Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-graduação em Ciência de Computação

Bio-TIM - Ambiente para convergência de informações em Bioinformática

Gustavo Borges de Oliveira

São Carlos
2006

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

O48ba

Oliveira, Gustavo Borges de.
Bio-TIM – Ambiente para convergência de informações em Bioinformática / Gustavo Borges de Oliveira -- São Carlos : UFSCar, 2006.
88 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2005.

1. Tipos específicos de bancos de dados. 2. Data warehousing. 3. Bioinformática. 4. Framework (Programa de computador). I. Título.

CDD: 005.75 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Bio-TIM – Ambiente para convergência de informações em Bioinformática"

GUSTAVO BORGES DE OLIVEIRA

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

Prof. Dr. Mauro Biajiz
(Orientador – DC/UFSCar)

Profa. Dra. Marilde Terezinha Prado Santos
(DC/UFSCar)

Prof. Dr. Nalvo Franco de Almeida Junior
(DCT/UFMS)

São Carlos
Agosto/2005

Dedico este trabalho ao meu irmão Felipe.

“Sua luz sempre brilhará”

Agradecimentos

Aos meus pais, Dr. José Antônio e Ligia, pelas oportunidades que me proporcionaram e pelo total apoio em todas as minhas decisões. Espero um dia poder retribuir tudo que vocês já fizeram por mim.

À minha irmã Carolina por ser além de minha irmã, ser minha amiga e sempre me aconselhar.

Ao Prof. Dr. Mauro Biajiz e à Prof. Dra. Marilde Santos pelo apoio, aulas e ajuda nos momentos que eu mais precisei, saibam que nunca esquecerei.

Resumo

Atualmente, vários genomas de seres vivos estão sendo mapeados gerando enormes quantidades de dados, as quais são armazenadas em diferentes fontes de informações. O fato dos dispositivos de armazenamento de dados serem heterogêneos e distribuídos resulta na necessidade de criação de técnicas de consulta e integração de dados para que os biólogos possam usufruir de maiores e melhores recursos durante o processo de pesquisa. Este trabalho propôs um ambiente de bioinformática para a integração de diferentes fontes de dados e sua reorganização em uma fonte centralizada propiciando consultas flexíveis e eficientes. O ambiente foi denominado Bio-TIM, o qual é composto por três camadas, sendo a principal denominada “Mediador”. Ela é composta por um gerenciador de conexões a sistemas gerenciadores de bancos de dados, por tradutores, por um Data Warehouse e por um banco de dados específico.

Abstract

Currently, some genomas of creatures are being mapping, so enormous amounts of data are being generated. This data is sometimes stored in different data sources and at many times, these sources are heterogeneous and distributed, which results in the necessity of the use of new query techniques and integration of data. After this, the biologists can usufruct greaters and better resources during the research process. This work has the proposal of to build a Bioinformatics environment for the integration of different data sources and their reorganization in a centered source that supports a flexible and efficient manner to build queries. The environment is called as Bio-TIM, which is composed by three layers. The main one is called "Mediator" and it is composed by a connection manager for databases management systems, by wrappers, by a Data Warehouse and by a specific database.

Lista de Figuras

Lista de Tabelas

Sumário

1. Introdução

Os trabalhos de identificação de Genomas tiveram seus inícios por volta de uma década atrás. Inicialmente o objetivo era seqüenciar todo o Genoma Humano, caracterizando um grande projeto de pesquisa básica para posteriormente os cientistas poderem almejar o desenvolvimento de curas para doenças através de tratamentos genéticos e produção de novos medicamentos. Com o passar dos anos, o genoma de outras espécies também começou a ser seqüenciado, com objetivos similares, ou seja, o de fornecer conhecimento sobre genes e sua locação no genoma para obtenção de melhorias genéticas para os mais diversos fins.

Para muitos, o trabalho de conclusão do seqüenciamento do genoma traria benefícios imediatos e todos iriam poder se beneficiar deste conhecimento [LENGAUER00A]. Entretanto, este foi apenas o passo inicial para uma etapa mais importante, a qual tem como objetivo, entre outras coisas, conhecer como o ácido desoxirribonucléico (DNA), enzimas, proteínas, aminoácidos e outras moléculas interagem no organismo dos seres vivos e como o homem pode influenciar ou alterar de algum modo estas interações.

Essa fase inicial foi denominada era pré-genômica, uma fase de pesquisa básica. Com o avanço das pesquisas, grandes volumes de dados foram sendo produzidos e disponibilizados, gerando forte demanda por ferramentas para a manipulação dos mesmos. Iniciou-se, então, uma nova fase que vem sendo denominada pós-genômica.

As características descritas acima estabelecem que a integração dos dados para realizar uma análise mais acurada é mais uma tarefa a ser cumprida [LENGAUER00B]. Outro ponto importante é que a descoberta de conhecimento constitui-se em um dos principais objetivos da era pós-genômica. Para esse fim, uma das principais tarefas constitui-se na implementação de técnicas computacionais que permitam a integração e interpretação do grande volume de dados gerado.

1.1 Motivação

Para exemplificar a ordem de grandeza dos dados, o genoma humano é composto por aproximadamente 3 bilhões de bases do DNA, o qual contém informações para, por exemplo, codificar uma proteína que desempenhe uma função específica no organismo que possui o gene (parte do DNA que codifica uma proteína). Além disso, também devem ser armazenadas as relações existentes entre tais bases, como estão dispostas, quais bases formam um gene, qual a função de cada gene, no que ele poderá influir durante a vida do organismo que está inserido e que proteínas ele codifica.

Para guardar e consultar as informações genômicas descritas anteriormente, laboratórios utilizam meios próprios, os quais atendem às necessidades de suas pesquisas com desempenho e custos adequados. Sabendo que as pesquisas tomam vertentes diferentes e que a disponibilidade de recursos para os laboratórios divergem consideravelmente, torna-se complexo, mas não impossível, haver uma padronização dos dados armazenados pelos diversos laboratórios com o intuito de integrar e compartilhar conhecimentos. A importância desta integração tem fundamentos, pois um gene cuja proteína é codificada por um determinado organismo, pode codificar a mesma proteína em um organismo diferente. Por exemplo, um laboratório responsável pelo seqüenciamento de uma planta pode trocar informações com um laboratório responsável pelo seqüenciamento do camarão branco. Visando essa troca de informações, através do uso de tradutores (*wrappers*), pode-se padronizar os dados, seja no armazenamento, através da integração de fontes heterogêneas, ou na fonte utilizada para consultas.

O volume de dados armazenado e manipulado, seja por Sistemas Gerenciadores de Banco de Dados (SGBD), que podem ser definidos, segundo [ELMASRI99], como um conjunto de programas que permite a usuários criar e manter um banco de dados, ou por arquivos semi-estruturados, cresce continuamente ano a ano; a Figura 1, ilustra o crescimento

do volume de dados armazenado no Genbank [GBK03], que é um banco de dados público, mantido pelo National Center of Biotechnology Information (NCBI), utilizado para o armazenazenamento de seqüências de DNA. Além disso, tem aumentado a demanda para novos tipos de dados, mais complexos, que requerem maior custo computacional e mais espaço para o seu armazenamento. Dentre esses, os mais comuns são: os próprios dados genômicos (como seqüências de DNA e proteínas), informações geo-referenciadas, séries temporais, dados de telemetria, dados de engenharia e estatísticos, entre outros, sendo o primeiro o objeto de estudo deste trabalho.

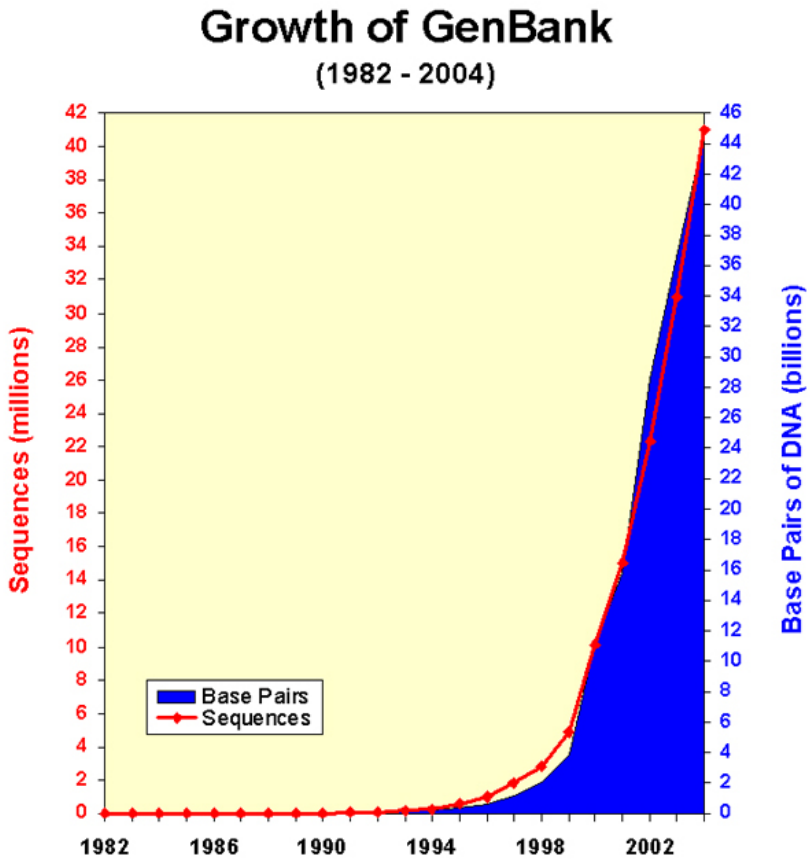

Figura 1 – Demonstração do crescimento do Genbank [GBK03]

O armazenamento de seqüências de DNA vem sendo feito, de forma muitas vezes primitiva, sem preocupação em relação ao desempenho, o qual é dependente da organização dos dados e da ferramenta utilizada. Essa falta de análise e critério deixa a cargo do servidor o

"peso" do processamento gerado pelas aplicações que têm como objetivo a realização de consultas no banco de dados do genoma. Logo, é necessária uma forma de equacionar esta distribuição de carga, facilitar e melhorar a organização dos dados, para que, por exemplo, se tenha desempenho satisfatório nas tarefas de extração de informação.

Além deste fato, pode-se perceber, através da literatura, um grande esforço na integração desses dados, como, por exemplo, o desenvolvimento de ferramentas que trabalham como um conector (*middleware*) entre as fontes de dados do genoma e as aplicações desenvolvidas em alguns centros de pesquisa e também o desenvolvimento de ferramentas de anotações que têm como objetivo a captura de informações relevantes. Uma das propostas que vem sendo desenvolvida é a construção de um repositório de dados centralizado e integrado para o auxílio nas diversas consultas. Estes repositórios são organizados através de uma estrutura multidimensional denominada Data Warehouse (maiores detalhes na seção 2.3.2.3).

Observando os diversos bancos de dados genômicos existentes, nota-se que além do volume de informação armazenada ser muito grande, ela é trabalhada de forma não muito elaborada. Na maior parte das vezes é uma tarefa árdua para o pesquisador trabalhar com aquele grande volume de dados de forma manual, pesando ainda o fato destes dados não estarem em uma única fonte de acesso, ou seja, estão dispersos pela Internet. Dentro de projetos específicos de seqüenciamento é muito interessante ter a informação centralizada para consultas rápidas e diretas, pois a informação dispersa desorienta o usuário no processo de conclusão de algum assunto [FREIER02], prejudicando assim o andamento da pesquisa.

Muitas das vezes esses dados são não homogêneos, conseqüentemente não é possível o levantamento de estatísticas (conseqüente descoberta de conhecimento) através da utilização de métodos matemáticos ou computacionais. Os sistemas atualmente existentes são em sua maioria muito restritos (não flexíveis), não permitindo que uma livre navegação dos

dados seja realizada (para a consulta de novas questões que estão sendo constantemente levantadas a partir do descobrimento de novas visões científicas, de novas teorias, proposições e formulação de hipóteses). Quando se faz necessária a elaboração de novas consultas, uma nova interface ou ferramenta deve ser desenvolvida, fazendo com que o pesquisador desvie o foco de sua pesquisa para a aprendizagem de técnicas computacionais e desenvolvimento de aplicações.

Para ilustrar esse fato, em um estudo recente, apresentado em [SONNICHSEN05], é evidenciada a importância e desafio da unificação e padronização da informação acerca do genoma. Neste trabalho, que envolveu pesquisas de diversos laboratórios espalhados pelo globo, foi realizado um estudo sobre as funções dos genes durante o processo de desenvolvimento embrionário do verme *C. Elegans*. Para este fim foram produzidos cerca de 40 mil casais do verme em laboratório, sendo que em cada um desses casais um dos genes estava desativado. Segundo análise feita em [REINACH05] sobre esse estudo, o processo de mapeamento das funções dos genes foi realizado através do preenchimento de uma matriz. Na primeira linha havia listado cada um dos genes e na primeira coluna cada um dos processos que ocorrem nos seres vivos. Após serem feitos os estudos eram preenchidas as respectivas posições da matriz, que evidenciavam a interseção entre gene e função. Em cada laboratório havia um pesquisador responsável por inserir no sistema de informação os dados provenientes das pesquisas a fim de garantir que eles ficassem homogêneos [SONNICHSEN05].

1.2 Objetivos

O laboratório de Banco de Dados do Departamento de Computação da UFSCar vem centralizando o processamento de dados do projeto (Shrimp EST Genome Project) de seqüenciamento do camarão *Litopenaeus Vannamei* (camarão branco). Para este fim está sendo utilizado o ambiente computacional denominado Pipeline Draft (definido na seção 2.3.1). Uma das características do ambiente utilizado, é que ele proporciona alguns tipos de

consultas aos diferentes participantes do projeto, mas restringe, ou simplesmente não permite, a obtenção de informações históricas do banco. Considerando essa última questão e outros aspectos levantados, foi estabelecido como objetivo deste trabalho a definição e a implementação de um ambiente de integração para a reunião de informações pertinentes e provenientes de diferentes fontes de dados genômicas através da convergência dessa informação para um Data Warehouse (DW), de forma a permitir que futuramente estes dados possam ser indexados e consultados de forma flexível .

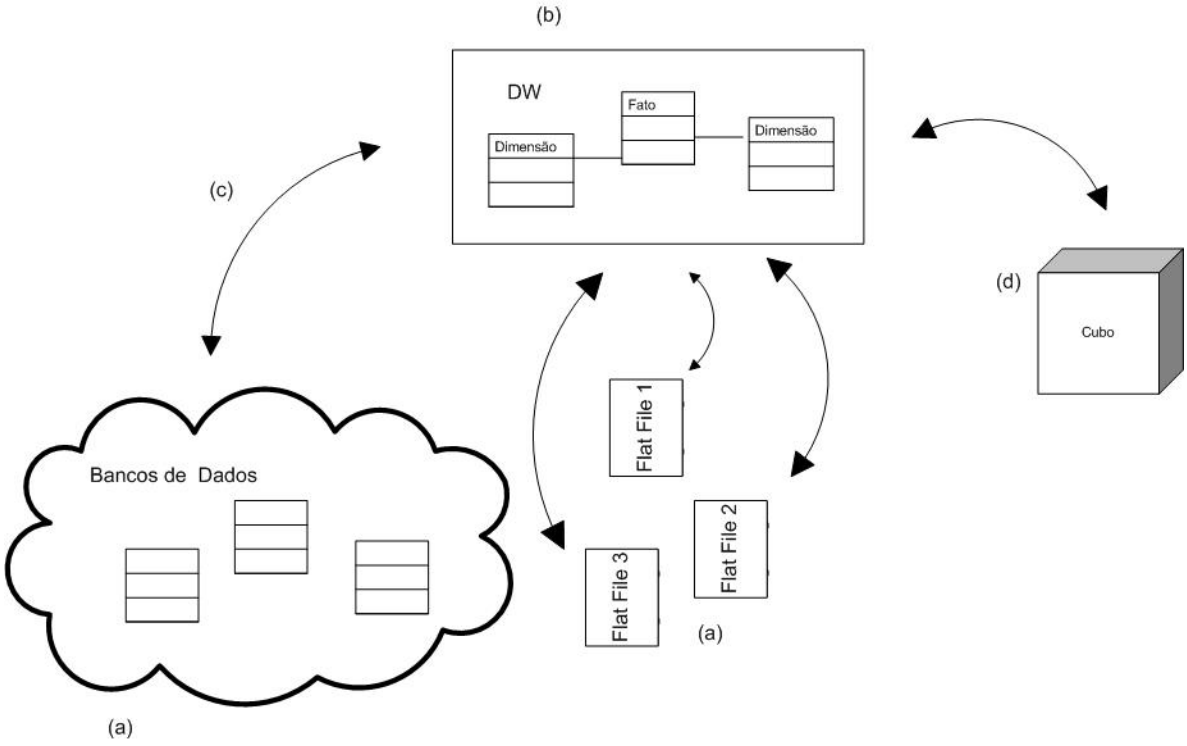

Figura 2 – Representação do fluxo de informação no projeto proposto neste trabalho

A Figura 2 ilustra a proposta. A parte (a) representa fontes definidas por bancos de dados relacionais e arquivos texto contendo dados genômicos. A parte (b) ilustra um Data Warehouse, o qual integra as diversas fontes de dados de uma maneira organizada de acordo com uma modelagem multidimensional do "problema". As setas (c) partindo das fontes (a) em direção ao DW (b) representam a extração de dados em si. Note que as fontes podem ser locais ou acessadas via web (canais de comunicação XML, por exemplo). Por fim, a parte (d)

representa o cubo (maiores detalhes na seção 2.3.2.3), o qual é montado para que as operações de consulta possam ser executadas no DW.

Para atingir os objetivos foram realizadas as seguintes etapas:

- Levantamento dos “requisitos biológicos”;
- Definição da arquitetura a ser implementada;
- Definição e criação dos tradutores que estabelecem a comunicação entre as fontes de dados e o Data Warehouse (DW);
- Implementação do ambiente, o qual foi denominado Bio-TIM.

1.3 Considerações Finais

Esta dissertação está estruturada da seguinte forma: no capítulo 2, são apresentados os conceitos utilizados no trabalho, ou seja, serão dadas as explanações básicas para a compreensão do trabalho relativas à Biologia Molecular e Celular, à Genética, Biotecnologia e Bioinformática. Posteriormente, no mesmo capítulo são apresentados os conceitos computacionais envolvidos no projeto como fontes de dados, mediadores, eXtensible Markup Language (XML), integração de dados, Data Warehouse (DW) e interface. No capítulo 3 são apresentados alguns trabalhos relacionados à integração de dados genômicos. No capítulo 4 é apresentado o projeto desenvolvido, denominado Bio-TIM. No capítulo 5 são demonstrados testes que foram realizados com algumas das possíveis consultas no DW, e no capítulo 6 serão apresentados a conclusão, enumeradas as contribuições e os trabalhos futuros

2. Conceitos Utilizados

2.1.Considerações Iniciais

Neste capítulo são apresentados os conceitos que devem ser entendidos para a correta compreensão da arquitetura proposta e implementada neste projeto. Serão explanados os conceitos básicos de biologia (incluindo a definição de termos como biotecnologia e bioinformática) que estão presentes neste projeto, de Data Warehouse (DW), eXtensible Markup Language (XML), mediador (*middleware*), integração de dados e interfaces.

2.2. Conceitos de Biologia Molecular e Celular

O intuito deste resumo de biologia é prover os conceitos biológicos básicos para a compreensão do trabalho proposto, assim como entrelaçar esses conceitos com a finalidade de possibilitar um entendimento básico sobre a síntese de proteína, seqüenciamento e a percepção dos mais variados tipos de dados que esse processo provê.

Primeiramente é feita uma explanação a respeito dos conceitos básicos envolvidos na biologia celular e posteriormente, são definidos os termos biotecnologia e bioinformática.

2.2.1. Genética

A Genética estuda os mecanismos de transmissão das características estruturais e funcionais de um organismo para as gerações subseqüentes [SBFIS98]. Essa transmissão se dá pelo código genético de cada organismo, que se localiza no núcleo das células e é conhecido como DNA. Seqüências de caracteres que representam ácidos nucleicos (DNA e ácido ribonucléico (RNA)) e proteínas são responsáveis pelas informações básicas da biologia molecular. Os ácidos nucleicos são responsáveis pela hereditariedade e as proteínas por funções estruturais ou fisiológicas nos seres vivos [SEIBEL00]. O DNA é responsável pelo armazenamento da informação genética e é composto de nucleotídeos dispostos em dois

longos filamentos unidos, antiparalelos e em forma de hélice, Figura 3, onde pontes de hidrogênio unem as bases dos filamentos opostos através de uma ligação química.

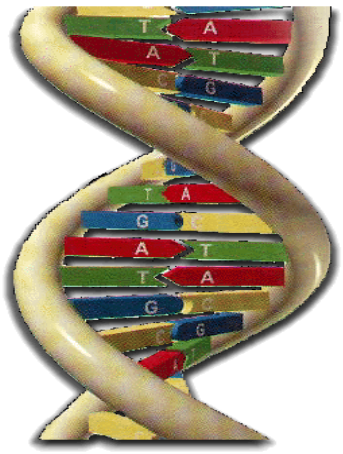

Figura 3 – Ilustração representando a dupla hélice de DNA

Uma pentose (um tipo de açúcar), um grupo fosfato e uma base nitrogenada formam um nucleotídeo que é nomeado dependendo da base nitrogenada que o compõe. Estas bases, classificadas como purinas, denominam-se Adenina (A) e Guanina (G). As pirimidinas denominam-se Citosina (C) e Timina (T). Uma purina só possui ligação com a pirimidina correspondente, e dessa forma, A e T são complementares, do mesmo modo que C e G, sendo que duas pontes de hidrogênio formam um par A-T, enquanto três pontes de hidrogênio formam um par G-C. As moléculas de DNA contêm a informação para todas as proteínas produzidas na célula. Cada códon, seqüência de três bases ao longo da fita de DNA, é um código químico para codificar um dos 20 aminoácidos que compõem as proteínas [SILVA00].

Usando várias técnicas diferentes, os cientistas foram capazes de descobrir quais as etapas que ocorrem desde a transmissão das mensagens provenientes do código do DNA até a produção final de proteínas no citoplasma da célula [BSCS73] [SPINGA98]. Uma cadeia de molécula de DNA controla a síntese de um tipo específico de RNA. Os nucleotídeos adequados pareiam-se – Adenina do DNA com a Uracila do RNA, Timina do DNA com a

Adenina do RNA, Citosina do DNA com a Guanina do RNA e finalmente a Guanina do DNA com a Citosina do RNA. Dessa forma, a molécula de RNA formada copia a mensagem contida no DNA (uma enzima intervém nessa cópia, a RNA-polimerase). Esse processo é conhecido como transcrição. Essa nova molécula de RNA, chamada RNA-mensageiro (RNAm), migra para o citoplasma onde se prende a um ribossomo, formando um molde para a síntese de proteínas. Outra molécula conhecida é o RNA-transportador (RNAt), que se liga a aminoácidos simples que estão no citoplasma. Cada aminoácido é apanhado por um tipo específico de RNAt que o encaixa no lugar adequado no RNAm. Portanto, o RNAm indica quais aminoácidos serão usados e qual o lugar que devem ocupar na molécula de proteína. Esta parte do processo de síntese de proteína é chamada de tradução. Este processo pode ser visualizado na Figura 4.

Depois da síntese, as novas moléculas de proteína se separam do ribossomo e do RNAt. Cada molécula deste RNA pega outra molécula de aminoácido, como foi feito anteriormente e a traz novamente para o ribossomo. O RNAt deve também ser codificado, de forma a reconhecer a mensagem do RNAm.

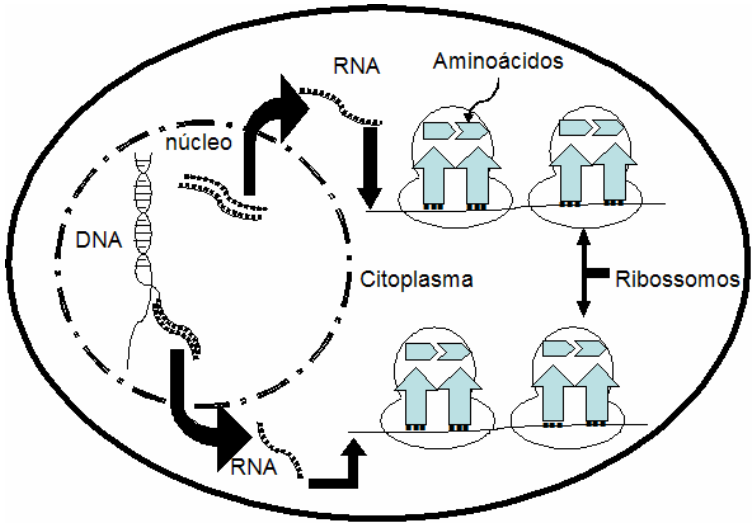

Figura 4 – Síntese de proteína [UZUNIAN91]

Desse modo, as mensagens do DNA servem para que a célula fabrique proteínas específicas, em um processo que se repete continuamente, havendo sempre síntese dos mesmos tipos de proteínas. Em resumo: DNA "fabrica" RNA e RNA "fabrica" proteína.

2.2.1.1. Seqüenciamento

O seqüenciamento do genoma se dá através da clonagem de fragmentos de DNA extraídos do núcleo das células. Existem diferentes métodos para o seqüenciamento, um dos mais utilizados atualmente, conhecido como shotgun [ADAMS00] seqüencia apenas uma parte do ácido desoxirribonucléico (DNA) por vez [COOPER94] [WATSON87] apud [LEMOS03][1], ou seja, são seqüenciadas cerca de 500 a 800 bases por vez.

O método referenciado caracteriza-se pela construção de uma biblioteca genômica utilizando o método de fragmentação aleatória do DNA que tem por objetivo gerar fragmentos de 500 a 800 pares de base que são a seguir clonados em plasmídeos (onde são inseridos material genético, no processo de clonagem) e seqüenciados massivamente, ou seja, de modo que cada base do genoma seja seqüenciada até que se garanta um determinado grau de confiabilidade nos resultados (processo conhecido como cobertura).

Como descrito, nota-se que para o DNA ser seqüenciado, ele é amplificado e posteriormente marcado em fragmentos, chamados *reads* (fragmento de DNA seqüenciado que contém de 500 a 800 bases), os quais são montados um a um a fim de obter a cadeia inicial completa. O fato é que este não é um processo simples, pois é comum ocorrerem erros no seqüenciamento e problemas durante a quebra e clonagem de *reads*, fazendo com que parte do genoma não seja devidamente seqüenciada, o que causa problemas depois na montagem do mesmo. Quando os *reads* são agrupados formam-se os *contigs* e a união de todos os *contigs* forma o genoma.

---

[1] Cooper, N.. The Human Genome Project. Univ. Science Books, Mill Valey, CA, 1994.
Watson, J., Hopkins, N., Roberts, J., Steitz, J., Weiner, A.. Molecular Biology of the Gene, 4th ed. Benjamin Cummings, Menlo Park, CA, 1987

Outro fato em relação ao seqüenciamento é que nem sempre ele é feito de forma completa, ou seja, são seqüenciadas todas as bases do material em estudo. Quando ele é muito extenso, às vezes, os pesquisadores optam por seqüenciar apenas as partes do DNA que traduzem proteínas, as quais muitas delas são consideradas as mais importantes. Estas partes do RNA mensageiro são chamadas de ESTs (*Expressed Sequence Tags*).

2.2.2. Biotecnologia

Biotecnologia pode ser definida como o conjunto de técnicas biológicas desenvolvidas através de pesquisa básica e que agora são aplicadas na pesquisa e desenvolvimento de produtos, através do uso de DNA recombinante, fusão celular e novas técnicas de bioprocessamento pela indústria [DOE05].

Em outras palavras, Biotecnologia consiste na aplicação em grande escala, ou transferência para indústria, dos avanços científicos e tecnológicos, resultantes de pesquisas em ciências biológicas [UCB04]. O próprio desdobramento da terminologia implica a biotecnologia como sendo o uso de organismos vivos (ou suas células e moléculas) para produção racionalizada de substâncias, gerando produtos comercializáveis. Diversas áreas de atuação utilizam a biotecnologia: saúde, agropecuária, bioengenharia e biodiversidade. Nem sempre o desdobramento da biotecnologia é voltado para a produção de substâncias. Também pode ser utilizado para o desenvolvimento de um método ou de uma técnica de diagnóstico ou prognóstico. Um dos empregos da biotecnologia diz respeito à modificação. Esta tecnologia implica na modificação direta do genoma do organismo alvo pela introdução intencional de fragmentos de DNA exógenos (genes exógenos) que possuem uma função conhecida. Sendo assim, por meio de engenharia genética, o gene (DNA) que contém a informação para síntese de uma determinada proteína de interesse pode ser transferido para outro organismo que, então, produzirá grandes quantidades da substância. Exemplos de substâncias assim desenvolvidas são a insulina humana e plantas resistentes a herbicidas. O desenvolvimento da

tecnologia do DNA recombinante e da biotecnologia trouxeram novas ferramentas que possibilitam flexibilidade quase total na escolha da característica a ser introduzida, permitindo a criação de combinações sem a limitação de cruzamentos. Outro uso importante da biotecnologia implica na produção de bactérias utilizadas para biodegradação de vazamentos de óleos ou lixos tóxicos.

2.2.3. Bioinformática

A Bioinformática, também denominada biologia computacional, compreende o uso de técnicas computacionais para análises de caracterização molecular, integrando algoritmos, modelos matemáticos e estatísticos visando a interpretação e análise de informações biológicas, especificamente de ácidos nucleicos e proteínas [IAL04]. Segundo [DOE05], Bioinformática é a ciência que gerencia e analisa dados biológicos utilizando técnicas de computação avançadas. As técnicas computacionais em bioinformática são oriundas de diversas áreas de pesquisa da computação como: Banco de Dados, Interface Homem Computador, Inteligência Artificial, Sistemas Distribuídos etc.

Outra definição para este termo é o uso de matemática aplicada, informática, estatística e ciência de computação para a resolução de problemas biológicos. As principais pesquisas nessa área de conhecimento são relativos aos alinhamentos de seqüências, predição das estruturas de proteínas e expressões de genes, interações entre proteínas e modelagens evolutivas. Embora a biologia computacional focalize tipicamente no desenvolvimento de algoritmo e em métodos computacionais específicos [BAXEVANIS01], uma linha que vem se tornando amplamente pesquisada é como o uso de ferramentas computacionais e matemáticas pode ajudar a extração de informação útil dos dados produzidos por técnicas biológicas que possuem elevada produção de dados [CLAVERIE03].

2.3. Conceitos Computacionais

Nesta seção são conceituados os termos computacionais necessários para o entendimento do projeto proposto, assim como é feita uma explicação a respeito do fluxo da informação através do sistema, associando os conceitos computacionais a cada uma das etapas assim que for necessário.

Observando a Figura 5, nota-se a presença de três camadas macro para o desenvolvimento do sistema BIO-TIM. A primeira camada, denominada fonte de dados, é a que contém os dados, os quais compõem as mais diversas fontes de informação, organizadas através de Sistemas Gerenciadores de Banco de Dados (SGBDs), arquivos texto (*flat files*) etc. A segunda camada, denominada Mediador, é a que contém os *softwares* que farão o papel de mediação (comunicação) entre os dados e a aplicação. Nela serão encontrados tradutores que farão a extração de dados das fontes de dados da camada inferior e o Data Warehouse (DW) que servirá como fonte de dados da aplicação que o usuário interagirá. Por fim, a terceira e última camada, denominada interface, é a que representa a interface da aplicação com o usuário.

Na seção 2.3.1 são dadas as devidas explicações sobre o que são e onde se encontram as fontes de dados que estão acima representadas; na seção 2.3.2 há a explicação a respeito do mediador (*middleware*), o qual é composto basicamente por tradutores (*wrappers*) e pelo Data Warehouse (DW). . Depois serão dados os conceitos para o entendimento do que é XML, mediador e será realizada uma explicação a respeito das dificuldades que podem ser encontradas na integração de dados a partir de fontes de dados heterogêneas. Sobre o DW serão enumerados e explicados os problemas que o cercam quando aplicado nas áreas de Biotecnologia e Bioinformática

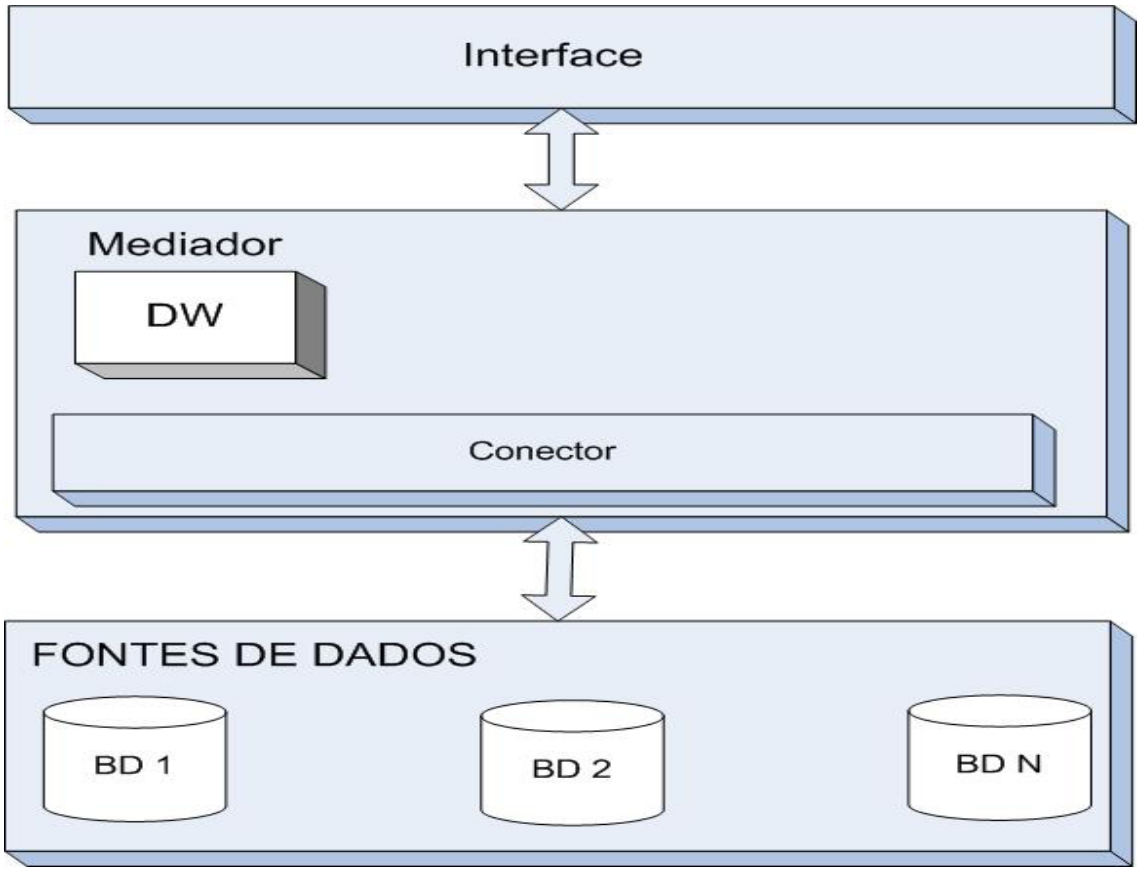

Figura 5 – Representação macro da arquitetura do sistema proposto

2.3.1. Fontes de Dados

Existem diversas fontes de dados biológicas dispersas pela Internet e fontes de armazenamento local de dados que podem, algumas vezes, serem disponibilizadas localmente através de arquivos estruturados ou SGBDs.

O ambiente utilizado para geração de informações biológicas no projeto Shrimp EST Genome da UFSCar é o Pipeline Draft, que pode ser definido como um conjunto de ferramentas de bioinformática, cujo objetivo é confrontar as seqüências com bancos de dados públicos, analisar as mesmas a fim de encontrar *repeats* (regiões de repetição), *vectors* (plasmídeos inseridos no DNA durante o processo de clonagem) e *primers* (pequenas seqüências previamente conhecidas), verificação de qualidade do seqüenciamento, etc.

Entre as informações geradas pelo Pipeline estão os arquivos produzidos através do uso de ferramentas de bioinformática, como BLAST [BLAST03], Cross-Match [PHRAP03] e Phred/Phrap [EWING05] [PHRAP03], contendo importantes informações a respeito do processamento e confronto das seqüências em relação aos bancos de dados públicos. Como exemplo de bancos de dados públicos, tem-se o GenBank [GBK03], Protein DataBank [PDB03] [BERMAN00] e Enzyme [ENZ03] [BAIROCH00]. Além das informações geradas automaticamente por ferramentas, outras podem ser inseridas pelos cientistas, como a análise realizada nas seqüências, artigos e referências publicadas por aquelas seqüências, etc.

Uma vez conceituada a camada de fontes de dados, iremos conceituar a camada intitulada Mediador.

2.3.2. Mediador

Segundo a descrição encontrada em [DMREV05], mediador é uma camada de comunicação que permite a troca de informações entre dois módulos de um sistema ou ambiente. Neste projeto, no mediador da aplicação serão encontrados os tradutores que irão extrair a informação das fontes de dados fazendo *uma* varredura por arquivos XML, depois será realizado um trabalho de integração dos dados e então os dados serão devidamente materializados no DW.

Primeiramente será definido o que é um arquivo no formato XML. Os arquivos neste formato serão lidos por tradutores que encaminharão os dados para que seja feita a carga no DW, através do processo chamado de integração de dados.

2.3.2.1. Extensible Markup Language (XML)

De acordo com a definição de [BRAY00] [ACHARD01], XML é o padrão adotado pela W3C (World Wide Web Consortium) para a representação de dados na web, uma versão enxuta do SGML (Standard Generalized Markup Language). Ela permite a definição de um

conjunto próprio de marcações (*tags*) que podem ser aplicadas em um ou vários documentos, fazendo com que seja possível a definição, validação e interpretação de dados entre aplicações entre diferentes organizações. Essas marcações identificam diferentes tipos de elementos no documento com a possibilidade de recursividade e referência [W3C05].

2.3.2.2. Integração de Dados

No contexto deste trabalho, o termo *Integração de dados* é o processo de "união" de dados provenientes de várias fontes de dados co-existentes que podem ou não ser heterogêneas para um único repositório de informações. Quando associado a um DW, essa integração de dados compreende uma fase denominada ETL (*Extract*, *Transform*, *Load*). Durante este processo de integração, várias tarefas devem ser realizadas, como:

1. Extrair informação das mais diversas fontes de dados que serão utilizadas;
2. Transformação dos dados para a sua correta utilização atendendo às necessidades do usuário;
3. Popular o DW;

Como enumerada acima, a primeira fase do processo de ETL é a extração da informação das fontes de dados, fato que aumenta a importância quando se avalia que a maior parte dos projetos de DW utilizam fontes de dados heterogêneas. Cada uma dessas fontes de dados pode conter dados organizados e formatados de diferentes formas, sendo que os tipos mais comuns são os bancos de dados relacionais e os arquivos estruturados, conhecidos também como *flat-files*.

A segunda fase do processo de integração de dados é a fase de transformação, momento no qual regras e/ou funções são aplicadas nos dados extraídos para que depois eles sejam carregados no DW. Existem casos nos quais poucas transformações são necessárias e outros que algumas combinações de transformações podem ser necessárias, por exemplo:

- Seleção de apenas algumas colunas de uma tabela;

- Tradução de dados (por exemplo, M para especificar que um animal é macho e F para especificar que é fêmea);
- Contabilizar várias linhas de dados;
- Calcular dados que são derivados de outros, etc;

A terceira fase corresponde à carga no DW.

2.3.2.3. Data Warehouse

Um DW é definido de formas diferentes de acordo com a literatura pesquisada. Segundo [INMON92], DW é uma coleção de dados orientada a assuntos, integrada, não volátil e variável no tempo, que é usada para apoio a decisões gerenciais. Em comparação a bancos de dados tradicionais, DWs geralmente contêm quantidades muito grandes de dados vindos de diversas fontes que podem incluir bancos de dados de diferentes modelos de dados e algumas vezes arquivos adquiridos de sistemas e plataformas independentes.

Antes de iniciar a explicação sobre o processo de criação de um DW é necessário conceituar modelagem multidimensional. Uma modelagem multidimensional define uma relação de "um para muitos" entre uma relação principal denominada "fato", e várias relações denominadas "dimensão". A relação "fato" estabelece uma ligação entre atributos identificadores e as informações a ele associadas. Cada associação define uma nova relação denominada "dimensão". A modelagem multidimensional do Data Warehouse definido neste trabalho será ilustrada na seção 4.3. Associado ao conceito de modelagem multidimensional e seu uso em um Data Warehouse, está o conceito de cubo. Um cubo é uma estrutura que pode ser "montada" para se utilizar adequadamente a estrutura multidimensional criada, permitindo a realização de consultas e visualização dos dados em diferentes perspectivas e níveis de detalhamento.

O processo de criação e manutenção de um DW é denominado Data Warehousing, e pode ser visualizado na Figura 6. Observando a figura percebe-se o processo de construção de

um DW, onde se encontra dados de várias fontes heterogêneas sendo integradas através de processos de ETL (descritos na seção anterior). Com o DW pronto são utilizadas ferramentas para a confecção de relatórios, levantamento de padrões a partir da mineração dos dados entre outros.

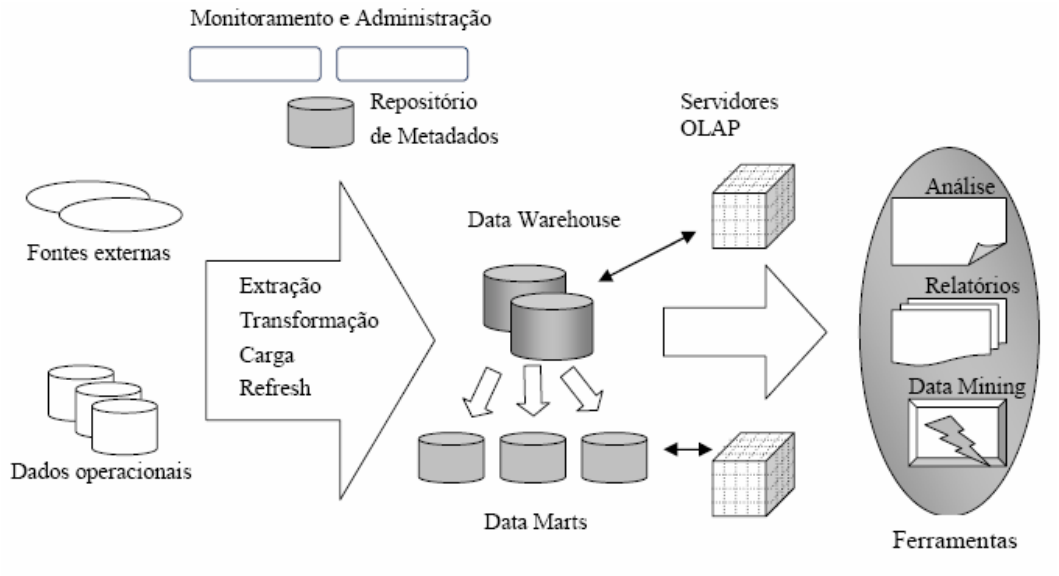

Figura 6 – Processo de Data Warehousing [VIEIRA02]

Na figura ainda pode ser visualizado os Data Marts que são subconjuntos de dados (voltados para um assunto específico dentro do contexto) de um DW e os servidores OLAP que são ferramentas que disponibilizam um conjunto de operações que podem ser executadas sobre o DW, a fim de viabilizar a extração eficaz de informações do mesmo [VIEIRA02]. Note que os servidores OLAP são os responsáveis pela criação e utilização dos cubos.

Um DW oferece ao ambiente de informação a possibilidade de acesso de dados em um único ponto através da convergência de dados a partir de fontes diferentes e a possibilidade de armazenamento de dados históricos, carregando-se fisicamente os bancos de dados do genoma dentro de um único banco. Um bom desempenho também é esperado, já que o processamento de consultas é sobre dados locais. No entanto, os dados de um DW são

materializados e não sofrem atualizações constantes, o que pode comprometer o trabalho de biólogos, geneticistas e pesquisadores afins que trabalham com dados que sofrem atualização freqüentemente [DUBITZKY01].

Um fato que deve ser destacado é que a fase de planejamento e construção de um DW é diretamente voltada para os resultados que se pretende conseguir com a integração da informação.

No mundo dos negócios (por exemplo: ações voltadas para as áreas de marketing ou *business inteligence* das empresas) esse contexto é muito bem definido, que é o de se ter dados a fim de encontrar padrões de comportamento de consumidores dadas algumas variáveis. No entanto, o estado da arte do processo de *data warehousing*, definido como um conjunto de tecnologias de software e hardware voltadas a viabilizar e otimizar a análise de dados em larga escala, gerando informações gerenciais valiosas [CHAUDHURI97], não atende a todas as características das pesquisas que envolvam Ciências Biológicas.

Para comprovar este fato, [DUBITZKY01] relata que no mundo dos negócios as consultas a serem realizadas em um ambiente de dados integrados, geralmente são bem definidas e os objetivos a serem alcançados com estas consultas também o são. No entanto, dado um contexto biológico isso nem sempre ocorre, devido ao levantamento de novas questões, teorias, proposições e formulação de hipóteses que serão testadas. Segundo levantamento realizado em seu estudo existem vários motivos que diferenciam um DW convencional (utilizado comercialmente no mundo dos negócios) de um DW idealizado para armazenar dados genômicos. Enquanto um DW voltado para a área de negócios tem como características:

- Um grande número de consultas que são previamente conhecidas;
- Processos de negócios são previamente conhecidos, permitindo que a pré-agregação dos dados seja feita de maneira mais simples;

- Dados necessários estão em um dos bancos de dados da empresa;
- Possibilidade de quebrar os dados em n-cubos de poucas e simples dimensões;
- Visão dos dados é temporal (semanal, quinzenal etc).

Os DWs voltados para a área de bioinformática devem possuir as seguintes características:

- Permitir consultas que variem freqüentemente a partir de novas visões científicas sob os dados;
- Pré-agregação dos dados não é simples, uma vez que novos conhecimentos estão sendo adquiridos;
- Dados relevantes estão dispersos em bancos de dados espalhados ao redor do mundo;
- Possui estruturas de dados complexas por natureza que são difíceis de se reduzir a poucas dimensões;
- Visão dos dados também é temporal (pode mudar a partir de descoberta de novas teorias ou suposições), no entanto é mais complexa que em um DW voltado para a área de negócios.

Observados os itens temos como grandes diferenças o volume de conhecimento adquirido no contexto biológico em relação ao contexto de negócios. Este conhecimento está na cabeça dos cientistas, publicações e relatórios. Grande parte das informações está disponível na Internet e de forma mais ou menos estruturada, no entanto o tempo necessário para materializar esta informação é muito grande. Algumas destas questões foram consideradas no desenvolvimento do projeto e estão abordadas no capítulo 4**.**

2.3.3. Interface

A camada de interface é a parte do sistema que possibilita ao usuário final interagir com o mesmo [PRINCE05]. Atualmente, muito tempo é despendido por parte dos usuários para efetuar uma navegação, uma vez que é obrigado a utilizar diversas ferramentas

disponíveis, e além do fato, de que os dados são distribuídos por diversas páginas dificultando a compreensão dos mesmos [FREIER02].

Uma das formas de minimizar estes problemas é a utilização de padrões de projeto de interfaces Web como Doormat Navigation [WELIE05], Double Tab Navigation [WELIE05], Information Seeking [WELIE05], Rule Storage [LAAKSO05] [LUUKKAINEN99] cujo objetivo é melhorar a usabilidade, deixando a navegação do usuário intuitiva, possibilitando a pessoa que estiver utilizando o sistema ter um rápido acesso a informação desejada.

- Doormat Navigation: este padrão propõe que o usuário tenha uma visão geral rápida e informativa dos assuntos que existem no sistema, possibilitando ao usuário chegar mais rápido até a informação que deseja consultar. A Figura 7 representa graficamente através da UML-based Web Engineering (UWE) [KOCH02] [KOCH03] como seria a aplicação deste padrão. Observa-se que o

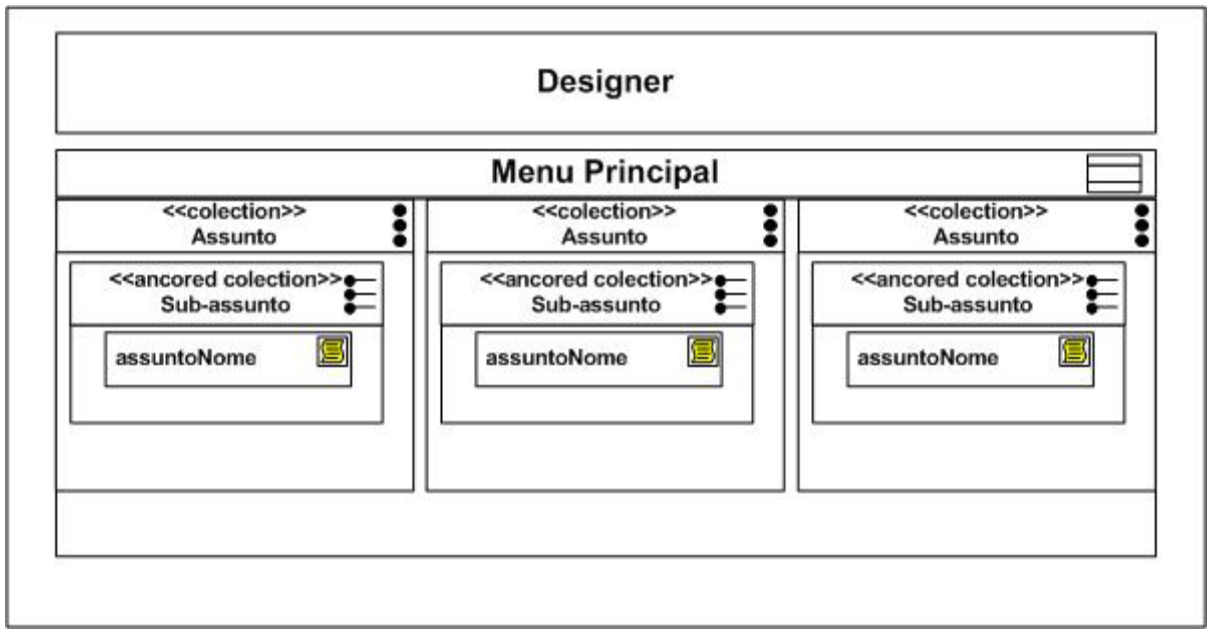

Figura 7 – Representação gráfica de uma página Web utilizando o Doormat Navigation Pattern

menu principal, já contém os principais tópicos de informação disponíveis no sistema (assuntos) e cada um destes tópicos contém a lista das possíveis informações (sub-assuntos) que podem ser encontradas.

- Double Tab Navigation: este padrão propõe a utilização de orelhas (*tabs*) que tem como finalidade facilitar a navegação entre grupos de informação. Este padrão é análogo ao padrão descrito anteriormente, no entanto os assuntos são exibidos em forma de orelhas e assim podem ser visualizados em todas as telas do sistema, possibilitando um rápido acesso às informações;
- Information Seeking: este padrão propõe que se faça um estudo do comportamento dos usuários no momento que eles fazem uma busca e a partir desse estudo encontrar qual será a melhor combinação de consultas e suas devidas regras de negócio devem ser disponibilizadas;
- Rule Storage: com o uso deste padrão o usuário armazena a regra de busca utilizada ao invés dos resultados da consulta na base de dados. Assim, quando o usuário faz um novo acesso à regra, as informações já atualizadas retornam para ele automaticamente. A Figura 8 representa em UWE como seria a página de um sistema utilizando este padrão. Observa-se que há uma lista de consultas pré-definidas, lado esquerdo da tela, e apenas com um clique o usuário obtém os resultados atualizados da busca que foi definida previamente.

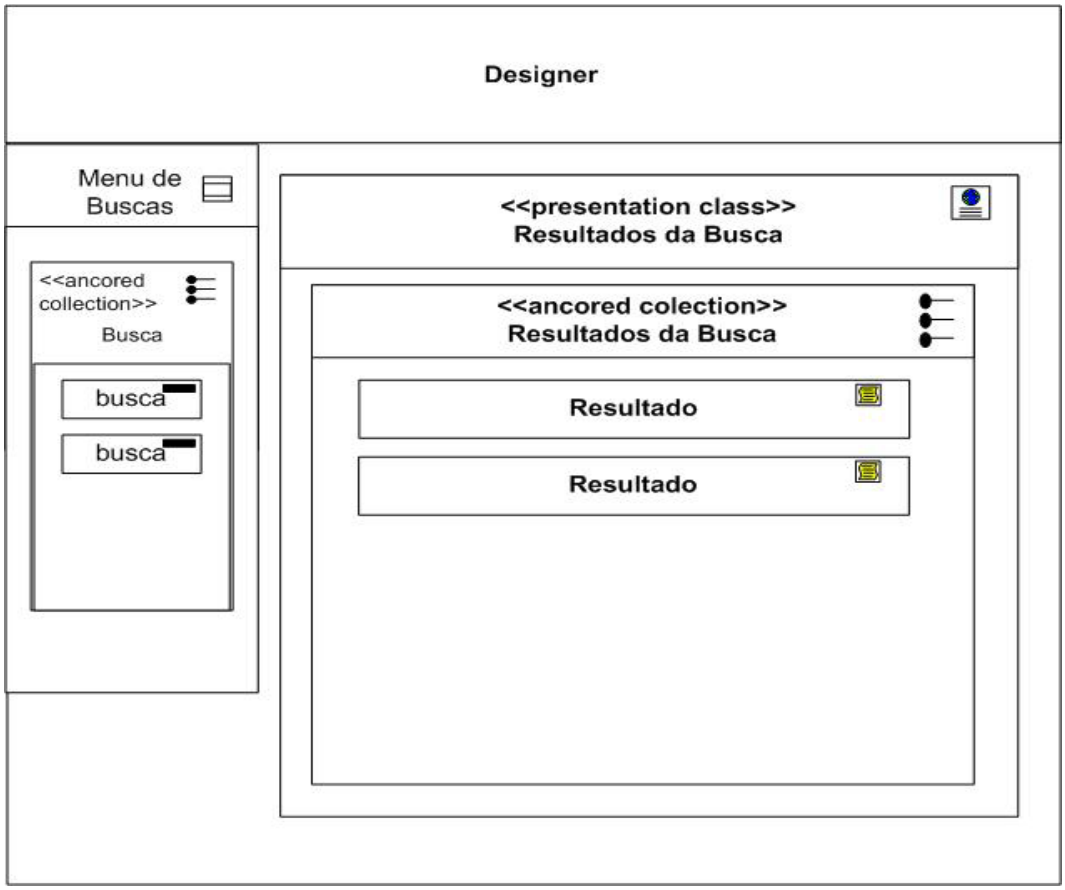

Figura 8 – Representação gráfica do padrão Rule Storage em UWE

2.4. Considerações Finais

Neste capítulo foram descritos os conceitos envolvidos neste trabalho. No próximo capítulo serão descritos alguns trabalhos relacionados a este projeto.

3. Arquiteturas de Ambientes de Bioinformática

Uma abordagem de integração que está sendo muito utilizada para o desenvolvimento de aplicações no contexto genômico é a utilização de *frameworks*. O termo *framework* é definido como uma arquitetura flexível e extensível para a construção de uma família de sistemas, aplicadas a um determinado domínio e criado com o objetivo de ser instanciado. Basicamente, um *framework* consiste de *frozen spots* e *hot spots*. Os *frozen spots* se preocupam com a arquitetura global de um sistema de software e permanecem imutáveis em qualquer instanciação do *framework*. Os *hot spots* são específicos para cada instanciação [SEIBEL00].

Um *framework* é construído para uma determinada aplicação. Assim, é função dos *hot spots* direcionar o *framework* para o domínio em questão. De uma maneira geral, a tecnologia de *frameworks* possui uma base comum, relacionada aos *frozen spots*, e uma parte flexível, que se refere aos *hot spots*. Um *framework* não pode ser caracterizado como um sistema de imediato, pois ele não é executável. Para que se construa um sistema a partir de um *framework*, o usuário terá que instanciar classes e interfaces já existentes e teoricamente testadas. Estas classes e interfaces formam componentes que acabam sendo reutilizados, acelerando assim o desenvolvimento de aplicações.

A Figura 9 demonstra basicamente como funcionam os *frameworks*. O usuário, através da aplicação de interface, submete uma ação que aciona o mediador. Em seguida, o mediador gerencia para onde deve ser encaminhada a requisição que o usuário enviou. Essa requisição chega até o tradutor que irá acessar a fonte de informação e retornar os dados para o mediador que por sua vez irá repassá-los para a aplicação que o usuário está utilizando.

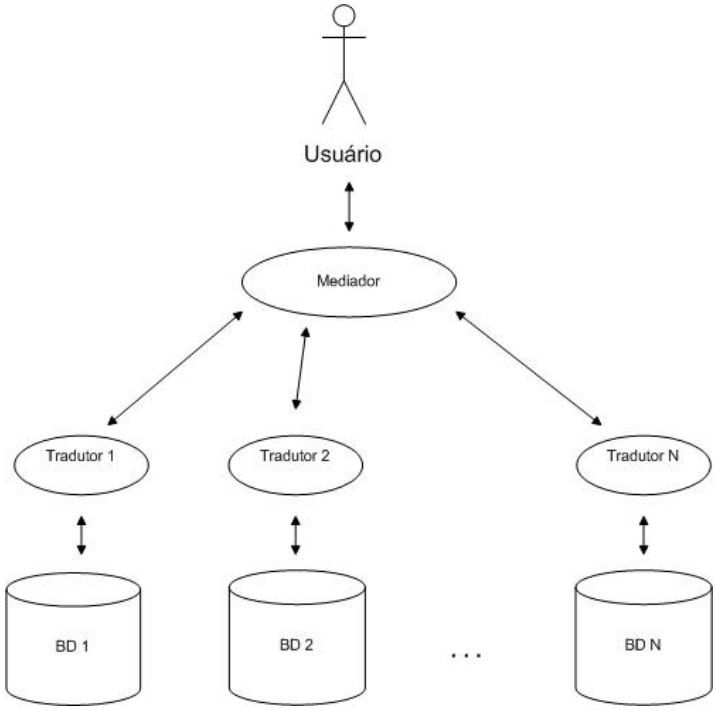

Figura 9 – Representação simples da arquitetura de um *framework*

3.1. Bio-AXS:: Uma Arquitetura para Integração de Fontes de Dados e Aplicações de Biologia Molecular

O objetivo do Bio-AXS é integrar dados de fontes de dados heterogêneas de biologia molecular. Para isso, o projeto inclui tradutores que capturam o esquema e os dados das fontes participantes da integração. Além de capturar, os tradutores traduzem o esquema e os dados para XMLSchema e XML (eXtensible Markup Language) sucessivamente. Com isso, tem-se um modelo de dados comum que facilita a comunicação com as aplicações genômicas [SEIBEL00].

A arquitetura do Bio-AXS é centrada em 4 módulos, como mostra a Figura 10:

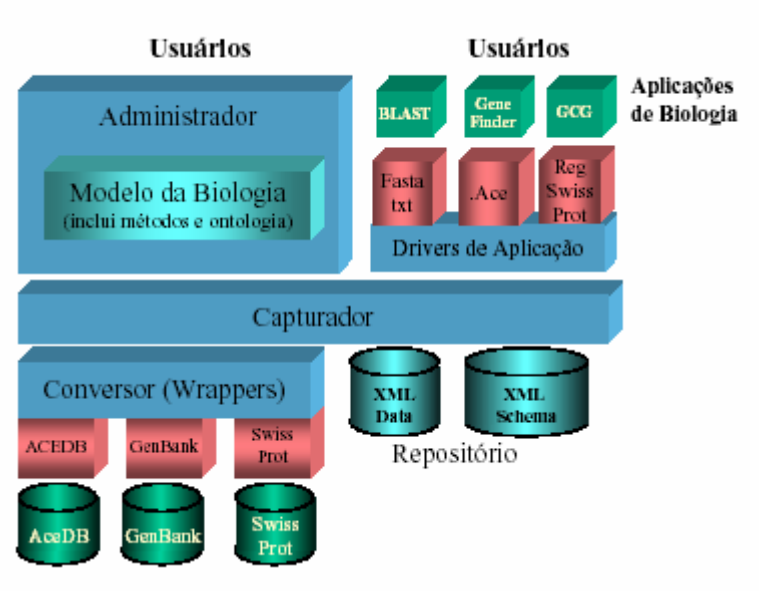

Figura 10 - Arquitetura do Framework Bio-AXS [SEIBEL00]

O módulo Administrador controla o *framework* de uma forma geral. Ele permite que o usuário gerencie os esquemas específicos, esquemas de dados que podem ser capturados, a formulação de consultas e a execução de aplicações externas sobre os dados previamente capturados. O módulo Capturador administra o repositório de dados e de esquemas da arquitetura. O outro módulo, o Conversor (tradutor), disponibiliza os dados de biologia em um formato comum para facilitar o acesso, isto é, efetua a tradução dos esquemas das fontes de dados para XML Schema e dos dados para XML. O último módulo a ser descrito, os *Drivers* de Aplicação, tem como aspecto principal as diversas aplicações que acessam os dados do *framework* e que trabalham com formato de dados distintos. Assim, o papel dos *Drivers* é traduzir os dados de XML para os formatos específicos que cada aplicação exige.

3.2. TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources

O TAMBIS [TAMB03] tem como objetivo ajudar os pesquisadores na ciência biológica fornecendo um único ponto de acesso para fontes de informação biológicas que estão espalhadas pelo mundo. O ponto de acesso é uma única interface (através da *World*

Wide Web) que faz uma "simulação", isto é, através da navegação pelo sistema tem-se a impressão que está sendo feito o acesso a uma única fonte de dados, quando, na realidade, estão sendo acessadas várias fontes de informação. O TAMBIS é responsável por encontrar as fontes de informações necessárias para as consultas dos usuários e "traduzir" a consulta do usuário para cada fonte, retornando os resultados de uma maneira consistente que inclua detalhes da fonte de informação. De maneira geral a arquitetura do TAMBIS é análoga à do Bio-AXS

3.3. SRS: Sequence Retrieval System

É um sistema que possui uma linguagem própria para que sejam feitas consultas em diversos bancos de dados. Ele possui alguns utilitários que podem ser utilizados conjuntamente. Ele varre arquivos e bancos de dados que contêm textos estruturados com nomes de campos. Então são criados e armazenados índices para cada campo e então estes índices são utilizados em tempo de consulta para retornar as informações relevantes [SPROT03].

Apesar de extensos índices serem armazenados localmente para serem utilizados no processo de consultas, o SRS não pode ser considerado um banco de dados, pois ele não modifica e não armazena dados externos localmente, ao contrário dos demais sistemas apresentados.

3.4. K2/BioKleisli

É um banco de dados distribuído, na qual as partições são independentes umas das outras. Ele pode ser basicamente definido, segundo [DAVIDSON01], como um sistema de transformação e integração de dados que pode ser utilizado por qualquer aplicação cujos dados obedeçam a uma forte estrutura de tipos. Ele estende o modelo de banco de dados relacional puro, onde o SQL é usado para a realização de consultas, para um modelo de dados

de objetos complexos suportado por uma linguagem chamada CPL. Suas características fazem com que o usuário tenha a impressão de estar usando uma única fonte de dados, através de transformações para consultas em bancos de dados autônomos e dispersos geograficamente. O modo de se fazer esse acesso tem o funcionamento análogo ao TAMBIS e BIO-AXS.

3.5. DiscoveryLink

Conforme descrito em [HAAS00] e [HAAS01], é um sistema de integração de bioinformática orientado a conectores. Assim, servem como os demais *frameworks* como um intermediário para as mais diversas aplicações que necessitam consultar dados dos mais diversos bancos de dados biológicos. As aplicações se conectam no DiscoveryLink e fazem suas requisições em SQL para um esquema global sem a ciência sobre quais fontes de dados estão por trás desse esquema. Isto quer dizer que existem mediadores entre aplicações e conectores (análogo aos tradutores) de acesso aos bancos, que deixam transparente o acesso as mais diversas fontes para o usuário ou desenvolvedor de aplicações que o utiliza.

3.6. Considerações Finais

Neste capítulo foram expostas técnicas que atualmente são utilizadas na integração de dados genômicos, que é o uso de *frameworks* integradores de informação Pode ser constatado que eles funcionam de maneira análoga, ou seja, possuem o mesmo princípio de ir até a fonte de dados, transformar estes mesmos dados e mostrar para o usuário.

No próximo capítulo será feita a apresentação do ambiente Bio-TIM.

4. Ambiente Bio-TIM

4.1. Considerações Iniciais

Neste capítulo será apresentado o ambiente Bio-TIM (sigla para Bioinformatics – Transparent Information Management) proposto neste projeto, o qual consiste: na especificação dos requisitos no contexto genômico; modelagem e implementação de um DW para armazenamento dos dados produzidos, tanto pelos biólogos quanto pelas ferramentas computacionais existentes e que necessitam de um armazenamento para posterior consulta; e a implementação das demais unidades da arquitetura para o suporte do ambiente.

4.2. Visão Geral da Arquitetura

O estágio atual das pesquisas genômicas, segundo [REINACH05], fornece aos cientistas toda a seqüência de bases que formam um determinado genoma. Entretanto, as ferramentas computacionais disponíveis para sua interpretação ainda não são suficientes. Assim, toda ferramenta que seja capaz de auxiliar a descoberta de conhecimento é de grande valia para a comunidade, cujo objetivo é "decifrar" o código genético. Como citado anteriormente, de acordo com [FREIER02], as ferramentas de bioinformática são muitas e cada uma possui sua funcionalidade particular de busca nas bases de dados, o que dificulta o trabalho dos pesquisadores. O projeto aqui desenvolvido foi proposto com a finalidade de definir e criar um ambiente (Bio-TIM) direcionado a pesquisadores que atuam na área genômica (neste projeto, para pesquisadores ligados ao Shrimp EST Genome Project). O ambiente compreende a definição e criação de um DW no contexto genômico, pois com ele o pesquisador poderá acompanhar a evolução do conhecimento no decorrer do tempo e através de uma única fonte de informação. Assim ele poderá interpretar os dados de maneira mais abrangente e entender quais as mudanças de cenário, devido à descoberta de novos conhecimentos científicos, os quais estão trazendo mudanças nas interpretações dos dados.

A Figura 11 representa, por exemplo, visualmente, como seria o armazenamento de relatórios de uma determinada seqüência ao longo do tempo, com o uso de um DW. Observa-se, neste caso, que o histórico dos relatórios criados podem variar de acordo com as fontes de dados consultadas para a sua geração, uma vez que estas fontes de dados são atualizadas constantemente. É possível observar, através de consultas no DW, que caso o *software* de geração de relatórios automáticos como o BLAST mude de versão, se a saída gerada será a mesma ou não, podendo assim ser realizada uma análise sobre o uso de utilitários na comparação de seqüências. Isso permite ao pesquisador a possibilidade de, no futuro, poder comparar com novas interpretações inseridas o porquê daquelas conclusões alcançadas no passado, ou ainda saber quais as condições de conhecimento da Biologia que ele tinha para chegar à aquelas conclusões. Caso novos conhecimentos biológicos sejam adquiridos o pesquisador poderá, por exemplo, voltar ao sistema, e então atualizar comentários inseridos junto com seqüências etc.

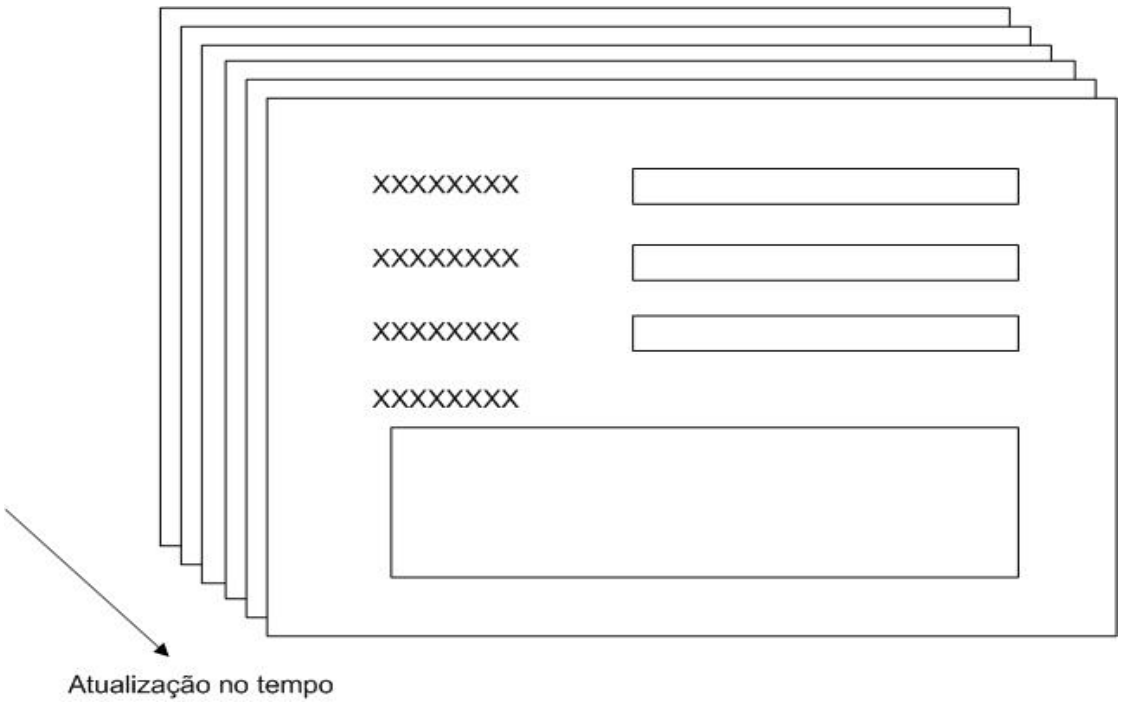

Figura 11 – Representação da possibilidade de acompanhamento da evolução do conhecimento com o uso de um DW

Observando a Figura 12, estão destacadas as três camadas que compõem a arquitetura do sistema. Há uma camada que representa a interface da arquitetura (não implementada neste projeto); outra representa o mediador e todos os elementos que o compõe; e a terceira camada representa as diferentes fontes de dados existentes.

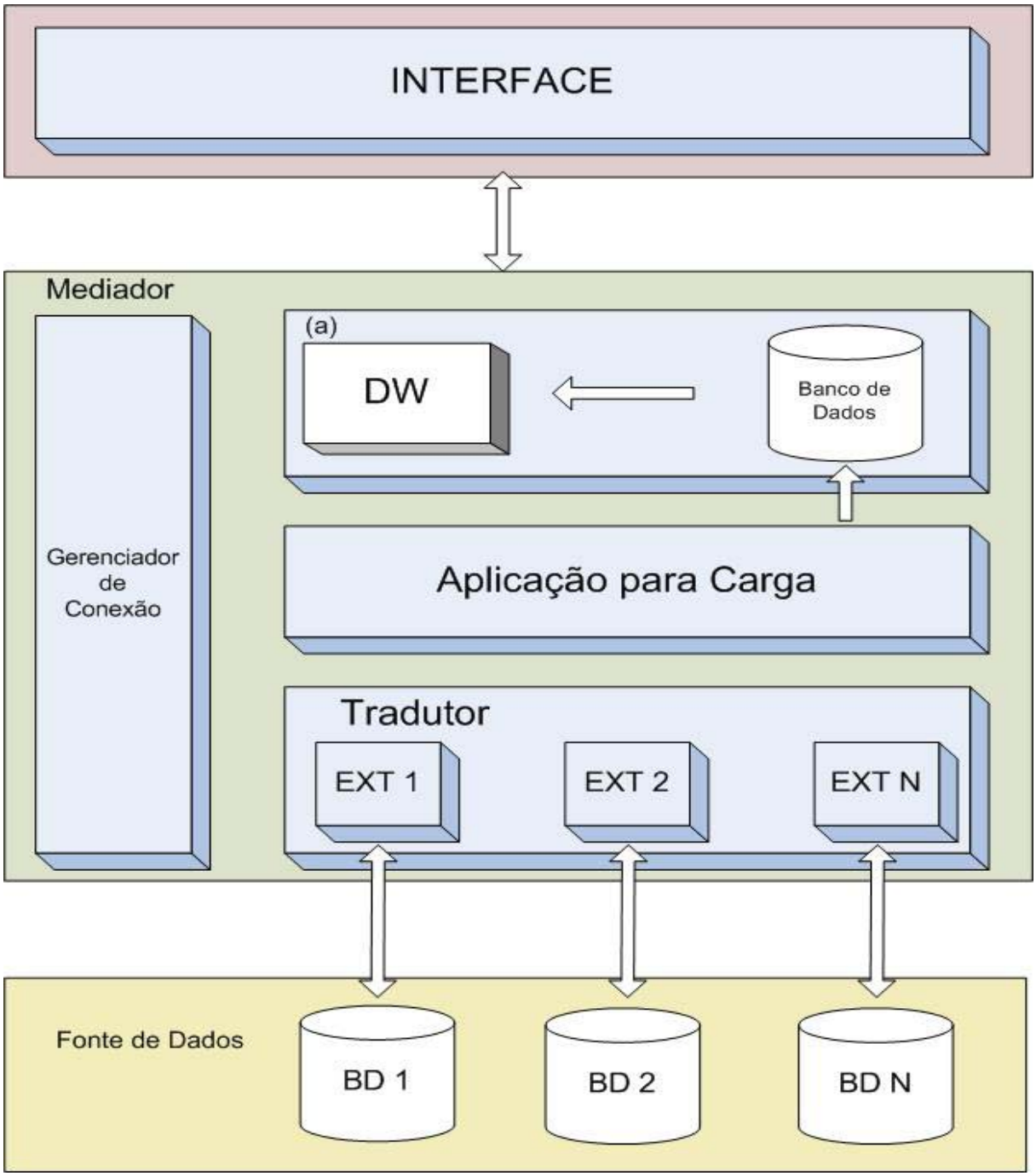

Figura 12 – Arquitetura do modelo proposto para o Bio-TIM

A camada que foi desenvolvida neste projeto é a do mediador, observado na Figura 12. Nela está implementado o gerenciador de conexão (código pode ser visto no Apêndice B) que possibilita ao desenvolvedor de aplicações o acesso ao banco de dados utilizado de forma transparente, uma vez que tenha disponível o *driver* JDBC. Ele pode através de um arquivo de configurações, quantificar o número de conexões que irão fazer parte do *pool* de conexões e o *timeout* dessas conexões.

No quadro identificado com (a), na Figura 12, há um banco de dados com modelagem análoga ao do DW. Este banco de dados é o destino de toda a informação antes de sua materialização no DW. Em tempos pré-determidados através de agendamento de métodos do DW, ele é rematerializado com os dados presentes no banco; após a materialização, os dados são removidos do banco. A modelagem multidimensional do DW será descrita na seção 4.3.

O tradutor é composto por uma classe genérica que contém métodos de navegação em árvores XML. Essa classe é estendida para outras classes (Ext 1, Ext 2 etc), pois cada árvore XML contém suas particularidades (elementos próprios). Maiores detalhes sobre o tradutor serão descritos na seção 4.4.

O fluxo da informação através do Mediador, representado através de um diagrama na Figura 13, é feito da seguinte forma:

- É feita um requisição de um serviço através de uma classe que pertence ao pacote Servlet.
- Esse serviço requisitado (ver pacote Service), consulta o tradutor (pacote Wrapper) para saber qual das suas extensões deve ser chamada para a manipulação dos dados que estão sendo enviados.
- Após isso, é instanciada a classe tradutora específica para os dados que estão sendo submetidos e então os objetos serão instanciados (através de classes do

pacote VO) e posteriormente serão inseridos no banco de dados, com o uso de métodos do pacote DAO.

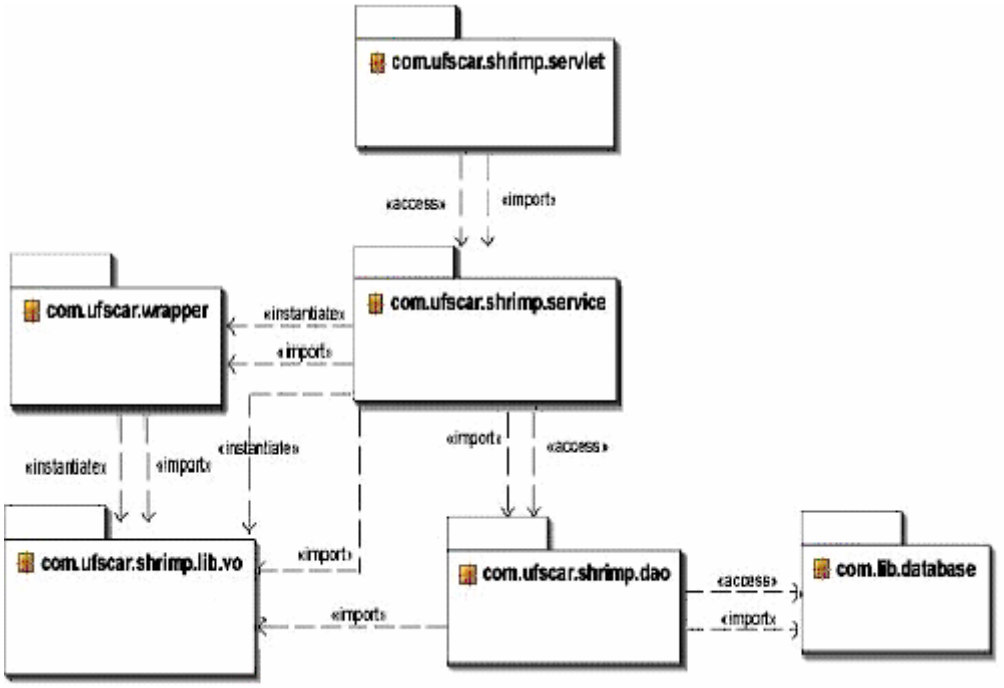

Figura 13 – Arquitetura implementada para o Mediador

4.3. Data Warehouse implementado

O DW proposto contempla o armazenamento dos dados considerados relevantes, após levantamento realizado em conjunto com os biólogos envolvidos no projeto Shrimp EST Genome Project. Muitas vezes, para chegar a um determinado resultado, é necessária a realização de vários passos durante o processo de busca. Alguns desses passos, segundo os pesquisadores, são realizados de forma manual**,** e o uso de uma fonte de informações integrada com ampla possibilidade de consultas flexíveis tende a acabar com este tipo de situação.

De maneira geral, todas as buscas que necessitam de sobreposição de dados têm causado dificuldades aos biólogos. Por exemplo, para a realização de uma pesquisa que retorne como resultado a região promotora de uma série de genes, é necessário inicialmente

comparar seqüências provenientes de RNA mensageiro e de DNA, determinar o início de transcrição, encontrar as seqüências sinalizadoras do promotor (Caat-box e tataa-box), verificar a presença de regiões de sinalização ao ribossomo na 5' UTR, verificar a existência de regiões com motivos de ligação para fatores de transcrição, etc. Atualmente, esse tipo de consulta, na maioria dos sistemas é feita de forma manual, passo a passo, e um dos motivos é a não existência de uma fonte de dados consolidada dentro dos laboratórios. O primeiro passo para que consultas complexas como a descrita acima seja possível de ser realizada de maneira intuitiva para o pesquisador, é iniciar a convergência das informações para um único ponto. Assim sendo, como primeiro passo para a convergência de informações, foi proposto o Bio-TIM.

Na Figura 13 está ilustrado o modelo multidimensional ao DW proposto. Para efeito de legibilidade, nesta primeira figura foram suprimidos os atributos. O dicionário de dados pode ser visualizado no Apêndice A.

Agora serão apresentadas as descrições de cada uma das tabelas que compõe o DW. Deve ser destacado que a partir que novas necessidades de pesquisa que venham a ser levantadas junto aos pesquisadores, o esquema do DW do Bio-TIM deve evoluir. Atualmente, como o projeto Shrimp EST Genome Project encontra-se na fase de seqüenciamento (pesquisa básica cuja finalidade é a obtenção de dados), a modelagem do DW teve como objetivo primário atender aos requisitos levantados para esta fase do projeto.

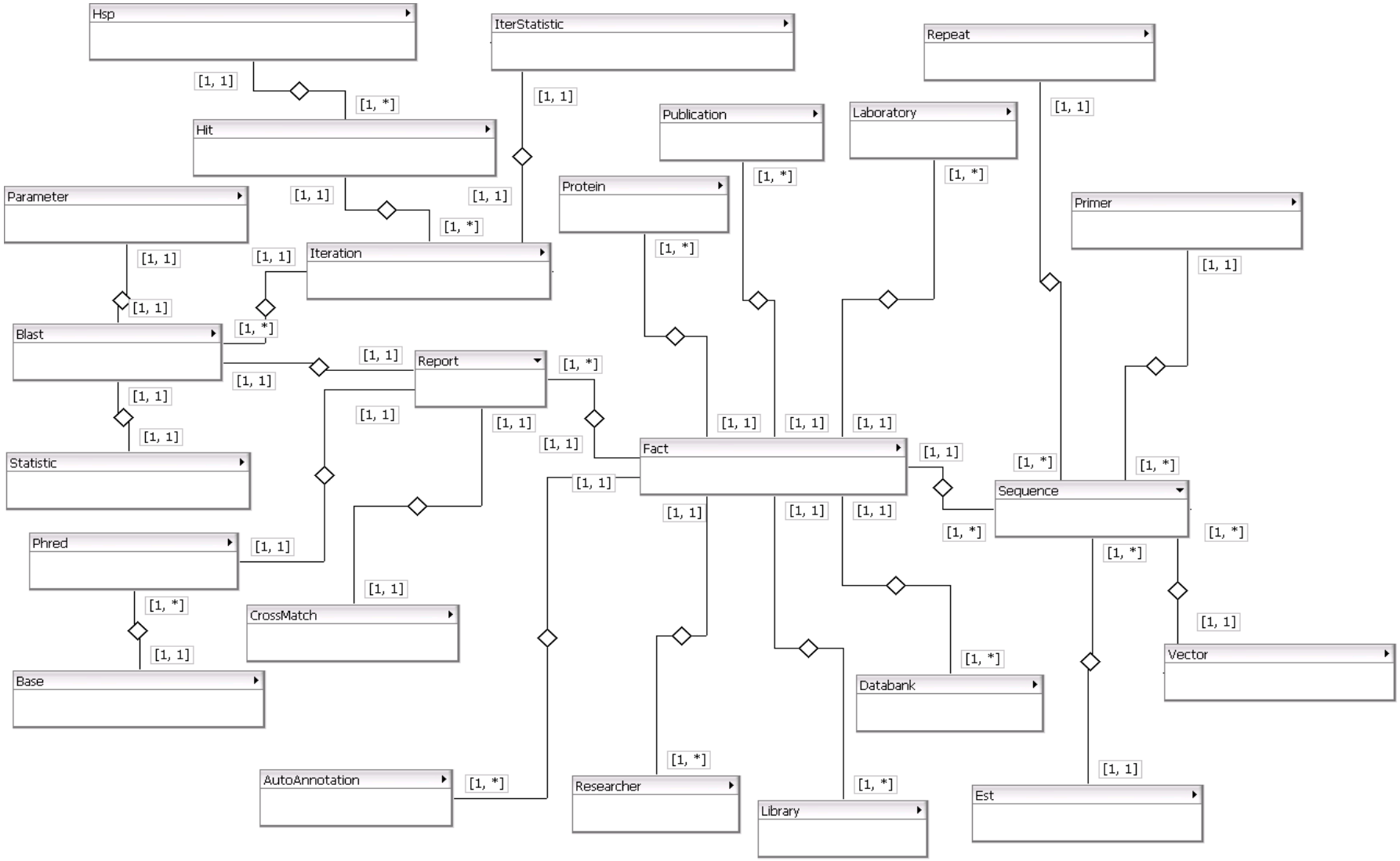

Figura 14 - Esquema Global do DW proposto para o armazenamento dos dados, sem os atributos para efeito de clareza

A tabela fato denominada Fact, como pode ser observado na Figura 15, armazena dados como:

- Identificador do fato;

- Data da submissão do fato;

- Comentário sobre o fato inserido.

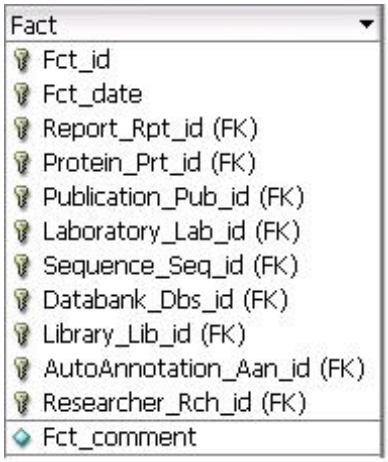

Figura 15 – Representação física de como são armazenados os dados na tabela Fact

Discorrendo sobre cada uma das dimensões do DW, tem-se:

- Reports: dimensão que representa os mais diversos relatórios que um seqüenciamento pode ter. Cada uma das ferramentas de bioinformática gera relatórios estruturados que são armazenados. Atualmente o Pipeline instalado nos laboratórios da UFSCar é composto por três utilitários de biologia computacional. Estas ferramentas geram relatórios que estão sendo representados no modelo do DW, conforme pode ser observado na Figura 16:

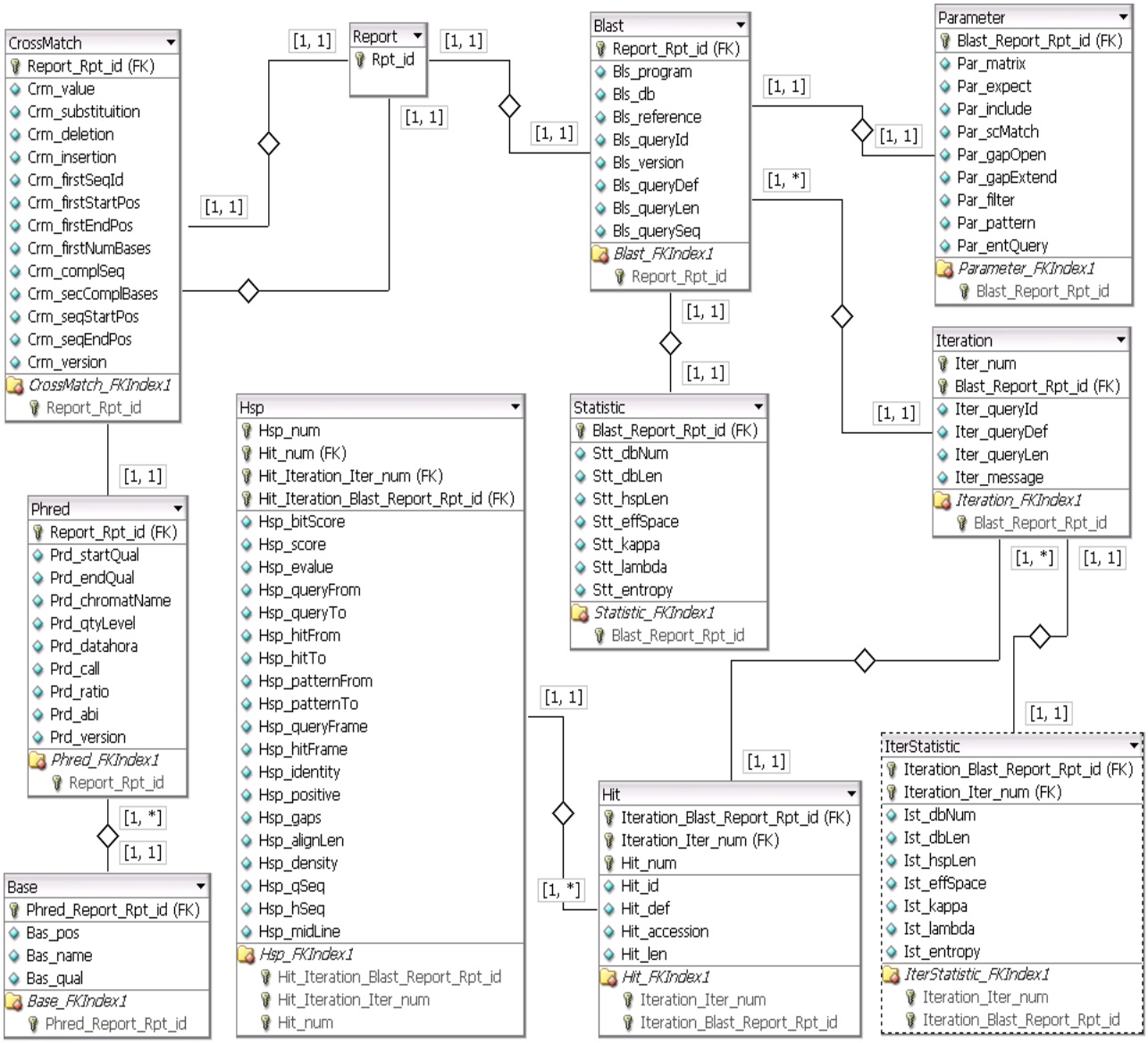

Figura 16 – Mapeamento físico de como são armazenados os relatórios produzidos pelo Pipeline no DW

- Phred: As informações armazenadas nesta tabela contêm o resultado gerado pelo Phred relacionado à qualidade das bases presentes na seqüência. Armazena também a posição inicial e final da seqüência que apresenta a qualidade mínima desejada entre outras informações.
- Blast: As informações armazenadas são: nome e versão do utilitário BLAST utilizado, qual banco de dados genômico foi utilizado para confrontar a seqüência, descrição da seqüência submetida, os parâmetros

utilizados pelo usuário quando utilizou o BLAST para confrontar a seqüência entre outras informações. Possui uma tabela com estatísticas gerais, uma de iterações, estatísticas por iterações, *hits, hsp* encontrados pelo utilitário.

- Cross-Match: As informações armazenadas nesta tabela correspondem ao resultado gerado pela ferramenta como a porcentagem de substituições, de inserções e supressões da primeira seqüência em relação à segunda utilizada na comparação, a posição de início e fim da região de confronto da primeira e segunda seqüência, número de bases da primeira seqüência após o final da região de confronto e versão do software utilizado.

- Publication: armazena informações relativas a publicações, onde podem ser encontrados dados a respeito de uma dada seqüência, gene, proteínas etc. São armazenadas informações como nome da referência, tipo, autores, edição, ano de publicação, páginas local de apresentação e editora (observar Figura 17).

Figura 17 – Representação física de como são armazenados os dados relativos à publicações no DW

- Sequence: armazena as informações sobre a seqüência de nucleotídeos que é objeto de estudo, como: o nome da seqüência (este nome é dado através de uma convenção pré-

estabelecida no momento da instalação do Pipeline), tamanho da seqüência, orientação, a seqüência propriamente dita, o acrônimo do tecido e sua orientação. Nesta dimensão o usuário também poderá verificar a existência de *primers*, vetores e repetições na seqüência. A Figura 18 representa esta dimensão.

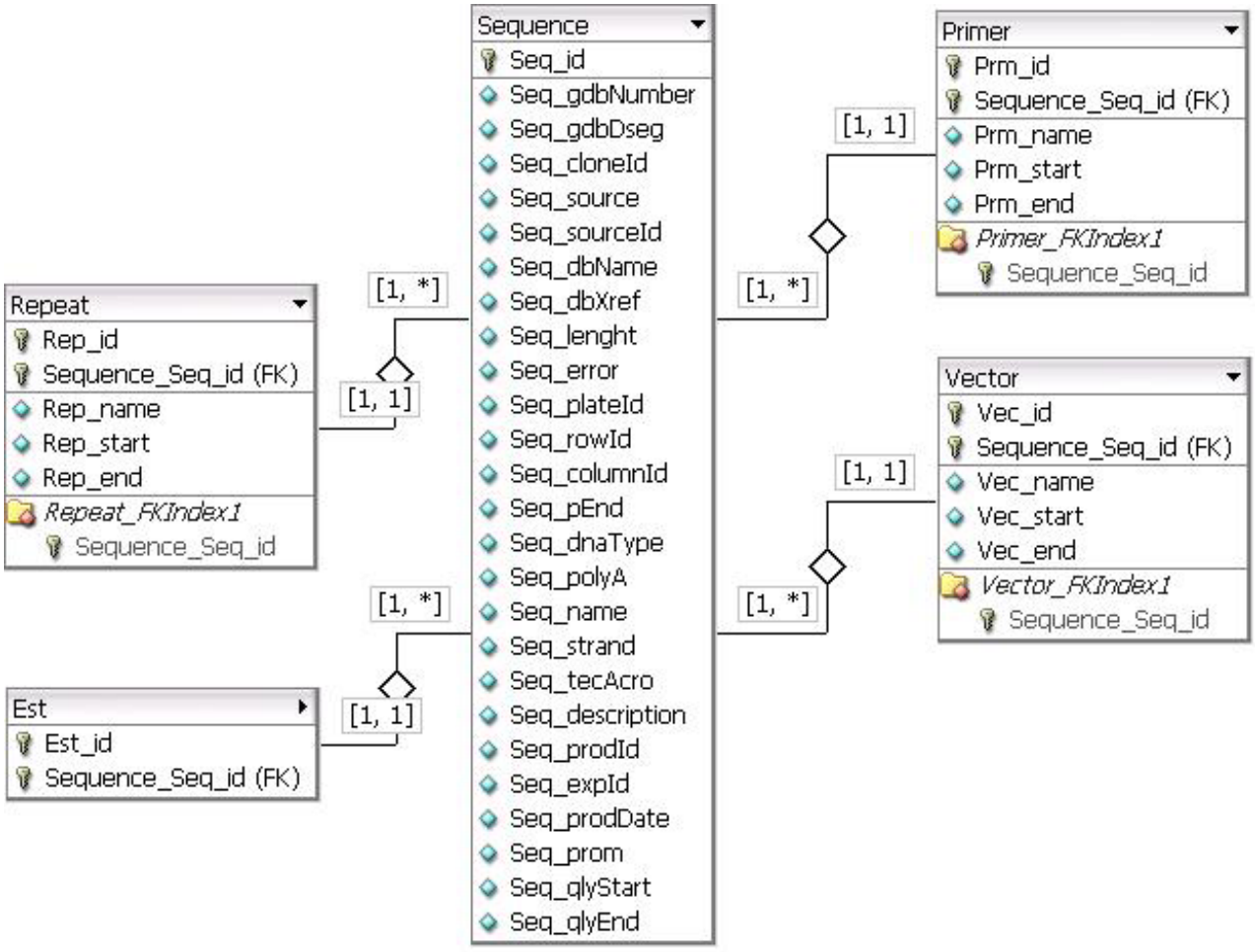

Figura 18 – Representação física de como são armazenados seqüências no DW

- Repeat: armazena o nome e posições de início e fim de pequenas subseqüências simples que são encontradas durante o processo
- Primer: armazena o nome e posições de início e fim de pequenas seqüências previamente conhecidas

- Vector: armazena nome, posição inicial e final dos plasmídeos que são o material genético inserido no DNA durante o processo de clonagem.

- Databank: armazena (Figura 19) as entradas da seqüência relacionada nos bancos de dados públicos, como o GenBank, Unigene, LocusLink Protein DataBank e Enzyme. Além de um campo para anotações gerais sobre esta entrada.

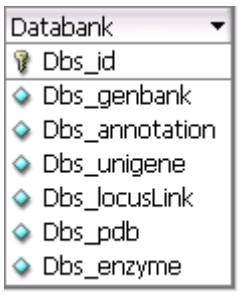

Figura 19 – Representação física de como são armazenadas os dados relativos a bancos de dados públicos no DW

- Laboratory: armazena informações como nome do laboratório, endereço e nome do responsável no laboratório pelo projeto (Figura 20).

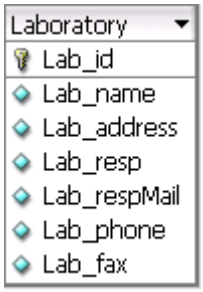

Figura 20 – Representação física de como são armazenados os dados relativos aos laboratórios no DW

- Researcher: armazena nome do pesquisador responsável pela submissão, e-mail, telefone e fax (Figura 21).

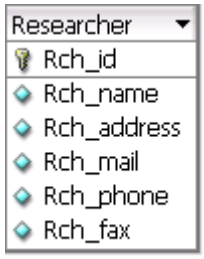

Figura 21 – Representação física de como são armazenadas os dados relativos aos pesquisadores no DW

- AutoAnnotation: armazena a posição inicial e final do alinhamento na seqüência, porcentagem de identidade, e-value e score do alinhamento, nucleotídeos da consulta no alinhamento, string que representa a homologia no alinhamento, identificador do banco de seqüências (Figura 22).

Figura 22 - Representação física de como são armazenadas anotações geradas automaticamente pelo Pipeline e armazenadas no DW

- Protein: armazena dados como nome, lista de aminoácidos que a compõe, local de produção, como órgão, célula etc e função em cada etapa do desenvolvimento do ser em estudo (Figura 23).

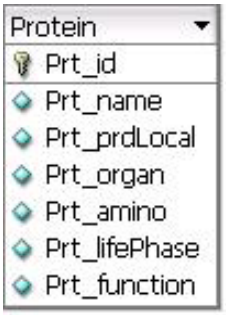

Figura 23 – Representação física de como são armazenadas as proteínas no DW

- Library: armazena dados como nome científico do organismo que a biblioteca foi preparada, local de onde a seqüência foi obtida, organismo, nome do órgão, nome do tecido, tipo de célula, estágio de desenvolvimento, vetor, enzima de restrição (Figura 24).

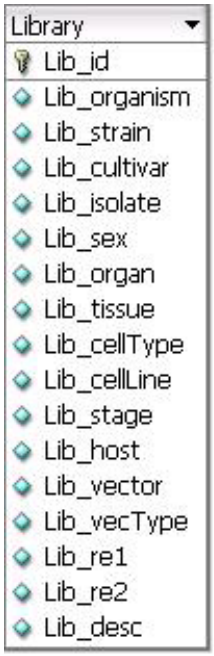

Figura 24 – Representação física de como são armazenadas as Libraries no DW

Algumas consultas para demonstrar o uso do DW podem ser visualizadas no capítulo 5.

Nesta seção foi descrita a modelagem multidimensional do data warehouse implementado, juntamente com uma descrição parcial de seus elementos.

4.4. Tradutores

Foram construídos tradutores para a leitura de arquivos XML válidos, cuja função é servir de ponte entre os dados distribuídos em arquivos XML e o sistema de armazenamento de informação, o Bio-TIM. Um exemplo de arquivo XML a ser traduzido pode ser observado na Tabela 1, que contém a saída de dados realizada pelo Blast, através do utilitário blastx, depois que uma seqüência é submetida ao Pipeline.

Tabela 1 – Saída XML realizada pelo Blast após a submissão de uma seqüência no Pipeline

```
<BlastOutput>
<BlastOutput_program>blastx</BlastOutput_program>
  <BlastOutput_version>blastx 2.2.1 [Aug-1-2001]</BlastOutput_version>
  <BlastOutput_reference>~Reference: Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schaffer,
~Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997), ~"Gapped BLAST and PSI-
BLAST: a new generation of protein database search~programs",  Nucleic Acids Res. 25:3389-
3402.</BlastOutput_reference>
  <BlastOutput_db>nr</BlastOutput_db>
  <BlastOutput_query-ID>lcl|1</BlastOutput_query-ID>
  <BlastOutput_query-def>pA262 </BlastOutput_query-def>
  <BlastOutput_query-len>1091</BlastOutput_query-len>
  <BlastOutput_param>
    <Parameters>
      <Parameters_matrix>BLOSUM62</Parameters_matrix>
      <Parameters_expect>10</Parameters_expect>
      <Parameters_include>0</Parameters_include>
      <Parameters_sc-match>0</Parameters_sc-match>
      <Parameters_sc-mismatch>0</Parameters_sc-mismatch>
      <Parameters_gap-open>11</Parameters_gap-open>
      <Parameters_gap-extend>1</Parameters_gap-extend>
      <Parameters_filter>L;</Parameters_filter>
    </Parameters>
  </BlastOutput_param>
  <BlastOutput_iterations>
    <Iteration>
      <Iteration_iter-num>1</Iteration_iter-num>
      <Iteration_hits>
        <Hit>
          <Hit_num>1</Hit_num>
          <Hit_id>gi|15223395|ref|NP_171648.1|</Hit_id>
          <Hit_def>(NC_003070) hypothetical protein [Arabidopsis thaliana]
>gi|9665150|gb|AAF97334.1|AC023628_15 (AC023628) Hypothetical protein [Arabidopsis
thaliana]</Hit_def>
          <Hit_accession>NP_171648</Hit_accession>
```

```
        <Hit_len>361</Hit_len>
        <Hit_hsps>
         <Hsp>
          <Hsp_num>1</Hsp_num>
          <Hsp_bit-score>34.2686</Hsp_bit-score>
          <Hsp_score>77</Hsp_score>
          <Hsp_evalue>12.12243</Hsp_evalue>
          <Hsp_query-from>891</Hsp_query-from>
          <Hsp_query-to>1016</Hsp_query-to>
          <Hsp_hit-from>24</Hsp_hit-from>
          <Hsp_hit-to>65</Hsp_hit-to>
          <Hsp_pattern-from>0</Hsp_pattern-from>
          <Hsp_pattern-to>0</Hsp_pattern-to>
          <Hsp_query-frame>3</Hsp_query-frame>
          <Hsp_hit-frame>0</Hsp_hit-frame>
          <Hsp_identity>18</Hsp_identity>
          <Hsp_positive>22</Hsp_positive>
          <Hsp_gaps>0</Hsp_gaps>
          <Hsp_align-len>42</Hsp_align-len>
          <Hsp_density>0</Hsp_density>
          <Hsp_qseq>IFCHFSLRHLLVLKFLDSGCKLLD*ACWIVLSFRMDGVEQLI</Hsp_qseq>
          <Hsp_hseq>LFCHITYRHLLVLSSDDNGCVILKKVITIADDFLKDEFLDLI</Hsp_hseq>
          <Hsp_midline>+FCH + RHLLVL   D+GC +L    I  F  D    LI</Hsp_midline>
         </Hsp>
          ….
        </Hit_hsps>
       </Hit>
      <Iteration_stat>
       <Statistics>
        <Statistics_db-num>837281</Statistics_db-num>
        <Statistics_db-len>264667755</Statistics_db-len>
        <Statistics_hsp-len>0</Statistics_hsp-len>
        <Statistics_eff-space>0</Statistics_eff-space>
        <Statistics_kappa>0.041</Statistics_kappa>
        <Statistics_lambda>0.267</Statistics_lambda>
        <Statistics_entropy>4.94066e-324</Statistics_entropy>
       </Statistics>
      </Iteration_stat>
    </Iteration>
  </BlastOutput_iterations>
</BlastOutput>
```

Através da Figura 25, pode ser observado os métodos que compõe a classe responsável pela tradução dos dados oriundos de fontes de dados externas. Esta classe, denominada tradutor é composta por 7 métodos, no entanto destacam-se 3 métodos principais, o getData(), o readXMLFile() e o stepThroughAll(). Os outros métodos devem ser definidos nas classes que o estendem, pois variam de acordo com os dados que cada documento possui e como é feito o seu armazenamento nas tabelas do banco.

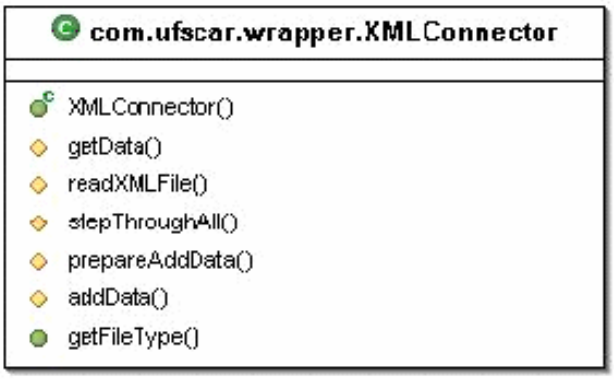

Figura 25 – Diagrama da classe XMLConnector

O primeiro método getData() recebe uma string para conexão com o documento XML. Este documento deve ser disponibilizado através da Internet. O segundo método readXMLFile() faz o *parser* no arquivo XML, verificando se ele é válido ou não. O terceiro método, stepThroughtAll(), percorre a árvore XML presente no arquivo e adiciona os atributos no objeto.

Tabela 2 – Trecho de código comentado do tradutor XML

```
public class XMLConnector {
        ....
          protected void getData( String url) {

                    InputStream inputFromServlet = null;

                    try {
                              String servletGET = url;

                              // Conecta no servlet que irá prover o XML
                              URL addressDBservlet = new URL( servletGET );
                              URLConnection servletConnection = addressDBservlet.openConnection();

                              //Set de parâmetros que informa que a conexão faz input e output
                               servletConnection.setDoInput(true);
                              servletConnection.setDoOutput(true);

                              //Garantia para não utilização de cache
                              servletConnection.setUseCaches (false);
                              servletConnection.setDefaultUseCaches (false);

                              // Leitura dos dados vindos do Servlet.
                              //
                              // O servlet retorna os dados em um stream
                              //
                              inputFromServlet = servletConnection.getInputStream();

                              readXMLFile (inputFromServlet);
```

```
            }
            catch (Exception e) {
                    System.out.print(e.getMessage());
            }
    }

    protected void readXMLFile (InputStream theInputFromServlet) {

            Document doc = null;

            try {
                    // Lendo o streamind de dados que está chegando
                    //
                    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
                    DocumentBuilder builder = factory.newDocumentBuilder();
                    doc = builder.parse(theInputFromServlet);

                    //Pegando o elemento root
                    //Após isso é feita uma navegação por todos elementos
                    //através do método stepThroughAll
                    //
                    Element root = doc.getDocumentElement();
                    stepThroughAll(root);

                    //Fechando a conexão com o Stream
                    theInputFromServlet.close();
            }
            catch (IOException e) {

                    System.out.print(e.getMessage());
            }
            catch (Exception e) {

                    System.out.print("Problem parsing the file: "+e.toString()+e.getMessage());
            }
    }

    /**
     * Método utilizado para navegação dentro dos nós do XML
     * Observar que ele é recursivo, assim navega por todas as folhas
     * Irá retornar um vetor com todos os dados necessários
     *
     * @return void
     */
    protected void stepThroughAll (Node start)   {

            if ( start.getNodeValue() != null ) {
                    if ( !(start.getNodeValue().trim().equals("\n") ||
                                    start.getNodeValue().trim().equals("")) ) {
                            prepareAddData ( start );
                    }
            }

            if (start.getNodeType() == start.ELEMENT_NODE) {

                    NamedNodeMap startAttr = start.getAttributes();

                    for (int i = 0;        i < startAttr.getLength(); i++) {
                            Node attr = startAttr.item(i);
                            System.out.println(" Attribute: "+ attr.getNodeName() + " = " +
```

```
attr.getNodeValue());
                        }
                }

                for (Node child = start.getFirstChild(); child != null; child = child.getNextSibling()) {
                        stepThroughAll(child);
                }
        }

        /**
         * Método que deve ser estendido para as demais classes, onde
         * será trabalhado o armazenamento dos vetores de dados no banco de dados
         * @param data
         */

        protected void prepareAddData ( Node start ) {
                addData (start.getParentNode());
                System.out.println(start.getNodeValue());
        }

        protected void addData ( Node parent ){
                if ( parent.getNodeName() != "#document" ) {
                        addData (parent.getParentNode());
                        System.out.print(parent.getNodeName() +':');
                }
        }

        /**
         * Método que retorna ao service que tipo de arquivo irá ser inserido
         *
         */
         public String getFileType (String url){

                        String servletGET = url;
                        Document doc = null;

                        try {
                                URL addressDBservlet = new URL( servletGET );
                                URLConnection servletConnection =
addressDBservlet.openConnection();

                                servletConnection.setDoInput(true);
                                servletConnection.setDoOutput(true);
                                servletConnection.setUseCaches (false);
                                servletConnection.setDefaultUseCaches (false);

                                DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
                                DocumentBuilder db = dbf.newDocumentBuilder();
                                doc = db.parse(servletConnection.getInputStream());
                        }
                        catch (Exception e) {
                                System.out.print(e.getMessage());
                        }

                        return (doc.getDocumentElement().getNodeName());
         }          .....
}
```

As classes que estendem o tradutor cujo nome é XMLConnector devem fazer uma redeclaração do método addData (String data, String name), pois cada um dos objetos a serem inseridos no sistema integrado de informação tem sua particularidade (os elementos são diferentes de arquivo para arquivo). Este método irá verificar o nome do elemento XML e seu valor e conseqüentemente atribuí-lo ao atributo correto, observar Tabela 3. A tabela mostra um exemplo simples disso, uma vez que o dado é retornado do XML como string, ele é transformado em float e então atribuído, observar (1) na Tabela 3. Também durante a atribuição podem ser realizadas ações no dado capturado (2) como forma de garantir a homogeneidade dos dados como observado no segundo exemplo, o qual demonstra como é feita a garantia de homogeneidade de dados, é realizada uma comparação com o dado de entrada caso ele seja igual a "Macho" ou "Fêmea", ele é transformado em "M"ou "F".

Tabela 3 – Exemplo de atribuição de dados a um atributo de um determinado objeto

(1)

```
package com.ufscar.shrimp.lib.vo;

/**
 * @author Gustavo
 *
*/

public class statisticVo {
	 ……..
	/**
	 * @param dblen The dblen to set.
	 */
	public void setDblen(String dblen) {
		this.dblen = Float.parseFloat((String)dblen);
	}
	 ..........
}
```

(2)

```
package com.ufscar.shrimp.lib.vo;

/**
 * @author Gustavo
 *
*/

public class libraryVo {
	 ……..
	String sex = new String ();
	/**
	 * @param sex The sex to set.
```

```
        */
        public void setSex (String sex) {

                if (sex.matches("Macho") {
                        sex = "M";
                }
                        else if( sex.matches("Fêmea")) {
                                sex = "F";
                        }
                this.sex = sex;
        }
}
```

4.5. Considerações Finais

Essa implementação foi realizada utilizando a linguagem JAVA e IDE para desenvolvimento Eclipse. Para a construção do sistema de armazenamento de dados foi utilizado o IBM DB2, o qual possui um SGBD, uma ferramenta para criação do DW e uma ferramenta para criação e visualização de cubos (esta última também desempenha o papel de interface). Deve-se observar, que o ambiente pode ser portado para um SGBD livre, tal como o PostgreSQL. Nesse caso, uma ferramenta para geração de cubos teria que ser desenvolvida.

No próximo capítulo serão apresentados alguns testes de possíveis consultas de serem realizadas no sistema.

5. Testes realizados

Na seção anterior foram descritos os aspectos de implementação do ambiente Bio-TIM. Após sua implementação foram materializados dados coletados a partir de fontes de dados públicas via Internet. Então, alguns cubos foram construídos e possibilitando a realização de consultas com a participação de biólogos do Departamento de Genética e Evolução da UFSCar. A seguir, é descrito um dos cubos definidos e uma possibilidade de consulta.

Na figura 24 está ilustrada a modelagem de um cubo, o qual possui três dimensões, que são: proteína (nome), seqüência (nomes de genes) e library (nome dos órgãos).

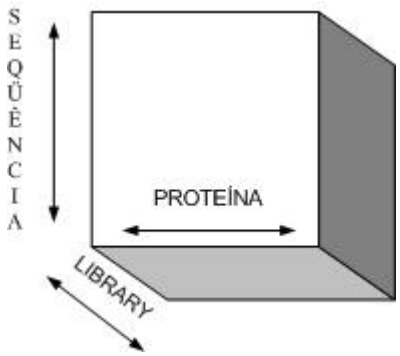

Figura 26 – Possível cubo a ser utilizado em consultas no DW proposto

Na figura 25 estão ilustrados exemplos de possíveis "cortes" no cubo para visualização. Por exemplo, no primeiro corte é escolhido um parâmetro da dimensão Library e é retornado como resultado todos os dados das dimensões de seqüências (Sequence) e proteínas (Protein) relacionados. No segundo corte é escolhido um parâmetro da dimensão proteína (Protein) e é retornado como resultado todos os dados das seqüências (Sequence) e libraries (Library) relacionados. Por sua vez, no terceiro corte é escolhido um parâmetro da dimensão seqüência e é retornado todos os dados das dimensões proteínas (Protein) e libraries (Library) relacionados.

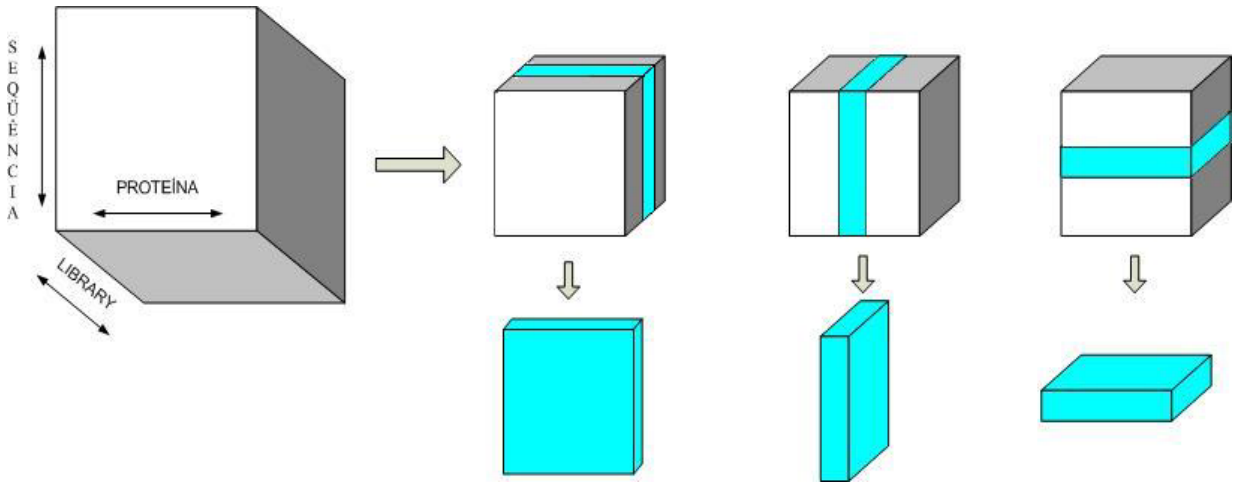

Figura 27 – Exemplos de cortes possíveis em cada dimensão de um cubo.

Além da possibilidade dos diferentes cortes e parâmetros escolhidos, pode-se, por exemplo, refinar ou generalizar os detalhes dos resultados obtidos. Na figura 26 está ilustrada uma busca onde o usuário definiu o nome da proteína de um órgão e obteve como resultado qual gene dentro das células daquele órgão é responsável pela transcrição da proteína escolhida.

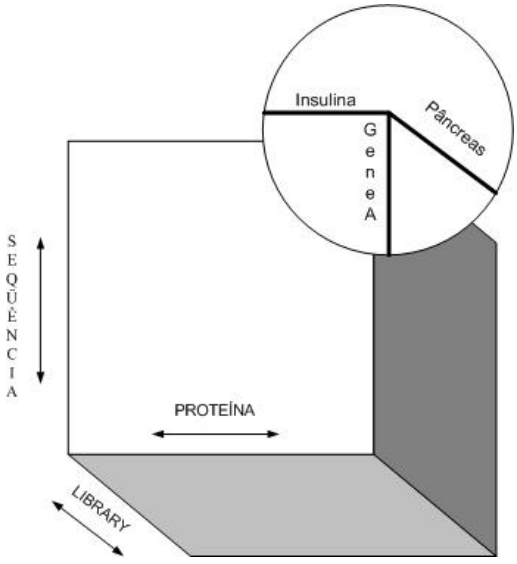

Figura 28 - Busca refinada de dados em um cubo

Os testes realizados demonstraram a utilidade do ambiente, incluindo as facilidades na realização de consultas *ad hoc*, demonstrando a flexibilidade e profundidade do ambiente, uma vez que cada um dos parâmetros de todas as dimensões funciona como um filtro de busca da informação. Assim é dada a possibilidade de uma busca diferente no cubo.

6. Conclusão

Neste projeto foi definida uma arquitetura para um ambiente de bioinformática denominada Bio-TIM (BioInformatics – Transparent Information Management), sendo implementados os tradutores necessários, um gerenciador de conexões a diferentes SGBDs e um Data Warehouse específico. Os resultados alcançados proporcionaram o uso de um ambiente centralizado para consultas flexíveis, rápidas e eficientes.

No levantamento bibliográfico realizado, não foi encontrado um sistema com características que possibilitasse consultas acessando somente uma única fonte de dados integrada, com armazenamento histórico da informação gerada no decorrer do seqüenciamento e construção de cubos para consultas. Apesar de existirem vários *frameworks* de bioinformática como Bio-AXS [SEIBEL00], DiscoveryLink [HAAS00] [HAAS01], Tambis [TAMB03] e outros que possibilitavam o desenvolvimento de aplicação com consultas *on the fly* (consultas feitas de modo *online*, isto é, quando a consulta é disparada o sistema busca a informação através da Internet), na maior parte das vezes com a utilização destes *frameworks* há a necessidade dos sistemas estarem disponíveis no momento das consultas. Além disso, a possibilidade de navegação sobre os dados integrados, agrega maior semântica de informações, o que pode proporcionar um caminho mais curto para a descoberta de conhecimento.

6.1. Principais contribuições

Estão entre as contribuições geradas por este trabalho:

- Definição da arquitetura para o ambiente Bio-TIM.
- Sistema integrado para a construção de consultas.

- Modelagem multidimensional e implementação do DW para o domínio de Bioinformática, no contexto do projeto Genoma do Camarão.
- Base de informação para futura implementação de mecanismos de indexação de dados dentro do contexto do genoma.

6.2. Trabalhos Futuros

- Interface de Navegação e Edição de Dados

Não foi construída uma interface de consulta ao sistema que utilize como meio para este fim a Internet. Esta interface deve ser idealizada e implementada de forma a permitir ao usuário uma navegação flexível pelo sistema, isto é, não deve haver somente disponíveis algumas consultas fixas. Essa interface de consulta deve fazer a utilização de ferramentas WOLAP (utilização de uma ferramenta OLAP através da web [CARVALHO04]).

A ferramenta de navegação deverá armazenar os parâmetros utilizados pelos usuários em suas consultas (utilizando XML para este fim) e assim poder sempre disponibilizar dados atualizados para o cientista da consulta alvo a qualquer momento. Seguir algumas das metodologias de desenvolvimento de interface listadas na seção 2.3.3 trará grandes benefícios aos usuários do sistema.

Com o objetivo de facilitar o posterior processo de desenvolvimento evolutivo dessa ferramenta de interface, seria de grande valia o uso extensivo de componentes, pois seria dada maior dinâmica na construção de novas aplicações, uma vez que os objetos pertinentes a uma aplicação de bioinformática já estariam desenvolvidos.

- Aplicação de métodos de busca no momento da submissão

Junto a interface a ser desenvolvida, deve também ser desenvolvido um sistema de controle de submissão, que busca a seqüência na fonte de dados antes de ser inserida. Caso

encontre a seqüência (medindo a distância entre a seqüência submetida e a encontrada no banco) dar a possibilidade ao pesquisador de fazer uma referência à seqüência encontrada na ficha da seqüência submetida.

- Utilização de métodos indexadores para criação de interface de navegação

Conhecendo o contexto que envolve a Genética, devem ser realizadas pesquisas relacionadas a métodos indexadores de dados genômicos, possibilitando a realização de estudos sobre desempenho de métodos indexadores aplicados nesta fonte de informação.

É de conhecimento da comunidade científica que existem vários parâmetros que fazem com que, por exemplo, um gene esteja "mais próximo"de outro, como por exemplo, o desempenho de uma mesma função no organismo ou a codificação de proteínas que tenham funções semelhantes em um determinado momento do desenvolvimento do ser vivo entre outros, ou seja, deve ser feito um método indexador de metadados associados às seqüências, proteínas, bibliotecas etc.

Pode-se convergir a utilização destes métodos com a construção de interfaces de navegação dinâmicas. Uma nova interface com o usuário que permite ao mesmo navegar de forma mais intuitiva no conjunto de dados. Neste ponto, heurística, cognição e percepção devem ser considerados no desenvolvimento de ferramentas de auxílio à navegação.

Como ilustrado na Figura 29, este índice/interface deveria ser disponibilizado para que o usuário pudesse interagir sobre ele, ou seja, que ele navegue sobre o índice tendo a possibilidade de visualizar que outros nós estão próximos da árvore e que metadado faz com que um gene, proteína ou seqüência esteja mais ou menos próximo do objeto pesquisado. Assim o pesquisador terá uma visão global do objeto de estudo e a ferramenta computacional poderá auxiliá-lo melhor na descoberta de conhecimento.

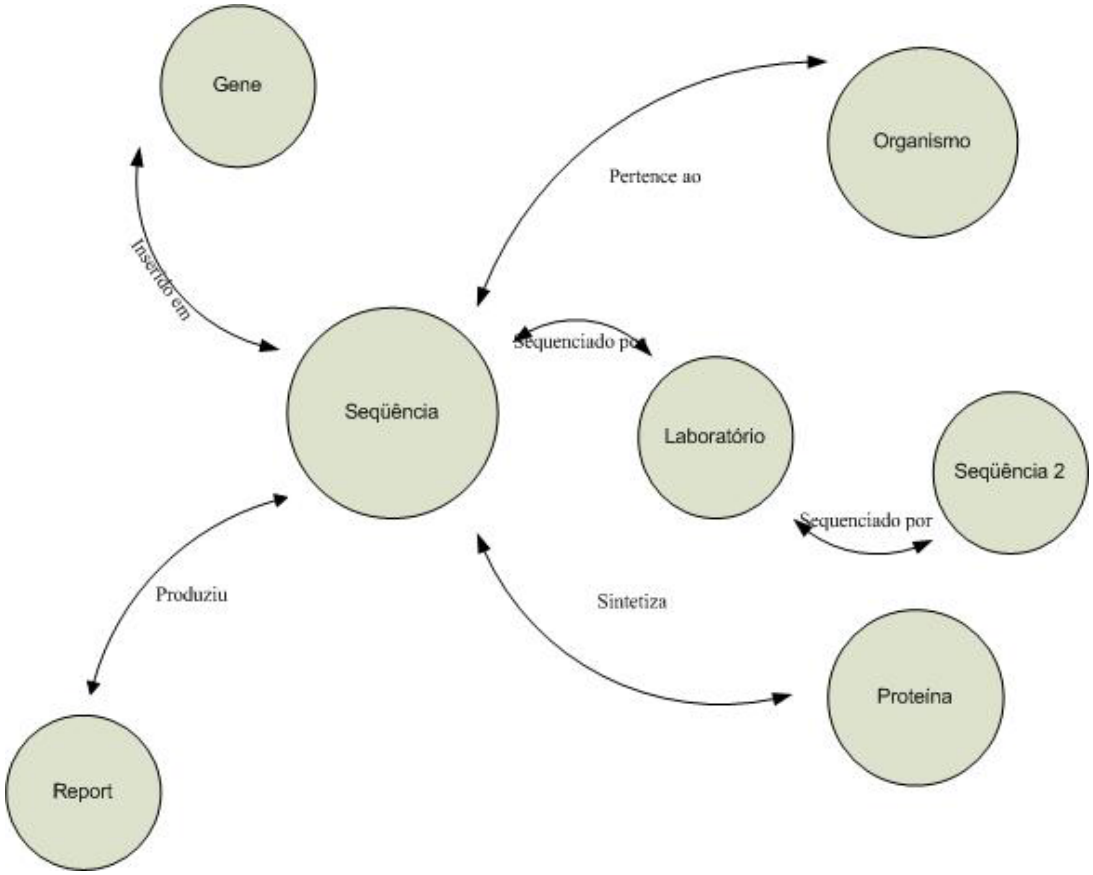

Figura 29 – Exemplificação do sistema de navegação a ser criado

- Estudo do Proteoma

Com o avanço da biotecnologia uma nova frente de estudos se abriu, a do proteoma, que pode ser definido como o conjunto de proteínas produzidas pelo genoma. O proteoma é maior, pois existem mais proteínas do que genes, e mais complexo que o genoma, porque o proteoma não pode ser limitado a um conjunto de seqüências de proteínas, ao contrário do genoma que é definido por uma seqüência de nucleotídeos. No proteoma é necessária a compreensão das estruturas das proteínas, suas funções e interações.

Então, assim como no genoma, existe a necessidade de uma fonte de dados convergente e padronizada, pode-se concluir que para o estudo do proteoma também há esta necessidade.

7. Referências Bibliográficas

[ACHARD01] Achard F.; XML, bioinformatics and data integration; Bioinformatics Review Vol. 17 no. 2 2001 Pages 115-125.

[ADAMS00] Adams, M.D., et al., "The genome sequence of Drosophila melanogaster" Science, 287(5461):2185-95, 24 March 2000

[BAIROCH00] Bairoch, A.; The Enzime Database in 2000; Nucleic Acids Research (2000), Vol. 28, No. 1, pp. 304-305

[BAXEVANIS01] Baxevanis, A., Ouellette, B; Bioinformatics – A Practical Guide to the analysis of Genes and Proteins; Wiley Inter-Science, Second Edition, 2001

[BERMAN00] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne: The Protein Data Bank. Nucleic Acids Research, 28 pp. 235-242 (2000)

[BLAST03] Basic Local Alignment Search Tool. Disponível em http://www.ncbi.nlm.nih.gov/BLAST/ Acessado em Dezembro/2003.

[BRAY00] Bryan T.; Extensible Markup Language (XML) 1.0 (Second Edition). Disponível em http://www.w3.org/TR/2000/REC-xml-20001006. Acessado em 03/12/2003.

[BSCS73] Biological Sciences Curriculum Study; Biologia das Moléculas ao Homem, Parte 1; Edart, 1973.

[CARVALHO04] Carvalho, B. F.; Arquiteturas de ferramentas OLAP; SQL Magazine; Ano 1; Edição 9; 2004.

[CHAUDHURI97] Charudhuri, S.; Dayal, U. An Overview of Data Warehousing and OLAP Technology. ACM SIGMOD, 1997.

[CLAVERIE03] Claverie J.M., Notredame C; Bioinformatics for Dummies;.Wiley, 2003

[DAVIDSON01] Davidson S., Wong L., The Kleisli Approach to Data Transformation and Integration. 2001.

[DOE05] DOEgenomes.org; Humane Genome Project Information; Genome Programs of the U. S. Departament of Energy Office and Science; Disponível em http://doegenomes.org/; Acessado em Maio/2005

[DMREV05] DMReview Magazine; Disponível em http://dmreview.com/; Acessado em Maio/2005

[DUBITZKY01] Dubitzky W., Krebs O., Eils R., "Minding, OLAPing, and Mining Biological Data: Towards a Data Warehousing Concept in Biology", in Proc. Network Tools and Applications in Biology (NETTAB), CORBA and XML: Towards a Bioinformatics Integrated Network Environment, pp78-82, Genova, Italy, 2001.

[ENZ03] Enzyme Nomeclature Database. Disponível em http://us.expasy.org/enzyme/ Acessado em Dezembro/2003

[ELMASRI99] Elmasri, R., Navathe, S. B.; Fundamentals of Database Systems; Third Edition; Addison-Wesley; 1999

[EWING05] Ewing, B; Phred – Base-Calling software with Quality Information; Disponível em http://www.genome.washington.edu/UWGC/analysistools/Phred.cfm; University of Washington Genome Center. Acessado em Abril/2005

[FREIER02] Freier A, Hofestadt R, Lange M, Scholz U, Stephanik A., BioDataServer: a SQL-based service for the online integration of life science data, em Silico Biology; v. 2, n.2, p. 37-57, 2002

[GBK03] GenBank. Disponível em http://www.ncbi.nlm.nih.gov/Genbank/index.html. Acessado em Dezembro/2003

[HAAS00] Haas L., Kodali P., Rice J., Schwarz P., Swope W., Integrating Life Sciences Data - With a Little Garlic. IEEE International Symposium on Bio-Informatics and Biomedical Engineering (BIBE'00), 2000.

[HAAS01] Haas L., Schwarz P., Kodali P., Kotlar E., Rice J. and Swope W., DiscoveryLink: A System for Integrated Access to Life Sciences Data Sources. IBM Systems Journal, 40(2), 489-511, 2001.

[IAL04] O que é Bioinformática?, Instituto Adolf Lutz. Disponível em http://www.ial.sp.gov.br/bioinfo/oqbioinfo.htm. Acessado em Janeiro/2004.

[INMON92] Inmon W. H. Building the Data Warehouse. John Wiley, 1992.

[KOCH02] Koch N., Kraus A.; The Expressive Power of UML-based Web Engineering; International Workshop on Web Oriented Software Technology; 2002

[KOCH03] Koch N., Kraus A., Melià C. C S.; Modeling Web Business Processes with OO-H and UWE; International Workshop on Web Oriented Software Technology; 2003

[LAAKSO05] Laakso S. A.; User Interface Design Patterns; Disponível em http://www.cs.helsinki.fi/u/salaakso/patterns/index.html; Acessado em Junho/2005

[LEMOS03] Lemos M., Seibel L., Casanova M.; Sistemas de Anotações em Biosseqüências; Monografia de Ciência de Computação; Puc-Rio; Março/2003

[LENGAUER00A] Lengauer, T.; "Bioinformatics: From the Pre-genomic to the Post-genomic Era", ERCIM News No.43, Outubro, 2000.

[LENGAUER00B] Lengauer, T ; "Computational Biology at the Beginning of the Post-genomic Era", Lecture Notes for Computer Science Volume 2000. "Informatics: 10 Years Back - 10 Years Ahead", Reinhard Wilhelm (Ed.), Springer, Berlin, p. 341-355 2000.

[LUUKKAINEN99] Luukkainen S., Vuorio S.; Design project of the User Interfaces course; Department of Computer Science, University of Helsinki, 1999.

[PDB03] RCSB Protein DataBank. Disponível em http://www.rcsb.org/pdb/. Acessado em Dezembro/2003.

[PHRAP03] Documentation for Phrap and Cross-Match. Disponível em http://www.phrap.org/phrap.docs/phrap.html. Acessado em Dezembro/2003.

[PRINCE05] Princeton University Cognitive Science Laboratory. Disponível em http://www.cogsci.princeton.edu/. Acessado em Maio/2005

[REINACH05] Reinach F.; Com quantos genes de faz uma canoa; Jornal da Ciência – Sociedade Brasileira de Progresso da Ciência; Maio/2005

[SBFIS98] Sociedade Brasileira de Fisiologia, Boletim 23 (1). Disponível em http://www.sbfis.org.br/boletim/Boletim%2023(1)%201998.htm. Acessado em Dezembro/2003.

[SEIBEL00] Seibel, L; Bio-AXS: "Uma Arquitetura para Integração de Fontes de Dados e Aplicações de Biologia Molecular", Tese de Doutorado, PUC-Rio, Dep. de Informática, 2000.

[SILVA00] Silva F.; Curso: Fundamentos em Biotecnologia; CBME (CEPID-FAPESP), 2000.

[SONNICHSEN05] Sonnichsen, B et al; Full-genome RNAi profiling of early embryogenesis in 'Caenorhabditis elegans'; Nature; Março 2005

[SPINGA98] Spinga R., Maia J.; A Biologia nos Vestibulares; Navegar Editora, 1998.

[SPROT03] Swiss-Prot. Disponível em http://us.expasy.org/sprot/. Acessado em Dezembro/2003

[TAMB03] Tranparent Access to Multiple Bioinformatics Information Sources. Disponível em http://imgproj.cs.man.ac.uk/tambis/ Acessado em Dezembro/2003

[UCB04] Entendendo a Biotecnologia. Disponível em http://www.ucb.br/posgraduacao/biotecnologia/defini%E7%E3o.htm. Acessado em Janeiro/2004.

[UZUNIAN91] Uzunian A.; Introdução à Biologia, Editora Anglo, 1991.

[VIEIRA02] Vieira M., Felipe J., Data Warehouse, Apostila, 2003.

[W3C05] World Wide Web Consortium; Disponível em http://w3c.org/ Acessado em Abril/2005

[WELIE05] Walie, M.; Patterns in Interaction Desgin; Disponível em http://www.welie.com/; Acessado em Maio/2005

8. Bibliografia

[CAP303] CAP3: A DNA sequence assembly program. Disponível em http://www.genome.org/cgi/content/full/9/9/868 Acessado em Dezembro/2003

[DDBJ03] DNA DataBank of Japan. Disponível em http://www.ddbj.nig.ac.jp/Welcome-j.html. Acessado em Dezembro/2003

[EMBL03] European Molecular Biology Laboratory. Disponível em http://www.ebi.ac.uk/embl/. Acessado em Dezembro/2003

[INMON96] Inmon W. H.; Como Construir um Data Warehouse, Tradução. Rio de Janeiro:Editora Campus Ltda., 2a edição, 1996

[KIMBALL96] Kimball R.; The Data Warehouse Toolkit - Practical Techniques for Building Dimensional Data Warehouses, John Wiley & Sons, 1996

[FASTA03] The FASTA Program Package. Disponível em http://bioinfo.tau.ac.il/man/fasta/fasta.html. Acessado em Janeiro/2004.

[MOURA02] MOURA, A. M. de C.; A web semântica: fundamentos e tecnologias. Rio de Janeiro: IME, 2002. Disponível em: http://ipanema.ime.eb.br/~anamoura/publicacoes.html. Acessado em Maio/2004

[NORMAN01] Norman W. Paton, Carole A. Goble: Information Management for Genome Level Bioinformatics. VLDB 2001

[PRBIO04] Large Molecules Problems Set. Disponível em http://www.projeto-biologico.arizona.edu/biochemistry/problem_sets/large_molecules/03t.html. Acessado em Maio/2004.

[PROSITE03] PROSITE Database of protein families and domains. Disponível em http://us.expasy.org/prosite/. Acessado em Dezembro/2003.

[ROCHA03] Rocha, R. L. A., Cardoso, L. F., Souza, J. M., “An Improved Approach in the Data Warehousing ETML Process for Detection of Changes in Source Data”. SBBD 2003

[SNUSTAD97] Snustad, Simmons, Jekins; Principles of Genetics; John Wiley and Sons, Inc

[STOESSER03] Stoesser, G. et all, Nucleic Acids Research, 2003, Vol. 31, No. 1 pp. 17-22

[THOMPSON01] Thompson H.; XML Schema Part 1: Structures. Disponível em http://www.w3.org/TR/2001/REC-xmlschema-1-20010502. Acessado em Dezembro/2003.

9. Glossário

Ácido Nucleico	Molécula biológica composta por uma longa cadeia de nucleotídeos. O DNA é formado por quatro tipos de nucleotídeos repetidos aleatoriamente milhares de vezes.
Adenina	Base nitrogenada encontrada no DNA e RNA.
Alelo	Uma das diversas formas alternativas de um gene específico que ocupa uma certa região do cromossomo.
Aminoácidos	Unidades fundamentais de uma molécula de proteína. Uma proteína é uma cadeia de centenas ou milhares de aminoácidos. Nosso corpo pode sintetizar a maioria dos aminoácidos a partir de seus componentes (carbono, nitrogênio, oxigênio, hidrogênio e, algumas vezes, enxofre). Entretanto, oito aminoácidos (chamados aminoácidos essenciais) devem ser obtidos através da dieta alimentar.
Bioinformática	É o uso de técnicas computacionais para análises de caracterização molecular, integrando modelos matemáticos e estatísticos utilizados na interpretação e análise dos problemas biológicos.
Biotecnologia	Utilização de organismos vivos para resolução de problemas e geração de produtos de interesse. Atualmente é definido como o conjunto de tecnologias que utilizam células vivas e/ou moléculas biológicas para resolução de problemas e geração de produtos de interesse.
Catalisador	Substância que torna a reação química mais rápida, mas não é modificado neste processo.
Citoplasma	Material celular contido dentro de uma membrana celular e que circunda o núcleo.
Citosina	Base nitrogenada encontrada no DNA ou RNA.
Contig	Conjunto de reads.
Cromossomo	Componentes celulares que contém as informações genéticas. Cada cromossomo contém inúmeros genes. Os cromossomos ocorrem aos pares: um proveniente da mãe e outro proveniente do pai. Cromossomos de casais diferentes são visivelmente diferentes.
DNA (ácido desoxirribonucléico)	Molécula que é a base do material genético encontrado em todas as células. O DNA carrega as informações genéticas de uma geração para a próxima. Como o DNA é uma molécula muito longa e fina, ele é arranjado em unidades, chamadas cromossomos. O DNA pertence a uma classe de moléculas biológicas, chamada de ácidos nucleicos.
Dupla hélice	Termo usado para descrever a configuração da

	molécula de DNA. A hélice consiste de duas fitas de nucleotídeos, espiralizadas, ligadas uma a outra por pontes de hidrogênio entre as bases.
Enzima	Proteínas que aceleram a velocidade das reações químicas. As enzimas são catalisadores que promovem repetidamente as reações sem serem modificadas por elas.
Expressão do gene	O processo pelo qual a informação codificada de um gene é convertida nas estruturas atuais e presentes na célula. Os genes expressados incluem aqueles que são transcritas no RNAm e então traduzidas na proteína e aqueles que são transcritas no RNA mas não traduzidas na proteína (por exemplo, transferência e RNA ribosomal).
Gene	Unidade de informação hereditária que define características específicas de cada individuo. Um gene é uma seção da molécula do DNA que especifica a produção de uma proteína em particular.
Genética	Ramo da Biologia que estuda os mecanismos de transmissão das características estruturais e funcionais de um organismo para as gerações subseqüentes
Genoma	É o conjunto de informação genética de um ser vivo, contido no DNA. Sinônimo de genótipo, patrimônio genético ou patrimônio hereditário. Formado por um conjunto de contigs.
Guanina	Base nitrogenada encontrada no DNA ou RNA.
Moléculas biológicas	Moléculas grandes e complexas, como proteínas, lipídeos e carboidratos, que são produzidas somente por organismos vivos. As moléculas biológicas são normalmente chamadas de macromoléculas ou biopolímeros.
Núcleo	A estrutura dentro das células eucarióticas, circundadas por uma membrana, que contém os cromossomos de um organismo.
Nucleotídeo	Constituinte elementar dos ácidos nucleicos (DNA e RNA). É composto por quatro bases nitrogenadas (adenina, guanina, citosina e tiamina), de um fosfato e de um açúcar. A seqüência de nucleotídeos constitui o código genético de um organismo.
Pipeline	É um sistema constituído por vários softwares, sendo que cada um deles tem um objetivo dentro do processo de montagem e anotação do seqüenciamento genético.
Primer	Pequenas seqüências previamente conhecidas
Polímero	Grande molécula formada por uma série de ligações covalentes, que unem várias unidades idênticas ou semelhantes (monômeros).
Polipeptídeo	Polímero linear composto por múltiplos aminoácidos. Proteínas são grandes polipeptídeos, e os dois termos podem ser usados como sinônimos.

Proteína	Uma molécula composta de aminoácidos. Existem muitos tipos de proteínas, todas desenvolvem um número diferente de funções essenciais para o crescimento da célula.
Read	Fragmento de DNA seqüenciado que contém de 500 a 800 bases
Região de Sinalização de Ribossomos	Motivo conservado em RNAm de procariotes que é complementar à seqüência próxima a extremidade 5´do RNA ribossomal 16S e envolvida com a iniciação da tradução.
Região Promotora	Uma determinada seqüência de nucleotídeos onde a RNA polimerase se liga e inicia a transcrição.
Ribossomo	Grânulo citoplasmático constituído por RNA e proteínas. É o responsável pela síntese de proteínas.
RNA (ácido ribonucléico)	Assim com o DNA, o RNA é um tipo de ácido nucléico. O RNA se diferencia do DNA em três aspectos: os nucleotídeos do RNA contem o açúcar ribose ao invés do desoxirribose; o RNA contém a base uracila ao invés da timina; e o RNA é uma molécula de fita simples ao invés de uma hélice dupla fita.
RNAm (RNA mensageiro)	Ácido, nucléico, fita simples, que carrega a instrução para o ribossomo sintetizar uma proteína em particular.
RNAt (RNA transportador)	Molécula de RNA que carrega aminoácidos para sítios nos ribossomos para síntese das proteínas.
Seqüência de DNA	A ordem de bases na molécula de DNA.
Seqüenciamento de DNA	Determinação da seqüência de bases do DNA.
Timina	Base nitrogenada encontrada no DNA.
Tradução	Processo que utiliza um RNA mensageiro molde para sintetizar uma proteína.
Transcrição	Processo de utilização de um DNA molde para fazer uma molécula de RNA complementar. É a primeira etapa da expressão do gene.
Uracila	Base nitrogenada encontrada no RNA.
Vetor	O agente (plasmídeo ou vírus) usado para transferir DNA para uma célula.

Apêndice A

Dicionário de Dados

AutoAnnotation

Aan_id	Identificador
Aan_qryStart	Posição inicial do alinhamento na seqüência
Aan_qryEnd	Posição final do alinhamento na seqüência
Aan_identity	Porcentagem de identidade
Aan_evalue	E-value do alinhamento
Aan_score	Score do alinhamento
Aan_nucl	Nucleotídeos da consulta no alinhamento
Aan_homoString	String que representa a homologia no alinhamento
Aan_hitString	Identificador do banco de seqüências

Base

Base_pos	Posição da base
Bas_qual	Qualidade
Bas_name	Nome

Blast

Bls_idReport	Identificador do relatório
Bls_program	Nome do programa Blast utilizado
Bls_version	Versão
Bls_reference	Referência
Bls_db	Nome do banco de dados utilizado
Bls_queryId	Id da consulta
Bls_queryDef	Definição da consulta
Bls_queryLen	Tamanho da consulta
Bls_querySeq	Seq da seqüência consultada

CrossMatch

Crm_idReport	Identificador do relatório
Crm_value	Valor “smith-waterman” do acerto
Crm_substituition	Porcentagem de substituições na região do acerto
Crm_deletion	Porcentagem de remoções na região do acerto
Crm_insertion	Porcentagem de inserções na região do acerto
Crm_firstSeqId	Identificador da primeira seqüência
Crm_firstStartPos	Posição inicial do acerto na primeira seqüência
Crm_firstEndPos	Posição final do acerto na primeira seqüência
Crm_firstNumBases	Número de bases anteriores ao início da

	região do primeiro acerto
Crm_complSeq	Id da seqüência complementar para o acerto
Crm_secComplBases	Número de bases em complemento a segunda seqüência para o início do acerto
Crm_seqStartPos	Posição inicial da segunda seqüência
Crm_seqEndPos	Posição final da segunda seqüência
Crm_version	Versão do software utilizado

Database

Dbs_id	Identificador
Dbs_genbank	Entrada para o GenBank
Dbs_annotation	Anotação relativa as entradas
Dbs_unigene	Acesso ao Unigene
Dbs_locusLink	Acesso ao LocusLink
Dbs_pdb	Entrada para o PDB
Dbs_enzyme	Entrada para o Enzyme

Fact

Fct_id	Identificador do fato
Fct_date	Data e hora de inserção do fato no sistema
Fct_comment	Comentário realizado sobre o fato

Hit

Hit_num	Número do Hit
Hit_id	SeqId do assunto
Hit_def	Linha de definição do assunto
Hit_accession	Acesso
Hit_len	Tamanho

Hsp

Hsp_num	Número do HSP (High-Scoring Segment)
Hsp_bitScore	Resultado em bits do HSP
Hsp_score	Resultado do HSP
Hsp_evalue	“E-value” do HSP
Hsp_queryFrom	Início do HSP
Hsp_queryTo	Final do HSP
Hsp_hitFrom	Início do HSP no assunto
Hsp_hitTo	Final do HSP no assunto
Hsp_patternFrom	Início do padrão PHI-BLAST
Hsp_patternTo	Final do padrão PHI-BLAST
Hsp_queryFrame	Tradução do frame de consulta
Hsp_hitFrame	Tradução do frame do assunto
Hsp_identity	Número de identificadores do HSP
Hsp_positive	Número de positivos no HSP
Hsp_gaps	Número de saltos no HSP
Hsp_alignLen	Tamanho do alinhamento utilizado
Hsp_density	Densidade do resultado
Hsp_qseq	String de alinhamento para a consulta (com saltos)

Hsp_hseq	String de alinhamento para o assunto (com saltos)
Hsp_midLine	Linha média formatada

Iteration

Iter_iterNum	Número da iteration
Iter_queryId	SeqId da consulta
Iter_queryDef	Linha de definição da consulta
Iter_queryLen	Tamanho da seqüência da consulta
Iter_message	Mensagem gerada

IterStatistic

Ist_dbNum	Número de seqüências no banco utilizado pelo Blast
Ist _dbLen	Tamanho do banco
Ist _hspLen	Tamanho efetivo do HSP
Ist _effSpace	Espaço efetivo da busca
Ist _kappa	Parâmetro Karlin-Altschul (K)
Ist _lambda	Parâmetro Karlin-Altschul (Lambda)
Ist _entropy	Parâmetro Karlin-Altschul (H)

Laboratory

Lab_id	Identificador da instituição
Lab_name	Nome
Lab_address	Endereço
Lab_resp	Nome do responsável
Lab_respMail	E-mail do responsável
Lab_phone	Telefone
Lab_fax	Fax

Library

Lib_id	Identificador
Lib_organism	Nome científico do organismo que a biblioteca foi preparada
Lib_strain	“Strain” do organismo
Lib_cultivar	“Plant cultivar”
Lib_isolate	Local de onde a seqüência foi obtida
Lib_sex	Sexo do organismo
Lib_organ	Nome do órgão
Lib_tissue	Nome do tecido
Lib_cellType	Tipo de célula
Lib_cellLine	Nome da linha da célula
Lib_stage	Estágio de desenvolvimento
Lib_host	Host
Lib_vector	Vetor
Lib_vecType	Tipo do vetor
Lib_re1	Enzima de restrição no “site 1”
Lib_re2	Enzima de restrição no “site 2”
Lib_desc	Descrição dos métodos de preparação da

	biblioteca

Parameter

Par_matrix	Matrix utilizada (-M)
Par_expect	Ponto inicial esperado (-e)
Par_include	Ponto inicial de inclusão (-h)
Par_scMatch	“Match score” (-r)
Par_scMismatch	“Mismatch score” (-q)
Par_gapOpen	Custo do salto aberto (-G)
Par_gapExtend	Custo da extensão do salto (-E)
Par_Filter	Opções de filtro (-F)
Par_Pattern	Padrão PHI Blast
Par_entQuery	Limite de requisições do Entrez

Phred

Prd_idReport	Identificador
Prd_startQual	Posição inicial da região de alta qualidade
Prd_endQual	Posição final da região de alta qualidade
Prd_chromatName	Nome do arquivo que contém o cronomatograma
Prd_qtyLevel	Valor máximo da qualidade
Prd_dataHora	Data e horário de criação do arquivo gerado pelo utilitário
Prd_call	Parâmetros utilizados na chamada do Phred
Prd_ratio	Taxa de bases identificadas por não identificadas
Prd_abi	Número fonecido pelo ABI ThumbPrint
Prd_version	Versão so software utilizado

Primer

Prm_id	Identificador do Primer
Prm_name	Nome
Prm_start	Posição inicial
Prm_end	Posição final

Protein

Prt_id	Identificador
Prt_name	Nome
Prt_function	Função
Prt_phase	Fase de desenvolvimento do organismo que a proteína tem a função descrita
Prt_prdLocal	Local de produção
Prt_prdOrgan	Orgão que produz a proteína
Prt_amino	Aminoácidos que a compõe

Publication

Pub_id	Identificador da publicação
Pub_name	Título
Pub_author	Autor

Pub_type	Tipo
Pub_year	Ano
Pub_edition	Edição
Pub_pages	Páginas
Pub_press	Editora
Pub_local	Local

Repeat

Rep_id	Identificador da repetição
Rep_name	Nome
Rep_start	Posição inicial
Rep_end	Posição final

Report

Rpt_id	Identificador do relatório

Research

Rch_id	Identificador do pesquisador
Rch_name	Nome
Rch_mail	E-mail
Rch_fone	Telefone
Rch_fax	Fax
Rch_address	Endereço

Statistic

Stt_dbNum	Número de seqüências no banco utilizado pelo Blast
Stt_dbLen	Tamanho do banco
Stt_hspLen	Tamanho efetivo do HSP
Stt_effSpace	Espaço efetivo da busca
Stt_kappa	Parâmetro Karlin-Altschul (K)
Stt_lambda	Parâmetro Karlin-Altschul (Lambda)
Stt_entropy	Parâmetro Karlin-Altschul (H)

Sequence

Seq_id	Identificador da seqüência
Seq_name	Nome
Seq_fasta	Fasta
Seq_lenght	Tamanho
Seq_strand	Orientação
Seq_tecAcro	Acrônimo
Seq_comment	Comentário sobre a seqüência
Seq_polya	Booleano que indica se a cauda polyA foi ou não encontrada no na EST
Seq_prodId	Identificador da seqüência que foi gerado
Seq_expId	Identificador do experimento
Seq_plateId	Identificador da placa do experimento
Seq_prodDate	Data de sequenciamento da seqüência
Seq_dnaType	Indica se é cDNA, genômico, viral, sintético

	ou outro
Seq_pEnd	Região de sinalização
Seq_dbXref	Acesso referenciado ao banco de dados
Seq_dbName	Nome do banco de dados para referência à outro banco de dados
Seq_idEst	Identificador associado ao EST pelo laboratório
Seq_prom	Posição de início da região de transcrição
Seq_qlyStart	Posição que inicia a região de alta qualidade
Seq_qlyEnd	Posição que termina a região de alta qualidade
Seq_type	Identifica se a seqüência é mitocondrial (M), de ribossomos (R), proteínas (P).

Vector

Vec_id	Identificador do vetor
Vec_name	Nome
Vec_start	Posição inicial
Vec_end	Posição final

Apêndice B

Gerenciador de Conexões

DataBaseLayer.java

```
package com.lib.database;

import java.sql.*;


public class DatabaseLayer {
	private static DatabaseLayer instance;
	private static DBConnectionBroker connectionPool = null;

	/**
	 * Create an instance of DataBase Layer using the properties definied on parameter
file
	 * @param String fileName -> Data Base properties file
	 * @throws SQLException
	 * @since
	 */
	public static DatabaseLayer getInstance(String fileName) throws SQLException {

		if (instance == null) {
			DBProperties props = DBProperties.getInstance(fileName);

			if (props == null ||
				props.getProperty("dbdriver") == null ||
				props.getProperty("dburl") == null ||
				props.getProperty("dbuser") == null ||
				props.getProperty("dbpwd") == null ||
				props.getProperty("dbmin") == null ||
				props.getProperty("dbmax") == null ||
				props.getProperty("dblogfile") == null ||
				props.getProperty("dbrefresh") == null) {

				System.err.println("Falha na leitura do arquivo
DataBaseProperties");
				throw new SQLException("Falha na leitura do arquivo
DataBaseProperties");
			}

			instance = new DatabaseLayer(props);
		}
		return instance;

	}

	private DatabaseLayer(DBProperties props) throws SQLException {
```

```
            try {
                connectionPool = new DBConnectionBroker(
                    props.getProperty("dbdriver"),
                        props.getProperty("dburl"),
                            props.getProperty("dbuser"),
                                props.getProperty("dbpwd"),

        Integer.valueOf(props.getProperty("dbmin")).intValue(),

        Integer.valueOf(props.getProperty("dbmax")).intValue(),

        props.getProperty("dblogfile"),

        Double.valueOf(props.getProperty("dbrefresh")).doubleValue()

                                        );

            } catch (Exception _ex) {
                System.err.println(new java.util.Date() + " - Falha inicializacao
DatabaseLayer: " + _ex.getMessage());
                throw new SQLException(_ex.getMessage());
            }

        }

        /**
         * Gets the connection attribute of the DBConnectionPool class
         *
         * @return The connection value
         * @exception SQLException Description of Exception
         * @since
         */
        public DBConnection getConnection()
            throws SQLException {
            return (new DBConnection(connectionPool.getConnection()));
        }




        /**
         * Description of the Method
         *
         * @param conn Description of Parameter
         * @since
         */
        public static void releaseConnection(Connection conn) {
            String result = connectionPool.freeConnection(conn);
        }
```

```
	public void destroy() {
		try {
			if (connectionPool!= null)
				connectionPool.destroy();
		} catch (Exception _ex) {
			System.err.println(new java.util.Date() + " - Falha em
DatabaseLayer.destroy(): " + _ex.getMessage());
		}
	}
}
```

DBConnection.java

```
package com.lib.database;


import java.sql.*;


public class DBConnection {
	private Connection conn = null;

	/**
	 *  Constructor for the DBConnection object
	 *
	 *@param  conn             Description of Parameter
	 *@exception  SQLException  Description of Exception
	 *@since
	 */
	DBConnection(Connection conn)
		throws SQLException {

		if (conn == null)
			throw new SQLException("Conexao retornada do pool é nula");

		this.conn = conn;
		this.conn.setAutoCommit(false);
	}

	/**
	 *  Gets the statement attribute of the DBConnection object
	 *
	 *@return                The statement value
	 *@exception  SQLException  Description of Exception
	 *@since
	 */
	public Statement getStatement()
		throws SQLException {
		return (conn.createStatement());
```

```
        }

        /**
         *  Gets the statement attribute of the DBConnection object
         *
         *@param  updateable       Description of Parameter
         *@return                  The statement value
         *@exception  SQLException  Description of Exception
         *@since
         */
        public Statement getStatement(boolean updateable)
                throws SQLException {
                if (updateable) {
                        return
(conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE));
                }
                else {
                        return (conn.createStatement());
                }
        }

        /**
         *  Gets the preparedStatement attribute of the DBConnection object
         *
         *@param  sqlCommand       Description of Parameter
         *@return                  The preparedStatement value
         *@exception  SQLException  Description of Exception
         *@since
         */
        public PreparedStatement getPreparedStatement(String sqlCommand)
                throws SQLException {
                return (conn.prepareStatement(sqlCommand));
        }

        /**
         *  Gets the preparedStatement attribute of the DBConnection object
         *
         *@param  sqlCommand       Description of Parameter
         *@param  updatable        Description of Parameter
         *@return                  The preparedStatement value
         *@exception  SQLException  Description of Exception
         *@since
         */
        public PreparedStatement getPreparedStatement(String sqlCommand, boolean
updatable)
                throws SQLException {
                if (updatable) {
                        return (conn.prepareStatement(sqlCommand
                                        , ResultSet.TYPE_SCROLL_INSENSITIVE
```

```
                                        , ResultSet.CONCUR_UPDATABLE
                                        )
                                        );
                }
                else {
                        return (conn.prepareStatement(sqlCommand));
                }
        }

        /**
         *  Description of the Method
         *
         *@exception  SQLException  Description of Exception
         *@since
         */
        public void commit()
                throws SQLException {
                conn.commit();
        }

        /**
         *  Description of the Method
         *
         *@since
         */
        public void release() {
                DatabaseLayer.releaseConnection(conn);
        }

        /**
         *  Description of the Method
         *
         *@exception  SQLException  Description of Exception
         *@since
         */
        public void rollback()
                throws SQLException {
                conn.rollback();
        }

}
```

DBProperties.java

```
package com.lib.database;


import java.io.InputStream;
//import java.util.Hashtable;
import java.util.Properties;
```

```
public class DBProperties {

	private static DBProperties instance = null;
	private Properties properties = null;

	public static DBProperties getInstance(String fileName) {
		if (instance == null) {
			instance = new DBProperties(fileName);
		}

		return instance;
	}

	private DBProperties(String fileName) {
		System.out.println("Lendo " + fileName + "...");
		properties = loadIniFile(fileName);
		System.out.println("DataBaseProperties Ok.");
	}

	private Properties loadIniFile(String fileName) {
		InputStream is = null;
		Properties result = null;


		try {
			is = getClass().getResourceAsStream("/" + fileName);
		} catch (Exception e) {
			System.err.println(
				"Falha ao abrir arquivo " + fileName + ": " + e.getMessage());
		}


		try {
			if (is != null) {
				result = new Properties();
				result.load(is);
			} else {
				result = new Properties();
				System.err.println(
					"Falha ao abrir arquivo " + fileName + ": is == null");

			}
		} catch (Exception e) {
			System.err.println(
				"Falha ao ler as propriedades do arquivo " + fileName + ": " +
e.getMessage());
		}
```

```
            return result;
        }


        public String getProperty(String propr) {
            return properties.getProperty(propr);
        }

}
```

DBConnectionBroker.java

```
package com.lib.database;

import java.io.*;
import java.sql.*;
//import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;


public class DBConnectionBroker
   implements Runnable
{

DBConnectionBroker

   public DBConnectionBroker(String s, String s1, String s2, String s3, int i, int j, String s4,
         double d)
      throws IOException
   {
      available = true;
      connPool = new Connection[j];
      connStatus = new int[j];
      connLockTime = new long[j];
      connCreateDate = new long[j];
      connID = new String[j];
      currConnections = i;
      maxConns = j;
      dbDriver = s;
      dbServer = s1;
      dbLogin = s2;
      dbPassword = s3;
      logFileString = s4;
      maxConnMSec = (int)(d * 86400000D);
      if(maxConnMSec < 30000)
         maxConnMSec = 30000;
      try
```

```
        {
            log = new PrintWriter(new FileOutputStream(s4), true);
        }
        catch(IOException _ex)
        {
            try
            {
                log = new PrintWriter(new FileOutputStream("DCB_" +
System.currentTimeMillis() + ".log"), true);
            }
            catch(IOException _ex2)
            {
                throw new IOException("Não foi possível abrir arquivo de log");
            }
        }
        SimpleDateFormat simpledateformat = new SimpleDateFormat("yyyy.MM.dd G 'at'
hh:mm:ss a zzz");
        Date date = new Date();
        pid = simpledateformat.format(date);
        BufferedWriter bufferedwriter = new BufferedWriter(new FileWriter(s4 + "pid"));
        bufferedwriter.write(pid);
        bufferedwriter.close();
        log.println("Starting DbConnectionBroker:");
        log.println("dbDriver = " + s);
        log.println("dbServer = " + s1);
        log.println("dbLogin = " + s2);
        log.println("log file = " + s4);
        log.println("minconnections = " + i);
        log.println("maxconnections = " + j);
        log.println("Total refresh interval = " + d + " days");
        log.println("----------------------------------------");
        boolean flag = false;
        byte byte0 = 20;
        try
        {
            for(int k = 1; k < byte0;)
                try
                {
                    for(int l = 0; l < currConnections; l++)
                        createConn(l);


                    flag = true;
                    break;
                }
                catch(SQLException sqlexception)
                {
                    log.println("--->Attempt (" + String.valueOf(k) + " of " + String.valueOf(byte0)
+ ") failed to create new connections set at startup: ");
                    log.println("    " + sqlexception);
```

```
                log.println("    Will try again in 15 seconds...");
                try
                {
                   Thread.sleep(15000L);
                }
                catch(InterruptedException _ex) { }
                k++;
             }


          if(!flag)
          {
             log.println("\r\nAll attempts at connecting to Database exhausted");
             throw new IOException();
          }
       }
       catch(Exception _ex)
       {
          throw new IOException();
       }
       runner = new Thread(this);
       runner.start();
    }


    public void run()
    {
       boolean flag = true;
       Statement statement = null;
       while(flag)
       {
          try
          {
             BufferedReader bufferedreader = new BufferedReader(new
FileReader(logFileString + "pid"));
             String s = bufferedreader.readLine();
             if(!s.equals(pid))
             {
                // log.close();
                for(int k = 0; k < currConnections; k++)
                   try
                   {
                      connPool[k].close();
                   }
                   catch(SQLException _ex) { }


                return;
             }
             bufferedreader.close();
```

```
        }
        catch(IOException _ex)
        {
            log.println("Can't read the file for pid info: " + logFileString + "pid");
        }
        for(int i = 0; i < currConnections; i++)
            try
            {
                currSQLWarning = connPool[i].getWarnings();
                if(currSQLWarning != null)
                {
                    log.println("Warnings on connection " + String.valueOf(i) + " " +
currSQLWarning);
                    connPool[i].clearWarnings();
                }
            }
            catch(SQLException sqlexception)
            {
                log.println("Cannot access Warnings: " + sqlexception);
            }


        for(int j = 0; j < currConnections; j++)
        {
            long l = System.currentTimeMillis() - connCreateDate[j];
            synchronized(connStatus)
            {
                if(connStatus[j] > 0)
                    continue;
                connStatus[j] = 2;
            }
            try
            {
                if(l > (long)maxConnMSec)
                    throw new SQLException();
                statement = connPool[j].createStatement();
                connStatus[j] = 0;
                if(connPool[j].isClosed())
                    throw new SQLException();
            }
            catch(SQLException _ex)
            {
                try
                {
                    log.println((new Date()).toString() + " ***** Recycling connection " +
String.valueOf(j) + ":");
                    connPool[j].close();
                    createConn(j);
                }
                catch(SQLException sqlexception1)
```

```
            {
               log.println("Failed: " + sqlexception1);
               connStatus[j] = 0;
            }
         }
         finally
         {
            try
            {
               if(statement != null)
                  statement.close();
            }
            catch(SQLException _ex) { }
         }
      }


      try
      {
         Thread.sleep(20000L);
      }
      catch(InterruptedException _ex)
      {
         return;
      }
   }
}


public Connection getConnection()
{
   Connection connection = null;
   if(available)
   {
      boolean flag = false;
      for(int i = 1; i <= 10; i++)
      {
         try
         {
            int j = 0;
            int k = connLast + 1;
            if(k >= currConnections)
               k = 0;
            do
               synchronized(connStatus)
               {
                  if(connStatus[k] < 1 && !connPool[k].isClosed())
                  {
                     connection = connPool[k];
                     connStatus[k] = 1;
```

```
                        connLockTime[k] = System.currentTimeMillis();
                        connLast = k;
                        flag = true;
                        break;
                     }
                     j++;
                     if(++k >= currConnections)
                        k = 0;
                  }
               while(!flag && j < currConnections);
            }
            catch(SQLException _ex) { }
            if(flag)
               break;
            synchronized(this)
            {
               if(currConnections < maxConns)
                  try
                  {
                     createConn(currConnections);
                     currConnections++;
                  }
                  catch(SQLException sqlexception)
                  {
                     log.println("Unable to create new connection: " + sqlexception);
                  }
            }
            try
            {
               Thread.sleep(2000L);
            }
            catch(InterruptedException _ex) { }
            log.println("-----> Connections Exhausted!  Will wait and try again in loop " +
String.valueOf(i));
         }


      } else
      {
         log.println("Unsuccessful getConnection() request during destroy()");
      }
      return connection;
   }


   public int idOfConnection(Connection connection)
   {
      String s;
      try
      {
```

```
            s = connection.toString();
        }
        catch(NullPointerException _ex)
        {
            s = "none";
        }
        int i = -1;
        for(int j = 0; j < currConnections; j++)
        {
            if(!connID[j].equals(s))
                continue;
            i = j;
            break;
        }


        return i;
    }


    public String freeConnection(Connection connection)
    {
        Date date = new Date();
        String s = "";
        int i = idOfConnection(connection);
        if(i >= 0)
        {
            connStatus[i] = 0;
            s = "freed " + connection.toString();
                                // log.println(date.toString() + "  Connection " + s );
        } else
        {
            log.println("----> Could not free connection!!!");
        }
        return s;
    }


    public long getAge(Connection connection)
    {
        int i = idOfConnection(connection);
        return System.currentTimeMillis() - connLockTime[i];
    }


    private void createConn(int i)
        throws SQLException
    {
        Date date = new Date();
        try
```

```
        {
            Class.forName(dbDriver);
            connPool[i] = DriverManager.getConnection(dbServer, dbLogin, dbPassword);
            connStatus[i] = 0;
            connID[i] = connPool[i].toString();
            connLockTime[i] = 0L;
            connCreateDate[i] = date.getTime();
        }
        catch(ClassNotFoundException _ex) {
            _ex.printStackTrace();
        }
        log.println(date.toString() + "  Opening connection " + String.valueOf(i) + " " +
connPool[i].toString() + ":");
    }


    public void destroy(int i)
        throws SQLException
    {
        available = false;
        runner.interrupt();
        try
        {
            runner.join(i);
        }
        catch(InterruptedException _ex) { }
        int j;
        for(long l = System.currentTimeMillis(); (j = getUseCount()) > 0 &&
System.currentTimeMillis() - l <= (long)i;)
            try
            {
                Thread.sleep(500L);
            }
            catch(InterruptedException _ex) { }


        for(int k = 0; k < currConnections; k++)
            try
            {
                connPool[k].close();
                log.println("Connection " + connPool[k].toString() + " closed.");
            }
            catch(SQLException _ex)
            {
                log.println("Cannot close connections on Destroy");
            }


        if(j > 0)
        {
```

```
            String s = "Unsafe shutdown: Had to close " + j + " active DB connections after " + i +
"ms";
            log.println(s);
            log.close();
            throw new SQLException(s);
        } else
        {
            log.close();
            return;
        }
    }


    public void destroy()
    {
        try
        {
            destroy(10000);
            return;
        }
        catch(SQLException _ex)
        {
            return;
        }
    }


    public int getUseCount()
    {
        int i = 0;
        synchronized(connStatus)
        {
            for(int j = 0; j < currConnections; j++)
                if(connStatus[j] > 0)
                    i++;


        }
        return i;
    }


    public int getSize()
    {
        return currConnections;
    }


    private Thread runner;
    private Connection connPool[];
```

```
    private int connStatus[];
    private long connLockTime[];
    private long connCreateDate[];
    private String connID[];
    private String dbDriver;
    private String dbServer;
    private String dbLogin;
    private String dbPassword;
    private String logFileString;
    private int currConnections;
    private int connLast;
    private int minConns;
    private int maxConns;
    private int maxConnMSec;
    private boolean available;
    private PrintWriter log;
    private SQLWarning currSQLWarning;
    private String pid;
}
```