PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 04/11

Um estudo sobre Fluxos de Dados e Bancos de dados Biológicos

Carlos Juliano Moura Viana
Sérgio Lifschitz
Edward Hermann Haeusler

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900
RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 04/11 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena Abril, 2011

Um estudo sobre Fluxos de Dados e Bancos de dados Biológicos

Carlos Juliano Moura Viana, Sérgio Lifschitz e Edward Hermann Haeusler

cviana@inf.puc-rio.br, sergio@inf.puc-rio.br, herman@inf.puc-rio.br

Abstract. With the advance in the genome sequencing techniques together with the arising of several repositories of biological data, computational techniques had become indispensable tools for a better characterization and understanding of the organisms in study. One way to analyze the genomes is to compare its sequences with other sequences from previously studied genomes, to define the similarities. Frequent updates in the biological data repositories led to the problem of reprocessing of the comparisons, that consists to avoid new comparisons among sequences of the study genome with already compared sequences from the repository. These comparisons are performed by biological comparasion tools, such as BLAST. This paper describes the problem, comparing it to the data flows, trying to visualize the frequent updates as a biological data flow, and addressing data streams techniques tha can be usefull to deal with the problems related to the reprocessing of the comparasions.

Keywords: Biological databases, sequence comparison, BLAST.

Resumo. Com o avanço nas técnicas de sequenciamento de genomas, juntamente com o surgimento de diversos repositórios de dados biológicos, técnicas computacionais tornaram-se indispensáveis ferramentas para uma melhor caracterização e compreensão dos organismos em estudo. Uma forma de analisar os genomas é comparar as suas sequências para determinar as sequências similares em relação a outros genomas estudados anteriormente. Atualizações frequentes nos repostiórios de dados biológicos deram origem ao problema da recomputação, que consistem em evitar recomparações entre as sequências dos genomas em estudo com as sequências dos repositórios, comparações essas efetuadas por ferramentas como o BLAST. Este trabalho apresenta o problema relacionando-o a fluxos de dados, visualizando as frequentes atualizações como um fluxo de dados biológicos, e abordando algumas técnicas em *data streams* que podem ser utilizadas para tratar os problemas relacionados a essas recomparações.

Palavras-chave: Fluxos de dados, bancos de dados biológicos, comparação de sequências, BLAST

Responsável por publicações:
Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

Sumário

Lista de figuras

1 Introdução

Os bancos de dados tradicionais tem sido utilizados em aplicações que necessitam de persistência dos dados e de execução de consultas a esses dados. Geralmente um banco de dados consiste em um conjunto de dados ou objetos que estão armazenados em algum repositório de dados, e esses dados são relativamente estáticos, ou seja, as operações de inserção, atualização e remoção são executadas menos frequentemente do que consultas a essas fontes de dados. As consultas executadas a essas fontes de dados refletem o estado atual do banco de dados.

No entanto, com o passar dos anos, uma classe emergente de aplicações tem surgido e que não se aplicam a esse paradigma de modelo e consultas. Nessas aplicações, as informações ocorrem naturalmente na forma de um fluxo (sequência) de valores. Nesse caso, os dados são melhor modelados como um fluxo transitório de dados, ao invés de relações ou dados persistentes. Exemplos dessas aplicações incluem aplicações para dados de sensores, tráfego de Internet, redes de monitoramento, aplicações financeiras, registros de transações, como por exemplo, de chamadas telefônicas e registros de *logs* da Web, entre outros.

Em todas essas aplicações, não é praticável carregar simplesmente os fluxos de dados em bancos de dados tradicionais e operar sobre esses fluxos. Os bancos de dados tradicionais, como o banco de dados relacional, não foram desenvolvidos para suportar cargas rápidas e contínuas de itens de dados. Em um modelo de fluxo de dados, itens de dados podem ser tuplas relacionais, tais como: medições de redes de computadores, registros telefônicos, páginas visitadas da Web, leituras de sensores, etc. Além da carga excessiva de itens, um banco de dados tradicional, segundo [1], não suporta diretamente consultas contínuas, que são típicas em aplicações de fluxo de dados. Na seção 2 abordaremos os principais conceitos relacionados a *fluxo de dados*.

Este trabalho descreve conceitos de *fluxo de dados* e de banco de dados biológico para tratar o problema da análise dos dados biológicos para o contexto de alta disponibilidade do banco de dados biológico não redundante NR. O trabalho está organizado da seguinte forma: Na seção 2 descreveremos os principais conceitos de *fluxo de dados* descrevendo as características de um modelo de *fluxo de dados*, relacionando algumas aplicações, citando algumas operações aplicadas a *fluxo de dados*, abordando características de modelo de dados e também citamos algumas pesquisas em desenvolvimento para *fluxo de dados*. Na seção 3 descreveremos os conceitos associados a bancos de dados biológicos, abordando o problema da análise dos dados biológicos. Na seção 4 descreveremos nossa contribuição e apresentaremos propostas de trabalhos futuros. Por fim, na seção 5 apresentamos a conclusão do trabalho.

2 Fluxo de Dados

Um *Fluxo de Dados*, ou *Data Stream* do termo em Inglês, é uma sequência de itens contínua, em tempo real e ordenada (implicitamente pelo tempo de chegada ou explicitamente por uma data) [2]. Em *data streams* não é possível controlar a ordem de chegada dos itens de dados e também nem armazenar localmente e inteiramente os dados do fluxo. Nesse modelo de fluxos, as consultas são executadas continuamente sobre um período de tempo e incrementalmente retornam novos resultados, assim que novos dados cheguem ao fluxo. Na literatura [1; 2], essas consultas são conhecidas como consultas de longa execução, contínuas e frequentes.

As características de um *fluxo de dados*, e das consultas contínuas executadas ditam um conjunto de necessidades que um possível sistema gerenciador de fluxos de dados (DSMS em inglês) deveria possuir. Dentre as necessidades podemos destacar as relacionadas abaixo:

- O DSMS deveria possuir um modelo de dados e de consultas capaz de permitir a execução de operações baseadas em tempo e ordem. Por exemplo, possibilitar a consulta sobre um determinado período (janela de execução) variável de 5 minutos;
- Possibilitar a utilização de estruturas contendo resumos dos dados, identificadas na literatura por *synopses* ou *digests*. A necessidade da utilização dessa estrutura nasce da impossibilidade da armazenagem completa do fluxo de dados. E como consequência de sua implementação, temos a possibilidade de que as respostas às consultas sejam não exatas, por causa da informação estar representada de forma resumida;
- Deve-se evitar a utilização de operadores bloqueantes em planos de consultas, por causa da necessidade desses operadores de consumirem toda a entrada de dados para posteriormente gerar produtos desses dados [2; 3; 4].
- Os algoritmos sobre fluxos devem ser restritos a fazerem apenas uma "passagem" sobre o fluxo de dados. Essa característica também é necessária devido a restrição de desempenho e armazenamento, por causa da contínua chegada dos dados em fluxos;
- O DSMS deve estar preparado para que as consultas de longa duração, submetidas ao fluxo, possam encontrar um ambiente diferente ao início de sua execução. Por exemplo, as consultas podem ser executadas sobre uma variação na taxa de chegada de dados do fluxo de dados.
- Assegurar a escalabilidade, permitindo o compartilhamento de execução de diversas consultas contínuas.

Sistemas propostos de *data stream* na literatura remetem a uma arquitetura abstrata em comum, como ilustrado na figura 1.

Nessa arquitetura, podemos observar cinco componentes principais: Monitor de entrada, repositório de dados estáticos, resumos e de dados relativos a consultas por período, repositório de consultas, um processador de consultas e um gerenciador de *buffer* de saída de dados.

Na figura 1, o componente do monitor de entrada (*Input Monitor*) regula as taxas de entrada, não permitindo a passagem de pacotes se o determinado sistema não é capaz de acompanhar o processamento das taxas. Os componentes de repositórios são divididos em:

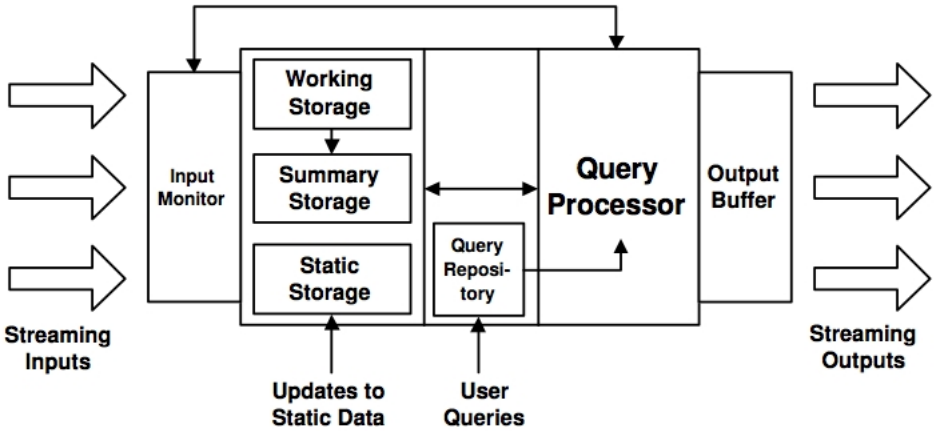

Figura 1: Arquitetura abstrata para um sistema de gerenciamento de *Data Streams*

repositório temporário de trabalho (por exemplo, para consultas restritas a um determinado tempo), repositório de resumos para fluxos com dados resumidos e um repositório de metadados (por exemplo, para dados de localização de cada fonte de dados). As consultas são registradas no repositório de consultas (*Query Repository*) e agrupadas para permitir compartilhamento de processamento. Nessa visão de arquitetura, o processador de consulta (*Query Processor*) comunica-se diretamente com o monitor de entrada para poder reotimizar os planos de consultas dependendo das taxas de dados de entrada. Por fim, os resultados ou são liberados para o usuário, na forma de fluxos, ou temporariamente armazenados em um *buffer*. Dessa forma, os usuários podem refinar as suas consultas baseados nos últimos resultados obtidos. Na seção a seguir abordaremos alguns exemplos de aplicações de fluxo de dados.

2.1 Aplicações de Fluxo de Dados

Nessa seção descreveremos três exemplos de aplicações de *fluxo de dados* e consultas que são em geral requisitadas em cada tipo de aplicação.

2.1.1 Redes de Sensores

As redes de sensores podem ser utilizadas para diversas finalidades, como por exemplos, para medir a taxa de congestionamento de uma rodovia, rastrear movimentações, monitorar sinais de vida (razões médicas) e outras. Em todos os exemplos citados, essas aplicações lidam com filtragens complexas de dados e ativação de alarmes quando detectados padrões não usuais nos dados. Nessas aplicações são realizadas operações de agregação (união) e junções. Operações de agregações e junções sobre múltiplos fluxos são necessárias para poder analisar dados de múltiplas origens. Por outro lado, a agregação sobre um único fluxo de dados pode ser utilizada para compensar falhas individuais de sensores. Nesse tipo de aplicação podemos destacar os seguintes ambientes e consultas:

- Analisar um fluxo de estatísticas de utilização de energia reportados a uma estação

de energia (agrupados por localizações da cidade) e deve-se ajustar a taxa de geração de energia caso seja necessário.

- Ativar um tipo de resposta se diversos sensores de uma mesma área emitirem dados que excedem um determinado valor limite.

2.1.2 Análise de Tráfego de Rede

Em aplicações de análise de tráfego de rede, deseja-se verificar dados de múltiplas fontes, monitorar pacotes, filtrar os pacotes e se possível, detectar padrões não usuais da rede. Nesses tipos de aplicações, estamos querendo saber sobre as condições críticas da rede, saber se temos congestionamento na rede, ou se a mesma está sofrendo algum tipo de ataque, como negação de serviço. Em aplicações desse tipo, temos o seguinte ambiente e consultas usuais:

- Determine a quantidade total de largura de banda utilizada por cada par (origem, destino), agrupando os dados pelo tipo de protocolo utilizado ou máscara de subrede.

2.1.3 Análise de *logs* de Transações

Logs de utilização de internet e registros de chamadas telefônicas são dois exemplos de aplicações para fluxos de dados. Nesses casos, os objetivos dessas aplicações consistem em identificar padrões de comportamento de compradores com o objetivo de identificar fraudes, entre outros. Nessas aplicações temos os seguintes tipos de consultas:

- Examinar os *logs* dos servidores da Web, em tempo real, e redefinir a rota de acesso dos usuários para servidores *backups* se o servidor primário estiver sobrecarregado;

- Encontrar todas as páginas da Web, de um determinado servidor, que foram acessadas nos últimos 15 minutos, com uma taxa de acesso que é pelo menos 40% maior do que a taxa média dos acessos diários.

2.2 Características de Aplicações em *Data Streams*

Em todos os exemplos das aplicações anteriores temos similaridades em relação ao modelo de dados e operações básicas realizadas (Seleção, Agregação, entre outras). Existem algumas diferenças relacionadas a características da carga de trabalho, tais como , as taxas de chegada do fluxo de dados ou a quantidade de dados históricos necessários para realizar uma análise. Baseado nas características das aplicações, Golab [2] propõe um conjunto de operações, ditas básicas, que as aplicações envolvendo fluxos de dados contém:

- Seleção: Grande parte das aplicações envolvendo fluxos de dados necessitam da execução de filtros complexos;

- Agregação "aninhada": Em aplicações que necessitam da computação de tendências nos dados, é necessário que existam operações de agregação uma sobre a outra (*nested*), por exemplo, na comparação de um valor mínimo com o valor médio da execução.

- Multiplexação e Demultiplexação: Operações similares a *group by* e *union* em consultas SQL, utilizadas em fluxos de dados para decompor e compor fluxos.
- Mineração de Fluxos: Operações como casamento de padrões, buscas por similaridades e de previsão são necessárias para o procedimento de mineração dos fluxos de dados.
- Junções: Operações desse tipo deveriam suportar a junção entre mútliplos fluxos e também entre fluxos e metadados;
- Operador "Windowed": Nesse caso, os tipos de consultas feitas a fluxos de dados deveriam permitir a aplicação de um operador para restringir os resultados dentro de uma possível "janela" de análise, por exemplo, dados das últimas 24 horas ou referente a última centena de pacotes.

A implementação desses operadores necessitam de características específicas para serem incluídas em modelos de dados e linguagens de consultas para fluxos de dados. Na seção seguinte descreveremos o modelo de dados e de consultas para um *fluxo de dados*.

2.3 Modelo de dados

Para possibilitar a implementação das operações relacionadas como básicas na maior parte das aplicações de fluxos de dados, são necessárias propostas de modelos de dados que suportem esses operadores.

Em um modelo para fluxo de dados, alguns ou todos os dados de entrada que estão sendo analisados ou operados, não estão disponíveis para acesso aleatório a partir do disco rígido ou em memória. Ao invés disso, esses dados chegam para análise, na forma de um ou mais fluxos de dados contínuos. Dessa forma, o modelo de *fluxo de dados* difere-se do modelo convencional relacional de armazenamento em algumas formas:

- Os dados chegam no fluxo de forma online, ou seja não estão "parados", sem atualização. Estão constantemente sendo inseridos novos items no fluxo;
- O determinado sistema de controle do fluxo não possui controle sobre a ordem de chegada dos items para serem processados;
- Os fluxos de dados são geralmente ilimitados em tamanho;
- Em um modelo para fluxo de dados, um elemente é discartado ou arquivado após o seu processamento. O dado não deve ser retomado, a menos que tenha sido explicitamente armazenado em memória.

Dado essas características de um modelo de fluxo de dados, segundo [3], um fluxo de dados pode ser modelado como uma sequência de listas de elementos. Nesse caso, itens de um fluxo de dados podem ser tuplas relacionais ou instanciações de objetos. Em modelos baseados em relações, como exemplo o STREAM [5], os itens de dados são registros transitórios armazenados em relações virtuais. Em modelos baseados em objetos, como exemplo o COUGAR [6], os itens são modelados como tipos de dados hierárquicos com métodos associados. No entanto, devido a evolução do dados e a imprecisão sobre a

construção de modelos que atuem sobre os dados inteiros de um fluxo de dados, é mais natural imaginar que os dados de um passado mais recente é mais significativo do que os dados de um passado mais distante. Desta forma, em algumas vezes, as aplicações em fluxos estão interessadas nas partes ou fluxos em um determinado período de tempo, originando os modelos baseados em "janelas", que podem ser classificados segundo os critérios abaixo:

- Direção de movimentação dos pontos de terminação da janela (*endpoints*): Nesse critério, dois pontos finais fixos definem uma janela fixa. Dois pontos finais deslizantes, tanto para frente quanto para trás, definem janelas deslizantes. Quando temos um ponto final fixo e um outro em movimento, temos uma janela de marcação.

- Físicas ou lógicas: Nesse critério, as janelas denominadas físicas são definidas em termos de intervalos de tempo, enquanto que as lógicas são definidas em termos de número de tuplas ou registros.

- Intervalos de Atualização: Para esse critério, as janelas de análise são definidas baseadas nas taxas de atualização de registros do fluxo. Nesses tipos de modelos, acredita-se que as janelas mais recentes são mais importantes do que as anteriores. Desta forma, o peso (importância) para os dados de uma janela decresce exponencialmente com o passar do tempo.

Dadas as necessidades de um modelo de dados para fluxos de dados, deve-se tratar também as consultas para possibilitar retornar resultados levando em consideração as modelagens dos itens de dados como listas de itens.

2.4 Consultas em *Fluxo de Dados*

Nesta seção descreveremos algumas diferenças entre consultas em *fluxo de dados* e bancos de dados tradicionais. Segundo [7], consultas em *fluxo de dados* possuem muito em comum com consultas em bancos de dados tradicionais. No entanto, existem algumas distinções entre elas. Um distinção é referente entre consultas denominadas *One-time* e consultas contínuas. Uma definição de consultas contínuas pode ser encontrada em [1]. O primeiro tipo de consulta inclui as classes de consultas tradicionais em sistemas gerenciadores de bancos de dados (SGBDs). Nessa classe, as consultas são avaliadas uma vez sobre uma determinada *snapshot* do repositório de dados, e as respostas são retornadas ao usuário. Por outro lado, as consultas contínuas são avaliadas continuamente tão logo, os dados no fluxo de dados continuem chegando ao sistema. As respostas a esses tipos de consultas são produzidas com o passar do tempo, sempre refletindo o fluxo dos dados visto até o momento.

No trabalho de Golab [3] são apresentados três paradigmas de linguagens para fluxos de dados: baseadas em relações (baseado no SQL do modelo relacional), baseadas em objetos e procedimentais.

2.4.1 Linguagens baseadas em relações

Exemplos de linguagens baseadas em relações são CQL [8], StreaQuel [9] e AQuery [10]. Cada uma dessas linguagens foram baseadas na sintaxe SQL e suportam opeações de ordenação e o retorno de "janela" de valores. A CQL foi utilizada no sistema *STREAM*. A

linguagem *StreaQuel* também proporciona a utilização de operadores do operação de resultados em "janelas", e foi utilizada no sistema *TelegraphCQ*. A linguagem *AQuery* consiste de uma algebra para consultas e de uma linguagem baseada em SQL para ordenação dos dados.

2.4.2 Linguagens baseadas em objetos

Uma linguagem proposta baseada em objetos possui uma sintaxe baseada também em SQL, mas possui um cláusula *$every()* que indica a frequência de re-execução das consultas. Maiores detalhes sobre esse tipo de linguagem pode ser encontrado em [3].

2.4.3 Linguagens procedimentais

Uma alternativa para as linguagens declarativas é possibilitar o usuário especificar o fluxo de dados a ser analisado. Na linguagem procedimental especificada no sistema Aurora [11], os usuários constróem planos de consultas via uma interface gráfica, arrastando os operadores de consultas (representados por caixas) e juntando-os com arcos dirigidos para especificar o fluxo de dados. O sistema pode posteriormente otimizar esses planos. Maiores detalhes sobre os tipos de linguagens abordados nesse trabalho podem ser vistos na literatura, em especial em [3; 7].

2.5 Pesquisa em *Fluxo de Dados*

A área de pesquisa em *Fluxo de Dados* (*Data Streams*) é um pouco recente. Segundo [12], possui 10 anos e desde o aparecimento dos primeiros artigos, tem recebido uma atenção cada vez mais crescente, devido as inovações tenológicas que tem facilitado a criação e manutenção desses dados. Existem diversos tópicos de pesquisa nessa área, em especial podemos destacar:

- *Clusterização de Data Streams*: Essa área envolve o desenvolvimento de técnicas para agrupar fluxos de dados. O livro de [12] traz abordagens para micro clusterização e para classificações de fluxos de dados. A *clusterização* de textos e fluxos de dados categóricos é um sub-problema da *Clusterização*. Embora os problemas de clusterização de textos e dados categóricos já foram abordados na literatura de uma forma *offline* [13; 14; 15], os métodos desenvolvidos não são facilmente extensíveis para o mesmo problema em fluxos de dados, por causa de alguns desafios computacionais envolvidos, tais como o grande volume de dados de entrada no fluxo. Aggarwal e Yu [16] descreve uma abordagem para esse problema.

- Técnicas de classificação de fluxos;

- Processamento e otimização de consultas contínuas;

- Definição para padrão de linguagem de fluxos: em especial, podemos citar o trabalho [17], que traz uma proposta de padronização de uma linguagem para fluxos de dados.

- Mineração de padrões frequentes: Aggarwal [18] traz uma abordagem para o problema de indentificação de padrões estruturais frequentes em fluxos de dados que

devido suas intensas interações, são modelados como um fluxo de arestas entre os nós de dados definidos para o problema em específico.

- Identificação de mudanças em fluxos de dados; entre outros. Willie [19] descreve um algoritmo para detectar mudanças em fluxos de dados, além de adotar um teste estatístico para fazer uma descrição quantitativa das mudanças detectadas.

No presente trabalho abordamos alguns conceitos principais envolvendo *data streams*, no entanto, existem diversos trabalhos por serem ainda desenvolvidos, e por consequência a especificação de padrões e algoritmos estão surgindo em artigos e livros. Em especial podemos citar os trabalhos em [17; 20; 21] entre outros. Na seção seguinte abordaremos os banco de dados biológicos.

3 Bancos de Dados Biológicos

Com os avanços da área da Biologia Molecular, em especial com o *boom* no processo de sequênciamento de DNA, surgiram grandes quantidades de bancos de dados biológicos, com um crescente volume de dados, que precisam ser analisados e interpretados.

Os bancos de dados biológicos contém dados sobre sequências (cadeias de caracteres) de DNA, RNA e proteínas, estruturas moleculares, como por exemplo, o posicionamento 3D dos átomos de uma determinada molécula, e também sobre anotações diversas sobre as sequências.

Com o aumento do processo de sequenciamento, diversos bancos de dados surgiram. Entre os principas bancos de dados ou repositórios, podemos citar:

- GenBank: Repositório de sequências do NIH (*National Institute of Health*), no *National Center for Biotechnology Information* - NCBI [22]. Centro dos USA que concentra os dados referentes a pesquisas em bioinformática.
- EMBL: Repositório de sequências de nucleotídeos e anotações. O repositório é conhecido por EMBL-Bank [23]. Esse repositório faz parte do instituto Europeu de Bioinformática - EBI [24].
- Protein Data Bank - PDB [25]: Repositório de dados sobre estruturas 3D de moléculas de proteínas. Esse repositório é do RCSB - (*Research Collaboratory for Structural Bioinformatics*) - *http://www.rcsb.org/pdb/*.

Nesses repositórios de dados estão armazenados diversas informações, como mencionado anteriormente. As informações sobre sequências estão armazenadas como cadeias de caracteres em arquivos texto. Existem diversos formatos para armazenagem dessas informações. Em especial, podemos citar os formatos: FASTA [26; 27; 28], *GenBank flat file* [29], XML, entre outros formatos proprietários de cada repositório (GenBank, EMBL e PDB).

3.1 Análise de Dados Biológicos

Com o surgimento dos repositórios de dados, surgiu também a necessidade de analisá-los e interpretá-los. Uma das formas mais básicas para analisar sequências de caracteres é realizar comparações entre elas. No contexto de sequências biológicas, a comparação entre sequências é realizada através do alinhamento entre elas. O **alinhamento** de duas sequências é uma forma de posicionar uma sequência sobre a outra, acrescentando espaços nas duas sequências, para que ambas fiquem do mesmo tamanho, com o objetivo de evidenciar a correspondência entre símbolos ou subcadeias similares das sequências, ou seja, para evidenciar quão similares elas são.

Uma das ferramentas mais conhecidas na literatura para realizar a comparação de sequências é o BLAST [30; 31]. BLAST é uma heurística para alinhamentos locais. Encontra regiões de similaridades entre as sequências. Essa ferramenta tem sido utilizada com vários objetivos, entre eles, para inferir relacionamentos funcionais e evolucionários entre sequências, e também para identificar membros de famílias de proteínas.

Quando surgem novas sequências de genomas recentemente sequenciados, são necessárias análises para extrair determinadas características desse genoma. Para a realização desse processo, utiliza-se a comparação entre as sequências do novo genoma com sequências que

se tem conhecimento prévio de informações. Nesse caso, costuma-se utilizar o repositório de dados de sequências não redundantes do NCBI, denominado por NR (*Non-Redundant*) [32; 33] e utiliza-se a ferramenta BLAST para realizar as comparações.

O repositório NR contém sequências de proteínas de outros repositórios: GenPet, SwissProt, PIR, PDB e NCBI RefSeq. Os conjuntos de dados correspondem a cerca de 6,4 GB de arquivos de sequências compactadas. O NCBI realiza atualizações diárias para essa fonte de dados, mas também mantém versões mensais (denominados por *releases*).

O procedimento de comparação, mesmo utilizando a ferramenta BLAST, não é muito rápido, demorando algumas horas para gerar os relatórios de saída. Dessa forma, com o aumento dos grandes volumes de dados biológicos, devidos aos esforços no sequenciamento de novos genomas e também pelos dados oriundos dos processos de expressão gênica e de outras técnicas, alguns desafios foram surgindo. Entre alguns desafios podemos destacar os seguintes:

- Como armazenar as sequências? Podemos utilizá-las em ferramentas de comparação de forma compactada no discos rídigos das máquinas? Qual o custo que esse tipo de procedimento acrescenta a computação final?
- Como serão feitas as comparações entre as sequências de novos genomas contra as sequências dos genomas que se tem conhecimento, de uma forma eficiente? Ou seja, temos que executar, por exemplo, a ferramenta BLAST para todos os dados de uma nova atualização de uma versão dos arquivos do repositório NR? ou simplesmente sobre apenas os novos dados atualizados?
- Podemos considerar os bancos de referência, como o NR, como fontes de fluxos de dados e utilizar algoritmos específicos para otimizar as comparações entre as sequências?

3.2 *Fluxo de Dados* e Análise de BD biológicos

Esta seção traz alguns relacionamentos entre os problemas de análise de dados biológicos descritos quando temos um aumento no volume dos dados biológicos, com alguns trabalhos referentes à *data streams*. Para tentar encontrar respostas as perguntas, descritas no final da subseção 3.1, procuramos soluções particulares da área de *data streams* que possam ser aplicadas a bancos de dados biológicos, com a finalidade de tratar melhor o problema da recomputação das comparações entre sequências para cada atualização de uma fonte de dados.

Nesse caso, estamos interessados nas técnicas de mineração de fluxos, em especial nas técnicas de busca de similaridades, busca de padrões, de clusterização e classificação, para que sejam utilizadas, com o objetivo de identificar, de alguma forma, quais os dados do fluxo de dados (conjunto de sequências nesse caso) foram alterados, fazer particionamento entre os fluxos e classificá-los, a fim de podermos comparar somente subconjuntos de sequências de interesse.

Um outro interesse está nas técnicas associadas as consultas restritas a "janelas" de respostas. Nesse caso, estamos interessados em apenas utilizar as sequências mais atualizadas do fluxo, com o objetivo de utilizá-las para novas comparações com o conjunto de dados em questão, tendo-se em vista que já tenhamos os resultados referentes as comparações anteriores. Nesse caso, evitaríamos que para cada alteração da fonte de dados de

referência, refletisse em uma recomparação desnecessária com todos os dados anteriormente comparados.

Para o problema da classificação dos fluxos de anotações e sequências biológicas, podemos citar o trabalho de Aggarwal [20], onde é realizada uma classificação em demanda dos fluxos de dados. Nesse trabalho os autores tratam o problema da classificação dos fluxos de dados do ponto de vista de uma abordagem dinâmica, na qual fluxos simultaneos de treinamento e testes são utilizados para classificar dinamicamente os conjuntos de dados dos fluxos. Nesse caso, os autores propõem a utilização de um conjunto de dados apropriados e selecionados dinamicamente a partir de uma "janela" de resultados do fluxo, como sendo um conjunto de dados para a construção do classificador.

No caso da clusterização, podemos citar as abordagens das técnicas descritas no livro [12]. Em especial, citamos a abordagem da técnica de micro-clusterização, onde consiste na captura de resumos de informações sobre a natureza do fluxo de dados. Nesse caso, a informação resumida é definida pelas estruturas:

- *Micro-clusters*: Essa estrutura mantém informações estatísticas sobre a localidade dos dados em termos de *micro-clusters*.

- *Pyramidal Time Frame*: Os *micro-clusters* são armazenados como imagens ou fotos (*snapshots*) no tempo, os quais seguem um padrão piramidal. Esse padrão provê um limiar efetivo entre as necessidades de armazenamento e a habilidade de obter estatísticas resumidas a partir de diferentes partes do tempo.

Essas estruturas são utilizadas na fase de clusterização da metodologia, onde temos uma seleção de dados estatísticos do fluxo. O objetivo principal dessa fase é manter estatísticas em um nível suficiente de granularidade espacial e temporal, para que possa ser utilizado por outros componentes de macro-clusterização, tanto quanto, pelos components de análise de evolução dos dados no fluxo.

Para a clusterização de dados de texto e categóricos em fluxos de dados, citamos a abordagem de Aggarwal [16]. Neste trabalho, os autores apontam que determinadas aplicações podem frequentemente exibir uma localidade temporal que não é levada em conta pela maioria dos algoritmos de processamento em lotes. Nesse caso, em um ambiente de fluxo de dados em evolução, a evolução contínua de determinados clusters traz a necessidade de sermos capazes de rapidamente identificar novos clusters nos dados. Por outro lado, enquanto o "processo de clusterização" necessita estar sendo executado continuamente e *online*, também é importante sermos capazes de possibilitar aos usuários analisar os clusters de uma forma *offline*, permitindo que possam "clusterizar" os dados baseado em parâmetros temporais, tais como a escolha dos dados em uma determinada janela de busca, ou por um conjunto de clusters.

Os autores propõem a construção de um *framework* para escolher dados resumidos, de forma estatística, que serão armazenados em determinados intervalos regulares. Nesse trabalho definem o conceito de *cluster droplets*, que devem fornecer uma representação estatística condensada dos dados analisados. Esses *clusters droplets* podem ser vistos como um resumo intermediário dos dados que podem ser utilizados para uma variedade de propósitos. Os autores utilizam esses dados resumidos na forma compactada.

Um trabalho interessante nesta área de compactação de dados de *fluxo de dados* é o trabalho de [34], onde é apresentado o modelo de compressão não linear para *fluxo de*

dados, denominado por ECM-DS. Segundo os autores, o modelo apresenta soluções efetivas e eficientes para o objetivo de construir metodologias de descoberta de conhecimento sobre *fluxo de dados*, implementadas sobre algoritmos de mineração de dados em *fluxo de dados*, com ênfase, em particular, ao suporte de consultas OLAP. Nesse modelo ECM-DS, o suporte a consultas OLAP, em *fluxo de dados*, têm sido descritas por meios de um conjunto de metodologias de compressão de *fluxo de dados*, onde o conhecimento sobre o fluxo está integrado com e correlacionado ao conhecimento relacionado aos eventos passados, que são considerados críticos para o cenário de uma aplicação OLAP.

Apesar das consultas a base de dados NR não serem OLAP, mesmo assim, esses dados poderiam ser organizados para poder responder consultas deste tipo. Neste caso, como estamos pretendendo modelar as atualizações no banco NR como um fluxo de dados, e como esses fluxos também podem ser multi-dimencionais e multi-níveis devido a própria natureza dos processos que geram esses dados (projetos de sequenciamento de DNA, de *transcriptomas* a partir de *short reads*, *microarrays* de expressão gênica, entre outros), talvez possamos utilizar o modelo não linear de compressão sobre fluxos de dados, desenvolvidas por Cuzzocrea e colegas [34] no processo de geração dos dados resumidos e, em determinados intervalos regulares possam ser armazenados, para que seja possível analisar melhor os dados do fluxo, e assim possibilitando obter uma melhor compreensão do domínio em estudo.

No domínio da Biologia Molecular, poderíamos estar interessados em formar clusters ou famílias de sequências de proteínas, a fim de eleger "representantes" dessas famílias para que possam ser utilizados primariamente como um filtro para futuras comparações entre sequências. Desta forma, ao invés de compararmos as sequências dos novos organismos com todo o banco de dados de referência, faríamos as comparações primariamente com as sequências representantes e posteriormente com as sequências que compõem esse grupo de representantes. Além disso, poderíamos ter usuários querendo agrupar as sequências de proteínas sob determinados parâmetros, como *score, E-value*, entre outros parâmetros disponibilizados por ferramentas de comparação de sequências, enquanto um determinado banco de dados, por exemplo NR, está sendo atualizado com novas sequências originadas de processos de sequenciamento de novos organismos. Assim, como o framework armazena determinadas informações estatísticas do fluxo de dados, poderíamos utilizar essas informações juntamente com outros valores estatíscios resultantes de comparações entre sequências, *E-value*, *score*, % de alinhamento entre as sequências, entre outros, para auxiliar no procedimento de construção das famílias de sequências (clusters).

Para o problema da detecção de mudanças em *fluxo de dados* podemos citar os trabalho de [19]. Neste trabalho os autores propõem um algoritmo para detecção de mudanças juntamente com a utilização de um teste estatístico para fazer uma descrição quantitativa da mudança que foi detectada. Os autores descrevem ainda que além de se manter um modelo dos dados atualizado, é imperativo avisar ao usuário quando uma mudança for detectada, afim de permitir aos usuários compreender a natureza da mudança para que assim, possam tomar determinadas ações para esta mudança e no menor tempo possível.

Fazendo o relacionamento com a comparação de sequências em uma base de dados, NR por exemplo, poderíamos ter um conjunto de sequências (referências) que seriam de referência para futuras comparações, quando tivermos mais sequências de genomas sequenciados. Neste sentido, como estamos considerando a base NR como um fluxo de sequências, a detecção de mudança ocorreria quando uma determinada sequência ou conjunto de sequências

fossem tão significativas para o fluxo (base NR) que poderiam alterar as próprias sequências de referência. Nesse caso, os usuários poderiam ser avisados para poderem avaliar especificamente se essas novas sequências são realmente significativas para justificar uma atualização do modelo de sequências de referência dos *clusters*.

No problema da detecção de mudanças em *fluxo de dados*, para podermos avaliar se um conjunto de sequências foi alterada (novas acrescentadas ou mesmo detectar mudanças em algumas bases das sequências já comparadas), podemos modelar este problema como como um fluxo sequencial de arestas (um fluxo de grafo) entre as sequências anteriormente comparadas, onde as arestas representam um relacionamento entre as sequências avaliadas por uma determinada similaridade. Nessa modelagem, os fluxos são relativamente grandes devido não somente a quantidade de dados sendo trafegados, mas também pela interação entre os conjuntos de dados. No domínio da Bioinformática, esse fluxo pode conter as interações entre as sequências comparadas, e também dados relativos a partes das sequências relacionadas com dados de anotações, entre outros dados. Nesse tipo de modelagem estamos interessados em determinar frequentes estruturas no fluxo de grafos, em geral, em determinar padrões frequentes e densos sobre as interações entre os dados (sequências).

Aggarwal e colegas [18] propõem um modelo para mineração de padrões densos e também algoritmos probabilísticos para determinar padrões estruturais, interações entre os dados, de forma efetiva e eficiente. Os autores citam que o modelo probabilístico proposto é utilizado para criar um resumo sobre o fluxo de grafo, o pode ser utilizado posteriormente por outros algoritmos de mineração de dados. Em específico, o modelo probabilístico que os autores apresentam é baseado em co-ocorrência de nós e na densidade das arestas entre os nós. Segundo os autores, essa abordagem é útil para problemas de determinados domínios, tais como de rede sociais, onde existe geralmente um número grande de cominações de nós e relacionamentos para serem considerados padrões relevantes. Neste caso, os autores assumem que o grafo de fluxo de arestas (fluxo de grafo) é concebido geralmente na forma de um conjunto de fluxo de arestas.

No domíno da Biologia Molecular, podemos ter essa mesma abordagem aplicada aos resultados das comparações entre sequências de um *fluxo de dados*. Neste caso, as interações poderiam ser os relacionamentos oriundos das classificações ou anotações dadas às sequências, juntamente com os dados de publicações, dados de relacionamentos com processos celulares (quais sequências participam de um mesmo processo celular), dados de domínios de proteínas, dados de Enzimas, entre outros dados. Assim, poderíamos ter um objeto biológico que contém nós correspondentes a sequênciase genomas, e as arestas correspondendo a relacionamentos de similaridade entre sequências de um mesmo genoma, ou entre genomas distintos e também relacionamentos para um conjunto de anotações no fluxo. Desta forma, poderíamos utilizar ou estender o modelo probabilístico proposto, com o objetivo de ser capaz de minerar outras estruturas padrões que possamos encontrar no fluxo de grafo.

Para o problema associado as técnicas de janelas, podemos citar o trabalho de [21]. Nesse trabalho, David e colegas apresentam novos métodos também para a detecção e estimação de mudanças em fluxos de dados. A técnica apresentada é não paramétrica e também não necessita de dados a priori sobre a natureza da distribuição que gerou os dados do fluxo. No entanto, a técnica assume que os pontos do fluxo são gerados independentemente, o que não acontece com os bancos de dados biológicos, como descrevemos anteriormente. Nesse caso, podemos ter conjuntos de dados originados a partir de outros,

como no caso dos dados de proteínas, que podem originar outras proteínas. No entanto, ainda assim consideramos o trabalho como uma fonte de idéias para identificar problemas de mudança em fluxos de dados biológicos.

4 Contribuições e Trabalhos Futuros

Uma das contribuições proporcionadas por este trabalho foi pesquisar algumas abordagens dos problemas relacionados a análise de bases de dados biológicas para a visão de *fluxo de dados*, descrevendo os principais conceitos relacionados a área de *data streams*, abordando os principais operadores executados em algumas aplicações de *data stream*, descrevendo as caractéristicas de um modelo de dados para *data stream*, e os tipos de paradimas para construção de linguagens de consultas em fluxos de dados, além de fazer referências as principais pesquisas que estão sendo desenvolvidas sobre *data streams*. Além dos conceitos referentes a *fluxo de dados*, abordamos também o problema da análise (comparação) de sequências biológicas no contexto da recomputação das comparações entre as sequências. Neste caso, o aumento do volume dos repositórios de dados biológicos pode influenciar negativamente nas comparações entre as sequências, podendo trazer dificuldades para obter as sequências similares aos genomas em estudo.

No presente trabalho relacionamos também esse problemas da análise de sequências biológicas com algumas das áreas de pesquisa em *data streams*, descrevendo suscintamente alguns trabalhos da literatura de *data streams* que poderiam ser utilizados na investigação de soluções para o problema da recomputação das sequências similares de um banco de dados biológicos em relação as sequências de estudo em questão.

Como sugestão de trabalho futuro que poderia ser desenvolvido é uma descrição mais formal do problema em relação aos trabalhos de pesquisa em *data streams*, além de um aprofundamento nesses trabalhos de pesquisa, e também além de utilizar alguns *frameworks* e metodologias que foram descritos anteriormente. Em específico, poderíamos utilizar o framework descrito por [16], para escolher os "resumos" de dados estatísticos sobre as comparações entre as sequências ou sobre as próprias sequências que chegam ao fluxo. Além da escolha dos "resumos", poderíamos talvez utilizar o modelo de compressão não linear desenvolvido em [34] para comprimir esses dados resumidos, a fim de que possam ser armazenados em intervalos regulares e assim possam ser utilizados na melhor compreensão do domínio em estudo.

Além do framework, sugerimos a implementação e testes do algoritmo de detecção de mudanças em fluxo proposto por Willie e Dash [19], a fim de propor ou utilizar um outro método estatístico de teste para considerar não somente mudança nos dados, o que pode ocorrer devido ao sequenciamento de novas sequências, mas sim, na mudança de "estado" do fluxo, na qual poderíamos considerar uma mudança, dita muitas vezes significativa, quando esta for capaz de mudar a estrutura ou conjunto de sequências que deram origem aos *clusters* das sequências representantes geradas previamente. Desta forma, poderíamos montar um ambiente de testes onde tivéssemos alguns grupos ou famílias de sequências previamente "clusterizadas" e utilizássemos algumas sequências de outros organismos similares a fim de testarmos se o "novo" algoritmo seria capaz de detectar que houve uma mudança no fluxo, pois em teoria, as sequências similares deveriam ser capazes de alterar a conformação dos "clusters" das outras sequências previamente agrupadas, no sentido da construção da sequência representativa do *cluster*.

Por fim, poderia também ser realizada um levantamento bibliográfico ainda maior sobre outros trabalhos da literatura de *data streams*, buscando outras técnicas, abordagens e metodologias que possam ser utilizadas para tratar o problema em questão.

5 Conclusão

Com o desenvolvimento da área da Biologia Molecular, em particular, com o desenvolvimento de novas técnicas para o sequenciamento de DNA, surgiram diversos bancos de dados contendo dados de sequências e anotações. O surgimento desse grande volume, deu origem também a necessidade de analisá-los e interpretá-los. Uma forma de análise é comparar as sequências a fim de obter as que são similares com o objetivo de predizer as suas funcionalidades. No entanto, com as diárias atualizações desses repositórios surge também um outro problema, que consiste em evitar a recomputação das comparações entre as sequências previamente comparadas, a fim de se evitar o desperdício de processamento e tempo.

Nesse trabalho abordamos incialmente o problema, visando as técnicas de *data streams*. Nesse caso, visualizamos um repositório de atualizações diárias como sendo um fluxo de dados, e dessa maneira, podemos tentar utilizar algumas das técnicas previamente estudadas na literatura para: 1) clusterizar as sequências e anotações a fim de que representem um grupo de sequências (um resumo de dados em fluxo de dados); 2) detectar mudanças nos fluxos (sequências) para evitar a recomparação entre sequências previamente comparadas; 3) classificar os dados desses fluxos, com a finalidade de especificar apenas tipos de sequência do conjunto de sequências para serem comparadas.

O presente trabalho contextualiza os principais conceitos de *data streams* com alguns dos problemas relacionados a análise de sequências biológicas em ambientes com altas atualizações de sequências. Após a apresentação dos conceitos de *data streams* e de levantarmos o problema da análise de sequências biológicas em crescentes repositórios de dados, relacionamos algumas técnicas de *fluxo de dados* com esse problema e descrevemos suscintamente algumas metodologias e abordagens encontradas na literatura. Por fim, concluímos sugerindo uma descrição mais formal do problema da análise de sequências biológicas, bem como um levantamento de outras técnicas de *fluxo de dados* que possam ser utilizadas para esse desafio.

Referências Bibliográficas

[1] Douglas Terry, David Goldberg, David Nichols, and Brian Oki. Continuous queries over append-only databases. *SIGMOD Rec.*, 21:321–330, June 1992. ISSN 0163-5808. doi: http://doi.acm.org/10.1145/141484.130333. URL `http://doi.acm.org/10.1145/141484.130333`.

[2] Lukasz Golab and M. Tamer Özsu. Data stream management issues – a survey. Technical Report CS-2003-08, University of Waterloo, Canada, April 2003.

[3] Lukasz Golab and M. Tamer Özsu. Issues in data stream management. *SIGMOD Rec.*, 32:5–14, June 2003. ISSN 0163-5808. doi: http://doi.acm.org/10.1145/776985.776986. URL `http://doi.acm.org/10.1145/776985.776986`.

[4] YAN-NEI LAW, HAIXUN WANG, and CARLO ZANIOLO. Relational Languages and Data Models for Continuous Queries on Sequences and Data Streams. *ACM Transactions on Embedded Computing Systems*, 36(3), March 2011.

[5] Rajeev Motwani, Jennifer Widom, Arvind Arasu, Brian Babcock, Shivnath Babu, Mayur Datar, Gurmeet Manku, Chris Olston, Justin Rosenstein, and Rohit Varma. Query processing, resource management, and approximation ina data stream management system. Technical Report 2002-41, Stanford InfoLab, 2002. URL `http://ilpubs.stanford.edu:8090/549/`.

[6] Philippe Bonnet, Johannes Gehrke, and Praveen Seshadri. Towards sensor database systems. In *Proceedings of the Second International Conference on Mobile Data Management*, MDM '01, pages 3–14, London, UK, 2001. Springer-Verlag. ISBN 3-540-41454-1. URL `http://portal.acm.org/citation.cfm?id=648058.746944`.

[7] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '02, pages 1–16, New York, NY, USA, 2002. ACM. ISBN 1-58113-507-6. doi: http://doi.acm.org/10.1145/543613.543615. URL `http://doi.acm.org/10.1145/543613.543615`.

[8] Arvind Arasu, Shivnath Babu, and Jennifer Widom. An abstract semantics and concrete language for continuous queries over streams and relations. Technical Report 2002-57, Stanford InfoLab, 2002. URL `http://ilpubs.stanford.edu:8090/563/`. A short version of this technical report appears in the proceedings of the 9th International Conference on Data Base Programming Languages (DBPL 2003). The most recent version this technical report is available on this publications server as technical report number 2003-67, titled "The CQL Continuous Query Language: Semantic Foundations and Query Execution", at http://dbpubs.stanford.edu/pub/2003-67.

[9] Sirish Chandrasekaran, Owen Cooper, Amol Deshpande, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong, Sailesh Krishnamurthy, Samuel R. Madden, Fred Reiss, and Mehul A. Shah. Telegraphcq: continuous dataflow processing. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*,

SIGMOD '03, pages 668–668, New York, NY, USA, 2003. ACM. ISBN 1-58113-634-X. doi: http://doi.acm.org/10.1145/872757.872857. URL http://doi.acm.org/10.1145/872757.872857.

[10] Alberto Lerner and Dennis Shasha. Aquery: query language for ordered data, optimization techniques, and experiments. In *Proceedings of the 29th international conference on Very large data bases - Volume 29*, VLDB '2003, pages 345–356. VLDB Endowment, 2003. ISBN 0-12-722442-4. URL http://portal.acm.org/citation.cfm?id=1315451.1315482.

[11] Don Carney, Uğur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Greg Seidman, Michael Stonebraker, Nesime Tatbul, and Stan Zdonik. Monitoring streams: a new class of data management applications. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 215–226. VLDB Endowment, 2002. URL http://portal.acm.org/citation.cfm?id=1287369.1287389.

[12] Charu C. Aggarwal. *Data Streams: Models and Algorithms (Advances in Database Systems)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387287590.

[13] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scatter/Gather: a cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '92, pages 318–329, New York, NY, USA, 1992. ACM. ISBN 0-89791-523-2. doi: http://doi.acm.org/10.1145/133160.133214. URL http://doi.acm.org/10.1145/133160.133214.

[14] S. Guha, R. Rastogi, and K. Shim. ROCK: a robust clustering algorithm for categorical attributes. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 512 –521, March 1999. doi: 10.1109/ICDE.1999.754967.

[15] David Gibson, Jon Kleinberg, and Prabhakar Raghavan. Clustering categorical data: an approach based on dynamical systems. *The VLDB Journal*, 8:222–236, February 2000. ISSN 1066-8888. doi: http://dx.doi.org/10.1007/s007780050005. URL http://dx.doi.org/10.1007/s007780050005.

[16] Charu C. Aggarwal and Philip S. Yu. On clustering massive text and categorical data streams. *Knowl. Inf. Syst.*, 24:171–196, August 2010. ISSN 0219-1377. doi: http://dx.doi.org/10.1007/s10115-009-0241-z. URL http://dx.doi.org/10.1007/s10115-009-0241-z.

[17] Namit Jain, Shailendra Mishra, Anand Srinivasan, Johannes Gehrke, Jennifer Widom, Hari Balakrishnan, Uğur Çetintemel, Mitch Cherniack, Richard Tibbetts, and Stan Zdonik. Towards a streaming sql standard. *Proc. VLDB Endow.*, 1:1379–1390, August 2008. ISSN 2150-8097. doi: http://dx.doi.org/10.1145/1454159.1454179. URL http://dx.doi.org/10.1145/1454159.1454179.

[18] Charu C. Aggarwal, Yao Li, Philip S. Yu, and Ruoming Jin. On dense pattern mining in graph streams. *Proc. VLDB Endow.*, 3:975–984, September 2010. ISSN 2150-8097. URL http://portal.acm.org/citation.cfm?id=1920841.1920964.

[19] Willie Ng and Manoranjan Dash. A test paradigm for detecting changes in transactional data streams. In Jayant Haritsa, Ramamohanarao Kotagiri, and Vikram Pudi, editors, *Database Systems for Advanced Applications*, volume 4947 of *Lecture Notes in Computer Science*, pages 204–219. Springer Berlin / Heidelberg, 2008. URL `http://dx.doi.org/10.1007/978-3-540-78568-2_17`. 10.1007/978-3-540-78568-2_17.

[20] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. On demand classification of data streams. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 503–508, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1. doi: http://doi.acm.org/10.1145/1014052.1014110. URL `http://doi.acm.org/10.1145/1014052.1014110`.

[21] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB '04, pages 180–191. VLDB Endowment, 2004. ISBN 0-12-088469-0. URL `http://portal.acm.org/citation.cfm?id=1316689.1316707`.

[22] National Center for Biotechnology Information - NCBI, 2010.

[23] Rasko Leinonen, Ruth Akhtar, Ewan Birney, Lawrence Bower, Ana Cerdeno-Tárraga, Ying Cheng, Iain Cleland, Nadeem Faruque, Neil Goodgame, Richard Gibson, Gemma Hoad, Mikyung Jang, Nima Pakseresht, Sheila Plaister, Rajesh Radhakrishnan, Kethi Reddy, Siamak Sobhany, Petra Ten Hoopen, Robert Vaughan, Vadim Zalunin, and Guy Cochrane. The european nucleotide archive. *Nucleic Acids Research*, 39(suppl 1): D28–D31, 2011. doi: 10.1093/nar/gkq967. URL `http://nar.oxfordjournals.org/content/39/suppl_1/D28.abstract`.

[24] European Bioinformatics Institute - EMBL-EBI. Disponível em Disponível em URL: `http://www.ebi.ac.uk/`. Acesso em 10 de Janeiro de 2011.

[25] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000. doi: 10.1093/nar/28.1.235. URL `http://nar.oxfordjournals.org/content/28/1/235.abstract`.

[26] FASTA format description. Disponível em Disponível em URL `http://www.ncbi.nlm.nih.gov/BLAST/fasta.shtml`. Acesso em 2 de Janeiro de 2011., .

[27] Help - About Nucleotide And Protein Sequence Formats. Disponível em Disponível em URL `http://www.ebi.ac.uk/help/formats.html#fasta`. Acesso em 20 de Janeiro de 2011., .

[28] FASTA format. Disponível em Disponível em URL `http://en.wikipedia.org/wiki/FASTA_format`. Acesso em 12 de Dezembro de 2010., .

[29] GenBank Flat File Format. Disponível em Disponível em URL `http://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html`. Acesso em 5 de Novembro de 2010.

[30] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. A basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[31] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acid Research*, 25:3389–3402, 1997.

[32] Kim D. Pruitt, Tatiana Tatusova, and Donna R. Maglott. NCBI Reference Sequence (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Research*, 33(suppl 1):D501–D504, 2005. doi: 10.1093/nar/gki025. URL `http://nar.oxfordjournals.org/content/33/suppl_1/D501.abstract`.

[33] Databases available for BLAST search. Disponível em Disponível em URL `http://www.ncbi.nlm.nih.gov/blast/blast_databases.shtml`.

[34] Alfredo Cuzzocrea and Sharma Chakravarthy. Event-based lossy compression for effective and efficient OLAP over data streams. *Data Knowl. Eng.*, 69:678–708, July 2010. ISSN 0169-023X. doi: http://dx.doi.org/10.1016/j.datak.2010.02.006. URL `http://dx.doi.org/10.1016/j.datak.2010.02.006`.

[35] Stream Query Repository. Disponível em Disponível em URL: `http://infolab.stanford.edu/stream/sqr/`. Acesso em 08 de Outubro de 2010., 2010.

[36] Taxonomy at National Center for Biotechnology Information - NCBI, 2010.