Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Proveniência de Dados de Workflows de Bioinformática usando o Banco de Dados NoSQL ArangoDB

Bruno Aires de Sousa

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientadora
Prof.ª Dr.ª Maristela Terto de Holanda

Brasília
2015

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Curso de Engenharia da Computação

Coordenador: Prof. Dr. Ricardo Zelenovsky

Banca examinadora composta por:

Prof.ª Dr.ª Maristela Terto de Holanda (Orientadora) — CIC/UnB
Prof.ª Dr.ª Aletéia Patrícia Favacho de Araújo — CIC/UnB
Prof.ª Dr.ª Maria Emília Machado Telles Walter — CIC/UnB

CIP — Catalogação Internacional na Publicação

Aires de Sousa, Bruno.

Proveniência de Dados de Workflows de Bioinformática usando o Banco de Dados NoSQL ArangoDB / Bruno Aires de Sousa. Brasília : UnB, 2015.
59 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2015.

1. ArangoDB, 2. Proveniência, 3. Banco de Dados, 4. NoSQL híbrido.

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Proveniência de Dados de Workflows de Bioinformática usando o Banco de Dados NoSQL ArangoDB

Bruno Aires de Sousa

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof.ª Dr.ª Maristela Terto de Holanda (Orientadora)
CIC/UnB

Prof.ª Dr.ª Aletéia Patrícia Favacho de Araújo
CIC/UnB

Prof.ª Dr.ª Maria Emília Machado Telles Walter
CIC/UnB

Prof. Dr. Ricardo Zelenovsky
Coordenador do Curso de Engenharia da Computação

Brasília, 15 de dezembro de 2015

Dedicatória

Dedico a minha família e amigos por todo apoio e confiança que me concederam ao longo desses anos universitários.

Agradecimentos

Primeiramente a Deus, pois sem Ele eu não teria forças para essa longa jornada. Aos meus pais, Heliomar e Lucirene, pelo amor, incentivo e apoio incondicional. A minha namorada, Bárbara, pela dedicação, amor e compreensão em todos os momentos. A minha orientadora, professora, chefe e amiga, Prof.ª Dr.ª Maristela Terto de Holanda, pelas oportunidades, orientações e empenho dedicados neste trabalho. Aos amigos que mesmo na minha ausência se mostraram sempre presentes.

Resumo

Este trabalho apresenta uma análise da utilização do sistema gerenciador de banco de dados NoSQL ArangoDB em *workflow* de Bioinformática. O ArangoDB é um banco híbrido que possui um modelo baseado em grafo e em documento para persistência de dados. Para isso, foi realizado um estudo sobre o armazenamento de dados gerados na fases de filtragem e de mapeamento de um *workflow* de Bioinformática, bem como a geração de grafos de proveniência a partir dos processos utilizados. O estudo foi motivado pela possibilidade de se armazenar os dados gerados ao longo do processamento do *workflow* e informações sobre sua execução em um mesmo lugar, o que facilitaria a reexecução de um *workflow* científico, visto que não seria necessário buscar novamente os dados que foram utilizados em um dado experimento. Como resultado, este trabalho demonstra como os dados gerados pelo *workflow* e seus dados de proveniência foram armazenados no ArangoDB utilizando o modelo PROV-DM.

Palavras-chave: ArangoDB, Proveniência, Banco de Dados, NoSQL híbrido.

Abstract

This work presents a study about the use of the database management system NoSQL ArangoDB in Bioinformatics workflow. The ArangoDB is a database that has a hybrid model based on graph and document for data persistence. It was studied the storing of data of the filtering and mapping stages from a Bioinformatics workflow, as well the provenance graph generated by the used process. The study was motivated by the possibility of storing data generated during the processing of the workflow and data about its execution in one place, which would facilitate the re-execution of a scientific workflow, because it would not be necessary to look again the data that were used in a given experiment. As a result, this work demonstrates how the data generated by workflow and its provenance were stored on ArangoDB using the PROV-DM model.

Keywords: ArangoDB, Provenance, Database, NoSQL multi-model.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Centros de pesquisa em Genômica utilizam ferramentas que constituem sistemas de informação para processamentos e análises genômicas [Guimarães and Cavalcanti, 2009]. Anotações relacionadas aos experimentos precisam ser armazenadas para dar suporte aos experimentos em laboratório. Devido à grande quantidade de dados dos sistemas de bioinformática, eles passaram a ser utilizados em conjunto com esquemas de bancos de dados genômicos com o objetivo de auxiliar pesquisas. Para uma posterior obtenção das informações relacionadas a um dado, é importante que sejam registradas informações sobre sua origem, isto é, o histórico de fatores envolvidos no processamento e na geração desse dado. Dessa forma, é possível obter informações sobre a proveniência de um dado, ou seja, sobre a origem ou procedência de um determinado dado.

As informações sobre a proveniência de um dado podem ser importantes para auxiliar a reexecução de experimentos por diferentes pesquisadores, visto que estarão disponíveis em bancos de dados. A possibilidade de uma nova execução por terceiros faz com que um experimento seja válido do ponto de vista científico, por isso é necessário que se armazenem os dados tanto do ambiente de execução quanto dos experimentos propriamente ditos [de Almeida, 2014]. Várias iniciativas na Bioinformática já auxiliam a captura de dados de proveniência que aliadas às descobertas de experimento científico podem ser associadas por meio do apoio de sistemas de informação.

Segundo Ogasawara [2012], a complexidade dos modelos utilizados nos experimentos científicos também acompanhou essa evolução, e diversas atividades, sendo algumas até encadeadas, foram utilizadas durante as simulações para produzir uma coleção de dados com determinada sintaxe e semântica. Esse encadeamento é representado através de *workflows* científicos.

Áreas como a Biologia, Química, Física, Ecologia, Astronomia, Geologia e Engenharias em geral compartilham algumas características, tais como a necessidade da manipulação de grandes volumes de dados e um alto poder computacional para processá-los [Deelman

et al., 2009].

Projetos de Bioinformática com sequências de DNA (ácido desoxirribonucleico) ou RNA (ácido ribonucleico) obtidas por sequenciadores de alto desempenho são capazes de gerar milhões de fragmentos de DNA na ordem de *terabytes* [Schuster, 2007]. Nestes experimentos de Bioinformática, é de fundamental importância armazenar os dados de proveniência para permitir a reexecução de um experimento. Para que isto aconteça, modelos de dados para a proveniência existem para facilitar o armazenamento e a recuperação desses dados.

Além dos dados de proveniência, também propõe-se armazenar os dados biológicos gerados nos projetos de Bioinformática. Dessa forma, o uso de um SGBD (Sistema de Gerenciamento de Banco de Dados) ou um sistema de arquivos é necessário para resolver o problema de armazenamento e administração desse grande volume de dados. A utilização de um SGBD também se dá pelas vantagens que ele oferece, tais como: segurança, organização, facilidade de consulta e compressão dos dados [Huacarpuma, 2012].

Neste contexto, esta monografia aborda o problema da dificuldade do armazenamento de dados de um experimento genômico, bem como sua proveniência em um mesmo banco de dados. A hipótese adotada é que um banco de dados híbrido seria adequado para a persistência de tais dados, tendo em vista que um SGBD híbrido além de oferecer uma melhor administração do grande volume de dados, segurança para o armazenamento e integridade dos dados, além de facilitar as consultas e armazenar os dados de proveniência.

A importância deste estudo é válida tanto para a área de Bioinformática quanto para a Computação, visto que seu desenvolvimento pode permitir:

- Armazenamento de dados e de proveniência de um *workflow* científico em um mesmo banco de dados;
- Análise de um banco de dados NoSQL híbrido.

1.1 Objetivo

O objetivo deste trabalho é analisar do comportamento do banco de dados híbrido ArangoDB com dados e proveniência de *workflows* de Bioinformática, buscando desenvolver uma implementação adequada e verificar se é uma opção viável em relação ao desempenho.

1.1.1 Objetivos Específicos

Para que o objetivo geral seja atingido, foram definidos alguns objetivos específicos:

- Definir uma metodologia para criação e utilização da interface do ArangoDB, incluindo o modelo e as estruturas de dados a serem utilizadas para inserção e extração de dados;
- Avaliar o desempenho do banco para inserção e extração de dados gerados pelas fases de filtragem e mapeamento de um *workflow* de Bioinformática;
- Realizar um estudo sobre o modelo de proveniência necessário para ser utilizado no ArangoDB.

1.2 Estrutura do Trabalho

Este trabalho está estruturado nos capítulos a seguir:

- Capítulo 2: Apresenta o referencial teórico necessário para o desenvolvimento desta pesquisa, tais como conceitos de proveniência de dados, o modelo PROV-DM e a utilização de bancos de dados não-relacionais.
- Capítulo 3: Apresenta o detalhamento do ArangoDB, mostrando sua origem, suas características gerais e seu funcionamento interno. Também são apresentadas as suas limitações e informações úteis de configuração.
- Capítulo 4: Apresenta o ambiente de testes e os dados utilizados, incluindo seu formato para os dados de entrada.
- Capítulo 5: Expõe os resultados atingidos seguido de uma breve análise destes.
- Capítulo 6: Traz conclusões finais e possíveis trabalhos futuros.

Capítulo 2

Referencial Teórico

Neste capítulo são tratados os conceitos relacionados à proveniência de dados bem como o modelo de dados aplicado a um banco de dados NoSQL. A Seção 2.1 apresenta a definição de proveniência de dados, seus modelos e objetivos. A Seção 2.2 trata de aspectos relacionados a bancos de dados não relacionais. A Seção 2.3 uma descrição sobre projetos de Bioinformática e sobre a estrutura geral do *workflow* de sequenciamento de alto desempenho enquanto a Seção 2.4 apresenta alguns trabalhos relacionados.

2.1 Proveniência de Dados

Proveniência de dados diz respeito à origem ou procedência de determinado dado. A proveniência armazena informações com mais significado do que a simples identificação do local de procedência [de Paula et al., 2013], ou seja, descreve de forma detalhada a geração de um objeto de dados.

Em [Buneman et al., 2001] proveniência de dados é definida como "a descrição das origens de uma peça de dados e do processo pelo qual ela chegou em um banco de dados". Isto mostra que a origem de um dado é feita a partir de um processo, em que se obteve um dado produto, e que sua origem foi um dado fonte utilizado como entrada.

Dessa forma, deve-se identificar quais são os dados relevantes para o armazenamento, tais como processos que o derivaram, objetos de dados utilizados por estes processos, agentes envolvidos na derivação, ambiente de execução etc. Questões sobre quem criou determinado dado, quando foi modificado, o responsável pela modificação, além de qual processo foi utilizado para criar o dado são questões que podem ser respondidas com o auxílio da proveniência.

Os dados obtidos da proveniência podem ser utilizados para reexecutar os experimentos, se proteger contra disputas de propriedade intelectual [Hasan et al., 2007], avaliar a qualidade dos dados, além da possibilidade de reprodução de experimentos.

A proveniência de dados pode ser classificada de acordo com [Davidson and Freire, 2008] em três tipos:

- Prospectiva: semelhante a uma receita, visto que captura os passos a serem seguidos para gerar um certo produto;
- Retrospectiva: trata das informações obtidas durante a execução dos processos de geração de dado, bem como o tempo de duração de cada atividade executada até a origem dos dados de entrada, além de informações sobre o ambiente utilizado. Dessa forma, é possível obter um *log* detalhado sobre a execução da tarefa;
- Dados definidos pelo usuário: anotações, conclusões, observações ou qualquer informação que o usuário julgar necessária para futura utilização. Também pode-se citar anotações, conclusões a respeito do processo e, até mesmo, observações sobre parâmetros utilizados.

Davidson and Freire [2008] também dividem o nível em que a captura é realizada quando é feita de forma automática, de acordo com a divisão abaixo:

- *Workflow*: utilizado pela grande maioria das soluções com SGWfC (Sistemas de Gerência de *Workflow* Científico) envolvendo a descrição da execução de um processo, e nesse caso deve ser adaptado para capturar os dados dos diferentes processos executados;
- Atividade: podendo ocorrer de duas formas, em que na primeira cada processo executado é alterado para capturar os dados de proveniência e, na segunda forma, em que programas específicos podem ser criados para monitorar a execução de um determinado processo e capturar os dados de proveniência;
- Sistema Operacional: utiliza os dados fornecidos pelo próprio sistema operacional como insumo para a proveniência.

Quando a obtenção da proveniência é executada somente no momento que é solicitada, [Tan, 2004] classifica como uma abordagem preguiçosa (*lazy*), porém quando a proveniência é obtida durante a geração da informação e é armazenada para consultas futuras é chamada de abordagem ansiosa (*eager*).

O principal objetivo dos modelos de proveniência é estabelecer uma estrutura para que os dados de proveniência possam ser armazenados e recuperados, de forma que seu significado seja mantido e seus benefícios possam ser potencializados [de Paula, 2013]. Uma breve revisão sobre diferentes modelos de proveniência que podem ser encontrados na literatura atual são descritos a seguir:

- *W7*: apresentado por Wong [2007], tem como base a ontologia de Bunge [Bunge, 1977], a qual objetiva descrever as propriedades de um objeto de caráter geral. A partir desse modelo estruturou-se a proveniência de uma peça de dado através da resposta a 7 perguntas (ou dimensões): O que?, Quem?, Quando?, Onde?, Como?, e Por quê?;

- *Provenance Vocabulary*: descrito por [MacGuinness et al., 2010] volta sua atenção para o problema da proveniência de dados publicados na web. A sua principal característica é fornecer classes e propriedades para que publicadores de dados para web possam armazenar, além dos dados publicados, também os metadados com informações uteis sobre a proveniência dos dados publicados;

- *Provenir Ontology*: descrito por [Sahoo and Sheth, 2009] foi desenvolvido para ser um modelo de proveniência de dados genético, priorizando a interoperabilidade entre diferentes sistemas e sua adaptação para qualquer aplicação. Assim como no modelo *Provenance Vocabulary*, define um núcleo comum e permite a criação de módulos específicos para o domínio da aplicação desejada;

- OPM (*Open Provenance Model*): começou a ser discutido em maio de 2006 no *Workshop* Internacional de Anotação e Proveniência. É um projeto aberto que consiste em caracterizar a proveniência de qualquer objeto, material ou imaterial. Um grafo acíclico direcionado representa e descreve a relação causal entre eventos que afetam objetos. O modelo OPM, descrito em [Moreau et al., 2009] procura demonstrar a relação causal entre eventos que afetam objetos (digitais ou não) e descreve essa relação por meio de um grafo acíclico direcionado;

- PROV-DM (*Provenance Data Model*): é um modelo recente baseado no OPM, cuja função é descrever pessoas, entidades e atividades envolvidas na produção de uma peça de dado ou um objeto qualquer através de um grafo direcionado. Essa descrição provê condições para demonstrar a proveniência de forma mais precisa. Foi desenvolvido em outubro de 2011 e sua versão mais recente publicada em abril de 2013 [de Almeida, 2014]. A raiz deste grafo representa a entidade cuja proveniência está sendo representada e as arestas são direcionadas para as atividades e entidades das quais foram originadas. O PROV-DM é detalhado a seguir.

De acordo com Moreau and Missier [2013], o PROV-DM é dividido em seis componentes, os quais contém tanto os elementos quanto as relações possíveis:

- Entidades e Atividades: entidades podem representar qualquer objeto (real ou imaginário), e atividades é algo que ocorre ao longo de um período de tempo e atua sobre ou com entidades;

- Agentes e Responsabilidades: agentes são entidades que influenciam, direta ou indiretamente, a execução das atividades, recebem atribuições de outros agentes e podem ter algum tipo de ligação (posse, direitos, etc...) sobre outras entidades;
- Derivações: descreve a relação entre diferentes entidades durante o ciclo de transformação executado pelas atividades permitindo demonstrar a dependência entre as entidades usadas e geradas;
- Alternativo: descreve a relação entre diferentes visões de uma mesma entidade;
- Coleções: são entidades que possuem membros, os quais são também entidades, e podem ter a sua proveniência demonstrada de forma coletiva;
- Anotações: fornece mecanismos para inclusão de anotações para os elementos do modelo.

A Figura 2.1 ilustra os símbolos utilizados pelo modelo PROV-DM para representar os diferentes nós do grafo. O símbolo da Entidade também é utilizado para representar o tipos Coleção, uma vez que representa subtipo do tipo Entidade. As relações da Figura 2.2 representam as arestas no grafo de proveniência que, por sua vez, indicam as relações possíveis entre cada nó. Além de descrever cada tipo que compõe o modelo, os seis componentes também detalham as relações que podem ocorrer entre cada um dos tipos.

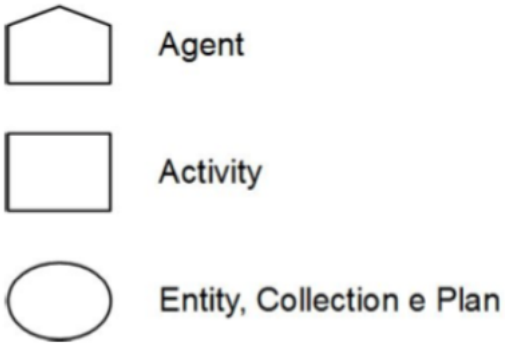

Figura 2.1: Representação gráfica dos nós do modelo PROV-DM.

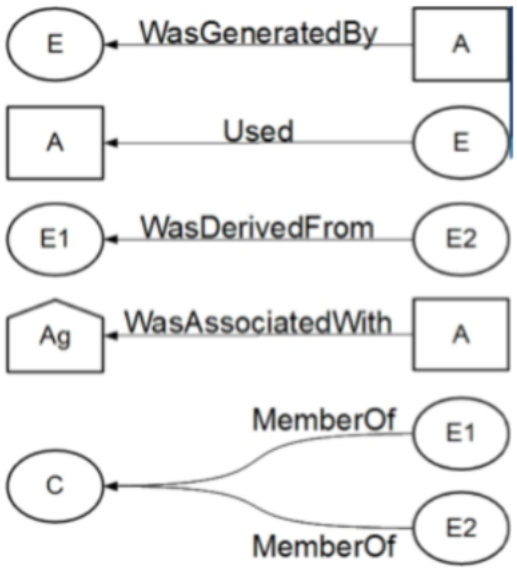

Figura 2.2: Representação gráfica das relações do modelo PROV-DM.

Um detalhamento dos tipos pode ser encontrado em de Paula [2013], bem como a base para a escolha do PROV-DM de acordo com os principais requisitos, dentre eles:

- Capacidade de representar a proveniência de uma peça de dado, descrevendo os processos e insumos utilizados em sua geração;
- Uma representação gráfica adequada, com diferentes símbolos para cada elemento, e relações suficientes para demonstrar a proveniência de forma objetiva;
- Símbolo para representar grandes conjuntos de dados;
- Extenso material disponível cobrindo diferentes aspectos da proveniência de dados;
- O fato de estar sendo desenvolvido pelo W3C e ser recomendação desta instituição em 2013;
- Capacidade do modelo de proporcionar o intercambio de informações entre diferentes sistemas.

Dessa forma, de acordo com de Paula [2013], a aplicação do modelo PROV-DM para representar a proveniência de dados em projetos de Bioinformática, se mostrou bastante simples e direta. Os componentes do modelo, tais como o agente, atividade e coleção, representam elementos presentes em grande parte dos experimentos executados em projetos de Bioinformática. As relações, por sua vez, demonstram de forma objetiva as dependências entre cada elemento no grafo, e a utilização das regras e do tipo de derivação permitem maior grau de especificidade quando necessário.

2.2 Bancos de Dados Não Relacionais

Bancos de dados não relacionais surgiram como alternativa à questão de escalabilidade no armazenamento e no processamento de grandes volumes de dados [de Diana and Gerosa, 2010]. O termo NoSQL foi utilizado pela primeira vez em 1998 para citar um banco de dados relacional *open-source* que omitia o uso de SQL, o Strozzi NoSQL, criado por Carlo Strozzi [Sadalage and Fowler, 2012], porém o uso do termo como é conhecido atualmente não se refere a esse banco. O movimento NoSQL é recente e vem crescendo rapidamente [**?**] e dentre os fatores do seu sucesso estão a natureza dos dados da web, a distribuição de sistemas em escala global e a necessidade de se atingirem altos graus de paralelismo no processamento de grandes volumes de dados [de Diana and Gerosa, 2010]. Em 2009, na conferência *"NoSQL Meetup"* utilizou-se o termo referenciando bancos de dados não-relacionais.

Assim, reconhecendo-se a utilidade dos modelos tradicionais de banco de dados e criando-se uma nova categoria de armazenamento de dados surgem os bancos de dados "não apenas SQL". O termo NoSQL não define precisamente o que são esses bancos, mas os agrupa de acordo com características em comum, dentre elas : não-relacional, distribuído, de código aberto, escalável horizontalmente, esquemas flexíveis, suporte à replicação nativo e acesso via APIs simples [de Diana and Gerosa, 2010].

Levando em consideração que são necessárias soluções otimizadas para a natureza particular dos dados, é conveniente o surgimento de tecnologias de gerenciamento de dados diferentes das tradicionais. Com o intuito de reduzir custos através de uso de hardware comum e barato para o processamento de grandes volumes de dados, também é necessário que essa nova solução seja escalável horizontalmente. Levando em consideração a área da Bioinformática, em que grupos de pesquisadores podem realizar colaborações mesmo distantes geograficamente [Mattoso et al., 2008], também é importante que o sistema seja distribuído para atender diversos usuários de forma eficiente, mas isso requer também um sistema robusto para tolerar possíveis falhas.

Logo, pode-se definir NoSQL como um conjunto de conceitos que permite o processamento de dados de forma rápida e eficiente com um foco em desempenho [McCreary and Kelly, 2013], sendo um modo alternativo de modelagem de dados sem os padrões rígidos do modelo relacional.

Os bancos de dados relacionais apresentam todas as características ACID (Atomicidade, Consistência, Isolamento e Durabilidade) enquanto os não relacionais se baseiam na propriedade BASE (*Basically Avaliable, Soft State, Eventual Consistency*), ou seja, basicamente disponível, estado leve e consistente em momento indeterminado [Alves, 2015]. De acordo com [McCreary and Kelly, 2013] algumas características comuns dos bancos NoSQL são:

- Grande escalabilidade horizontal: capacidade de requisição de mais recursos de armazenamento e de processamento conforme a necessidade da aplicação aumentar seu numero de nós;
- Baixa latência: a partir da escalabilidade horizontal deve-se otimizar o tempo de uma grande processamento através da divisão em pequenos processos distribuídos em diferentes máquinas, possibilitando assim a diminuição do tempo de resposta;
- Disponibilidade de acesso: o grande poder de processamento permite que o banco seja utilizado por mais usuários que os bancos de dados relacionais.

2.2.1 Modelos de Bancos de Dados NoSQL

Estão disponíveis diversos modelos de banco de dados NoSQL, dentre os quais destacam-se os de chave/valor, orientado a documentos, orientado a colunas e grafos [Holanda and Souza, 2015]. Não existe um modelo superior ao outro, pois cada um possui vantagens e desvantagens dependendo do contexto em que for aplicado. Vale ressaltar que cada modelo tem formas diferentes de armazenamento e consulta. Assim, os principais modelos são destacados a seguir:

- Chave/Valor: por meio de pares de chaves e valores, o sistema armazena dados estruturados. É o modelo de estrutura mais simples, visto que uma chave é um identificador para diversos valores, que ainda podem ser expressos por índices *hash* [Alves, 2015]. Todas as consultas e inserções de dados são realizadas sobre as chaves [Carniel et al., 2012]. Consultas mais complexas ficam prejudicadas com esse tipo de modelo de banco [Lóscio et al., 2011], mas geralmente esses sistemas fornecem um mecanismo de persistência e funcionalidades adicionais, bem como replicação, versionamento, travamento de edição, transações dentre outros [Cattell, 2011]. Exemplos de banco de dados que seguem o modelo chave/valor são DynamoDB, Oracle NoSQL Database, allegro-C e Sophia [Archive, 2015];
- Orientado a Documentos: este modelo permite o armazenamento de dados mais complexos que os de chave/valor. Os documentos são coleções de atributos e valores, podendo ser textos, listas ou outros documentos. Também diferentemente do modelo chave/valor, este tipo de armazenamento permite o suporte a chaves secundárias e múltiplos tipos de documentos por banco de dados ou aninhados a outros documentos e listas. Com uma gama mais ampla de valores, os documentos diferem das tuplas, visto que os atributos não são definidos em um esquema global o que acaba se tornando uma grande vantagem desse modelo. Exemplos de banco

de dados que seguem o modelo orientado a documentos são CouchDB e MongoDB [Alves, 2015];

- Orientado a Colunas: através da indexação de uma tripla, os dados são diferenciados por múltiplas versões utilizando o *timestamp*, e as linhas e colunas são identificadas por chaves. Dessa forma, a tripla é composta pela linha, coluna e *timestamp*. A propriedade de atomicidade está presente nas operações de leitura e escrita, em que todos os valores associados a uma linha são considerados na execução destas operações. Pode-se agrupar colunas pelo tipo de dados através do recurso de família de colunas. Esse modelo de persistência de dados surgiu com o Google, fazendo assim com que o modelo também seja conhecido como *BigTable* [Chang et al., 2008]. Apesar de fornecer forte consistência dos dados, este modelo não garante alta disponibilidade [Lóscio et al., 2011]. Cassandra é um exemplo de NoSQL que utiliza esse modelo;

- Grafos: permitem o armazenamento de entidades e os relacionamentos entre essas entidades, que também são conhecidas como nós, os quais possuem propriedades. Os relacionamentos são conhecidos como arestas, que também podem ter propriedades. As arestas têm significância direcional. Assim, os nós são organizados por relacionamentos, os quais permitem encontrar padrões entre eles. A execução de consultas complexas é facilitada por esse modelo de banco de dados. Exemplos de banco de dados que seguem o modelo orientado a documentos são o Neo4j e o AllegroGraph [Alves, 2015];

- Híbridos: a definição de um modelo de dados híbrido para persistência de dados utilizado neste estudo é a de um banco NoSQL que pode utilizar-se de vários modelos em conjunto para realizar o armazenamento de seus dados, e relacioná-los a outros dentro do mesmo banco de dados. O ArangoDB exemplifica essa definição a partir do armazenamento dos dados através de coleções de documentos, cada um contendo sua chave própria de identificação, e podendo relacionar vários documentos, até mesmo de coleções diferentes, através de grafos.

2.3 Projetos de Bioinformática

Por meio de recursos computacionais, softwares e bancos de dados, um pesquisador pode processar informações obtidas em laboratório. A área da computação responsável por investigar o armazenamento e análise dos dados obtidos é a Bioinformática [Luscombe et al., 2001].

Diz-se que um cientista realiza um experimento *in-silico* quando utiliza ferramentas computacionais [Greenwood et al., 2003] para testar uma hipótese, buscar padrões ou demonstrar fatos. Tais experimentos são compostos por uma sequência de programas encadeados, chamada de *workflow* [da Silva Almendra Gomes, 2011].

A WfMC (*Workflow Management Coalition*) define *workflow* como "a automação de um processo de negócio, total ou parcial, na qual documentos, informações ou tarefas são transferidas entre participantes de acordo com um conjunto de regras"[Hollingsworth and Hampshire, 1993]. *Workflows* podem ser utilizados para implantar processos de negócios, com atividades que fazem parte do processo a fim de atingir certo objetivo.

Estes *workflows* são compostos por fluxos de dados, ou seja, a entrada de uma tarefa é determinada pelo resultado da tarefa anterior. Esse encadeamento pode gerar inconvenientes, caso o cientista queira testar seu *workflow* com parâmetros e entradas diferentes. Nesse caso, será necessário que se execute novamente todo o procedimento, ou seja, um mesmo experimento pode gerar inúmeras execuções de *workflows* [Mattoso et al., 2010].

Com a variedade de processos que podem ser executados, através de diferentes opções de programa, parâmetros e configurações, faz-se necessário uma solução que permita armazenar a relação entre os dados processados, as informações sobre execuções de *workflows* e seus resultados. Além desses dados também é necessário registrar quem atuou na execução de cada experimento, visto que uma maior confiabilidade aos resultados é devido ao usuário e a função exercida por ele em determinado experimento [de Paula, 2013].

A captura dos dados de proveniência para um *workflow* científico pode ser implementada manual ou automaticamente [de Paula, 2013]. A automatização pode ser realizada através de *scripts* ou de SGWfC (Sistemas Gerenciadores de *Workflow* Científicos). Para a utilização de *scripts* também é necessário um maior cuidado em relação à documentação, pois se estiver incompleta ou falha, a reprodutibilidade e a compreensão do experimento por outro cientista pode ser dificultada [de Paula, 2013].

Os SGWfC são alternativas que foram desenvolvidas para substituírem os *scripts ad-hoc*. Esses sistemas são construídos de forma que o usuário não precise programar um *script*, tudo é feito através de uma interface gráfica, de forma que o usuário possa adicionar os processos e gerar seu *workflow*. Além de gerar o *workflow*, os SGWfC são capazes de executar o fluxo produzido. Dessa forma, tais sistemas podem ser considerados ferramentas completas em que se constrói, se executa e se visualizam os resultados do experimento. Para a proveniência de dados, sua grande funcionalidade é oferecer a captura automática de dados de proveniência.

Especificamente em relação a *workflow* em Bioinformática, utiliza-se um *workflow* de três fases: filtragem, mapeamento e análise. A estrutura do *pipeline*, representada na Figura 2.3, permite voltar a uma fase anterior de acordo com a necessidade dos usuários

do projeto. Por exemplo, pode-se voltar da fase de mapeamento à fase de filtragem, da fase de análise à fase de mapeamento ou filtragem.

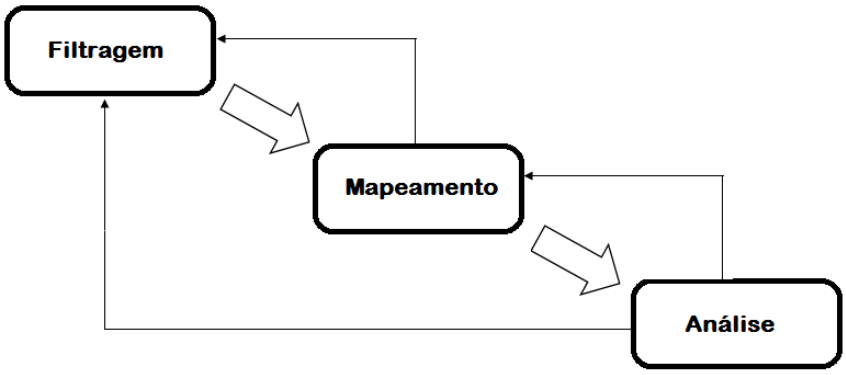

Figura 2.3: Estrutura do *Workflow* de Alto Desempenho.

A fase de filtragem remove sequências que apresentam DNA/RNA de regiões cujo sequenciamento não possui uma boa precisão em relação a qualidade, ou simplesmente contém regiões que não serão interessantes para as fases seguintes do *workflow* [Huacarpuma, 2012]. As SRS (*Short Read Sequences*) geradas a partir do sequenciamento possuem uma qualidade associada que expressa a probabilidade de erro associada a cada base, sendo que cada projeto é responsável por determinar um padrão mínimo de qualidade para seus experimentos [Huacarpuma et al., 2011].

A fase de mapeamento conta com um genoma de referência, normalmente, um organismo próximo ao que está sendo sequenciado que seja conhecido. A partir daí mapeiam-se as SRS pelos sequenciadores, e são agrupadas conforme a posição das mesmas no mapeamento. Logo, resume-se a fase de mapeamento como uma tarefa de buscar a localização em que um SRS em relação à de referência [Huacarpuma, 2012].

A fase de análise depende do foco do projeto, visto que seu objetivo é procurar informação relevante das SRS agrupadas na fase anterior, e interpretá-las para obter seu significado biológico e colocá-lo no contexto de entendimento dos processo biológicos [Stein, 2001]. Essa fase também serve para encontrar informações relevante, como sequências de genes e regiões reguladoras, identificação de expressão de genes, análises de filogênia, assim como outras análises para a formulação de hipóteses biológicas [Frishman and Valencia, 2009].

Vale ressaltar que neste trabalho focou-se nos dados relacionados à fase de filtragem cujas SRS utilizadas estão no formato FASTQ[1] e nos dados relacionados à fase de mapeamento que estão no formato BAM[2].

[1]Formato dos arquivos relativos a fase de filtragem.
[2]Formato dos arquivos relativos a fase de mapeamento gerados pelo TopHat.

2.4 Trabalhos Relacionados

Não é recente o problema da persistência e proveniência de dados genômicos, tendo em vista as dificuldades de gerenciamento dos dados como relata Bloom and Sharpe [2004]. Uma das dificuldades nos últimos anos é a quantidade de dados geradas pelos sequenciadores de desempenho em projetos genoma no mundo todo, como relatam Huacarpuma et al. [2011].

Röhm and Blakeley [2009] e Huacarpuma [2012] consideraram modelos relacionais para armazenar os dados genômicos das sequências biológicas e suas respectivas qualidades. Eles utilizaram, respectivamente, o SQL Server 2008 e o PostgreSQL em seus trabalhos, armazenando os arquivos no formato FASTQ. Huacarpuma [2012] ainda faz a comparação do tempo de inserção e extração de dados do banco relacional, e o tempo total gasto durante o processo de execução do *workflow* de Bioinformática.

Assim, sugeriu-se a partir de Bateman and Wood [2009], a utilização de bancos NoSQL para o sistema de armazenamento desses dados genômicos, considerando os reais problemas de armazenamento de grandes volumes de dados na ordem de *petabytes*.

Aniceto and Xavier [2014] mostram o avanço quando se escala os recursos físicos do banco, tornado assim as operações mais eficientes e dando suporte aos projetos de Bioinformática através de um banco de dados em nuvem. Um dos resultados deste trabalho é a melhora no desempenho das inserções e extrações de dados do banco em relação ao modelo relacional.

Guimaraes et al. [2015] realiza um estudo sobre o armazenamento de dados de proveniência da Bioinformática no banco de dados NoSQL MongoDB. É discutido o modelo de dados PROV-DM, sua implementação e a comparação com o estudo de Huacarpuma [2012] no PostgreSQL.

Hecht and Jablonski [2011], motivados pelo desenvolvimento de aplicações para web 2.0, comparam o modelo de dados, possibilidades de consulta, controle de concorrência, particionamento, replicação e consistência de dados para mostrar que cada banco de dados NoSQL possui vantagens e desvantagens, e não existe um banco melhor do que o outro, visto que existem requerimentos diferentes para cada aplicação específica.

A análise e os desafios necessários para o provimento do apoio computacional ao desenvolvimento de experimentos em larga escala são a motivação principal do trabalho de Mattoso et al. [2008]. São relatadas questões sobre a modelagem da proveniência devido ao formato heterogêneo dos dados, incluindo a especificidade do domínio da aplicação científica para cada contexto. É relatada a necessidade de ferramentas e apoio ao cientista para o desenvolvimento de ciência em larga escala por meio de uma infraestrutura computacional adequada.

A criação de um único grafo sumário representando todos os grafos de proveniência gerados durante um experimento, mas com um tamanho reduzido e eliminando dados repetidos é o objetivo do trabalho de El-Jaick et al. [2013]. A abordagem de sumarização permite reduzir o tempo de processamento de consultas de proveniência através da utilização apenas do grafo sumário, eliminando assim a necessidade de reconstruir os grafos originais. É utilizado um banco de dados de grafos. Mesmo com esse tipo de BD, ainda persiste o problema da manipulação de um grande volume de dados dos grafos de proveniência, mas a sumarização é uma abordagem para se utilizar com grafos volumosos.

Um estudo comparativo é realizado por Guimarães and Cavalcanti [2009] em alguns cenários da Biologia para investigar a real necessidade da proveniência de dados utilizando alguns bancos de dados. É explicado o conceito de anotação no contexto do estudo da Bioinformática, e como banco de dados são criados a partir dessas anotações, em que a meta-informação de uma anotação é o dado correspondente àquela proveniência.

Trabalhos para armazenamento da proveniência em bancos de dados NoSQL utilizando o modelo PROV-DM são realizados por Guimaraes et al. [2015], de Almeida [2014] e de Paula [2013]. Em Huacarpuma [2012] e Aniceto and Xavier [2014] são apresentados diferentes SGBD (Sistema Gerenciador de Banco de Dados) relacional e não-relacional, respectivamente, apenas para armazenamento dos dados gerados durante a execução do *workflow* de Bioinformática. Diferentemente destes trabalhos, apresenta-se uma proposta de armazenamento dos dados gerados e informações sobre proveniência na execução do *workflow* em Bioinformatica em um mesmo SGBD.

De acordo com os trabalhos relacionados, podemos resumi-los na Tabela 2.1 de acordo com as características que foram abordadas em cada um, em que X representa a presença de determinada característica no estudo e - sua ausência.

Dessa forma, propõe-se que este trabalho utilize o ArangoDB na área de Bioinformática para que armazene dados biológicos e sua devida proveniência através de grafos.

Artigo	**Dados Biológicos**	**Proveniência**	**SGBD utilizado**	**Bioinformática**	**Grafos**
Guimaraes et al. [2015]	X	X	MongoDB	X	-
de Almeida [2014]	X	X	Neo4j	X	X
de Paula [2013]	X	X	-	X	X
Huacarpuma [2012]	X	-	PostgreSQL	X	-
Aniceto and Xavier [2014]	X	-	Cassandra	X	-
Guimarães and Cavalcanti [2009]	-	X	Vários	X	-
El-Jaick et al. [2013]	-	X	-	X	X
Mattoso et al. [2008]	-	-	-	X	-
Röhm and Blakeley [2009]	X	-	SQLServer	X	-
Hecht and Jablonski [2011]	-	-	Vários	-	X

Tabela 2.1: Tabela-resumo

Capítulo 3

ArangoDB

Como citado anteriormente, o banco de dados utilizado neste trabalho foi o ArangoDB. Este capítulo descreve o banco de dados através de suas características e funcionalidades pertinentes ao estudo que foi realizado. A Seção 3.1 trata da apresentação do banco. A linguagem de consulta do ArangoDB é descrita na Seção 3.2. Por fim, a Seção 3.3 relata aspectos relacionados à configuração do banco na máquina.

3.1 Definição e Modelo de Dados

O ArangoDB é um banco de dados NoSQL desenvolvido pela ArangoDB GmbH e lançado em 2011. É um banco de dados de código aberto que utiliza uma linguagem de consulta similar ao SQL tradicional ou extensões de JavaScript. Apesar de ser um banco de dados NoSQL, ele também permite utilizar transações ACID, caso seja necessário, além da escalabilidade horizontal e vertical [ArangoDB, 2015].

Este banco pode ser classificado como híbrido, visto que suporta a combinação chave/-valor, documentos e grafos. Seu modelo de persistência é determinado automaticamente, não sendo indicado pelo usuário. Dessa forma, o usuário pode modelar os dados tanto como documentos, pares chave/valor ou grafos. O tipo de coleção de um documento indicará se um determinado dado é um nó ou uma aresta na construção de um grafo, mas caso seja necessário apenas armazenar os dados, não é necessário especificar todos os atributos que farão parte daquela coleção de documentos, ou seja, podem-se ter dados com diferentes quantidades de atributos em uma mesma coleção de documentos.

3.1.1 Características Gerais

O ArangoDB é *multi-threaded* e baseado em memória, pois os dados são sincronizados com o sistema de arquivos enquanto metadados são armazenados na memória. Devido ao

Controle Concorrente de Multi-versões (MVCC - *Multiversion Concurrency Control*) os documentos não são apagados, em vez disso, uma nova versão do documento é armazenada, o que permite leitura paralela e ações de escrita. Dentre as principais características do ArangoDB, destacam-se:

- *Schema-Free*: combinação entre a eficiência de espaço do MySQL com o poder de performance do NoSQL;
- Servidor de Aplicativos: integração entre a aplicação e o banco de dados para o máximo de rendimento;
- JavaScript: utilização de uma única linguagem desde o navegador até o *back-end*;
- Modelo de dados flexível: modelagem dos dados através de combinações de chave/-valor, documentos e grafos;
- Escolha livre de índice: possibilidade de escolha do tipo de index;
- Linguagem de consulta (AQL - *ArangoDB Query Language*) para recuperação e modificação dos dados;
- Transações: execução de consultar em múltiplos documentos ou coleções com opção de consistência transacional e isolamento;
- Replicação: o banco de dados pode ser utilização em uma configuração mestre-escravo;
- Código Aberto.

Além disso, o pacote nativo do ArangoDB já conta com os seguintes programas:

- arangod: programa servidor que se destina a executar os processos como um *daemon* e serve para as várias conexões clientes ao servidor via TCP/HTTP;
- arangosh: cliente que implementa um ambiente interativo de programacao (REPL - *Read-Eval-Print*) e provê funções, acesso e administração do servidor ArangoDB;
- arangoimp: importador em lotes para o servidor ArangoDB;
- arangodump: criação de *backups* para o banco de dados;
- arangorestore: ferramenta para recarregar dados de um *backup*;
- arango-dfdb: utilizado como um *debugger* nos arquivos de dados do ArangoDB durante seu desenvolvimento.

3.1.2 Conceitos Fundamentais

Nesta seção são apresentados os conceitos fundamentais sobre os elementos que compõem o banco de dados, sua utilização e formas de acesso, e consultas ao banco, dentre eles destacam-se:

Índices

Índices permitem o acesso aos documentos, desde que o atributo indexado seja utilizado em uma consulta. O próprio banco já indexa alguns atributos automaticamente, mas o usuário também tem a liberdade de criar índices adicionais em atributos fora do sistema de documentos. Um índice definido pelo usuário é criado no nível da coleção, que pode ser especificado através dos nomes dos atributos que devem ser indexados.

O ArangoDB geralmente utiliza um único índice por coleção em uma determinada consulta, porém, caso existam vários índices em diferentes atributos da mesma coleção, a consulta, ainda assim, utilizará apenas um índice. No entanto, a criação de vários índices em diferentes atributos também pode ajudar na eficiência de consultas distintas através de condições de filtro em outros atributos.

Dessa forma, o otimizador de consulta tentará utilizar o índice mais seletivo possível quando se deparar com a escolha entre vários índices com uma estimativa de seletividade conhecido. A política do otimizador é produzir resultados corretos, independentemente de qual índice é usado para satisfazer as condições do filtro. Se ele não tem certeza se um índice violará essa politica, então ele não irá fazer uso do índice. Os atributos do sistema *_id*, *_key*, *_from* e *_to* são gerados e indexados automaticamente pelo ArangoDB. Além disso também fornece os seguintes tipos de índices:

- *Hash*: úteis para buscas de igualdade;
- *Skiplists*: podem ser utilizados quando as buscas envolverem intervalos de dados;
- *Fulltext Indexes*: ainda em desenvolvimento, mas proverá buscas mais rápidas para longas cadeias de caracteres;
- *Geo Indexes*: permite buscas em documentos em volta de certa localidade ou área.

Alguns tipos de índices permitem a indexação de um único atributo, por exemplo, o índice de texto completo, enquanto outros tipos de índices permitem a indexação de vários atributos ao mesmo tempo.

Databases

O ArangoDB é capaz de trabalhar com vários *databases* na mesma instância do servidor. Cada *database* pode ser utilizado para agrupar dados logicamente relacionados e separá-

los de outros grupos de dados. Cada banco de dados criado possui seu próprio conjunto de coleções que precisam ser configurados quando ele é criado. Se for necessário trocar de *database* durante a utilização do ArangoDB é necessário que o *driver* cliente armazene a *database* atual, visto que as conexões não contém informações de estado, sendo todas essas informações contidas nas requisições HTTP.

Sempre haverá um banco de dados padrão no ArangoDB, denominado *_system*. Essa *database* não pode ser eliminada, visto que provê operações especiais para criação, eliminação e enumeração de outras *databases*. Cada *database* é armazenada fisicamente em subdiretórios separados sob o diretório de *databases*, que está localizada no diretório de dados da instância. Além disso, cada *database* possui seu próprio subdiretório, chamado de *database*, que contém as coleções desse banco de dados e um arquivo *parameter.json* responsável por armazenar a ID e o nome da *database*.

Coleções

O ArangoDB é constituído por um conjunto de coleções, sendo que uma coleção é constituída por documentos. É unicamente identificada por seu identificador de coleções, também possuindo um nome único que pode ser utilizado pelos usuários para identificar e acessá-la. As coleções podem ser renomeadas a qualquer tempo, mas seu identificador continuará o mesmo, não podendo ser modificado. Os documentos contidos nas coleções podem ser de dois tipos: documentos (padrão) ou *edges*. O tipo de documento a ser armazenado deve ser especificado na criação da coleção e não poderá ser modificado posteriormente.

Documentos

Documentos são objetos JSON (*JavaScript Object Notation*) - um formato de intercâmbio de dados independente de linguagem baseada em texto, comumente usado em aplicações de Internet [Kewisch, 2014] - que podem ser aninhados e também conter listas. Cada documento é unicamente identificado por seus identificadores de documento [ArangoDB, 2015].

Todos os documentos possuem atributos especiais, sendo o *_id* seu identificador, o *_key* sua chave única e uma ETag, também conhecida como *_rev* responsável pela revisão do documento, e mantido autonomamente pelo próprio ArangoDB. O atributo *_id* é uma sequencia de caracteres e consiste do nome da coleção e a chave única do documento (*_key*) separados por uma /. Como citado anteriormente, o atributo *_key* é criado automaticamente, mas seu valor pode ser especificado pelo usuário no momento da criação do documento. Juntamente com o atributo *_id* não pode ser modificado após a criação do documento [ArangoDB, 2015].

O Controle Concorrente de Multi-versões, suportado pelo ArangoDB, permite que documentos existam em mais de uma revisão. As revisões podem ser utilizadas para atualizações condicionais, substituições ou exclusões de documentos no banco de dados. Caso seja necessário localizar uma revisão específica utilizar-se-ão os atributos *_id* e *_rev*. Atualmente, é utilizado um padrão de valores inteiros de 64 bits para manter internamente as revisões de documentos.

Grafos

A utilização de grafos é completamente gerenciada pelo ArangoDB, sendo inclusive visível pela interface web. As funcionalidades gráficas podem ser acessadas através de várias interfaces:

- Operações gráficas do AQL;
- Interface Gráfica RESTful;
- Implementação Gráfica JavaScript, podendo ser utilizada através de serviços FOXX - *framework* que permite a escrita de micro serviços.

3.1.3 Acesso aos Dados

Existem algumas formas de acesso às informações contidas na base de dados do ArangoDB. Segue uma descrição de cada uma:

- Interface REST: provimento de simples medidas para criar, acessar a manipular os dados através da identificação dos documentos ou consultas.
- *Arango Query Language*(AQL): linguagem de consulta inspirada do tradicional SQL, mas com recursos adicionais devido à sua natureza dinâmica de armazenamento de documentos. Os documentos podem ser construídos com a instrução *SELECT*, e aqueles que são aninhados também podem ser consultados. As instruções de consulta são transmitidas via HTTP da aplicação do ArangoDB e o formato de resposta é o JSON.
- JavaScript: Os documentos são acessados como objetos JavaScript e funções podem ser executadas através da interface REST. Na versão atual o acesso a todas as informações dos grafos só podem ser obtidas através da API JavaScript.

3.1.4 Interface Web

Também é possível que se realizem consultas, inserções e extrações de dados, criação de aplicações, gerenciar os bancos de dados criados, além de verificar os recursos utilizados e

as estatísticas de uso do banco através de sua interface gráfica, representada pela Figura 3.1, que mostra as estatísticas de requisição de acordo com seus tipos (GET, PUT, POST, DELETE etc), o número de conexões de clientes e a quantidade de *bytes* transferidos.

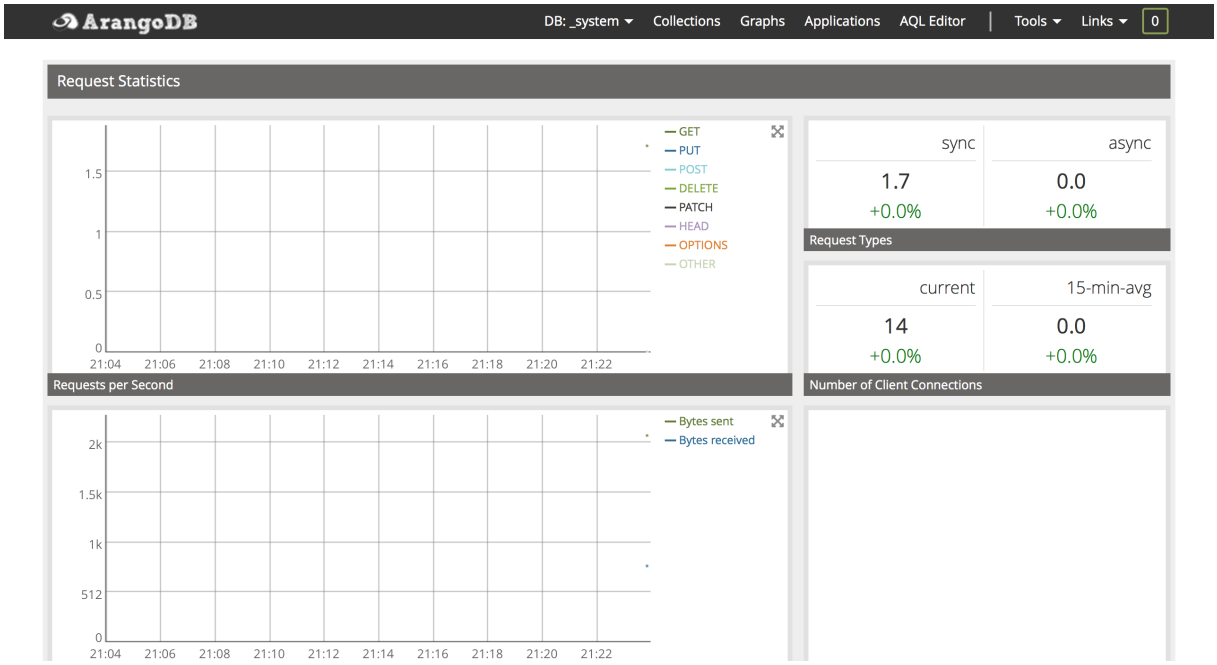

Figura 3.1: Interface Web do ArangoDB.

3.1.5 Cluster, Replicação e Sharding

A configuração de um *cluster* pode ser uma tarefa complicada devido ao *firewall*, portas, diferentes tipos de máquinas, porém o ArangoDB está preparado para realizar esta tarefa de maneira simples. A arquitetura de um *cluster* no ArangoDB consiste nas entidades de coordenadores e servidores de bancos de dados. Os coordenadores são responsáveis pela comunicação com os clientes, a distribuição deles entre os servidores de dados e o processamento dos resultados, enquanto os servidores de bancos de dados armazenam os dados propriamente ditos.

A replicação de dados em uma configuração mestre-escravo pode ser útil para garantir uma cópia de segurança dos dados. Todas as alterações devem ser feitas no banco de dados mestre do ArangoDB. Todas essas alterações, inclusões e exclusões serão copiadas para o banco de dados escravo. É possível que se mantenham diversos bancos escravos de um mesmo mestre, visto que as modificações no mestre são indicadas em um *log* que será replicado em todos os escravos correspondentes. No entanto, a replicação de dados em um banco de dados escravo pode sofrer um atraso devido a características da capacidade

da rede entre o mestre e o servidor, da carga de informações e da frequência com que o mestre atualiza seus dados.

Para que se utilize um *cluster* a partir de diferentes máquinas em um único banco de dados é necessário que se faça uso da técnica de *Sharding*. Ela permite que uma quantidade maior de dados seja armazenada e que o próprio ArangoDB faça a distribuição desses dados entre os diferentes servidores. O *Sharding* permite que se reduza o gargalo da carga de dados, visto que ela será dividida entre várias máquinas.

O processo para utilização de *Sharding* é constituído de duas fases: planejamento e execução. No planejamento é decidido o papel a ser executado em cada máquina e que portas serão utilizadas. Um documento no formato JSON é gerado com essas informações e pode ser utilizado posteriormente para a criação de um novo *cluster* com as mesmas características do antigo, uma espécie de restauração de *cluster*. A fase de execução consiste na ativação do *cluster*, em que cada instância deve ser configurada da mesma maneira em cada máquina participante do *cluster*.

A Figura 3.2 apresenta a interface web disponibilizada pelo ArangoDB para o gerenciamento do *cluster*. A interface de gerenciamento do *cluster* indica quantas máquinas foram disponibilizadas e a localização dos coordenadores e servidores com sua respectiva quantidade de memória virtual.

Figura 3.2: Interface de Gerenciamento do *Cluster*.

3.2 ArangoDB Query Language

A Linguagem de Consulta do ArangoDB (AQL - *ArangoDB Query Language*) pode ser utilizada para obter e modificar dados que estão armazenados no ArangoDB. Ela é uma

linguagem predominantemente declarativa, a consulta do AQL expressa que resultado deve ser alcançado, mas não a forma como deve ser feita, ou seja, as estruturas de consulta são descritas, mas a forma como elas serão utilizadas depende de cada usuário.

Com propósito similar a Linguagem de Consulta Estruturada (SQL - *Structured Query Language*), também utiliza palavras-chave em inglês e tem como objetivo ser legível para humanos. Porém, a sintaxe é diferente do SQL tradicional, apesar de algumas palavras-chave serem as mesmas em ambos. O processo de criação e exclusão de bases de dados, coleções e índice não é suportado pelo AQL, sendo essas operações realizadas por comandos no console *arangosh* ou pela interface gráfica web.

Outra característica do AQL é que os clientes utilizam a mesma linguagem e sintaxe para suas consultas, independentemente, daquela em que a aplicação do cliente foi construída.

As operações de modificação de dados utilizando o AQL, que o ArangoDB suporta, são descritas abaixo:

- INSERT: insere novos documentos em uma coleção;
- UPDATE: atualiza parcialmente os documentos existentes em uma coleção;
- REPLACE: substitui completamente documentos existentes em uma coleção;
- REMOVE: remove documentos existentes de uma coleção;
- UPSERT: insere ou atualiza condicionalmente documentos em uma coleção.

Tais operações podem ser combinadas dentro de um *loop* de repetição em uma lista de documentos ou também através de instruções de filtragem. Um exemplo de consulta AQL está ilustrado no código abaixo, em que se busca o nome de usuários do tipo *newbie* e que estejam ativo na coleção *users*.

```
FOR u IN usuarios
FILTER u.tipo == "novato" && u.ativo == true
RETURN u.nome
```

Consultas no AQL requerem a palavra-chave *RETURN* em suas instruções. Porém, diferentemente do SQL que pode separá-las por ponto e vírgula, não é permitido que uma dada instrução possua mais de uma operação no mesmo comando, ou seja, é retornado um erro caso a instrução possua mais que uma palavra-chave de operação de modificação de dados ou de consulta em um mesmo comando.

Para realizar operações mais complexas que necessitem de mais uma palavra-chave em seu comando pode-se utilizar as funções disponibilizadas pelo AQL, tais como concatenar

valores, substituir valores, dividir textos, calcular tamanho de *string* etc, porém o usuário tem a liberdade de criar funções conforme suas necessidades.

Outra forma de realizar essas operações mais complexas sem a utilização de funções se dá a partir das subconsultas, em que o resultado de uma consulta será utilizado para executar outra consulta. Essa situação está ilustrada no código a seguir.

```
FOR p IN pessoas
COLLECT cidade = p.cidade INTO g
RETURN {
cidade : cidade,
numPessoas : LENGTH(g),
maxAvaliacao: MAX(
FOR r IN g
RETURN r.p.avaliacao
)}
```

Os tipos de dados suportado pelo AQL são divididos em duas categorias:

- Tipos primários: composto por um único valor. Assim,

  Null: indica a inexistência de um valor;

  Bool: os valores possíveis são true e false;

  Number: os valores são representados por números reais.

  String: os valores são compostos por textos.

- Tipos compostos: compostos por múltiplos valores. Logo,

  Array: são referenciados por suas posições dentro de uma sequência de valores;

  Objetos e documentos: são referenciados por seus nomes dentro de uma sequência de valores.

3.3 Configuração do ArangoDB

O ArangoDB oferece suporte de instalação a diversos sistemas operacionais, dentre eles os principais: Mac OS, Linux e Windows. Em todos os casos é possível realizar a instalação através de pacotes ou compilando o código fonte original através do terminal de comando.

Após a instalação, pode-se utilizá-lo através de seu console, uma espécie de terminal de comando, ou através de sua interface gráfica web. Em ambos os casos é possível criar, inserir, atualizar e deletar dados, porém algumas ferramentas, tal como a importação em

lotes, só estão disponíveis para utilização através do console. Configuração de aplicações dentro do banco e execução de arquivos JavaScript, também podem ser executadas diretamente na interface web.

A interface web também permite que se faça a configuração de *cluster*, quando se deseja utilizar um ambiente distribuído, porém deve-se habilitar essa opção no arquivo "arangod.conf". Esse arquivo de configuração também permite a habilitação de *logs* de eventos e erros para realização de um *debugger* quando necessário.

Também é permitido que se alterem alguns parâmetros de determinados programas do ArangoDB para apenas uma ocasião, ou seja, não se muda o valor padrão de um parâmetro no arquivo de configuração, mas pode-se realizar uma única operação com o valor diferente do padrão. Isso é útil para o programa de importação em lotes, visto que modifica-se o tamanho de cada lote para determinada operação sem modificar o valor original de importação padrão dos lotes de arquivos de dados.

Capítulo 4

Estudo e Implementação

Neste capítulo é apresentado o estudo para analisar o armazenamento dos dados biológicos no banco NoSQL ArangoDB. Na Seção 4.1, são apresentadas as informações referentes aos arquivos de entrada. A Seção 4.2 apresenta o modelo de dados utilizado para armazenar a proveniência. A Seção 4.3 descreve o desenvolvimento das ferramentas utilizadas na implementação dos testes. Por fim, a Seção 4.4 descreve o ambiente distribuído criado para utilização de um *cluster*.

4.1 Características dos Dados da Bioinformática

O sequenciamento de amostras de células é um processo bioquímico que tem como objetivo determinar a ordem das bases nitrogenadas da molécula de DNA e a filtragem é o processo de filtrar as SRSs (*Short Read Sequence*) de baixa qualidade que podem afetar negativamente o estudo com estes arquivos [Huacarpuma, 2012]. Um dos principais formatos de dados na execução da filtragem em *workflows* de Bioinformática é o FASTQ, enquanto para a fase de mapeamento destaca-se o formato BAM.

Os dados a serem inseridos estão no formato FASTQ [Simon et al., 2009], que trata-se de um arquivo de texto codificado através de caracteres ASCII abreviados [Simon et al., 2009] para armazenar uma sequência biológica e seus índices de qualidade correspondentes. O arquivo é dividido em blocos denominados SRS, em que cada bloco contém 4 linhas compostos pelas cadeias de sequências de bases e suas respectivas qualidades associadas a cada base. Arquivos no formato FASTQ possuem um grande volume de dados, como no caso dos arquivos utilizados para este estudo que se aproximam a 12GB de dados, e armazenam a informação gerada pelo sequenciador *Illumina* em formato de texto [Huacarpuma, 2012]. A Figura 4.1 apresenta um trecho do arquivo FASTQ, em que é possível observar um bloco com os dados contidos no arquivo, a saber, a identificação de cada SRS, sua sequência propriamente dita e a caracterização de sua qualidade.

```
@SRR002323.1 080317_CM-KID-LIV-2-REPEAT_0003:3:1:112:566 length=36
TGGGGTTGTGATTTTGATATTGGTGGATTGAGGGTT
+SRR002323.1 080317_CM-KID-LIV-2-REPEAT_0003:3:1:112:566 length=36
IIIIIIIIIIIIIIII=&IIIIIIIIAIIIIIII:
@SRR002323.2 080317_CM-KID-LIV-2-REPEAT_0003:3:1:121:542 length=36
TATGAATATGCAAGAAGGCATCCTGATTACTCTGTC
+SRR002323.2 080317_CM-KID-LIV-2-REPEAT_0003:3:1:121:542 length=36
IIIIIIIIIIIIIIIIIIII0IIIIIIIIIIII6
@SRR002323.3 080317_CM-KID-LIV-2-REPEAT_0003:3:1:123:599 length=36
TCTGTATCTGGTCCTGTGTTACTGTAGTGGTAATTA
+SRR002323.3 080317_CM-KID-LIV-2-REPEAT_0003:3:1:123:599 length=36
IIIIIIIIIIIIIIIIIII8IIIII$IIIIIII-I
```

Figura 4.1: Exemplo de FASTQ.

Visto que o foco deste trabalho é o estudo do banco de dados ArangoDB para armazenar os dados gerados pelo *workflow* e os de proveniência, não se faz necessário aprofundar o processo de obtenção destes dados, visto que são de domínio público. O que deve ser considerado apenas é a integridade dos dados, garantindo que seu conteúdo não seja modificado nem sofra perdas ao longo do processo.

4.2 Modelo de Dados de Proveniência

Os dois tipos básicos do modelo PROV-DM, atividade e entidade, são representados como nós no ArangoDB e diferenciados pela propriedade Tipo. Já as relações são representadas pelas arestas no banco, sendo diferenciadas também por uma propriedade Tipo. Logo, propõe-se a utilização do modelo adaptado do PROV-DM, como apresentado em de Almeida [2014], sendo os nós representados pela Figura 4.2 e as relações pela Figura 4.3. Ressalta-se que os tipos Projeto e Experimento foram adicionados por de Almeida [2014], visto que não pertencem originalmente ao modelo PROV-DM.

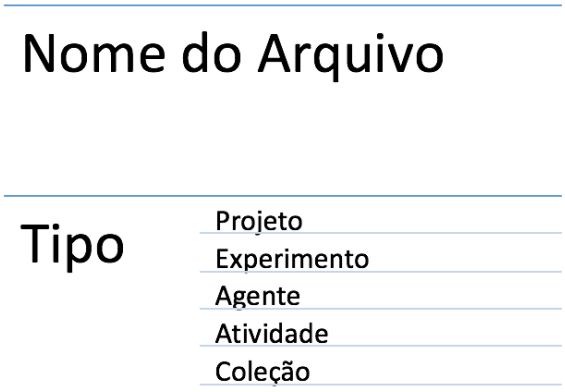

Figura 4.2: Mapeamento dos Nós para o Modelo de Dados do ArangoDB.

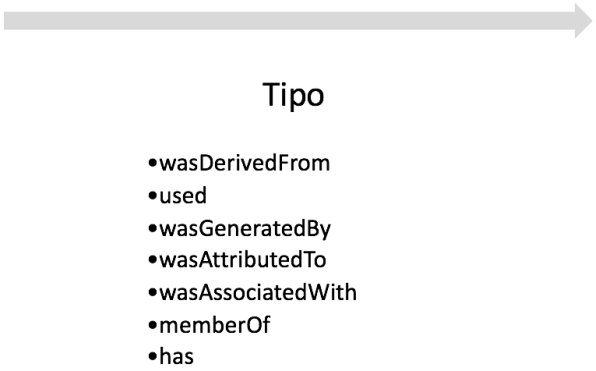

Figura 4.3: Mapeamento das Arestas para o Modelo de Dados do ArangoDB.

A geração dos grafos pelo ArangoDB é realizada automaticamente a partir das coleções existentes no banco, porém é necessário que se definam as relações entre elas. Dessa forma, para armazenar dados gerados durante a execução do *workflow* e de proveniência são necessários, no mínimo, três coleções, sendo uma para os dados propriamente ditos, outra para os dados de proveniência, e por fim uma para estabelecer as relações entre as coleções.

A coleção de dados de proveniência é responsável por armazenar os atributos, tais como nome, descrição, instituição, coordenador, comando, ambiente, anotações dentre outros dependendo do tipo a ser descrito. Os atributos de cada tipo de nó estão representados nas Figuras 4.4 e 4.5.

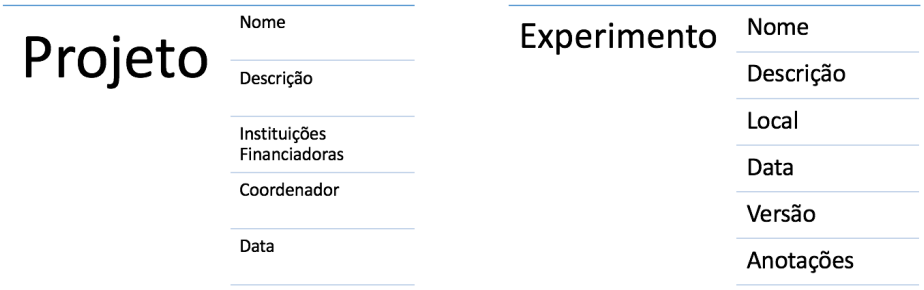

Figura 4.4: Mapeamento dos Tipos para o Modelo de Dados do ArangoDB.

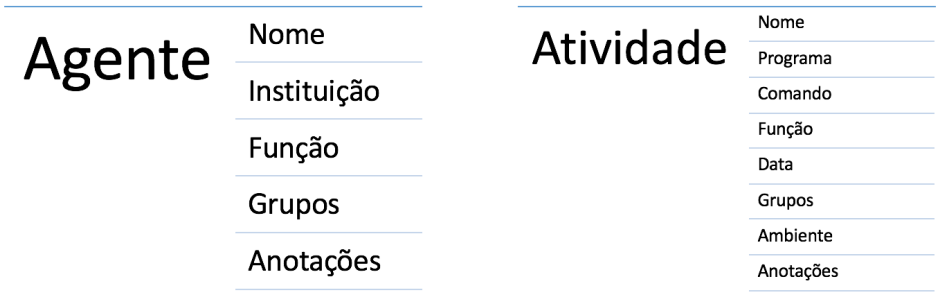

Figura 4.5: Mapeamento dos Tipos para o Modelo de Dados do ArangoDB.

4.3 Desenvolvimento das Ferramentas

Nesta seção é detalhado o processo de desenvolvimento das ferramentas que auxiliaram no tratamento dos dados, e avaliaram os resultados obtidos bem como a descrição do ambiente onde foram realizados os testes. Como requisitos funcionais da aplicação o sistema deve:

- Realizar a conversão dos formatos de arquivo;
- Utilizar a ferramenta nativa de importação do banco de forma automática;
- Associar os dados dos arquivos ao seu respectivo grafo de proveniência;
- Extrair os dados do banco com o conteúdo de um arquivo FASTQ.

Como requisito não-funcional busca-se:

- Utilização das ferramentas nativas do ArangoDB;
- Utilização de arquivos JSON pelo cliente, que faz a inserção e retirada do banco;
- Alto desempenho nas operações de inserção e extração de dados;
- Consistência dos dados.

A metodologia criada para utilização do ArangoDB é dividida em duas etapas: inserção de dados no banco e extração de dados do banco. Considerando que o ArangoDB possui ferramentas de importação e exportação de documentos no formato JSON, preocupou-se com o desenvolvimento de ferramentas para o tratamento dos arquivos de entrada e saída no formato FASTQ, bem como automatização do processo para o desenvolvimento dos testes.

A primeira etapa, ilustrada na Figura 4.6, é dividida em três fases: *parser*, quebra e importação. Sendo assim, o arquivo de entrada FASTQ é convertido para o formato JSON e, em seguida, é quebrado em várias partes de acordo com a número de SRS que estarão presentes nele. Em seguida, utiliza-se a ferramenta de importação do próprio ArangoDB para colocar os arquivos no banco de dados.

A segunda etapa, ilustrada na Figura 4.7, é dividida nas fases de exportação e *reverseParser*. O arquivo gerado através da ferramenta de extração está no formato JSON, sendo necessário que seja feita uma conversão para o formato original FASTQ.

Baseado na métrica de custo de tempo gasto para o armazenamento de dados apresentado na Equação 4.1 [Huacarpuma, 2012], é possível utilizá-la para mostrar a porcentagem de tempo usada no processo de inserção e de extração no ArangoDB em relação ao processo de execução do *workflow* em Bioinformática.

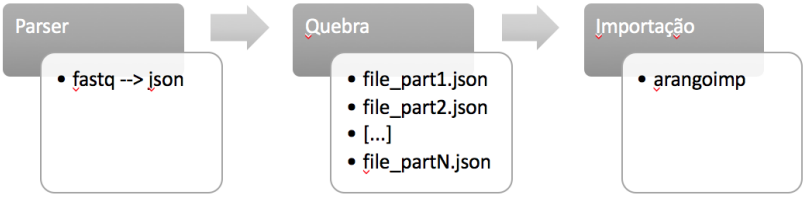

Figura 4.6: Etapa de Inserção.

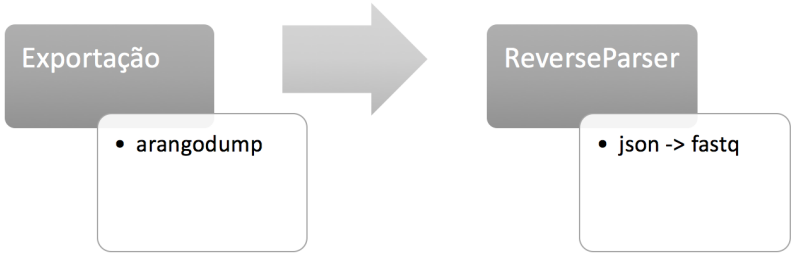

Figura 4.7: Etapa de Exportação.

$$\%TempoGasto = \frac{T_{SGBD}}{TempodoProcesso} * 100 \tag{4.1}$$

Sendo T_{SGBD} a soma do tempo gasto para inserção e exportação no ArangoDB, ou seja, $T_{SGBD} = t_{parser} + t_{quebra} + t_{importação}$ para inserção, em que t_{parser} é o tempo para a etapa de *parser* entre o formato do dado e o JSON, t_{quebra} é o tempo para quebra do arquivo quando necessário e $t_{importação}$ é o tempo de inserir o dado no banco. Para exportação, $T_{SGBD} = t_{exportação} + t_{ReverseParser}$, em que $t_{exportação}$ representa o tempo de extração dos dados e $t_{ReverseParser}$ é o tempo de conversão do formato JSON para o FASTQ.

A seguir é detalhado a função de cada aplicação desenvolvida visando o armazenamento dos dados.

- *Parser*: Levando em conta que o ArangoDB aceita documentos no formato JSON foi necessário o desenvolvimento de um programa que fizesse a conversão a partir do FASTQ. Nesta etapa identificou-se a necessidade de tratamento dos caracteres originais do arquivo que coincidiam com os caracteres especiais utilizados pelo formato JSON. Essa análise foi baseada no padrão de caracteres estabelecidos para o FASTQ [Cock et al., 2010]:

  !"#$%&'()*+,-./0123456789:;<=>?@
  ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_‘
  abcdefghijklmnopqrstuvwxyz{|}~

Esses caracteres são utilizados na definição da sequência de qualidade nos arquivos FASTQ, sendo que a ordem de valor de qualidade é crescente da esquerda para a direita.

Em linguagem C, a primeira aplicação desenvolvida também é responsável pelo tratamento dos caracteres especiais utilizados em comum pelo FASTQ e JSON. Os símbolos que se repetiam nos dois formatos foram trocados por caracteres únicos, tal como "ç"que não é utilizado por nenhum deles. Essa troca só ocorre para não causar conflitos de linguagem com o padrão utilizado pelo banco para a importação de dados. Posteriormente, quando os dados forem extraídos do banco, esses caracteres serão substituídos novamente pelo seu valor original de acordo com a conversão previamente estabelecida.

- Quebra: A segunda aplicação, também desenvolvida em linguagem C, é responsável pela quebra do arquivo resultante da conversão de formatos em blocos menores de arquivos JSON. Essa fase é necessária para que se avalie o desempenho da inserção no banco de acordo com a quantidade de arquivos a serem inseridos e com o tamanho de cada um. A variação do tamanho de cada arquivo está de acordo com a quantidade de SRSs agrupadas em cada arquivo. Testou-se com valores de 250.000, 500.000 e 750.000 SRSs por arquivo. A variação na quantidade de SRSs na inserção de dados também é estudada por Alves [2015].

- Importação: A terceira aplicação está relacionada à utilização da ferramenta nativa de importação do ArangoDB, denominada *arangoimp*. Para que esta ferramenta pudesse importar todos os arquivos gerados pela aplicação anterior para uma mesma coleção de documentos dentro do banco, foi necessário criar um *script* de forma que o processo fosse automatizado e a coleção fosse criada corretamento com todos os dados originais.

- Exportação: Para a exportação dos dados do banco, a ferramenta utilizada foi a nativa disponiblizada pelo ArangoDB, denominada *arangodump*. O formato do arquivo de saída gerado por essa ferramenta é o JSON. Essa ferramenta também pode ser utilizada na criação de *backups* dos dados.

- *ReverseParser*: A última aplicação desenvolvida está relacionada à conversão dos arquivos JSON para o formato FASTQ tratando novamente os caracteres que são comuns aos dois formatos. O desenvolvimento desta aplicação foi realizado na linguagem Perl.

4.4 Arquitetura do Ambiente

A criação de um ambiente distribuído para o ArangoDB é simples e de fácil configuração. Inicialmente, cada máquina deverá ter a aplicação do banco instalada. Também é necessário que se habilite a opção para utilização de *cluster* e altere o local do endereço do servidor de *localhost* para o IP da máquina no arquivo "*arangod.conf*". Após essa configuração, inicia-se a interface web do ArangoDB, escolhe-se a opção de ambiente com várias máquinas e indica-se o endereço IP de cada máquina que fará parte do ambiente.

No entanto, a implementação desta arquitetura foi inviabilizada pela versão atual do ArangoDB, que não permite a utilização da ferramenta de importação em lotes para *clusters*.

Capítulo 5

Resultados e Discussão

Neste capitulo são apresentados os resultados do estudo realizado. A Seção 5.2 descreve os resultados da inserção e extração dos dados da fase de filtragem, enquanto a Seção 5.3 descreve os resultado obtidos na fase de mapeamento. A Seção 5.4 apresenta uma análise sobre a visualização dos grafos de proveniência, e, por fim, a Seção 5.5 apresenta as limitações da versão atual do ArangoDB.

5.1 Ambiente de Testes

Para a avaliação da performance e consistência dos dados no banco, foram realizados testes em duas máquinas, sendo uma máquina comum e a outra um servidor, de acordo com a especificação a seguir:

- Máquina I: Macbook Pro retina, modelo 2014, com processador Intel Core i5 2,6 GHz, 8GB de RAM com o sistema operacional Mac OS na versão Yosemite 10.10.5.
- Máquina II: Servidor HP (Intel(R) Xeon(R) de 8 CPUs de 2.13GHz, 38G de memória RAM com frequência de 1333MHz, 1 HD de 264GB SCSI) sobre o sistema operacional Linux Server Ubuntu/Linaro 4.4.4-14.

Devido a limitações das máquinas, tanto em espaço em disco como em processamento, dividiu-se o estudo conforme as fases do *workflow* de Bioinformática. Dessa forma, utilizou-se a Máquina I para os testes da fase de filtragem e a Máquina II para a fase de mapeamento.

5.2 Fase de Filtragem

Como entrada, inicialmente, foram utilizados seis arquivos FASTQ que tratam do sequenciamento de amostras de células de rim e fígado para identificar a expressão diferencial de

genes em comparação com tecnologias de arranjos existentes [Hoheisel, 2006]. Também foram utilizados na publicação de Guimaraes et al. [2015], no mestrado de Huacarpuma [2012] e no trabalho de graduação de Aniceto and Xavier [2014]. Nas Tabelas 5.1 e 5.2 são apresentadas as características gerais dos arquivos utilizados:

Tabela 5.1: Arquivos Fígado.

Nome	Tamanho	Quantidade de SRS
SRR002321	9,0 GB	54.856.271
SRR002322	4,0 GB	18.437.696
SRR002323	3,2 GB	12.511.940

Tabela 5.2: Arquivos Rim.

Nome	Tamanho	Quantidade de SRSs
SRR002320	6,9 GB	39.266.713
SRR002324	3,8 GB	17.292.434
SRR002325	5,3 GB	27.137.793

Inicialmente, para o armazenamento dos dados gerados na fase de filtragem, foi avaliado o tempo de inserção variando-se a quantidade de SRS presentes em cada arquivo gerado na etapa de quebra, como visto anteriormente na Figura 4.6. Para determinar a quantidade de registros em cada arquivo, baseou-se no estudo de Alves [2015], visto que cada banco de dados possui um comportamento diferente para o tamanho e quantidade de arquivos a serem inseridos. Dessa forma, a ferramenta de quebra de arquivos fez a divisão em 250.000, 500.000 ou 750.000 SRS por arquivo gerando uma quantidade diferente de arquivos para cada caso, conforme mostrado na Tabela 5.3.

Tabela 5.3: Quantidade de Arquivos Gerados na Fase de Quebra.

	Número de SRS		
Arquivo	**250.000**	**500.000**	**750.000**
SRR2320	158	79	53
SRR2321	220	110	74
SRR2322	74	37	25
SRR2323	60	30	20
SRR2324	70	35	24
SRR2325	109	55	37

Para que os dados fossem validados a partir da análise estatística dos resultados obtidos, realizaram-se vários testes para que se obtivesse o tempo médio de inserção dos

arquivos no banco. A Figura 5.1 mostra o tempo de inserção dos dados no banco de acordo com os arquivos que foram quebrados anteriormente. O tempo de inserção está representado eixo vertical e cada arquivo de acordo com suas divisões está representado no eixo horizontal. Percebe-se que a utilização de 250.000 SRS por arquivo é o ponto ideal para inserção, visto que o tempo é reduzido. Apesar do banco ter que inserir mais arquivos provenientes de um lote maior, ele consegue obter um resultado satisfatório. A partir dos dados relatados na Figura 5.1, percebe-se que quando se agrupam mais SRS em um único arquivo, o tempo de importação para o banco aumenta. Apesar de serem realizadas mais inserções de arquivo quando ele contém poucas SRSs, seu tempo de inserção é menor do que no outro caso. Dessa forma, considera-se que ArangoDB tem um melhor desempenho para inserção de arquivos menores, mesmo em um lote maior de inserções.

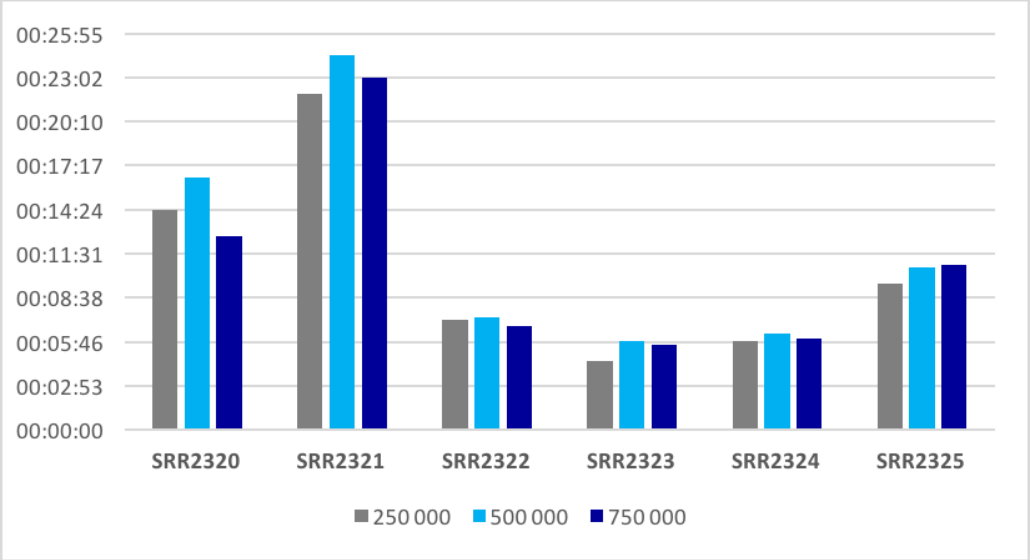

Figura 5.1: Tempo de Inserção de Acordo com a Quebra de Arquivos.

Como pode ser observado, o pior caso ocorreu para a inserção de arquivos que contiam 500.000 SRS. A quantidade de arquivos gerados com a utilização de 500.000 SRS por arquivo está entre a quantidade gerada no caso de 250.000 e 750.000, porém o tempo de inserção para este caso se mostra superior em relação aos outros. A diferença entre o tempo de inserção do caso de 250.000 para o de 750.000 é de apenas 3%, sendo assim considerado que para que a otimização do processo de inserção dos dados gerados pelo *workflow* de Bioinformática deve-se utilizar arquivos com 250.000 SRS, visto que mostraram um tempo menor para a inserção dos dados, mesmo gerando uma quantidade maior de arquivos a serem inseridos no banco.

A partir dos dados relatados na Figura 5.1, percebe-se que quando se agrupam mais SRS em um único arquivo, o tempo de importação para o banco aumenta. Apesar de serem realizadas mais inserções de arquivo quando ele contém poucas SRSs, seu tempo de

inserção é menor do que no outro caso. Dessa forma, considera-se que o ArangoDB tem um melhor desempenho para inserção de arquivos menores, mesmo em um lote maior de inserções.

O tempo necessário para a quebra em uma quantidade maior de arquivos é pequeno em relação ao tempo total do processo de importação de dados. O tempo total de quebra dos arquivos representa apenas 3% do tempo total de inserção de dados no ArangoDB. Na Figura 5.2 pode-se observar o tempo total para a importação de dados de acordo com a metodologia estabelecida anteriormente, mostrando assim o tempo gasto especificamente em cada uma das fases (*parser*, quebra e inserção), utilizando os arquivos divididos a cada 250.000 SRS.

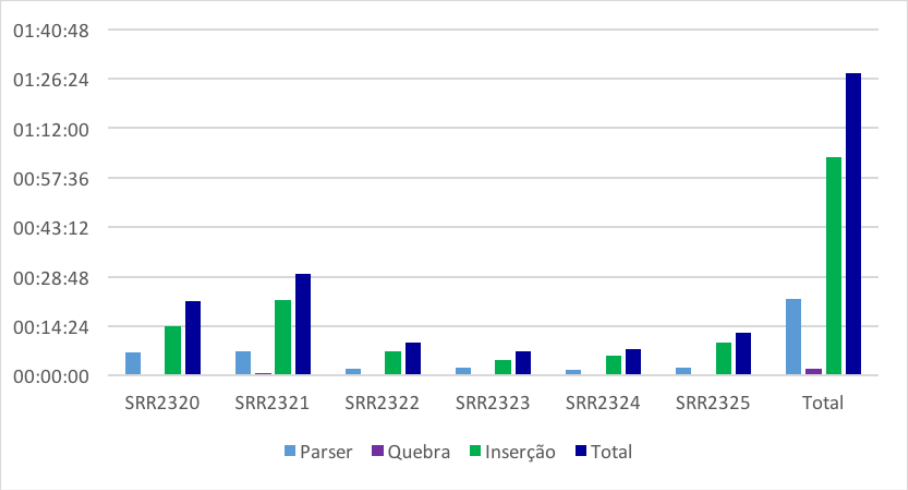

Figura 5.2: Tempo total de Extração dos Dados Filtrados.

Assim como o esperado, nota-se que o maior arquivo (SRR2321 - 9,0GB) foi o que levou o maior tempo para ser inserido no banco. O tempo total para a inserção de todos os arquivos foi de 1h28m02s.

Para a extração dos dados do banco, utilizou-se a ferramenta *arangodump* e, em seguida, o *reverseParser*, que foi desenvolvido para transformar novamente os dados no formato FASTQ. O tempo de extração total dos dados (16m 23s) pode ser visualizado na Figura 5.3.

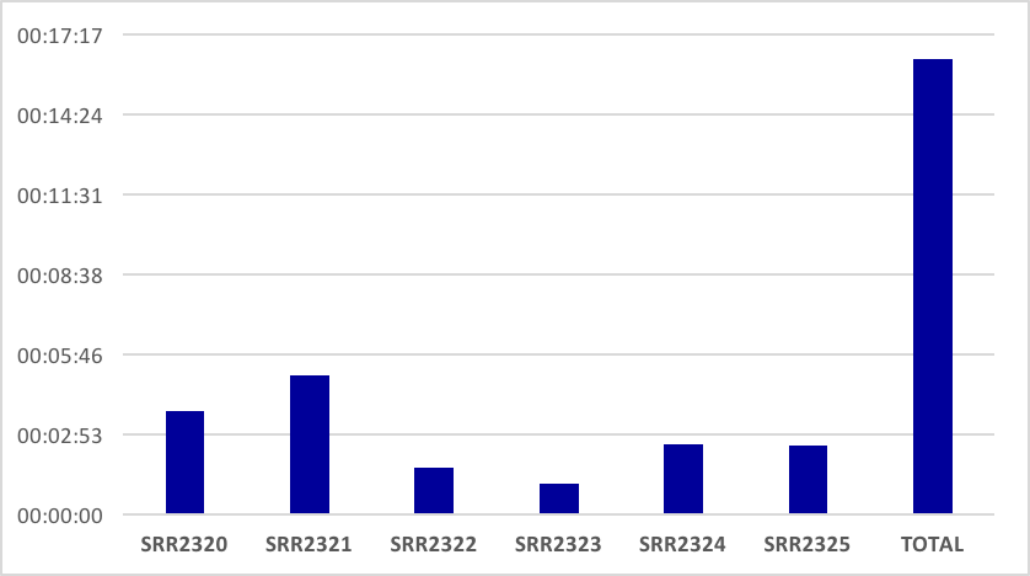

Figura 5.3: Tempo total de extração dos dados filtrados.

5.3 Fase de Mapeamento

Os testes executados com os arquivos gerados pela fase de mapeamento foram realizados na Máquina II. A partir dos dados filtrados, foi realizado o mapeamento dos mesmos com o programa *TopHat*, que precisa de um genoma de referência. Foi utilizado o genoma humano *hg19*. Os arquivos gerados com a execução do *TopHat* possuem o formato *.bam* e estão representados juntamente com seus respectivos tamanhos e tempo de execução na Tabela 5.4.

Tabela 5.4: Mapeamento com TopHat.

Nome	Tamanho	Tempo
2320.bam	961 MB (100.7098.854 bytes)	01h 16m 35s
2321.bam	1.4 GB (1.479.840.810 bytes)	01h 49m 20s
2322.bam	634 MB (664.039.210 bytes)	36m 47s
2323.bam	426 MB (446.415.011 bytes)	34m 48s
2324.bam	532 MB (557.030.821 bytes)	31m 35s
2325.bam	752 MB (787.666.584 bytes)	55m 20s

Os dados mapeados, foram inseridos no ArangoDB. Como os dados de saída do *TopHat* são arquivos no formato BAM, o primeiro passo para inserção desses dados no banco de dados foi a conversão para o formato JSON, como ilustrado na Figura 5.4.

Observa-se na Tabela 5.5 o tempo gasto para conversão de cada arquivo, sendo o maior tempo justamente do maior arquivo gerado pela fase de mapeamento. Percebe-se que o tempo de conversão de arquivos representa 8% em relação ao tempo gasto na execução da fase de mapeamento.

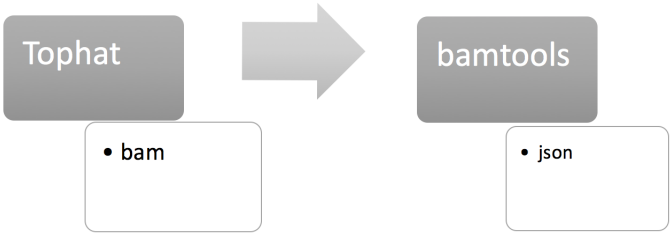

Figura 5.4: Conversão dos Dados Mapeados.

Tabela 5.5: Conversão (BAM para JSON).

Arquivo	Tamanho do arquivo	Tempo execução
2320.json	12.0 GB (12.500.568.728 bytes)	05m 35s
2321.json	17.0 GB (18.233.361.191 bytes)	08m 15s
2322.json	8.5 GB (9.116.976.368 bytes)	04m 00s
2323.json	4.7 GB (5.018.219.644 bytes)	02m 24s
2324.json	7.8 GB (8.302.713.935 bytes)	03m 47s
2325.json	8.8 GB (9.387.935.403 bytes)	04m 23s

Após a conversão dos dados, foi realizado a inserção dos mesmos do ArangoDB, com a ferramenta de importação do banco. Em seguida, os dados foram extraídos. A Figura 5.5 mostra os tempos gastos na inserção e extração desses dados. Percebe-se que o tempo de inserção de dados no ArangoDB é maior do que o de extração, o que dá ao banco um melhor desempenho nas suas operações de extração em relação às suas inserções.

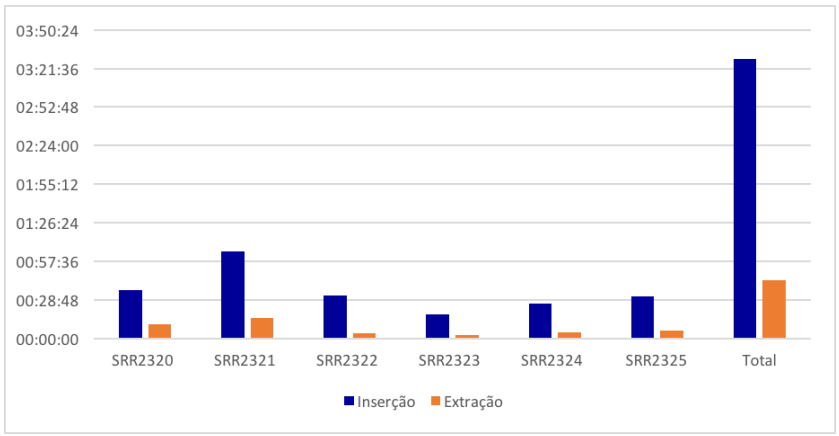

Figura 5.5: Tempo de Inserção e de Extração dos Arquivos Mapeados pelo TopHat.

Considerando uma das fases mais demoradas na execução de um *workflow*, foi escolhida a fase de mapeamento para analisar o custo de armazenamento de dados em um SGBD em relação ao tempo de execução do *workflow*. A Equação 4.1 foi utilizada para obtenção dos valores apresentados na Tabela 5.6.

Tabela 5.6: Comparação de Tempo de Processamento e Armazenamento.

Fase	Processamento	Inserção	Extração
Mapeamento	05h 44m 25s	03h 28m 25s (60,5%)	43m 15s (12,6%)

Percebe-se que o tempo total de inserção e de extração de dados no banco, da fase de mapeamento, representa 60,5% do tempo total gasto no processo de mapeamento, porém ressalta-se ainda que a execução da fase de mapeamento do *workflow* neste estudo reduziu significativamente o tempo de Huacarpuma [2012], visto que a Máquina II teve um incremento de memória RAM desde então e foi utilizada a versão mais recente do *Tophat*.

Apesar do elevado tempo gasto para a utilização do banco de dados, justifica-se um SGBD no armazenamento de dados gerados pelo *workflow* de Bioinformática, pois além de armazenar e recuperar, também é possível fazer a gerência desses dados para utilização em novos experimentos ou fases do *workflow*. Além disso, após a inserção dos dados o tempo para ter acesso aos mesmos é pequeno.

5.4 Grafos

Com o propósito de verificar as ferramentas e características próprias do ArangoDB neste trabalho, não se utilizou nenhuma biblioteca externa, apenas a visualização nativa que o banco oferece em sua interface gráfica.

Além disso, de acordo com de Paula [2013], o modelo PROV-DM pode ser aplicado em *workflow* de Bioinformática, visto que através de um grafo é possível facilmente representar a proveniência em um experimento da Bioinformática. Os nós e arestas, juntamente com seus atributos, fazem toda a representação da proveniência.

A Figura 5.6 também representa a proveniência dos arquivos utilizados na fase de filtragem gerada pelo ArangoDB. É representado os arquivos de origem da fase de filtragem (SRR002320, SRR002321, SRR002322, SRR002323, SRR0023204 e SRR002325), as atividades de filtragem (A001-Filtro-Rim, A002-Filtro-Rim, A003-Filtro-Rim, A004-Filtro-Figado, A005-Filtro-Figado e A006-Filtro-Figado), o agente associado a elas (AG001-Executor) e as coleções geradas pela execução das atividades (SRR002320-filtered, SRR002321-filtered, SRR002322-filtered, SRR002323-filtered, SRR002324-filtered, SRR002325-filtered e SRR002326-filtered).

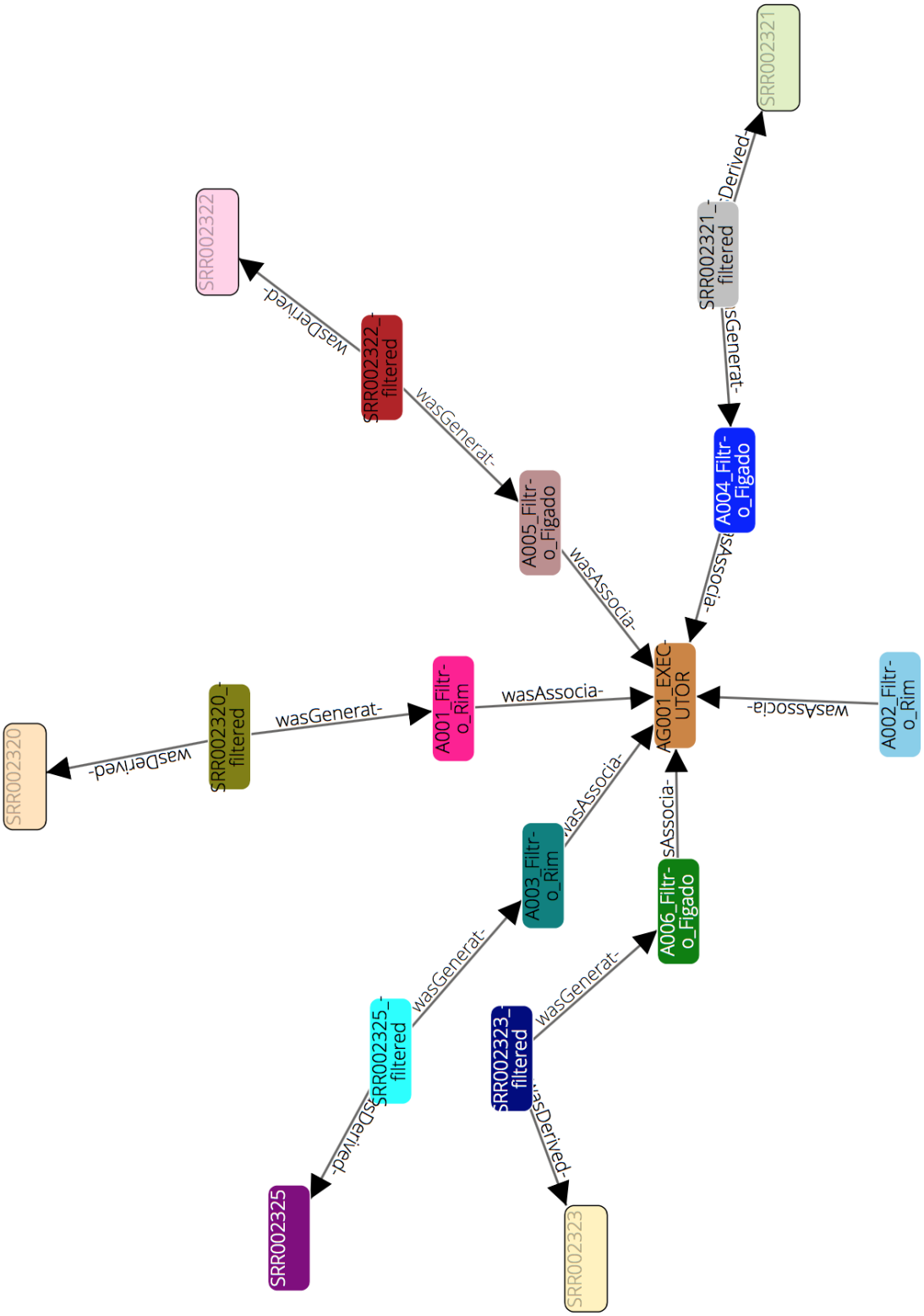

Figura 5.6: Grafo de Proveniência Gerado pelo ArangoDB.

O modelo PROV-DM foi representado no ArangoDB, sendo possível armazenar os nós e arestas com seus respectivos atributos. Todos os tipos de nós, levando em conta os tipos das entidades do modelo PROV-DM, puderam ser descritos e armazenados nas coleções de documentos do banco e as arestas foram armazenadas nas coleções de relações do ArangoDB.

A visualização do grafo não permite a diferenciação das entidades, visto que possui uma forma única de representação (retângulo). Além disso, a visualização através de cores também ocorre de forma limitada, visto que só é possível mudar as cores de cada elemento individualmente, mas não se podendo selecionar uma única cor para representar um conjunto de determinado tipo, como por exemplo, utilizar apenas uma cor para representar todos os arquivos de origem da fase de filtragem.

Levando em consideração que existem ferramentas próprias para otimizar a visualização de grafos, conclui-se que o ArangoDB foi satisfatório no armazenamento de todas as informações de cada entidade, de cada tipo, armazenando assim toda a proveniência do *workflow* de Bioinformática.

5.5 Limitações

A versão atual do ArangoDB (2.7.0) não suporta a utilização da ferramenta nativa de importação de dados quando um *cluster* está implementado. Dessa forma, fica inviabilizado a inclusão dos dados de Bioinformática no banco que deveria ser feito um por um através da interface web ou via console através da ferramenta *arangosh*. Em contato com os desenvolvedores do ArangoDB obteve-se a informação que o foco da versão 3.0, que será lançada em 2016, será a melhoria das operações de *sharding*. Dessa forma, apenas na futura versão será possível utilizar todo o poder computacional deste banco para o armazenamento de grandes quantidade de dados como é o caso da Bioinformática.

Capítulo 6

Conclusões e Trabalhos Futuros

Com o surgimento de novas tecnologias e diferentes tipos de dados, fica difícil utilizar apenas um banco de dados com um modelo rígido de dados. Hoje em dia as aplicações necessitam que os bancos trabalhem em conjunto com o objetivo de obter um ecossistema mais poderoso, capaz e robusto, através da participação da melhor parte de cada um em virtude de um objetivo específico. Além disso, outras dimensões devem ser consideradas na escolha de um banco de dados para um objetivo específico, tais como disponibilidade, consistência, escalabilidade e segurança.

Nesta monografia foi realizado um estudo sobre a utilização do ArangoDB como um SGBD para armazenar dados brutos e de proveniência gerados a partir de *workflows* de Bioinformática. Tomando o ArangoDB, lançado recentemente, armazenaram-se os dados brutos e de proveniência em um mesmo banco. O estudo preliminar analisou as características necessárias para o armazenamento de diferentes modelos de dados em um mesmo banco.

Verificou-se a necessidade de um mesmo banco que pudesse armazenar e relacionar os dois tipos de dados: gerados pelo *workflow* e os dados de proveniência. Buscou-se então por bancos de dados híbridos. A simplicidade de configuração e instalação do ArangoDB estende-se também a diferentes sistemas operacionais, em que se exige apenas a instalação de um simples software para a execução do banco.

Os resultados obtidos possibilitaram otimizar o processo de inserção e extração de dados no ArangoDB através da metodologia e estruturas criadas para utilização do banco. Como proposto, foi possível avaliar o desempenho do banco para inserção e extração através do armazenamento de dados das fases de filtragem e de mapeamento de um *workflow* de Bioinformática. Além disso, verificou-se que o ArangoDB possui um desempenho melhor para extração de dados em relação à inserção dos mesmos.

O modelo de proveniência PROV-DM utilizado em outros trabalhos também foi adaptado para que os dados de proveniência fossem armazenados de forma que se pudesse

visualizar tanto a proveniência quanto os dados que a ela pertencem no mesmo banco de dados.

Tendo em vista que o ArangoDB é de fácil configuração, possui uma interface simples de comunicação, seus formatos de trabalho são equivalentes com outros bancos NoSQL utilizados atualmente, e sua equipe de desenvolvimento é bastante ativa buscando melhorias e corrigindo as falhas existentes, considera-se que foi concluído o objetivo de utilizar o banco para o armazenamento e gerenciamento de dados em *workflows* de Bioinformática, visto que se armazenaram a proveniência e dos dados em um mesmo lugar.

Além disso, também é importante que se realizem os mesmo testes com outros bancos NoSQL no mesmo ambiente para realizar uma comparação de desempenho do quesito de inserção e extração de dados dos bancos. Pode-se utilizar o Segemehl[1] para realizar a fase de mapeamento na execução do *workflow* de Bioinformática, visto que o formato de saída deste programa de mapeamento de dados é o formato SAM e seria necessário um estudo para a conversão de dados para o JSON.

Levando em consideração que a versão 3.0 do ArangoDB suportará a ferramenta de importação para *cluster*, sugere-se que avancem estes estudos para a realização de testes de desempenho em um ambiente computacional com mais máquinas.

Em relação aos grafos, sugere-se um estudo para melhorar sua visualização através de ferramentas e bibliotecas externas, para que as entidades utilizadas do PROV-DM sejam diferenciadas e o entendimento do grafo fique mais claro.

[1] *Software* para mapeamento de SRSs que utiliza genomas de referência.

Referências

Juccelino Rodrigues Alves. Análise de desempenho de bancos de dados relacionais e não relacionais em dados genômicos. 2015. Trabalho de Conclusão de Curso do Departamento de Estatística e Informática da Universidade Federal Rural de Pernambuco. 9, 10, 11, 33, 36

Rodrigo Cardoso Aniceto and Renê Freire Xavier. Um estudo sobre a utilização do banco de dados nosql cassandra em dados biológicos. 2014. Trabalho de Conclusão de Curso do Departamento de Ciência da Computação da Universidade de Brasília. 14, 15, 16, 36

ArangoDB. Arangodb v2.7.0 documentation, 2015. URL `https://docs.arangodb.com/index.html`. Acesso em: 05/08/2015. 17, 20

NoSQL Archive. List of nosql databases, 2015. URL `http://nosql-database.org`. Acesso em: 19/10/2015. 10

Alex Bateman and Matt Wood. Cloud computing. *Bioinformatics*, 25(12):1475–1475, 2009. 14

Toby Bloom and Ted Sharpe. Managing data from high-throughput genomic processing: a case study. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 1198–1201. VLDB Endowment, 2004. 14

Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. Why and where: A characterization of data provenance. In *Database Theory—ICDT 2001*, pages 316–330. Springer, 2001. 4

Mario Bunge. Treatise on basic philosophy: Volume 3: Ontology 1: The furniture of the world. *Reidel, Boston*, page 77, 1977. 6

Anderson Chaves Carniel, Aried de Aguiar Sa, Vinicius Henrique Porto Brisighello, Marcela Ribeiro, Renato Bueno, Ricardo Rodrigues Ciferri, and Cristina Dutra de Aguiar Ciferri. Query processing over data warehouse using relational databases and nosql. In *Informatica (CLEI), 2012 XXXVIII Conferencia Latinoamericana En*, pages 1–9. IEEE, 2012. 10

Rick Cattell. Scalable sql and nosql data stores. *ACM SIGMOD Record*, 39(4):12–27, 2011. 10

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson Hsieh, Deborah Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4, 2008. 11

Peter Cock, Christopher Fields, Naohisa Goto, Michael Heuer, and Peter Rice. The Sanger FASTQ file format for sequences with quality scores, and the solexa/illumina fastq variants. *Nucleic acids research*, 38(6):1767–1771, 2010. 32

Luciana da Silva Almendra Gomes. Proveniência para workflows de bioinformática. *Mestrado, Pontíficia Universidade Católica do Rio de Janeiro, Centro Técnico e Científico, PUC-Rio*, 2011. 12

Susan Davidson and Juliana Freire. Provenance and scientific workflows: challenges and opportunities. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1345–1350. ACM, 2008. 5

Rodrigo Pinheiro de Almeida. Proveniência de dados em workflow de bioinformatica com prov-dm e armazenamento em banco de dados baseado em grafo. 2014. Dissertação do Mestrado em Informática do Departamento de Ciência da Computação da Universidade de Brasília. 1, 6, 15, 16, 28

Mauricio de Diana and Marco Aurélio Gerosa. Nosql na web 2.0: Um estudo comparativo de bancos não-relacionais para armazenamento de dados na web 2.0. In *IX Workshop de Teses e Dissertações em Banco de Dados*, 2010. 9

Renato de Paula. Proveniência de dados em workflows de bioinformática. 2013. Trabalho de Conclusão de Curso do Departamento de Ciência da Computação da Universidade de Brasília. 5, 8, 12, 15, 16, 41

Renato de Paula, Maristela Holanda, Luciana Gomes, Sergio Lifschitz, and Maria Emilia Walter. Provenance in bioinformatics workflows. *BMC bioinformatics*, 14(Suppl 11): S6, 2013. 4

Ewa Deelman, Dennis Gannon, Matthew Shields, and Ian Taylor. Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540, 2009. 1

Daniele El-Jaick, Marta Mattoso, and Alexandre AB Lima. Sgprov: Mecanismo de sumarização para múltiplos grafos de proveniência. In *Simpósio Brasileiro de Banco de Dados–SBBD*, 2013. 15, 16

Dmitrij Frishman and Alfonso Valencia. *Modern genome annotation: the BioSapiens Network*. Springer Science & Business Media, 2009. 13

Mark Greenwood, Carole Goble, Robert Stevens, Jun Zhao, Matthew Addis, Darren Marvin, Luc Moreau, and Tom Oinn. Provenance of e-science experiments-experience from bioinformatics. In *Proceedings of UK e-Science All Hands Meeting 2003*, pages 223–226, 2003. 12

Valeria Guimaraes, Maristela Holanda, Fernanda Hondo, Rodrigo Almeida, Harley Vera, Aleteia Araujo, and Maria Emilia Walter. A study of genomic data provenance in noSQL document-oriented database systems. 2015. 14, 15, 16, 36

Milene Pereira Guimarães and Maria Claudia Reis Cavalcanti. Proveniência de dados na área de bioinformática. 2009. Trabalho de Conclusão da Seção de Engenharia de Computação do IME. 1, 15, 16

Ragib Hasan, Radu Sion, and Marianne Winslett. Introducing secure provenance: problems and challenges. In *Proceedings of the 2007 ACM workshop on Storage security and survivability*, pages 13–18. ACM, 2007. 4

Robin Hecht and Stefan Jablonski. Nosql evaluation: A use case oriented survey. *International Conference on Cloud and Service Computing*, 2011. 14, 16

Jörg D Hoheisel. Microarray technology: beyond transcript profiling and genotype analysis. *Nature reviews genetics*, 7(3):200–210, 2006. 36

Maristela Holanda and Jane Adriana Souza. Query languages in nosql databases. *Handbook of Research on Innovative Database Query Processing Techniques*, page 415, 2015. 10

David Hollingsworth and UK Hampshire. Workflow management coalition the workflow reference model. *Workflow Management Coalition*, 68:26, 1993. 12

Ruben Huacarpuma. Modelo de dados para um pipeline de seqüenciamento de alto desempenho transcritômico. 2012. Dissertação do Mestrado em Informática do Departamento de Ciência da Computação da Universidade de Brasília. 2, 13, 14, 15, 16, 27, 31, 36, 41

Ruben Cruz Huacarpuma, Maristela Holanda, and Maria Emilia Walter. A conceptual model for transcriptome high-throughput sequencing pipeline. In *Advances in Bioinformatics and Computational Biology*, pages 71–74. Springer, 2011. 13, 14

Philipp Kewisch. jCard: The JSON format for vCard. *Internet Engineering Task Force (IETF)*, 2014. RFC 7095. 20

Bernadette Farias Lóscio, Hélio Rodrigues de Oliveira, and Jonas César de Sousa Pontes. Nosql no desenvolvimento de aplicações web colaborativas. *VIII Simpósio Brasileiro de Sistemas Colaborativos*, 2011. 10, 11

Nicholas M Luscombe, Dov Greenbaum, Mark Gerstein, et al. What is bioinformatics? a proposed definition and overview of the field. *Methods of information in medicine*, 40 (4):346–358, 2001. 11

Deborah MacGuinness, James Michaelis, and Luc Moreau. *Provenance and Annotation of Data and Processes: Third International Provenance and Annotation Workshop, IPAW 2010, Troy, USA, June 15-16, 2010: Revised Selected Papers*. Springer, 2010. 6

Marta Mattoso, Cláudia Werner, G Travassos, Vanessa Braganholo, and Leonardo Murta. Gerenciando experimentos científicos em larga escala. *SBC*, page 121, 2008. 9, 14, 16

Marta Mattoso, Claudia Werner, Guilherme Travassos, Vanessa Braganholo, Leonardo Murta, Eduardo Ogasawara, Daniel Oliveira, Sergio Cruz, and Wallace Martinho. Towards supporting large scale in silico experiments life cycle. *Int J Bus Process Integr Manage (IJBPIM)*, 5(1):79–92, 2010. 12

Dan McCreary and Ann Kelly. Making sense of nosql. *Greenwich, Conn.: Manning Publications*, 2013. 9

Luc Moreau and Paolo Missier. Prov-dm: The prov data model. *World Wide Web Consortium*, 2013. 6

Luc Moreau, Juliana Freire, Joe Futrelle, Jim Myers, and Patrick Paulson. Governance of the open provenance model. *URL http://twiki. ipaw. info/pub/OPM/WebHome/governance. pdf*, 2009. Acesso em 13/09/2015. 6

Eduardo Soares Ogasawara. *Uma Abordagem Algébrica para Workflows Científicos com Dados em Larga Escala*. PhD thesis, Universidade Federal do Rio de Janeiro, 2012. 1

Uwe Röhm and José A Blakeley. Data management for high-throughput genomics. In *CIDR*, 2009. 14, 16

Pramod Sadalage and Martin Fowler. *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Pearson Education, 2012. 9

Satya Sahoo and Amit Sheth. *Provenir ontology: Towards a framework for escience provenance management*. The Ohio Center of Excellence in Knowledge-Enabled Computing, 2009. 6

Stephan Schuster. Next-generation sequencing transforms today's biology. *Nature*, 200 (8):16–18, 2007. 2

Stacey Simon, Jixian Zhai, Raja Sekhar Nandety, Kevin McCormick, Jia Zeng, Diego Mejia, and Blake Meyers. Short-read sequencing technologies for transcriptional analyses. *Annual review of plant biology*, 60:305–333, 2009. 27

Lincoln Stein. Genome annotation: from sequence to biology. *Nature reviews genetics*, 2 (7):493–503, 2001. 13

Wang Chiew Tan. Research problems in data provenance. *IEEE Data Eng. Bull.*, 27(4): 45–52, 2004. 5

Peter Chen Wong. Active conceptual modeling of learning. 2007. 6