Proposta de integração de dados heterogêneos em um ambiente com arquitetura *Polystore*

Ludmila Ribeiro Bôscaro Yung

JUIZ DE FORA

DEZEMBRO, 2019

Proposta de integração de dados heterogêneos em um ambiente com arquitetura *Polystore*

Ludmila Ribeiro Bôscaro Yung

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Victor Ströele de Andrade Menezes

JUIZ DE FORA

DEZEMBRO, 2019

Proposta de integração de dados heterogêneos em um ambiente com arquitetura *Polystore*

Ludmila Ribeiro Bôscaro Yung

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Victor Ströele de Andrade Menezes
Doutor em Engenharia de Sistemas e Computação

Mário Antônio Ribeiro Dantas
PhD. em Ciência da Computação

Fernanda Cláudia Alves Campos
Doutora em Engenharia de Sistemas e Computação

JUIZ DE FORA
06 DE DEZEMBRO, 2019

Aos meus pais, irmãos, ao Lucas, amigos e familiares pelo apoio.
Ao meu orientador Victor, pela paciência, dedicação e tempo investidos e ao professor Mário pelas dicas.

Resumo

O crescente volume de dados (*Big Data*) gera a necessidade de oferecer suporte à iniciativas avançadas de análise, tais como análise de dados heterogêneos, análise preditiva, mineração de dados e aplicativos de aprendizado de máquina. Mecanismos como *NoSQL*, *NewSQL* e *SQL-on-Hadoop* foram desenvolvidos para otimizar a execução de diferentes tarefas de processamento de dados.

Vislumbra-se um ambiente organizacional em que há grande volume de dados para serem analisados e, ao mesmo tempo, procurar extrair informações úteis a partir desses dados. Sendo assim, há a preocupação de entender como esses dados estão organizados em que, muitos deles, são originários das mais diversas fontes. Isto proporciona um desafio, pois além do grande volume de dados, eles são heterogêneos e precisam de análise por parte dessas organizações, que sofrem com a dificuldade de realizar a integração destes.

Neste contexto, este trabalho utiliza de uma abordagem *Polystore* com o intuito de possibilitar a integração de dados heterogêneos para um determinado usuário de uma dada base de dados, promovendo consultas pré-processadas e possibilitando que ele realize o processo de integração sem necessariamente conhecer a fundo conceitos e aplicações de Banco de Dados. Os resultados obtidos mostram que é possível desenvolver soluções que possibilitem esta integração sem necessidade de conhecimentos específicos por parte dos usuários finais, assim como promove a integração de dados utilizando duas diferentes abordagens, uma relacional e a outra orientada a grafos.

Palavras-chave: Banco de Dados, Integração de Dados, Dados Heterogêneos, *Polystore*.

Abstract

The growing volume of Big Data generates the need to support advanced analytics initiatives such as heterogeneous data analytics, predictive analytics, data mining, and machine learning applications. Mechanisms such as NoSQL, NewSQL, and SQL-on-Hadoop were developed to optimize the execution of different data processing tasks.

An organizational environment is envisioned in which there is a large volume of data to analyze and, at the same time, seek to extract useful information from this data. Thus, there is a concern to understand how this data is organized in which, many of them, originate from the most different sources. This presents a challenge because, besides the large volume of data, they are heterogeneous and need analysis by these organizations, which suffer from the difficulty of integrating them.

In this context, this work uses a Polystore approach in order to enable the integration of heterogeneous data for a given user of a given database, promoting preprocessed queries and enabling to perform the integration process without necessarily knowledge in Database concepts and applications. The results show that it is possible to develop solutions that enable this integration without the need for specific knowledge from end users, as well as promoting data integration using two different approaches, one relational and the other is graph driven.

Keywords: Data Base, Data Integration, Heterogeneous Data, Polystore.

Agradecimentos

Em primeiro lugar, sem dúvida alguma, eu gostaria de agradecer ao meu Bom Deus por ter se feito presente nas situações, pessoas, ações, onde menos esperei. Por ter me dado toda a capacidade de pensamentos e ideias as quais pensei que não existiam mais ou não sabia que existiam; pude perceber que eu era capaz de continuar, mesmo sabendo de minhas limitações.

Em segundo lugar a mim mesma, pois caminhar ao longo desse tempo, mesmo diante de tantos obstáculos, sejam eles de saúde, de bloqueios mentais, dificuldades que foram surgindo ao longo da jornada, é um grande feito; ter mantido a fé, a esperança, o empenho, a alegria mesmo diante do cansaço físico, mental e espiritual.

Agradeço aos meus pais, por tudo o que fizeram para a construção e contribuição de minha educação moral, escolar e acadêmica, desde o meu nascimento. Por todos os momentos em que tive que me ausentar e não pude ajudá-los ou até mesmo não comparecer a eventos em família, pela compreensão sem limites, pelo apoio incondicional. Mesmo sabendo que era difícil para mim, eles se fizeram presentes o tempo todo. Pelos "puxões de orelha", pela ajuda (mesmo sem conhecerem nada sobre o curso de Ciência da Computação), pelo amor que se manteve como sempre.

Aos meus irmãos queridos, por todos os momentos de gargalhadas sem fim, brincadeiras, pelo carinho, pelas desavenças que me fizeram crescer como ser humano, bem como pelo apoio, amizade e pelos abraços. Aos meus avós: Dadi (*in memorian*), Nadir e José pelas inúmeras orações e carinho; aos demais familiares pela força e apoio.

Ao Lucas que ficou ao meu lado, pelo companheirismo, amor e carinho, por ter me ensinado muito, compreendido, respeitado, me ajudado a caminhar, pelos momentos gastronômicos, de risadas e descontração.

Aos meus amigos e colegas de caminhada acadêmica (fica impraticável poder citar todos vocês) que pude ter o prazer de conhecer ao longo do curso; às "meninas"da

computação queridas e amigas Ana Paula, Mayra (não é da computação, mas mora no coração também), Míria, Rebeca e Gisele, por terem compreendido as vezes em que não pude comparecer aos nossos "Encontrinhos", pelas risadas, comidas saborosas e amizade. A todos vocês meu muito obrigada, por terem sido um apoio moral, pelos nossos estudos em dupla ou em grupo, pelos trabalhos realizados juntos, pelas aulas que ficaram mais divertidas.

Aos amigos externos à UFJF, os quais foram tão importantes como os acima citados, cada um a sua maneira. Em especial aos meus amigos da Trupe de Frassati e de caminhada do Ministério de Música e ao pessoal da comunidade a qual faço parte, pelo carinho constante.

Aos professores, por todo o conhecimento educacional e pessoal transmitido a mim ao longo dos anos, pela paciência, por terem sido canais de inspiração, por nutrirem o ardor de querer continuar. Ao meu orientador, Victor, por todo o empenho em me auxiliar para que este trabalho pudesse ser concluído; pelas dicas, sugestões e conhecimento compartilhados, por acreditar que eu era capaz de chegar até aqui. Ao professor Mário, por ter estado presente desde o momento em que começou a conhecer mais sobre o trabalho, juntamente com o professor Victor, por todas as ideias e compartilhamento de ótimos trabalhos e conhecimento, pela alegria e descontração e por acreditar em minha capacidade. Ao professor Jairo, que me acolheu na época em que era coordenador do noturno, por me explicar o funcionamento das disciplinas e do curso como um todo e ao Rodrigo por ter me orientado nas dúvidas que tive. À Alessandreia pela alegria e simpatia e por ter sido minha primeira orientadora de bolsa, Marcelo Moreno por ter sido um grande motivador e orientador, Stênio e Luciana pela compreensão, carinho e orientação. À Fernanda por todas as sugestões de melhoria, críticas construtivas e pelo apoio para que esta monografia fosse concluída com êxito.

Aos professores e funcionários do *DCC* como um todo, pela atenção, paciência, tiradas de dúvidas, pelo carinho e aos demais departamentos da *UFJF* nos quais cursei disciplinas, por oferecerem atividades complementares, pelas aulas e bons momentos.

"Não venci todas as vezes que lutei, mas perdi todas as vezes que deixei de lutar!"

Cecília Meireles

Conteúdo

Lista de Figuras

Lista de Abreviações

DCC	Departamento de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
SQL	do inglês: Structured Query Language (Linguagem de Consulta Estruturada)
NoSQL	do inglês: Not only SQL (Não somente SQL)
API	do inglês: Application Programming Interface (Interface de Programação de Aplicativos)
BD	Banco de Dados
BDD	Banco de Dados Distribuídos
BDR	Banco de Dados Relacional
OLTP	do inglês: Online Transaction Processing (Processamento de Transações *Online*)
GDBMS	do inglês: Graph Database Management System (Sistema de Gerenciamento de Banco de Dados Gráficos)
SBDD	Sistema de Banco de Dados Distribuído
SGBD	Sistema de Gerenciamento de Banco de Dados
RDD	do inglês: Resilient Distributed Dataset (Conjunto de Dados Distribuídos e Resilientes)
HDFS	do inglês: Hadoop Distributed File System (Sistema de Arquivos Distribuídos do Hadoop)

1 Introdução

Este capítulo apresenta informações gerais da proposta abordada neste trabalho, bem como traz consigo a contextualização a qual o tema se enquadra e suas justificativas, motivação, objetivos e a organização do trabalho.

1.1 Apresentação do Tema e Contextualização

Quando o assunto em questão é fluxo constante de dados, automaticamente pensa-se em cenários diversos que precisam lidar com tarefas de analisar e processar informações que são recebidas a todo momento. Não é difícil de imaginar nos dias atuais o quão intenso é o vincular de informações juntamente com dados heterogêneos que circulam diariamente na *web* que chegam a casa do milhões (WU; ZHANG; WANG, 2019) facilmente.

Tendo em vista esses desafios de lidar com informações de diversos tipos, fica evidente que utilizar uma abordagem de integração de dados venha a facilitar o manuseio dessas informações e dados (MASSER, 2019), de modo que isso possiblite que o usuário do sistema faça buscas integradas, detecte mudanças de comportamento baseado em análises históricas e de tomada de decisão em tempo real.

Ao longo dos anos foram desenvolvidos diversos tipos de armazenamento de dados, em especial aqueles que trabalham com grandes volumes (*Data Warehouses*). Em particular, surgiu o armazenamento de documento nos quais um registro de banco de dados consiste em uma coleção de pares de valores-chave. Exemplos dessa classe de sistema incluem o *CouchDB* e o *MongoDB* (STONEBRAKER, 2010).

Sendo assim, é importante que seja possível reunir um conjunto de dados de tipos distintos, providos de locais diferentes, de modo que a pesquisa a esses dados, em uma aplicação, seja efetuada provendo a integração dos mesmos. A implementação de sistemas que utilizam *SQL*, *NoSQL* e *NewSQL* (STROHBACH et al., 2016) tem crescido cada vez mais, focando em facilitar a integração dos dados para que não seja obrigatório que o usuário tenha conhecimento em todas as linguagens de consulta em banco de dados.

O trabalho desenvolvido por (RODRIGUES, 2018) faz com que este processo, através de uma *API*, possibilite, através da utilização do *framework Spark*, a realização de pesquisas em dois tipos distintos de Banco de Dados com tipos distintos de dados. No trabalho de (RODRIGUES, 2018), os dados estão dispostos em forma de tabela e grafo, sendo verificada a possibilidade da realização da integração.

1.2 Justificativa e Motivação

Por se tratar de um assunto cada vez mais discutido, tanto em meio acadêmico como em meio empresarial, a área de Ciência de Dados tem a preocupação em ser interdisciplinar, ou seja, abranger áreas de conhecimento matemático, estatístico, de negócio, mídias digitais, bem como ser uma fusão entre disciplinas clássicas, tais como mineração de dados, banco de dados e sistemas distribuídos (AALST, 2016).

Além disso, Ciência de Dados está ligada ao termo cada vez mais utilizado: o *Big Data* (DUGGAN et al., 2015). Devido ao fato de grandes volumes de dados gerados com a utilização de aplicações (sejam elas *mobile* ou via *web*), dispositivos tecnológicos vestíveis (*wearables*), sensores, etc. assim como a análise desses dados que torna-se uma questão-chave, tão importante como o trabalho com o volume e armazenamento dos mesmos. Surgiu assim a necessidade de se trabalhar melhor as arquiteturas de *BD* escolhidas devido a forma estruturada ou não desses dados, em vista de proporcionar uma análise mais apurada.

A implementação de sistemas que utilizam *NoSQL* e *NewSQL* têm crescido cada vez mais, assim como de sistemas que usufruem de ambas as arquiteturas para diferentes tipos de contexto. Essas arquiteturas proporcionam uma escalabilidade horizontal (POKORNý, 2011), permitindo o particionamento de dados e possibilitando o armazenamento e processamento em locais diferentes, ou seja, distribuídos.

Em síntese, é possível afirmar que há sistemas que utilizam diversos tipos de dados os quais são armazenados em tipos diferentes de Bancos de Dados e que precisam ser integrados para que a consulta apresente os resultados esperados.

1.3 Objetivo

Sendo assim, visto que existe uma motivação de se utilizar abordagens que promovem o melhoramento do uso e aplicação de *BD*, este trabalho propõe-se a realizar a integração de dados heterogêneos utilizando uma abordagem *Polystore* para Bancos de Dados Heterogêneos e Distribuídos, a fim de que se tenha retorno de consultas relacionadas a esses bancos, mesmo para usuários que não sejam especialistas em *BD*, retornando a nível de interface *web* os resultados obtidos dessas consultas.

1.4 Problema

O processo de pesquisa procura solucionar problemas que sejam de interesse de uma instituição (e/ou de um pesquisador) que tem como associados a ela pesquisadores que, através de estudos, desenvolvimento, testes, comparações, apresentação de resultados, promovem o conhecimento a respeito de determinado tema.

Considerando o assunto e tópicos iniciais apresentados nas seções anteriores, o problema abordado neste trabalho é: *Como auxiliar usuários não especialistas em Banco da Dados na obtenção e integração de dados oriundos de diversas fontes de dados heterogêneos e geograficamente distribuídos?*

1.5 Organização do Trabalho

Este trabalho está organizado da seguinte maneira:

- O capítulo 2 trata sobre a Fundamentação Teórica, divido em seções explicando sobre os conceitos principais e necessários que proporcionam um melhor entendimento sobre o contexto geral o qual este trabalho se enquadra;
- O capítulo três mostra um levantamento de trabalhos relacionados a esta monografia;
- O capítulo quatro apresenta uma introdução acerca da proposta deste trabalho, assim como a explicação do preparo do ambiente computacional e da abordagem;

- O capítulo cinco apresenta o ambiente computacional e os resultados obtidos, assim como a metodologia que fora utilizada ao longo do desenvolvimento e pesquisa;

- O capítulo seis apresenta as conclusões e considerações finais oriundas dos resultados obtidos, sobre a experiência e conhecimento adquiridos ao longo do desenvolvimento desta monografia, fechando com sugestões de melhorias, sofisticação e sugestões para trabalhos futuros e

- Por fim, todas as referências bibliográficas utilizadas durante a produção textual e prática.

2 Fundamentação Teórica

No quesito compreensão de fundamentos teóricos básicos, sejam para profissionais da área de *BD* e Computação, como para entusiastas ou até mesmo para leigos no assunto, torna-se indispensável o entendimento dos tópicos abordados neste trabalho. Além disso, são necessários para melhor compreensão e aplicação no que dizem respeito à implementação de bancos relacionais, assim como outros tipos de *BD*.

2.1 Banco de Dados

Todo sistema que armazena dados, seja ele mais arcaico como, por exemplo: um catálogo, lista de números de telefone, planilha de papel, etc, não deixa de ser um *BD*. Nesses exemplos citados temos tipos de dados que podem ter o mesmo tipo (numéricos) ou até mesmo diferirem entre si (texto, números, data, entre outros). Por exemplo, na lista telefônica geralmente tem-se dados de tipo *nome* (da pessoa, empresa, instituição, etc), *número de telefone* (comercial, residencial, celular), *endereço*, *observações* com relação a este contato.

Portanto, podemos entender como *BD* qualquer sistema que reúna uma série de dados relacionados a um determinado assunto, podendo prover o processo de tomada de decisão através de análise, informação e conhecimento. O objetivo principal de um *BD* é fornecer um ambiente que seja adequado e eficiente para o uso na recuperação e no armazenamento da informação[1].

2.1.1 Banco de Dados Relacional

Um banco de dados relacional é um conjunto de tabelas a partir das quais os dados podem ser acessados ou remontados de várias maneiras diferentes sem a necessidade de reorganizar as tabelas do *BD*. Em um *BDR* a Linguagem de Consulta Estruturada (*SQL*) possui instruções que são usadas para manipular os dados desse banco.

[1] ⟨https://www.geeksforgeeks.org/introduction-of-dbms-database-management-system-set-1/⟩

Uma pequena curiosidade acerca do *BDR* é que ele fora desenvolvimento e criado em 1970 por (CODD, 1970). Em seu artigo, Codd propôs a mudança do armazenamento de dados em estruturas hierárquicas ou de navegação para a organização de dados em tabelas contendo linhas e colunas.

Cada tabela, ou relação, em um *BDR* contém uma ou mais categorias de dados em colunas ou atributos. Cada linha, também chamada registro ou tupla, contém uma instância exclusiva de dados ou chave para as categorias definidas pelas colunas. Cada tabela possui uma chave primária exclusiva (*primary key*), que identifica as informações em uma tabela. O relacionamento entre tabelas pode ser definido através do uso de chaves estrangeiras (*foreign key*), um campo em uma tabela que se vincula à chave primária de outra tabela.

Por exemplo, um *BD* típico de entrada de pedidos comerciais incluiria uma tabela que descrevesse um cliente com colunas para *nome*, *endereço*, *número de telefone* e assim por diante. Outra tabela descreveria um pedido: *produto*, *cliente*, *data*, *preço de venda* e assim por diante. Um usuário de um *BDR* pode obter uma visualização do *BD* para atender às suas necessidades. Por exemplo, um relatório de compra de todos os clientes durante um determinado período.

Ao criar um *BDR*, pode-se definir o domínio de valores possíveis em uma coluna de dados e outras restrições que podem ser aplicadas a esse valor. Sendo assim, por exemplo, é possível que se queira restringir um domínio, que permite até 10 nomes, para um de menor tamanho, de apenas três, citando caso análogo.

Duas restrições estão relacionadas à integridade dos dados e às chaves primárias e estrangeiras:

- A integridade da entidade garante que a chave primária em uma tabela seja exclusiva e que o valor não esteja definido como nulo.

- A integridade referencial requer que cada valor em uma coluna de chave estrangeira seja encontrado na chave primária da tabela da qual se originou.

2.2 *NoSQL*

Um banco de dados *NoSQL* é uma alternativa aos bancos de dados relacionais que é especialmente útil para trabalhar com grandes conjuntos de dados distribuídos (KUMAWAT; PAVATE, 2019). Esses bancos de dados podem suportar uma variedade de modelos de dados, incluindo formatos de valores-chave, documentos, colunas e gráficos.

NoSQL foi criado para descrever a variedade de mecanismos e técnicas revolucionárias de armazenamento de dados que estão sendo desenvolvidos de forma agressiva para lidar com conjuntos de dados massivos e dinâmicos [2]. O aumento da tendência para técnicas não tradicionais de armazenamento de dados tem visto uma amplitude e profundidade de variedade de novas tecnologias, ideias e, o mais importante, inovação que resolve problemas específicos como conjuntos de dados em escala.

Cada uma delas realiza *dados em escala* por meio de diferentes técnicas:

- Teorema *CAP*: é um conceito onde um sistema de Banco de Dados Distribuído pode ter somente duas características das três: Consistência, Disponibilidade e Tolerância a Partição;
- Orientado a coleções e
- Orientado a colunas e valor-chave.

as quais otimizam diferentes processos de fluxo de trabalho.

Essas soluções dominam o componente de armazenamento de dados dos aplicativos há muitos anos e em muitos ambientes e funcionam muito bem em determinados fluxos de trabalho. Isso foi ainda mais exacerbado com a consolidação de várias implementações de *SQL* por fornecedores que compram um ao outro em uma batalha constante para ser a "única"solução, bem como uma unidade de marketing desenfreada e geralmente desmarcada para garantir o domínio contínuo da *SQL* no mercado.

[2] ⟨https://www.voodootikigod.com/nosql-a-modest-proposal/⟩

2.3 *NewSQL*

Os bancos de dados *NewSQL* são bancos de dados *SQL* modernos que resolvem alguns dos principais problemas associados ao tradicional processamento de transações *online* (*OLTP*) (PAVLO; ASLETT, 2016), este processamento é encarregado de registrar todas as transações contidas em uma determinada operação organizacional. Esses bancos procuram obter a escalabilidade e o desempenho aprimorado dos bancos de dados *NoSQL*, mantendo os benefícios dos tradicionais *SGBD*s.

Em outras palavras, os bancos de dados *NewSQL* são sistemas de *BDR*s que combinam o *OLTP*, o alto desempenho e a escalabilidade do *NoSQL*. Eles mantêm as garantias *ACID* (Atomicidade, Consistência, Isolamento e Durabilidade) do SGBD tradicional.

As transações do *ACID* garantem processos de negócios completos, transações simultâneas, em caso de falhas ou erros no sistema os dados conseguem sobreviver e consistência antes e depois de uma transação.

As principais categorias de sistemas *NewSQL* incluem novas arquiteturas, *middleware*(fornecedor de serviços para *softwares* aplicativos) de *sharding*(partição horizontal de dados em um *BD* ou mecanismo de busca) transparente, mecanismos *SQL* e *DBaaS* (*Database-as-a-Service*).

2.4 Banco de Dados Distribuído

De acordo com (SILBERSCHATZ et al., 1997), um Banco de Dados Distribuído é basicamente um *BD* que não se limita a um sistema, ou seja, está espalhado por locais diferentes, sendo assim, em vários computadores ou em uma rede de computadores. Um Sistema de Banco de Dados Distribuído está localizado em vários locais que não compartilham componentes físicos.

Alguns tipos de *BDD*[3] importantes:

- Banco de Dados Homogêneo: todos os diferentes locais armazenam o *BD* de forma idêntica, são mais fáceis de serem gerenciados;

[3]⟨https://www.geeksforgeeks.org/distributed-database-system/⟩

- Banco de Dados Heterogêneo: como o próprio nome diz, locais diferentes podem usar *schemas*(esquemas) e *softwares* diferentes na parte de armazenamento. Isso pode levar a problemas no processamento e transações das *queries*. Daí, portanto, um desafio a fazer com que um local não desconheça outros locais devido a diferenças de *SGBD*s, Sistemas Operacionais, etc.

Existe a possibilidade de se utilizar uma abordagem híbrida com relação a armazenamento. Pode, também, ser escolhida apenas uma, dentre duas abordagens. A primeira se trata de replicação, como se tratam de locais diferentes, essa abordagem permite que toda relação seja armazenada de forma redundante em dois ou mais locais. Ou seja, nessa abordagem, os sistemas mantém cópias dos dados.

As solicitações de consulta podem ser processadas em paralelo, uma vez que ao optar por replicação, aumenta a disponibilidade dos dados, o que é uma boa vantagem de se utilizar essa abordagem. Acontece que, se for um sistema que requer atualização constante, torna-se uma desvantagem pois todos os dados precisam ser atualizados constantemente. Qualquer alteração feita em algum local precisa ser passada aos demais, ou pode levar a inconsistência do BDD (SILBERSCHATZ et al., 1997).

A segunda se chama fragmentação, onde as relações são divididas em partes menores. Esses fragmentos são armazenados em locais diferentes. Nesta abordagem há necessidade de armazenar cada uma dessas partes e garantir que os fragmentos possam ser usados para reconstruir a relação original, sem que haja perda de dados. Ela é vantajosa pois, diferentemente da replicação, não cria cópias de dados, sendo assim a consistência não se torna um problema.

2.5 Banco de Dados orientado da Grafos

Popularmente chamado de Sistema de Gerenciamento de Banco de Dados Gráficos (*GDBMS*), os *GDBMS*s geralmente são considerados parte do cenário *NoSQL*, que inclui soluções não relacionais para o gerenciamento de dados caracterizados por modelos de dados complementares, bancos de dados sem esquema, operações básicas de acesso a dados e, eventualmente, transações consistentes (VIRGILIO; MACCIONI; TORLONE,

2014).

Os *GDBMS*s são considerados, no entanto, um mundo à parte dos outros sistemas *NoSQL* (por exemplo, armazenamento de valores-chave, documentos e colunas), pois seus recursos são bastante exclusivos no modelo de dados que adotam e nas primitivas de acesso a dados que eles oferecem.

Um *GDBMS* armazena dados por meio de um multigrafo, geralmente chamado de gráfico de propriedade, onde nós e arestas são rotulados com dados na forma de pares de valores-chave.

2.6 *Big Data*

Para compreendermos o termo *Big Data* é necessário entender que o mesmo tem origem incerta. Atribui-se que provavelmente surgiu em meados dos anos 90, mas somente a partir de 2011 que o termo começou a ser mais generalizado. Esse crescimento de popularidade pode ser atribuído devido a incentivos promocionais de grandes empresas que são líderes em tecnologia, as quais investiram na construção do mercado de análise de dados, de acordo com (GANDOMI; HAIDER, 2015).

Traduzindo de maneira literal, claramente, o tamanho é a primeira característica que vem à mente. No entanto, outras características de *Big Data* surgiram ao longo do tempo. *Os três V's* foram sugeridos por (LANEY, 2001), onde ele cita um exemplo de *E-commerce* (comércio eletrônico) e o crescimento de sua popularidade eminente. Sendo assim, os *V's* que Laney sugeriu seriam as três dimensões no gerenciamento de dados, são eles: **Volume**, **Variedade** e **Velocidade**.

Por exemplo, o Gartner, Inc.[4] define *Big Data* em termos semelhantes:
"*Big Data* são ativos de informações de alto volume, alta velocidade e grande variedade que exigem formas inovadoras e econômicas de processamento de informações para uma visão e tomada de decisão aprimoradas."

O primeiro *V*, Volume, refere-se à magnitude dos dados. As definições de volumes de *Big Data* são relativas e variam de acordo com fatores, como tempo e tipo de dados. O que pode ser considerado *Big Data* hoje pode não atingir o limite no futuro, porque

[4]⟨https://www.gartner.com/en/information-technology/glossary/big-data⟩

as capacidades de armazenamento aumentarão, permitindo que conjuntos de dados ainda maiores sejam capturados (GANDOMI; HAIDER, 2015).

Além disso, o tipo de dados, discutido sob o segundo *V*, Variedade, define o que se entende por "grande". Dois conjuntos de dados do mesmo tamanho podem exigir diferentes tecnologias de gerenciamento de dados com base em seu tipo, por exemplo, dados tabulados vs. dados de vídeo. Assim, as definições de *Big Data* também dependem da indústria. Portanto, essas considerações tornam impraticável definir um limite específico para grandes volumes de dados. Variedade refere-se à heterogeneidade estrutural em um conjunto de dados. Os avanços tecnológicos permitem que as empresas usem vários tipos de dados estruturados, semiestruturados e não estruturados.

Por último, o terceiro *V*, Velocidade refere-se à taxa na qual os dados são gerados e a rapidez na qual eles devem ser analisados e acionados. A proliferação de dispositivos digitais, como *smartphones* e sensores, levou a uma taxa sem precedentes de criação de dados e está gerando uma necessidade crescente de análises em tempo real e planejamento baseado em evidências.

É importante lembrar que esses três *V*s foram sugeridos por (LANEY, 2001), porém, depois vieram muitos estudos e mostrou-se que esses três poderiam ser expandidos para sete *V*s. Os que foram acrescentados foram *Veracidade*, Visualização e *Valor*. O primeiro está intimamente ligado a *Velocidade* devido a veracidade dos dados ser mais apurada se analisada e coletada em tempo real. O segundo refere-se à forma como os dados são apresentados à gerência para sua tomada de decisão[5]. Já o terceiro, *Valor*, tem como premissa que "o importante é saber fazer as perguntas certas no início de todo processo de análise"[6].

2.7 *Polystore*

Um sistema *Polystore* é qualquer *SGBD* que é construído sobre vários mecanismos de armazenamento integrados e heterogêneos (ELMORE et al., 2015). Sistemas *PolyStore* são aqueles construídos sobre múltiplos SGBDs heterogêneos e integrados. Além disso, um

[5] ⟨http://blogsofdatawarehousing.blogspot.com/2017/01/7-vs-of-big-data.html⟩
[6] ⟨https://www.gta.ufrj.br/grad/15_1/bigdata/vs.html⟩

sistema *PolyStore* se distingue dos SGBDs federados tradicionais uma vez que necessita apenas de um mapeamento das diversas fontes de dados (*i.e.,* formatos dos dados), que são acessados em tempo de execução. Um *SGBD* federado é um *middleware* que é executado sobre *SGBD*s locais e apresenta uma interface para sistemas distintos com esquemas de *SGBD*s construídos independentemente. Os sistemas nessa categoria incluem *R**, *Ingres**, *Garlic*, o *IBM Information Integrator* e vários outros. Devem ser comparadas com os *SGBD*s paralelos, que são únicos com tabelas particionadas e/ou replicados em um único esquema.

Sendo assim, um *Polystore* deve possuir várias fontes de dados, porém, não devem ser confundidos com um *SGBD* distribuído pois este, por sua vez, consiste em instâncias replicadas de um mecanismo de armazenamento atrás de um único mecanismo de consulta. O ponto chave para entender uma arquitetura do tipo *Polystore* é que os vários mecanismos de armazenamento são distintos e acessados separadamente por meio de seu próprio mecanismo de consulta (*query*) (GADEPALLY et al., 2016).

Por fim, os mecanismos de armazenamento devem ser integrados. Em um *SGBD* federado, os mecanismos de armazenamento individuais são independentes. Em um sistema *Polystore*, os mecanismos de armazenamento são gerenciados juntos como um conjunto integrado. Isso é fundamental, pois significa que, em um sistema de *Polystore*, você pode modificar os mecanismos ou o *middleware*, gerenciando-os de modo que "o todo seja maior que a soma de suas partes"[4].

[4]⟨https://bigdawg-documentation.readthedocs.io/en/latest/intro.html\#polystore-systems⟩

3 Trabalhos Relacionados

Ao realizar determinado estudo científico, é necessário ter em mente que é fundamental que o processo de Revisão da Litetarura seja feito, uma vez que é uma atividade de busca, análise e descrição de um conhecimento que busca responder uma pergunta específica. Existem alguns tipos de revisão da literatura como a narrativa, a sistemática e a integrativa[7].

Neste capítulo são apresentados trabalhos relacionados a "Proposta de integração de dados heterogêneos num ambiente com arquitetura *Polystore*" os quais proporcionaram melhor entendimento no âmbito de pesquisa e área de atuação do mesmo.

3.1 *Big data integration of heterogeneous data sources: the re-search alps case study*

De acordo com (GUERRA et al., 2019), é notável perceber que a integração de fontes de dados é uma das questões mais desafiadoras e duradouras que envolvem a comunidade de pesquisa desde os últimos 30 anos. Existem muitos resultados científicos alcançados mas ainda assim não pode ser considerado um problema resolvido.

Contudo, poucas ferramentas de integração de dados saíram de aplicativos desenvolvidos para esta finalidade, tanto para aplicativoes industriais, como para a utilização em cenários de negócios. Além disso, nenhuma tecnologia líder para integração de dados surgiu e foi amplamente adotada pelos profissionais.

As principais etapas adotadas em um *pipeline* de *Big Data*, de acordo com (GUERRA et al., 2019), são:

- *Schema Alignment*, é um processo que transforma os dados tabulares em um projeto no formato de dados usado por uma base de conhecimento. Um esquema é uma estrutura de dados que mapeia cada linha para um conjunto de instruções ou outros

[7] ⟨https://www.fca.unesp.br/Home/Biblioteca/tipos-de-evisao-de-literatura.pdf⟩

registros, dependendo da base de destino;

- *Entity Matching*, é a tarefa de identificar entidades (objetos, instâncias de dados) referentes à mesma entidade do mundo real;

- *Data Fusion*, é o processo de integração de várias fontes de dados para produzir informações mais consistentes, precisas e úteis do que as fornecidas por qualquer fonte de dados individual.

O trabalho proposto por (GUERRA et al., 2019) descreve como o *pipeline* de integração de dados foi implementado no projeto *Re-search Alps*, promovendo a construção de um conjunto de dados *Re-serach Alps*. É mostrado neste trabalho a arquitetura funcional do aplicativo, assim como, componentes e alguns resultados que foram alcançados no cenário fornecido pelo projeto.

O *M-STEP* é um algoritmo de *Entity Matching*(*EM*) que promove a adoção de uma estratégia de comparação baseada em dados contextuais para identificar entidades semelhantes; também promove a exploração de conhecimento externo sobre as fontes e realiza a introdução de uma estratégia de *EM*.

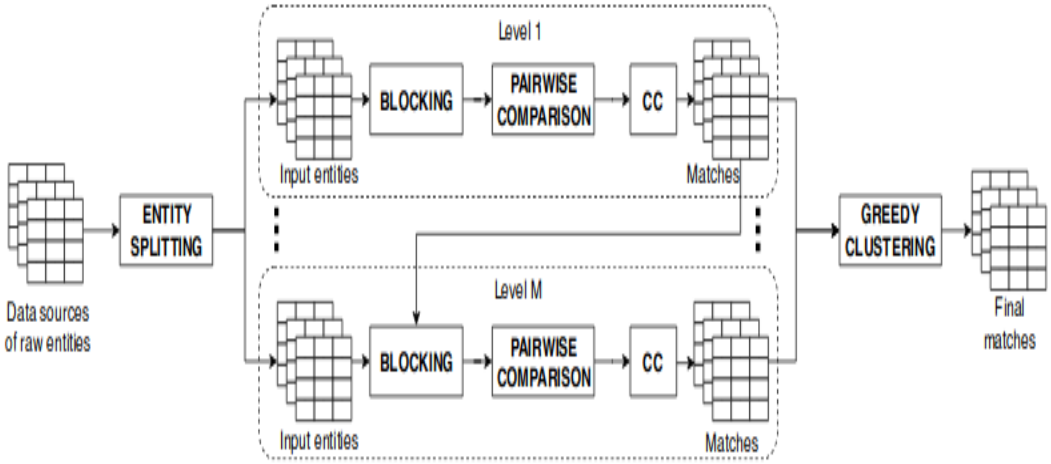

Figura 3.1: Visão geral da abordagem *M-STEP* por (GUERRA et al., 2019)

3.2 *The BigDAWG Polystore System*

Os sistemas *BigDAWG* (DUGGAN et al., 2015) representam o grupo de trabalho *Big Data*: a integração desses avanços entre vários sistemas de processamento de dados.

O sistema *BigDAWG Polystore* foi validado através do desenvolvimento de *benchmarks*(forma de avaliar a performance de um objeto, submetendo-o a testes) e bancos de testes e do trabalho com pesquisadores da comunidade médica e oceanográfica.

O *BigDAWG* é um projeto de grande escala destinado a alterar a maneira de interação com conjuntos de dados muito grandes. Ele foi projetado para suprir análise e ingestão de alto desempenho e muitas das vantagens relativas de diferentes sistemas de *BD*. No *BigDAWG*, é garantido o *design* que suporta as noções muitas vezes concorrentes de transparência de localização e integridade semântica. Isso permite realmente desenvolver uma arquitetura escalável e flexível.

Foi desenvolvido um número de operadores fundamentais: uma interface comum, vários tradutores analíticos comuns através de *shims*(biblioteca que intercepta de forma transparente as chamadas da *API*), tradutor de dados comum por meio de operações de conversão e foi capaz de suportar vários bancos de dados em idiomas nativos através do conceito de ilhas.

A figura abaixo representa a arquitetura utilizada no *BigDAWG*:

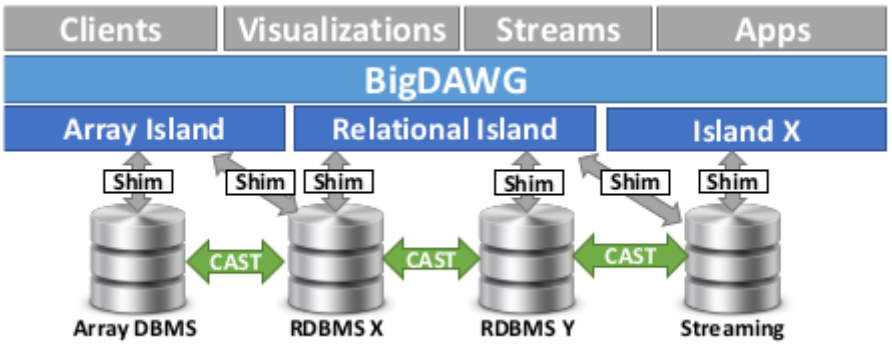

Figura 3.2: Arquitetura do *BigDAWG Polystore* (DUGGAN et al., 2015)

Supondo que um usuário que está desenvolvendo um aplicativo de visualização emite uma consulta na interface comum do *BigDAWG*, isso vai para o otimizador que trabalha com o monitor para determinar o plano de consulta ideal para uma consulta específica. É enviado ao executor responsável por realizar fisicamente essa consulta nos

diferentes mecanismos de *BD* presentes. Um plano de consulta enviado ao executor pode depender de fontes de dados diferentes e pode precisar conversar com o migrador responsável por transferir dados entre diferentes sistemas, conforme necessário. O executor, após concluir a consulta, retorna uma saída para o cliente ou aplicativo de visualização.

Outro componente importante do sistema *Polystore* é o conceito de ilhas e as trocas entre funcionalidade ou completude semântica e transparência da localização.

O lançamento da versão 0.1 do *BigDAWG* consiste em suportar um número de ilhas e mecanismos de *BD*. Especificamente, ele suporta uma ilha relacional, uma ilha de *array* e uma ilha de texto. O *BigDAWG* inclui banco de dados no suporte a mecanismos de banco de dados como *PostgreSQL*, *Accumulo* e *SciDB*.

Esse trabalho se diferencia dos estudos apresentados no que diz respeito a interface com o usuário final. Em todos os trabalhos identificados, é necessário que o usuário conheça a estrutura dos bancos de dados para executar as consultas. Neste *TCC* buscamos apresentar os elementos (estruturas dos bancos de dados) para que o usuário os combine da maneira que desejar.

4 Proposta

Depois de promover um estudo fundamentado em conceitos importantes e relevantes que contribuem para o entendimento teórico e prático de um assunto, vem depois, dependendo da abordagem que está em uso, uma visão e aplicação mais prática. Têm-se uma linha de pesquisa mais teórica quando trabalhos pautados em pesquisas e comparações que são um aprofundamento nos estudos nas obras da literatura em um determinado contexto, porém, há aqueles que são fruto de uma pesquisa que gera um trabalho experimental, como é o caso desta monografia.

No trabalho desenvolvido anteriormente por (RODRIGUES, 2018), há a preocupação de tornar válida a integração de dados heterogêneos feita através da utilização do *Apache Spark* via uma *API* com o objetivo de abstrair toda a coleta de informações vindas de fontes de dados independentes e heterogêneas.

Conforme dito anteriormente em 1.3, uma das propostas deste trabalho é dar ao usuário não somente uma interface para realizar as consultas aos bancos heterogêneos, mas também possibilitar que o usuário execute consultas sem necessidade de conhecimentos específicos de banco de dados. Assim, propomos uma solução onde o usuário possa selecionar os elementos que irão compor a consulta. Após a escolha, a consulta é montada automaticamente, independente do modelo e do tipo de dados envolvido.

4.1 Abordagem

Ao se utilizar integração de dados em algum projeto, o objetivo a ser alcançado é que seja possível realizar o processo de integração dos dados em tempo de execução, isto garante que os mesmos serão mantidos atualizados constantemente (RODRIGUES, 2018). Devido às várias funcionalidades que a arquitetura permite, realizar algumas tarefas podem vir a ser custosas, até mesmo para as mais simples.

Pensado em alguma forma de realizar este processo, é que a aplicação utilizada neste trabalho foi implementada. A proposta visa entregar uma interface ao usuário,

permitindo que o mesmo procure compreender e visualizar as estruturas dos bancos os quais ele está trabalhando, além de poder realizar consultas pertinentes em dois *BD*s ao mesmo tempo e também promovendo o objetivo de deixar as operações mais transparentes com consultas previamente implementadas.

Por trás desta interface foi feita a parte de integração de dados que realiza uma busca nos dois bancos utilizados, o *MySQL* e o *Neo4J*, de acordo com os campos que o usuário deseja realizar a pesquisa. Para que o usuário possa usufruir da aplicação, é necessário que um técnico execute todas as instalações corretas que serão tratadas de maneira mais especificada na próxima seção.

Por se tratar de um conjunto de rotinas e padrões estabelecidos e documentados por uma aplicação, o objetivo desta *API*[8] é de, justamente, fazer com que outras aplicações consigam utilizar as funcionalidades da aplicação principal, de modo que estas não precisam ter conhecimento de detalhes da implementação do *software*, por exemplo.

Pensando nisso, conforme explicado na introdução deste capítulo, é promovida uma interface *web* que oferece ao usuário a possibilidade dele poder escolher quais tabelas ele gostaria de utilizar, assim como as colunas das tabelas, de modo que sejam realizadas buscas com campos semelhantes em ambos os *BD*s.

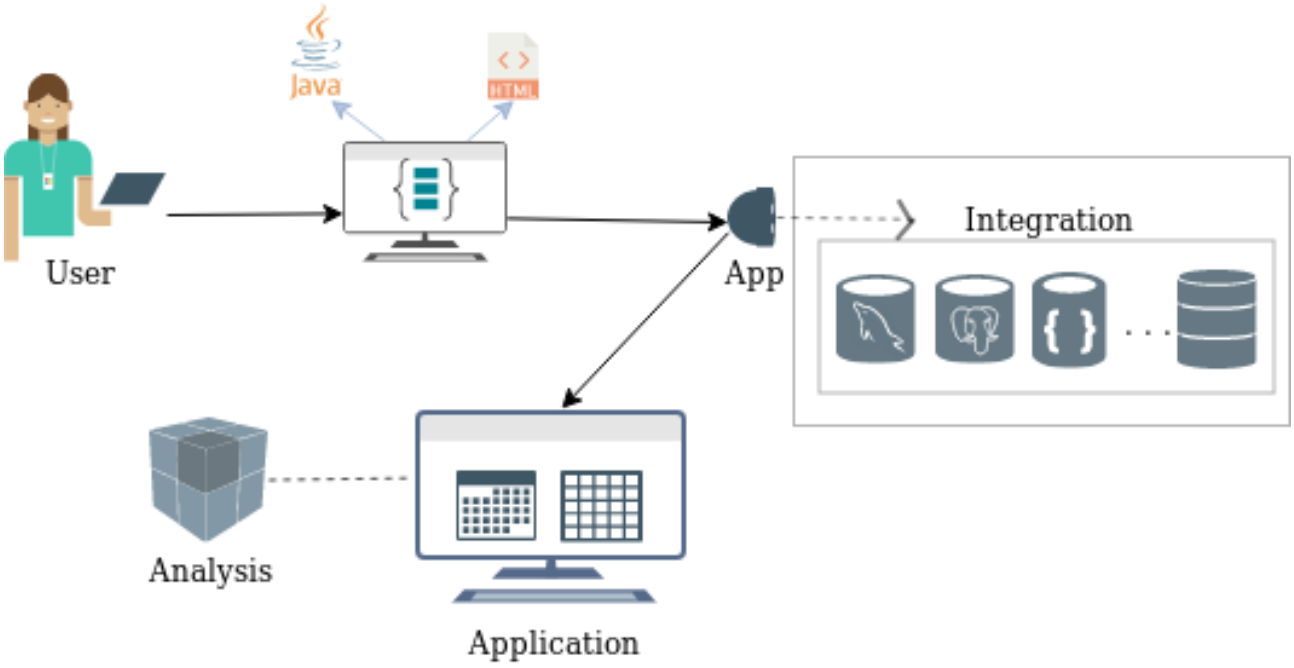

Figura 4.1: Representação da proposta

[8] ⟨https://becode.com.br/o-que-e-api-rest-e-restful/⟩

Podemos ver o funcionamento geral desta aplicação na Figura 4.1. Ao entrar na aplicação *web*, a mesma oferecerá as opções possíveis que podem ser realizadas, tais como:

- Instruções iniciais ao usuário;
- Quais *BD*s estão disponíveis para consulta, assim como tabelas e suas configurações;
- Possibilidade de escolher as funcionalidades que o usuário pode vir a querer operar.

A proposta descrita na figura 4.1 procura entregar ao usuário uma interface que tem por trás conexões com os *BD*s que ele deseja realizar consultas. O funcionamento da aplicação promove a integração de dados, realizando buscas em ambos os bancos e retornando aquilo que é pertinente a consulta efetuada. De modo que o usuário tenha como resultado impresso na tela sobre qual funcionalidade ele veio a escolher, assim como o que ele pesquisou.

Para que a etapa do usuário chegar na aplicação seja alcançada, é necessário que antes alguns pontos sejam executados, tais como: configuração correta, carregar e conectar aos bancos. A configuração consta na parte correta da instalação dos pré-requisitos necessários para eles serem executados. Já na parte de carregamento, é necessário escolher bases de dados que tratem de mesmo assunto (mesmo que em locais e tipos diferentes), realizar o tratamento dos dados, caso necessário. Por fim, realizar a conexão com os bancos. Além disto é necessáiro que seja possibilitado o uso da *IDE Eclipse*. Mais detalhes serão falados na próxima seção.

4.2 Desenvolvimento

Nesta etapa, alguns passos foram feitos antes da proposta ser desenvolvida. Primeiramente, é necessário arrumar o ambiente computacional como um todo para poder dar início ao desenvolvimento. Foi utilizada uma máquina com processador Intel® Core® $i5-4200U$ de $1.6GHz$, memória $6GB\ DDR3$, $500GB\ HDD$. A distribuição do sistema operacional utilizado foi a *Linux Ubuntu* 18.0.3 de $64bits$.

4.2.1 Preparação do ambiente

Foram instalados e utilizados: *Neo4J* 3.5.12; *OpenJDK* na versão 11.0.4; a versão 5.7.27 do *MySQL* que foi executado via linha de comando e o *Apache Tomcat*®(o servidor *web* para *JAVA*) na versão 7.0 e a *IDE Eclipse* na versão 4.13.0. Após as configurações iniciais e com os bancos em funcionamento e carregados, deu-se início ao processo de implementação da aplicação *web* através da linguagem *JAVA*.

As tecnologias utilizadas no desenvolvimento requerem de uma pessoa com conhecimento mais técnico para efetuar as instalações corretas; possivelmente o usuário da aplicação não saberá realizar estes passos, ele somente irá usufruir da aplicação final. Como a distribuição do sistema operacional é a *Ubuntu* 18.0.3, algumas medidas para efetuar a instalação do *MySQL* devem ser tomadas a nível de terminal, tais como:

1. Utilizar o comando *sudo apt update* todas as vezes em que um pacote for instalado;
2. *sudo apt install mysql-server* para efetuar o *download* do servidor de *MySQL*
3. Instalar utilizando o código: *sudo mysql_secure_installation*
4. Dar início ao servidor de *MySQL* criando um novo usuário e senha: *sudo -u root -p* colocando senha da máquina e senha padrão do *MySQL*. Após esse procedimento utilizar o comando *CREATE USER 'novousuario''localhost' IDENTIFIED BY 'password';* e garantir que esse novo usuário poderá ter acesso com *GRANT ALL PRIVILEGES ON * . * TO 'novousuario'@'localhost';*
5. Criar o Banco de Dados, utilizando linguagem *SQL*. Esta etapa pode variar de acordo com a base de dados escolhida. Pode ser que ela já venha em linguagem *SQL* já preparada, porém, ela pode estar em formato de tabela, sendo necessário um *script* realizar o processo de transformação dos dados da tabela para convertê-los em comandos *SQL*, para poder popular o banco adequadamente.
6. Utilizar o comando *USE nome_do_banco;* para utilizar o novo banco criado.

Já para o *Neo4J*, também são necessários alguns passos, são eles:

1. Iniciar com o comando *sudo su* para entrar como super usuário.
2. Utilizar o código *wget –no-check-certificate -O - https://debian.neo4j.org/neotechnology.gpg.key — sudo apt-key add -*
3. Exeutar o código *echo 'deb http://debian.neo4j.org/repo stable/' > /etc/apt/sources.list.d/neo4j.list*
4. Utilizar novamente o comando *apt update*
5. Terminar a instalação com o código *apt install neo4j*
6. Executar o servidor de *Neo4J* através do comando *service neo4j start*
7. Acessar *http://localhost:7474* endereço do servidor do *Neo4J* e efetuar a criação de um novo *login*
8. Utilizar, através da linguagem *Cypher* a execução do banco de filmes usando *play: movies*. Logo em seguida copiar o comando gerado pela aplicação e colar na execução e mandar rodar.

Após o processo de organizar os bancos e os servidores dos mesmos, efetua-se, de modo simples, a instalação dos demais componentes.

4.2.2 Explicações finais

Após as instalações iniciais, foi configurado o servidor *Neo4J* que recebeu como conjunto de dados um exemplo padrão chamado *Movies*[9], onde foi criado o banco no servidor. Esse banco, por sua vez, possui relacionamentos, através dos nós e arestas do grafo *Movies*, de filmes, atores, diretores e notas dadas por usuários aos filmes.

Assim como o *Neo4J* fora configurado, optou-se por trabalhar com o *MySQL*. Foi feito o *download* do *movies-metadata.csv*[10]. A base dados passou por um *script* feito em

[9] ⟨https://neo4j.com/developer/example-data/⟩
[10] ⟨https://www.kaggle.com/rounakbanik/the-movies-datasetmoviesmetadata.csv⟩

Python que montou os comandos *SQL* e removeu excessos e que, posteriormente, foram executados via terminal através de linha de comando do *MySQL*.

É necessário salientar que, além das informações descritas acima, uma tabela chamada *people* fora adicionada ao *MySQL*, de modo a procurar buscar relações entre o banco *MySQL* com o banco *Neo4J*. Esta tabela foi inclusa no banco relacional denominado de *movies_metadata* de acordo com a fonte previamente citada.

O objetivo de inclusão desta tabela é que ela contém algumas informações pertinentes a atores, atrizes, diretores, filmes e as indicações de prêmios que foram indicados a concorrer e se foram vitoriosos na categoria. Sendo assim, possibilita realizar consultas em ambos os *BD*s proporcionando a realização de um *merge*(mescla) de dados presentes em ambos.

Esta tabela descrita ao início deste parágrafo proveio da base de dados *The Academy Awards, 1927-2015*[11].

Após as configurações iniciais e com os bancos em funcionamento, deu-se início ao processo de implementação da aplicação *web* através da linguagem *JAVA*.

[11] ⟨https://www.kaggle.com/theacademy/academy-awards⟩

5 Ambiente e resultados

Experimentos podem variar de acordo com os tipos de trabalho acadêmicos e científicos, podendo ser experimentos teóricos com um aprofundamento em comparações, por exemplo, mas também podem ser experimentais. Conforme dito em 4, este trabalho enquadra-se em mostrar um estudo mais voltado a parte experimental. Sendo assim, o caráter da metodologia de pesquisa abordada nesta monografia enquadra-se como exploratória de cunho tecnológico e busca-se com ela, encontrar possíveis soluções a partir de experimentos.

5.1 Experimentos

Esta seção apresenta as *screenshots* tiradas diretamente da aplicação *web*. Conforme dito na seção 4.1, é necessário configurar a máquina para realizar os testes nos bancos. Para estes experimentos foram usados, a fins de demonstração, apenas dois *BD*s, porém é possível que outros possam ser utilizados caso a configuração da aplicação seja ajustada corretamente.

Antes mesmo de explicar sobre a aplicação é necessário levar em conta o processo de conexão dela com os *BD*s. Este processo leva em consideração uma poderosa ferramenta do *JAVA*, a *Apache Maven*^{TM}. É interessante notar que esta ferramenta leva em consideração a compilação primária de projetos em *JAVA*, o que facilitou muito a conexão com os bancos, uma vez que era necessário apenas colocar as dependências no arquivo *pom.xml* dentro da pasta *WebContent* do projeto e configurar dentro das *Servlets* do *JAVA*, que são basicamente classes que estendem funcionalidades de servidor.

O processo para começar fazer com que a aplicação funcione começa em conectar-se ao banco do *MySQL* desejado, digitando usuário e senha; após este processo é necessário realizar conexão com o *Neo4J* e executar o *Eclipse*. Uma vez com o *Eclipse* aberto, o *Tomcat* executando e as devidas instruções feitas, a aplicação está pronta para uso. Ela pode ser acessada no endereço *http://localhost:8080/web_application_polystore/Servlet1* e

aparecerá uma tela conforme figura abaixo:

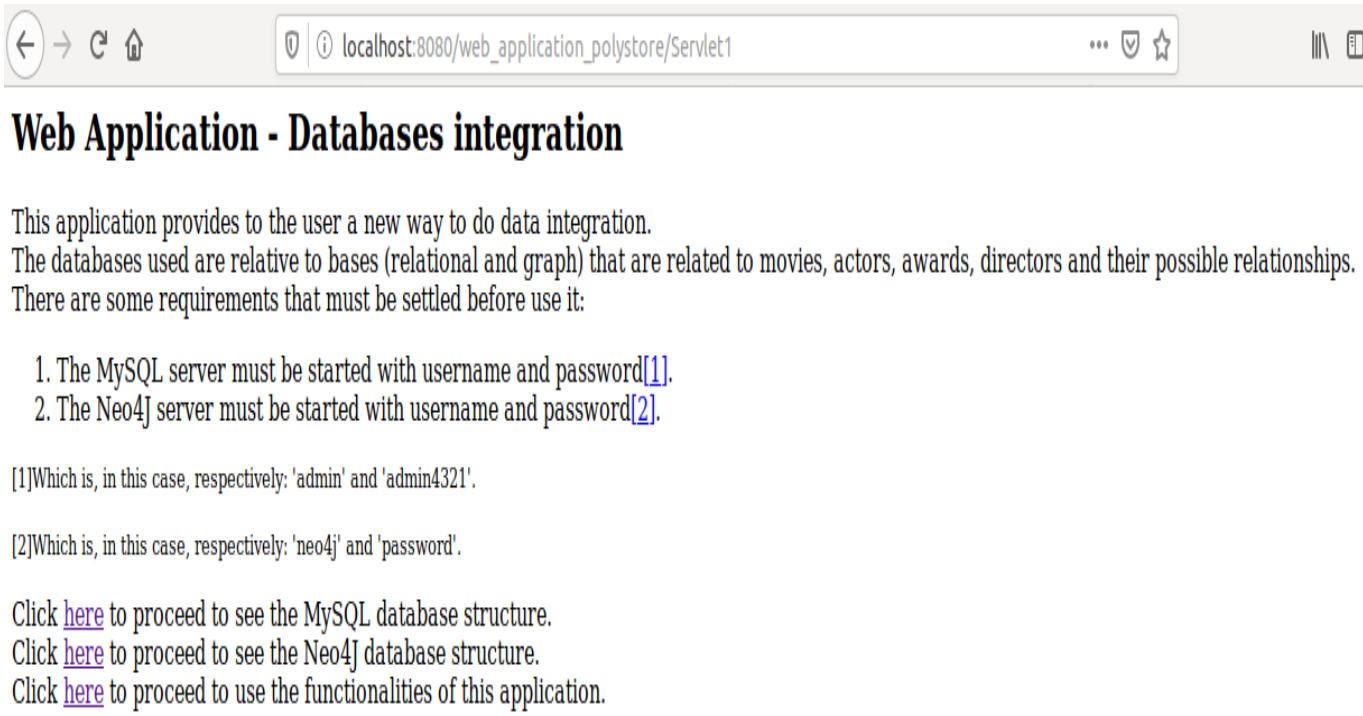

Figura 5.1: Tela inicial da aplicação, algumas instruções são dadas e opções de verificação e funcionalidades

Essa tela permite ao usuário uma certa familiarização com a ferramenta de modo a mostrar exatamente as opções possíveis, ela permite que o usuário verifique a estrutura das tabelas do banco relacional e da estrutura dos nós do banco orientado a grafos, caso o usuário queira verificar que tipos de dados estão sendo retornados e quais as colunas e propriedades de nós existem em cada um dos *BD*s. Isto pode ser visto nas figuras 5.2 e 5.3 demonstradas nas páginas seguintes.

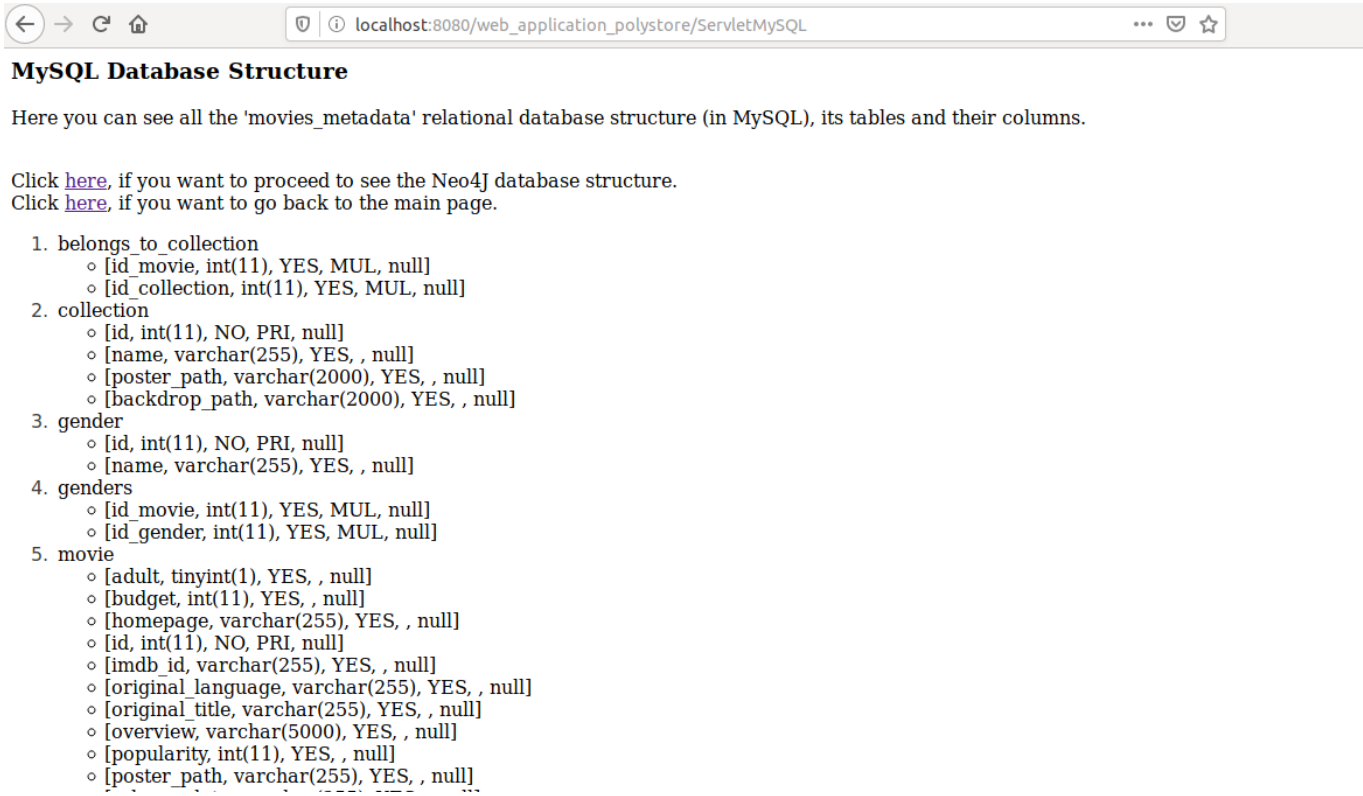

Figura 5.2: Opção da aplicação para verificar a estrutura do *BD MySQL*

As características relacionadas a algum filme podem variar não somente pelos tipos das variáveis, como também cria conjuntos de dados. Como é o caso da tabela *belongs_to_collection* que agrega filmes que possuem sequências a uma coleção.

Além destas características apresentadas na figura 5.2, existem outras seis tabelas inclusas nesse *BD* que são: *people*, *production_companies*, *production_company*, *production_countries*, *production_country*, *spoken_language* e *spoken_languages* .

Já as características do banco do *Neo4J* podem ser vistas na próxima imagem. É interessante notar que elas são menos numerosas e possuem características semelhantes às do banco relacional que, por sua vez, carrega mais dados.

Neste exemplo, foram mantidas apenas características dos nós, porém, podem ser inclusos, também, as características das relações, ou seja, filmes e pessoas possuem relações que podem ser, por exemplo: atores possuem a relação *ACTED_IN* para com um filme, diretores possuem a relação *DIRECTED*, assim também como pessoas podem possuir relações *FOLLOWS*, *REVIEWED* (caso ela tenha feito alguma resenha de algum filme) e *WROTE* (caso tenha participado da escrita de roteiro).

Essas telas proporcionam um ponto interessante, pois não força o usuário a ter

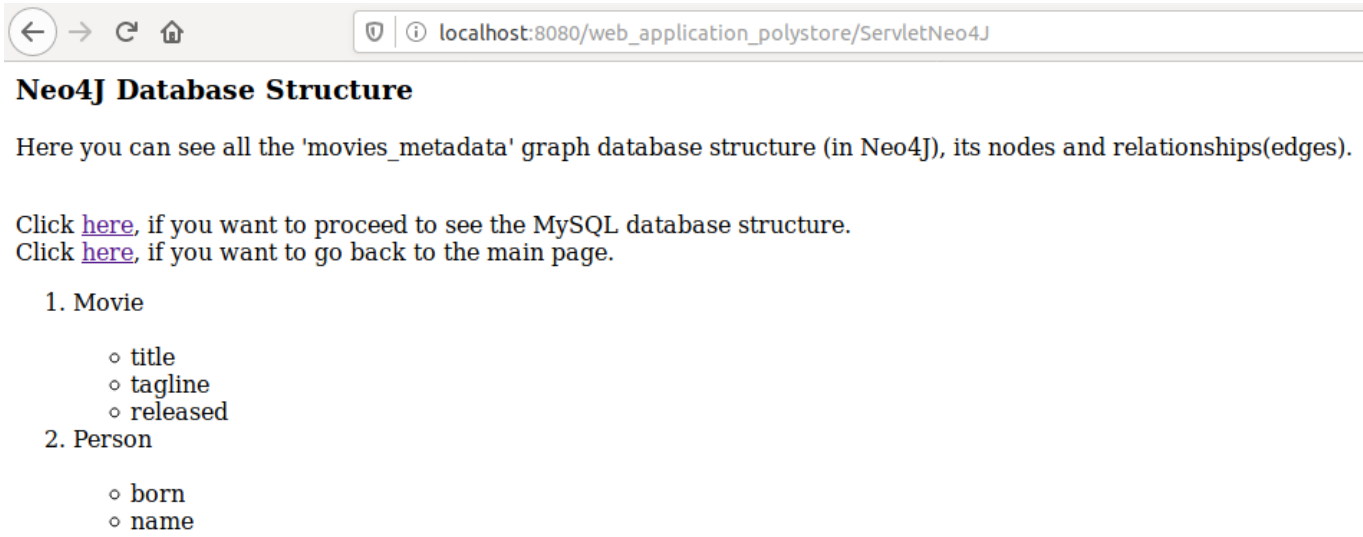

Figura 5.3: Opção da aplicação para verificar a estrutura do *BD Neo4J*

necessidade de decorar as tabelas, colunas, tipos de dados em campos, etc., basta apenas acessar essa parte na aplicação que ela retorna as estruturas.

Visto que a parte de apresentação das estruturas fora demonstrada, as telas a seguir representam as funcionalidades do sistema seguidas de comentários acerca delas.

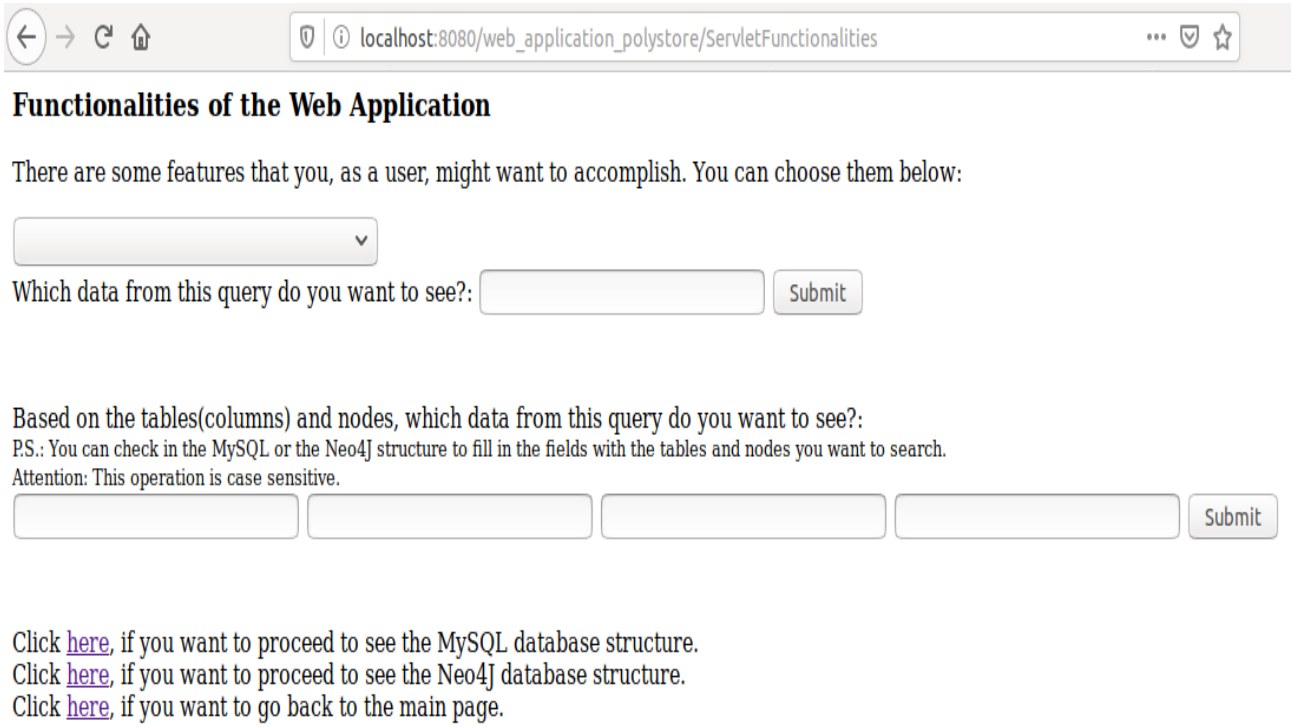

Figura 5.4: *Servlet* responsável por apresentar as funcionalidades da aplicação

É possível verificar na figura 5.4 que existem duas opções de se fazer uma consulta.

Na primeira, caso o usuário deseje, ele pode optar por selecionar, através do *ComboBox*, o tipo de consulta. Para fins de demonstração do funcionamento da aplicação, foram disponibilizadas algumas consultas previamente montadas. Elas foram inspiradas nas consultas realizadas por (RODRIGUES, 2018). São elas: *averageVoteMoviesOfActor*, *averageRevenueMoviesOfActor*, *averageVoteMoviesOfDirector*, *averageRevenueMoviesOfDirector*, *averageProfitMoviesOfDirector* e *popularityOfActor*.

Elas realizam buscas com relação a, respectivamente: votação média de filmes em que atores ou atrizes participaram; a rentabilidade de um ator ou atriz baseado nos filmes em que participou; votação média de filmes em que diretores estiveram a frente da direção; média de rentabilidade dos filmes que um diretor participou; média de lucro de filmes de um diretor e a popularidade de um ator ou atriz.

Suponhamos que o usuário deseje realizar uma consulta baseada na primeira opção, *averageVoteMoviesOfActor*, é desejável que ele possua um conhecimento da base

de dados para poder facilitar esta consulta. Uma vez que, caso necessário, esse usuário poderá buscar no banco desejado, caso tenha dúvidas com relação a algum dado estar presente nas estruturas ou não. Isto se torna possível pois entende-se que este usuário possui prévio conhecimento de consultas em *BD*s.

O resultado desta consulta pode ser verificado nas figuras a seguir:

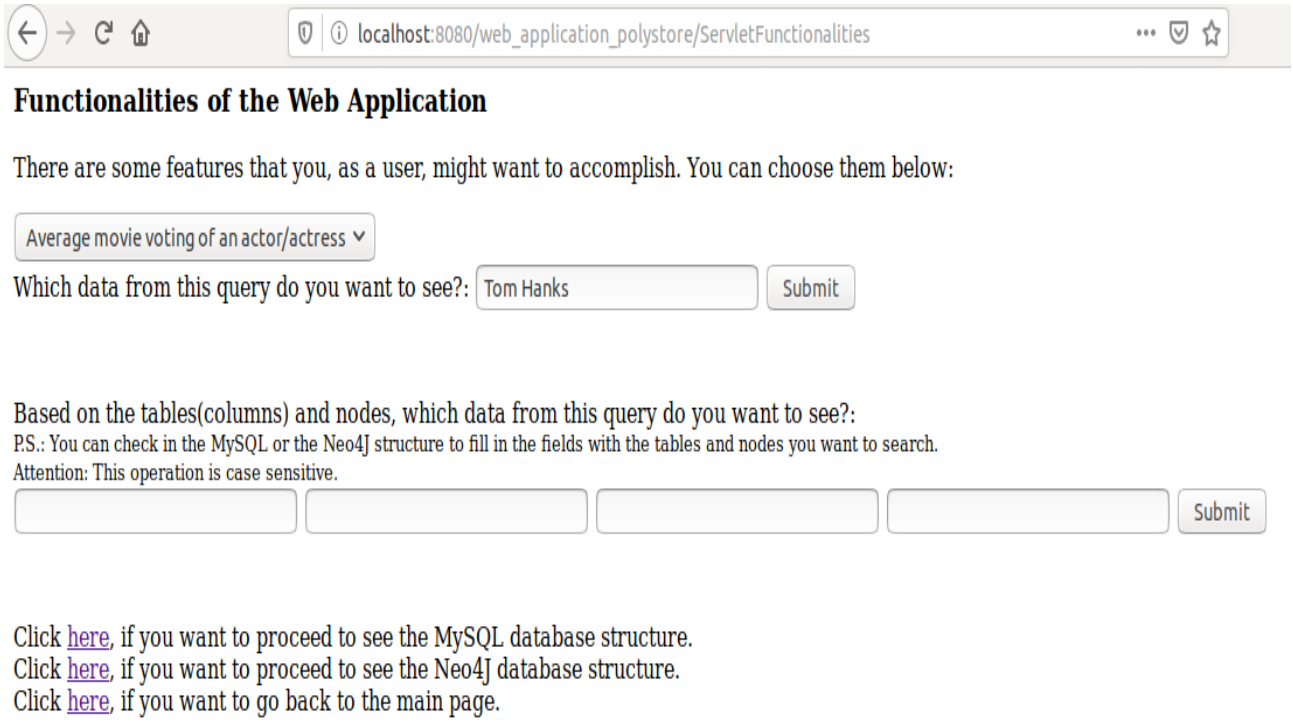

Figura 5.5: Exemplo do desempenho da funcionalidade de consultar a votação média de filmes em que um determinado ator ou atriz participou

Com relação ao resultado obtido, extrai-se a informação de que filmes os quais o ator Tom Hanks participou, consta-se que os mesmos possuem votação média de, aproximadamente, 7.26.

Nas próximas páginas são apresentados esses resultados relativos a experimentos das funcionalidades *averageVoteMoviesOfActor* e *popularityOfActor*.

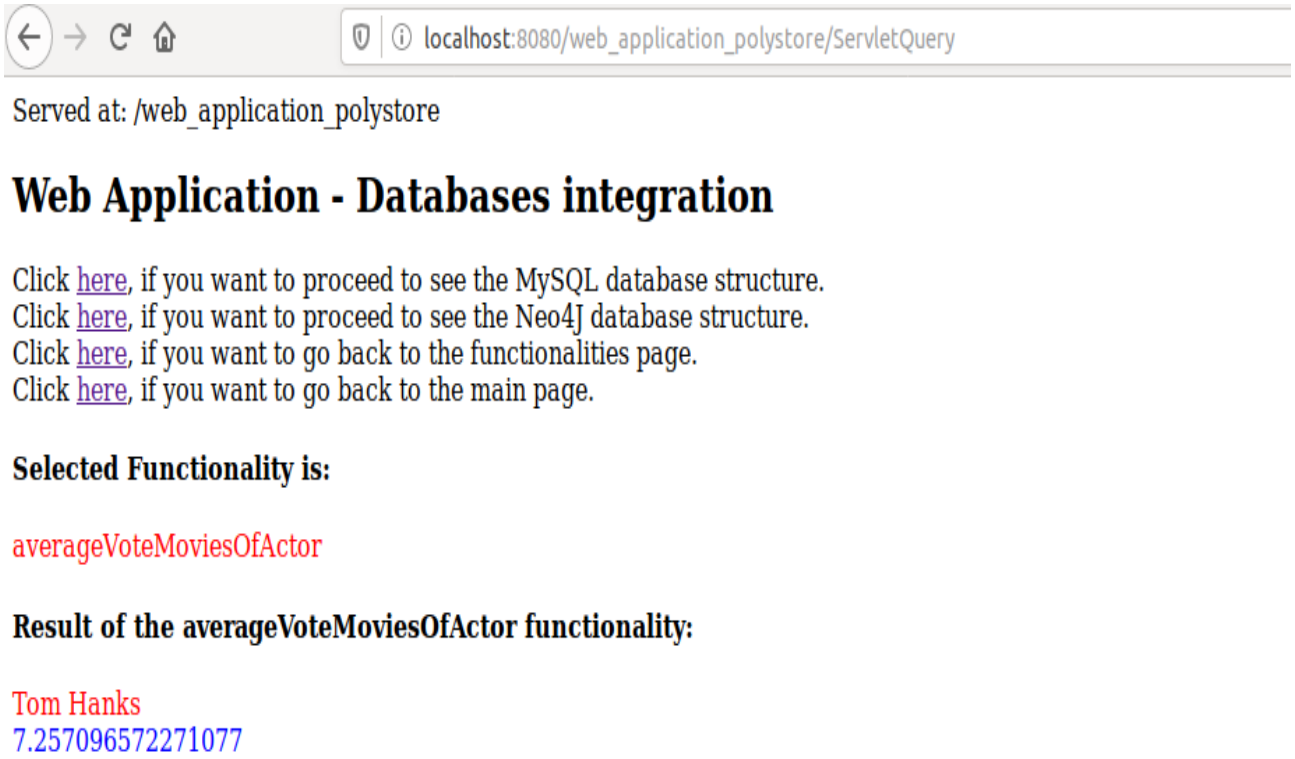

Figura 5.6: Resultado do exemplo de consulta a votação média de filmes de um determinado ator ou atriz dado na figura 5.5

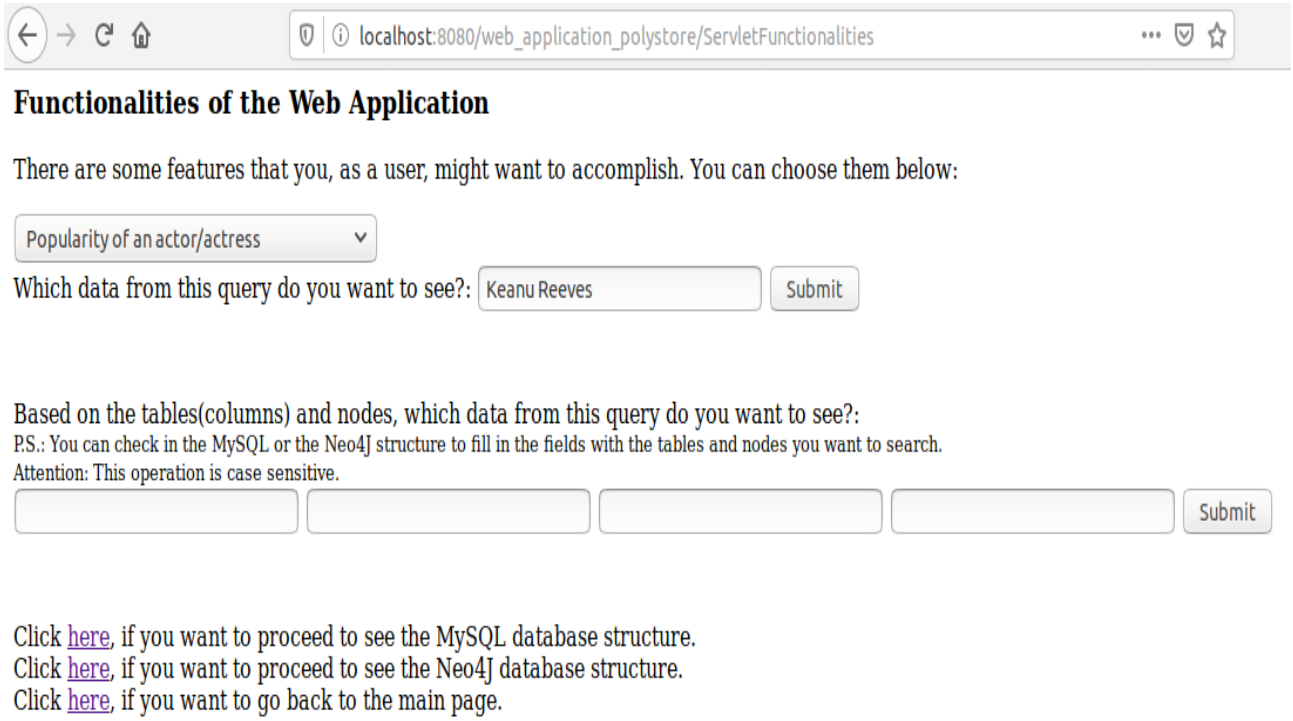

Figura 5.7: Exemplo do desempenho da funcionalidade de consultar a popularidade de um ator ou atriz

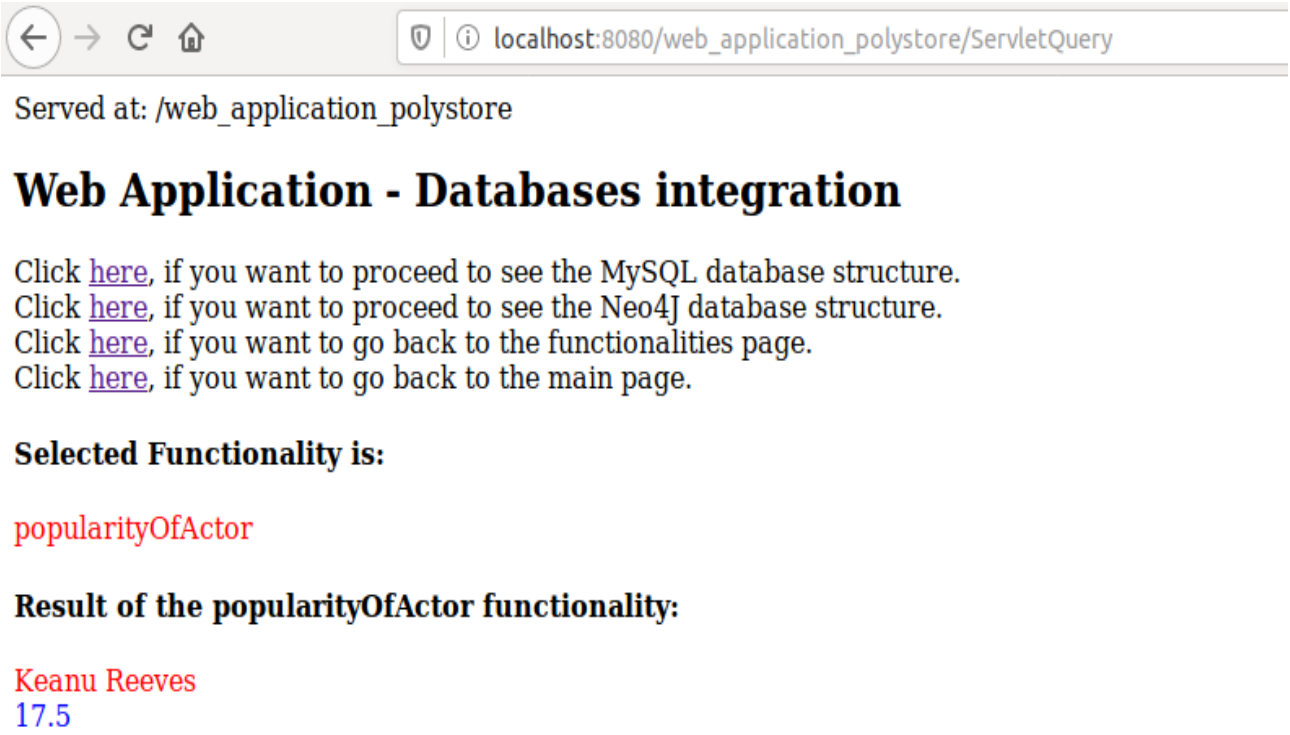

Figura 5.8: Resultado do exemplo de consulta a popularidade um determinado ator ou atriz, dado na figura 5.7

As figuras 5.5, 5.6, 5.7 e 5.8 expostas nas páginas anteriores demonstram que é possível refinar as buscas e alterar nomes de diretores e atores, por exemplo.

A próxima funcionalidade demonstrada na figura 5.4 é a que realiza um *merge*, ou seja, uma mescla, de ambos os *BD*s presentes na aplicação. Sendo assim, é necessário a atenção do usuário para que esta etapa seja realizada com sucesso.

Como os campos de pesquisa baseiam-se exclusivamente nos nomes das colunas e dos nós, é necessário que a escrita deles seja de forma correta. Sendo assim, a aplicação é *case sensitive* (campo sensível a letras maiúsculas e minúsculas).

A parte interessante disso é que, originalmente, o banco relacional de filmes possuía todas as tabelas descritas em 5.1 menos a *people*. Esta última fora assim nominada para facilitar a consulta relativa a atores e atrizes que possuem registros tanto em um banco como no outro e que fazem relação a não somente ao nome e ano de nascimento (como no caso do banco *Neo4J*) e nem somente nas consultas disponibilizadas, mas ele busca conteúdos iguais em ambos *BD*s e exibe ao usuário como resultado.

Para facilitar a explicação dada anteriormente, é possível verificar a aplicação dessa funcionalidade nas figuras a seguir:

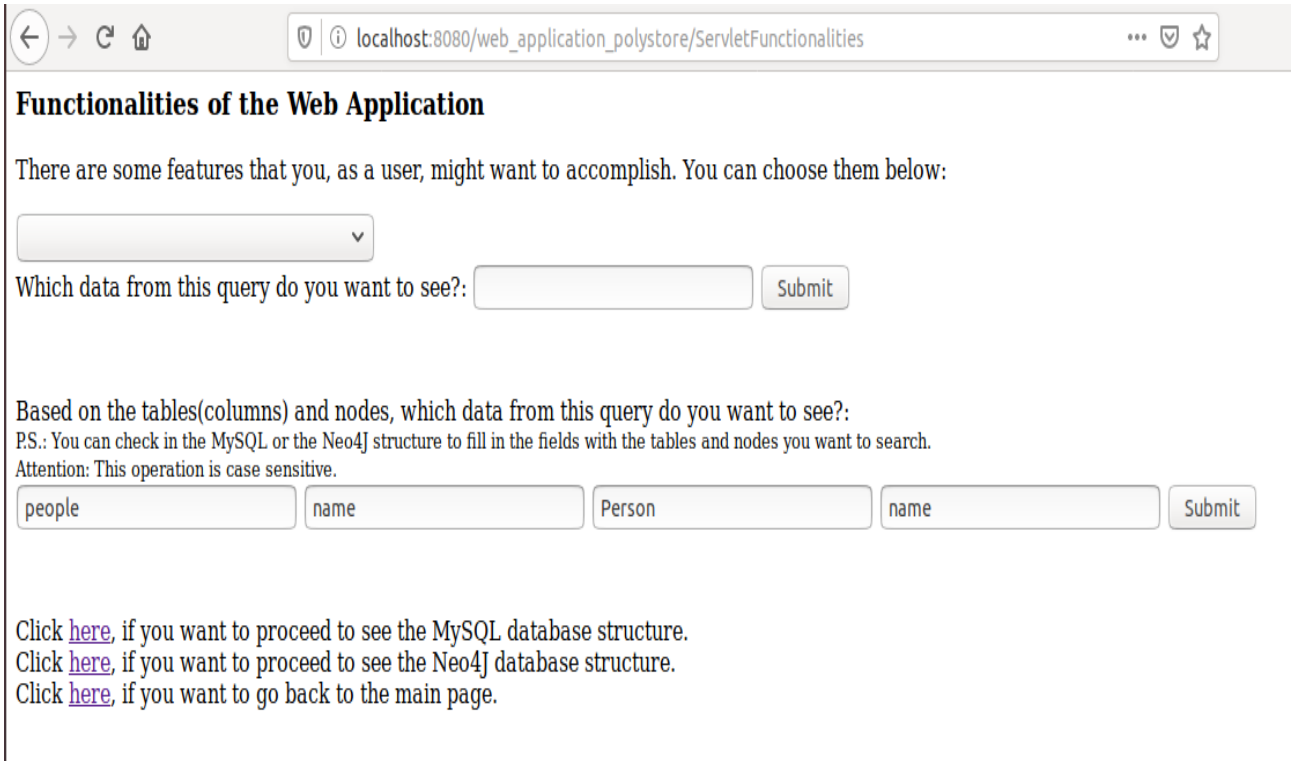

Figura 5.9: Exemplo do desempenho da funcionalidade de consultar em ambos os *BD*s com relação a pessoas

É interessante verificar que ambos os bancos tem a oferecer dados, porém, somente com um *merge* foi possível extrair mais resultado pertinente aos dados relativos à pessoas desses bancos.

Como esse exemplo se referiu a pessoas, nota-se que o *merge* foi bem sucedido pois buscou valores presentes na tabela *people* com o campo *name*, para o relacional e o nó *Person* com o campo *name*, para o orientado a grafos.

O acréscimo da tabela *people* propocionou uma busca mais apurada, pois, além do nome, ela exibe o tipo de premiação ao Oscar o qual o ator ou atriz em questão veio a concorrer.

Neste exemplo, caso o ator ou atriz tenha logo após o nome da premiação, o valor zero, significa que ele ou ela não ganhou esta premiação no ano em que ela foi dada, apenas foi indicado(a). Sendo assim, de forma análoga, caso tenha valor um, significa que ele ou ela ganhou o prêmio.

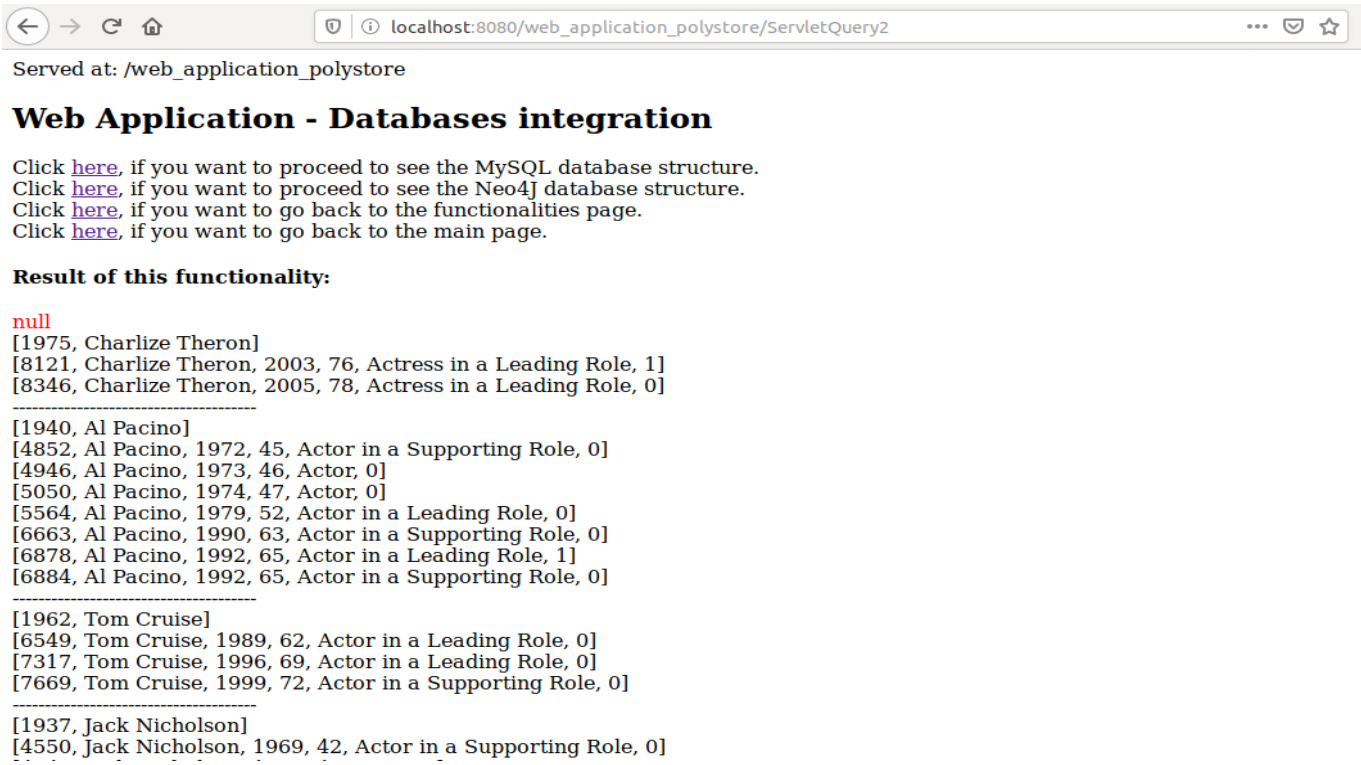

Figura 5.10: Resultado do exemplo de consulta a ambos os bancos para o campo de pessoas dado na figura 5.9

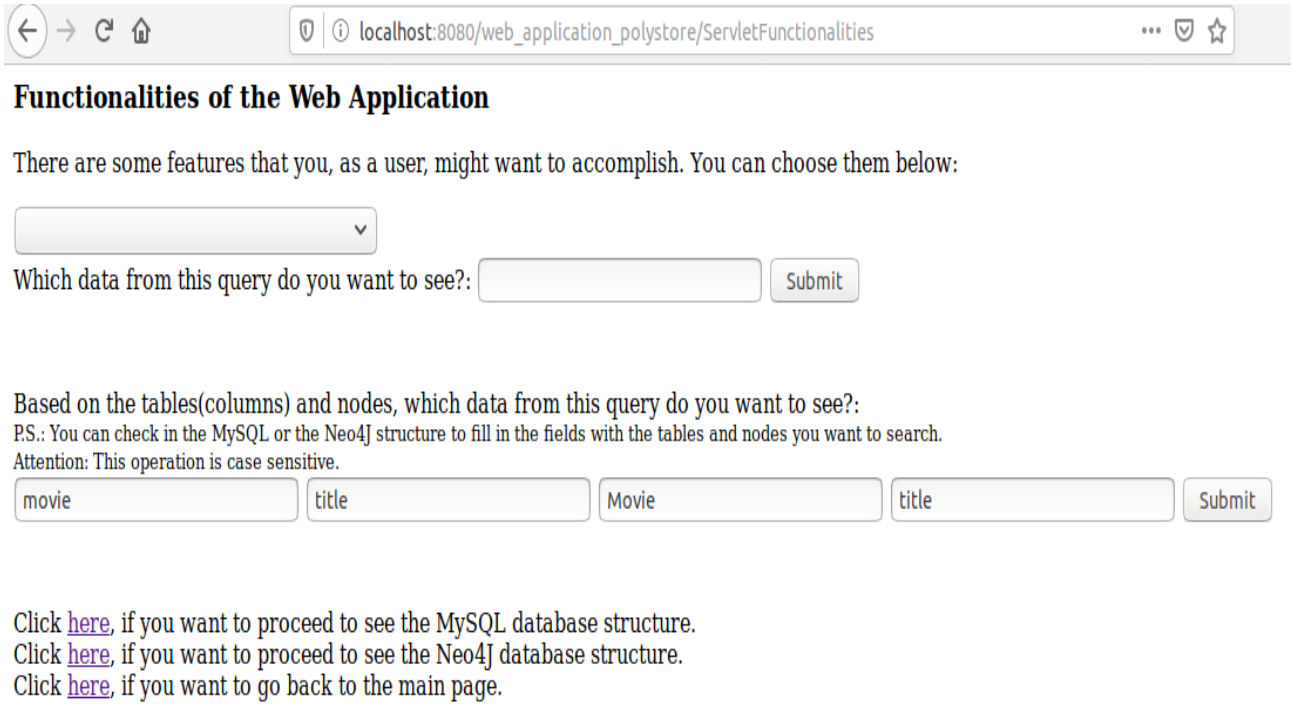

Figura 5.11: Exemplo do desempenho da funcionalidade de consultar em ambos os *BD*s com relação a filmes

De forma semelhante, é feito também um experimento usando a tabela *movie* com o campo *title* e do nó *Movie* com o campo *title*. O resultado é dado na figura 5.12.

A primeira linha de cada resultado, é relacionada ao obtido através do *Neo4J* com o nome do filme, uma *tagline* referente a ele (como se fosse um subtítulo) e o ano de estreia. Os primeiros campos da segunda linha em diante, até a marcação que delimita a próxima consulta, representam características que podem ser conferidas em 5.2 que são: se o filme possui classificação para adultos, o investimento, *id* do *IMDb*, depois a língua falada, título, sinopse, etc.

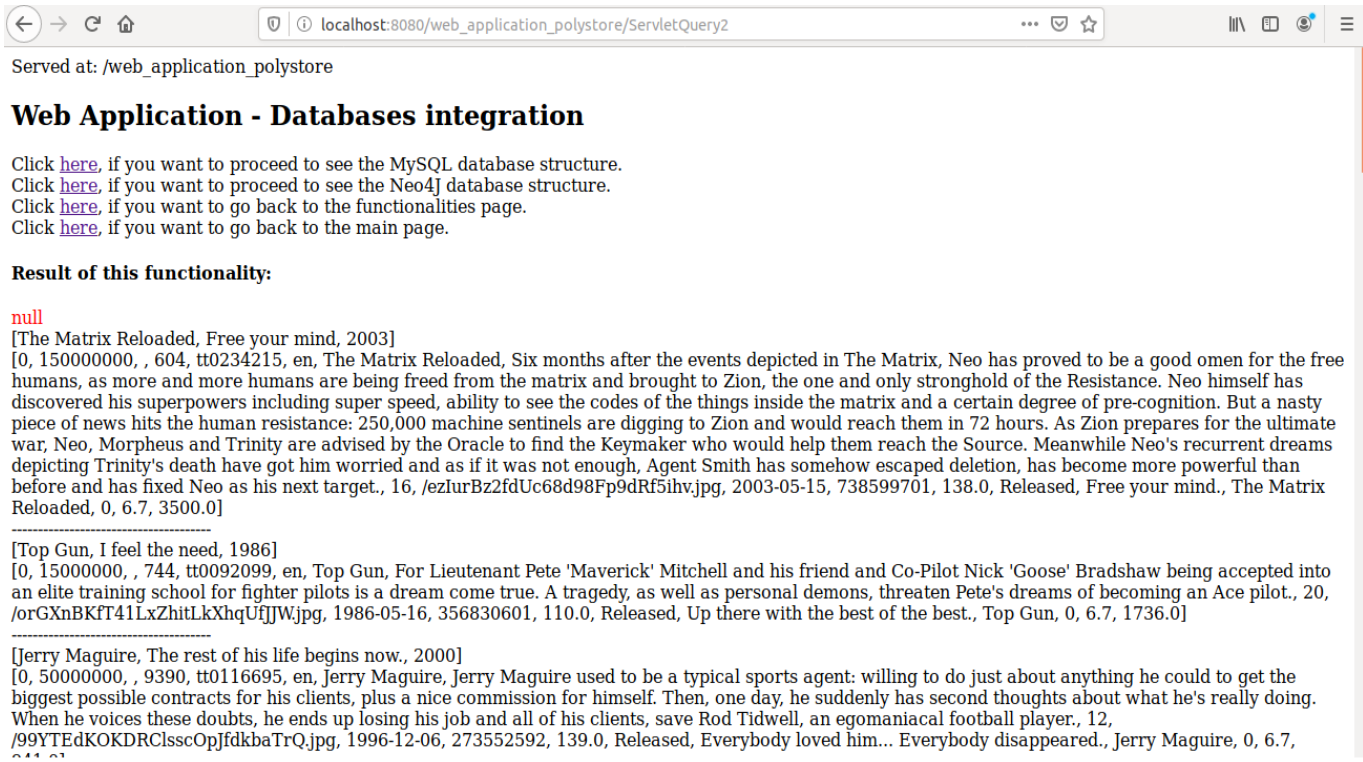

Figura 5.12: Resultado do exemplo de consultar filmes em ambos os bancos dado na figura 5.11

6 Considerações Finais

Este trabalho promoveu a realização da implementação de uma ferramenta do tipo aplicação *web* que realizou o trabalho de fazer com que fosse possível uma integração de dados heterogêneos.

A contribuição da proposta apresentada foi de justamente elaborar o processo de integração de dados de forma que pudesse ser exposto em forma de aplicação. Com os experimentos, nota-se que, através de uma abordagem mais simples, mas eficaz, foi possível realizar as consultas aos bancos, assim como para verificar e testar consultas já prontas. Promover uma ferramenta intuitiva ao usuário para realizar tarefas é algo crucial, assim como exemplificar o funcionamento das estruturas, sem que o mesmo tenha que recorrer a operar através de uma linha de comando ou ter que instalar outros *softwares*, o que pode vir a ser mais desgastante e complicado.

6.1 Trabalhos Futuros

Ao longo do desenvolvimento e depois da mostra de resultados, verificou-se a possibilidade de procurar permitir futuramente ao usuário que ele possa realizar consultas de maneira mais livre, sem ser recorrer a consultas pré-processadas. Isto facilita não somente para que a compreensão do sistema seja melhor, como também possibilita ao usuário uma maneira de melhorar as consultas, trazendo assim, possivelmente, resultados mais sofisticados, assim como promover uma interface amigável ao usuário.

Conforme dito em alguns pontos deste trabalho, o trabalho realizado por (RODRIGUES, 2018) procurou realizar esta integração utilizando um *framework* chamado *Apache Spark* que permite diversas funcionalidades próprias, além das presentes na *API* que fora desenvolvida pelo mesmo. Seria interessante, assim, realizar uma junção deste trabalho para como o acima exposto.

Bibliografia

AALST, W. V. D. Data science in action. In: *Process Mining*. [S.l.]: Springer, 2016. p. 3–23.

CODD, E. F. A relational model of data for large shared data banks. *Communications of the ACM*, ACM, v. 13, n. 6, p. 377–387, 1970.

DUGGAN, J. et al. The bigdawg polystore system. *SIGMOD Rec.*, ACM, New York, NY, USA, v. 44, n. 2, p. 11–16, ago. 2015. ISSN 0163-5808. Disponível em: ⟨http://doi.acm.org/10.1145/2814710.2814713⟩.

ELMORE, A. et al. A demonstration of the bigdawg polystore system. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 8, n. 12, p. 1908–1911, 2015.

GADEPALLY, V. et al. The bigdawg polystore system and architecture. In: IEEE. *2016 IEEE High Performance Extreme Computing Conference (HPEC)*. [S.l.], 2016. p. 1–6.

GANDOMI, A.; HAIDER, M. Beyond the hype: Big data concepts, methods, and analytics. *International journal of information management*, Elsevier, v. 35, n. 2, p. 137–144, 2015.

GUERRA, F. et al. Big data integration of heterogeneous data sources: the re-search alps case study. In: IEEE. *2019 IEEE International Congress on Big Data (BigDataCongress)*. [S.l.], 2019. p. 106–110.

KUMAWAT, D.; PAVATE, A. Correlation of nosql sql database. In: . [S.l.: s.n.], 2019.

LANEY, D. *3-D Data Management: Controlling Data Volume, Velocity and Variety. META Gr. Res.* [S.l.], 2001.

MASSER, I. Data integration research. *Geographic Information Systems to Spatial Data Infrastructures: A Global Perspective*, CRC Press, p. 49, 2019.

PAVLO, A.; ASLETT, M. What's really new with newsql? *ACM Sigmod Record*, ACM, v. 45, n. 2, p. 45–55, 2016.

POKORNý, J. Nosql databases: a step to database scalability in web environment. In: . [S.l.: s.n.], 2011. v. 9, p. 278–283.

RODRIGUES joão P. F. *Um estudo de integração de dados heterogêneos*. 2018. Monografia (Bacharel em Ciência da Computação), UFJF (Universidade Federal de Juiz de Fora), Juiz de Fora, Minas Gerais, Brasil.

SILBERSCHATZ, A. et al. *Database system concepts*. [S.l.]: McGraw-Hill New York, 1997. v. 4.

STONEBRAKER, M. Sql databases v. nosql databases. *Commun. ACM*, ACM, New York, NY, USA, v. 53, n. 4, p. 10–11, abr. 2010. ISSN 0001-0782. Disponível em: ⟨http://doi.acm.org/10.1145/1721654.1721659⟩.

STROHBACH, M. et al. Big data storage. In: _____. [S.l.: s.n.], 2016. p. 119–141. ISBN 978-3-319-21568-6.

VIRGILIO, R. D.; MACCIONI, A.; TORLONE, R. Model-driven design of graph databases. In: SPRINGER. *International Conference on Conceptual Modeling*. [S.l.], 2014. p. 172–185.

WU, X.; ZHANG, J.; WANG, F.-Y. Stability-based generalization analysis of distributed learning algorithmsfor big data. *IEEE transactions on neural networks and learning systems*, IEEE, 2019.