e dois denotam o final de uma entrada;
g) a segunda parte se inicia na décima-terceira posição e termina na 80.ª coluna para as linhas que têm palavras-chaves e na décima-primeira posição e termina na 80.ª coluna para as sequências.

A seguir, são apresentadas a descrição dos campos de um registro do GBFF e as respectivas palavras-chave que os identificam em um arquivo texto:

- LOCUS: Identifica a primeira linha de uma entrada e um nome curto mnemônico que identifica a entrada, escolhido para sugerir a definição da sequência. Chave obrigatória, exatamente uma linha;
- DEFINITION: uma descrição concisa da sequência. Chave obrigatória, um ou mais linhas;
- ACCESSION: é uma chave de acesso única e que não se altera. Chave obrigatória, um ou mais linhas;
- VERSION: o campo versão é composto por duas unidades; à primeira unidade é composto pelo número de acesso, um ponto (.) e o número incremental da versão. A segunda unidade é um número de identificação interna do NCBI para a sequência, precedida pelo texto “GI:”. Chave obrigatória, uma linha;
- NID: tornou-se obsoleto em dezembro de 1999. Tratava-se de uma forma alternativa aos códigos GIs;

- PROJECT: um identificador para um projeto (identificar um projeto de sequenciamento de um genoma). Tornou-se obsoleto em abril de 2009;
- DBLINK: fornece um mecanismo para referência cruzada aos recursos de outros bancos de dados, tais como o NCBI Trace Assembly Archive. Chave opcional pode ter uma ou mais linhas;
- KEYWORDS: frases curtas descrevem os produtos dos genes ou outras informações sobre a entrada. Campo obrigatório, um ou várias linhas;
- SEGMENT: informações sobre a ordem em que esta entrada aparece em uma série de sequências descontínuas da mesma molécula. Chave opcional, exatamente uma linha;
- SOURCE: nome do organismo ou nome mais frequentemente visto na literatura. Chave obrigatória em todas as entradas das anotações ou no geral com uma ou mais linha.
- ORGANISM: Nome científico formal do organismo – posicionado na primeira linha – e níveis de classificação taxonomia – linhas subsequentes. Campo obrigatório, com duas ou mais linhas;
- REFERENCE: citações para todos os artigos que contêm dados informados dessa entrada (registro do GBFF). Inclui sete subdivisões e podem ocorrer repetições no arquivo. Chave obrigatória com uma ou mais linhas;
- AUTHORS: lista dos autores da citação. Campo opcional pode ter uma ou mais linhas;
- CONSRTM: lista os nomes dos consórcios associados com a citação. Chave opcional;
- TITLE: título completo da citação. Opcional e pode ter uma ou mais linhas;
- JOURNAL: o nome do periódico, volume, ano e páginas. Chave obrigatória com uma ou mais linhas;
- MEDLINE: fornece o identificador exclusivo de um MEDLINE da citação. Campo opcional – uma linha. Campo obsoleto, removido do GBFF em abril de 2005;
- PUBMED: fornece o identificador único do PubMed da citação. Campo opcional pode ter uma ou mais linhas;
- REMARK: especifica a importância de uma citação para uma anotação. Campo opcional com uma ou mais linhas;

- COMMENT: referências cruzadas com outros registros de anotações, comparações com outras coleções, alterações nos nomes dos locus etc. Chave opcional com uma ou mais linhas, podendo ter linhas em branco;
- FEATURES: tabela com as informações sobre partes das sequências que codificam proteínas e moléculas de RNA, informações sobre determinadas regiões de importância biológica obtidas de forma experimental. Chave opcional e pode ter uma ou mais linhas;
- BASE COUNT: resumo do número de ocorrência de cada par de bases na sequência. Campo opcional. Exatamente uma linha. Tornou-se obsoleto em outubro de 2003;
- CONTIG: este tipo de linha fornece informações sobre como cada sequência de linhas pode ser combinada para formar objetos biológicos em escala maior, tais como cromossomos ou genomas completos;
- ORIGIN: especifica a localização da primeira base da sequência relatada dentro do genoma. Campo obrigatório e exatamente com uma linha;
- a linha de ORIGIN é seguida pela sequência de nucleotídeos;
- // identifica o final do registro do GBFF. Campo obrigatório e ocupa exatamente uma linha.

Os termos (palavras-chave) podem apresentar subdivisões e detalhamentos descritos em *Features Table*, disponíveis no seguinte endereço da web: http://www.ncbi.nlm.nih.gov/projects/collab/FT/index.html.

Para D'addabbo (2004) e Lee (2008), o formato de arquivo texto do GenBank é popular e possui distribuição, acesso e manutenção facilitados, porém a extração de informação e a análise exigem o uso de bibliotecas e linguagens de programação.

2.9 DESEMPENHO DE LEITURA E ESCRITA EM JAVA

O acesso as informações nos dispositivos secundários de armazenamento (qualquer dispositivo diferente da memória RAM) possui a exigência de um tempo significativamente maior. Na linguagem Java não é diferente; existem diversas classes e muitos métodos, cujo uso pode promover um maior ou menor tempo de resposta. No artigo Tuning Java *I/O Performance,* de Glen McCluskey (1999), é apresentada uma discussão ampla sobre o tema. Para este estudo, destacam-se:

1. Linhas gerais

> ***Basic Rules for Speeding Up I/O***
> *As a means of starting the discussion, here are some basic rules on how to speed up I/O:*
> - *Avoid accessing the disk.*
> - *Avoid accessing the underlying operating system.*
> - *Avoid method calls.*
> - *Avoid processing bytes and characters individually.*
>
> *These rules obviously cannot be applied in a "blanket" way, because if that were the case, no I/O would ever get done! But to see how they can be applied, consider the following three-part example that counts the number of newline bytes ('\n') in a file.* (MCCLUSKEY,1999)

2. *“Caching”*

> *A detailed discussion of hardware caching is beyond the scope of this paper. But sometimes software caching can be used to speed up I/O. Consider a case where you want to read lines of a text file in random order. One way to do this is to read in all the lines, and store them in an ArrayList (a collection class similar to Vector)*
> (MCCLUSKEY,1999)

3. *“Buffering”*

> *Approaches 2 and 3 use the technique of buffering, where large chunks of a file are read from disk, and then accessed a byte or character at a time. Buffering is a basic and important technique for speeding I/O, and several Java classes support buffering (BufferedInputStream for bytes, BufferedReader for characters).*
>
> *An obvious question is: Will making the buffer bigger make I/O go faster? Java buffers typically are by default 1024 or 2048 bytes long. A buffer larger than this may help speed I/O, but often by only a few percent, say 5 to 10%.*
> (MCCLUSKEY,1999)

Em consequência deste estudo, a classe BufferedReader foi recomendada e amplamente empregada nas aplicações deste trabalho.

2.10 PROTOCOLO DE TRANSFERÊNCIA DE ARQUIVOS E BIBLIOTECAS JAVA

Para o *World Wide Web Consortium*, o protocolo de transferência de arquivos tem por objetivos: promover o compartilhamento de arquivos; encorajar de forma indireta ou implícita o uso remoto de computadores; proteger usuários da variação dos sistemas de arquivos dos servidores; transferir dados de forma confiável e segura e embora possa ser utilizado diretamente por um usuário em um terminal, é

projetado principalmente para o uso por programas (WORLD WIDE WEB CONSORTIUM, 2010, RFC959[6]).

As linguagens de programação em geral oferecem funções em suas bibliotecas de baixo nível para acesso e comunicação entre computadores. Se for avaliado o conjunto de funções em relação ao modelo OSI (*Open Systems Interconnection,* em inglês) do padrão definido pelo ISO (*International Organization for Standardization*, em inglês), observa-se que das sete camadas (camada física, camada de enlance, camada de rede, camada de transporte, camada de sessão, camada de apresentação e camada de aplicação), as funções atendem até a sexta camada. O conjunto de funções para atender os protocolos, mesmo padronizados, previstos para a camada de aplicação (sétima camada), normalmente está incompleto ou ausente das APIs das linguagens. Alguns protocolos previstos nesta camada são: HTTP, SMTP, FTP, SSH, RTP, Telnet, SIP, RDP, IRC, SNMP, NNTP, POP3, IMAP, DNS, entre outros.

A linguagem Java tem suporte completo para a especificação RFC 1738[7] (*Uniform Resource localizadores* - URL), porém não coloca à disposição nenhum recurso da RFC959 (*File Transfer Protocol*).

Em busca de comparação entre as bibliotecas disponíveis, foram encontrados no sítio do JavaWorld dois artigos sobre o assunto: *Java FTP client libraries reviewed* (NORGUET, 2003); *Update: Java FTP libraries benchmarked* (NORGUET, 2006[8]), publicado no site especializado JavaWorld.. A Tabela 4 a seguir foi adaptada dos artigos.

As bibliotecas avaliadas foram:

a) Oracle/Sun JDK (GOSLING et al. 2010);
b) Jakarta Commons/Net (APACHE, 2010);
c) JScape Secure FTP Factory (JSCAPE, 2010);
d) EnterpriseDT FTPj (ENTERPRISE DISTRIBUTED TECHNOLOGIES, 2010);
e) SourceForge JFtp (SOURCE FORGE, 2010);
f) Florent Cueto Java FTP API (FRESHMEAT, 2010);
g) Bea Petrovicova jvFTP, (SOURCE FORGE, 2010);

[6] Disponível em: http://www.w3.org/Protocols/rfc959/
[7] Disponível em http://www.w3.org/Addressing/rfc1738.txt
[8] Disponível em: http://www.javaworld.com/javaworld/jw-03-2006/ftp/jw-0306-ftptable.html

h) The Globus Project – Java CoG Kit (LASZEWSKI[9], 2001), e

i) Glub Tech – Secure FTP Bean (GLUB TECH, 2010).

TABELA 4 - COMPARAÇÃO DAS BIBLIOTECAS DE CLIENTE FTP PARA JAVA

Biblioteca	**JDK (a)**	**Jakarta (b)**	**JScape (c)**	**EnterpriseDT (d)**	**JFtp (e)**	**Florent (f)**	**Bea (g)**	**Globus (h)**	**Glub (i)**
Versão avaliada	1.2.2	1.4.1	5.4	1.5.2	1.47	2.6	0.72	1.2	2.5.3
Versão avaliada anteriormente	1.2.2	1.0.0	4.2	1.2.1	1.07	2.6	0.70	1.0a	n/a
Suporte									
Javadoc	S	S	S	S	S	-	S	S	S
Exemplos de códigos	-	S	S	S	S	S	S	S	S
Fórum	S	-	-	S	S	-	-	-	S
Lista de e-mails	-	S	S	S	S	-	-	S	S
Apoio por e-mail	-	-	S	S	US $200	S	S	S	S
Sistema de acompanhamento de falhas	S	S	-	-	S	-	-	S	S
Base de conhecimento on-line	S	S	S	-	-	-	-	-	-
Formulário da equipe de desenvolvimento	-	G	P	P	GP	I	I	P	P
Licença	L	L	C	L/C	G/C	G	-	L	C
Fontes disponíveis			US $599						US $250
Tratamento de erros	S	S	S	S	S	-	S	S	S
Características									
Analisador de diretório	-	S	S	S	S	S	S	S	s
Analisador de data/hora	-	S	S	S	S	S	S	-	s
Suporta Analisador do usuário	-	P	S	C	S	P	P	P	s
Suporte ao comando MDTM	-	-	S	S	-	-	-	S	s
Suporta Proxy	-	S	S	S	S	-	-	-	S
Suporta túnel HTTP	-	-	-	-	-	-	-	-	-
Conexão ativa/passiva	S	S	S	S	S	S	S	S	s
Transferência									
Baixar/receber	S	S	S	S	S	S	S	S	s
Enviar (upload)	S	S	S	S	S	S	S	S	s
Transferências simultâneas	-	S	S	-	S	S	-	-	s
Suporte de monitoramente	-	S	S	S	S	S	S	S	s
Transferências múltiplas (wildcard - *)	-	-	S	US $299	-	-	S	-	-
Transferência recursiva de diretório	-	-	S	US $299	S	-	S	-	-
Retomada de transferência	-	S	S	S	S	S	-	S	S

FONTE: o autor, com base em NORGUET (2003; 2006).

[9] Conforme orientações contidas em http://wiki.cogkit.org/wiki/Main_Page

Onde:

- Suporte do Produto (I – voluntário individual, G – grupo de voluntários e P – entidade profissional, serviço cobrado;
- Licenças: C – Comercial, G – GPL, GNU General Public License e L – livre;
- Suporta Analisador do usuário: S – por meio de um método adicional, obtém textos simples de resposta. C – por meio de um método adicional que retorna uma coleção de textos. P – analisadores plugáveis.

As bibliotecas da Fundação Apache são livres de licenciamento, permitem o uso em projetos de código-aberto e/ou comerciais, possuem boas referências nos sítios de discussão e algumas fazem parte das especificações de referência do JCP (como o TomCat etc). Dentre as avaliadas, optou-se pela adoção dessa biblioteca.

2.11 ANALISADOR

No estudo de compiladores, em geral se associa ao conceito de sintaxe a ideia de forma, enquanto o significado e conteúdo ficam associados à semântica. Assim, a sintaxe de uma linguagem de programação tem a responsabilidade de especificar tudo que seja necessário para a escrita de um programa de forma correta e a semântica deve descrever o que acontece quando o programa é executado. Portanto, as análises sintáticas e semânticas devem assegurar que o código objeto (ou de máquina) produzido preserve o significado do programa fonte.

Por questões de evolução ou praticidade, a análise (*parser*[10]) se divide em análise léxica, análise sintática e análise semântica. A análise léxica (AL) tem por objetivo a separação e identificação dos símbolos do programa. A análise sintática deve verificar as regras de composição, reconhecer a estrutura do programa. A análise semântica verifica se os aspectos semânticos do programa estão corretos, ou seja, verifica a coerência quanto ao significado das construções feitas pelo programador.

[10] Parser – termo técnico amplamente empregado em informática. Pode-se adotar “analisador” como tradução para o termo. Nome aplicado ao conjunto de análises realizadas durante as etapas iniciais dos compiladores de linguagens de programação.

As explicações foram dadas no contexto de linguagens de programação, mas se aplicam ao contexto de análises de expressões, análises de linguagens marcadoras ou de arquivos com formatos pré-estabelecidos.

Um analisador é um programa, ou parte de um compilador, que tem por objetivo quebrar um texto de entrada em partes menores e classificar essas partes segundo a gramática de uma linguagem (DELAMARO, 2004, p. 37).

2.12 O ANALISADOR DE ARQUIVOS GENBANK

Stajich et al. (2002) desenvolveram a biblioteca BioPerl (BIOPERL, 2010) para a linguagem Perl com recursos para ler e escrever arquivos no formato GenBank. Holland et al. (2008) desenvolveram uma biblioteca para a linguagem Java (BIOJAVA, 2010) que possui funções para analisar diversos formatos de arquivos, inclusive para o GenBank. Cock et al. (2009), desenvolveram a biblioteca BioPython (BIOPYTHON, 2010) contendo recursos para análise de arquivos no formato GenBank.

Lee et al. (2008) afirma que o formato (GenBank) tem sido amplamente adotado para descrever não apenas uma sequência genética relativamente curta, mas também sequências longas, como as dos genomas de eucariotos. Para usar os dados no arquivo de um computador, um processo de análise (*parser*) se faz necessário e é realizado em conformidade com uma gramática (um padrão ou especificação) descrita pelo NCBI GenBank. No mesmo artigo, os autores apresentam as seguintes considerações:

> ***Therefore, the GBF parser has become a routine program in bioinformatics***. [grifo nosso] *NCBI C/C++ toolkit [6] provides a lot of functions to deal with a sequence including parsers for ASN.1 format since NCBI employed the format as a central data format,* ***but the toolkit does not provide a parser for the GBF***. [grifo nosso] *Thus, currently, several parser libraries for the GBF have been developed by other groups, such as BioPython [7], BioPerl [8] and the AJAX library in the EMBOSS package [9].*
>
> ***However, parsing a large GBF file (such as a eukaryotic genome sequence) with these libraries inevitably takes a long time since the parser libraries were not designed just for parsing speed.*** [grifo nosso] *For example, parsing time for the GBF of a chromosomal sequence of Arabidopsis thaliana by the GBF parser of BioPython is estimated to be 109.2 seconds (Table 1).*
>
> ***Thus, the biological community needs a faster parser that can parse a large GBF file, such as a eukaryotic chromosome, in a single-digit***

> ***number of seconds at the personal computer level***[grifo nosso]***.*** *(LEE et al., 2008).*

Destacam-se, respectivamente, a utilidade do “analisador” para a área da bioinformática; a inexistência de uma função com essa finalidade nas ferramentas disponíveis no NCBI, a ineficiência das opções existentes nas bibliotecas existentes e a necessidade de que a comunidade tenha essas técnicas desenvolvidas para alto desempenho.

2.13 CAMADA DE ACESSO AO BANCO DE DADOS

As linguagens de programação normalmente dispõem de funções para acessar e escrever arquivos, as quais formam a API entrada e saída de baixo nível. Pode-se considerar essas funções como o primeiro modelo de acesso aos dados da era da computação. As linguagens Basic, C, Pascal fazem parte da segunda geração; ou seja, uma geração de linguagens que não tinha funções em suas bibliotecas originais para acesso a banco de dados.

As linguagens de terceira geração ou de alto nível, tais como o DBase®, o Clipper® e o Fox® representam um conjunto de linguagens que tinham as funções de baixo nível e também um conjunto de alto nível. Essas funções de alto nível possibilitavam o gerenciamento e o emprego de tabelas e índices sem a necessidade de conhecer à estrutura e ao funcionamento interno, bem como à manipulação de registros de forma transparente.

Os sistemas gerenciadores de banco de dados oferecem ferramentas e linguagens próprias para acessos aos seus recursos. A Oracle, por exemplo, fornecia a linguagem PL e o Forms para desenvolver aplicações clientes do banco de dados. Até essa época, o fabricante da linguagem ou do banco era também o fornecedor dos recursos para desenvolvimento e acesso aos bancos de dados.

O padrão de Conectividade de Banco de Dados Aberta (Open Database Connectivity – ODBC)

> [...] define uma maneira para um programa de aplicação se comunicar com um servidor de banco de dados. O ODBC define uma interface de programa de aplicação (API) que as aplicações podem usar para abrir uma conexão com um banco de dados, enviar consultas e atualizações e trazer resultados (SILBERSCHATZ, 2006, p. 90).

Com a evolução da linguagem SQL e a criação do ODBC, pela Microsoft, iniciou-se uma fase de separação da camada de acesso ao banco de dados por parte das linguagens de programação. O ODBC se tornou um padrão para os fabricantes de SGBD desenvolverem suas bibliotecas para que os programadores pudessem aproveitá-las nas diferentes linguagens de programação.

Com a mudança do paradigma de programação estruturada para o orientado objeto, criou-se um vácuo de incompatibilidade entre as linguagens e os SGBD. No lado das linguagens orientadas a objetos, (OO) não existiam mais os recursos para representar estruturas de dados (como os registros) e no lado dos SGBD a migração para o modelo OO ainda não ocorreu. Apesar de existir SGBD OO, o mercado mantém os Relacionais principalmente por questões de desempenho e segurança e alguns fabricantes, como a Oracle, afirmam que seus bancos são SGBD Objetos-Relacional (SGBDOR).

Esse vácuo tem sido preenchido com a linguagem SQL, de modo que nos sistemas de informação criou-se o conceito de camada de persistência. Essa camada é constituída por um conjunto de funções que tem a responsabilidade de “preservar” e “recuperar” os objetos. Em geral, essas operações são realizadas por meio da conversão das informações disponíveis nos objetos em instruções SQL e vice-versa.

2.13.1 JDBC

De acordo com a Oracle, o Java Database Connectivity (JDBC) é composto por dois conjuntos de interfaces. O primeiro se destina a desenvolvedores de aplicações Java e o segundo, a fabricantes de SGBDs, o que permite às aplicações o acesso aos bancos de dados, como é demonstrado nas FIGURAS 8 e 9.

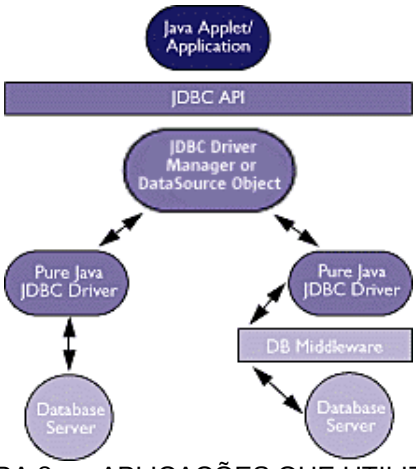

FIGURA 8 - APLICAÇÕES QUE UTILIZAM JDBC DOS TIPOS 3 E 4.
FONTE: ORACLE, http://japa.sun.com/productsjdbc/overview.html

Left side, Type 4: Direct-to-Database Pure Java Driver
This style of driver converts JDBC calls into the network protocol used directly by DBMSs, allowing a direct call from the client machine to the DBMS server and providing a practical solution for intranet access.

Right side, Type 3: Pure Java Driver for Database Middleware
This style of driver translates JDBC calls into the middleware vendor's protocol, which is then translated to a DBMS protocol by a middleware server. The middleware provides connectivity to many different databases.

The graphic below illustrates JDBC connectivity using ODBC drivers and existing database client libraries.

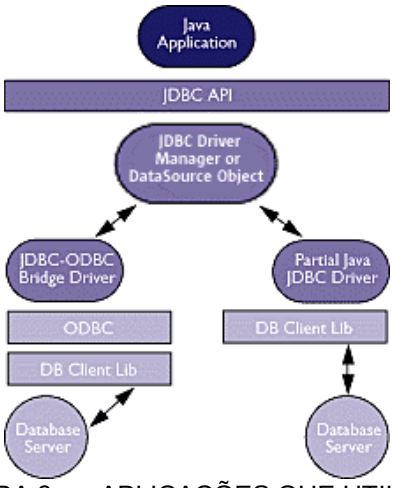

FIGURA 9 - APLICAÇÕES QUE UTILIZAM JDBC DOS TIPOS 1 E 2
FONTE: ORACLE, http://:java.sun.com/products/jdbc/overview.html

> *Left side, Type 1: JDBC-ODBC Bridge plus ODBC Driver*
> *This combination provides JDBC access via ODBC drivers. ODBC binary code -- and in many cases, database client code -- must be loaded on each client machine that uses a JDBC-ODBC Bridge. Sun provides a JDBC-ODBC Bridge driver, which is appropriate for experimental use and for situations in which no other driver is available.*
>
> *Right side, Type 2: A native API partly Java technology-enabled driver*
> *This type of driver converts JDBC calls into calls on the client API for Oracle, Sybase, Informix, DB2, or other DBMS. Note that, like the bridge driver, this style of driver requires that some binary code be loaded on each client machine. (ORACLE, http://java.sun.com/products/jdbc/overview.html)*

Atualmente, a especificação do JDBC está na quarta versão e é identificada pelo Pedido de Especificação Java (JSR - *Java Specification Requests*, em inglês) 221 no Processo da Comunidade Java (JCP - Java Community Process, em inglês). O JCP é um mecanismo criado para a comunidade de desenvolvedores e os fabricantes decidirem sobre a evolução da linguagem.

2.13.2 JPA

A *Java Persistence API* (JAVA COMMUNITY PROCESS, 2010) é um conjunto de interfaces responsável em automatizar o processo de persistência de objetos em banco de dados relacional, a conversão ocorre por diversos mecanismos de mapeamento Objeto/Relacional. Os desenvolvimentos das bibliotecas que programam essas interfaces não são fornecidos pela Oracle/Sun, de modo que as bibliotecas mais conhecidas e usadas são o Hibernate, TopLink (Oracle) e EclipseLink. Esse recurso foi introduzido no Java na versão 5 e atualmente se encontra na segunda versão da especificação (JSR 317[11]). Essa API é considerada de alto nível e adota internamente os recursos do JDBC antes já discutidos.

Seguindo o modelo do *Enterprise Java Beans* (EJB), as classes que possuem o mapeamento para as tabelas dos SGBD Relacionais são denominadas “entidades”. Apenas objetos de classes entidades podem ser persistidos com o uso da JPA.

[11] A especificação pode ser encontrada no endereço da internet:
http://www.jcp.org/en/jsr/detail?id=317

2.13.3 Otimizações na camada de persistência

A camada de persistência é o conjunto de funções, rotinas, métodos, classes e outros artefatos dos programas responsáveis pela interação com o banco de dados, realizando a conversão de informações contidas nos Objetos para comandos estruturados e vice-versa.

Em informática, o termo “otimização” é empregado para designar os procedimentos, técnicas e métodos que conseguem realizar uma mesma tarefa em menor tempo e/ou menor consumo de recursos.

Em Java “acontece em alguns casos de a velha API JDBC ser a melhor solução para a persistência. Ou mesmo a única.” (DOEDERLEIN, 2005). Porém, segundo o autor, a camada de persistência automática tem algumas vantagens:

> - há muito menos código para escrever;
> - podemos contar com um cache de memória dos POJOs, que elimina execuções repetitivas de consultas que retornam os mesmos objetos;
> - podemos ter facilidades de locking automático, otimista ou pessimista;
> - o gerador automático de consultas pode gerar código otimizado para cada SGBD;
> - recursos importantes de orientação a objetos são mapeados para o mundo relacional, por exemplo usando tabelas diferentes para cada camada de herança e gerando joins para ler os objetos(DOEDERLEIN, 2005.)

Apresenta também duas desvantagens:

> - qualquer consulta que não seja trivial e que não seja mapeável; diretamente para um modelo orientado a objetos não será suportada pelo sistema de persistência;
> - mesmo para as operações que são facilmente mapeáveis para um modelo OO, podemos ser obrigados a usar consultas ad hoc por motivo de desempenho.(DOEDERLEIN, 2005).

A API JDBC prevê duas classes para envio de comandos para o servidor de banco de dados, *Statement* e *PreparedStatement,* de modo que a segunda é uma especialização da primeira. Conforme Doederlein (2005), entretanto, a classe *PreparedStatement* resolve três problemas:

> **- Simplifica a representação dos parâmetros**: não é necessário envolver strings com aspas nem converter datas para a sintaxe aceita pelo banco de dados (...).
> **- Aumenta o desempenho**: a execução é mais eficiente porque o driver e/ou banco de dados processam o texto SQL uma só vez, fazendo cache da sua representação compilada. (...)

> **- Suporta updates em batch:** o **PreparedStatement** permite executar updates repetitivos, que variam somente pelos parâmetros, com os métodos addBatch() e executeBatch(). (DOEDERLEIN, 2005).

Esse último recurso é apontado pelo autor como uma opção para quando se precisa ler ou atualizar um grande volume de dados de uma só vez.

Ainda são recomendados pelo autor, para garantir melhores desempenhos, a atualização do *driver* (DOEDERLEIN, 2005) JDBC, o uso de sintaxe proprietária no lugar da sintaxe padrão do ANSI SQL e a adoção de técnicas de “cache” na camada DAO.

2.14 DESIGNER PATTERNS

A real origem do termo e do conceito é controversa, porém os autores que organizaram, classificaram e sistematizaram as “boas práticas” de desenvolvimento Orientado a Objetos foram Erich Gamma, Richard Helm, Ralph Johnson e John M. Vlissides, no livro *Design Patterns: Elements of Reusable Object-Oriented Software,* publicada pela Addison-Wesley Professional, em 1994. Por esse trabalho, os quatro autores ficaram conhecidos e são referidos Gang of Four (GoF). O trabalho do GoF reuniu 23 padrões, recomendações e boas práticas para problemas comuns encontrados no dia-a-dia de todo programador.

Os autores apresentam o conceito e a definição de “padrões de projetos” por meio de uma comparação com os padrões da arquitetura e construção civil.

> Christopher Alexander afirma: “cada padrão descreve um problema no nosso ambiente e o cerne da sua solução, de tal forma que você possa usar essa solução mais de um milhão de vezes, sem nunca fazê-lo da mesma maneira” [AIS*77, pág.x]. Muito embora Alexander estivesse falando acerca de padrões em construções e cidades, o que ele diz é verdade em relação aos padrões de projeto orientado a objeto. Nossas soluções são expressas em termos de objetos e interfaces em vez de paredes e portas, mas no cerne de ambos os tipos de padrões está a solução para um problema num determinado contexto.
> [...]
> Padrões de projeto, neste livro, são **descrições de objetos e classes comunicantes que precisam ser personalizadas para resolver um problema geral de projeto num contexto particular**.(GAMMA et al., 2000, p. 19).

Ainda de acordo com os autores, a justificativa para utilizar padrões de projeto é a facilidade de reaproveitamento dos projetos e arquiteturas bem-sucedidas (GAMMA et al., 2000, p. 18).

Os 23 padrões são: Adapter; Facade; Composite; Bridge; Singleton; Observer; Mediator; Proxy; Chain of Responsibility; Flyweight; Builder; Factory Method; Abstract Factory; Propotype; Memento; Template Method; State; Strategy; Command; Interpreter; Decorator; Iterator e Visitor.

A FIGURA 10 apresenta as relações existentes entre os diversos padrões de projeto, conforme descrito pela GoF.

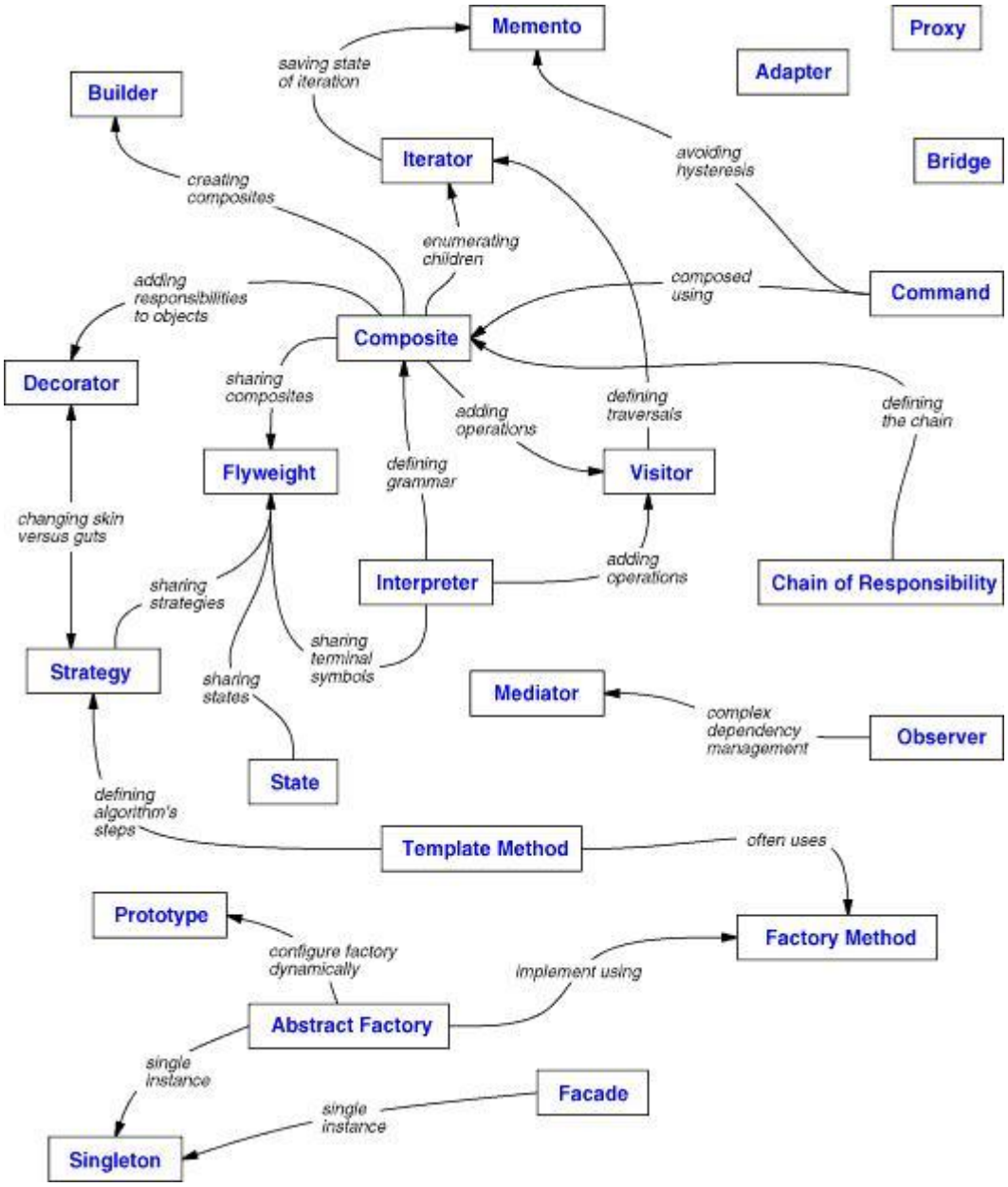

FIGURA 10 - RELACIONAMENTO ENTRE PADRÕES DE PROJETO

FONTE: GAMMA et al. (2000, p. 27)

Neste trabalho, foram considerados os padrões de projeto Fábrica Abstrata (Abstract Factory), métodos fábrica (Method Factory), Singleton e DAO.

2.14.1 ABSTRACT FACTORY

O padrão de projeto *Abstract Factory* se destina a procedimentos de criação de objetos, pois permite que a aplicação delegue a criação dos objetos para um método utilitário. GAMMA et al. (2000, p.95) descrevem o padrão como:

> **Intenção**
> Fornecer uma interface para criação de famílias de objetos relacionados ou dependentes sem especificar suas classes concretas. (GAMMA et al., 2000, p.95-97).

Os autores ainda relacionam os diversos padrões de projeto e fazem recomendações para sua implementação:

> **Padrões relacionados**
> As classes AbstractFactory são frequentemente implementadas com métodos-fábrica (Factory Method), mas elas também podem ser implementadas usando Prototype.
> Uma fábrica concreta é frequentemente um singleton.(GAMMA et al., 2000, p. 103).

2.14.2 Método-fábrica

O padrão de projeto método-fábrica (Factory Method) é um padrão de projeto a ser aplicado nos procedimentos de criação de objetos, permitindo adiar a decisão sobre qual classe concreta instanciar. GAMMA et al. (2000, p.95) descrevem o padrão como:

> **Intenção**
> Definir uma interface para criar um objeto, mas deixar as subclasses decidirem que classe instanciar. O Facoty Method permite adiar a instanciação para subclasses. (GAMMA et al., 2000, p.112-113).

2.14.3 Singleton

O padrão de projeto Singleton propõe mecanismos para assegurar que exista na aplicação apenas um objeto de uma determinada classe. Gamma et al. (2000, p.95) descrevem o padrão como:

> **Intenção**
> Garantir que uma classe tenha somente uma instância e fornecer um ponto global de acesso para a mesma. (GAMMA et al., 2000, p.130).

2.14.4 DAO

Um *Data Acess Object* (DAO) é usado para abstrair e encapsular o acesso à fonte de dados. O DAO gerencia a conexão com a fonte para a obtenção das informações e armazenamento dos dados. O DAO programa um mecanismo de acesso necessário para trabalhar com a fonte de dados. A fonte de dados pode ser um armazenamento persistente como um SGBD, um serviço externo (web-service), um repositório LDAP entre outros.

Para Doederlein (2005), a mais importante evolução a partir da programação JDBC pura foi o padrão de projeto DAO (Data Access Object). A motivação desse padrão é organizar melhor o código de persistência, tornando-o reaproveitável e desacoplado da lógica de negócios.

Conforme a SUN, a implementação do padrão DAO pode ser muito flexível, caso se adapte o Abstract Factory [GoF] e Factory Method [GoF] (ORACLE, 2010).

2.15 VISÃO GERAL DO PROCESSO DE DESCOBERTA DE CONHECIMENTO EM BASES DE DADOS

Goldschmidt e Passos (2005) afirmam que

> [...] os projetos de pesquisa, tais como missões espaciais da Nasa e o Projeto do Genoma Humano, têm alcançado proporções gigantescas. (...) A análise de grandes quantidades de dados pelo homem é inviável sem o auxílio de ferramentas computacionais apropriadas. (GOLDSCHMIDT e PASSOS (2005, p. 1).

Os mesmos autores concluem que

> [...] torna-se imprescindível o desenvolvimento de ferramentas que auxiliem o homem, de forma automática e inteligente, na tarefa de analisar, interpretar e relacionar esses dados para que se possa desenvolver e selecionar estratégias de ação em cada contexto de aplicação. (idem)

Goldschmidt e Passos (2005) explicam que descoberta de conhecimento em bases de dados (KDD – *Knowledge Discovery in Database*, em inglês) é uma nova

área do conhecimento e que o termo KDD foi formalizado em 1989 em referência ao conceito de buscar conhecimento a partir de bases de dados. E acrescentam:

> Uma das definições mais populares foi proposta em 1996 por um grupo de pesquisadores (FAYYAD et al., 1996a): “KDD é um processo, de várias etapas, não trivial, interativo e iterativo, para a identificação de padrões compreensíveis, válidos, novos e potencialmente úteis a partir de grandes conjuntos de dados”. (GOLDSCHMIDT e PASSOS, 2005, p. 3).

Ainda, de acordo com os autores, o KDD é um processo que pode ser representado por três etapas: pré-processamento, mineração de dados e pós-processamento. Na etapa de pré-processamento ocorre a obtenção, organização e tratamentos dos dados. Na segunda etapa, de mineração de dados, ocorre a busca efetiva por conhecimentos; e na terceira e última etapa, de pós-processamento, se verifica o tratamento do conhecimento obtido, nem sempre necessário (GOLDSCHMIDT e PASSOS, 2005).

Neste trabalho, caracterizam-se os pesquisadores das áreas biológicas e da bioinformática como elementos centrais do processo de KDD, na expectativa representada na FIGURA 11, cujos objetivos são analisar e ver os dados na esperança de ter um “estalo” que o leve à solução do problema.

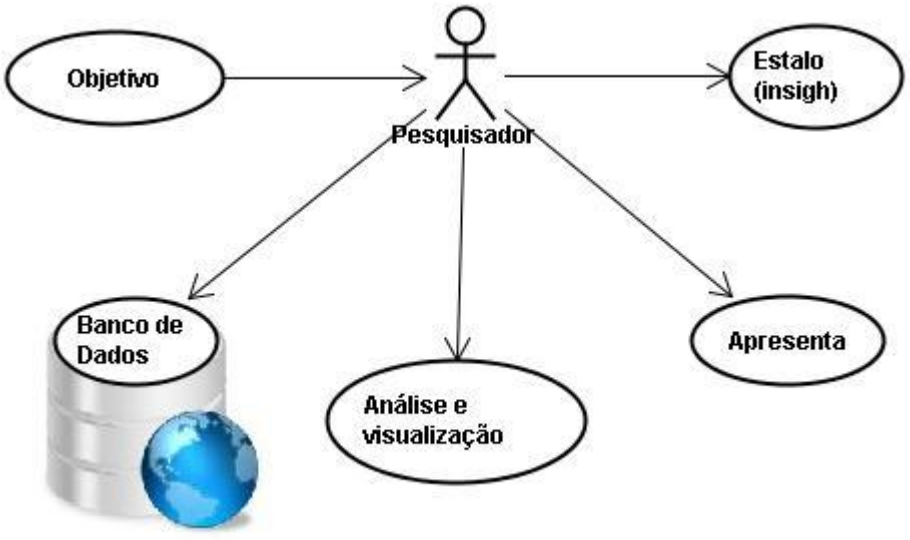

FIGURA 11 - O PESQUISADOR NO CENTRO DO PROCESSO DE DESCOBERTA DE CONHECIMENTO

FONTE: o autor (2010).

2.15.1 Extração, transformação e carga (ETL)

Conforme Kimball (2010), o processo de extrair, transformar e carregar (ETL - *Extract, Transform and Load*, em inglês) é o que consome mais de 70% do esforço e tempo que leva à construção de um armazém de dados (*data warehouse*, em inglês). Para o autor, todos compreendem o sentido das três letras, onde se recebe os dados (E), faz-se alguma coisa com ele (T) para depois carregá-lo (L). Ainda de acordo com Kimball, "*I have spent the last 18 months intensively studying ETL practices and ETL products. I have identified a list of 38 subsystems that are needed in almost every data warehouse back room*" (KIMBALL, 2010).

O processo de extração dos dados pode ser descrito como as ferramentas aplicadas ou processos, para obter os dados de fontes diferentes ou de formatos de arquivos diferenciados.

No processo de transformação, trata-se dos procedimentos para padronizar um conjunto de informação, adequar os tipos de dados e eliminar redundância do sistema. Por sua vez, carga é o processo de carregar os dados, normalmente, em um banco de dados.

2.16 TRABALHOS CORRELATOS DE BANCO DE DADOS DE GENOMAS

Em estudos preliminares realizados, foram identificados trabalhos do BioSQL, GUS (*The Genomics Unified Schema*, em inglês), BioWarehouse, GeneRecords, "*Design and Implementation of a Simple Relational Database for GenBank-Derived Data*" e a proposta de Nathan Mann, que apresentam propostas semelhantes aos objetivos desta dissertação. Apresenta-se a seguir uma análise dos trabalhos identificados na literatura e nos principais sítios de pesquisa da internet.

2.16.1 BioSQL

Conforme o BioSQL (2010), o BioSQL foi projetado por Ewan Birney[12], líder de pesquisa do EBI, com o propósito de ser um banco de dados relacional para os dados do GenBank. Posteriormente, em 2001 tornou-se um projeto de colaboração

[12] Página pessoal de Ewan Birney: http://www.ebi.ac.uk/~birney/

entre os projetos BioPerl, BioPython, BioJava e BioRuby. O objetivo é construir um esquema suficientemente genérico para o armazenamento persistente de sequências, funções e anotação de uma forma que seja interoperável entre os projetos. Cada um dos projetos citados, possuem um conjunto de funções/métodos para acessar as tabelas do modelo do BioSQL.

Não foram identificadas publicações próprias desse trabalho, apenas citações em outros trabalhos que indicam o seu uso.

Os **aspectos positivos** verificados na análise foram:

- o BioSQL apresenta suporte (codificação própria) para os seguintes SGBD: PostgreSQL, MySQL, Oracle, HSQLDB, Apache Derby;
- o BioSQL tem disponível um documento no formato PDF contendo o diagrama entidade relacionamento, e
- utilizado com padrão dos projetos BioPerl, BioPython, BioJava e BioRuby, o que possibilita o emprego do mesmo modelo em ferramentas desenvolvidas nas diferentes linguagens.

Os **aspectos negativos** verificados na análise foram:

- não possui ferramenta de carga, fica delegado para os projetos co-participantes o desenvolvimento dos mesmos, o que infelizmente não ocorre na integralidade; ou seja, é parcialmente suportado pelos projetos;
- todos os termos (*features, qualifiers* e valores das anotações) são armazenados na tabela *Term*, o que facilita a busca por termos, porém isso amplia muito o número de registros em uma única tabela;
- os *location_qualifier_value* é uma tabela de ligação entre *location* e *term*, em que na realidade os *qualifiers_values* deveriam estar associados à tabela *seqfeature*, o que pode produzir dois problemas na base de dados: ausência dos rótulos em algumas localizações ou redundância de informação, e
- não foram identificados onde ficam armazenados vários campos previstos na especificação do GenBank, apenas as entradas de anotação, sequência e referência são largamente suportadas.

2.16.2 GUS

O GUS (The Genomics Unified Schema) é um esquema de banco de dados relacional associado a um conjunto de aplicações e bibliotecas para armazenar, integrar, analisar e apresentar as informações dos genomas. O GUS está preparado para manipular informações da genômica, expressão gênica, regiões de transcrição, proteômicas, entre outras. Os autores entendem que o projeto enfatiza os padrões de ontologia.

O projeto propôs uma modelagem diferenciada, de modo que o núcleo central do modelo é baseado no dogma da biologia molecular. Conforme a FIGURA 12, apresenta as entidades principais e suas relações são: um gene pode ter vários RNAs, um RNA pode dar origem a várias proteínas. O GUS também separa as anotações dos genes das anotações de RNAs.

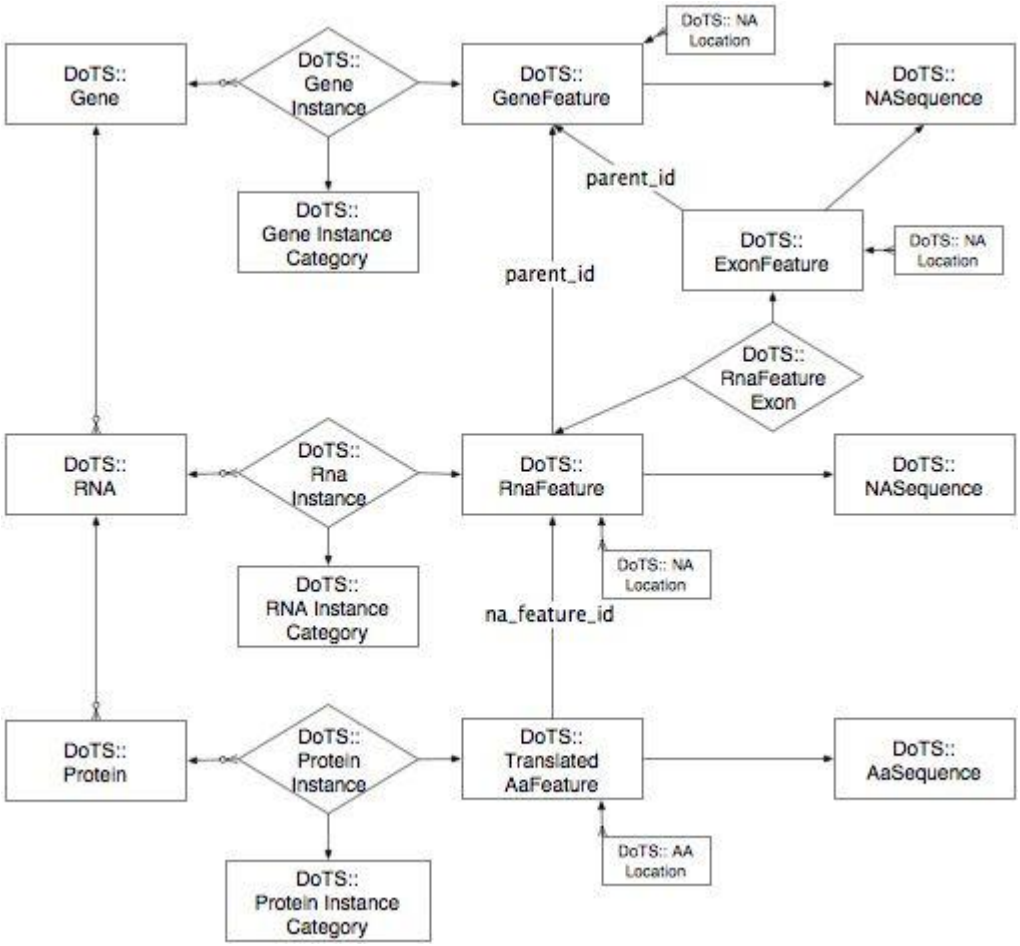

FIGURA 12 - DIAGRAMA ENTIDADE RELACIONAMENTO

FONTE: Internet. Documentação do projeto GUS.

O GUS foi desenvolvido em Java e Perl. Suporta os bancos de dados Oracle e PostgreSQL.

O site possui link para o diagrama entidade relacionamento geral, entretanto, não foi possível acessá-lo.

Os **aspectos positivos** verificados na análise foram:

- possui uma boa documentação disponível online;
- possui funções para integração com o Blast, para pesquisa de similaridade;

Os **aspectos negativos** verificados na análise foram:

- não foi identificado o analisador para arquivos no formato GenBank (GBFF);
- não possui atualizações recentes, a última ocorreu em 2005;
- não possui os códigos-fontes para criação das bases de dados, o dicionário está descrito em um arquivo XML, com mais de 1Mb;
- não tem o propósito de reproduzir todas as informações disponíveis nos arquivos GenBank no modelo relacional;
- o modelo das localizações é incompleto, de maneira que não é possível definir os operadores "*join*" previstos na especificação do GenBank.

Conforme os autores, os seguintes projetos utilizam o GUS:

> *Projects and Groups Using GUS*
> - *GeneDB -- Houses the pathogen sequences and annotation generated by PSU -- Pathogen Sequencing Unit (PSU), Wellcome Trust Sanger Institute.*
> - *TcruziDB -- A Trypanosoma cruzi genome resource located at the University of Georgia. This resource contains genomic, proteomic and expression analyses for T. cruzi, the causative agent of Chagas' disease. It is funded by the American Heart Association.*
> - *CryptoDB -- The Cryptosporidium genome resource is located at the University of Georgia. It contains the genomic sequence of two Cryptosporidium species. It is funded by NIH contract N01-AI-40037 and is a member of the ApiDB Bioinformatics Resource Center.*
> - *ApiDB -- A federated gateway to apicomplexan parasite genome resources located at the University of Pennsylvania and University of Georgia, funded by NIH contract N01-AI-40037.*
> - *Centromere Analysis System -- An investigation of centromere structure and function in plants. Stores sequence data produced locally, warehouses public data for analysis and store analysis results along with the primary sequence data -- Pruess Lab, University of Chicago and Terry Clark, University of Kansas.*
> - *Phytophthora Genome Project -- Releasing the 30,000 EST sequences and 8x genome sequences of P.sojae and provide complete annotation to the user community. It also provides an annotator's interface. -- Virginia Bioinformatics Institute (VBI) and the DOE Joint Genome Institute.*
> - *Microarray and QTL Dissection of Quantitative Resistance in Soybean Against P.sojae -- Analyzing the P.sojae infection and quantitative*

> *resistance in Soybean plants with microarray and QTL techniques. -- VBI and Ohio State.*
> - *Cross-kingdom comparative genomics of pathogens, including Phytophtora and malaria -- Currently the project is comparing the oxidative stress response in organisms from four different kingdoms, yeast (fungi), Arabidopsis (plants), Phytophthora (oomycetes) and malaria (apicomplexans) in order to learn rules for cross-kingdom comparative genomics. – VBI.*
> - *MLB-platform, and open soruce platform for environmental genomics -- Interfacing GUS to the ERFE-SVM system for predictive classification of microarray data. Using the GUS schema to provide the experimental microarray data descriptions and phenotype information to covariate with the visual and numerical procedures for subsample and outlier detection. GIS visualization and analysis will also be developed, based on our open source GIS GRASS system. -- MPBA group of the ITC-irst, Trento, Italy.*
> - *BiowebDB -- A comparative genomics and transcriptomics project aiming to complement GeneDB and TcruziDB, providing a function genomics resource for Trypansoma cruzi, Leishmania brasiliensis and Triatomine. -- Instituto Oswaldo Cruz.*
> - *Penn Bioinformatics Core Facility -- University of Pennsylvania.*
> - *John Maris -- Studying Array CGH data -- Children's Hospital of Pennsylvania.*
> - *PlasmoDB.org -- A genomic database for the malaria-causing parasite Plasmodium falciparum – CBIL.*
> - *EPConDB -- The Endocrine Pancreas Consortium web site – CBIL.*
> - *RAD -- A gene expression database of array-based (microarrays, high-density oligo arrays, macroarrays) and non-array-based (e.g., SAGE) experiments – CBIL.*
> - *Allgenes.org -- A human and mouse gene index derived from assembled ESTs and mRNAs – CBIL. (GUSDB, 2010).*

Conforme os autores do projeto BioWarehouse, as limitações do GUS são:

> *The GUS [11] project at the University of Pennsylvania has somewhat different goals than BioWarehouse. GUS is not designed for integration of public bioinformatics DBs, but is oriented toward implementation of bioinformatics applications that require integration of custom local data collections. Therefore, GUS does not provide loader tools for public bioinformatics databases. GUS emphasizes issues of data provenance and of detecting changes in the underlying data sources. GUS is implemented for the Oracle and Postgresql DBMSs. GUS has an extremely large schema (approximately 480 tables – see [41]) that is likely to limit its understandability by other groups and therefore their ability to query and extend GUS. GUS provides a publicly queryable DBMS server.(LEE et al., 2006).*

2.16.3 BioWarehouse

O BioWarehouse é um componente do projeto Bio-SPICE. É um projeto de código aberto, com o objetivo de integrar um conjunto de banco de dados biológicos em um único SGBD para a manipulação, mineração e exploração de dados.

Os autores indicam que as características do BioWarehouse são:

- *A **relational database schema** that models important bioinformatics datatypes.*
- *BioWarehouse instances can be implemented using either the **Oracle** or **MySQL** database management systems.*
- *A collection of **loader programs** that populate the warehouse with data from public biological databases.*
- *The loader programs transform the syntax of the source databases into relational form, and transform the diverse semantics of the source database into the common semantics of the BioWarehouse schema. (BIOWAREHOUSE, 2010[13]).*

A FIGURA 13 apresenta as principais relações entre os objetos do projeto.

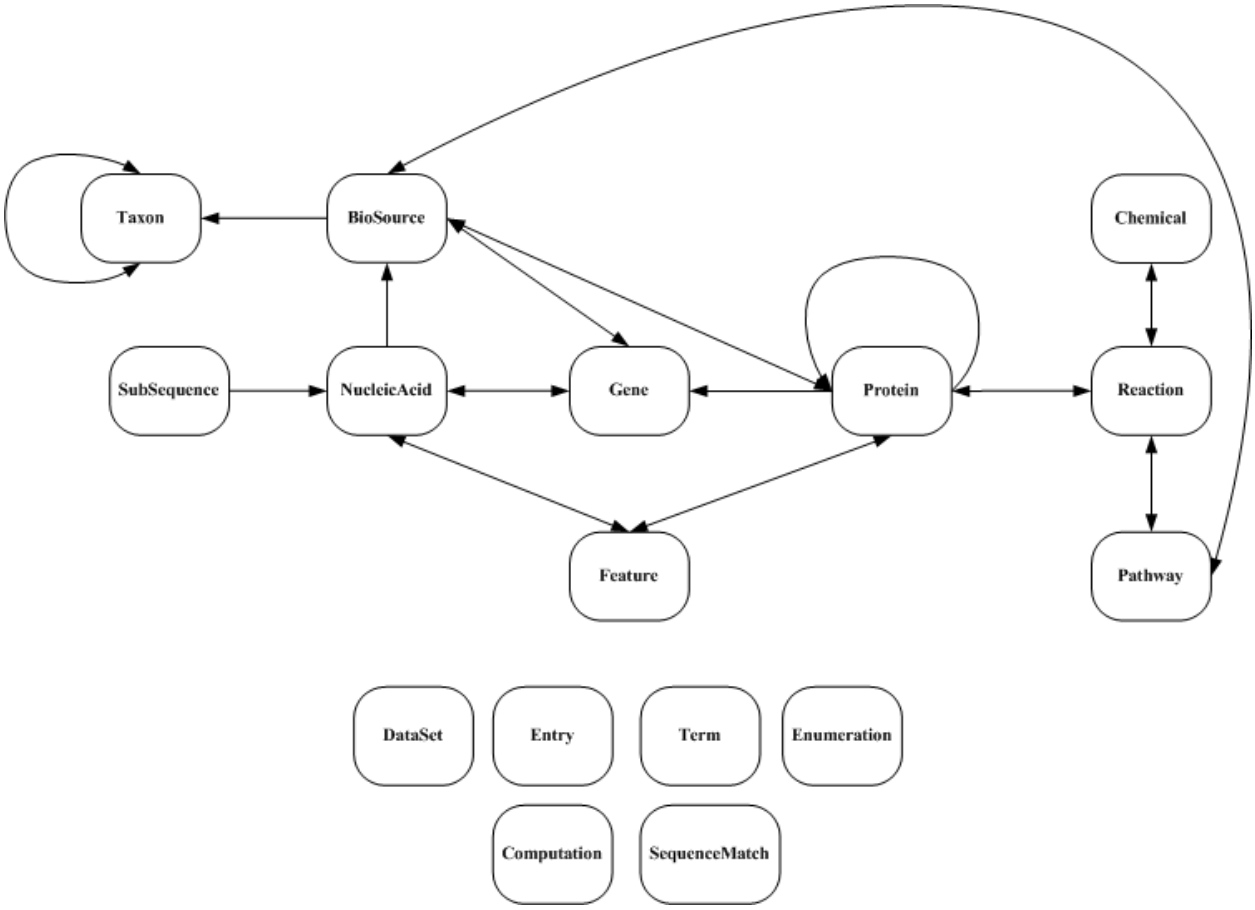

FIGURA 13 - RELAÇÃO ENTRE OS PRINCIPAIS OBJETOS DO BIOWAREHOUSE

FONTE: Internet. Documentação do projeto. LEE et al (2006).

[13] Disponível em: http://biowarehouse.ai.sri.com/

Os autores indicam para quais bancos de dados os carregadores de dados (*loaders*) foram desenvolvidos e em quais linguagens:

BioWarehouse loaders:

Source DB	*Contents*	*Language*
BioCyc DBs	*Genomes, genes, proteins, metabolic pathways, reactions, compounds*	*C*
BioPAX format	*BioPAX format describes biological pathway and protein interaction data. Currently this loader can process BioPAX Level 2 only -- protein interaction data.*	*JAVA*
Comprehensive Microbial Resource (CMR)	*Genomes, genes, proteins, reactions*	*C*
ENZYME DB	*Reactions, proteins*	*JAVA*
Eco2dbase	*E. coli 2D protein gel database*	*JAVA*
GenBank – bacteria only	*Bacterial genes and proteins*	*JAVA*
Gene Ontology	*A controlled vocabulary to describe gene and gene product attributes*	*JAVA*
Kyoto Encyclopedia of Genes and Genomes (KEGG)	*Genomes, genes, proteins, metabolic pathways, reactions, compounds*	*C*
MetaCyc Ontology	*The MetaCyc ontology of metabolic pathways, and the*	*C*
	MetaCyc ontology of chemical compounds	
MAGE-ML format	*The MAGE-ML file format describes gene expression datasets*	*JAVA*
NCBI Taxonomy DB	*Taxonomical organism classification*	*C*
UniProt (Swiss-Prot and TrEMBL)	*Protein knowledgebase*	*JAVA*

(BIOWAREHOUSE, 2010)

Os **aspectos positivos** verificados na análise foram:

- possui uma boa documentação, disponível online, incluindo dicionário de dados e diagramas iterativos;
- modelo de banco de dados bastante amplo;
- descreve as estratégias de cargas e as ferramentas para carga dos dados;
- tem disponível os programas, ou seja, com código-fonte aberto;
- trabalha com localizações aproximadas;
- os carregadores são projetos para diminuir a redundância, sobretudo de informações provenientes de diversas fontes;
- contém uma análise dos trabalhos correlatos na publicação.

Os **aspectos negativos** verificados na análise foram:

- não foi identificado o analisador para arquivos no formato GenBank (GBFF);
- o modelo das localizações é incompleto, de modo que não é possível definir os operadores "*join*" previstos na especificação do GenBank;
- não interpreta o formato texto do GenBank; eles adotam o formato ASN.1 e o conversor para XML fornecido pelo NCBI[14].

Limitações do analisador do BioWarehouse para arquivos no formato do GenBank, conforme entendem os autores, são:

> - *As of version 3.0, this load has been tested on the BCT and CON divisions. However, running the loader on the CON division takes approximate 80 hours.*
> - *As of version 3.0, this loader has not been tested on the following GenBank subdivisions (Bug 287):*
> - *High-throughput genomic sequences: HTGS division*
> - *Genome survey sequences: GSS division*
> - *Patented sequences: PAT division*
> - *In addition, the following division MAY contain bacterial sequences as well:*
> - *High throughput cDNA sequencing: HTC division*
> - *As of version 3.0, point sequence features are ignored (Bug 525). We ignore any <Seq-feat> that contains a <Seq-loc_pnt> as its location. BIOWAREHOUSE (2010, GenBank Loader Developer Manual)*

2.16.4 GeneRecords

De acordo com D'Addabbo et al. (2004), o GeneRecords é a solução para projetos que precisam converter os arquivos textos do GenBank em um banco de dados relacional, possibilitando a recuperação, indexação e análise dos dados em um computador pessoal. Para os autores, as limitações do programa são:

> *Maximum size of GenBank entry to be imported: up to 1,102,200 bp.*
> *Maximum size of GenBank 'Features' section for each entry, allowing a correct Features splitting:*
> *64,000 characters following text processing.*
> *Entries with a larger 'Features' section will be processed, but the splitting of the features in the subdatabases could be incomplete.*
> *A single GeneRecords database may store up to 2 Gbyte (a physical limit of the core database).*
> *The CON division of GenBank contains data for joining other sequences, and it can not be imported. (UNIVERSITÀ DI BOLOGNA, 2010).*

[14] Observações obtidas no endereço:
http://biowarehouse.ai.sri.com/repos/genbank-loader/docs/Genbank_Loader.html

A busca por artefatos adicionais sobre o trabalho não apresentou resultados. Não se conseguiu identificar a descrição do banco de dados nem as ferramentas empregadas nem a documentação técnica. As informações estão restritas às disponíveis no artigo e ao tutorial constante no sítio da internet.

2.16.5 Design and Implementation of a Simple Relational Database for GenBank-Derived Data

O artigo de Crow et al. (2007) apresenta a primeira proposta identificada para o desenvolvimento de um banco de dados relacional para as informações do GenBank. Conforme os autores, o RelGB foi projetado para acesso local dos dados do GenBank, em que a ideia é, a partir da base de RelGB, ferramentas de consultas possam ser construídas para abordar questões de interesse e que os extratos do banco de dados possam servir para gerar o banco de dados local do Blast.

O artigo está bastante consistente, pois descreve a motivação, os princípios considerados na modelagem, o diagrama do modelo proposto e a programação em Perl das funções para acesso ao banco de dados em um SGBD Oracle. O modelo não contempla todas as seções da especificação atual do GenBank; porém, para a época, o modelo deveria atender às especificações da seção *features* e das sequências correspondentes.

Para os autores, as ferramentas de análise (parsers) disponíveis pelo NCBI para interpretar os formatos ASN.1/XML são difíceis de usar, uma vez que a documentação é extensa, entretanto, reconhecidamente desatualizada. E ainda, a adoção no trabalho do módulo para Perl chamado GenBank:ParseFeatures antes desenvolvida para uso no gbtoxml.conversor(). Finalizam com a descrição da origem dos dados, de modo que "*In the end, it was decided that using this latter format would provide a reasonable basis for the input data for this system*." (CROW et al., 2007).

2.16.6 Dissertação de Nathan Mann

Nathan Mann (2004) descreve a necessidade e a dificuldade do grupo de pesquisa em Bioinformática da Universidade de Louisville, como:

> *Researching EST through a computer program for the Bio-informatics research group (BRG) at the University of Louisville has been difficult, because of the amount of time involved when querying and the inability for specific queries. The objective of this thesis project is to design an object-relational database called Gene Database to store GenBank data. The versatility of Gene Database will allow the database to be populated not only with human EST information but also with the genomic and EST information of other species. A computer interface for Gene Database will allow more specific research goals to be met. The ultimate goal of Gene Database is aiding in the research of bioinformatics.(MANN, 2004, p.1)*

De acordo com o autor, a área do projeto de tese está preocupada apenas com os dados humanos EST (*Expressed Sequence Tags*, em inglês), cujo foco do trabalho é o banco de dados, mais precisamente:

> *The database is the main focus for this project. It will allow for faster access for the full GenBank record, for specifically queried fields. The major focus is on the features table, which allows features information to be searchable. As seen from the discussion about the features table, it is the most complex field in an EST (MANN, 2004, p.12).*

Em relação ao modelo, o autor defendeu o do objeto-relacional desenvolvido pela Oracle. Dessa forma, o modelo fere as regras básicas da normalização que, de acordo com a visão do autor, são justificáveis:

> *An object-relational model for designing databases goes against the normalization process that is used in any good database schema. A summary of the entities that were decided on is shown in Table I. Most of the entities follow a similar style to an entry. The modifications were made to make the database more efficient than the flat file. (MANN, 2004, p.16)*

> *The object-relational database approach still requires the separation of the descriptor and description, and keeps track of how to organize the features table to be consistent with any entry. Taking the descriptor and description pair, it was made into a data-type. That data-type was used to create an object-table that is nested in a column. That would allow the features to be included for each row, which speeds up queries and creation of the entry once a match is found. Only one row is needed to hold all the possible descriptor and description pairs, allowing the key name and location to be stored in the same row as well. (MANN, 2004, p.19)*

O autor explica o tratamento dado ao campo “localização”, como “*the location field is left intact since there are numerous ways to report a location and it is not a priority to have a better way of searching the locations*.” (MANN, 2004, p.19)

O autor descreve ainda as limitações do SQL Loader, o que exigiu o desenvolvimento de uma ferramenta própria para importação e carga dos dados;

apresenta o diagrama entidade-relacionamento e os resultados obtidos de modo que, para o presente trabalho, destacam-se:

> *The compression that the files have from GenBank is roughly 91% smaller. For a file that is 19 Megabytes from GenBank, it would require approximately 212 Megabytes uncompressed. The GenBank information that is human EST requires 1800 Megabytes. Uncompressed human EST files would require approximately 20199 Megabytes, or 21 Gigabytes of hard drive space.*
> *[…]*
> *The number of records that the import tool can handle is 6.2 records per second. If there were a million records to be inserted, it would take approximately 9 days to do all the insertions.*
> *[…]*
> *However, GenBank only releases updated information every quarter, so a week it takes to populate the database with the new information is acceptable (MANN, 2004, p.26-27).*

No presente, as atualizações do GenBank são bimestrais, o volume de dados é maior e o tempo de carga longos inviabilizam a adesão em larga escala de aplicações que necessitem de banco de dados.

3 MATERIAIS E MÉTODOS

3.1 CONSIDERAÇÕES INICIAIS

Neste capítulo, apresenta-se a metodologia aplicada no planejamento e execução desta dissertação. Também são demonstrados os métodos e técnicas de análise, a modelagem e a especificação de banco de dados, bem como as técnicas aplicadas no desenvolvimento de cada componente da solução proposta. Ainda, faz-se uma síntese das discussões de fundo que influenciaram os resultados obtidos.

3.2 VISÃO GERAL

O presente trabalho descreve a sistematização do banco de dados biológicos do Grupo de Pesquisa em Bioinformática da UFPR, a partir dos arquivos textos disponibilizados pelo NCBI GenBank, para futuras aplicações de mineração de dados. A FIGURA 14 apresenta os componentes de apoio do banco de dados e suas dependências, conforme as setas e linhas pontilhadas.

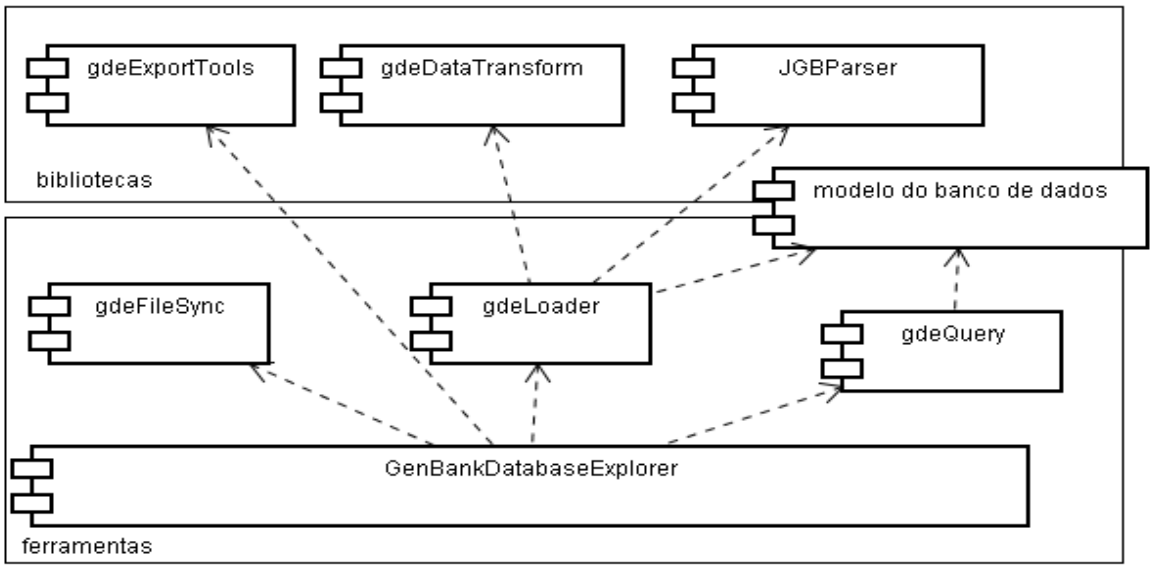

FIGURA 14 - COMPONENTES E RELAÇÃO DE DEPENDÊNCIA DAS APLICAÇÕES DE APOIO DO BANCO DE DADOS

FONTE: o autor (2010).

As funções dos componentes são:

a) modelo do banco de dados – esquema criado para manter todas as informações do GenBank;
b) gdeExportTools – conjunto de bibliotecas que permite a leitura e escrita de arquivos nos formatos FASTA e GenBank;
c) gdeDataTransform – conjunto de funções para padronizar os dados. (Ex: nomenclatura dos autores);
d) JGBParser – analisador de alto desempenho de arquivo textos no formato GenBank;
e) gdeFileSync – ferramenta para sincronizar o sistema de arquivos locais com o servidor de arquivos do NCBI. Essa ferramenta permite identificar e baixar os arquivos novos ou que sofreram atualizações;
f) gdeLoader – ferramenta de carga dos dados extraídos com o analisador para um SGBD suportado;
g) gdeQuery – ferramenta básica de submissão de consultas SQL nos diversos SGBD;
h) GenBankDatabaseExplorer – um ambiente *desktop* para organizar e integrar as ferramentas.

Todos os componentes foram desenvolvidos em Java, o que permite o seu uso nos mais diferentes sistemas operacionais e equipamentos disponíveis. O modelo de banco de dados foi projetado segundo o padrão ANSI-SQL e possui *scripts* de criação disponíveis para os sistemas de gerenciamento de banco de dados: MySQL, PostgreSQL, Oracle, JavaDB (Derby) e H2 Database.

Em paralelo ao desenvolvimento do trabalho, algumas discussões de fundo ocorreram. Entre elas, destacam-se:

- bancos de dados, sistemas de bancos de dados e banco de dados biológicos;
- redundância versus desempenho;
- qualidade das informações;
- execuções das aplicações de bioinformática, e
- desempenho das aplicações.

Essas questões influenciaram o desenvolvimento do presente trabalho, de forma que as soluções não puderam ser somente eficazes; precisaram obter a

melhor eficiência para atingir as metas propostas. Dessa forma, as próximas seções evidenciam uma síntese dessas questões e como influenciaram o desenvolvimento das aplicações e do modelo proposto.

3.3 BANCO DE DADOS, SISTEMAS DE BANCOS DE DADOS E BANCO DE DADOS BIOLÓGICOS

Ao se realizar o levantamento de trabalhos na área de banco de dados e que faziam referência aos dados do GenBank, detectou-se a necessidade de rever os conceitos e classificações empregados. Tal fato se deve à verificação de que os principais bancos de dados biológicos são denominados *banco de dados* – armazenam um conjunto de dados significativo –, porém, não expõem seus modelos nem suas tecnologias e tampouco sua infraestrutura, o que impede comparar termos técnicos dos diferentes trabalhos. Em função das informações oferecidas pelos pesquisadores, propõe-se uma classificação dos trabalhos em “Banco de Dados”, “Sistemas de Banco de Dados” e “Repositório de Dados Biológicos”.

Para fins desse trabalho, o conceito de “banco de dados” é definido e explicado a partir dos contextos de sistema de informação e de tecnologia da informação.

No contexto de sistemas de informação, deve-se observar que a maioria dos bancos de dados, em geral, nasce como parte de um sistema de informação ou aplicação de banco de dados. Na evolução dos sistemas ou do processo de sistematização das organizações, os depósitos de dados de uma aplicação passam a servir outras aplicações. Assim, quando uma base de dados é dada por diversos sistemas, ela deixa de ser o banco de dados de um sistema e passa a ser o banco de dados de um departamento, organização ou contexto maior. Neste trabalho, esses bancos de dados que alimentam múltiplos sistemas são denominados então “banco de dados”, enquanto nas situações de banco de dados de aplicações específicas são chamados simplesmente de base de dados dos sistemas ou dados do sistema.

No contexto de tecnologia de informação, os bancos de dados são um conjunto de programas, métodos e tecnologias que permitem organizar e administrar a informação, assegurando acesso, recuperação eficiente, integridade e qualidade dos dados. Para fins deste trabalho, portanto, **banco de dados é o conjunto de informações relacionado a um tema, organizado e estruturado conforme um**

modelo, geralmente relacional, e, de preferência, **gerenciado por um SGBD**, que agrega às propriedades transacionais e aos elementos tecnológicos necessários à manutenção da informação.

Ao estender o conceito aos bancos de dados biológicos, tem-se que são amplamente utilizados por diversos sistemas, então denominados **banco de dados biológicos.** Os demais casos são chamados de **sistemas ou aplicações de banco de dados biológicos**.

Por fim, não se deve confundir a definição de banco de dados aqui proposta com o conceito de banco de dados primários e secundários encontrados na literatura especializada (PROSDOCIMI, 2002). Os bancos de dados primários, quando acessados diretamente por diversos outros sistemas ou bases de dados, podem ser classificados como banco de dados biológicos. Na ausência de indício de emprego de tecnologias de banco de dados, contudo, recomenda-se o termo “repositório de dados biológicos”. Nesta categoria, enquadra-se o GenBank.

Por sua vez, o EMBL Nucleotide Sequence Database (WANG et al., 2000) pode classificar como “Banco de Dados Biológicos” por apresentar todos os elementos de um banco de dados – modelo ou esquema de dados, tecnologia usada, Oracle, mecanismos de acesso etc.

Entre os bancos de dados derivados ou secundários, com a mesma classificação, o Gene Ontology (GO) apresenta o modelo, a tecnologia MySQL e os dados são redistribuídos como um backup do MySQL.

Outros bancos que podem ser classificados conforme essa metodologia são: Swiss-Prot, PIR, COG, KEGG.

Assim, **este trabalho propõe um banco de dados biológicos**, ou seja, uma base de informações atualizadas e associadas a uma tecnologia de banco de dados para subsidiar outros projetos de sistemas de informação.

3.4 REDUNDÂNCIA VERSUS DESEMPENHO

Em informática, redundância é o termo adotado para descrever duplicação de dados não justificada ou relações desenvolvidas nos modelos computacionais. Por sua vez, desempenho está relacionado à velocidade com que um equipamento ou programa realiza determinadas tarefas. Eventualmente, os programadores

promovem ou permitem, de modo deliberado, a redundância nos modelos em prol do ganho de desempenho.

Todos os trabalhos propostos e observados na área, com o objetivo de organizar e estruturar as informações do GenBank em um modelo relacional, tiveram de tomar a decisão entre ter algum nível de redundância nos dados ou agravar em muito o custo do desempenho. Neste trabalho, não foi diferente. Algumas opções de modelagens foram estudadas; os dados foram carregados mais de uma vez em modelos diferentes e o tempo de processamento das consultas foi mensurado para a tomada da decisão. Ao observar o modelo atual, as estruturas para representar as localizações e as anotações são as que mais sofrem possibilidades de normalização e revisão no modelo.

No caso das localizações, os endereços são representados por um, dois ou mais pontos, de maneira que a opção de modelagem e armazenamento foi para cada caso:

- para um ponto: uso de apenas um registro na tabela gblocation e repetição do valor do primeiro ponto no segundo ponto, preservando o conceito de maior ocorrência (região delimitada por dois pontos). Essa situação foi observada em 678 casos dos 7.106.933 verificados na versão 178 dos dados do GenBank;
- para uma região delimitada por dois pontos: opção empregada de um registro na tabela gblocation. Situação observada em 6.966.283 dos 7.106.933 casos existentes, ou seja, 98% dos casos;
- para regiões informadas por conjuntos de três ou mais pontos: uso de um registro na tabela gblocation e de outros para pares de pontos na tabela gbsublocation, de modo que os pontos contidos no primeiro registro denotam os extremos da região, facilitando a obtenção da sequência de nucleotídeos completa. Foram observadas 140.551 ocorrências, ou aproximadamente 2%.

Pelas razões expostas, obteve-se o modelo apresentado na FIGURA 15, sobre a modelagem das informações de localização.

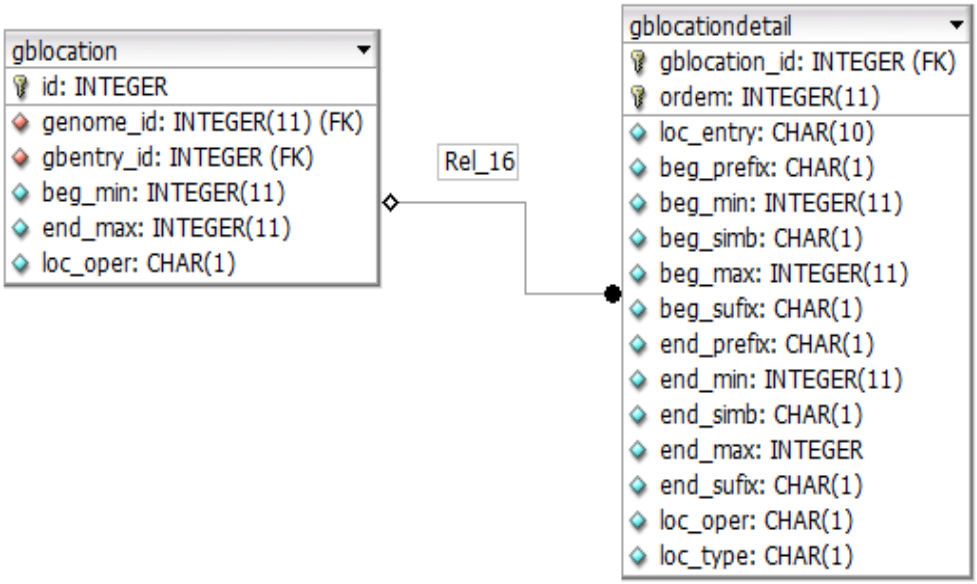

FIGURA 15 - DIAGRAMA DAS ENTIDADES DO BANCO DE DADOS QUE ARMAZENAM AS LOCALIZAÇÕES DAS ANOTAÇÕES NOS RESPECTIVOS GENOMAS

FONTE: o autor (2010).

A tabela gblocation apresenta a situação mais frequente observada – 98% dos casos –, enquanto na tabela gblocationdetail tem-se as situações dos operadores “join” e “order“ previstos na especificação, identificados pela variação do campo ordem, com valores de 1 a N, onde N representa o número de termos identificados na localização da anotação. Nos casos raros em que a anotação apresenta algum prefixo, símbolo de ligação diferente dos dois pontos ou sufixos, então é criado um registro na tabela gblocationdetail com o valor 0 na ordem.

A modelagem das anotações é outro segmento deste trabalho que produz grandes potenciais de redundância. A questão básica era reduzir o máximo a redundância sem causar consequências no processo de carga e recuperação das informações. A primeira modelagem desenvolvida e testada é apresentada na FIGURA 16:

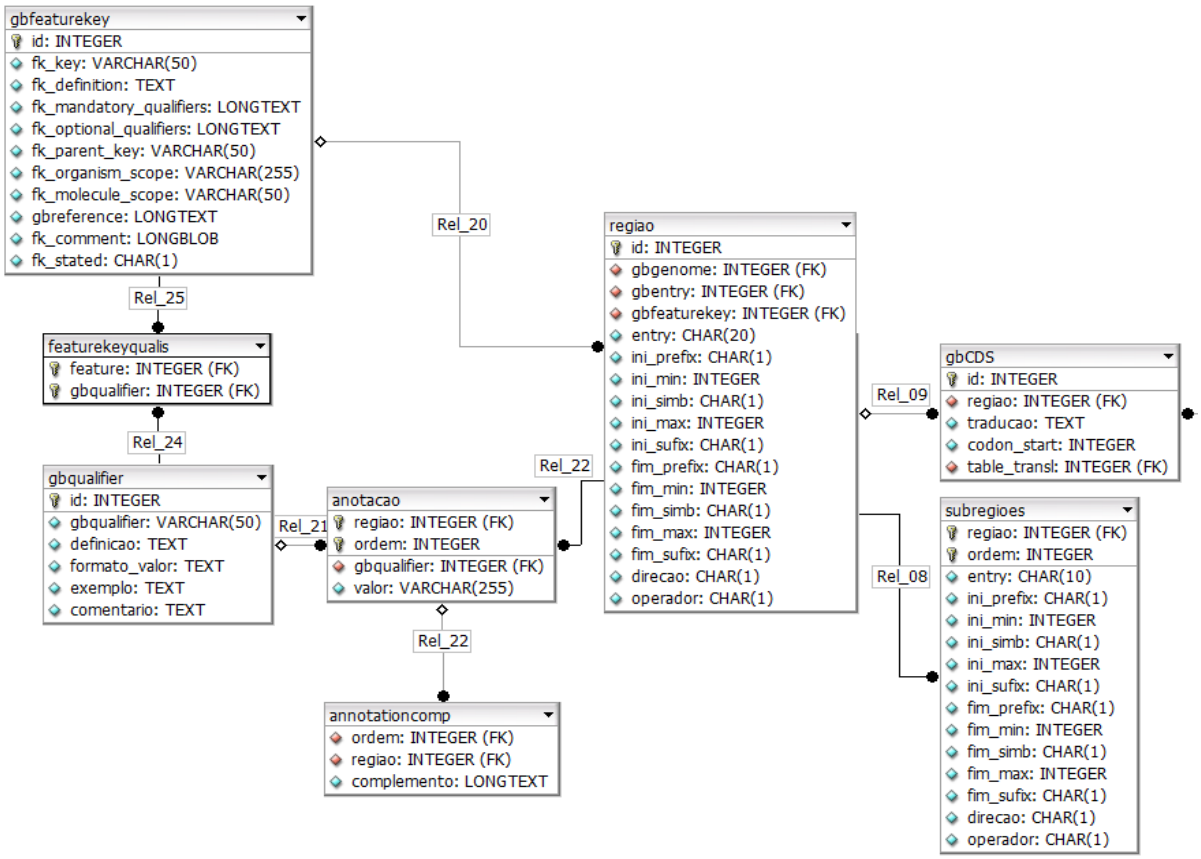

FIGURA 16 - MODELAGEM INICIAL DAS CARACTERÍSTICAS E ANOTAÇÕES

FONTE: o autor (2010).

O grande problema observado nessa proposta de modelagem foi a redundância observada no campo valor da tabela anotação, que teve mais de quarenta milhões de registros para aproximadamente 17 milhões de valores únicos. Outro ponto de redundância estava na tabela “regiao", onde uma mesma localização para duas etiquetas (tags) de características geravam dois registros praticamente idênticos (situação muito frequente – exemplo: as etiquetas GENE e CDS). As remodelagens destes pontos levaram à versão apresentada na FIGURA 17.

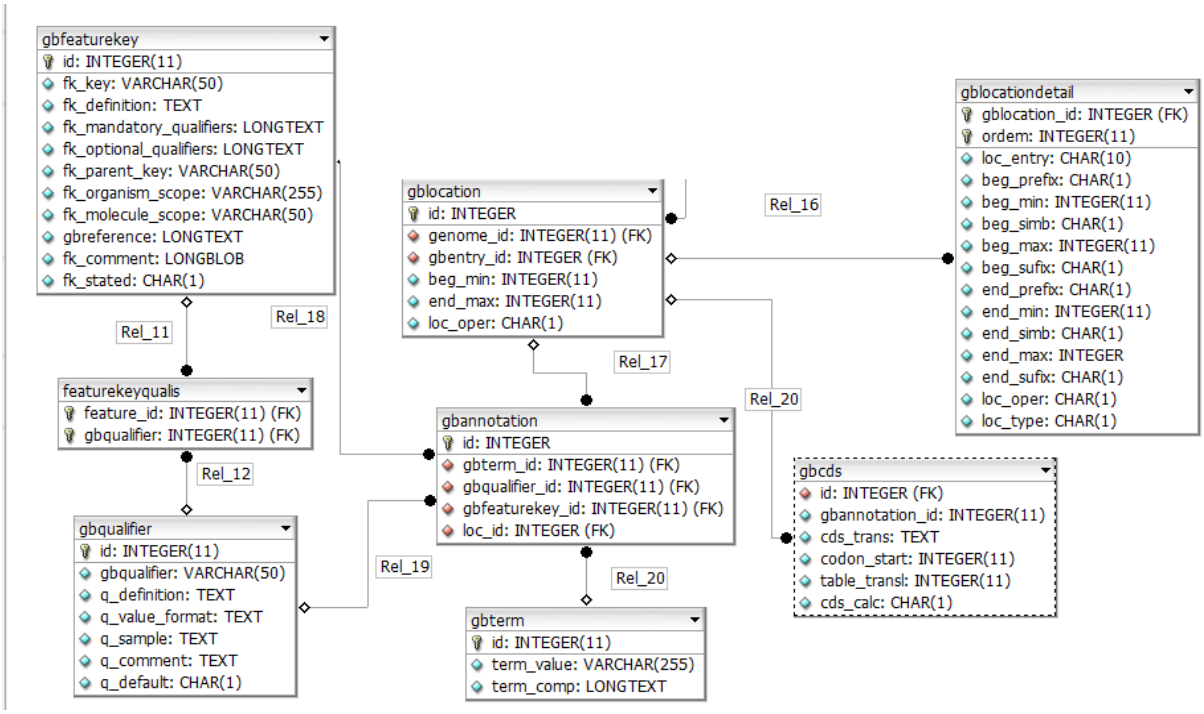

FIGURA 17 - MODELAGEM DOS ELEMENTOS DE ANOTAÇÃO

FONTE: o autor (2010).

3.5 QUALIDADE DAS INFORMAÇÕES

A origem dos dados do GenBank está no trabalho de pesquisadores e centros de pesquisas no mundo inteiro, os quais desenvolvem o sequenciamento, a montagem e a anotação dos organismos. É esperado que nesse processo ocorram divergências nos padrões de anotação, na nomenclatura e nos conceitos empregados.

Para Schnoes et al. (2009), erros de anotação estão sendo propagados em função dos mecanismos de anotação automática. É estimado que exista pelo menos um erro de anotação em cada organismo completamente sequenciado. Estudos indicam uma confiança maior nos bancos de dados curados, tais como o Swiss-Prot.

Este trabalho se propõe a oferecer as informações existentes e distribuídas pelo NCBI GenBank em um modelo relacional; não realiza nenhum processo de filtro nos dados, porém, a depender do trabalho a ser desenvolvido, esse cuidado deverá ser considerado. Os dados disponíveis nessa base de dados foram literalmente transcritos das informações contidas nos arquivos textos do GenBank, isto é, não são de responsabilidade deste autor eventuais erros de anotação.

3.6 EXECUÇÕES DAS APLICAÇÕES DE BIOINFORMÁTICA

A comunidade científica se acostumou à necessidade de desenvolver receitas (*pipeline*) para executar algumas atividades. Tal fato se reflete nas aplicações verificadas nos diversos processos. Como cada aplicação é executada geralmente como parte de um fluxo ou por um arquivo sequencial de chamadas de programas (arquivos *batch* ou *script* de execução), não se detectam as limitações no gerenciamento de memórias, no tempo computacional total despendido ou nas opções de modelagem utilizadas por essas aplicações.

Na contramão, sequenciadores rápidos estão produzindo quantidades maiores de sequências menores que as tecnologias anteriores. Eles levam as aplicações atuais ao limite de suas capacidades, o que exige muito mais recursos de *hardware* – memória e processamento – do que os disponíveis, normalmente, na UFPR.

Esse efeito foi observado nas bibliotecas disponíveis para os procedimentos de bioinformática, opções de modelagem, como as adotadas pelo BioJava[15], de representar cada nucleotídeo de uma sequência como uma referência a um objeto que consome oito vezes mais memória do que realmente é necessário para armazenar a mesma sequência. Também pode ocorrer de a opção de alocar a memória dinamicamente para cada elemento, como foi feito pelo projeto GbParsy (LEE et al, 2008), causar o efeito de fragmentação de memória, o que impede uma aplicação em uma única execução de ler e interpretar todos os arquivos no formato GenBank dos genomas de bactéria disponíveis atualmente.

Dessa forma, evidencia-se a necessidade de melhores padrões de projetos e de boas práticas de programação no desenvolvimento de bibliotecas e aplicações para a bioinformática.

3.7 PREPARAÇÃO DO BANCO DE DADOS PARA APLICAÇÃO DE MÉTODOS ESTATÍSTICOS

Métodos estatísticos são frequentemente aplicados em processos de análise dos dados e mineração de dados (GOLDSCHMIDT e PASSOS, 2005). Considerados os conceitos e princípios na preparação dos dados propostos por

[15] Encontrada em: http://www.biojava.org/wiki/BioJava:Cookbook:Sequence

Munro (2004, p.8 e p.10), bem como os aspectos da análise semântica, entende-se de extrema relevância a obtenção da atomicidade na organização dos dados e alguma redundância nas relações entre as entidades propostas.

3.8 SINCRONIZADOR DE ARQUIVOS/DIRETÓRIOS

A função do programa de sincronização é manter a cópia dos arquivos do GenBank que estão no sistema de arquivos locais atualizados do usuário. O programa acessa por meio do sérvio de FTP o servidor de arquivos do NCBI, consulta os arquivos, faz as comparações com os arquivos locais e realiza o *download* dos arquivos novos ou modificados.

Ao preservar a filosofia do desenvolvimento de componentes que podem ser usados isoladamente ou em conjunto, o sincronizador foi desenvolvido em um projeto à parte. O aplicativo permite seu uso em modo terminal (texto) ou em modo gráfico.

Como a API padrão do Java não tem disponível um cliente FTP, optou-se pelo uso da biblioteca da Fundação Apache (Projeto Jakarta Commons Net) em função do estudo apresentado na seção 2.10.

A biblioteca “commons-net-2.0” da Fundação Apache apresentou a falha de execução da função “getModificationTime” implementada, o que exigiu a obtenção dos programas fontes da biblioteca, a correção e o emprego da versão corrigida. A referida falha está registrada sob o código 309[16] e tem previsão para correção junto com a liberação da versão 2.1. Obtiveram-se os programas fontes da biblioteca, de modo que foram aplicadas as correções indicadas no controle de falhas do projeto. Por essa razão, junto aos programas fontes do projeto estão também os programas fontes dessa biblioteca.

A FIGURA 18 apresenta a saída padrão da aplicação gdeFileSync quando executada no modo console. Nesta figura podem-se observar as opções disponíveis na versão atual.

[16] Disponível no jira: https://issues.apache.org/jira/browse/NET-309

```
C:\mestrado\tese\programas\gdeFileSync\dist>java -jar gdeFileSync.jar
GenBank Database Explorer - File Synchronization Tool
version 0.1

use: javar -jar gdeFileSync options
 -win enable graphics app
 -dst local directory destination [ default ./ ]
 -ftp FTP server [default: ftp.ncbi.nih.gov ]
 -rpt remote path [default: /genomes/Bacteria/ ]
 -v verbose mode
 -n|u|nu -n new files only or -u update only or -nu both
 -n new files is default option
```

FIGURA 18 - EXEMPLO DE EXECUÇÃO DA FERRAMENTA DE SINCRONIZAÇÃO DO SERVIDOR DO NCBI COM OS ARQUIVOS NO SISTEMA LOCAL

FONTE: o autor (2010).

Em função do formato de arquivos disponíveis no NCBI e dos recursos existentes na linguagem, recomenda-se que, na primeira vez, obtenha-se o arquivo dos genomas compactado (all.gbk.tar.gz[17]); que seja descompactado no diretório local e depois utilizada a ferramenta gdeFileSync para atualizar os dados. Isso porque o formato compacto exigirá menor tempo de *download* e, mesmo somados os tempos de *download* e descompactação, esse será menor que o tempo de execução da aplicação. Já no caso da atualização, essa relação de custo e beneficio se inverte, de modo que o uso da ferramenta se torna mais eficiente.

3.9 ANALISADOR (PARSER)

A função do analisador é interpretar uma sequência de texto (ou *bytes* em arquivos binários) identificando os elementos e transformá-la em uma estrutura composta pelas informações originais segmentadas na forma de campos, onde os dados estão normalmente na forma atômica.

A FIGURA 19 apresenta o diagrama de classes da estrutura desenvolvida para armazenar as informações de um arquivo GenBank. Os nomes e atributos das classes foram compilados da descrição da especificação apresentada no item 2.8.2.

[17] Disponível em: ftp://ftp.ncbi.nlm.nih.gov/genomes/Bacteria/

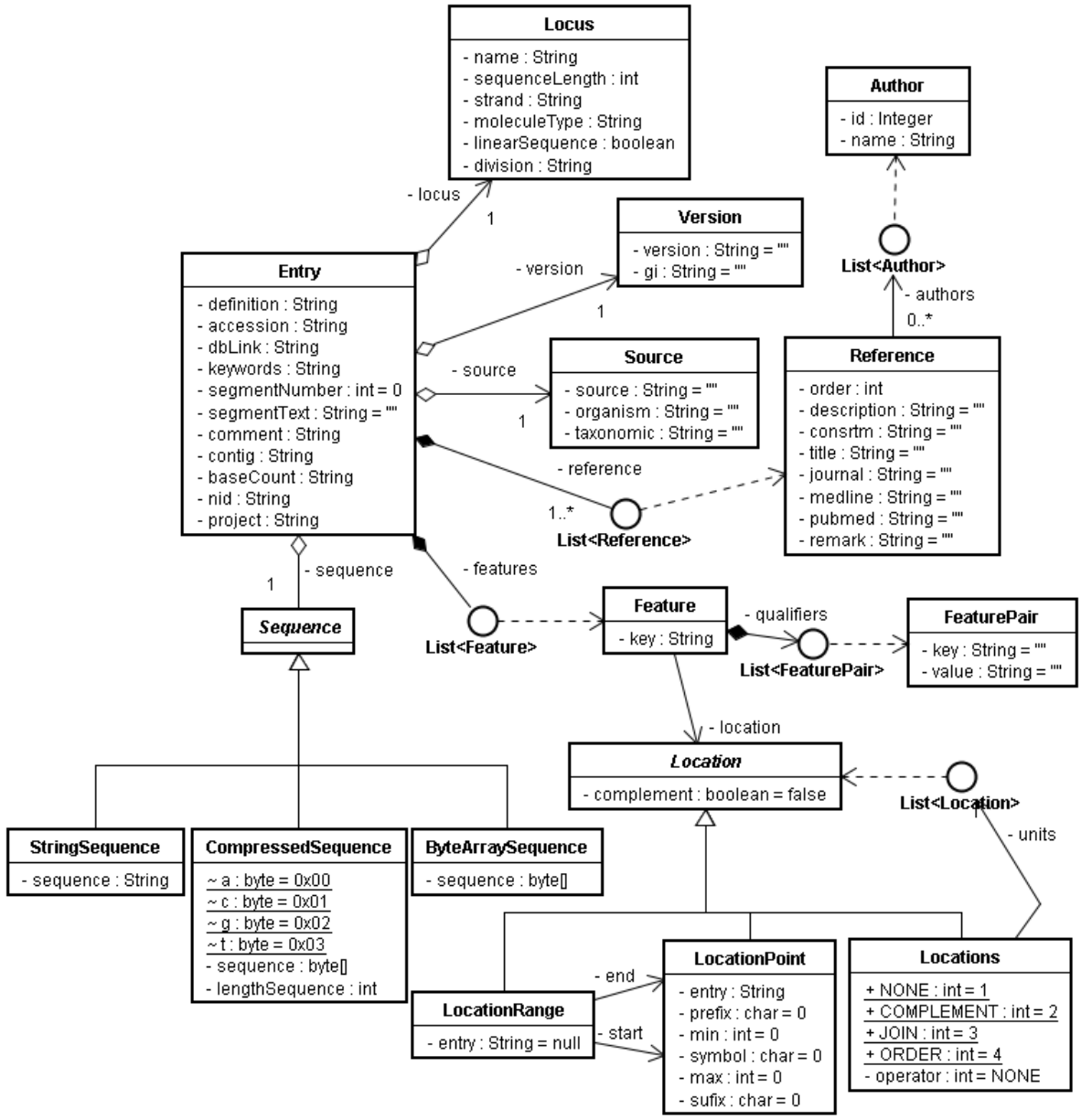

FIGURA 19 - MODELO DE CLASSE DO GENBANK (OMITIDOS OS MÉTODOS POR QUESTÕES DE LEGIBILIDADE)

FONTE: o autor (2010).

As classes Locus, Version e Source foram separadas da classe Entry em objetos distintos por poderem ter uma mesma instância em registros do GenBank diferentes. Os atributos *reference* e *features* de Entry são agregações, pois para um objeto de Entry existir, devem ter as listas, mesmo que vazias de Reference e Features. A opção das associações previstas pelos atributos *reference* e *features* estarem representadas com ligações às interfaces e essas dependerem das classes que implementam os respectivos objetos ocorreu em função dos recursos da

linguagem que permitem o emprego de tipos genéricos associados às classes que implementam a interface *List*, dispensando a construção de classes coleções.

Sequence e *Location* são classes abstratas que possuem especializações distintas para cada uma das situações previstas pela aplicação. A coleção *qualifiers,* presente na classe *Feature* armazena os descritores (*qualifiers*) e os respectivos valores anotados. O atributo *Location* presente em *Feature* identifica uma região da sequência de nucleotídeos (armazenados em Sequence), os quais são objetos das anotações.

A complexidade no desenvolvimento do analisador de arquivos no formato GenBank advém das especificações às vezes incompletas e às vezes interpretadas pelos diferentes pesquisadores ou programas produtores desse formato de arquivo. Provém, ainda da validação não muito rígida do NCBI nos processos de submissão direta das informações. Contribuem para essas impressões as afirmações do Biowarehouse (2010):

> *The GenBank loader loads the NCBI GenBank database of nucleotide sequences into the BioWarehouse (also reffered to as the "Warehouse"). The input files to the GenBank loader are a set of large XML files. The XML schema is quite complex, containing over 8,000 element definitions. The large size and complexity of the XML make for a complex loader.* ***The information in this document is meant to given an overview of how the loader proceeds, but does not document the specifc details in most cases****.[negrito nosso] BIOWAREHOUSE (2010, GenBank Loader Developer Manual)*

Em especial, esses aspectos influenciam o modelo de classe apresentada na FIGURA 19 e também o Diagrama Entidade Relacionamento (FIGURA 20), em que as relações entre as tabelas gbfeatureskey e gbqualifiers não são rigidamente exigidas nas relações similares que ocorrem nas anotações.

3.10 DESENVOLVIMENTO DE MODELO PARA UM OU MUITOS SGBDS

Apesar dos SGBD atenderem de forma razoável a especificação do padrão ANSI SQL, as diferenças existem e trazem como consequência dois aspectos: primeiro: para cada tipo de problema ou situação, existe um SGBD com recursos mais apropriados; segundo: para obter melhor desempenho e maior aproveitamento dos recursos disponíveis nos SGBD, os programas e modelos precisam ser

especializados, ou seja, desenvolvido para as particularidades de cada SGBD. Conforme Prosdocimi, os SGBD mais empregados na área da bioinformática são:

> Existem vários sistemas de gerenciamento de banco de dados, sendo que cada sistema possui seus prós e contras. O MySQL é um sistema muito utilizado pela comunidade acadêmica e em projetos genoma por ser gratuito, possuir código aberto e acesso veloz aos dados, mas apresenta certas limitações em suas ferramentas. O PostgreSQL também é um SGBD gratuito, com ferramentas muito poderosas, entretanto não é muito utilizado pela dificuldade no seu gerenciamento. Os SGBDs Oracle e SQLServer são robustos e sofisticados, mas devido ao alto custo de suas licenças possuem seu uso limitado às grandes empresas. (PROSDOCIMI, 2002, p.14).

Dessa forma, nesta pesquisa, optou-se por desenvolver versões do programa de criação do banco de dados para os SGBD: MySQL, PostgreSQL, Oracle e JavaDB. Os dois primeiros por serem vastamente empregados na área e terem licenças gratuitas, o Oracle por ser o líder do mercado no segmento comercial e o JavaDB, por ser uma opção de banco de dados embarcada (pode ser anexado e usado pela aplicação sem precisar de um SGBD instalado).

Algumas consequências, principalmente no processo de carga, são discutidas na seção 3.12.

3.11 MODELAGEM DO BANCO DE DADOS PARA AS INFORMAÇÕES DO GENBANK

Seguindo as fases descritas por Silberschatz et al. (2006), a fase inicial do projeto prevê entrevistas com os usuários para definir e caracterizar o problema inicialmente. Neste trabalho, o problema estava devidamente caracterizado desde o princípio, ou seja, o objetivo estava bem definido: transpor os dados disponíveis em mais de 1000 arquivos do GenBank para uma estrutura de banco de dados relacional.

A segunda fase prevista por Silberschatz et al. (2006) se refere à escolha da base para a modelagem conceitual; neste caso, escolheu-se o modelo entidade-relacionamento que de acordo com o autor é geralmente usada, e a seguir foi desenvolvido o diagrama entidade-relacionamento.

As execuções das fases seguintes ocorreram ao longo do trabalho, que seguiu a seguinte metodologia:

a) análise da especificação do NCBI GenBank e desenvolvimento do primeiro protótipo do modelo;
b) desenvolvimento do programa para ler o arquivo no formato texto e carregar os dados na estrutura projetada;
c) teste de carga com os arquivos do NCBI;
d) identificação de informações não previstas no modelo;
e) revisão do modelo e do programa de carga;
f) repetição das etapas três, quatro e cinco até que todos os dados fossem carregados.

Nas etapas um e cinco foram aplicadas a técnica das formas normais no processo de modelagem. Foram verificadas as entidades e sua correspondência com os atributos, a atomicidade dos atributos e as associações observadas na especificação e nos arquivos.

As etapas quatro e cinco foram realizadas diversas vezes em função da existência de itens previstos e removidos da especificação; também porque os pesquisadores aplicaram “termos” para as *features* e *qualifiers* que não estão previstos nos apêndices da especificação, bem como em consequência de falhas de anotações e da flexibilidade do NCBI no processo de validação dos arquivos submetidos pelos pesquisadores em relação à especificação.

A etapa três foi realizada diversas vezes, até que todos os arquivos disponíveis no NCBI, até julho de 2010, fossem corretamente convertidos e carregados no banco de dados.

No final, obteve-se o modelo que é ao mesmo tempo aderente à especificação e capaz de receber os dados de todos os arquivos oferecidos pelo GenBank.

No desenvolvimento do modelo lógico e físico, vários foram os aspectos observados e considerados. Entre eles, destacam-se:

a) preocupação em evitar o emprego das palavras-reservadas dos principais SGBD, possibilitando o uso do modelo nos mais variados sistemas disponíveis. A relação dos termos encontrados estão disponíveis no site do projeto;
b) uso do menor conjunto de tipos de dados possível, novamente procurando ampliar a adoção do modelo;
c) uso de prefixos nos atributos para evitar coincidência de termos em SGBD não avaliados;

d) desenvolvimento de uma versão mínima, conforme a especificação do padrão ANSI SQL, para contemplar os SGBD não desenvolvidos em programas específicos.

A FIGURA 20 demonstra o Diagrama de Entidade Relacionamento (DER) final obtido nesse processo.

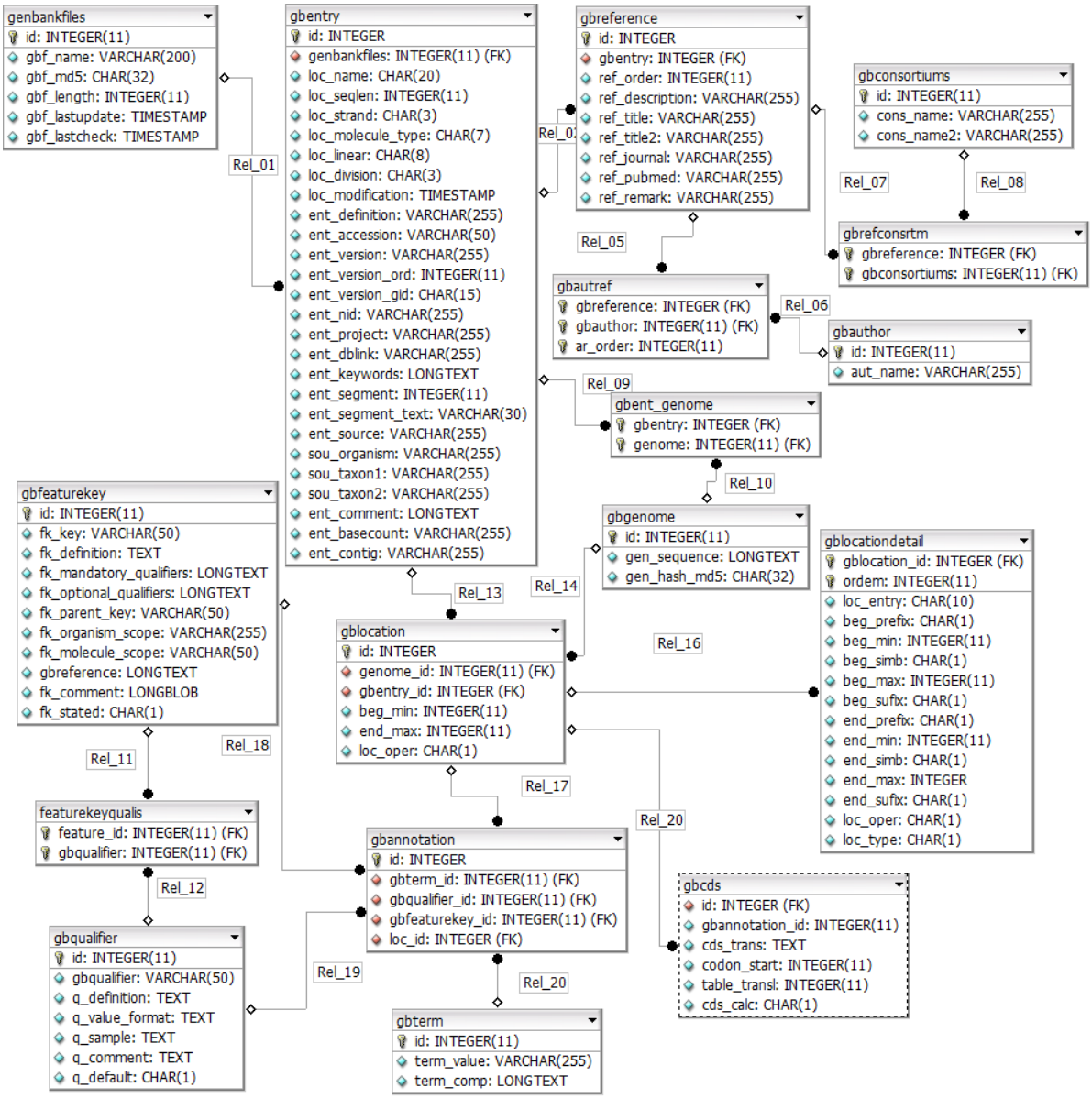

FIGURA 20 - DIAGRAMA ENTIDADE RELACIONAMENTO

FONTE: o autor (2010).

A implementação do modelo físico foi realizada para os sistemas gerenciadores de banco de dados: MySQL, PostgreSQL, Oracle, H2 Database. Também foi feita uma versão baseada no padrão ANSI SQL.

Nas próximas seções, são discutidas as opções de modelagem para cada parte do arquivo GenBank.

3.11.1 A tabela genbankfiles

A tabela genbankfiles foi desenvolvida para representar os arquivos no formato GenBank e permitir dois tipos de comparações para detectar atualizações. A primeira comparação é realizada em função do tamanho. O programa de atualização pode ser configurado para baixar o arquivo somente quando ocorre a variação do tamanho, reduzindo o tempo e demanda de rede. O segundo teste é realizado pela comparação da chave hash (md5) gerada com base em todo arquivo. Infelizmente, o NCBI não tem disponível as assinaturas (hash) dos arquivos do seu FTP. Se isso ocorresse, essa seria a primeira e melhor opção de comparação.

3.11.2 A tabela gbentry

Originalmente, previmos as tabelas gbsegment e gbentry, em que a primeira entidade era responsável em representar a seção SEGMENT. Depois, optou-se pela fusão com a tabela gbentry, em função de não ter sido detectada nenhuma ocorrência dessa seção nos arquivos de genomas completamente sequenciados.

A tabela gbentry representa os elementos das seções que são descritas na documentação que ocorrem no máximo uma vez.

3.11.3 As tabelas gbreference, gbauthor e gbconsortiums

As tabelas gbreference, gbauthor, gbconsortiums, gbautref e gbrefconsrtm são responsáveis em representar e preservar as informações da seção REFERENCE. O nome do autor sofreu uma padronização, para reduzir os nomes duplicados.

Basicamente, cada campo previsto na especificação foi mapeado para um campo na tabela gbreference, são eles: ref_title, ref_title2, ref_journal, ref_pubmed, ref_remark. O campo título teve de ser dividido em dois em função do MySQL não permitir textos com mais de 255 caracteres em campos do tipo char/varchar e a alternativa do uso de campos blob causaria o efeito colateral de performance indesejado. Nesse caso, com a margem de segurança, os títulos maiores que 250

são divididos em dois campos. A título de exemplo, o artigo do MATSUBARA et al. (1989), identificado pelo código 2777793 do PUBMED, possui o título “Molecular cloning and nucleotide sequence of cDNAs encoding the precursors of rat long chain acyl-coenzyme A, short chain acyl-coenzyme A, and isovaleryl-coenzyme A dehydrogenases. Sequence homology of four enzymes of the acyl-CoA dehydrogenase family.”, contendo 252 letras.

A relação NxN entre autores e referência teve de ter acrescido o campo ar_order para preservar a ordem em que aparecem os autores e, também, foi acrescida a chave primária, em função de ter referência com as iniciais de autores repetidas, tais como ocorre com o texto “Souza, E. M.” no arquivo NC_014323.gbk (PEDROSA, 2010).

3.11.4 A tabela gbgenome

Em gbgenome são armazenadas as sequências de ácidos nucléicos dos genomas; ou melhor, são armazenadas as informações contidas na seção ORIGIN dos arquivos GenBank.

Adicionalmente, foi criada a coluna hash_md5 que mantém o hash da sequência para facilitar a identificação de eventuais alterações no genoma, em casos de atualizações.

3.11.5 A tabela gbfeaturekey

A tabela gbfeaturekey foi criada para armazenar as características (*features*) previstas no item “*7.2 Appendix II: Feature keys reference*” do “*The DDBJ/EMBL/GenBank Feature Table: Definition*.”.

> *The following has been organized according to the following format:*
> *Feature Key - the feature key name*
> *Definition - the definition of the key*
> *Mandatory qualifiers - qualifiers required with the key; if there are no mandatory qualifiers, this field is omitted.*
> *Optional qualifiers - optional qualifiers associated with the key*
> *Organism scope - valid organisms for the key; if the scope is any organism, this field is omitted.*
> *Molecule scope - valid molecule types; if the scope is any molecule type, this field is omitted.*
> *References - citations of published reports, usually supporting the feature consensus sequence*

Comment comments and clarifications
Abbreviations:

Accnum -	*an entry primary accession number*
<amino_acid> -	*abbreviation for amino acid*
<base_range> -	*location descriptor for a simple range of bases*
<bool> -	*Boolean truth value. Valid values are yes and no*
feature_label -	*the feature label (follows naming conventions for all feature table components)*
<integer> -	*unsigned integer value*
<location> -	*general feature location descriptor*
<modified_base> -	*abbreviation for modified nucleotide base*
[number] -	*integer representing number of citation in entry's reference list*
<repeat_type> -	*value indicating the organization of a repeated sequence.*
"text" -	*any text or character string. Since the string is delimited by double quotes, double quotes may only appear as part of the string if they appear in pairs. For example, the sentence:*

The feature label "ops-tata" is used with the "promotor" feature key would be formatted thus: "The feature label""ops-tata" " is used with the " "promoter" " feature key" NCBI, The DDBJ/EMBL/GenBank Feature Table: Definition

Dessa descrição e da observância do Apêndice II saíram dois produtos: o primeiro, a tabela; o segundo, um programa de análise (parser) especialmente desenvolvido para converter o texto para o banco de dados.

No início, a intenção era assegurar que todas as anotações (seção *features*) fossem ancoradas nos termos (*feature key*) previstos na especificação; porém, em razão de que foram encontrados muitos casos de inconsistência dos arquivos em relação ao padrão, optou-se por adicionar o campo “fk_stated” para identificar os itens previstos na especificação e a cada identificação de “novas chaves” foi optado em incluir automaticamente na tabela.

Essa tabela é carregada com os valores padrões toda vez que o banco de dados é criado.

3.11.6 A tabela gbqualifier

Análogo ao processo que ocorreu no estudo das características, os descritores (*qualifiers)* foram obtidos com base na especificação constante do item “*7.3 Appendix III: Summary of qualifiers for feature keys*” do “*The DDBJ/EMBL/GenBank Feature Table: Definition*.”.

Qualifier - *name of qualifier; qualifier requires a value if followed by an equal sign*
Definition - *definition of the qualifier*
Value format - *format of value, if required*
Example - *example of qualifier with value*
Comment - *comments, questions and clarifications*
NCBI, The DDBJ/EMBL/GenBank Feature Table: Definition

Dessa descrição e da análise do Apêndice III saíram dois produtos: o primeiro foi a tabela gbqualifier, o segundo foi um programa de análise (parser) especialmente desenvolvido para converter o texto para o banco de dados.

Novamente foi optado pela criação do campo “qua_stated” e a inclusão automática de todos os termos encontrados nos arquivos GenBank e que não estavam constantes da especificação.

Essa tabela também é carregada com os valores padrões toda vez que o banco de dados é criado.

Complementar, a tabela featurekeyqualis mantém as chaves do relacionamento NxN existente entre as tabelas gbfeaturekey e gbqualifier.

3.11.7 As tabelas gblocation, gblocationdetail, gbannotation e gbCDS

O conjunto de tabelas descritas nessa seção foi desenvolvido para armazenar e representar as anotações (seção *features*) dos arquivos GenBank.

A FIGURA 21 apresenta um exemplo de anotação, em que os termos “gene” e “CDS” são as chaves das características (*features*), os números logo a frente da chave identifica a localização do gene e da região codificante, respectivamente. O conjunto de chave, seguida pelo símbolo de igual (=) e o valor compõe uma anotação. Neste caso, essas chaves são os descritores (*qualifiers*) apresentados na seção anterior.

```
                     /mol_type="genomic DNA"
     gene            212..8668
                     /gene="NF1"
     CDS             212..8668
                     /gene="NF1"
                     /note="putative"
                     /codon_start=1
                     /product="GAP-related protein"
                     /protein_id="AAA59924.1"
                     /translation="MAAHRPVEWVQAVVSRFDEQLPIKTGQQNTHTKVSTE.......
---------+---------+---------+---------+---------+---------+---------+---------
1        10        20        30        40        50        60        70        79
```

FIGURA 21 - EXEMPLO DE ANOTAÇÃO EM ARQUIVOS DO GENBANK

FONTE: o autor (2010).

Como toda anotação se refere a uma região específica da sequência de ácidos nucleicos, optou-se por modelar esse conjunto baseado no modelo estrela da modelagem dimensional, em que a localização é a tabela fato e a essa tabela ficam associadas às tabelas: anotação (gbannotation), áreas codificantes (gbCDS), subregiões (gblocationdetail), genomas (gbgenome) e as entradas (gbentry).

As localizações apresentaram uma dificuldade à parte, em função do conjunto de possibilidades. De acordo com a especificação do NCBI, elas podem ser:

> ***3.5 Location***
> ***3.5.1 Purpose***
>
> *The location indicates the region of the presented sequence which corresponds to a feature.*
>
> ***3.5.2 Format and conventions***
> *The location contains at least one sequence location descriptor and may contain one or more operators with one or more sequence location descriptors. Base numbers refer to the numbering in the entry. This numbering designates the first base (5' end) of the presented sequence as base 1.*
>
> *Base locations beyond the range of the presented sequence may not be used in location descriptors, the only exception being location in a remote entry (see 3.5.2.1, e).*
>
> *Location operators and descriptors are discussed in more detail below.*
>
> *3.5.2.1 Location descriptors*
> *The location descriptor can be one of the following:*
> *(a) a single base number*
> *(b) a site between two indicated adjoining bases*
> *(c) a single base chosen from within a specified range of bases (not allowed for new entries)*
> *(d) the base numbers delimiting a sequence span*
> *(e) a remote entry identifier followed by a local location descriptor (i.e., a-d)*

> *A site between two adjoining nucleotides, such as endonucleolytic cleavage site, is indicated by listing the two points separated by a carat (^). The permitted formats for this descriptor are n^n+1 (for example 55^56), or, for circular molecules, n^1, where "n" is the full length of the molecule, ie 1000^1 for circular molecule with length 1000.*
>
> *A single base chosen from a range of bases is indicated by the first base number and the last base number of the range separated by a single period (e.g., '12.21' indicates a single base taken from between the indicated points). From October 2006 the usage of this descriptor is restricted: it is illegal to use "a single base from a range" (c) either on its own or in combination with the "sequence span" (d) descriptor for newly created entries. The existing entries where such descriptors exist are going to be retrofitted.*
>
> *Sequence spans are indicated by the starting base number and the ending base number separated by two periods (e.g., '34..456'). The '<' and '>' symbols may be used with the starting and ending base numbers to indicate that an end point is beyond the specified base number. The starting and ending base positions can be represented as distinct base numbers ('34..456') or a site between two indicated adjoining bases.*
>
> *A location in a remote entry (not the entry to which the feature table belongs) can be specified by giving the accession-number and sequence version of the remote entry, followed by a colon ":", followed by a location descriptor which applies to that entry's sequence (i.e. J12345.1:1..15, see also examples below)*
>
> *3.5.2.2 Operators*
>
> *The location operator is a prefix that specifies what must be done to the indicated sequence to find or construct the location corresponding to the feature. A list of operators is given below with their definitions and most common format.*
>
> *complement(location)*
> *Find the complement of the presented sequence in the span specified by "location" (i.e., read the complement of the presented strand in its 5'-to-3' direction)*
>
> *join(location,location, ... location)*
> *The indicated elements should be joined (placed end-to-end) to form one contiguous sequence*
>
> *order(location,location, ... location)*
> *The elements can be found in the specified order (5' to 3' direction), but nothing is implied about the reasonableness about joining them*
>
> *Note : location operator "complement" can be used in combination with either " join" or "order" within the same location; combinations of "join" and "order" within the same location (nested operators) are illegal.*
> NCBI, The DDBJ/EMBL/GenBank Feature Table: Definition

A interpretação, modelagem e implementação dos programas para analisar esse aspecto da notação levou um tempo considerável e uma complexidade maior que a esperada. Em essência, quando o modelo estava relativamente interessante

(no sentido de atender aos objetivos propostos), tornava-se inviável para as estruturas de banco de dados e vice-versa.

Em busca da maior fidelidade possível à metodologia aplicada, inicialmente o programa foi escrito para realizar o teste. Para isso, foram escolhidos os exemplos apresentados pelo próprio NCBI, conforme transcrito a seguir:

> *Location - Description*
>
> *467 - Points to a single base in the presented sequence*
>
> *340..565 - Points to a continuous range of bases bounded by and including the starting and ending bases*
>
> *<345..500 - Indicates that the exact lower boundary point of a feature is unknown. The location begins at some base previous to the first base specified (which need not be contained in the presented sequence) and continues to and includes the ending base*
>
> *<1..888 - The feature starts before the first sequenced base and continues to and includes base 888*
>
> *1..>888 - The feature starts at the first sequenced base and continues beyond base 888*
>
> *102.110 - Indicates that the exact location is unknown but that it is one of the bases between bases 102 and 110, inclusive*
>
> *123^124 - Points to a site between bases 123 and 124*
>
> *join(12..78,134..202) Regions 12 to 78 and 134 to 202 should be joined to form one contiguous sequence.*
>
> *complement(34..126) Start at the base complementary to 126 and finish at the base complementary to base 34 (the feature is on the strand complementary to the presented strand)*
>
> *complement(join(2691..4571,4918..5163)) - Joins regions 2691 to 4571 and 4918 to 5163, then complements the joined segments (the feature is on the strand complementary to the presented strand)*
>
> *join(complement(4918..5163),complement(2691..4571)) - Complements regions 4918 to 5163 and 2691 to 4571, then joins the complemented segments (the feature is on the strand complementary to the presented strand)*
>
> *J00194.1:100..202 - Points to bases 100 to 202, inclusive, in the entry (in this database) with primary accession number 'J00194'*
>
> *join(1..100,J00194.1:100..202) - Joins region 1..100 of the existing entry with the region 100..202 of remote entry J00194*
> NCBI, The DDBJ/EMBL/GenBank Feature Table: Definition

Posteriormente, escreveu-se o modelo de classes para representar as informações acima, conforme demonstrado na FIGURA 22. A seguir, foi escrito o programa de análise (parser) e foram verificados os testes. Após isso, foi desenvolvida uma aplicação para comparar os resultados desse analisador com a versão oferecida na biblioteca BioJava e, por fim, executado em todas as anotações disponíveis pelo NCBI na pasta Bactéria do servidor de arquivos.

Atualmente, adotam-se as duas expressões regulares "(\d+)\.\.(\d+)" e "(complement)\((\d+)\.\.(\d+)\)" para as duas situações de maior ocorrência. As situações seguem os exemplos "123..1011" e "complement(133..300)", respectivamente. As demais situações foram resolvidas por um algoritmo que analisa símbolo a símbolo.

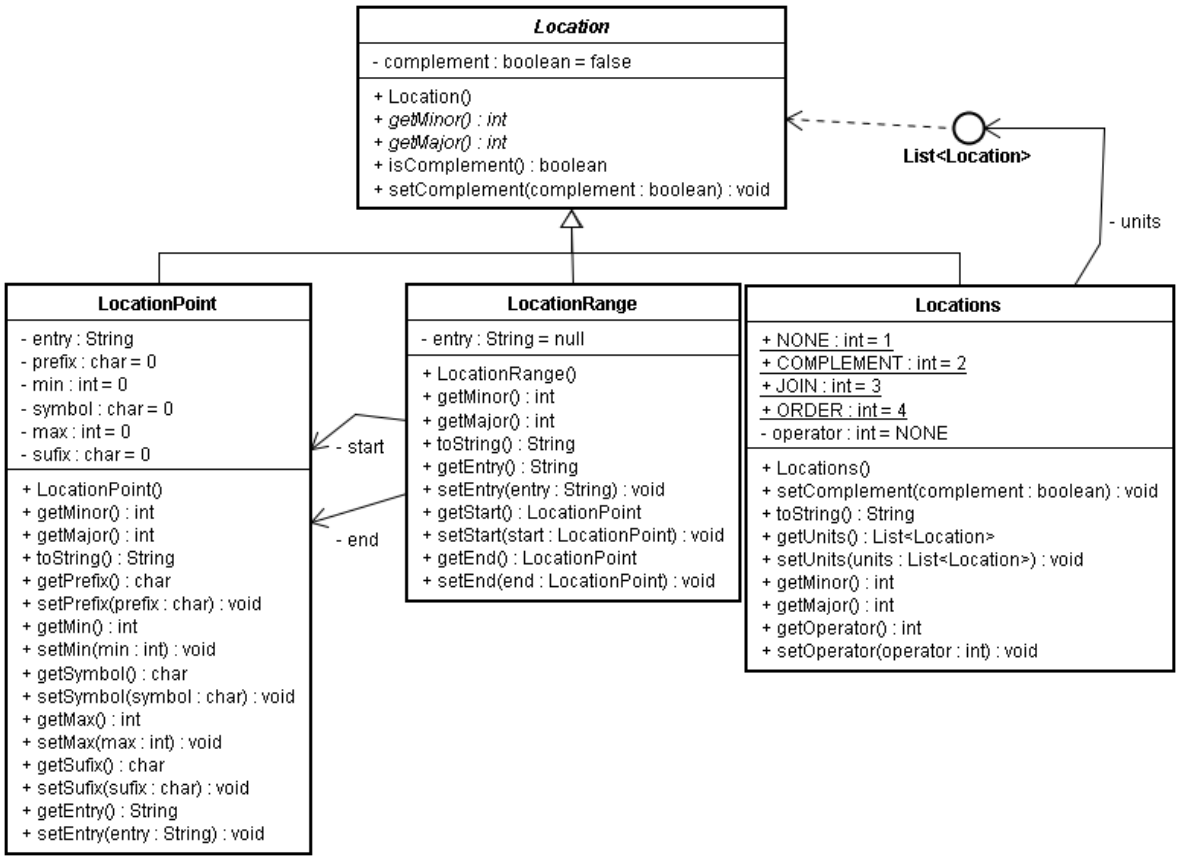

FIGURA 22 - DIAGRAMA DE CLASSE DAS LOCALIZAÇÕES

FONTE: o autor (2010).

3.11.8 Considerações finais sobre a modelagem

São esperados modificações, refinamentos e agregações de novas entidades no modelo proposto; portanto, não se pode considerar o presente modelo como definitivo. Em análises semânticas, o esquema de banco de dados relacional pode ser modificado (KRUSINSKA,1991) para a aplicação de muitos métodos estatísticos, os dados terão de ser modificados, normalizados ou terem revista sua representação. Tais questões, porém, apesar de relevantes e correlatas ao presente trabalho, fogem ao escopo previsto.

3.12 APLICAÇÃO DE CARGA DOS DADOS

Para realizar a carga dos dados, foram desenvolvidos diversos protótipos e aplicadas varias técnicas. Cada estratégia apresentou vantagens e desvantagens em relação às outras, de modo que o resultado final apresenta limitações que não o consolidam como melhor solução esperada.

3.12.1 Protótipo 1 – Camada de persistência automática

O primeiro protótipo escrito em Java permitia a seleção do diretório em que os arquivos GenBank estavam e realizava a leitura, a análise (parse), a padronização dos dados e a persistência dos dados diretamente no banco de dados, utilizando os recursos da Java Persistence API (JPA). Foram testados com as bibliotecas da Oracle (Toplink) e do Hibernate.

Esse protótipo levou 26 minutos para carregar os dados do GenBank da *Escherichia coli* e em três semanas de execução contínua carregou 226 arquivos dos 1.822 então disponíveis.

3.12.2 Protótipo 2 – Camada de persistência manual

O segundo protótipo foi desenvolvido com a reescrita da camada de persistência, substituindo os recursos do JPA por classes que empregam o Java Database Connectivity (JDBC) em conformidade com o padrão de projetos *Data Access Object* (DAO).

Essa versão levou 23 minutos para carregar o genoma da *Escherichia coli*, ou seja, um pouco melhor, porém longe do necessário.

3.12.3 Protótipo 3 – Técnicas de otimização

Nas classes DAO da terceira versão, foram reescritas a lógica de criação e a execução dos procedimentos de inserção, cujas alterações foram:

a) substituição do procedimento de criação do objeto da classe PreparedStatement a cada inserção para uma única vez, no construtor da classe DAO. E reaproveitamento desses objetos nos métodos que invocavam o procedimento de inserção;
b) substituição da execução imediata para execução em *batch* em todas as tabelas que não possuíam outras tabelas dependentes, em função da necessidade de gerar o código autoincremento para preservar os relacionamentos;
c) revisão e substituição de todas as concatenações por referências ao método format da classe String nas classes DAO, e
d) atualização da versão do driver JDBC do MySQL para a versão 5 e para um tipo 4 (Native-protocol).

Essa versão leva em média um minuto e trinta e seis segundos para carregar todas as informações da *Escherichia coli* str. K-12 substr. MG1655 (arquivo NC_000913.gbk).

Um aspecto importante e consequente das técnicas acima se relaciona ao fato de que a tabela gbCDS, que seria melhor manter a relação com a tabela gbannotation, teve o relacionamento alterado para a tabela gblocation. Essa alteração permitiu a aplicação do método de inserção em *batch* na tabela gbannotation, onde há mais de 40 milhões de registros e o melhor desempenho obtido nos testes.

3.12.4 Protótipo 4 – Especialização para o SGBD MySQL

Após a carga do banco completo no MySQL, observou-se que a recuperação da cópia de segurança (*backup*) era mais rápida que os processos de inserção. Um exame desse arquivo sugeriu que avaliasse os comandos *lock table* e *insert*. Nos testes, normalmente, não ocorria concorrência, de modo que o uso do primeiro

comando não influenciava o resultado. O segundo comando é largamente empregado nos dois casos, ou seja, nas soluções antes adotadas e no arquivo da cópia de segurança; porém, com uma diferença de sintaxe que pode ser observada na FIGURA 23:

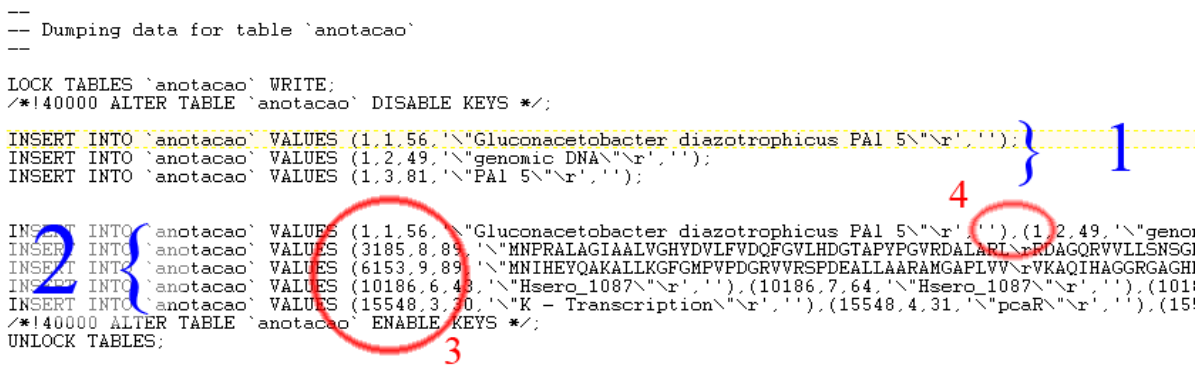

FIGURA 23 - EXTRATO DO ARQUIVO DE BACKUP PRODUZIDO PELO MYSQL

FONTE: o autor (2010).

Os comandos em destaque na região um da FIGURA 23 demonstram uma sequência de comandos para inserções comumente utilizadas pelas aplicações de carga. A região dois apresenta os mesmos comandos; porém, escritos pelo programa que faz a cópia de segurança do MySQL. Nessa segunda área, pode-se observar nos destaques três e quatro que o programa de cópia de segurança (*backup*) escreve poucas vezes o comando de *insert* e envia vários registros em apenas uma instrução de inserção, separados por vírgula.

Em consequência dessa observação, desenvolveu-se o teste de ler o arquivo NC_014323, que contém o genoma anotado da *Herbaspirillum seropedicae* SmR1 (PEDROSA, 2010), e inserir as linhas desse arquivo em uma tabela no MySQL. A tabela tem apenas os dois campos: id e linha. Esse teste foi realizado por meio de três procedimentos, a saber:

a) processo de inserção por meio da sintaxe básica da linguagem Java (uso do método executeUpdate da classe PreparedStatement);
b) processo de inserção em *batch* da linguagem Java (uso do método addBatch e executeBatch da classe PreparedStatement), e
c) processo de inserção, enviando dois registros a cada submissão.

Os resultados obtidos foram, respectivamente, 43769 ms, 36663 ms, 24444 ms, o levou a repetir-se o teste do terceiro processo, enviando ao servidor 2, 4, 8, 16 e 32 instruções simultâneas. A TABELA 5 apresenta os tempos obtidos.

TABELA 5 - NÚMERO DE INSTRUÇÕES VERSOS TEMPOS OBTIDOS COM A CLASSE PREPAREDSTATEMENT

Número de registros por instrução	Tempo em milissegundos (ms)
1	43769
2	24444
4	12660
8	10030
16	44222

FONTE: o autor (2010).

Como se pode observar, pelo experimento e uso da classe PreparedStatement, o número ideal de registros enviados simultaneamente estaria próximo a oito por instrução, contraditório ao recurso adotado pelo programa que faz a cópia de segurança e restauração do MySQL. Foi feita nova tentativa com base na classe Statement, com parâmetros de configuração do MySQL “rewriteBatchedStatements = true”, “useServerPrepStmts = true”, “cachePrepStmts = true”, cujo resultados obtidos são apresentados na FIGURA 24.

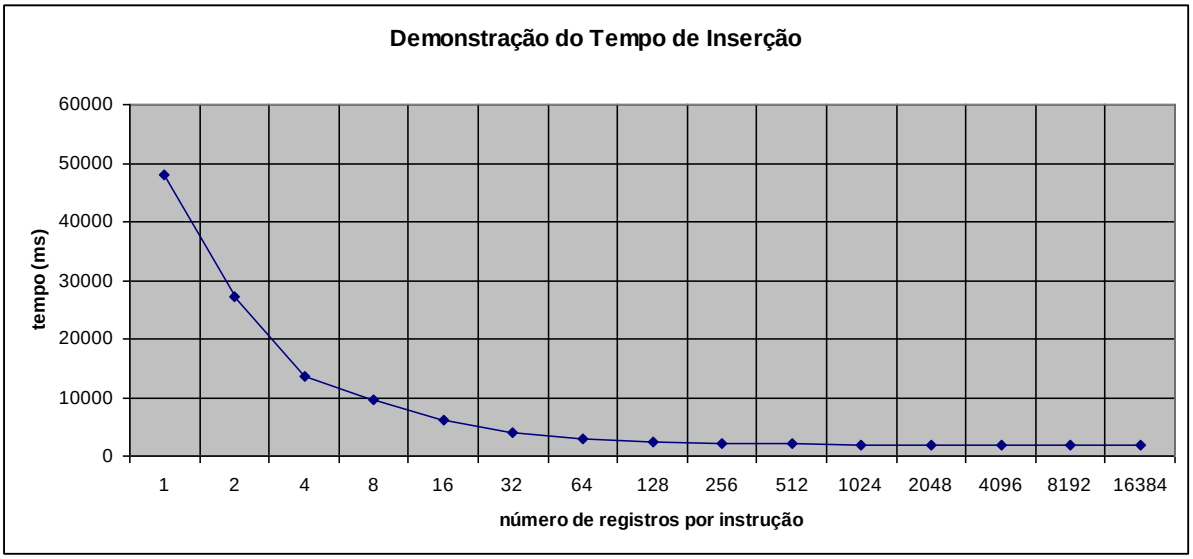

FIGURA 24 - TEMPO DE PROCESSAMENTO OBTIDO NO TESTE DE ENVIO DE VÁRIOS REGISTROS POR INTRUÇÃO DE INSERÇÃO

FONTE: o autor (2010).

A FIGURA 24 demonstra que, acima de 4096 registros por instrução, o tempo de processamento final é praticamente o mesmo, o que explica o recurso usado pelo programa de cópia de segurança do MySQL para o volume de informações usado nos testes.

O manual do MySQL não explica os parâmetros citados e foram relatados várias falhas no dispositivo (*driver*) de conexão do Java para o emprego desses recursos. No caso deste estudo, os testes apresentaram sucesso de uso.

4 RESULTADOS

4.1 O BANCO DE DADOS

A diversidade de sistemas de banco de dados encontrada na literatura demonstra que, para cada aspecto estudado na área da biologia molecular, existe uma abordagem específica para armazenar, analisar e distribuir as informações. Se essa filosofia continuar, terá em pouco tempo tantos sistemas de banco de dados quanto genes, proteínas ou organismos estudados.

Tal prática prolifera a duplicação das informações em tempos e proporções inimagináveis. Por conseqüência, a comunidade científica precisa estar mais atenta aos processos de atualização e sincronização das bases de dados. Isso porque as atualizações e novos sequenciamentos têm sido depositados e permutados pelos três grandes bancos de dados mundiais. A replicação desses bancos para os demais existentes, contudo, ocorre de diversas formas e em tempos desiguais. O uso de arquivos de impressões digitais, para cada arquivo disponível nos servidores de arquivo, pode simplificar e facilitar a atualização e respectiva sincronização dos bancos secundários.

Atualmente, os bancos de dados biológicos são classificados em primários e secundários, de modo que primários são os bancos de dados receptores das sequências e anotações. Por sua vez, os bancos de dados derivados são classificados como secundários.

Considera-se relevante a adoção de uma nova classificação, em que as classes ou categorias sejam sistemas ou aplicações de banco de dados, banco de dados e repositórios de dados. Assim, as diferentes visões do GenBank seriam classificadas separadamente, pois os milhares de arquivos existentes e disponíveis no servidor de arquivo são classificados como repositório de dados. Os sistemas Sequim, Entrez, Taxonomy, Blast, entre outros, são classificados na categoria “sistemas de banco de dados”.

Projetos como o *The Gene Ontology*, que distribuem as informações conforme um modelo de banco de dados e associados a uma tecnologia de banco de dados, são classificados como “banco de dados”.

Dessa forma, propõe-se a revisão do conceito de banco de dados, de modo que seja considerado: **o conjunto de informações relacionadas a um tema, organizadas e estruturadas conforme um esquema ou modelo,** para cujo uso se recomenda **um Sistema de Gerenciamento de Banco de Dados (SGBD) que assegure as propriedades ACID** – acrônimo de atomicidade, consistência, isolamento e durabilidade – das transações.

O uso de sistemas de gerenciamento de bancos de dados na área de bioinformática requer um modelo ou esquema de dados, ou seja, programas que façam a conversão das informações em formato texto para o modelo desenvolvido, o que implica uma estratégia de atualização e manutenção dos dados e aplicações para desenvolvimento das consultas e recuperação das informações.

4.1.1 O modelo de dados desenvolvido

O modelo desenvolvido neste trabalho se apresenta como a proposta mais completa para reprodução das informações disponíveis em arquivos no formato GenBank pelo NCBI, pois reduz significativamente a redundância das informações, amplia e preserva a consistência das informações e possibilita o uso dos dados nos Sistemas Gerenciadores de Banco de Dados: MySQL, PostgreSQL, Oracle, H2 Database e Derby ou JavaDB. Desta forma, o banco de dados pode ser utilizado em sistemas WEB, *desktop* ou em aplicações embarcadas.

No contexto deste trabalho, compreende-se por “completo” a transposição de todos os termos e relações observadas no arquivo em formato texto do GenBank para a estrutura de dados do banco de dados desenvolvido, sobretudo com a preservação da integridade dos dados.

A motivação principal para a proposta de um novo modelo consiste na seguinte hipótese: as anotações são etiquetas associadas a um par de bases ou a uma sequência de bases identificada em uma ou mais regiões de um genoma. Esta abordagem coloca as localizações das anotações no centro do modelo, reduzindo a redundância da informação e facilitando a recuperação das sequências de nucleotídeos a partir de buscas por anotações.

Em função da normalização, o modelo desenvolvido é formado por dezessete tabelas; três delas são dicionários previstos na especificação – características, descritores e a tabela de relacionamento –, uma para identificar os arquivos e

preservar as características de tamanho, data e impressão digital; tabela esta empregada para facilitar a atualização da base de dados. Outra tabela preserva as informações correspondentes as entradas do GenBank, com as informações do *locus*, de acesso, versão, organismo etc. Duas tabelas se destinam à sequência de genomas, de maneira que uma é utilizada para relacionar com a tabela de entrada, recurso necessário em função de arquivos que possuem o elemento segmento que implica que mais de uma entrada possa ter a mesma sequência. Cinco tabelas compõem o subsistema de referências, com seus respectivos autores e consórcios. As cinco tabelas restantes descrevem as anotações propriamente ditas, de modo que duas se referem a localizações e divisões internas, normalmente presentes nos eucariotos (*exons* e *introns*); duas se destinam aos termos adotados como valores dos descritores, uma às áreas codificantes e a última se presta à ligação entre os elementos de uma anotação – localização, característica, descritor e valor.

Um modelo, por mais consistente que se apresente, não será amplamente aproveitado se não tiver um conjunto de ferramentas que auxilie seu uso. Portanto, neste trabalho, desenvolveu-se um conjunto de ferramentas que possibilita a manutenção dos dados, buscando um processo automatizado, a partir de alterações registradas nos servidores do GenBank. Tais ferramentas podem ser empregadas isoladamente para outros projetos; porém, seu foco é subsidiar atualização dos dados nos bancos de dados locais.

A primeira parte do processo de manutenção das informações, identificada por esta análise, constituiu desenvolver uma estratégia de sincronização dos arquivos textos em um repositório local dos existentes e disponíveis nos servidores do NCBI.

4.1.2 Estratégia desenvolvida para sincronização do repositório de arquivos locais com os repositórios do NCBI

A ferramenta “gdeFileSync” desenvolvida neste trabalho permite a varredura do servidor de arquivos do NCBI e a atualização dos arquivos locais. Para isso, adotou-se a estratégia de comparar as datas dos arquivos locais com as datas dos arquivos remotos. Quando os arquivos remotos apresentam datas mais recentes, a ferramenta copia o arquivo (*download*) para o sistema local, o que reduz significativamente o tráfego de rede e o tempo desse processo. Nos testes, durante a etapa de criação do repositório local, chegou-se à conclusão de que a obtenção do

arquivo compactado com todos os arquivos é mais eficiente. Nos momentos de sincronização subsequentes, entretanto, a aplicação desenvolvida apresentou redução de tempo e consumo de rede, uma vez que o volume de dados atualizados a cada dois meses pelo NCBI é inferior a 20% do conjunto total existente, ou seja, inferior à versão compactada disponível.

A segunda parte do processo de manutenção das informações prevê uma análise para identificar arquivos cujos conteúdos foram modificados e uma forma para atualizar esses dados no banco de dados. A ferramenta “gdeLoader” foi desenvolvida para resolver essa etapa.

4.1.3 Abordagens feitas no processo de carga do banco de dados

A obtenção do melhor desempenho dos SGBD disponíveis no mercado se faz com aplicações escritas especialmente para esses servidores. Tais aplicações podem desenvolver o modelo de dados, com interesse nos recursos existentes e nas estratégias e recursos especiais para inserção de grandes volumes de dados. Essa situação não se pretende neste trabalho porque o desenvolvimento da proposta específica para um SGBD implicaria impor um limitador no uso; ou seja, limitaria o aproveitamento do banco apenas para usuários de um SGBD específico. A alternativa escolhida foi escrever o modelo por meio de um conjunto bastante limitado de recursos e, na aplicação de carga, especializar a camada de persistência para cada banco de dados suportado.

O uso de bibliotecas de persistência automática foi tentado; contudo, o desempenho dessas bibliotecas se evidenciou inadequado para a missão de carregar mais de dois mil arquivos. Assim, optou-se em escrever a camada de persistência manualmente, programando uma versão das classes, com o uso do conjunto mínimo de recursos dos SGBD, suportados em teoria pela padronização do SQL realizada pelo ANSI. E especializou essa camada para os demais SGBD anteriormente descritos. Nas classes especializadas, foram reescritos principalmente os métodos para inserção dos dados, o que se fez necessário para aplicações futuras à programação dos demais métodos.

Algumas consultas foram substituídas por técnicas de *cachê* nas classes da camada de persistência, reduzindo significativamente o número de requisições e tempo de resposta da aplicação.

Ainda, como técnica de otimização de banco de dados, nas tabelas de final das relações, foi empregada a de inserção em *batch* e nas anotações que não se referem à transcrição, em função da chave primária ser requerida na tabela CDS.

Por último, especificamente para o SGBD MySQL foi escrita uma camada especial, que reproduz o arquivo de *backup*, reduzindo em 78% o tempo de inserção.

4.1.4 O analisador de arquivos textos no formato do GenBank

A inexistência de um analisador fornecido pelo NCBI, responsável pela especificação do formato e disponibilidade dos dados, favoreceu o desenvolvimento de diversas propostas, sendo as mais conhecidas as contidas nas bibliotecas BioPerl, BioPython e BioJava. São muito lentas, porém, e não analisam todos os elementos contidos na especificação do GenBank. A proposta do GBParsy (LEE et al., 2008), desenvolvida em C e Python, apresenta desempenho bem melhor que as concorrentes. Nos testes, entretanto, demonstrou problemas de fragmentação de memória, impossibilitando em uma mesma execução interpretar mais de trinta arquivos.

Em função de tais limitações, desenvolveu-se o analisador JGBParser em Java, que compõe o conjunto de ferramentas. Esse analisador apresenta desempenhos equiparáveis ao do GBParsyPy (LEE et al., 2008), porém, não apresenta fragmentação ou vazamentos de memória, sendo empregado nos testes durante horas sem causar a falha de alocação ou perda de desempenho.

A versão atual do analisador de arquivos (JGBParser), desenvolvido nesse trabalho, leva 29 minutos para carregar e interpretar os 2.091 arquivos do GenBank. O analisador identifica e separa as 6.660 referências bibliográficas, 8.051.629 anotações, sendo 3.786.732 de genes e 3.636.654 de CDS. As características (*features*) apresentam, ainda, 46.551.006 termos e qualificadores. Uma característica relevante para o desempenho do analisador é que os arquivos no formato GenBanK podem ser divididos em três grandes segmentos:

a) segmento de armazenamento das sequências que ocupação em média 42,2% do tamanho do arquivo;
b) segmento das anotações que ocupa 55,8% do arquivo, e
c) demais elementos, com aproximadamente 2%.

O desvio padrão observado foi de 3,8% nos três segmentos.

O analisador interpreta a formatação das informações e assegura a integridade dos dados, as respectivas associações e, sobretudo, a fragmentação da informação em todos os campos previstos nas especificações do NCBI GenBank.

O analisador ainda permite a análise parcial – por seção pré-definida –, incremental ou completa. O analisador permite ao desenvolvedor escolher um dos três mecanismos de codificação das sequências de nucleotídeos, a seguir:

a) em formato texto (String), que consome dois *bytes* por cada nucleotídeo da sequência;
b) vetor de *bytes* que ocupa um byte de memória por nucleotídeo e;
c) compactado, com a estratégia do Banco de dados do Blast, que permite armazenar 4 nucleotídeos em um byte.

Com essas opções, o analisador permite o menor consumo de memória pelas aplicações escritas em Java. Ainda, os testes demonstram que o analisador não apresenta vazamento de memória, teste de fundamental importância para quem deseja aproveitar o analisador de forma contínua ou em um servidor de aplicações (um servidor de serviços como o do NCBI).

O analisador está encerrado em uma biblioteca java (jar), o que permite sua integração em qualquer projeto existente ou em novos projetos. Por ter sido escrito totalmente na linguagem Java, o analisador pode ser utilizado em qualquer sistema operacional que tenha uma máquina virtual Java. As outras vantagens da biblioteca são: melhor desempenho da categoria; menor consumo de memória; o arquivo da biblioteca tem aproximadamente 100kb - diminuindo o impacto de tamanho nas aplicações usuárias e; conjunto reduzido de classes e interfaces, o que reduz a curva de aprendizado.

4.1.5 Distribuição dos dados conforme o modelo desenvolvido

A proposta de redistribuição dos dados do NCBI GenBank no formato de *backup* dos principais SGBD se apresenta como uma solução inédita e que permite o acesso da comunidade científica às informações do GenBank, com os recursos e tecnologias do banco de dados relacional, no menor tempo atualmente possível.

Supõe-se que a redistribuição das informações atualizadas do NCBI, em formato de banco de dados, contribuirá para o emprego da tecnologia de banco de

dados na área, bem como a redução na proliferação de bancos derivados diretamente do NCBI e potencializará a derivação deste trabalho pela comunidade científica.

4.1.6 Comparação com os trabalhos correlatos

Os trabalhos descritos na seção 2.18 foram comparados com este trabalho nos aspectos associados a banco de dados e ferramentas de uso. Os elementos de comparação incluem as seguintes características:

a) contempla todos os dados;
b) estratégia de sincronização de arquivos com os dados do NCBI;
c) parser;
d) modelo relacional;
e) modelo documentado (DER, Dicionários);
f) ferramentas de carga;
g) ferramentas de consulta;
h) suporte a vários SGBDs, e
i) suporte para várias linguagens de programação (LP).

A TABELA 6 apresenta os resultados das comparações, de modo que os números identificados nos títulos das colunas se referem à numeração dos elementos acima descritos.

TABELA 6 - COMPARAÇÃO DESTE TRABALHO COM OS TRABALHOS CORRELATOS

Trabalhos	Todos os dados (a)	Sincro-nização (b)	Parser (c)	Relaci-onal (d)	Docu-menta-do (e)	Carga (f)	Consulta (g)	Vários SGBDs (h)	LP (i)
BioSQL	parcial	não	não	sim	parcial	não	Não	A	M
GUS	parcial	não	não	sim	sim	sim	parcial	B	N
BioWarehouse	parcial	não	sim	sim	sim	sim	Não	C	O*
GeneRecords	parcial	não	sim	sim	?	sim	Sim	não	não
Design and Implementation*	parcial	não	sim	sim	sim	sim	Não	Oracle	Perl
Nathan Mann	parcial	não	sim	parcial	parcial	sim	Sim	Oracle	Java
Este trabalho	sim	sim	sim	sim	sim	sim	Não	D	P

* *Design and implementation of a simple relational database for genbank-derived data* (2.18.5).
FONTE: o autor (2010).

No item suporte a vários SGBD (item 8), as letras representam:

- A - PostgreSQL, MySQL, Oracle, HSQLDB, Apache Derby;
- B - Oracle e PostgreSQL;
- C - Oracle e MySQL, e
- D - MySQL, PostgreSQL, Oracle, Apache Derby e ANSI SQL.

Em relação ao suporte a várias linguagens (item 9), tem-se:

- M - Perl, Python, Java e Ruby;
- N - Java + Perl;
- O - Os carregadores foram escritos em C e Java, porém a linguagem de acesso aos dados é a SQL, e
- P - As ferramentas foram escritas em Java. Porém, o banco pode ser acessado pelas principais linguagens disponíveis no mercado.

As comparações indicam que o presente trabalho tem plenas condições de substituir os outros acima, em situações que os pesquisadores precisem de informações do GenBank nas análises. Do mesmo modo, representa uma opção melhor para os novos projetos.

4.2 EXEMPLO DE APLICAÇÕES

Este trabalho não esgota os estudos na área de descoberta de conhecimento (KDD) em banco de dados, pelo contrário, tem por objetivo apenas uma das atividades da primeira etapa do processo – o pré-processamento das informações (GOLDSCHMIDT e PASSOS, 2005). Porém, algumas pesquisas foram realizadas e, no que se refere na exploração dos dados, são descritos a seguir:

4.2.1 Identificação dos organismos fixadores de nitrogênio

O banco de dados, contendo os arquivos da versão 176 do GenBank, foi utilizado para identificar os 56 organismos que tem anotado pelo menos um entre os 20 genes nif. Do banco de dados foi obtido a relação dos organismos e os respectivos genes, bem como as respectivas sequências de nucleotídeos.

O pôster foi aceito e apresentado no *XXXIX Annual Meeting of Brazilian Biochemistry and Molecular Biology Society*, sob o título *Development of a Unified Biological Database for Nitrogen Fixing Organisms* (GEHLEN et al., 2010). O respectivo resumo foi aceito no *2nd Internacional INCT Symposium on Biological Nitrogen Fixation*, com o título *Development of a comprehensive nif gene database* (GEHLEN et al., 2010).

4.2.2 GeneBingo: identificação de genes por meio de rede neural artificial – metodologia e resultados preliminares

O estudo para identificar e classificar os genes implicou o uso do banco de dados, com os dados da versão 178 do GenBank, nas seguintes etapas:

a) para cada genoma depositado no banco de dados de bactéria e archea foram calculados os percentuais de guanina e citosina (%GC) presentes no DNA de cada organismo;
b) os organismos foram organizados em cinco grupos, por faixas do %GC;
c) foram sorteados dez organismos de cada faixa e extraídos do banco as sequências de nucleotídeos das regiões codificantes, bem como as sequências imediatamente anteriores aos respectivos start-codons.

Além dos cinquenta organismos sorteados para o processo de análise e treinamento, foram extraídas as características de 24 dos 30 organismos citados nos testes do Glimmer (SALZBERG et al., 1998), para fins de comparação e validação da rede neural. Todas as operações foram realizadas com comandos de SQL e com o uso de funções e extensões disponíveis no MySQL.

A rede usada neste trabalho foi o FAN (RAITTZ, 1998), em função de ser uma RNA neuro-fuzzy, por ter aprendizado supervisionado e permitir ver graficamente as características e neurônios de uma rede treinada.

5 DISCUSSÃO

5.1 O BANCO DE DADOS

Os aspectos observados sobre os bancos de dados sugerem o aprofundamento de dois elementos: da classificação proposta e da avaliação dos mais de 1400 trabalhos de banco de dados biológicos relatados nas publicações especializadas.

Ainda, observa-se a pertinência da revisão e a recomendação para os desenvolvedores de sistemas de banco de dados biológicos se preocuparem mais com a integração dos bancos de dados e o armazenamento integral dos dados disponíveis e não apenas com recortes das informações disponíveis.

Se essas medidas forem adotadas, haverá bancos de dados mais completos no futuro, os quais potencialmente poderão melhorar as condições para aplicação de técnicas de descoberta de conhecimento (KDD) em banco de dados.

Por meio deste trabalho, concentrou-se o olhar sobre o aspecto da integridade das informações. Todavia, pode ser aprofundado quanto à integração com outros bancos de dados; por exemplo, as referências de projetos e ligações descritas nas anotações e nas etiquetas das anotações que poderiam estar referentes em uma tabela, o que facilitaria a integração com outros bancos.

5.1.1 Estratégia desenvolvida para sincronização do repositório de arquivos locais com os repositórios do NCBI

A estratégia desenvolvida neste trabalho reflete as condições existentes e permite avanços consideráveis neste campo, uma vez que não foram encontradas ferramentas para esse tipo de operação, com a especialidade de tratar arquivos do GenBank. Também, deve-se considerar que se o NCBI produzisse os arquivos de assinaturas ou impressões digitais, de acordo com as técnicas de *hash* vastamente utilizadas (como MD5 ou SHA), e se tornasse disponíveis essas informações junto com os arquivos no seu servidor de arquivos, então a aplicação de sincronização poderia se basear na análise de conteúdo e não apenas na característica de data

atualmente utilizada, o que reduziria ainda mais o tráfego de rede e o tempo necessário para atualização.

A presente solução reduz para menos de 20% a quantidade de arquivos baixados em função da estratégia adotada e pode ser aplicada em diversas outras situações de sincronização dos repositórios de dados baseados em arquivos e servidores de arquivos (FTP).

5.1.2 Abordagens feitas no processo de carga do banco de dados

Várias técnicas foram exploradas, as quais podem ser aplicadas em diversas situações em que se faz necessário reduzir o tempo de carga de dados em banco de dados, sobretudo, em linguagem Java. O caminho encontrado para o MySQL permite levantar a hipótese de que existem mecanismos similares nos demais SGBD e que precisam ser avaliados e explorados.

Independentemente disso, a estratégia adotada pode impor algumas restrições, entre elas, impede a concorrência do uso do banco durante a carga, para contornar esse problema, sugere-se uma aplicação que simule o servidor MySQL como possibilidade de solução. Essa sugestão foi avaliada; porém, não foi programada, impedindo a explanação sobre as vantagens e desvantagens.

5.1.3 Analisador de arquivos para o formato GenBank

O produto final apresentou nos testes velocidades 15, 9 e 2 vezes mais rápidas que a disponível pelo BioPython, BioPerl e GBParsy, respectivamente. Estima-se que o analisador extraia todas as informações contidas em um arquivo GenBank com velocidades de aproximadamente 16.6Mb/s e de 30Mb/s para extrair apenas as informações da seção *features*.

Com esse método, as aplicações de bioinformática poderão processar mais informações em menor tempo.

Para fim de comparação, em um mesmo ambiente de avaliação (*software* e *software*), realizou-se o teste de carga do arquivo NC_003070, referente ao genoma anotado da *Arabidopsis thaliana* cromossomo um, com o uso do analisador desenvolvido e das opções disponíveis nas bibliotecas do BioPythom, BioPerl, BioJava. Os resultados são apresentados na FIGURA 25.

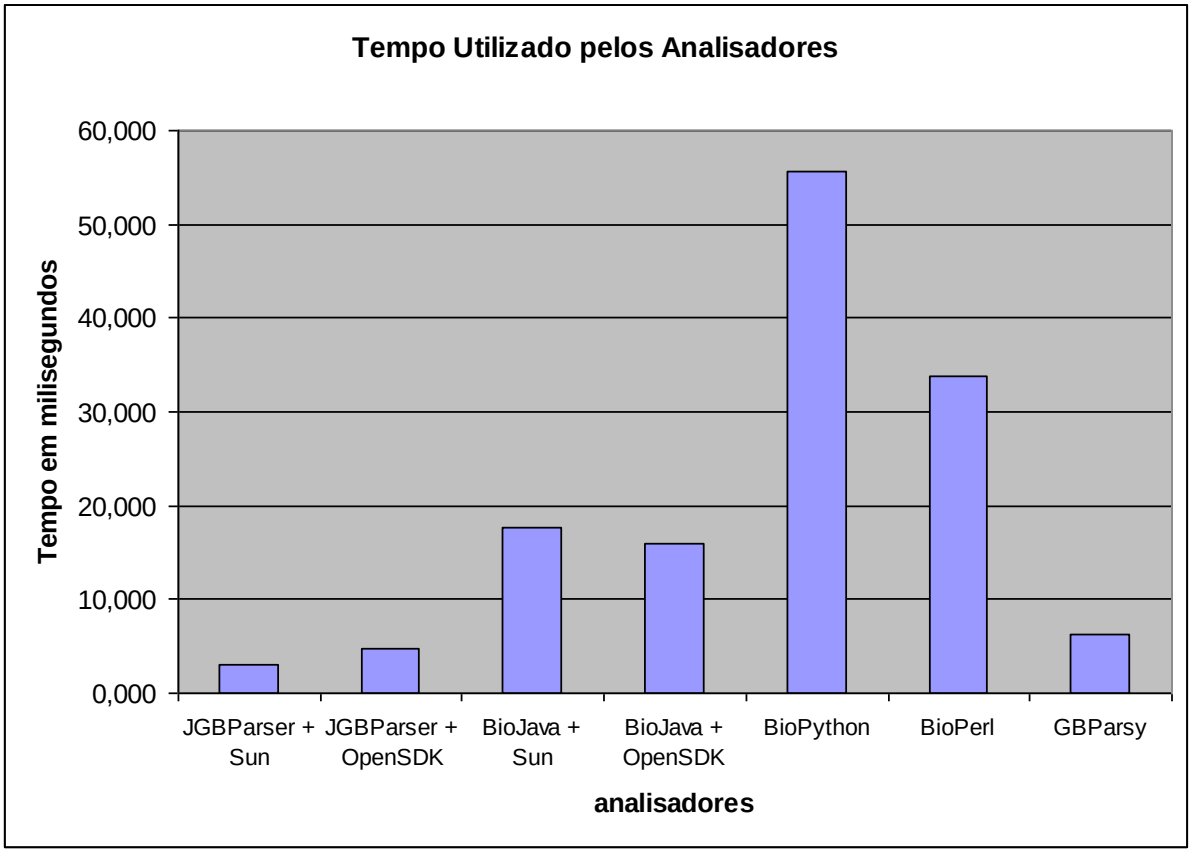

FIGURA 25 - TEMPO UTILIZADO PELOS ANALISADORES

FONTE: o autor (2010).

Os tempos descritos na FIGURA 25 foram obtidos com o uso do comando *time* do Linux. Os parâmetros de memória –Xms2048m foram adotados porque a versão em BioJava não conseguiu carregar o arquivo com os valores padrões e com a opção de 1024m.

Deve-se considerar que as bibliotecas têm propostas diferenciadas, que as estruturas de dados resultantes são diferentes, de maneira que, em geral, detecta-se a não completeza da análise. Obviamente, essas falhas podem ser justificadas pelos quesitos tempo, recursos de memória necessários e propósito da biblioteca. Por exemplo, o BioJava, até a versão 1.7.1, ignorou os campos *source* e *organism*[18]. A proposta do analisador JGBParser é dar viabilidade à conversão dos arquivos textos do GenBank para uma estrutura de banco de dados; consequentemente, todos os campos previstos na especificação 178 do NCBI GenBank foram tratados.

[18] BioJava – limitações na carga dos arquivos GenBank – obtidos no endereço http://www.biojava.org/wiki/BioJava:BioJavaXDocs, consultado em 04/07/2010.

As diferenças a serem consideradas entre o JGBParser e a versão do BioJava são das opções de modelagem das estruturas para representar as informações. Em decorrência dessas opções, o BioJava consome mais memória. Para cada nucleotídeo da sequência, existe um objeto da Gramática associado, em vez de um *char*. No quesito tempo de processamento, o JGBParser está 4,95 vezes mais rápido que a versão do BioJava.

Nas outras comparações, estão desconsideradas as diferenças de linguagens e de modelagem. Nesse contexto, o JGBParser está 15.63 vezes mais rápido que o BioPython e 9.47 vezes mais eficiente que a versão em BioPerl e duas vezes mais rápido que a proposta do GBParsy.

O analisador foi utilizado para testes nos 2.091 arquivos de genomas completos disponíveis no servidor de arquivos do GenBank. A aplicação de testes levou 29 minutos e trinta segundos, mensurado com o comando time do Linux, para analisar os 2.091 arquivos contendo 6.660 referências bibliográficas, 8.051.629 de anotações (*features*), sendo 3.786.732 de genes e 3.636.654 de CDS. As anotações (*features*) estavam compostas por 46.551.006 de qualificadores/descritores (*qualifiers*). A aplicação ainda paralisava dez ms entre um arquivo e outro para aumentar as possibilidades da operação de limpeza de memória (*Garbage Collection*[19]). Em tempo contado pela própria aplicação, o tempo de processamento para carga e análise dos arquivos foi de 26 minutos e 15 segundos.

O teste de vazamento de memória foi realizado com os recursos da telemetria da máquina virtual disponível no ambiente de desenvolvimento NetBeans. Para o teste, o programa percorre uma estrutura de diretórios equivalente à disponível no serviço FTP do NCBI GenBank; carrega cada um dos arquivos com extensão “gbk”; analisa as informações e grava em um arquivo o respectivo resumo: nome do arquivo, taxonomia, quantidade de referências, quantidades de *features*, tamanho da sequência, o tempo de análise, entre outras. Após o processamento de cada arquivo, foi chamado o *Garbage Collector* (System.gc()) e solicitado que a thread principal dormisse por dez ms.

Em uma execução com aproximadamente vinte minutos, realizada para produzir a FIGURA 26, utilizando os parâmetros –Xms1024m e –Xmx1024m [25], pode-se observar que a pilha (em roxo) e o heap (em rosa) não apresentam

[19] Tuning Garbage Collection with the 5.0 JavaTM Virtual Machine [http://java.sun.com/docs/hotspot/gc5.0/gc_tuning_5.html]

consumo crescente de memória, o que ocorre quando existe vazamento de memória em aplicações Java, conforme descrito por Jiri Sedlacek (2010).

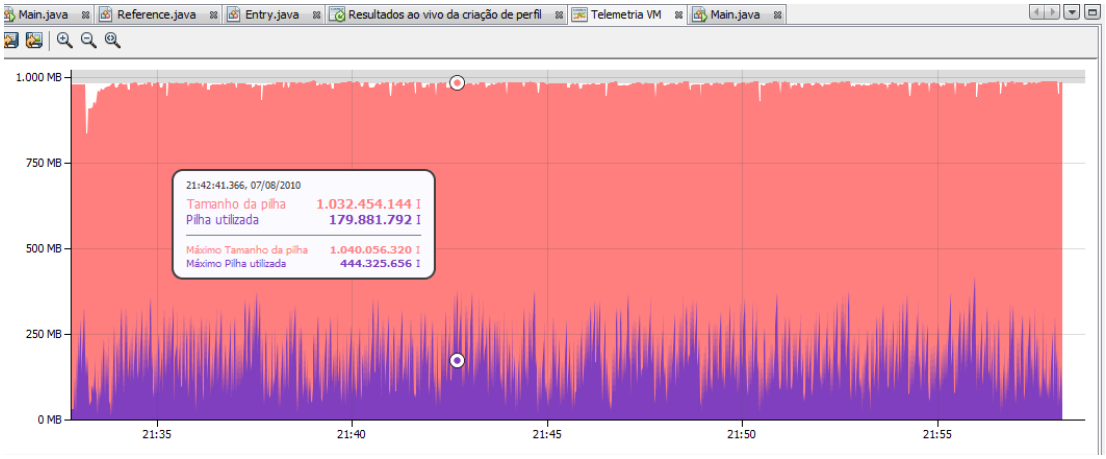

FIGURA 26 - IMAGEM CAPTURADA DA TELEMETRIA DA MÁQUINA VIRTUAL NO AMBIENTE DE DESENVOLVIMENTO NETBEANS

FONTE: o autor (2010).

6 CONCLUSÃO

Construiu-se um banco de dados relacional com as informações disponíveis no NCBI GenBank dos genomas sequenciados de organismos procariotos, integrado com ferramentas de apoio que asseguram sua atualização e sincronização de dados.

As estratégias de sincronização reduzem a 20% o tráfego de rede e o tempo de sincronização. De fato, os estudos apontam maiores possibilidades de economia se a metodologia for adotada pelo NCBI. O analisador desenvolvido reduziu de 55 segundos para 6 segundos o tempo de processamento do arquivo que contém as informações da *Escherichia coli*.

O analisador desenvolvido não apresenta fragmentação ou vazamento de memória, o que possibilita o emprego desse recurso em servidores *web* e aplicações de alta demanda.

A operação de carga dos dados em banco de dados que fora relatada pela equipe do Biowarehouse, de 68 horas, agora é possível ser realizada em menos de oito horas, ou seja, o processo é 750% mais breve. Apesar das grandes diferenças nos contextos – diferenças nos equipamentos e na quantidade de dados –, o ganho real em desempenho é fato concreto.

Por meio desta dissertação, propõe-se a distribuição das informações do NCBI GenBank no formato de arquivos de cópia de segurança do banco de dados.

Vale destacar, ainda, que os interessados nas respectivas informações estruturadas no modelo relacional, apoiadas nas tecnologias dos Sistemas de Gerenciamento de Banco de Dados, poderão obtê-las diretamente nos arquivos em servidores de acesso público e restaurá-los em seus computadores pessoais. Desse modo, os pesquisadores poderão ter disponíveis todas as informações do banco de dados em aproximadamente uma hora, tempo estimado para realizar a cópia do arquivo de segurança e restaurá-lo no servidor MySQL.

Com a aplicação e uso dos recursos dos SGBD, as pesquisas e os levantamentos que duravam semanas agora são realizados em minutos por meio de consultas realizadas em SQL.

O modelo e as ferramentas efetivam uma das atividades da etapa de pré-processamento prevista no processo de descoberta de conhecimento (KDD). As

informações disponíveis nos bancos de dados públicos e os trabalhos precisam favorecer o inter-relacionamento existente nos diversos estudos biológicos, ao contrário da lógica especializada e fragmentada que atualmente se pratica. A organização das informações deve ampliar as possibilidades de aplicação de técnicas de descoberta de conhecimento (KDD).

Se considerada a quantidade de pessoas ao redor do mundo que diariamente baixam arquivos do NCBI, analisam-nos, extraem informações e armazenam em seus bancos de dados locais, perceber-se-á o significativo número de tarefas repetidas sendo realizadas por diferentes pessoas, bem como o tempo despendido nestes processos secundários, em lugar de estarem realizando suas pesquisas.

Pelo fato de valorizar o tempo dos usuários da bioinformática, bem como por obter economia expressiva em processos de pesquisa de dados, torna-se legítima a afirmação de que este trabalho contribui para um novo modelo de organização, acesso e distribuição das informações do NCBI GenBank.

7 PESQUISAS FUTURAS

A integração com outros bancos de dados será importante para aplicações futuras.

A construção de dicionários de sinônimos para os termos existentes nas anotações ampliarão o escopo de pesquisa e melhorarão os resultados obtidos no banco de dados.

O desenvolvimento de interfaces também se faz necessário para a utilizar os recursos do banco de dados pelo usuário final, sem o necessário conhecimento de SQL exigido na versão atual desse trabalho.

O desenvolvimento de analisadores em C/C++, a partir do modelo e técnicas exploradas neste trabalho, pode obter desempenho melhor que os aqui alcançados.

Pesquisa e desenvolvimento de métodos de carga que explorem os recursos de cada um dos SGBD suportados.

Desenvolver métodos e aplicações para a realização da mineração de dados para fins da descoberta de novos conhecimentos.

Por fim, vale considerar a pertinência da publicação dos resultados e as metodologias aplicadas no desenvolvimento desta dissertação.

REFERÊNCIAS

ALTMAN, Russ B. *A Curriculum for Bioinformatics: The Time is Ripe* (Editorial). ***Bioinformatics***, v. 14, n. 7, 1998, p. 549-550.

ALTSCHUL, S. F. et al. *Basic local alignment search tool.* ***Journal of Molecular Biology*****.** v. 215, 1990, p. 403-410.

APACHE. **Jakarta Commons Net.** Disponível em: <http://commons.apache.org/net/>. Último acesso 08/09/2010.

BENSON, D. A.; MIZRACHI, I. K.; LIPMAN, D. J.; OSTELL, J.; RAPP, B. A.; WHEELER, D. L. *GenBank.* ***Nucleic Acids Research,*** v. 28, 2000.

BENSON, D. A.; MIZRACHI, I. K.; LIPMAN, D. J.; OSTELL, J.; WHEELER, D. L. *GenBank: update.* ***Nucleic Acids Research,*** v. 32, 2004.

BENSON, D. A.; MIZRACHI, I. K.; LIPMAN, D. J.; OSTELL, J.; WHEELER, D. L. *GenBank.* ***Nucleic Acids Research,*** v. 35, 2007.

BENSON, D. A.; MIZRACHI, I. K.; LIPMAN, D. J.; OSTELL, J.; SAYERS, Eric W. *GenBank.* ***Nucleic Acids Research,*** v. 38, 2010.

BIODATABASE. **MetaBase.** Disponível em: < http://biodatabase.org/index.php/Main_Page>. Último acesso: 07/09/2010.

BIOJAVA. **BioJava.** Disponível em: <http://www.biojava.org/>. Último acesso: 06/09/2010.

BIOPERL. **BioPerl.** Disponível em: <http://www.bioperl.org/>. Último acesso: 06/09/2010.

BIOPYTHON. **BioPython.** Disponível em: <http://www.biopython.org/>. Último acesso: 06/09/2010.

BIOSQL. **BioSQL.** Disponível em: <http://www.biosql.org/wiki/Main_Page>. Último acesso: 06/09/2010.

BIOWAREHOUSE. ***BioWarehouse – Database integration for bioinformatics.*** Disponível em: <http://biowarehouse.ai.sri.com/>. Último acesso: 06/09/2010.

BIOWAREHOUSE. ***GenBank loader developer manual.*** Disponível em: < http://biowarehouse.ai.sri.com/repos/genbank-loader/docs/Genbank_Developer_Manual.html>. Último acesso: 10/09/2010.

BIRNEY, Ewan; CLAMP, Michele; HUBBARD, Tim. *Databases and tools for browsing genomes.* ***Annual Review of Genomics and Human Genetics*****.** v. 3, 2002, p. 293-310.

BUTZEN, Fred; FORBES, *Dorothy.* ***The Linux database*****.** Mis:Pres: New York, 1997.

CATANHO, Marcos; MIRANDA, Antônio B. de. *Comparing genomes: databases and computational tools for comparative analysis of prokaryotic genomes.* ***Reciis – Eletronic Journal of Communication Information & Innovation in Health.*** v. 1, n. 2, sup. 1, Jul.-Dec., 2007, p. sup. 334 – sup. 355.

COCHRANE, G. R.; GALPERIN, M. Y. *The 2010 nucleic acids research database issue and online database collection: a community of data resources.* ***Nucleic Acids Research.*** v. 38, 2009. DOI 10.1093/nar/gkp1077.

COCHRANE, Guy et al. *Petabyte-scale innovations at the European Nucleotide Archive.* ***Nucleic Acids Research***, v. 37, suppl. 1, 2008

COCK, P. J. A. et al. *Biopython: freely available python tools for computational molecular biology and bioinformatics.* ***Bioinformatics***, v. 25, n. 11, 2009, p. 1422-1423.

CREATIVE COMMONS. ***CopyLeft.*** Disponível em: <http://www.creativecommons.org.br/>. Último acesso: 06/09/2010.

CROW, J. A.; STAGGS, R. A.; et al. *Design and Implementation of a Simple Relational Database for GenBank-Derived Data*. **Scientific Literature Digital Library**, 2007. *Disponível em:* < http://en.scientificcommons.org/42754928>, Último acesso: 06/09/2010.

D'ADDABBO, P. et al. *GeneRecords: a relational database for GenBank flat file parsing and data manipulation in personal computers.* ***Bioinformatics.*** v. 20, n. 16, 2004.

DANDSLINI, G. A.; PACHECO, R. C. S.; MARTINS, A.; GAUTHIER, F. A.; BARCIA, R. M.; RAITTZ, R. T.; SOUZA, J. A. *Fan: Learning by means of free associative neurons.* ***Congress On Computational Intelligence***, FUZZ-IEEE'98, 1998.

DATE, C. J. **Introdução a sistemas de banco de dados.** 7. ed. trad. Vandenberg Dantas de Souza. Rio de Janeiro: Campus, 2000.

DELAMARO, M. **Como construir um compilador utilizando ferramentas Java.** São Paulo: Novatec, 2004. ISBN 85-7522-055-1.

DNA DATA BANK OF JAPAN. **Introduction of DDBJ**. Disponível em: <http://www.ddbj.nig.ac.jp/intro-e.html>. Último acesso: 08/09/2010.

DOEDERLEIN, Osvaldo Pinali. Persistência turbinada: DAOs otimizados, caching e JDBC zvançado. **Revista Java Magazine.** v. 25, junho, 2005.

DOEDERLEIN, Osvaldo Pinali. Persistência turbinada II: o lado negro da força JDBC. **Revista Java Magazine.** v. 26, julho, 2005.

EBI. ***EMBL Nucleotide sequence database.*** Disponível em: <http://www.ebi.ac.uk/embl/>. Último acesso> 10/09/2010.

ELMASRI, Ramez E. & NAVATHE, Shamkant. **Sistemas de banco de dados.** 4. ed. Addison-Wesley, 2005.

ENTERPRISE DISTRIBUTED TECHNOLOGIES. ***Free Java FTP library gives Java developers.*** Disponível em: <http://www.enterprisedt.com/>. Último acesso: 08/09/2010.

FERRO, Milene. **Desenvolvimento e validação de protocolos para a anotação automática de sequências Orestes de Eimeria spp. de galinha doméstica.** 125 f. Dissertação (Mestrado em Ciências). Instituto de Ciências Biomédicas da Universidade de São Paulo, São Paulo, 2008.

FRESHMEAT. ***Florent cueto Java FTP API.*** Disponível em: <http://freshmeat.net/users/florentcueto/>, último acesso: 08/09/2010.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. **Padrões de Projeto**. Bookman, 2000.

GANE, Chris; SARSON, Trish. **Análise estruturada de sistemas.** trad. Gerry Edward Tompkins. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 1983.

GEHLEN, M.; RIGO, L. et al. *Development of a comprehensive nif gene database.* ***12th International Symposium on BNF with Non-Legumes/2nd INCT - BNF symposium.*** Buzios, outubro de 2010. Aceite Disponível em: <http://www.isbnfnl2010.org/abstracts_approved.htm>.

GLUB TECH. ***Secure FTP bean*****.** Disponível em: <http://www.glub.com/products/bean/>. Último acesso: 08/09/2010.

GOLDSCHMIDT, Ronaldo; PASSOS, Emmanuel. ***Data mining:*** **um guia prático.** Rio de Janeiro: Elsevier, 2005.

GOSLING, James; JOY, Bill, STEELE, Guy; BRACHA, Gilad. ***The Java language specification.*** 2. ed. *sítio da Oracle/Sun*. Disponível em: <http://java.sun.com/docs/books/jls/second_edition/html/j.title.doc.html>. Último acesso: 08/09/2010.

GUIMARÃES, Ana Carolina Ramos. **Identificação, classificação e anotação de enzimas análogas em tripanossomatídeos.** 106 f. Dissertação (Mestrado em Ciências). Pós-Graduação em Biologia Celular e Molecular. Instituto Oswaldo Cruz, Rio de Janeiro, 2006.

GUSDB. ***The genomics unified schema.*** Disponível em: <http://www.gusdb.org/about.php>. Último acesso: 06/09/2010.

HOLLAND, Richard C. G. et al. *BioJava: an open-source framework for bioinformatics.* ***Bioinformatics*****.** v. 24, n. 18, 2008, p. 2096-2097.

KIMBAL, Ralph. *The 38 Subsystems of ETL.* ***Intelligent Enterprise.*** Disponível em: http://intelligent-enterprise.informationweek.com/showArticle.jhtml?articleID=55300422. Último acesso: 08/09/2010.

KRUSINSKA, Ewa et al. *Integrated approach for designing medical decision support systems with knowledge extracted from clinical databases by statistical methods.* ***Proc Annu Symp Comput Appl Med Care***, 1991.

JAVA COMMUNITY PROCESS. ***JSR 317: JavaTM persistence 2.0.*** Disponível em: <http://www.jcp.org/en/jsr/detail?id=317> . Último acesso: 12/09/2010.

JSCAPE. ***Secure FTP factory*****.** Disponível em: < http://www.jscape.com/sftp/>. Último acesso: 08/09/2010.

LASZEWSKI Gregor von; FOSTER, Ian; GAWOR, Jarek; LANE, Peter. ***Concurrency and computation:*** *Practice and experience.* 13(89): p. 643- 662, 2001.

LEE, Tae-Ho; KIM, Yeon-Ki; NAHM, Baek Hie. *GBParsy: A GenBank flatfile parser library with high speed.* ***BMC Bioinformatics,*** *v.* 9 n. 321, 2008. doi:10.1186/1471-2105-9-321.

LEE, Thomas J et al. BioWarehouse: a bioinformatics database warehouse toolkit. ***BMC Bioinformatics.*** *v. 7, p. 170,* 2006. doi:10.1186/1471-2105-7-170

LEMOS, Melissa. ***Workflow*** **para bioinformática.** Rio de Janeiro, 2004. 239 f. Tese (Doutorado em Informática). Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

MANN, Nathan. ***Developing a Database for GenBank Information***. Louisville, KY, USA, 2004. 67 f. Dissertação (mestrado em Engenharia). Departamento de Engenharia da Computação e Ciência da Computação. University of Louisville.

MARTIN, James. ***Computer data-base organization*****.** Prentice-Hall, 1975

MATSUBARA, Y; INDO, Y; NAITO, E; OZASA, H; GLASSBERG, R; VOCKLEY, J; IKEDA, Y; KRAUS, J; TANAKA K. *Molecular cloning and nucleotide sequence of cDNAs encoding the precursors of rat long chain acyl-coenzyme A, short chain acyl-coenzyme A, and isovaleryl-coenzyme A dehydrogenases. Sequence homology of four enzymes of the acyl-CoA dehydrogenase family*. ***The Journal Biological Chemistry***, 25;264(27):16321-31. Set/1989

MCCLUSKEY, Glen. ***Tuning Java I/O Performance***. *Oracle – Sun Developer Network (SDN)*, 1999. Disponível em: <http://java.sun.com/developer/technicalArticles/Programming/PerfTuning/> Último acesso: 06/09/2010.

MARDIS, Elaine R. *Next-Generation DNA Sequencing Methods*. ***Annu. Rev. Genomics Hum. Genet***. n. 9:387–402, 2008. dói: 10.1146/annurev.genom.9.081307.164359

MEDEIROS, L. F. **Banco de dados: princípios e prática.** Curitiba: IBPEX, 2007.

MIZRACHI,Ilene. ***GenBank: the nucleotide sequence database.*** *Disponível em:* <http://www.ncbi.nlm.nih.gov/bookshelf/br.fcgi?book=handbook&part=ch1#GenBank_ASM>. Último acesso: 06/09/2010.

MUNRO, Barbara Hazard. ***Statistical methods for health care research.*** 5. ed. Lippincott Williams & Wilkins, 2004. Disponível no *Google Books*. Último acesso: 10/09/2010.

NAGARAJAN, N.; COOK, C.; BONAVENTURA, M.; GE, H.; RICHARDS, A.; BISHOP-LILLY, K. A.; DESALLE, R.; READ, T. D.; POP, M.. *Finishing genomes with limited resources: lessons from an ensemble of microbial genomes*. ***BMC Genomics***. n. 11: 242, 2010. doi: 10.1186/1471-2164-11-242.

NCBI. ***A science primer - just the facts: a basic introduction to the science underlying*** NCBI Resources. Disponível em: <http://www.ncbi.nlm.nih.gov/About/primer/bioinformatics.html>, (p. w1). Último acesso: 04/09/2010.

NCBI. ***GenBank growth.*** Disponível em: <http://www.ncbi.nlm.nih.gov/genbank/genbankstats.html>. Último acesso: 04/09/2010.

NCBI. ***Reference sequence (RefSeq)*** – key.html. Disponível em: <http://www.ncbi.nlm.nih.gov/RefSeq/key.html#method> . Último acesso: 08/09/2010.

NELSON, David L.; COX, Michael M. ***Lehninger principles of biochemistry***. 4. ed. W. H. Freeman, 2004.

NORGUET, Jean-Pierre. *Java FTP client libraries reviewed.* ***JavaWorld*****,** 2003. Disponível em: http://www.javaworld.com/javaworld/jw-04-2003/jw-0404-ftp.html. Último acesso: 08/09/2010.

NORGUET, Jean-Pierre. *Update: Java FTP libraries Benchmarked.* ***JavaWorld***, 2006. Disponível em: <http://www.javaworld.com/javaworld/jw-03-2006/jw-0306-ftp.html>. Último acesso: 08/09/2010.

ODF ALLIANCE. ***Open document format.*** Disponível em: <http://www.odfalliance.org/>. Útimo acesso: 06/09/2010.

ORACLE. ***Core J2EE Pattern Catalog. Core J2EE Patterns - data access object.*** Disponível em: <http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html>. Último acesso: 02/09/2010.

OXFORD JOURNALS, 2010 NAR ***Database summary paper category list.***, Disponível em: <http://www.oxfordjournals.org/nar/database/cat/1>. Último acesso: 05/10/2010.

PAN, Wei. *A comparative review of statiscal methods for discovering differentially expressed genes in replicated microarray experiments.* ***Bioinformatics.*** v. 18, n. 4, 2002.

PEDROSA, F. O. et al. ***The genome of Herbaspirillum seropedicae SmR1, an endophytic, nitrogen-fixing, plant-growth promoting beta-Proteobacteria.*** Disponível em: GenBank sob o nome de NC_014323.gbk. Último acesso: 08/09/2010.

PENDER, T. **UML - a bíblia.** trad. Daniel Vieira. Rio de Janeiro: Elsevier, 2004.

PROSDOCIMI, F. et al. Bioinformática: Manual do Usuário. **Biotecnologia Ciência & Desenvolvimento.** n. 29, 2002.

RAITTZ, Roberto Tadeu. **Fan 2002: Um modelo neuro-fuzzy para reconhecimento de padrões.** Tese (Doutorado em Engenharia de Produção), Universidade Federal de Santa Catarina, Florianópolis, 2002.

SALZBERG, S.; DELCHER, A.; KASIF, S.; White, O. *Microbial gene identification using interpolated markov models.* ***Nucleic Acids Research***, v. 26, n. 2, jan./1998, p. 544–548.

SCHNOES, Alexandra M.; BROWN, Shoshana D.; DODEVSKI, Igor; BABBITT, Patrícia C. *Annotation error in public databases: misannotation of molecular function in enzyme superfamilies.* ***Plos Computational Biology*****.** v. 5, n. 12, dezembro, 2009.

SEDLACEK, Jiri. *Uncovering Memory Leaks Using NetBeans Profiler*. **NetBeans Documents and Support**, 2010. Disponível em <http://netbeans.org/kb/articles/nb-profiler-uncoveringleaks_pt1.html> Último acesso: 08/09/2010.

SILBERSCHATZ, Abraham; KORTH, Henry F; SUDARSHAN, S. **Sistemas de banco de dados.** 5. ed. trad. Daniel Vieira. Rio de Janeiro: Elsevier, 2006.

SOURCE FORGE. *JFtp - The Java Network Browser*. Disponível em: <http://j-ftp.sourceforge.net/>. Último acesso: 08/09/2010.

SOURCE FORGE. *jvFTP:* ***Bea Petrovicova.*** Disponível em: < http://jvftp.sourceforge.net/>. Último acesso: 08/09/2010.

STAJICH, J. E. et al. *The bioperl toolkit: perl modules for the life sciences.* ***Genome research***, v. 12, n. 10, 2002, p. 1611-1618.

STOESSER, Guenter et al. *The EMBL Nucleotide sequence database.* ***Nucleic Acids Research.*** v. 26, n. 1, 1998.

SYBASE. ***Sybase powerDesigner conceptual data model -*** *User's Guide. Version* 9. 2001

TANENBAUM, A. S. & WOODHULL, A. S. **Sistemas operacionais: projeto e implementação.** 2. ed. trad. Edson Furmankiewicz. Porto Alegre: Bookman, 2000.

UNIVERSITÀ DI BOLOGNA. ***GeneRecords 1.0 Tutorial.*** Disponível em: <http://apollo11.isto.unibo.it/software/GeneRecords_1.0/Win_Tutorial/Win_Tutorial.htm>. Último acesso: 11/09/2010.

VILLAS, Marcos Vianna; VILLASBOAS, Luiz Felipe P. **Programação:** conceitos, técnicas e linguagens. Rio de Janeiro: Campus, 1988.

WANG, L.; RODRIGUEZ-TOMÉ, P.; REDASCHI, N.; MCNEIL, P.; ROBINSON, A. & LIJNZAAD, P. *Accessing and distributing EMBL data using CORBA (common object request broker architecture).* ***Genome Biology***. v. 1, n. 5, 2000.

WEISS, Vinicius Almir. ***Estratégias de Finalização da Montagem do Genoma da Bactéria Diazotrófica Endofítica Herbaspirillum seropedicae SmR1*****.** Curitiba, 2010. 72 f. Dissertação (mestrado em Ciências - Bioquímica). Departamento de Bioquímica. Universidade Federal do Paraná.

WORLD WIDE WEB CONSORTIUM. ***File transfer protocol*****.** Disponível em: <http://www.w3.org/Protocols/rfc959/>. Último acesso: 08/09/2010.

WOTSIT ORG. ***The programmer's file and data format resource.*** Disponível em: <http://www.wotsit.org/>. Último acesso: 06/09/2010.

YONG, Chu Shao. **Banco de dados:** organização sistemas e administração. São Paulo: Atlas, 1983.