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

Proveniência de dados em workflow de Bioinformática com PROV-DM e armazenamento em banco de dados baseado em grafo

Rodrigo Pinheiro de Almeida

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientadora
Prof.ª Dr.ª Maristela Terto de Holanda

Brasília
2014

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado em Informática

Coordenador: Prof. Dr. Ricardo Pezzoul Jacobi

Banca examinadora composta por:

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

CIP — Catalogação Internacional na Publicação

Almeida, Rodrigo Pinheiro de.

Proveniência de dados em workflow de Bioinformática com PROV-DM e armazenamento em banco de dados baseado em grafo / Rodrigo Pinheiro de Almeida. Brasília : UnB, 2014.
63 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2014.

1. Proveniência de dados, 2. computação em nuvem, 3. NoSql, 4. Neo4J.

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 em workflow de Bioinformática com PROV-DM e armazenamento em banco de dados baseado em grafo

Rodrigo Pinheiro de Almeida

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

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

Prof. Dr. Prof. Dr.
CIC/UnB CIC/UnB

Prof. Dr. Ricardo Pezzoul Jacobi
Coordenador do Mestrado em Informática

Brasília, 25 de Fevereiro de 2014

Agradecimentos

Primeiramento a Deus, pela saúde e força de vontade. A minha esposa Thayres que esteve sempre ao meu lado, me estimulando e encorajando nos momentos de fraqueza. A minha orientadora Prof.ª Dr.ª Maristela Terto de Holanda pelas sábias orientações, ajuda inestimável, paciência e principalmente pela oportunidade dada. E em especial aos meus pais e minha tia Maria de Lurdes, que sem eles nada disso seria possível.

Resumo

Muitos experimentos científicos na bioinformática são executados como *workflows* computacionais. Cada vez mais, esses experimentos estão sendo executados em ambientes de computação em nuvem devido as facilidades oferecidas. Para a validação e reconhecimento de um experimento é necessário reexecutá-lo sob as mesmas circustâncias que foi originado em um ambiente de computação em nuvem, algumas informações importantes para reexecução de um experimento são ocultadas, portanto sendo necessário realizar a proveniência dos dados para obter metadados importantes sobre o ambiente e execução dos experimentos. Contudo, os bancos de dados relacionais não apresentam um bom desempenho em ambientes de nuvens computacionais, originando um novo paradigma de bancos de dados, os *NoSQL*. Esta pesquisa apresenta uma arquitetura capaz de realizar a proveniência de dados de experimentos científicos na bioinformática utilizando o modelo PROV-DM em ambientes de nuvens computacionais e armazenando os metadados de proveniência em bancos de dados *NoSQL*

Palavras-chave: Proveniência de dados, computação em nuvem, NoSql, Neo4J.

Sumário

Lista de Figuras

Capítulo 1

Introdução

1.1 Contextualização

Projetos de Bioinformática consistem na execução de diversos experimentos com sequências de DNA ou RNA obtidas por sequenciadores de alto desempenho tais como *Illumina* [Bentley, 2006] ou 454 *Roche* [Rothberg and Leamon, 2008]. Estes equipamentos são capazes de gerar, em poucas horas, milhões de fragmentos de DNA [Schuster, 2007] que precisam ser analisados. Esses milhões de fragmentos podem gerar *terabytes* de dados, que são armazenados em diferentes arquivos com diversos formatos e envolvem a utilização de diferentes ferramentas computacionais cujas configurações e parâmetros de entrada podem afetar fortemente os resultados obtidos.

O gerenciamento da execução desses experimentos é realizada normalmente através de um *workflow* científico, que é uma abstração que define as etapas de execução de um experimento e a sequência em que tais etapas ocorrem, de forma a corroborar ou refutar uma hipótese científica [Jarrard, 2001]. Cada etapa (i.e atividade do *workflow*) envolve a execução de diversos programas, que são responsáveis pela transformação dos dados de entrada e a produção de dados de saída. Entretanto para que um experimento seja válido sob o ponto de vista científico, o seu resultado deve ser passível de reprodução por terceiros, logo é importante armazenar dados tanto do ambiente de execução quanto dos experimentos propriamente ditos. Estes dados podem ser obtidos a partir de dados de proveniência seja em ambientes centralizados ou distribuídos [Altintas et al., 2006].

A proveniência de dados visa descrever os acontecimentos e insumos utilizados na geração de uma determinada informação. Segundo [Buneman et al., 2001] proveniência de dados é "... a descrição das origens de uma peça de dados e do processo pelo qual ela chegou em um banco de dados." (livre tradução), ou seja, para garantir a proveniência de dados se faz necessário guardar tanto a origem dos dados utilizados como matéria-prima, quanto os processos que transformaram estes dados no produto final.

Com o objetivo de fornecer uma estrutura para os dados de proveniência, facilitando o armazenamento e a recuperação são criados modelos, que definem como as informações são representadas. Existem na literatura alguns modelos de dados para a proveniência de dados. Logo, ao manipular dados de proveniência é necessário saber as informações de proveniência suportadas e o modo como estas informações serão acessadas. Para uma definição e comparação mais detalhadas dos modelos de proveniência pode-se ver em [de Paula et al., 2013].

Dentre os modelos de proveniência presentes na literatura, o PROV-DM tem se destacado, já que o principal objetivo é descrever as pessoas, recursos e atividades envolvidas na produção de uma peça de dado, criando condições para que a proveniência seja demonstrada e trocada entre diferentes sistemas.

O ambiente de computação em nuvem tem se tornado atraente para a execução de experimentos científicos devido a escabilidade, interoperabilidade e a idéia de recursos infinitos. Porém, informações como *cluster*, nodos, bancos de dados usados, parâmetros de entrada e saída, tempo de execução, métodos invocados e processos iniciados e finalizados são importantes, porque há a necessidade de validar o experimento e reproduzí-lo.

Para um ambiente tão heterogêneo, distribuído e de alta disponibilidade como o de computação em nuvens, os bancos relacionais não apresentam um bom desempenho, surgindo os bancos de dados *NoSQL*, que gerenciam grades volumes de dados e, em geral, fornecem garantias de consistência fraca, estruturas e interfaces simples. Como um tipo de bancos de dados *NoSQL*, se destacam os bancos de dados de grafos que permitem o armazenamento de entidades e também relacionamentos entre essas entidades.

Já existem trabalhos na literatura que ressaltam a importância da proveniência de dados no ambiente de computação em nuvem, como em [Imran and Hlavacs, 2013] e em projetos de Bioinformática, como pode ser visto em [Muniswamy-Reddy et al., 2010]. Portanto, esse trabalho propõe realizar a proveniência de dados em projetos de Bioinformática utilizando o modelo PROV-DM, armazenando os dados em bancos de dados de grafos no ambiente de computação em nuvem.

1.2 Objetivos

Esse trabalho propõe a definição de uma arquitetura de proveniência de dados para um ambiente de computação em nuvem no contexto da Bioinformática, utilizando o modelo de proveniência PROV-DM e bancos de dados de grafo.

1.2.1 Objetivos Específicos

No intuito de atingir o objetivo geral, foram definidos alguns objetivos específicos:

- Definir uma arquitetura de proveniência de dados para um ambiente de computação em nuvem em projetos de Bioinformática, utilizando bancos de dados de grafos;
- Implementar a arquitetura proposta;
- Realizar estudo de caso com *workflows* científicos reais da Bioinformática;
- Avaliar os resultados obtidos.

1.3 Estrutura do Trabalho

Este documento está estruturado nos capítulos a seguir:

- O Capítulo 2 apresenta o referencial teórico necessário para o desenvolvimento desa pesquisa, tais como a proveniência de dados, o modelo PROV-DM, o armazenamento de dados na nuvem baseados em *NoSQL* e os trabalhos relacionados.

- O Capítulo 3 define uma política de proveniência de dados para um ambiente de computação em nuvem no contexto da Bioinformática.
- Capítulo 4, apresenta a metodologia e o cronograma de trabalho.

Capítulo 2

Fundamentação Teórica

Neste capítulo são tratados os conceitos relacionados a proveniência de dados em um ambiente de computação em nuvem com a seguinte estrutura: A Seção 2.1 faz uma pequena revisão da definição e de modelos de proveniência. A Seção 2.2 mostra as alternativas de bancos de dados utilizados na computação em nuvem. E por último a Seção 2.3 apresenta alguns trabalhos presente na literatura sobre proveniência de dados.

2.1 Proveniência de Dados

O termo Proveniência de dados diz respeito à origem ou procedência dos dados. A proveniência também estar relacionada à auditoria, triagem, linhagem e origem do dado [Davidson and Freire, 2008]. A proveniência ajuda a responder questões sobre os dados, tais como: quem criou este dado e quando, quando o dado foi modificado e por quem e qual foi o processo usado para criar o dado.

De acordo com [Davidson and Freire, 2008], a proveniência pode ser dividida em três tipos:

- Prospectiva: trata-se da sequência de processos utilizados (receita) para a geração do dado, ou seja, captura os passos que devem ser seguidos para a geração de um dado produto.
- Retrospectiva: trata-se das informações obtidas durante a execução dos processos de geração do dado. Compreende desde o tempo de duração de cada atividade executada até a origem dos dados de entrada. Além disso, não depende do tratamento da proveniência prospectiva para ser utilizado. Em outras palavras, é como se fosse um *log* detalhado da execução de uma tarefa.
- Dados definidos pelo usuário: qualquer informação que o usuário julgar necessária para futura utilização. Como exemplo, pode-se citar anotações, conclusões a respeito do processo e, até mesmo, observações sobre parâmetros utilizados.

Segundo [Tan, 2004], a obtenção da proveniência pode seguir duas abordagens, abordagem preguiçosa (*lazy*) na qual a obtenção da proveniência é executada somente no momento que é solicitada, e abordagem ansiosa (*eager*) na qual a proveniência é obtida durante a geração da informação e é armazenada para permitir futuras consultas.

Cabe ressaltar que as duas abordagens podem ser combinadas, fazendo com que algumas informações sejam obtidas pela abordagem preguiçosa e outras pela ansiosa.

Ainda segundo [Davidson and Freire, 2008], quando a proveniência é capturada de forma automática, pode-se dividir o nível em que é feita a captura conforme segue:

- *Workflow*: envolve a descrição da execução de um processo, ou seja, das tarefas que dele fazem parte, é usado pela grande maioria das soluções com SGWfC (Sistemas de Gerência de *Workfow*) e nesse caso deve ser adaptado para capturar os dados dos diferentes processos executados;
- Atividade: pode ocorrer de duas formas. Na primeira, cada processo executado é alterado para capturar os dados de proveniência. Na segunda, podem ser criados programas específicos 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.

Modelos de proveniência de dados tem como principal objetivo fornecer uma estrutura para que os dados de proveniência possam ser armazenados e recuperados, mantendo seu significado e potencializando os seus benefícios. Para modelar os dados de proveniência foram especificados alguns modelos, tais como:

- *W7*: foi apresentado por [Ram and Liu, 2007] e tem como base a ontologia de Bunge [Bunge, 1977], a qual objetiva descrever as propriedades de um objeto de caráter geral. A partir deste estudo, o modelo *W7* estruturou a proveniênica de uma peça de dado através da resposta a 7 perguntas (ou dimensões): O que?, Quem?, Quando?, Onde?, Como?, Qual?, e Por quê?.
- *Provenance Vocabulary*: descrito por [Hartig and Zhao, 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 úteis 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. Da mesma forma que no modelo *Provenance Vocabulary*, o modelo *Provenir Ontology* 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 modelo aberto voltado a caracterização da proveniência de qualquer "coisa", material ou imaterial. 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 através de um grafo acíclico direcionado.

2.1.1 PROV-DM - Provenance Data Model

O PROV-DM teve a sua primeira versão desenvolvida em outubro de 2011 sendo uma recomendação do W3C. A sua versão mais nova foi publicada em abril de 2013. Este modelo tem como principal função descrever as pessoas, entidades e atividades envolvidas na produção de uma peça de dado ou de um objeto qualquer. Sendo assim, o modelo cria as condições para que a proveniência seja demonstrada e trocada entre diferentes sistemas. O modelo PROV-DM tem como principal característica demonstrar a proveniência de qualquer objeto (real ou imaginário) através de um grafo direcionado. 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 modelo é dividido em seis componentes que contém tanto os elementos como as relações possíveis entre eles [W3C, 2014].

- Entidades: entidades (*Entities*) podem representar qualquer objeto (real ou imaginário);
- Atividades: atividades (*Activities*) é algo que ocorre ao longo de um período de tempo e atua sobre ou com entidades;
- Agente e responsabilidades: Agentes (*Agents*) 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;
- 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.
- Plano: Representa um conjunto de ações ou passos que um Agente deve seguir para chegar a um determinado objetivo
- Conta: Representa um conjunto de informações (tipos e relações) que compõe um grafo de proveniência.

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 os tipos Coleção e Plano, uma vez que representam subtipos do tipo Entidade. O tipo Conta não tem símbolo pois representa o próprio grafo de proveniência.

As relações 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. A seguir são descritas as algumas relações:

- *wasDerivedFrom*: indica, de forma geral, que uma Entidade (original) foi usada, direta ou indiretamente, na geração de outra Entidade (derivada);

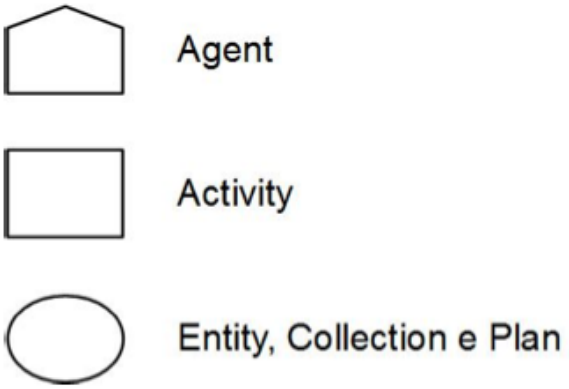

Figura 2.1: Representação gráfica dos diferentes tipos no modelo PROV-DM [W3C, 2014].

- *used*: indica que uma Entidade foi usada por uma Atividade;
- *wasGeneratedBy*: indica que uma Entidade foi gerada por uma Atividade;
- *wasAttributedTo*: atribui algum tipo de responsabilidade a um Agente sobre uma Entidade;
- *wasAssociatedWith*: atribui algum tipo de responsabilidade a um Agente sobre uma Atividade;
- *actedOnBehalfOf*: indica que um Agente está agindo em nome de outro.

A fim de permitir um histórico mais detalhado das derivações feitas durante o processo representado pela proveniência, a relação *wasDerivedFrom* possui diversos subtipos.

A seguir tem-se a descrição dos subtipos da relação *wasDerivedFrom* [W3C, 2014]:

- *wasQuotedFrom*: indica que a Entidade derivada foi gerada a partir da cópia de parte da Entidade original;
- *hadOriginalSource*: indica que uma Entidade corresponde a fonte da informação que corresponde a Entidade derivada;
- *wasRevisionOf*: indica que a Entidade derivada foi gerada a partir da revisão da Entidade derivada, sendo que tal responsabilidade da revisão pode ser atribuída a um Agente;
- *tracedTo*: indica, de forma genérica, que existe uma relação de dependência entre duas Entidade, sem especificar qual é a derivada nem qual é a original.

O tipo de restrição mais importante é chamada restrição de ordem dos eventos. Neste tipo de restrição se pressupoem que os elementos do grafo de proveniência são dotados de características temporais representadas por propriedades como hora inicial ou final e que podem ser avaliadas e validadas. Tal validação pretende garantir que um grafo de proveniência represente uma ordem de acontecimentos possíveis.

As restrições são dividias em três grupos: restrições de Atividade, restrições de Entidade e restrições de Agente. A seguir são descritas as restrições relevantes para este trabalho:

- Restrições de Atividade - restrições relacionadas à execução das Atividades. Exemplos:
  - Início/Fim - o início da execução de uma Atividade deve preceder o seu fim;
  - Uso: o uso de uma Entidade por uma Atividade deve ocorrer entre o início e o fim da sua execução;
  - Geração - a geração de uma Entidade por uma Atividade deve ocorrer entre o início e o fim da sua execução.
- Restrições de Entidade - restrições relacionadas ao ciclo de vida de uma Entidade. Exemplos:
  - Geração/Uso: a geração de uma Entidade deve preceder o seu uso;
  - Derivação/Uso/Geração: para os casos em que existe uma derivação entre duas Entidades, por exemplo E2 é derivado de E1, e o uso de E1 é conhecido, então o uso de E1 deve preceder a geração de E2;
  - Derivação/Geração/Geração: para os casos em que existe uma derivação entre duas Entidades, por exemplo E2 é derivado de E1, e o uso de E1 não é conhecido, então a geracção de E1 deve preceder a geração de E2.
- Restrições de Agente - restrição relacionada ao ciclo de vida de um Agente:
  - Associação: a associação entre um Agente e uma Atividade deve ocorrer entre o início e o fim da execução desta Atividade.

O modelo PROV-DM fornece um recurso para adição de informações extras no grafo de proveniência através de um identificador chamado Nota (*Note*). Este identificador representa um conjunto de pares atributo-valor que permite ao usuário criar diversos tipos de anotações. Cada Nota, por sua vez, pode ser conectada a qualquer tipo ou relação existente no grafo a partir de uma relação chamada *hasAnnotaion*. A Figura 2.2 mostra um exemplo de grafo baseado no modelo PROV-DM. Na figura podem ser vistas quatro anotações, sendo duas relacionadas às arestas *wasAssocitedWith* informando o papel de cada Agente na Atividade, uma relacionada com a Atividade e outra relacionada com a Entidade gerada.

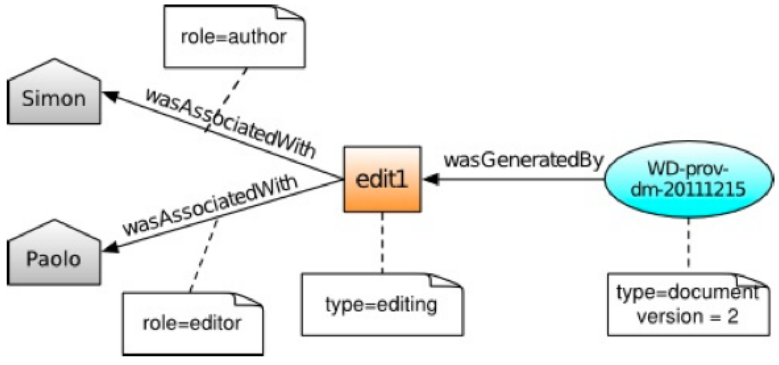

Figura 2.2: Exemplo de grafo do modelo PROV-DM que utiliza Nota [W3C, 2014].

A escolha do PROV-DM deu-se baseada em [de Paula et al., 2013] onde é apresentada uma comparação entre os principais modelos de proveniência de dados disponíveis na literatura. Os principais requisitos considerados foram:

- 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 vindo a ser recomendação desta instituição em 2013;
- Capacidade do modelo de proporcionar o intercâmbio de informações entre diferentes sistemas.

Dessa forma, de acordo com [de Paula et al., 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 Armazenamento de Dados na Nuvem baseado em NoSQL

O modelo de computação em nuvem foi desenvolvido com o objetivo de fornecer serviços de fácil acesso e de baixo custo, garantindo três benefícios. O primeiro benefício é reduzir o custo na aquisição e composição de toda infraestrutura necessária para atender as empresas. O segundo é a flexibilidade que esse modelo oferece no que diz respeito à adição e troca de recursos computacionais, podendo escalar tanto em nível de *software* como em nível de *hardware*. O último benefício é prover uma abstração e facilidade de acesso aos usuários destes serviços.

Infraestruturas, plataformas e *software* estão sendo disponibilizados como serviços, sendo estes fornecidos por ambientes de Computação em Nuvem, no qual empresas e usuários podem alugar capacidade de computação e armazenamento de forma transparente e sob demanda. Estas empresas e usuários estão movendo seus dados e aplicações para a nuvem de forma a acessá-los a qualquer momento e independente de localização. Entretanto, este novo modelo de computação requer grandes mudanças nos sistemas de gerenciamento de dados, pois estes sistemas necessitam de escalabilidade, disponibilidade, desempenho e custo [Sousa et al., 2010].

Com a popularização da internet e o aumento na geração de dados, o volume se torna um grande desafio para os bancos de dados relacionais convencionais que se mostraram ineficientes, uma vez que não possuem escabilidade quando existe uma grande quantidade de

dados armazenados. Dessa forma, aspectos de armazenamento de dados, processamento de consultas e controle transacional têm sido flexibilizados por algumas abordagens para garantir a escabilidade, mas ainda não existem propostas na literatura que combinem estes aspectos de forma a melhorar o desempenho sem comprometer a consistência dos dados [Sousa et al., 2010].

Surgiram alguns sistemas não-relacionais que gerenciam grandes volumes de dados e, em geral, fornecem garantias de consistência fraca, estruturas e interfaces simples, que foram classificados como bancos *NoSQL* (*Not Only SQL*).

Os bancos *NoSQL* podem ser classificados em quatro classes gerais [Padhy et al., 2011]:

- Chave/Valor: os dados são armazenados como pares chave-valor que são indexados para recuperação por chaves. Os mais populares são o *Riak* [Riak, 2014], *Redis* [Redis, 2014], *Memcached*, *Berkeley DB* [Olson et al., 1999], *Amazon DynamoDB* [Dynamodb, 2014], *Porject Voldemort* [Voldemort, 2014] e *HamsterDB* [HamsterDB, 2014].

- Orientado a coluna: armazena os dados em linhas com colunas associadas, fazendo uso de uma chave de linha. Famílias de colunas são grupos de dados relacionados que, frequentemente, são acessados juntos [Fowler and Sadalage, 2013]. Exemplos são o *BigTable* [Chang et al., 2008], *Cassandra* [Cassandra, 2014] e *HBase* [HBase, 2014].

- Armazenamento baseado em documentos: os dados são armazenados e organizados como uma coleção de documentos. Eles armazenam documentos baseados no formato *JSon*, XML, BSON, entre outros. Exemplos são o *MongoDB* [MongoDB, 2014], *Apache CouchDB* [CouchDB., 2014] e *RavenDB* [RavenDB, 2014].

- Bancos de dados de grafos: atividades sobre bancos de dados de grafos surgiram na primeira metade dos anos noventa e depois desapareceram. Resurgindo recentemente devido a projetos onde este estilo de base de dados é necessário (por exemplo química, biologia, mineração e semântica web)[Angles, 2012]. Eles permitem armazenar entidades e também relacionamentos entre essas entidades. Entidades também são conhecidas como nodos, os quais possuem propriedade. Os relacionamentos são conhecidos como arestas que podem ter propriedades. Exemplos são *Neo4J* [Neo4J, 2014], *Infinite Graph* [InfiniteGraph, 2014] ou *FlockDB* [FlockDB, 2014].

2.2.1 Modelo de Dados de Grafo

Os bancos de dados relacionais tem sido a principal solução adotada pela indústria por décadas, porém muitas aplicações estão buscando modelos de dados alternativos, devido a ineficiência do modelo relacional quando se precisa realizar muitas junções com tabelas grandes.

Um outro desafio é o armazenamento de dados em aplicações que tem colaboração interativa e compartilhamento de informações, tais como *Facebook*, *Linkedln* e *Flickr*. Outras aplicações onde bancos de dados de grafos podem ser aplicados é a interação genética, onde nós representam proteínas, e as arestas entre os nós representam as interações físicas entre proteínas. Aplicações que envolvem a análise de redes para sistemas de recomendação, com o objetivo de apresentar informações que possam ser interessantes para os usuários com base em conhecimento prévio extraído do ambiente social.

O modelo de dados de grafo é uma estrutura na qual o esquema e/ou instâncias são modeladas como um grafo dirigido, possivelmente rotulado, ou generalizações da estrutura de dados do grafo, onde a manipulação de dados é expressa por operações orientadadas para o grafo e construtores de tipos, e restrições de integridade que podem ser definidas sobre a estrutura do grafo [Angles and Gutierrez, 2008].

Normalmente, os bancos de dados de grafos são aplicados onde a informação sobre a interconectividade entre os dados ou topologia é mais importante, ou tão importante, quanto os dados. Tem-se em [Angles and Gutierrez, 2008] as principais vantagens de se usar esse tipo de modelo:

- Permite uma modelagem mais natural dos dados porque as estruturas de grafos são visíveis para o usuário.
- Permite expressar as consultas em um nível maior de abstração.
- Permite a implementação de algoritmos eficientes para realizar específicas operações.

A representação de entidades e relações é fundamental para os modelos de grafos. Uma entidade ou objeto representa algo que existe como uma unidade simples e completa. Uma relação é uma propriedade ou predicado que estabelece uma conexão entre duas ou mais entidades. No entanto, dependendo da aplicação em particular, o modelo grafo pode ser diferente, mostrando um grau de complexidade diferente.

Em [Dominguez-Sal et al., 2011], tem-se uma descrição de alguns componentes importantes como:

- Atributos: além de nós e arestas, bancos de dados de grafos armazenam informações associadas a esses nós e arestas. A informação é tipicamente uma *string* ou valores numéricos, que indicam as características da entidade ou relação. Para o caso particular de arestas, alguns grafos incluem atributos numéricos que quantificam a relação, na qual é geralmente interpretada como o comprimento, peso, custo ou a intensidade da relação. Além disso, muitas aplicações podem definir um identificador exclusivo para cada nó e aresta do grafo com o objetivo de rotular cada elemento gráfico;
- Direcionamento: dependendo do problema a relação entre dois nós pode ser simétrica ou não. Se a relação é simétrica, as duas pontas da aresta são diferentes, mas indistinguíveis, ou seja, não há ponta inicial nem ponta final. Se a relação não é simétrica, é possível diferenciar as duas pontas da aresta, em outras palavras, a ponta inicial da aresta é o nó a partir do qual começa a aresta, e a ponta final da aresta é o nó na qual a aresta termina;
- Nós e arestas rotuladas: em algumas aplicações, é possível diferenciar rótulos (ou tipos) de nós e arestas. Rotulagem tem um impacto importante porque alguns aplicativos requerem distinção entre os diferentes tipos de relações. Por exemplo, uma rede social pode aceitar relações ”positivas” ou ”negativas” de amizade;
- Multigrafos: diferem dos grafos em que dois nós podem ser conectados por várias arestas. Multigrafos aparecem geralmente quando grafos possuem arestas diferenciadas porque muitas vezes dois nós estão relacionados por diferentes categorias. Por

exemplo, em uma rede de telefonia móvel, que representa os telefones celulares como os nós e ligações telefônicas como as arestas, dois nós terão múltiplas conexões;

- Hipergrafos: um hipergrafo é uma generalização do conceito de um grafo, no qual as arestas são substituídos por hiperarestas. Uma hiperaresta liga um número arbitrário de nós. Hipergrafos são utilizados, por exemplo, para a construção de modelos de inteligência artificial.

Ainda, de acordo com [Dominguez-Sal et al., 2011], há vários tipos de operações importantes em grafos. Essas operações não são típicas de um domínio específico, mas são operações que podem ser necessárias em algum contexto. As principais operações são:

- Travessias: são operações que se iniciam de um único nó e explora recursivamente os vizinhos até que uma condição final, tais como a profundidade ou visitar um nó de destino seja alcançado;
- Análise Gráfica: basicamente inclui o estudo da topologia de grafos para analisar a sua complexidade e para caracterizar objetos do grafo. É basicamente usado para verificar algumas distribuições de dados específicos, para avaliar um padrão específico, ou para obter informações detalhadas sobre o papel de nós e arestas;
- Componentes: um componente conectado é um subconjunto de nós do grafo onde existe um caminho entre qualquer par de nós. Assim, apenas um nó pertence a um único componente conectado do grafo. Encontrar componentes conectados geralmente é crucial em muitas operações, normalmente utilizados em uma fase de pré-processamento. Além disso, algumas operações são úteis para estudar a vulnerabilidade de um grafo, ou a probabilidade de separar um componente conectado em dois outros componentes;
- Comunidades: é geralmente considerada como um conjunto de nós, onde cada nó é mais próximo dos nós dentro da comunidade do que dos nós fora dela;
- A correspondência de padrões: reconhecimento de padrões lida com algoritmos que visam reconhecer ou descrever padrões de entrada. Padrões de grafos são geralmente classificados em exato ou aproximado;
- Equivalência estrutural: encontrar semelhança entre nós em um grafo tem se mostrado muito importante em problemas de análise de redes socias, ou seja, refere-se à medida em que nós têm um conjunto de ligações com outros nós do sistema;
- Transformação: compreende as operações que alteram o banco de dados de grafo: cargas de um grafo, adicionar/remover nós ou arestas dos grafos, criar novos tipos de nós/arestas/ atributos ou modificar o valor de um atributo. O resto das consultas são consideradas consultas de análise;
- Acesso em cascata: uma operação segue um padrão em cascata se a consulta executa operações com uma profundidade de pelo menos dois nós. Já uma operação que não seja em cascata pode acessar apenas um nó, uma aresta ou vizinhos de um nó;
- Escala: classificam-se as consultas, dependendo do número de nós acessados. Pode-se distinguir dois tipos de consultas, consultas globais e de proximidade. Em outras

palavras, considera-se como consultas globais aquelas que acessam todos os nós ou as arestas do grafo, já as de proximidade acessam apenas uma parte do gráfico;

- Atributos: bancos de dados não só tem que gerir a informação estrutural do grafo, mas também dos dados associados às entidades do grafo. Aqui, classificam-se as consultas de acordo com o conjunto de atributos que ela acessa: conjunto de atributos da aresta, atributos do nó, atributos do nó e da aresta ou sem atributos;
- Resultado: diferenciam-se três tipos de resultados: grafos, agregados e conjuntos. A saída mais comum para uma consulta de banco de dados grafo é outro grafo, que é normalmente uma transformação, uma seleção ou uma projeção do grafo original, que inclui nós e arestas. O segundo tipo de resultados é a construção de agregados, cuja aplicação mais comum é resumir as propriedades do grafo. Finalmente, um conjunto é uma saída que contém tanto as entidades atômicas ou conjuntos de resultados que não estão organizados na forma de grafos.

Há uma grande variedade de modelos de bancos de dados de grafos, sendo que cada modelo de grafo é otimizado para um conjunto específico de funcionalidades ou consultas. Segundo [Buerli and Obispo, 2012], os bancos de dados de grafos podem ser classificados em:

- Bases de dados de grafos: fornecem um modelo tradicional de objetos para nós e relacionamentos.
- Bases de dados de grafos distribuídas: abstrai a distribuição de grandes grafos.
- Bases de dados de grafos baseados em chave-valor: simplificam o modelo de objetos relacionados aos bancos de dados de grafos para permitir uma maior escalabilidade horizontal. Esses modelos são contruídos sobre um banco de chave-valor existente.
- Bases de dados de grafos baseados em documentos: adiciona a complexidade de bancos de dados baseados em documentos a um nó de uma base de dados de grafo.
- Bases de dados de grafos baseados em SQL: a estrutura de um grafo é criado sobre um banco de dados SQL.
- Bases de dados de grafos baseados em *Map-Reduce*: permite a manipulação de grandes grafos, tem o objetivo de aumentar ao máximo o paralelismo, particionando os nós do grafos entre diversas máquinas.

2.2.2 Banco de dados relacional versus Banco de dados de grafo

Um estudo feito por [Vicknair et al., 2010] compara uma base de dados relacional, como MySQL, com o modelo de grafos, tal como *Neo4J*, para armazenamento de dados de proveniência. Ele propõe um *benchmark* objetivo e uma comparação subjetiva baseada na documentação e experiência. Os testes objetivos incluem a velocidade de processamento em um conjunto de consultas pré-definidas, requisitos de espaço em disco e escabilidade. Já, os testes subjetivos incluem maturidade/nível de suporte, facilidade de programação, flexibilidade e segurança.

A comparação objetiva mostrou que os bancos de dados de grafos apresentaram melhor desempenho do que os bancos de dados relacionais em consultas estruturais e que

envolvem busca de textos. Entretanto, devido ao fato do mecanismo de indexação usado em bancos de grafos ser baseado em *strings*, as consultas que envolvem contagem numérica apresentaram menor eficiência.

Já a comparação subjetiva mostrou que em relação ao nível de maturidade/suporte os bancos de dados de grafos, em geral, possuem um mercado menor, consequentemente menos usuários, ausência de uma linguagem unificada e suporte. Em relação à facilidade de programação, depende do problema, para consultas transversais no grafo é mais simples em bancos de dados de grafos enquanto procurar valores de atributos em uma tabela é extremamente fácil com o modelo relacional.

Quanto a flexibilidade o modelo de grafos, especificamente o *Neo4J*, tem um esqema facilmente mutável. O esquema do modelo relacional pode ser alterado, porém fazer isso é mais complexo do que no modelo de grafos. E por fim, quanto a segurança, o modelo relacional possui suporte interno a multi-usuários, por outro lado, muitas bases de dados de grafos não possuem suporte para ambientes multi-usuários. Portanto, de acordo com [Vicknair et al., 2010], o modelo de dados de grafos ainda se apresenta prematuro para ser executado em um ambiente de produção.

Em seu livro *Neo4J In Action*, [Partner et al., 2013] executou um experimento comparando um banco de dados relacional com o banco de dados de grafos *Neo4J*. O experimento consite em encontrar amigos de amigos em uma rede social com uma profundidade de no máximo 5 nós, ou seja, dada duas pessoas escolhidas aleatoriamente, há um caminho que as conecta com no máximo 5 relações? A base de dados foi populada com um 1.000.000 de pessoas, sendo que cada uma com aproximadamente 50 amigos.

Conforme mostra a Tabela 2.1, em uma profundidade de dois nós (amigos de amigos), ambos os modelos de dados apresentam uma boa performance. Já com uma profundidade de 3 nós (amigo de amigo de amigo) os bancos de dados de grafos conseguem reduzir o tempo de execução da consulta consideravelmente. E finalmente, para encontrar todos os amigos até uma profundidade de cinco nós, os bancos de dados relacionais simplesmente não conseguiram finalizar a consulta, enquanto o *Neo4J* retorna o resultado em aproximadamente dois segundos.

Tabela 2.1: Comparação MySQL versus Neo4J. [Partner et al., 2013]

Profundidade	Tempo de execução no MySQL (s)	Tempo de execução no Neo4J (s)	Registros retornados
2	0.016	0.01	2.500
3	30.267	0.168	110.000
4	1543.505	1.359	600.000
5	—	2.132	800.000

2.3 Trabalhos Relacionados

A seguir são apresentados alguns trabalhos relacionados a proveniência de dados que focam no modelo de proveniência usado, ambiente de execução no qual a proveniênca é capturada e sistema de armazenamento de dados utilizado.

Pode-se ver a importância da proveniência em ambientes de computação em nuvem em [Muniswamy-Reddy et al., 2010], que defende a proveniência como primeiro conjunto de dados a ser salvo trazendo benefícios tanto para os provedores de armazenamento (como *Amazon*) como para os usuários. Os provedores de armazenamento em ambiente de computação em nuvem podem tirar proveito das informações inerentes a proveniência de dados para diversas aplicações que vão desde a aplicações de segurança, pesquisa a melhoria de desempenho. Por sua vez, os usuários reduzem esforço de desenvolvimento extra para o armazenamento da proveniência.

Em [Muniswamy-Reddy and Seltzer, 2010] trata sobre os benefícios de realizar proveniência na computação em nuvem, principalmente em projetos científicos, que é importante para o usuário saber informações como *cluster*, nodos, bancos usados, parâmetros de entrada e saída, tempo de execução, métodos invocados e processos iniciados e finalizados, porque há a necessidade de reproduzir o experimento, ajustar os parâmetros de entrada do *workflow* e comparar os resultados. Tudo isso com o objetivo de obter a validação do projeto.

Em [Paulino et al., 2010], tem-se uma abordagem que apoia a coleta de metadados de proveniência em experimentos científicos, baseada na evolução da arquitetura *Matrioska* para o ambiente de nuvens. Além disso, também apresenta um modelo de dados capaz de armazenar os metadados de proveniência específicos da nuvem baseado no modelo *Open Provenance Model*. Para validação do trabalho foi aplicado como estudo de caso um *workflow* de Mineração de Texto (MT) concebido e executado com o SGWfC (Sistema de Gerência de *Workflow*) *Vistrails* e como resultado conseguiu-se capturar um conjunto inicial de dados que não podiam ser coletados somente com os mecanismos locais de proveniência dos SGWfC, tais como, os identificadores de produtos de dados e a identificação das instâncias utilizadas (endereço IP), instância e o tipo de banco de dados e quais usuários executaram o *workflow*.

Uma abordagem desenvolvida na forma de um componente de software chamada *ReproeScience* para reprodução do ambiente onde o experimento computacional foi originalmente executado, de forma que o mesmo possa ser instanciado sob demanda e reproduzido em iguais condições foi proposto por [de Oliveira et al., 2012]. O *software* desenvolvido usa um modelo de dados baseados no modelo de proveniência PROV-DM. O *ReproeScience* coleta dados de proveniência por meio de sinais do sistema operacional, tais como interrupções e *system calls*, em seguida executa uma atividade de clonagem do ambiente original gerando imagens de máquinas virtuais prontas para serem reproduzidas em uma estação de trabalho ou na nuvem.

Devido a grande quantidade de dados gerados durante um experimento científico, pode ser inviável analisar os dados após a execução. Por isso, [Costa et al., 2013] propõe uma solução para monitorar o experimento durante a sua execução usando dados de proveniência que podem ser capturados tanto em ambientes *dekstop* como de computação em nuvem. A análise da proveniência em tempo de execução permitirá aos cientistas monitorar o *workflow* e tomar ações antes do seu final. O modelo de proveniência PROV-DM foi usado como base para a geração da modelagem do banco de dados. Para a captura foram desenvolvidos diversos componentes que executam em paralelo com o *workflow* e importam os dados gerados pelos diferentes SGWfC para a base de dados PROV-Wf. Uma vez todas as informações de proveniência importada dentro do modelo PROV-Wf, consultas podem ser realizadas em tempo de execução para realizar a análise do

experimento e assim sejam tomadas medidas de controle caso algum erro ocorra.

Em um cenário onde a execução de *workflow* produz uma grande quantidade de dados, sua execução pode levar dias ou semanas, até mesmo em ambientes de computação de alto desempenho. Além disso, considerando o fato que esses *workflows* podem ser executados em ambientes de computação em nuvem, onde há a necessidade de reduzir custos financeiros. Nesse contexto, uma forma de melhorar a performance é reduzir os dados a serem processados, ou seja, estabelecer eventos que possam determinar se os dados produzidos são válidos ou não para serem consumidos pela próxima atividade no *workflow*. Em [Gonçalves et al., 2013] é proposto uma forma aos cientistas de eliminar dados intermediários com pouca qualidade analisandos dados de proveniência em tempo de execução do *workflow*. Para isso foi desenvolvido um componente de software, chamado *Provenance Analyzer* que permite extrair dados de proveniência de arquivos intermediários (baseados em critérios definidos e submitidos) e filtrar esses dados baseados em critérios também fornecidos. Para a validação da proposta, foram executado diversos experimentos na nuvem da *Amazon*, obtendo resultados que melhoraram o tempo de execução do *workflow* em até 36.2%.

É proposto em [de Paula et al., 2013] o uso de proveniência de dados com base no modelo PROV-DM para fluxos de trabalho de projetos genoma, permitindo aos cientistas estudar detalhes de suas experiências e, sempre que necessário re-executá-los de forma mais planejada e controlada. Para validação da proposta foi desenvolvido um simulador de proveniência para facilitar a inclusão e armazenamento dos dados. Observa-se que a proveniência de dados é feita manualmente, semelhante a um livro de registo, onde o cientista registra os dados durante a realização da experiência e os dados são salvos em arquivos *xml*.

Muitos experimentos científicos na área de Bioinformática são executados através de *workflows* computacionais devido ao fato de facilitar a implementação e análise. Entretanto, a quantidade de dados gerados aumenta em cada fase dificultando a identificação dos dados e rastreabilidade das transformações. Portanto, é necessário criar novas ferramentas para verificar automaticamente quais recursos e parâmetros foram usados para gerar os resultados e outras informações importantes para validação e publicação do experimento. Em [Pinheiro et al., 2013] é proposto a captura automática de dados e o armazenamento em um esquema relacional baseado no modelo PROV-DM. Para a validação da proposta, o simulador implementado em [de Paula et al., 2013] foi modificado para realizar a captura automática dos dados de proveniência e salvá-los em um banco de dados relacional. Além disso, foi executado um estudo de caso que lida com sequências de DNA de bactérias extremófilos *Alfa-amylase* obtida a partir de dados UniProt [UNIPROT, 2012], tendo por objetivo encontrar o genoma que codifica para uma proteína em particular no genoma de Bacillus, assim como comparar a sequência encontrada com outros genomas disponíveis. Como resultado conseguiu-se gerar um grafo de proveniência de toda a execução do *workflow*.

Em [Woodman et al., 2011] é encontrado uma descrição de como um sistema de armazenamento de proveniência é usado pelo *e-Science Central* (uma plataforma portável de computação em nuvem instalada tanto em nuvens privadas como públicas, incluindo *Amazon EC2* e *Windows Azure*) que pode ser usado para responder questões importantes para a pesquisa científica. Como por exemplo, qual a proveniência de uma peça de dado e quais os efeitos ao mudar as versões de um artefato? Aproveitando a estrutura de arma-

zenamento de proveniência e controle de versão disponibilizados pelo *e-Science Central* é possível responder a essas questões, assim como executar o *workflow* em uma versão antiga e comparar os resultados com versões mais recentes. O modelo de proveniência usado é o OPM, que foi implementado com um banco de dados de grafo, *Neo4J*, devido a facilidade de realizar consultas transversais sobre a estrutura do grafo.

Em [Korolev and Joshi, 2014] usa uma ferramenta de captura de dados de proveniência para alcançar a reprodução de experimentos científicos envolvidos com *workflow* de *big data*, o PROB. A ferramenta é baseada no *Git*, *Git-Annex* e *Git2Prov*. O *Git* é usado como um sistema de controle de versão e um repositório dos dados de proveniência. O *Git-Annex* é uma extensão do *Git* que permite rastrear informações de versão de grandes arquivos sem buscá-los dentro do repositório. O *Git2Prov* é uma ferramenta para converter os metadados armazenados no *Git* para o formato PROV. Além disso, o PROB define sua própria ontologia para representar a informação de proveniência, que foi baseada na ontologia do *Git2Prov* e do *NiPype*, que é usado para expressar fatos sobre análise de dados dos *workflows* e ambientes de execução.

A seguir tem-se a Tabela 2.2, que resume todo o referencial teórico em termos do ambiente de execução, modelo de proveniência e sistema de armazenamento utilizado. Como pode ser observado a maioria dos modelos de proveniência usados são OPM ou PROV-DM, o ambiente de computação em nuvem está sendo utilizado para desenvolvimento de sistemas de *workflow* e poucos trabalhos na literatura utilizam bancos de dados de grafos para armazenar dados de proveniência.

Tabela 2.2: Resumo referencial teórico.

Proposta	**Modelo de Proveniência**	**Ambiente de execução**	**Sistema de armazenamento**
Provenance for the Cloud	—	Computação em nuvem	Storage
Provenance as first class cloud data	—	Computação em nuvem	Storage
Captura de Metadados de Proveniência para Workflows Científicos em Nuvens Computacionais	OPM	Computação em nuvem	Banco de dados relacional
Reprodução de Experimentos Científicos Usando Nuvens	PROV-DM	Dekstop	Banco de dados relacional
Capturing and querying workflow runtime provenance with PROV: a practical approach	PROV-DM	Dekstop e Computação em nuvem	Banco de dados relacional
Performance analysis of data filtering in scientific workflows	—	Computação em nuvem	Banco de dados relacional
Provenance in bioinformatics workflows	PROV-DM	Dekstop	Arquivos XML
Automatic capture of provenance data in genome project workflows	PROV-DM	Dekstop	Banco de dados relacional
Achieving reproducibility by combining provenance with service and workflow versioning	OPM	Computação em nuvem	Banco de dados de grafo
Prob: A tool for tracking provenance and reproducibility of big data experiments	PROV-DM	Computação em nuvem	Sistema de versionamento *GitHub*

Capítulo 3

A Arquitetura Proposta

Este capítulo é dividido em três seções, a primeira realiza uma contextualização da proposta, a segunda o mapeamento das entidades e relacionamentos do modelo PROV-DM para o modelo de um banco de dados de grafo e a terceira especifica a arquitetura de armazenamento de proveniência.

3.1 Contextualização da Proposta

Como apresentado no capítulo anterior, existem na literatura trabalhos usando os modelos de proveniência OPM e PROV-DM, armazenamento dos dados de proveniência em bancos de dados de grafos e proveniência de dados para processos executados em um ambiente de computação em nuvem, porém não tratam de um modelo de dados para proveniência de *workflow* de Bioinformática usando bancos de dados de grafos para armazenar os dados do modelo PROV-DM com execução em um ambiente de computação em nuvem. Neste contexto, os principais aspectos que levaram a definição desta arquitetura são:

- Como apresentado em [de Paula et al., 2013], o modelo PROV-DM pode ser aplicado em *workflow* de Bioinformática onde através de um grafo é possível facilmente representar a proveniência em um experimento da Bioinformática;
- Como o PROV-DM é um modelo baseado em grafo, onde toda proveniência pode ser representada através de nós e arestas, este trabalho propõe como contribuição o armazenamento dos dados de proveniência em um banco de dados de grafo;
- A execução do *workflow* em um ambiente de computação em nuvem é uma realidade em vários experimentos na Bioinformática, devido as característica de elasticidade, redução de custo na aquisição e composição de toda infraestrutura e prover uma abstração e facilidade de acesso aos usuários.

3.2 Modelo de dados de proveniência no NoSQL baseado em bancos de dados de grafo

Banco de dados de grafos permitem o armazenamento de entidades e relacionamentos entre essas entidades, que também são conhecidas como nodos, os quais possuem

propriedades. Os relacionamentos são conhecidos como arestas que também podem ter propriedades. As arestas têm significância direcional; nodos são organizados por relacionamentos, os quais permitem encontrar padrões entre eles.

Sendo assim os dois tipos básicos do modelo PROV-DM, atividade e entidade serão representados como nodos no banco de dados de grafo, e serão diferenciados pela propriedade Tipo. Já as relações serão representadas pelas arestas no banco de dados de grafo, sendo diferenciadas também por uma propriedade Tipo. Logo, teremos o seguinte modelo:

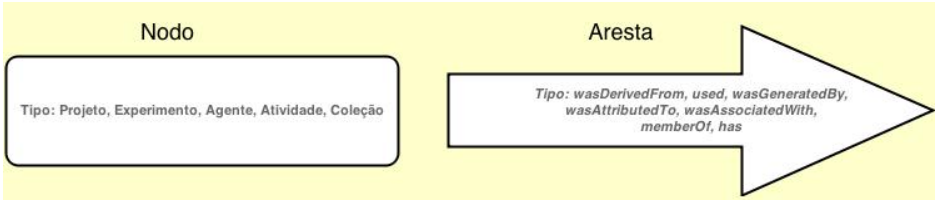

Figura 3.1: Mapeamento dos tipos e relações para o modelo de dados baseado em grafo.

Na Figura 3.2 tem-se o mapeamento mais detalhado dos tipos de entidades do modelo PROV-DM e dados de proveniência para o banco de dados de grafo. Pode-se observar o acréscimo de duas entidades que não pertencem ao modelo PROV-DM: Projeto e Experimento, são necessárias porque existe a necessidade de agrupar vários experimentos em um mesmo projeto para facilitar o compartilhamento do grafo de proveniência gerado. Essa mesma idéia é aplicada entre as entidades Experimento e Atividades.

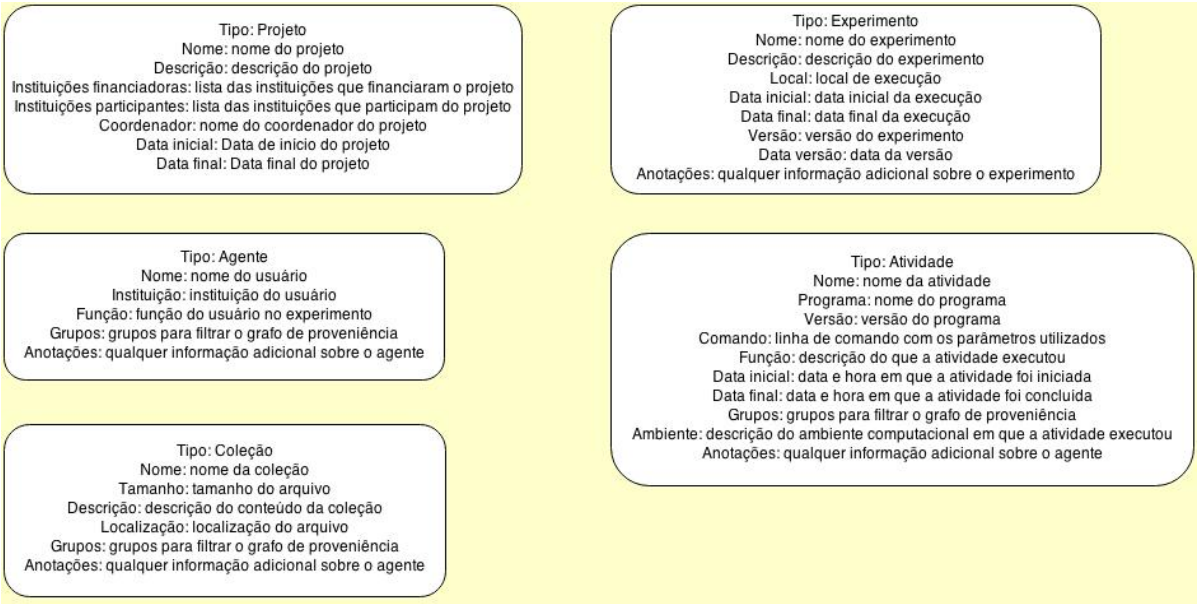

Figura 3.2: Mapeamento dos tipos e relações para um modelo de grafos.

3.3 Arquitetura proposta

O objetivo desta seção é definir uma arquitetura para a coleta de proveniência de dados em projetos de bioinformática de forma automática para o ambiente de computação em nuvem usando o modelo PROV-DM.

Na Figura 3.3 pode-se ver os principais componentes da arquitetura:

- Uma interface web simples e amigável para o usuário, permitindo-o criar projetos, experimentos e atividades. Os dados informados pelo usuário serão enviados para a nuvem para serem processados;
- Módulo de execução: responsável por executar os comandos ou programas recebidos do usuário;
- Módulo de captura de dados de proveniência: responsável por receber os dados informados pelo usuário e interpretá-los e realizar a captura da proveniência de forma automática;
- Módulo de armazenamento: será responsável pela comunicação com o banco de dados de grafo e prover uma interface de acesso;
- Banco de dados de grafo: responsável pelo armazenamento dos dados na forma de grafo.

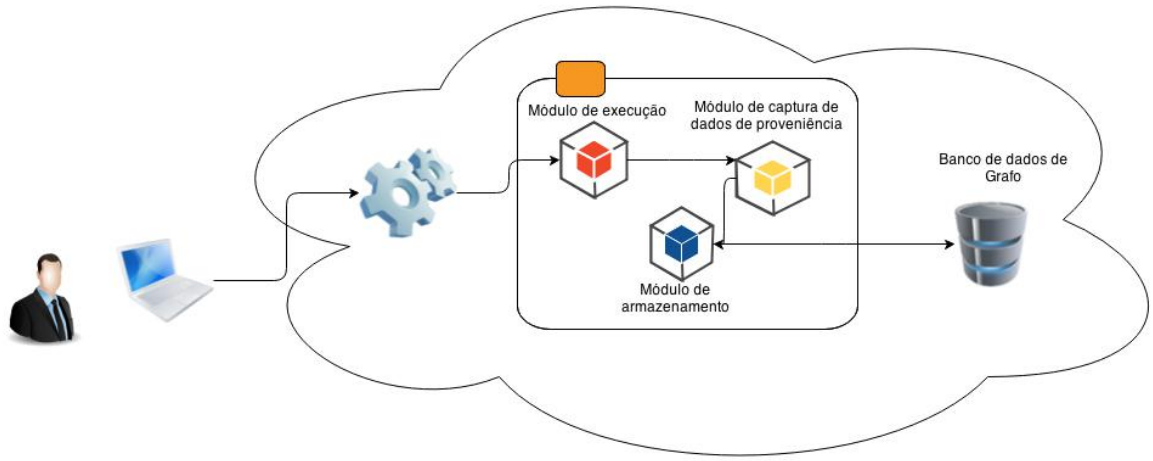

Figura 3.3: Arquitetura Proposta.

Capítulo 4

Metodologia e Cronograma

4.1 Metodologia

4.1.1 Primeira fase: estudo e análise

Inicialmente, será realizado um estudo sobre proveniência de dados, computação na nuvem e bancos *NoSQL*, mais especificamente bancos de grafos. O objetivo dessa fase é identificar os principais desafios na captura automática de proveniência de dados na computação na nuvem.

4.1.2 Segunda fase: especificação da arquitetura

Na segunda fase deve ser especificado um modelo de arquitetura que atenda os desafios definidos na primeira fase. Essa fase envolve a especificação e uma política de proveniência de dados na nuvem no contexto da Bioinformática usando o modelo de proveniência PROV-DM.

4.1.3 Terceira fase: implementação

Na terceira fase serão definidas as características técnicas para o desenvolvimento da arquitetura, ou seja, as ferramentas necessárias para a construção da arquitetura proposta na segunda fase.

4.1.4 Quarta fase: testes

Nesta fase pretende-se definir os casos de uso a serem executados e executá-los em diversas nuvens computacionais com o objetivo de validar e realizar ajustes necessários em relação a arquitetura proposta na terceira fase.

4.1.5 Quinta fase: avaliação e correção

Na fase cinco pretende-se avaliar o desempenho da arquitetura proposta nos casos de uso definidos na quarta fase, assim como fazer as devidas correções e reavaliações da proposta para que os objetivos sejam cumpridos.

4.1.6 Sexta Fase: publicação e defesa da dissertação

Nesta fase pretende-se discutir os resultados conseguidos, apresentar artigos acadêmicos e elaborar a dissertação do mestrado. O objetivo desta fase é a apresentação do trabalho à comunidade através de meios formais e a defesa da dissertação.

4.2 Cronograma de Execução

Durante esse primeiro ano do mestrado foram completados os 24 créditos de disciplinas necessários para a obtenção do título de mestre de acordo com o regimento do Programa de Pós-graduação em Informática da UnB, faltando apenas a defesa da dissertação do mestrado.

Nesse período foi realizada a publicação do artigo [Pinheiro et al., 2013] que propõe a captura automática de dados de proveniência e o armazenamento em um esquema relacional baseado no modelo PROV-DM. Para a validação da proposta, o simulador implementado em [de Paula et al., 2013] foi modificado para realizar a captura automática dos dados de proveniência e salvá-los em um banco de dados relacional.

Neste período também foi realizado um levantamento bibliográfico, a definição dos objetivos da pesquisa e a definição da arquitetura proposta. As etapas desse projeto são apresentadas na Tabela 4.1, a saber:

1. Pesquisa bibliográfica para conhecer a teoria relacionada com o projeto, detalhamento do projeto de pesquisa;
2. Definição das ferramentas de implementação e arquitetura que será executada na nuvem;
3. Implementação da arquitetura definida;
4. Testes;
5. Discussão, avaliação e correção;
6. Elaboração da dissertação, escrita de artigos científicos e defesa da dissertação.

Tabela 4.1: Cronograma de atividades.

Etapa	**2013/1**	**2013/2**	**2014/1**	**2014/2**
1	X	X	X	
2		X	X	
3			X	
4			X	X
5				X
6				X

Referências

[Altintas et al., 2006] Altintas, I., Barney, O., and Jaeger-Frank, E. (2006). Provenance collection support in the kepler scientific workflow system. pages 118–132. 1

[Angles, 2012] Angles, R. (2012). A comparison of current graph database models. In *Data Engineering Workshops (ICDEW), 2012 IEEE 28th International Conference on*, pages 171–177. IEEE. 10

[Angles and Gutierrez, 2008] Angles, R. and Gutierrez, C. (2008). Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1):1. 11

[Bentley, 2006] Bentley, D. R. (2006). Whole-genome re-sequencing. *Current opinion in genetics and development.*, 16(6):545–552. 1

[Buerli and Obispo, 2012] Buerli, M. and Obispo, C. P. S. L. (2012). The current state of graph databases. *Department of Computer Science, Cal Poly San Luis Obispo, mbuerli@ calpoly. edu*. 13

[Buneman et al., 2001] Buneman, P., Khanna, S., and Wang-Chiew, T. (2001). Why and where: A characterization of data provenance. pages 316–330. 1

[Bunge, 1977] Bunge, M. (1977). Treatise on basic philosophy: Volume 3: Ontology 1: The furniture of the world. *Reidel, Boston*. 5

[Cassandra, 2014] Cassandra (Acessado em: 06 de Abril de 2014.). Disponível em: `http://cassandra.apache.org/`. 10

[Chang et al., 2008] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. E. (2008). Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4. 10

[Costa et al., 2013] Costa, F., Silva, V., de Oliveira, D., Ocaña, K., Ogasawara, E., Dias, J., and Mattoso, M. (2013). Capturing and querying workflow runtime provenance with prov: a practical approach. pages 282–289. 15

[CouchDB., 2014] CouchDB. (2014.). Disponível em: `http://couchdb.apache.org.` 10

[Davidson and Freire, 2008] Davidson, S. B. and Freire, J. (2008). Provenance and scientific workflows: challenges and opportunities. pages 1345–1350. 4, 5

[de Oliveira et al., 2012] de Oliveira, A. H. M., de Souza Martins, M., Modesto, I., de Oliveira, D., and Mattoso, M. (2012). Reprodução de experimentos científicos usando nuvens. *Anais do XVII Simpósio Brasileiro de Banco de Dados (SBBD 2012)*. 15

[de Paula et al., 2013] de Paula, R., Holanda, M., Gomes, L. S., Lifschitz, S., and Walter, M. E. M. (2013). Provenance in bioinformatics workflows. *BMC Bioinformatics*, 14(Suppl 11):S6. 1, 9, 16, 19, 23

[Dominguez-Sal et al., 2011] Dominguez-Sal, D., Martinez-Bazan, N., Muntes-Mulero, V., Baleta, P., and Larriba-Pey, J. L. (2011). A discussion on the design of graph database benchmarks. pages 25–40. 11, 12

[Dynamodb, 2014] Dynamodb (Acessado em: 06 de Abril de 2014.). Disponível em: http://aws.amazon.com/dynamodb/. 10

[FlockDB, 2014] FlockDB (Acessado em: 06 de Abril de 2014.). Disponível em: https://github.com/twitter/flockdb. 10

[Fowler and Sadalage, 2013] Fowler, M. and Sadalage, P. J. (2013). *NoSQL Um Guia Conciso para o Mundo Emergente de Persistência Poliglota Essencial*. NOVATEC, São Paulo. 10

[Gonçalves et al., 2013] Gonçalves, J., de Oliveira, D., Ocaña, K., Ogasawara, E., Dias, J., and Mattoso, M. (2013). Performance analysis of data filtering in scientific workflows. *Journal of Information and Data Management*, 4(1):17. 16

[HamsterDB, 2014] HamsterDB (Acessado em: 06 de Abril de 2014.). Disponível em: http://hamsterdb.com/. 10

[Hartig and Zhao, 2010] Hartig, O. and Zhao, J. (2010). Publishing and consuming provenance metadata on the web of linked data. pages 78–90. 5

[HBase, 2014] HBase (Acessado em: 06 de Abril de 2014.). Disponível em: https://hbase.apache.org/. 10

[Imran and Hlavacs, 2013] Imran, M. and Hlavacs, H. (2013). Provenance framework for the cloud infrastructure: Why and how? *International Journal On Advances in Intelligent Systems*, 6(1 and 2):112–123. 2

[InfiniteGraph, 2014] InfiniteGraph (Acessado em: 06 de Abril de 2014.). Disponível em: http://www.objectivity.com/infinitegraph. 10

[Jarrard, 2001] Jarrard, R. D. (2001). Disponível em: http://www.iibhg.ukim.edu.mk/obrazovanie/sm_all.pdf. 1

[Korolev and Joshi, 2014] Korolev, V. and Joshi, A. (2014). Prob: A tool for tracking provenance and reproducibility of big data experiments. *Reproduce'14. HPCA 2014*. 17

[MongoDB, 2014] MongoDB (Acessado em: 06 de Abril de 2014.). Disponível em: https://www.mongodb.org/. 10

[Moreau et al., 2009] Moreau, L., Freire, J., Futrelle, J., Myers, J., and Paulson, P. (2009). Governance of the open provenance model. *URL http://twiki. ipaw. info/pub/OPM/WebHome/governance. pdf*. 5

[Muniswamy-Reddy et al., 2010] Muniswamy-Reddy, K.-K., Macko, P., and Seltzer, M. I. (2010). Provenance for the cloud. In *FAST*, volume 10, pages 15–14. 2, 15

[Muniswamy-Reddy and Seltzer, 2010] Muniswamy-Reddy, K.-K. and Seltzer, M. (2010). Provenance as first class cloud data. *ACM SIGOPS Operating Systems Review*, 43(4):11–16. 15

[Neo4J, 2014] Neo4J (Acessado em: 06 de Abril de 2014.). Disponível em: http://www.neo4j.org/learn/neo4j. 10

[Olson et al., 1999] Olson, M. A., Bostic, K., and Seltzer, M. I. (1999). Berkeley db. pages 183–191. 10

[Padhy et al., 2011] Padhy, R. P., Patra, M. R., and Satapathy, S. C. (2011). Rdbms to nosql: Reviewing some next-generation non-relational database's. *International Journal of Advanced Engineering and Technologies.*, 11(1):015–030. 10

[Partner et al., 2013] Partner, J., Vukotic, A., and Watt, N. (2013). *Neo4j in Action*. O'Reilly Media. 14

[Paulino et al., 2010] Paulino, C., Oliveira, D., Cruz, S., Campos, M. L. M., and Mattoso, M. (2010). Captura de metadados de proveniência para workflows científicos em nuvens computacionais. *Anais do XXV Simpósio Brasileiro de Banco de Dados*. 15

[Pinheiro et al., 2013] Pinheiro, R., Holanda, M., Araujo, F., Walter, E., and Lifschitz, S. (2013). Automatic capture of provenance data in genome project workflows. *Bioinformatics and Biomedicine (BIBM), 2013 IEEE International Conference on*, pages 15 – 20. 16, 23

[Ram and Liu, 2007] Ram, S. and Liu, J. (2007). Understanding the semantics of data provenance to support active conceptual modeling. In *Active conceptual modeling of learning*, pages 17–29. Springer. 5

[RavenDB, 2014] RavenDB (Acessado em: 06 de Abril de 2014.). Disponível em: https://ravendb.net/. 10

[Redis, 2014] Redis (Acessado em: 06 de Abril de 2014.). Disponível em: http://redis.io/. 10

[Riak, 2014] Riak (Acessado em: 06 de Abril de 2014.). Disponível em: http://basho.com/riak/. 10

[Rothberg and Leamon, 2008] Rothberg, J. M. and Leamon, J. H. (2008). The development and impact of 454 sequencing. *Nature biotechnology*, 26(10):1117–1124. 1

[Sahoo and Sheth, 2009] Sahoo, S. S. and Sheth, A. (2009). Provenir ontology: Towards a framework for escience provenance management. 5

[Schuster, 2007] Schuster, S. C. (2007). Next-generation sequencing transforms today's biology. *Nature*, 200(8). 1

[Sousa et al., 2010] Sousa, F. R., Moreira, L. O., de Macêdo, J. A. F., and Machado, J. C. (2010). Gerenciamento de dados em nuvem: Conceitos, sistemas e desafios. *Topicos em sistemas colaborativos, interativos, multimidia, web e bancos de dados, Sociedade Brasileira de Computacao*, pages 101–130. 9, 10

[Tan, 2004] Tan, W. C. (2004). Research problems in data provenance. *IEEE Data Eng. Bull.*, 27(4):45–52. 4

[UNIPROT, 2012] UNIPROT (2012). Disponível em: http://www.uniprot.org. 16

[Vicknair et al., 2010] Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y., and Wilkins, D. (2010). A comparison of a graph database and a relational database: a data provenance perspective. page 42. 13, 14

[Voldemort, 2014] Voldemort, P. (Acessado em: 06 de Abril de 2014.). Disponível em: http://www.project-voldemort.com/voldemort/. 10

[W3C, 2014] W3C (Acessado em 06 de Abril de 2014.). Disponível em: www.w3.org/TR/prov-dm/. iv, 6, 7, 8

[Woodman et al., 2011] Woodman, S., Hiden, H., Watson, P., and Missier, P. (2011). Achieving reproducibility by combining provenance with service and workflow versioning. pages 127–136. 16