UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

BERNARDO SAAB MARTINIANO DE AZEVEDO

PROCESSO DE PUBLICAÇÃO DE DADOS ABERTOS MULTIDIMENSIONAIS EM BANCOS DE DADOS NoSQL

RIO DE JANEIRO
2016

Bernardo Saab Martiniano de Azevedo

PROCESSO DE PUBLICAÇÃO DE DADOS ABERTOS MULTIDIMENSIONAIS EM BANCOS DE DADOS NoSQL

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto de Matemática da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Professora Maria Luiza Machado Campos, Ph.D.

Rio de Janeiro

2016

CIP - Catalogação na Publicação

A994p

Azevedo, Bernardo Saab Martiniano de
Processo de publicação de dados abertos multidimensionais em bancos de dados NoSQL / Bernardo Saab Martiniano de Azevedo. -- Rio de Janeiro, 2016.
96 f.

Orientadora: Maria Luiza Machado Campos.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Bacharel em Ciência da Computação, 2016.

1. NoSQL. 2. Cassandra. 3. Padrão dados abertos. 4. Banco de dados orientado a colunas. 5. Monitoramento de rede. I. Campos, Maria Luiza Machado , orient. II. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

Bernardo Saab Martiniano de Azevedo

PROCESSO DE PUBLICAÇÃO DE DADOS ABERTOS MULTIDIMENSIONAIS EM BANCOS DE DADOS NoSQL

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto de Matemática da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado por:

Profa. Maria Luiza Machado Campos, Ph.D.
(Orientadora)

Profa. Ana Carolina Brito de Almeida, D.Sc.

Profa. Giseli Rabello Lopes, D.Sc.

Rio de Janeiro
2016

AGRADECIMENTOS

Agradeço aos meus pais, que sempre me apoiaram em todos os momentos difíceis durante a realização deste trabalho.

À orientadora, Maria Luiza, sempre dedicada em apoiar, através de sua disciplina e iniciativa de acompanhar meu desenvolvimento, de forma a direcionar os esforços para a conclusão de todos os objetivos propostos.

Por fim, agradeço aos colegas de curso que estiveram comigo ao longo dos últimos anos.

RESUMO

Com o advento da vasta quantidade de dados com que as organizações se propõem a lidar, ainda há uma discussão sobre as vantagens do consagrado modelo relacional. Apesar dos esforços para a melhoria da representação destes dados neste modelo, ainda foram encontrados outros desafios como a capacidade de lidar com a diversidade de informações decorrentes do volume de dados. Isto significa que podemos encontrar não apenas conjuntos de dados caracterizados por diferentes formatos, como também oriundos de fontes distintas, o que pode levar a limitações básicas no que diz respeito à modelagem de dados. Esta motivação fez surgir uma nova categoria de banco de dados, conhecida como NoSQL. Neste trabalho, apresentamos as principais características desses bancos de dados, que são complementadas através de uma abordagem prática, com a comparação entre dois tipos de bancos de dados em um contexto particular. Nesse contexto, foram realizados experimentos sobre o mesmo conjunto de dados, utilizando o armazenamento em um banco de dados MySQL, que representa o modelo relacional, e o Apache Cassandra, na representação do modelo de dados não-relacional e livre de esquemas. Além da comparação do desempenho entre eles, também foi investigada a solução mais funcional e adequada para uma aplicação de publicação de dados, com o intuito de estimular futuras implementações e correlações destas análises, adicionando novas fontes de dados e considerando a integração de fontes de dados heterogêneas.

Palavras-chave: NoSQL. Cassandra. Padrão Dados Abertos. Banco de Dados Orientado a Colunas. Monitoramento de Rede. Processo de Publicação de Dados.

ABSTRACT

With the arrival of large amount of data that organizations have to deal with, there is a discussion of the benefits of the well-known relational model. Despite efforts to improve the representation of data in this model, people also come across challenges such as the ability to deal with the diversity of information resulting from data volume. That means that we can find data sets characterized not only by different formats but also from many sources , which could lead into data modeling process limitations. This motivation has given rise to a new category of database, known as NoSQL. In this work, the main features of these databases are presented, and complemented through a practical approach, comparing two types of databases in a particular context. Some experiments on the same data set were performed on MySQL database, for storage in the approach of the relational model, and on Apache Cassandra database, for non-relational and schema-less data model. Beyond the comparison of performance between both databases, it was also explored the most functional and appropriate solution for data publishing application, in order to encourage future implementations and correlations of these studies, with the addition of new sources of databases and considering the integration of heterogeneous data sources.

Keywords: NoSQL. Cassandra. Open Data Standard. Column Oriented Database. Networking Monitoring. Data Publication Process.

LISTA DE FIGURAS

Figura 1 - Modelo Multidimensional - Desenvolvimento do esquema inicial. 44

Figura 2 - Modelo Multidimensional do PingER e fontes de dados complementares, baseado no esquema floco de neve (Snow Flake). 45

Figura 3 - Consulta em SPARQL utilizando o endpoint sobre a DBPedia. 49

Figura 4 - Descrição da falha ao carregar mais de 1.000.000 de linhas no Instaview 53

Figura 5 - Tabelas utilizadas para elaborar as consultas. 57

Figura 6 - Resultado da execução da consulta 1 (parte 1). 59

Figura 7 - Resultado da execução da consulta 1 (parte 2). 59

Figura 8 - Resultado da execução da consulta 2. 60

Figura 9 - Resultado da execução da consulta 3. 61

Figura 10 - Resultado da execução da consulta 4. 61

Figura 11 - Resultado da execução da consulta 5. 62

Figura 12 - Resultado da execução da consulta 6. 63

Figura 13 - Resultado da execução da consulta 7. 64

Figura 14 - Resultado da execução da consulta 8. 64

Figura 15 - Resultado da execução da consulta 10. 65

Figura 16 - Tempo de carga da tabela fato. 69

Figura 17 - Tempos de carga das tabelas dimensão no Cassandra. 69

Figura 18 - Exemplo de execução - detalhamento da consulta 5. 71

LISTA DE ABREVIATURAS E SIGLAS

BD	Banco de Dados
BI	Business Intelligence
DW	Data Warehouse
ETL	Extraction, Transformation and Loading
FTP	File Text Protocol
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
IEPM	Internet End-to-end Performance Monitoring
LOD	Linked Open Data
NoSQL	Not Only SQL
PingER	Ping End-to-end Reporting
RDF	Resource Description Format
SGBD	Sistema de Gerenciamento de Banco de Dados
SLAC	Stanford National Accelerator Laboratory
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
UFRJ	Universidade Federal do Rio de Janeiro
URL	Unified Resource Locator
WWW	World Wide Web

SUMÁRIO

1 INTRODUÇÃO 10
1.1 MOTIVAÇÃO 10
1.2 COMPLEXIDADE DA INFORMAÇÃO 11
1.3 VALOR DOS DADOS 12
1.4 ESTÁGIO ATUAL DA TECNOLOGIA 14
1.5 OBJETIVO GERAL 15
1.6 ESTRUTURA DO TRABALHO 16

2 O PROJETO PINGER E OS DADOS DE MEDIÇÕES DE DESEMPENHO DA INTERNET 18
2.1 PROJETO MultiLOD 18
2.2 PROJETO PingER 19
2.2.1 Geração dos dados do PingER e métricas calculadas 20
2.2.2 Acesso aos dados 21
2.3 PingER LOD 22
2.4 CONSIDERAÇÕES FINAIS 24

3 SISTEMAS DE GERENCIAMENTO DE BANCOS DE DADOS RELACIONAIS E NÃO RELACIONAIS 25
3.1 SISTEMAS DE GERENCIAMENTO BASEADOS NO MODELO RELACIONAL 25
3.2 SISTEMAS DE GERENCIAMENTO DE BANCO DE DADOS NÃO RELACIONAL — NoSQL 27
3.3 MODELOS DE DADOS NoSQL 29
3.4 EXEMPLOS DE TECNOLOGIAS NoSQL 32
3.4.1 HBase 32
3.4.2 Apache Cassandra 33
3.5 BANCOS DE DADOS NoSQL VERSUS BANCOS DE DADOS RELACIONAIS 35

4 EXPERIMENTAÇÃO: TRATAMENTO E ACESSO DE DADOS DO PINGER EM BANCOS DE DADOS RELACIONAL E NoSQL 37
4.1 CARACTERÍSTICAS DO AMBIENTE DE TESTES 38
4.1.1 Sistema computacional, SGBDs e outras ferramentas 38
4.1.2 Características da aplicação 39
4.2 PLANEJAMENTO E MODELAGEM MULTIDIMENSIONAL 41
4.2.1 Descrição das fontes de dados complementares 42
4.2.2 Criação do esquema inicial 43

4.3 PROCESSOS DE ETL 45
4.3.1 Processo de tratamento e carga dos dados no banco de dados relacional 46
4.3.2 Processo de tratamento e carga dos dados no Cassandra 51
4.4 CONSULTAS ANALÍTICAS 56
4.4.1 Consultas analíticas realizadas 58
4.4.2 Comparação do desempenho nas consultas analíticas 66
4.5 ANÁLISE DOS RESULTADOS E CONSIDERAÇÕES GERAIS 71

5 CONCLUSÃO 75
5.1 CONSIDERAÇÕES FINAIS 75
5.2 TRABALHOS FUTUROS 76

REFERÊNCIAS 78

APÊNDICE A - CONSULTA SPARQL PARA GERAR O MAPA DE MÉTRICA DE REDE X MÉTRICA DE UNIVERSIDADE 81

APÊNDICE B - SCRIPT DE CRIAÇÃO DAS TABELAS DO MODELO MULTIDIMENSIONAL NO MYSQL 83

ANEXO A - DEFINIÇÃO DAS ETAPAS DO PROCESSO DE ETL 87

ANEXO B - GLOSSÁRIO DOS DADOS DO PROJETO PINGER 89

ANEXO C - GLOSSÁRIO DOS TERMOS UTILIZADOS NA MODELAGEM 92

ANEXO D - SCRIPT DE CRIAÇÃO DO MODELO FÍSICO NO CASSANDRA 96

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Organizações que se propõem a obter uma atuação relevante no mercado precisam usar a informação como ferramenta estratégica nas suas decisões. Para isso, buscam competitividade e qualidade dos seus serviços através da aplicação constante do tratamento dos seus dados. Contribuem com essa prática os sistemas de informação analíticos, proporcionando aos líderes de negócio a análise crítica dos dados a partir de uma base de dados que ofereça suporte à tomada de decisão. O projeto de um banco de dados apropriado é de suma importância para que o Sistema de Informação cumpra seu papel, agregando valor aos dados da organização e transformando-os em informações úteis.

Além disso, saber como encontrar as informações, como apresentá-las e utilizá-las, mostra-se tão importante quanto conhecê-las. Já prevendo o excesso da informação online e a dificuldade para obter fontes de dados que fossem confiáveis, afirmava-se, de forma peculiar, que, no contexto do gerenciamento de informações, cerca de 20% das informações eram responsáveis por 80% das decisões de uma organização. (KANAZAWA, 2006).

É um consenso que a maioria das empresas não consegue obter toda a informação necessária quando devem tomar uma decisão estratégica. Na realidade, podemos supor que estas decisões são quase sempre baseadas em informações incompletas. Como exemplo, ao comprarmos um carro nem sempre pensamos em todas as variáveis que justificariam a nossa escolha e na maioria das vezes acabamos por decidir através das características mais relevantes e prioritárias. Desse modo, os líderes mais experientes que entendem este princípio sabem que algumas informações despendem um enorme esforço para que sejam recuperadas e para que as decisões sejam fundamentadas com mais qualidade. Porém, ao aplicar a seletividade para o conjunto de dados mais importante eles conseguem ainda assim tomar decisões bastante acuradas, pois é compreensível que aguardar pela outra parte da informação pode prejudicar o processo como um todo.

Um dos grandes desafios computacionais da atualidade é armazenar, manipular e analisar, de forma inteligente, a grande quantidade de dados existente. É importante ressaltar que estamos sempre nos referindo a um enorme conjunto de informações coletadas, e a como gerar dados de valor dessas informações numa base de dados.

Sistemas corporativos, serviços e sistemas Web, tais como redes sociais e ferramentas de busca com acesso a bases de dados de imagens e mapas, além de diversas fontes

heterogêneas, tais como documentos de texto, arquivos de áudio e vídeo, produzem um volume de dados acima da capacidade de processamento realizada por tecnologias tradicionais de banco de dados relacional.

Neste contexto está inserido o termo Big Data, o qual se refere a um enorme conjunto de dados que, devido à sua complexidade e volume, apresenta desafios para abordagens de armazenamento e exploração usualmente apoiadas por Sistemas de Gerenciamento de Banco de Dados relacionais.

1.2 COMPLEXIDADE DA INFORMAÇÃO

Os problemas mais frequentemente observados com Big Data dizem respeito à heterogeneidade de dados. Segundo She (2009), esta heterogeneidade entre SGBDs pode ser do tipo sintática, semântica e estrutural, e, como consequência, promove a interoperabilidade de dados de acordo com a origem deles. Tratando-se da heterogeneidade sintática, isso pode ser facilmente resolvido através de um dicionário de termos léxicos, para tratar as sintaxes diferentes representadas pelas diversas fontes de dados. Já no caso da heterogeneidade semântica, em que há uma diferença entre o significado e a interpretação do uso dos dados, podemos ter que considerar os diversos conceitos, e, dessa forma, recomenda-se o uso de ontologias. E em relação ao ambiente de heterogeneidade estrutural, encontramos o dado em estruturas com esquemas distintos, ou seja, ele pode estar armazenado em formato estruturado e armazenado em um banco definido anteriormente, ou então em formato não estruturado e distribuído em fontes, tais como arquivos de texto, vídeos, áudios, dados coletados de sensores, dados esparsos (cada chave com um valor diferente), entre outros. Essa complexidade é um desafio nas soluções que visam atribuir valor (significado relevante e mensurável) a partir de dados brutos capturados de diferentes fontes.

As decisões tomadas para selecionar quais dados terão valor na grande massa, a variedade de fontes disponíveis, e o objetivo que se deseja na análise, também são fatores de suma importância. O valor agregado dos dados selecionados aumenta consideravelmente quando são associados a outros conjuntos de dados, fazendo com que a interoperabilidade e a integração entre eles sejam fatores imprescindíveis para geração de valor. Os seres humanos absorvem informação de forma heterogênea, ou seja, a informação é separada em suas diversas naturezas. A nossa percepção do mundo real é filtrada pela nossa mente, cada pessoa da sua forma, conforme os seus conceitos adquiridos ao longo da vida. A linguagem natural humana proporciona muitas nuances e interpretações diversas. Por isso, os algoritmos de

processamento de linguagem natural, ou de aprendizado de máquina, dificilmente se associam a um significado preciso. Em consequência, os dados devem ser cuidadosamente tratados, contextualizados e enriquecidos semanticamente, sempre que possível, nas etapas anteriores à análise e exploração. Da mesma forma, é necessário adotar técnicas de redução do volume de dados, para que os dados brutos sejam processados de maneira eficiente, e, ao mesmo tempo, sem perder o foco no objetivo da exploração. Além disso, são necessárias técnicas on-line de análise que possam processar dados de streaming em tempo real, já que nem sempre podemos investir em armazenar primeiro e reduzir depois, como num sistema de Data Warehousing convencional, onde o processo de tratamento usualmente envolve essas etapas. Um Data Warehouse é um sistema de computação utilizado para armazenar informações relativas às atividades de uma organização em bancos de dados, de forma consolidada. No processo de coleta de dados na internet, principalmente, muitos dos dados coletados não são úteis (para a análise que se quer), e podem ser filtrados e comprimidos por ordem de magnitude. Um desafio é definir esses filtros de tal forma que eles não descartem nenhuma informação necessária. Por exemplo, suponha que a leitura de um sensor difere substancialmente do restante dos demais sensores. É provável que seja devido ao sensor falho, ou simplesmente que não haja mesmo informação.

A modelagem, organização, recuperação e análise dos dados são outros desafios fundamentais. A modelagem e organização são abordagens essenciais para apoiar, com eficiência, as etapas seguintes de recuperação e análise, tendo, por isso mesmo, papel fundamental no processo de projeto das bases de dados. Já a análise dos dados é um gargalo claro em muitas aplicações, tanto devido à falta de escalabilidade dos algoritmos, quanto à complexidade dos dados que necessitam ser analisados. Finalmente, a apresentação dos resultados e sua interpretação por especialistas de domínio não técnicos se mostra fundamental para extrair conhecimento útil.

1.3 VALOR DOS DADOS

Quantificar o valor de um conjunto de dados volumoso é sempre um desafio. A partir disso, podemos definir economicamente o valor dos dados ao mensurar e avaliar a significância e o impacto financeiro destes dados e as informações que uma empresa possui, de modo que essas informações possam ser valorizadas monetária e contabilmente. Caso não seja possível atribuir uma quantia monetária a um conjunto de dados, então dificilmente vamos conseguir agregar valor a esses para uso comercial. Algumas pessoas acreditam que o

valor do Big Data cresce exponencialmente apenas com o seu volume, mas isso não será verdade se a métrica for pouco detalhada, ou se a qualidade dos dados disponibilizados for baixa. A qualidade dos dados pode definir, inclusive, o valor da informação, reduzindo ou aumentando a probabilidade de interpretação ambígua pelo emissor, ou seja, quanto mais precisa, mais valiosa ela se torna. É um trabalho árduo definir qual o valor agregado dos dados com um critério bem definido e que tenha alguma aplicação no seu negócio. A melhor prática deve ser focada no valor que vai gerar recursos para suportar a tomada de decisões, caracterizando, assim, a função principal da inteligência do negócio e do próprio Big Data. Análises avançadas e aplicações técnicas de investigação sobre os dados analisados também são importantes para se levar em conta.

Uma comparação simples pode ser feita ao tomar o exemplo de uma companhia de petróleo. O valor da empresa é estimado pelos depósitos de petróleo de que ela dispõe (suas reservas), bem como pela sua capacidade de extrair e refinar esse petróleo. Ao considerarmos o conceito contábil, uma empresa será valorizada pelo valor dos dados, as informações que contém, e pela sua capacidade de explorá-los adequadamente. O conjunto de dados armazenados, ainda que não sejam usados, tem seu valor. Assim como uma mercadoria em estoque possui seu valor (valor do estoque) antes mesmo de ser vendida, os dados têm valor mesmo antes de serem tratados por tecnologias de análise de dados. Isto faz com que o valor potencial dos dados possa ser avaliado, ou seja, o valor agregado dos dados representa as suas reservas, de modo resumido.

Um volume pequeno de dados pode ter mais valor do que um grande volume de dados, ou Big Data. Um banco de dados menor, mas bem estruturado, pode ter mais qualidade para a análise do que um maior e com design inadequado. De fato, uma amostra retirada a partir de um Big Data pode ter mais valor econômico e permitir mais análises do que as próprias coleções de petabytes das bases das quais foram extraídas. Considerando que é gerado valor para os resultados que são retornados pela utilização do Big Data, podemos definir alguns conceitos que caracterizem o valor dos dados para fundamentar a escolha da solução utilizando um Big Data.

Assim, com relação ao que considerar em projetos típicos envolvendo Big Data, de forma geral a literatura contempla quatro categorias relacionadas ao valor de um conjunto de dados (HILBERT, 2016):

- Valor baseado no Volume: Quanto mais abrangente a visão do negócio e mais dados históricos podemos extrair dele, maior o conhecimento que se pode extrair dessa informação com todas as circunstâncias consideradas.
- Valor baseado na Velocidade: Velocidade na criação e captura dos dados brutos a taxas muito rápidas. Quanto maior a quantidade de dados a serem tratados rapidamente, e quanto mais questões se possam representar e explorar a partir deles (via queries, reports, dashboards, etc.) em um curto período de tempo, melhores são as decisões tomadas no tempo certo para alcançar os objetivos pretendidos.
- Valor baseado na Variedade: Quanto maior a diversidade dos dados a serem explorados, sejam eles oriundos de mídias sociais, web, sensores, bases de dados públicas ou privadas, etc., maiores as possibilidades das informações e conhecimento extraídos.
- Valor baseado na Veracidade: O resultado obtido ao considerar um grande conjunto de dados será pouco relevante se os dados não forem confiáveis, e, por isso, é necessário ter mecanismos que garantam o máximo possível sua consistência. Quanto mais consolidados, limpos e consistentes os dados disponíveis, melhores e mais acertadas serão as decisões neles baseadas.

A combinação dos quatro tipos de valor dos dados apresentados (volume, velocidade, variedade, veracidade), além de todo e qualquer outro aspecto que caracteriza uma solução de Big Data, se mostrará pouco útil se o resultado não trouxer benefícios significativos e que compensem o investimento feito. Assim, é necessária a constante avaliação destes benefícios, com vistas a buscar melhorias contínuas no tipo de solução adotada.

Uma vantagem proporcionada ao considerar o valor dos dados é tornar mais simples a proposição de projetos que envolvam os conceitos de manuseio de informação. Portanto, conseguimos defender que determinado projeto valorizará o valor de determinada informação, facilitando a geração das estimativas de retorno sobre investimento dele.

1.4 ESTÁGIO ATUAL DA TECNOLOGIA

Diante da crescente demanda por dados, a necessidade de maior desempenho dos repositórios foi ficando imperativa. Para isso, diversas técnicas para tratar apropriadamente as

diversas fontes de dados armazenadas em diferentes origens foram estudadas, de forma a adaptar o poder computacional para processá-los.

Faz-se necessária então a utilização de um Data Warehouse para a adequada estruturação e o bom desempenho de um banco de dados, cuja modelagem deve seguir os preceitos da modelagem multidimensional. A modelagem multidimensional, ou dimensional como às vezes é referenciada, é a técnica de modelagem de banco de dados que complementa as consultas do DW nas mais diferentes perspectivas. A concepção multidimensional permite o uso mais intuitivo da informação para o processamento analítico pelas ferramentas OLAP (*Online Analytical Processing*), sobre dados originados de sistemas de banco de dados relacionais ou não relacionais.

Durante os próximos capítulos, serão utilizados dados do projeto PingER (Ping End-to-end Reporting). Este projeto foi idealizado com o objetivo de coletar e analisar dados de monitoramento do desempenho de links de internet entre pares de nós que representam diversos locais do planeta. Caracterizaremos os dados coletados e analisados pelo PingER, já que a natureza dos dados é a mesma dos experimentos e análises que pretendemos pesquisar. No projeto PingER, foi observado um volume crescente de dados de natureza distinta, considerando que o projeto utiliza fontes de dados diversas para a geração de resultados consistentes. Portanto, estes dados precisam ser explorados sob diferentes perspectivas, e também complementados com outros tipos de informações para melhor contextualização das análises.

Um dos pontos impactantes a se considerar, em relação aos dados do PingER, é o seu volume. Devido ao dispendioso esforço necessário, principalmente durante as etapas de análise de dados gerados constantemente para os usuários, podemos caracterizar as aplicações sobre estes dados (coleta, tratamento e análise de dados) como um problema que requer uma solução através de um Big Data, para lidar com toda a quantidade de dados gerada.

1.5 OBJETIVO GERAL

O objetivo geral deste trabalho é discutir o tratamento de dados com características de Big Data, em especial os dados do projeto PingER, através da utilização de uma solução do tipo NoSQL, baseada em modelo não relacional utilizado pelo sistema de gerenciamento de dados Cassandra, comparativamente à utilização de uma solução baseada em estrutura relacional. Inicialmente, a escolha deste tipo de banco de dados em detrimento de outros tipos de bancos de dados cujas características são similares se fundamenta não apenas no histórico

de utilização por parte de grandes empresas, como também pelo amplo suporte oferecido através da Internet, pela alta escalabilidade oferecida através de seu modelo de dados e pelo poder de replicação de modo assíncrono através de múltiplos nós. Além disso, o Cassandra pode escalar dinamicamente adicionando mais servidores sem a necessidade de refragmentar ou reiniciar o cluster. Como objetivos específicos, colocados na forma de etapas que devem ser cumpridas para atingir este objetivo geral, este trabalho também contempla:

(i) Elaboração da forma de extração de dados obtidos a partir do FTP do SLAC (Stanford National Accelerator Laboratory);
(ii) Criação do modelo multidimensional que represente os dados extraídos;
(iii) Construção de um conjunto de transformações de ETL (vide Anexo A) para carregar os dados conforme o modelo multidimensional criado, e armazenar os dados em um banco de dados relacional e em um outro não relacional;
(iv) Implementação de um mecanismo que garanta a consistência dos dados temporais, de forma que os intervalos de tempo armazenados em diversos registros sempre formem uma linha contínua, e que permita manter o histórico do banco de dados em qualquer momento passado do tempo (entre 1998 e 2015);
(v) Definição de uma interface de consulta sobre as dimensões definidas no modelo de dados multidimensional, e avaliação da eficiência das consultas, considerando o grande volume de dados da base e os possíveis cenários de agregação;
(vi) Realização de análises para as métricas armazenadas no banco de dados relacional e não relacional, de forma a extrair informações e constatar observações que caracterizem o valor da informação;
(vii) Comparação dos resultados das experimentações obtidos através dos dois formatos de banco de dados utilizados, de acordo com os critérios definidos pelo projeto.

1.6 ESTRUTURA DO TRABALHO

Quanto à estrutura deste trabalho, o capítulo 2 apresenta em maior detalhe o cenário do PingER, caracterizando o problema-alvo desta pesquisa. No capítulo 3, é feita uma revisão da literatura do modelo relacional e de suas diferenças em relação às abordagens NoSQL, assim como é contemplada uma definição e caracterização dos bancos de dados não estruturados, focando, essencialmente, nos aspectos práticos fundamentais e essenciais, com o objetivo de evidenciar a escolha do tipo de solução adotada neste trabalho.No capítulo 4,

discute-se esta solução aplicada aos dados do PingER, apresentando as etapas do processo de tratamento dos dados e sua representação neste tipo de abordagem. Finalmente, o capítulo 5 concluirá, sintetizando os principais pontos dos capítulos 2, 3 e 4, e resumirá os benefícios proporcionados por este projeto de pesquisa e desenvolvimento. Também abordaremos as dificuldades enfrentadas ao longo do projeto, bem como as decisões tomadas para encontrarmos outros meios de atender aos objetivos apresentados. Por último, vamos analisar as dificuldades encontradas e as alternativas, para viabilizar pesquisas futuras relacionadas à utilização de outros tipos de banco de dados e de outros tipos de tecnologia, que podem ser empregados, em geral, de forma a complementar os resultados do presente projeto.

2 O PROJETO PINGER E OS DADOS DE MEDIÇÕES DE DESEMPENHO DA INTERNET

Este trabalho se insere no contexto maior do projeto MultiLOD, concebido como um dos projetos exploratórios de implantação do CRBD (Centro de Referência em Big Data), na UFRJ, visando à formação de pessoal qualificado para enfrentar os desafios da área de Big Data no Brasil. Neste projeto, são investigadas diversas técnicas utilizadas na extração, preparação, transformação, armazenamento e análise de grandes volumes de dados estruturados multidimensionais. O principal objetivo é a avaliação de alternativas para tratamento deste tipo de dados, em especial comparando com representações na forma de dados interligados, seguindo o padrão RDF (Resource Description Format). Assim como o mundo relacional se baseia em tabelas e chaves identificadoras, o formato RDF se baseia em declarações tipo sujeito, predicado. Estas declarações são conhecidas como triplas, que, juntas, formam um grafo. O conceito de triplas oferece um significado para a publicação e estruturação de dados na Web, e são extremamente importantes no mundo de Web Semântica, pois grande parte da representação dos dados se dá nesse formato triplificado. Neste sentido, foram escolhidos diferentes tipos de SGBDs, técnicas de distribuição e de organização de dados em ambientes diversos, além de outros mecanismos para modelagem e tratamento de dados.

2.1 PROJETO MultiLOD

O projeto MultiLOD prioriza dados de natureza multidimensional, a exemplo de dados estatísticos e de dados de medidas em geral, por sua importância para áreas estratégicas, como as áreas de governo e científica. As avaliações quantitativas levam em consideração o desempenho de consultas multidimensionais, ao utilizar técnicas de processamento analítico sobre esses dados. Como bases para as experimentações, são utilizados dados referentes às medidas de desempenho de rede do projeto PingER,[1] que foram posteriormente alvo do projeto PingER LOD (SOUZA, 2013), desenvolvido originalmente em parceria com o Stanford Linear Aceleration Center, e considerando o RDF como formato de representação dos dados.

[1] http://www-iepm.slac.stanford.edu/pinger.

No caso do presente projeto, o formato RDF não foi considerado, mas, sim, o tratamento do dado multidimensional, buscando tirar vantagem das funcionalidades oferecidas por um SGBD NoSQL.

O objetivo deste capítulo é descrever o projeto PingER, bem como caracterizar os dados coletados e analisados por aquele projeto, por se tratarem dos dados experimentais de nossa pesquisa.

2.2 PROJETO PingER

O projeto PingER foi desenvolvido pelo IEPM (Internet End-to-End Performance Monitoring), no SLAC, e é liderado pelo departamento de redes e telecomunicações, na divisão de Computação do SLAC.

Para proporcionar uma melhor expectativa do desempenho da rede entre os sites com os quais a SLAC colabora, o projeto PingER, iniciado em janeiro de 1995, monitorou cerca de 100 hosts em todo o mundo, a partir do SLAC. Desde 2000, a ênfase deste projeto é voltada para a medição do desempenho da rede. Já em abril de 2007, havia mais de 35 sites de monitoramento fazendo medição de mais de 600 sites remotos, localizados em mais de 150 países. Essa ordem de grandeza, abrangia, à época, mais de 99% da população mundial conectada à Internet. Esta representatividade garantiu que o projeto alcançasse cerca de 80 nós monitores para mais de 800 nós monitorados pelo mundo, perfazendo cerca de 8.200 pares, em cerca de 160 países, combinando os nós monitores e os monitorados (COTTRELL, 2001).

Segundo Cottrell, líder do projeto PingER, com os dados obtidos a partir das medições, pode-se obter aplicações de cunho técnico, econômico, colaborativo e quantitativo do impacto de eventos, bem como auxiliar na identificação de problemas e de roteamentos.

Diversos casos de estudos do PingER comprovam as aplicações dos dados colhidos pelo projeto, sua importância e utilidade. Como são capazes de medir a qualidade de Internet ao redor do mundo, são também capazes de identificar situações, lugares e eventos de caráter crítico e que necessitam de Internet e acesso à informação, e, com isso, oferecem perspectiva de crescimento tecnológico, social e educacional.

Portanto, o problema-alvo pode ser assim caracterizado: os dados do PingER não são armazenados de uma forma facilmente acessível, não sendo também uma forma padrão.

De acordo com o problema definido, faz-se necessário disponibilizar os dados do PingER de uma maneira eficiente, fácil, padronizada e que seja útil a qualquer pessoa que utilize suas informações, além dos próprios integrantes do PingER.

2.2.1 Geração dos dados do PingER e métricas calculadas

O projeto PingER utiliza o mecanismo ICMP, também conhecido como o mecanismo Ping. Ele permite enviar um pacote de comprimento selecionado pelo usuário para um nó remoto, que ecoa de volta. Hoje em dia, geralmente, esse comando vem pré-instalado em quase todas as plataformas. Então, não é necessária nenhuma instalação nos clientes, sendo, portanto, transparente para os hosts monitorados, de forma independente, e fácil de se implementar. O comando Ping é executado pelo servidor em prioridade alta, sendo, por isso, o melhor método para fornecer uma boa medida do desempenho da rede, se comparado a uma aplicação de usuário. Uma de suas qualidades é a baixa exigência de largura de banda (cerca de 100 bits por segundo para monitoramento do host remoto). No projeto PingER, a cada 30 minutos a partir de um nó de monitoramento (Measurement Point — MP), é emitido um conjunto de 11 pings de 100 bytes para cada par de nós remotos. Os pings são separados por pelo menos um segundo, e o timeout padrão do ping é de 20 segundos. O primeiro ping é jogado fora, pois, segundo Martin Horneffer,[2] o primeiro pacote demora cerca de 20% mais tempo para voltar. O mínimo, o médio e o máximo RTT (Round Trip Time — ver item no Anexo B) para cada conjunto de 10 pings são gravados. Este procedimento é repetido durante dez pings de 1.000 bytes. O uso de dois tamanhos de pacotes de ping permite fazer estimativas de taxas de dados de ping, e também detectar comportamentos que diferem entre pequenos e grandes pacotes, para estabelecer os limites de tamanho dos pacotes. Em geral, o RTT é proporcional a l (onde l é o comprimento do pacote), até o tamanho máximo do datagrama (geralmente 1.472 bytes, incluindo o 8 ICMP echo bytes).

O conjunto de hosts remotos que receberão os pings é fornecido por um arquivo chamado pinger.xml. Este arquivo é composto de duas partes: hosts Beacon, que são automaticamente inicializados, diariamente, do SLAC, e monitorados por todos os MPs e outros hosts que são de particular interesse para o administrador do MP. Os hosts Beacon (um conjunto de hosts confiáveis de monitoramento)[3] — e outros hosts particulares, monitorados pelo MP do SLAC — são mantidos em um banco de dados Oracle, contendo o respectivo nome, endereço IP, site, apelido, localização, contato, etc. A lista Beacon (lista dos hosts

[2] http://www.advanced.org/IPPM/archive.2/0246.html.
[3] http://www-iepm.slac.stanford.edu/pinger/beacon.html.

particulares do SLAC) e uma cópia do banco de dados, em um formato para simplificar o acesso Perl para scripts de análise, são gerados automaticamente a partir do banco de dados em uma base diária.

2.2.2 Acesso aos dados

A arquitetura do monitoramento possui os seguintes componentes:

- Sites Remotos Monitorados: Estes, simplesmente fornecem um host remoto passivo com os requisitos apropriados.
- Sites de Monitoramento: As ferramentas de monitoramento do PingER precisam ser instaladas e configuradas em um host em cada um desses sites.

  Também os dados de ping recebidos precisam ser disponibilizados nos arquivos dos hosts através do HyperText Transport Protocol (HTTP), ou seja, há um servidor Web que fornece os dados sob demanda através da Internet. Há também ferramentas PingER para permitir que um site de monitoramento seja capaz de fornecer análises e relatórios de curto prazo sobre os dados armazenados em seu cache local.
- Sites de Análise e Arquivamento: Os sites de análise e arquivamento (um para cada projeto PingER) podem ficar alocados num site apenas, ou podem ser hospedados em sites diferentes. O projeto PingER possui dois desses sites: um em NUST, em Islamabad, Paquistão; o outro no SLAC. Ambos são sites de arquivamento e análise. Eles se complementam, providenciando redundância aos dados armazenados.

Através do FTP do SLAC, pode-se baixar o volume total de arquivos do SLAC que estão numa série de arquivos compactados, separados pela métrica e pelo tamanho de pacote. Cada arquivo compactado inclui todos os arquivos relativos àquela métrica, em todas as agregações temporais (Tick-Type), desde o início do projeto (1998) até o ano corrente.

O SLAC sugere, para aperfeiçoar o download dos dados, usar um diretório temporário, como forma intermediária de armazenamento, com tamanho aproximado de 100 GB. Deve-se baixar uma métrica por vez, definindo o tamanho do pacote considerado, o critério (“by-node” ou “by-site”).

É estimado, para cada métrica, cerca de 6.500 arquivos. A quantidade de arquivos tende a aumentar conforme o período de tempo aumenta também. O tempo de download via

FTP e a devida compressão dependem exclusivamente das características da conexão de rede e do hardware da máquina. Este arquivo compactado ocupa aproximadamente 1,5 GB em disco para cada uma das métricas, e pode ser acessado através do site do laboratório do SLAC.[4]

Devido à falta de padronização encontrada em um sistema de banco de dados, tanto a recuperação de dados daqueles arquivos, quanto sua gerência, não são muito efetivas, limitando ou dificultando por demais a manutenção e a realização de análises mais elaboradas sobre os dados, se optarmos por utilizar esse sistema de armazenamento de dados. Assim, como primeiro passo, devemos obter localmente os arquivos contendo toda a informação necessária. Sabemos que a PingTable executa HTTP GETs no servidor para poder recuperar o arquivo ".txt" referente a determinado cruzamento de parâmetros, para então, mostrar os dados do arquivo .txt em um documento HTML.

Como a abordagem de utilizar a PingTable para acessar os dados é extremamente lenta para pegar um grande volume de dados, foram disponibilizados os dados em arquivos compactados no FTP do SLAC (ftp://ftp.slac.stanford.edu/users/cottrell/), para que possam ser extraídos e manipulados localmente, em um sistema de banco de dados. Adicionalmente, para obter uma descrição completa de como acessá-los e extrair as suas informações, pode se utilizar do link disponível no site do laboratório SLAC.[4]

Pode-se perceber que os dados armazenados estão compactados, o que reduz significativamente o quanto eles ocupam em disco fisicamente. Considerando os dados descompactados de medidas de rede desde o início das medições, em 01/01/1998, até a data corrente (10/05/2015), no grão hora, de todos os nós fonte para todos os nós destino, para todas as 16 métricas do PingER, temos o total de 68,42 GB em disco. Para algumas combinações de dia, mês e ano, pode ocorrer de nenhum arquivo ser gerado; então, temos um valor nulo para o campo correspondente ao tamanho do arquivo. Para outras, o arquivo gerado tem apenas o cabeçalho com as horas daquele dia, sem nenhuma medida.

2.3 PingER LOD

O projeto PingER LOD teve como objetivo aplicar os conceitos de Web Semântica em um cenário real (do PingER) e abrir os dados no formato de Dados Abertos Interligados (Linked Open Data — LOD), cujo nome dado ao mesmo foi PingER Linked Open Data. Isto

[4] https://confluence.slac.stanford.edu/display/IEPM/Archiving+PingER+data+by+tar+for+retrieval+by+anonymous+ftp.

foi feito através da publicação de dados do PingER em formato LOD, evidenciando, ao final dos experimentos, diversas vantagens. Portanto, elementos conceituais e tecnológicos da Web Semântica foram reunidos para desenvolver um projeto que publicasse dados neste formato. Uma das características principais do LOD é ter vários bancos de dados separados, mas que podem ser interoperados e "enxergados" como um só, por conta das ligações entre os diferentes arquivos ou bancos (SOUZA, 2013). Adicionalmente, as tecnologias de Web Semântica possibilitam que as pessoas criem bancos de dados na web, construam vocabulários e escrevam regras para lidar com os dados (WORLD WIDE WEB CONSORTIUM, 2005). Como solução, foi utilizada uma base de dados em RDF com SPARQL Endpoints (WORLD WIDE WEB CONSORTIUM, 2008), que possibilitou o cruzamento de dados de natureza e fontes diferentes. Isto ocorreu através da realização de Mashups — técnica para cruzamento de dados, de modo a gerar aplicações e relatórios úteis que dão suporte à tomada de decisão. O uso de consultas SPARQL complexas nesta etapa capturou precisamente o que se deseja, e facilitou bastante o trabalho e o desempenho das consultas.

Finalmente, para a modelagem da informação do escopo selecionado, foram usadas as técnicas de ontologias atuais. Resumidamente, ontologia é a maneira de modelar conceitos no mundo da Web Semântica. Os conceitos de modelos de dados em RDF e triplas complementam as técnicas dessa modelagem.

Como o projeto PingER envolve uma quantidade enorme de dados, tornou-se necessário definir que tipos de dados seriam mais importantes e relevantes no contexto do PingER LOD. Sem isso, seria impraticável o seu desenvolvimento, devido ao prazo estabelecido e à capacidade de processamento necessária para tal. Considerando estes fatos, foi estabelecido o escopo, que se divide entre a quantidade de nós da rede, nível de detalhe geográfico, nível de detalhe temporal, tipo de métrica e tamanho do pacote.

Além dos dados do PingER que foram selecionados, o projeto utilizou diversos bancos de dados publicados em formatos de triplas, que auxiliaram na tomada de decisão, através da realização de diversas consultas, experimentos e aplicações úteis. Dentre estes bancos, foram utilizados alguns de extrema importância durante o projeto, tais como DBPedia,[5] Freebase,[6] Geonames,[7] The World Bank[8] e FactForge.[9]

[5] http://dbpedia.org/sparql.
[6] https://www.freebase.com.
[7] http://www.geonames.org/ontology/documentation.html.
[8] http://worldbank.270a.info/sparql.
[9] http://factforge.net/sparql.

2.4 CONSIDERAÇÕES FINAIS

As tecnologias de Web Semântica adotadas pelo PingER LOD evidenciaram diversos benefícios durante o seu desenvolvimento. Além disso, o projeto foi de expressiva utilidade ao SLAC, pois, por ter sido o primeiro trabalho essencialmente relacionado à área da Web Semântica naquele laboratório, gerou interesse de seus membros no assunto. Sobretudo, ainda foi capaz de viabilizar diversas aplicações aos dados do PingER, algumas delas bastante inovadoras; e, especialmente, os abriu para a comunidade, através de uma forma padronizada (padrões LOD).

No entanto, ficou claro o problema do grande aumento de volume de dados ao se utilizar uma representação de dados em RDF. Com o projeto PingER LOD, ficou evidente a necessidade de se buscar alternativas em termos de representações mais eficientes para os dados, uma vez que o desempenho e o espaço requeridos para o tratamento dos dados no formato RDF constituem um desafio, especialmente se considerarmos todas as métricas e escopo de dados tratados no PingER.

No capítulo 3, serão exploradas diversas propriedades e conceitos da classe de bancos de dados não relacionais. A grande motivação para o novo paradigma de armazenamento de dados é resolver o problema de escalabilidade dos bancos tradicionais, já que pode ser muito caro ou complexo escalar um banco em SQL. Portanto, este estudo deverá fornecer as bases para facilitar o entendimento dos potenciais benefícios para a manipulação de um grande volume de dados.

3 SISTEMAS DE GERENCIAMENTO DE BANCOS DE DADOS RELACIONAIS E NÃO RELACIONAIS

Neste capítulo, serão explorados alguns conceitos sobre o modelo relacional, e, a partir de suas limitações, vamos definir a alternativa que se propõe a corrigir ou evitar tais dificuldades. Além disso, aprofundaremos as principais características deste tipo de banco de dados, com o objetivo de encontrar as vantagens e desvantagens de cada exemplo de banco, de forma a escolher o mais adequado para o desenvolvimento dos experimentos futuros.

3.1 SISTEMAS DE GERENCIAMENTO BASEADOS NO MODELO RELACIONAL

Nos dias de hoje, o Modelo Relacional ainda é o mais utilizado pela maioria dos sistemas de gerenciamento de bancos de dados. Esse modelo é o sucessor do modelo em rede e do modelo hierárquico, e ele se baseia no conceito de que todos os dados estão armazenados em tabelas, e que essas possuem relações definidas entre si. A característica fundamental deste modelo são as restrições de integridade, que são utilizadas para manter a integridade entre os dados, evitando redundâncias. As bases para essas restrições de integridade são as chamadas Chaves. A chave primária tem o objetivo de assegurar a identificação única das tuplas de uma tabela, associando uma informação a essa única chave. Outro conceito, Chave Estrangeira, torna os valores de um atributo dependentes dos valores de outro atributo de outra tabela, geralmente sua chave primária. Essas relações de integridade referencial garantem que as informações numa tabela correspondam às informações em outra tabela (CODD, 1970).

De acordo com os tipos de relações inerentes entre as tabelas, como, por exemplo, a relação um-para-um, um-para-muitos e muitos-para-muitos, podemos estabelecer as bases para a implementação do modelo relacional.

A linguagem padrão atual para manipular os dados que foram definidos através do modelo relacional é o SQL. Foi desenvolvida pela IBM para trabalhar com bancos de dados relacionais. Inspirada na álgebra relacional, o SQL é uma linguagem declarativa para banco de dados. De uso simples e intuitivo, as expressões SQL fizeram com que a linguagem se tornasse um padrão de fato e ajudou a consolidar o modelo relacional.

Os bancos de dados construídos com tecnologia relacional têm as propriedades ACID. Um banco que tem essas propriedades significa que possui Atomicidade, Consistência, Isolamento e Durabilidade, ou seja, permite o compartilhamento seguro de dados, oferecendo

otimização de consultas, recuperação de falhas, validação, controle de concorrência e verificação de integridade dos dados. Este tipo de banco funciona relativamente bem com tipos simples de dados, que envolvem poucas associações entre relações; no entanto, apresentam algumas deficiências quando aplicações de banco de dados mais complexas precisam ser projetadas e implementadas (ELMASRI e NAVATHE, 2005).

A evolução dos requisitos das aplicações estimuladas pela evolução dos sistemas, fez surgir características que acrescentaram maior complexidade às existentes. A grande quantidade de aplicações, recursos e soluções, que foram desenvolvidos para os sistemas computacionais nos últimos anos, levou a um crescimento acelerado do volume de dados utilizado. Sendo assim, toda a garantia fornecida pelos bancos relacionais foi impactada pela crescente demanda que as empresas necessitam armazenar atualmente. Surgiram diversas necessidades de representação para tipos de dados longos e complexos, de expansão dos modelos de dados e linguagens de consulta, para acomodar novos tipos de dados e dar suporte a transações longas, contribuindo, também, para o aumento no grau de inter-relacionamentos entre os dados. Além disso, o armazenamento de imagens, dados de forma não estruturada e em sistemas, linguagens e formatos diferentes, e, em alguns casos, incompatíveis entre si, também impactam de forma significativa no desempenho dos bancos relacionais (CARDOSO, 2003).

As soluções para SGBDs relacionais com gargalo de desempenho costumam ser complexas e caras, além de possuírem um alto custo-benefício. Por exemplo, caso haja um crescimento do número de clientes cadastrados na aplicação, observamos uma queda na eficiência, e, com isso, teremos que investir na melhoria do desempenho do servidor, ou, até mesmo, adquirir um novo servidor. Esse modelo de banco trabalha melhor com a escalabilidade vertical, que consiste em adicionar mais poder de processamento, memória ou disco em uma máquina. Porém, caso o crescimento do tráfego de dados seja esporádico, o problema vai se concentrar no acesso à base de dados. Nesse caso, a solução mais viável seria escalar o banco de dados para suportar esse aumento fazendo sua distribuição em várias máquinas, mas tal realização se torna complexa.

Deste modo, e conforme mencionamos previamente, o presente trabalho pretende explorar o problema com o foco em uma solução com uso de banco de dados não relacional conhecido como NoSQL.

3.2 SISTEMAS DE GERENCIAMENTO DE BANCO DE DADOS NÃO RELACIONAL — NoSQL

Devido à problemática explorada no item anterior, diversas alternativas ao uso do Modelo Relacional vêm sendo consideradas nos últimos anos, de forma a ter representações de bases de dados com maior flexibilidade. O termo NoSQL abrange uma grande variedade de diferentes tecnologias de banco de dados, que foram desenvolvidas em resposta a um aumento constante do volume e diversidade de dados armazenados, sobre os mais diversos assuntos, e em frequências de acesso cada vez mais críticas para os bancos, com uma crescente necessidade de desempenho de processamento.

Projetistas de banco de dados de algumas organizações sugeriram diversas estratégias de armazenamento, de acordo com os requisitos específicos de suas aplicações. Uma forma mais eficiente de abordar esse problema é fazer com que esses bancos se tornem estruturados, para se trabalhar com escalabilidade horizontal, que é a capacidade de adicionar novas máquinas, para, de forma distribuída, aumentar os recursos de processamento, memória e disco. Nessas soluções, algumas das regras e estruturas do Modelo Relacional tiveram de ser suprimidas. Uma das mais importantes, é que nem sempre se tem a preocupação de fornecer garantias ACID, podendo por algum tempo se conviver com alguma inconsistência entre bases distribuídas, para ganho de desempenho.

O termo NoSQL não significa que ele tenha sido criado para substituir a utilização das técnicas existentes do SQL. Na verdade, o surgimento dessa nova técnica não invalida as aplicações apropriadas aos bancos relacionais. Independente da escolha adotada, cada uma delas tem a sua devida importância, considerando o melhor desempenho nas aplicações.

Em 1998, criou-se, então, o conceito de NoSQL, com um banco ainda baseado na arquitetura relacional, mas que não oferecia interface SQL. Anos mais tarde, o termo NoSQL volta com uma importância ainda maior, devido ao surgimento de um número crescente de sistemas de banco de dados não relacionais.

Neste cenário, surge uma nova geração de banco de dados não relacionais, como uma maneira de lidar com o crescente volume e diversidade de dados. Tecnologias NoSQL atendem aos requisitos do ambiente de computação distribuída em larga escala, o que permite escalabilidade, alta disponibilidade, alto desempenho e confiabilidade. Diversos tipos de sistemas NoSQL, especialmente os pioneiros, são baseados no conceito BASE (Basic Availability, Soft-State, Eventual Consistency). As propriedades BASE são consideradas opostas às ACID. Enquanto as propriedades ACID são pessimistas e forçam a consistência no

final de cada transação, as propriedades BASE aceitam inconsistências do banco de dados de forma eventual. Embora, na realidade, estas propriedades pareçam improváveis de serem aceitas, elas são bastante flexíveis, e permitem níveis de escalabilidade que não podem ser obtidos quando se respeita as propriedades ACID tradicionais dos SGBDs relacionais (PRITCHETT, 2008).

A abordagem BASE é implementada de forma diferente, de acordo com o tipo de banco de dados utilizado. Alguns tipos, por exemplo, são denominados "eventualmente consistentes", pois possuem controle de concorrência multiversão (MCC), que é um método normalmente utilizado por SGBDs para fornecer acesso simultâneo ao banco de dados. Isso faz com que todos os processos aguardem, até que a atualização dos outros processos termine, para iniciarem, evitando a ocorrência de um dado inconsistente. Resumindo, nos sistemas BASE existem múltiplas cópias de todos os dados em um estado que está constantemente alterando, de forma a garantir consistência eventual.

Para atingir uma solução capaz de se adaptar ao processamento analítico em larga escala, foi necessário descartar algumas das propriedades que caracterizam as bases de dados relacionais. Surgiu uma nova forma de acessar os dados através de um conjunto de operações simples (get, put, scan e delete), no lugar da linguagem SQL. Outras operações atômicas de incrementos podem ser complementadas por alguns sistemas. A desnormalização dos dados é realizada pela duplicação de partes dos mesmos em múltiplas tabelas, de forma a evitar a necessidade de agregações e operações join, além de tornar as leituras mais rápidas e eficientes. Esta é uma das características principais dos sistemas NoSQL, que implica na modificação do esquema de dados em relação ao modelo relacional. Dessa forma, com o intuito de promover uma maior flexibilidade na forma como os dados são armazenados, o NoSQL não possui uma estrutura dos dados bem definida. Com isso, é permitido que parte de sua estrutura, como, por exemplo, as colunas, seja ajustável conforme é feita a manipulação de dados.

A grande vantagem dos bancos NoSQL é resolver o problema de escalabilidade, que não é bem resolvido nos bancos relacionais devido ao seu esquema complexo. A formação de grids de servidores distribuídos, pode ser feita com servidores baratos, e de forma bem mais simples. No entanto, é necessário analisar todas as suas características, para obter uma visão geral de seus prós e contras, quando comparado ao modelo relacional.

Uma das primeiras utilizações importantes de bancos não relacionais, foi o Google BigTable, um banco de dados de alto desempenho, criado para trazer maior escalabilidade e disponibilidade, fazendo com que a flexibilidade aumentasse consideravelmente; e o Dynamo,

desenvolvido pela Amazon, em 2007, e utilizado para os serviços web da Amazon. Logo a seguir, surgiu o Cassandra, para lidar com grandes volumes de dados. Entre suas características, podemos citar a alta disponibilidade e distribuição de dados, características essas típicas dos bancos NoSQL de sucesso. Na mesma época, surgiu também o CouchDB, um banco de dados livre e orientado a documentos, armazenando os dados sem que houvesse um esquema pré-definido, e utilizando o conceito de MapReduce, para suportar computação distribuída em larga escala.

A otimização para comportar servidores distribuídos (escalabilidade), foi feita com características como: não possuir esquemas de tabela fixa e trabalhar com pares chave-valor. Mas, por outro lado, traz uma desvantagem, pela ausência dos conceitos ACID e da sintaxe SQL nativa, com as úteis operações de junção SQL. O não cumprimento dos conceitos ACID é causado pela possível inconsistência das informações que os bancos NoSQL realizam de modo eventual no modelo. Isso só garante que, se nenhuma atualização for realizada sobre um determinado item de dados, todos os acessos a este item devolverão o último valor atualizado.

Algumas das principais diferenças entre a modelagem de dados em bancos NoSQL e em relacionais, estão ligadas à forma na qual suas respectivas tecnologias funcionam. Enquanto os bancos relacionais são centralizados, nos bancos não relacionais o conjunto de dados costuma ser distribuído. Portanto, os dados podem ser armazenados não apenas em um data center local, como também em múltiplos data centers. Isto é feito através de um mecanismo chamado replicação, que é uma característica inerente dos sistemas NoSQL.

3.3 MODELOS DE DADOS NoSQL

Pensando nas principais questões apresentadas no item anterior, bem como nas necessidades dos usuários, surgiram diferentes alternativas para suporte aos bancos de dados não relacionais, que modelam os seus dados de diferentes maneiras. Atualmente, existem quatro principais categorias de aplicações NoSQL que se destacam. São elas:

a) Armazenamento Chave-Valor

O modelo chave-valor é o mais simples e fácil de se implementar. Este, de modo geral, é o modelo com melhor desempenho, pois possui um algoritmo bastante eficiente. Usa-se uma tabela Hash, na qual há uma chave única e um indicador, de um dado, valor ou qualquer informação, que, inclusive, pode ser multivalorado. Alguns bancos de Chave/Valor alocam os dados em memória, enquanto outros continuam mantendo estes dados em um disco. Estas

estruturas de armazenamento favorecem a alta escalabilidade, em detrimento de uma perda de consistência. Entretanto, ele é ineficiente quando se está interessado apenas em consultar ou atualizar parte de um valor (STRAUCH, 2012). As suas aplicações típicas envolvem principalmente cache de conteúdo, que aumentam o foco em escalar imensas quantidades de dados. Os bancos de dados usualmente conhecidos e que utilizam este modelo são o Tokyo Cabinet, Redis e Voldemort.

Exemplo de dados armazenados através de um BD Chave/Valor:

12345 / iPod Nano

12346 / MacBook Air

b) Armazenamento orientado a Colunas

Este modelo foi idealizado com o intuito de armazenar e processar enormes quantidades de dados que estão distribuídos em muitos servidores. Este tipo de armazenamento permite que somente dados reais sejam armazenados, evitando o consumo de espaços com valores nulos. Ele aloca os dados em colunas (ao invés das linhas, no Modelo Relacional), e possui um identificador conhecido como Row Key, ou, apenas, Key (Chave). Esses dados são armazenados de forma ordenada e sequencial em uma mesma coluna. As chaves, nesse caso, apontam para colunas múltiplas, e as colunas são organizadas por famílias de colunas (similar a uma tabela). E o agrupamento de famílias de colunas é chamado de Keyspace, que pode ser comparado a um banco de dados do modelo relacional. Entre os tipos de bancos de dados colunares mais utilizados estão o Accumulo[10], Cassandra[11] e HBase[12].

Exemplo de dados armazenados através de um BD orientado a Colunas:

1	2	3
Mariana	José	Pedro
Ribeiro	Carlos	Oliveira
23	25	27

c) Banco de Dados orientado a Documentos

Modelo similar ao armazenamento chave-valor, mas num nível acima, consistindo de documentos versionados, que são coleções de outras coleções de chave-valor, permitindo valores aninhados, associados a cada chave. Foram inspirados no Lotus Notes. Esses

[10] https://accumulo.apache.org/
[11] http://cassandra.apache.org/
[12] https://hbase.apache.org/

documentos são semi-estruturados, e armazenados em formatos que suportam consultas mais eficientes. Ou seja, documentos são estruturas de dados que estão em formato XML (Extensible Markup Language) ou JSON, e podem ser localizados pelo seu identificador único ou por qualquer registro de identificação dentro do arquivo. Além disso, a estrutura de documentos permite a existência de documentos filhos. São principalmente recomendados para utilizações em aplicações Web e como exemplo de bancos de dados orientados a documentos é importante citar o CouchDB, MongoDb e o Riak.

Exemplo de dados armazenados através de um BD orientado a Documentos:

```
{
primeiro_nome: “João”
endereco: “rua a, 21”
artigos: [
  {titulo: “Encapsulamento de dados”, paginas:“10”}
  {titulo: “Banco de dados NoSQL”, paginas:“15”}
]
}
```

d) Banco de Dados orientado a Grafos

Como os modelos anteriores, este também é utilizado para ser escalado em servidores distribuídos. Entretanto, ele está diretamente relacionado a um modelo de dados estabelecido, o modelo de grafos.

O modelo de grafos é mais eficiente que os outros quando desejamos obter informações sobre a interconectividade ou a topologia dos dados, mais importantes do que os dados propriamente ditos. Desse modo, o modelo possui, como componentes básicos, os nós — ou vértices do grafo —, os relacionamentos (arestas) e as propriedades, que, neste caso, representam os atributos dos nós e relacionamentos. O banco de dados pode ser visto, então, como um multigrafo rotulado e direcionado, onde cada par de nós pode estar ligado por uma ou mais de arestas (LÓSCIO, 2011).

No modelo relacional, as consultas que representam uma quantidade enorme de relacionamentos poderiam ser muito complexas, devido à necessidade de múltiplas junções, como, por exemplo, se desejamos saber o número de cidades que foram visitadas por milhares de pessoas em um grande intervalo de tempo. Esta consulta poderia acarretar uma diminuição no desempenho da aplicação. Porém, por meio dos relacionamentos presentes nos grafos, estas consultas tornam-se mais simples e eficientes.

Alguns Sistema de Gerenciamento de Banco de Dados que utilizam esse padrão são os seguintes: Neo4J,[13] Infinite Graph,[14] InfoGrid,[15] HyperGraphDB,[16] etc.

3.4 EXEMPLOS DE TECNOLOGIAS NoSQL

3.4.1 HBase

Originalmente criado para uso com o Apache Hadoop, o Apache HBase é um framework de software com suporte a aplicações distribuídas que fazem uso intensivo de dados sob licença livre. É um banco de dados orientado a colunas constituído para fornecer pedidos com baixa latência sob Hadoop HDFS, uma plataforma de software em Java de computação distribuída voltada para clusters e processamento de grandes massas de dados. Isto fornece consistência ao HBase, o que não ocorre com os bancos de dados orientados a coluna que fornecem consistências eventuais. Sendo quase um clone do BigTable do Google, foi originalmente criado para ser usado com o Hadoop no Hadoop File System, pois foi um subprojeto do projeto Hadoop do Apache. Pode ser usado como uma fonte para trabalhos do MapReduce, pois possui as capacidades de banco de dados do Hadoop.

O HBase pode ser executado de forma autônoma, pseudodistribuída (diversas instâncias do HBase executando em um mesmo host) ou no modo completamente distribuído. Idealmente, ele suporta dados não estruturados e parcialmente estruturados. Para fazer isso, os dados são organizados em famílias de colunas e um registro individual, chamado de “célula” no HBase, é endereçado com uma combinação de linha, família de coluna, identificador da célula e timestamp. Em oposição aos sistemas de gerenciamento de bancos de dados relacionais, nele deve ser definida a tabela com antecedência. Com o HBase, é possível simplesmente nomear uma família de colunas e, então, permitir que a célula se qualifique para ser determinada a um certo tempo de execução. Isso permite que este tipo de banco de dados seja bastante flexível, suportando uma abordagem ágil de desenvolvimento.

[13] http://neo4j.com/.
[14] http://www.objectivity.com/products/infinitegraph/.
[15] http://infogrid.org/trac/.
[16] http://hypergraphdb.org/.

3.4.2 Apache Cassandra

O Apache Cassandra é um sistema de armazenamento de dados NoSQL de arquitetura distribuída, usa o modelo de família de colunas, composta por keyspaces, supercoluna e coluna, e tem armazenamento híbrido (configurável), conforme explicado mais abaixo.

Seu projeto foi iniciado pela equipe do Facebook, e, atualmente, é mantido pela Apache. Foi desenvolvido na plataforma Java. As principais implementações do Cassandra ocorreram na Apple, com mais de 75.000 nós armazenando dados de mais de 10 PB; na Netflix (2.500 nós, 420 TB, mais de 1 trilhão de acessos por dia); e no eBay, com mais de 100 nós e 250 TB de dados. Outros usuários importantes são o Twitter, o Facebook e o Digg. Possui API para as linguagens Ruby, Perl, Scala, Python, PHP e Java.

Os dados no Cassandra são indexados por uma chave do tipo String, que faz referência a uma linha. Os dados estão armazenados nessas linhas, que são divididas em colunas e famílias de colunas. Cada coluna, no Cassandra, tem um nome para identifica-la, um valor e um timestamp. Quando um dado é inserido, a aplicação fornece tanto o valor quanto o timestamp, que serve para identificar quando a inserção ocorreu.

No Cassandra, definimos duas classes de colunas: as colunas normais e as super colunas. Uma coluna normal pode ser criada em tempo de execução pela aplicação, sem que esta precise declarar ou alterar nada, bastando informar um valor para a nova coluna e para uma determinada chave. As super colunas possuem outras colunas como valores, ao invés de objetos.

Para agrupar colunas, o Cassandra tem o conceito de Família de Colunas semelhante ao conceito das tabelas de um banco de dados relacional. Ao contrário das colunas, as famílias de colunas não são dinâmicas, e precisam ser declaradas anteriormente em um arquivo de configuração. Tal como as colunas, as Famílias de Colunas também possuem as Super Famílias de Colunas, que são as Famílias de Colunas formadas somente de colunas do tipo Super Colunas.

Um dos principais objetivos do Cassandra é alcançado pelas características da sua arquitetura, que não contém pontos de falha, resultando em uma disponibilidade contínua para aplicações críticas de negócios. Também oferece suporte a transações através da manutenção das propriedades de atomicidade, isolamento e durabilidade (ACID), devido ao uso de log de registro que captura todas as gravações, garantindo a consistência dos dados em caso de falhas de hardware. Além disso, possui um rápido desempenho em escala linear por conseguir duplicar o seu rendimento com dois nós, quadruplica com quatro, e assim por diante.

A escalabilidade elástica que o Cassandra utiliza permite o aumento da capacidade de armazenamento para mais clientes e mais dados, sempre que necessário. Isto é derivado de uma propriedade da escalabilidade horizontal, que escala de cima para baixo. Assim, basta disponibilizar a máquina, de forma que o cluster a identifique, sem que seja necessário fazer uma cópia de todos os dados para essa nova máquina, pois o próprio cluster se propõe a fazer isso, e mantém todas as máquinas sincronizadas. Não é preciso, então, interromper o processamento, reiniciar ou realizar alguma manutenção no banco de dados. Desse modo, quanto maior a quantidade de máquinas adicionadas, melhor.

De acordo com a maturidade do Cassandra pode adaptar o nível de consistência aplicado, para garantir a consistência do banco. Por isso, temos algumas opções configuráveis, para que em cada transação a consistência seja determinada, como, por exemplo, no caso de uma leitura, onde é possível dizer quantos nós deve conter o registro que está sendo lido para garantir a sua consistência. Isto significa que, mesmo possuindo alguns nós fora de sincronização, o Cassandra retorna a transação de um modo correto. Para garantir a alta disponibilidade do sistema é recomendado utilizar uma quantidade igual ou superior a três nós. Por exemplo, se considerarmos apenas dois nós e um fator de replicação de dados igual a dois então não será possível realizar operações de escrita caso um dos nós fique inativo. Ao utilizar três nós é possível realizar leituras e escritas ainda que um dos nós fique inativo.

Seu armazenamento de dados é flexível, e acomoda dinamicamente alterações em suas estruturas de dados, conforme os dados se alterem, e também acomoda toda a gama de formatos de dados (estruturados, semiestruturados e não estruturados).

Uma de suas características está relacionada com a distribuição de dados, em que o Cassandra fornece a simples e máxima flexibilidade para distribuição dos dados por meio da replicação de dados através de diversos data centers, em nuvem, e até em nuvem mista. Realiza tarefas de leitura e escrita para todos os nós, sincronizando automaticamente através de um cluster. Assim, caso um dos nós saia do cluster, os outros nós podem continuar realizando as solicitações de leitura e escrita. E para simplificar ainda mais a administração e a configuração do Cassandra, todos os nós do sistema podem ser atribuídos a um único cluster.

3.5 BANCOS DE DADOS NoSQL VERSUS BANCOS DE DADOS RELACIONAIS

Geralmente, o melhor caso para se usar a tecnologia NoSQL é quando o modelo de dados é simples e a flexibilidade é mais importante que o controle rígido sobre as estruturas de dados. O NoSQL deve ser utilizado sobretudo onde alto desempenho seja obrigatório, e quando não é estritamente necessário que exista uma forte consistência de dados. Por fim, ele é recomendado também quando é fácil mapear valores complexos para chaves conhecidas.
Bancos de dados NoSQL se mostram opções interessantes para projetos que necessitam de alta escalabilidade. A grande variedade de Sistemas de Gerenciamento de Banco de Dados NoSQL disponíveis, bem como a diversidade que cada banco proporciona, tornam viável sua utilização em inúmeras aplicações. Outro caso bastante interessante para provar a sua aplicação é a flexibilidade com que é feita a distribuição de dados em múltiplos servidores sem que haja a necessidade de requisitar assistência da aplicação. Sendo assim, servidores podem ser adicionados ou removidos da camada de dados e ao mesmo tempo a disponibilidade de dados é mantida sem que seja perceptível uma queda durante a execução das aplicações.

Contudo, esta categoria de banco de dados não é uma solução definitiva para substituir os bancos de dados relacionais. Qualquer sistema que necessite de uma boa consistência de dados, aliada a um isolamento tal que não ocorram falhas, faz com que a escolha do desenvolvimento em um banco de dados NoSQL deixe de ser uma boa opção. Um exemplo disto seria um sistema de banco, onde uma transação de conversão cambial executada diariamente dependesse do isolamento do banco e da consistência de dados, caso contrário poderia existir um risco financeiro crescente. Entretanto, sistemas que não possuam regras de negócio rígidas, podem utilizar os bancos de dados NoSQL, para aumentar a escalabilidade, latência e tempo total de resposta de suas requisições, e, em troca, ainda oferecer maior flexibilidade aos usuários. Como exemplo de utilização de um banco de dados NoSQL, podemos citar uma rede social, onde a inconsistência (eventual) de um dado (um comentário ou outra ação que não produza perdas financeiras) não tem grande impacto total para o usuário.

Sistemas de Gerenciamento de Banco de Dados relacionais, por outro lado, não foram projetados para lidar com alguns tipos de desafios de escalabilidade e agilidade que apresentam algumas das aplicações atuais. Mas é de se esperar que os projetistas de soluções de banco de dados aos poucos incorporem estruturas e mecanismos complementares às relacionais para apoiar esse tipo de aplicação. Outra dificuldade encontrada em bancos de

dados relacionais é a complexidade de reajustar sua estrutura caso haja uma mudança em alguma parte dela. Por exemplo, se o banco foi projetado para operar com um limite de tamanho físico e o dado inserido não couber em uma tabela então será necessário reajustar todo o banco de dados, rever relacionamentos, índices, entre outras propriedades.

Tanto o SQL quanto a tecnologia NoSQL têm mostrado importantes resultados ao longo do tempo para garantir o armazenamento de dados e sua recuperação de forma otimizada e correta. Portanto, decidir sobre um ou outro tipo não é vantajoso ainda. Se existe um alarde em relação ao NoSQL atualmente, isto não significa que esta tecnologia vai resolver todas as necessidades do problema. Ambas tecnologias são excelentes no que elas se propõem a fazer. Cabe apenas ao desenvolvedor retirar o melhor de cada uma delas dependendo da situação e do do que for mais conveniente para o objetivo proposto.

4 EXPERIMENTAÇÃO: TRATAMENTO E ACESSO DE DADOS DO PINGER EM BANCOS DE DADOS RELACIONAL E NoSQL

Este capítulo tem como objetivo definir as etapas e critérios considerados de forma a propiciar a comparação entre um banco de dados relacional MySQL e um banco de dados NoSQL Cassandra para os dados do Projeto PingER.

Primeiramente, será introduzido o ambiente computacional utilizado como base para todos os experimentos. O objetivo deste ambiente é garantir a mesma plataforma de testes, para que não haja tendências para um ou outro tipo de banco de dados. Serão discutidas algumas características da instalação dos softwares, assim como as diversas particularidades de cada um para atender aos critérios propostos. Em seguida, vamos descrever o planejamento, através da descrição das fontes de dados necessárias e de como elas devem ser integradas, de forma a compor o esquema inicial dos dados no modelo lógico multidimensional.

A partir disso, o próximo passo é aplicar as etapas do ETL e garantir que os dados estejam consistentes com o modelo e preparados para a realização das visualizações analíticas após a carga deles em cada um dos tipos de bancos de dados. O processo de ETL deve considerar a extração a partir das fontes de dados definidas previamente, bem como a realização de tratamentos adequados para que as tabelas do modelo físico reflitam as mesmas informações da modelagem inicial.

Como forma de comparar o desempenho nos ambientes relacional e NoSQL, introduziremos um conjunto de consultas que manipulam diversas fontes de dados, e que são relevantes para o projeto como um todo, ao mesmo tempo. Isto é feito com o intuito de possibilitar a geração de análises que respondam a questionamentos importantes, e que também sejam capazes de medir computacionalmente os experimentos, através de indicadores quantitativos. Com isso, as visualizações analíticas podem ser realizadas pelo cruzamento das informações contidas nas tabelas do modelo. De acordo com as métricas estabelecidas no início do capítulo, é feita uma tabela de comparação para cada uma das consultas realizadas sobre os dois bancos de dados. Por último, faremos as considerações gerais em relação ao que foi observado durante as análises de todas as consultas em cada um dos dois ambientes de testes.

4.1 CARACTERÍSTICAS DO AMBIENTE DE TESTES

4.1.1 Sistema computacional, SGBDs e outras ferramentas

Como forma de mensurar os resultados, utilizamos uma plataforma única durante toda a realização prática deste trabalho, para evitar diferenças computacionais de processamento e de transferência de dados entre os experimentos. Sendo assim, ambos avaliam toda a distribuição dos dados. Os testes foram realizados sob um cluster de apenas um nó para os dois ambientes de teste; e sob um cluster de dois nós através de duas instâncias do Cassandra que são executadas paralelamente em uma mesma máquina apenas. O artigo que fundamenta a aplicação dos testes pode ser encontrado no site oficial do DataStax[17]. A máquina utilizada possui um processador Intel Core i7 @2.80GHz, com 24.0 GB de memória RAM e HD SSD Samsung de 256GB, executando Windows 7 Ultimate 64-bit como sistema operacional. Todas as bases de dados foram armazenadas em uma única unidade de disco, e, durante a execução de todas as transformações e consultas, foram utilizados apenas os processos padrões do Windows, além do processo da própria ferramenta. A versão do Java instalada e utilizada é a 1.7.

Para configurar o ambiente de testes do banco de dados no MySQL, instalamos o software MySQL Installer 5.6, que inclui todo o pacote de ferramentas necessárias para a criação e definição de campos e tabelas, assim como a geração dos índices e relacionamentos entre todas tabelas. Estes elementos representam a estrutura física do banco de dados MySQL, que, a partir deste momento, está preparado para a carga dos dados na etapa de ETL.

Uma vez definido o Cassandra como o tipo de SGBD a ser utilizado e comparado com o MySQL, precisamos armazenar os dados e processá-los, para, somente então, responder a todas as consultas necessárias. Foi utilizada uma versão estável do Cassandra neste trabalho de forma a garantir que os erros de projeto fossem minimizados (Apache Cassandra 2.1.5). Definimos, neste momento, o ambiente que vamos utilizar, para que possamos estudar e comparar os resultados das consultas de acordo com as métricas pré-estabelecidas.

Configurar o ambiente Cassandra no Windows é uma tarefa que exige um esforço maior, se comparado ao Linux. Deste modo, o tempo para configuração impactou diretamente no cronograma de desenvolvimento do trabalho.

Conforme o manual disponível no site oficial do Cassandra, o processo de instalação deve seguir algumas etapas estritamente necessárias para que a configuração do cliente/servidor Cassandra esteja correta. Em primeiro lugar, efetuamos o download de uma

[17] http://www.datastax.com/dev/blog/running-multiple-datastax-enterprise-nodes-in-a-single-host

versão binária estável do site oficial do Apache Cassandra e, após nos certificarmos de que o Java SDK foi previamente instalado, configuramos as variáveis de ambiente JAVA_HOME e CASSANDRA_HOME.

Com o Java instalado e configurado, estabelecemos o diretório de armazenamento de todas as tabelas do Cassandra no arquivo Cassandra.yaml, da pasta conf. Ao abrirmos o arquivo, procuramos as linhas onde estiver escrito "/var/lib/cassandra", e alteramos para o caminho da aplicação no Windows, como, por exemplo, "C:/Cassandra".

Por último, acessamos o diretório C:\Cassandra\bin e abrimos o arquivo cassandra.bat. E, finalmente, após executarmos todos os passos, temos o servidor Apache Cassandra funcionando.

Assim como outros SGBDs NoSQL, o Cassandra não suporta transações ACID, e também não suporta JOINs. Quando um usuário precisa unir duas famílias de colunas, é necessário recuperar e unir os dados separadamente. Este é um processo computacionalmente caro e demorado caso o banco seja muito grande. Por isso, decidimos armazenar a maior quantidade de dados possível em uma mesma linha na tabela fato, de forma a tornar a recuperação mais eficiente. Sendo assim, as consultas foram respondidas da melhor maneira possível.

Ao final dos experimentos, desenvolvemos as visualizações *ad-hoc* das consultas, tanto no MySQL quanto no Cassandra, de acordo com o que foi proposto pelo projeto MultiLOD, nas bases de dados do modelo multidimensional, no Cassandra. Com isso, fomos capazes de responder a diversas perguntas, no que tange ao desempenho das consultas de acordo com as métricas estabelecidas. Caso o desempenho não se mostrasse satisfatório, deveríamos analisar novamente os relacionamentos feitos entre as tabelas do modelo, e, se necessário, refazer o modelo construído.

4.1.2 Características da aplicação

Após a configuração inicial dos dois ambientes de testes, e definidas todas as fontes de dados e suas características, estabelecemos o planejamento, que serviu de base para o ETL e, consequentemente, para a análise e visualização dos resultados. Fez-se necessário colocar a modelagem dos dados como um critério obrigatório nesta etapa de planejamento. Desenhamos o modelo multidimensional em uma ferramenta apropriada, que utiliza uma notação adequada para tal. A notação de modelagem recomendada para os modelos de dados, por padrão, é a IE (Information Engineering ou Engenharia da Informação). Com isso, foi escolhida uma ferramenta CASE (do inglês Computer-Aided Software Engineering) —

EDraw Max 7.2 — como forma de representar a construção dos modelos lógicos, fiéis ao modelo físico do banco de dados. Esta ferramenta simplifica a modelagem, criação e manutenção de bases de dados, data warehouses, entre outros modelos de dados. Ela não apenas permite que o usuário defina as necessidades e regras na forma de um modelo de dados lógico, mas também que as converta em seu equivalente físico para um dos bancos de dados suportados. Neste trabalho, no entanto, não exploramos esta funcionalidade.

Utilizamos também neste trabalho a versão gratuita do software Navicat for MySQL (Windows) 9.1.8 para manipular os dados armazenados nesse sistema de banco de dados, pois, comparado ao MySQL Workbench (uma das ferramentas do MySQL Installer), ele se comportou como uma poderosa ferramenta para a administração e desenvolvimento de todo o banco de dados MySQL. A ferramenta possui amplo suporte para a administração e construção de servidores de banco de dados MySQL, além de ser utilizada por empresas conceituadas, como Google e Apple. Possui, como uma de suas principais características, a de ser um aplicativo sofisticado, voltado a desenvolvedores profissionais, e, ao mesmo tempo, de fácil entendimento para os iniciantes. Durante todas as fases do ETL, alimentamos a base relacional carga_ping com os dados oriundos das diversas fontes de dados mencionadas anteriormente, e, logo em seguida, carregamos todos os dados dessa base para a base multidimensional carga_ping_dwh. Para isso, foi utilizado o software Pentaho Data Integration (PDI) 5.2 também usualmente conhecido como Kettle. Conforme foi descrito no capítulo anterior, este software tem como finalidade realizar soluções de ETL a partir de diversas fontes de dados, sendo eles estruturados ou não.

O Kettle possui quatro programas que são utilizados em diferentes fases de desenvolvimento e deploy: Spoon, Kitchen, Pan e Carte. Nesta etapa, que compreende o ETL, focamos apenas na ferramenta Spoon, que é a IDE do Kettle, e essencial para toda a fase de desenvolvimento de um ETL. Alguns conceitos básicos do Kettle foram definidos antes, para o entendimento do processo de ETL. São eles: transformations ou transformações (que é a nomenclatura utilizada durante o trabalho), steps e jobs.

Uma transformação é o principal elemento do processo de ETL. Em linhas gerais, ela pode ler dados de arquivos ou tabelas, manipular, filtrar e “limpar” esses dados, e inseri-los em uma base de dados ou em um arquivo CSV, por exemplo. A transformação é composta por um ou mais steps. É importante ter em mente que cada step é executado em paralelo em uma transformação.

Um step faz parte de uma transformação, e pode ter diversas funcionalidades, tais como: ler dados estruturados (e.g.: bases de dados relacionais, CSV); ler dados não estruturados

(e.g.: arquivos de texto); executar alterações sobre esses dados; e, também, escrever dados em algum arquivo ou base de dados. Neste trabalho não descrevemos a funcionalidade de cada step. Porém, existe uma enorme variedade de steps que são listados na documentação do Spoon.[18]

Um job, entre outras funcionalidades, pode executar um conjunto de transformações, capturar qualquer erro que a execução de uma transformação possa gerar, e salvar logs dessas execuções, por exemplo. Um job consiste de diversas entradas de outros jobs e/ou transformações, que podem se equiparar aos steps de uma transformação. Ao contrário das transformações, os jobs são executados respeitando a ordem dos steps.

4.2 PLANEJAMENTO E MODELAGEM MULTIDIMENSIONAL

Antes de abordar uma solução para o problema, é necessário definir o escopo e entender qual o objetivo do problema, para depois elaborar uma solução viável para ele.

Complementando a definição prévia do projeto, apresentada na seção anterior, o Projeto PingER tem como objetivo principal a coleta de dados que caracterizam diversos indicadores de qualidade de rede, sobre vários nós representativos, em mais de 160 países do mundo. A informação é medida através da relação de medições entre um par de nós (cerca de 80 nós monitores para aproximadamente 800 nós monitorados), o que corresponde a algo em torno de 8.200 pares. A informação mais atual sobre a quantidade de nós existentes é de 69 nós monitores e 751 nós monitorados.

Após fazer a carga de todos os arquivos com as métricas de rede, através do link do Laboratório SLAC, foi necessário definir seu escopo. Os arquivos de cada tipo de métrica são categorizados pelas medições diárias (alcançando o grão temporal do formato de hora) e por medições mensais, que atingem o de grão temporal no formato de dia.

Nesta etapa de análise do domínio, que abrange a seleção dos dados que estão no escopo do trabalho, considerou-se publicar os dados de medida com a dimensão temporal em seu menor grão (maior detalhe) disponível, isto é, em hora. Porém, descartamos esta opção, para compatibilizarmos a análise com o trabalho realizado por Souza (2013) — onde ele menciona que o líder do PingER alertou que seriam dados demais — e para viabilizarmos a realização deste trabalho, em relação ao tamanho da base, considerando o uso de uma máquina doméstica de alto desempenho. Desse modo, utilizamos os dados diários (grão dia), desde 1998, como o menor grão temporal.

[18] http://wiki.pentaho.com/display/EAI/Pentaho+Data+Integration+Steps.

Definido o menor grão a ser utilizado no modelo multidimensional, iniciamos a elaboração da etapa do ETL dos dados. Antes de iniciar o *workflow* de transformações no Kettle, foi preciso definir alguns procedimentos de limpeza dos dados. Primeiro, removemos, de cada um dos 11 diretórios dos indicadores, todos os arquivos diários dos diretórios que representam medições de hora em hora, para que fossem deixados apenas os arquivos mensais com representações diárias.

Pelas características do PingER, percebeu-se uma natureza de dados passíveis de manipulação numérica e estatística, os quais são gerados a partir de cruzamento de parâmetros (período de tempo, região, tipo de métrica de rede, etc.). Vimos também, que os dados são acumulados ao longo dos anos, e são utilizados para realizar estudos que se relacionam à qualidade de rede, e, ainda, dão suporte à tomada de decisão. Dados com essas características são amplamente estudados, e já existem modelos clássicos para descrevê-los e representá-los. Um dos modelos conceituais mais usados em dados dessa natureza é resultante dos estudos de Data Warehousing (DW), conhecido como esquema estrela (*star schema*).

Resumidamente, um esquema estrela baseia-se numa visão multidimensional dos dados, na qual o valor sendo medido é comumente visualizado numa posição central (em DW, chamado de tabela fato), e os parâmetros (chamados de dimensões), que o definem, ficam em posições satélites, formando um esquema análogo a uma estrela (LEARN DATA MODELING, 2012).

Conforme foi mencionado, uma medida de rede é calculada sobre uma solicitação de ping, a partir do nó origem, até que ela seja recebida pelo nó destino. Essa medida é estabelecida pelo cruzamento de diversos parâmetros, que definem onde a medida ocorreu, quando ela ocorreu, e o que a medida se propôs a medir, ou seja, qual a sua métrica de rede correspondente.

4.2.1 Descrição das fontes de dados complementares

Para que uma parte das consultas do projeto pudesse corresponder às já executadas e utilizadas em projetos anteriores, extraímos diversas informações de fontes de dados complementares, como DBPedia, World Bank, GeoNames e UCDP.

a) DBPedia

- É um projeto cujo objetivo é extrair conteúdo estruturado das informações da Wikipédia. Essa informação estruturada é, então, disponibilizada na Web, criando uma rede de ligações entre os dados, e permitindo, aos usuários,

realizar consultas sobre o conteúdo da Wikipédia de forma similar a consultas de banco de dados.

- Os dados são acessados usando uma linguagem de busca semelhante ao SQL para RDF, chamada SPARQL.

b) GeoNames

- GeoNames é uma base de dados geográfica RDF, disponível e acessível através de diversos serviços web. Essa base é aberta e pública, e possui mais de 10 milhões de nomes de lugares, cobrindo todos os países, cidades, estados, oceanos e lagos, ruas, aeroportos e diversos outros locais.

c) World Bank

- Fornece o acesso aberto e público de dados através de indicadores de desempenho (social, econômico, educacional, ambiental, etc.) de diversos países do mundo. Neste trabalho, utilizamos apenas indicadores econômicos como fonte de dados.

d) UCDP

- Coleta informação sobre uma grande quantidade de perspectivas de conflitos, desde 1946. Em um esforço de registrar todos os detalhes acerca dos conflitos e crises, este programa abrange diversos critérios e particularidades que pertencem a cada tipo de conflito.
- O conjunto de dados do UCDP é um dos mais confiáveis e utilizados para representar a dinâmica dos conflitos armados que ocorrem em todo o planeta, e servem, também, como estudo, para entender como os conflitos são originados e classificados.

4.2.2 Criação do esquema inicial

A seguinte nomenclatura foi respeitada para as tabelas do modelo físico e lógico. utilizamos, o prefixo “fato_”, acompanhado da tabela fato, e o prefixo “dim_” acompanhado da tabela correspondente, para as tabelas de dimensões definidas no modelo.

Além dos arquivos em formato texto, que foram extraídos do Laboratório SLAC, contendo informações sobre as medições dos nós de rede, também definimos outras fontes de dados que complementaram o restante das informações. Abaixo, definimos a origem, e, logo em seguida, a estrutura de cada um dos arquivos extraídos.

Foi desenvolvido um esquema inicial para os dados do experimento, a partir do modelo elaborado no projeto PingER LOD. Este esquema está de acordo com a base de dados originada pela estrutura de arquivos do projeto PingER, baseando-se no esquema estrela:

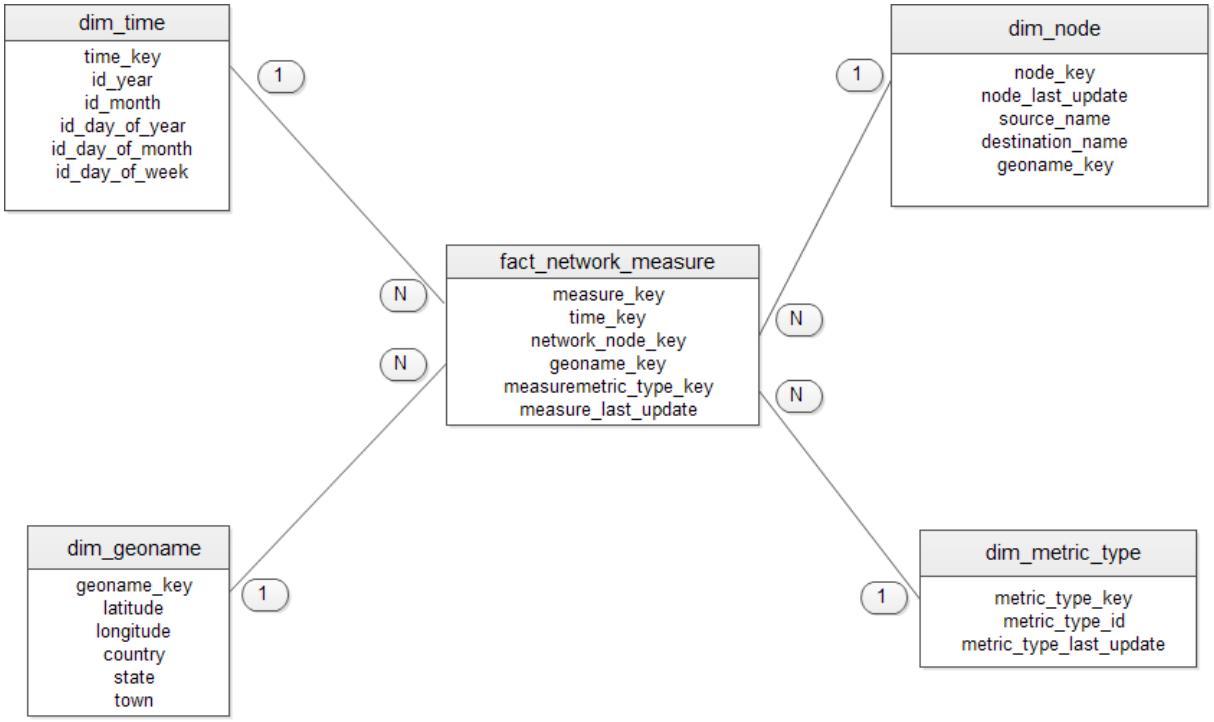

Figura 1 — Modelo Multidimensional — Desenvolvimento do esquema inicial.

Em adição ao esquema estrela construído, a modelagem foi complementada a partir das fontes de dados de origem, através da organização de todas as tabelas em um modelo floco de neve. No modelo floco de neve, as tabelas dimensionais relacionam-se com a tabela de fatos, mas algumas dimensões relacionam-se apenas entre si mesmas. Isto ocorre para fins de normalização das tabelas dimensionais. Neste modelo, as tabelas de dimensões auxiliares normalizam as tabelas de dimensões principais, visando diminuir o espaço ocupado por estas tabelas.

Através da manipulação do banco conforme mencionado acima, passamos a utilizar mais tabelas para representar as mesmas dimensões, mas ocupamos um espaço em disco menor do que o do modelo desenvolvido no esquema estrela. Este modelo chama-se floco de neve, pois cada dimensão se divide em várias outras tabelas, que, organizadas de certa forma, lembram um floco de neve.

Desta forma, tornamos a modelagem mais complexa em relação à navegação através da ferramenta de manipulação do banco de dados. Considerando que este modelo utiliza mais tabelas para executar uma consulta, foram realizadas mais instruções JOINS do SQL,

tornando o acesso aos dados um pouco mais lento do que no modelo estrela. Entretanto, os benefícios deste modelo justificaram a sua adoção neste projeto, pela forma como suas tabelas foram construídas e pela quantidade de registros que elas contêm. A partir dessas ideias, e com base tanto no modelo conceitual prévio quanto nas fontes de dados definidas anteriormente, foi elaborado um modelo lógico multidimensional (Figura 2) para o domínio considerado neste trabalho, conforme o modelo floco de neve.

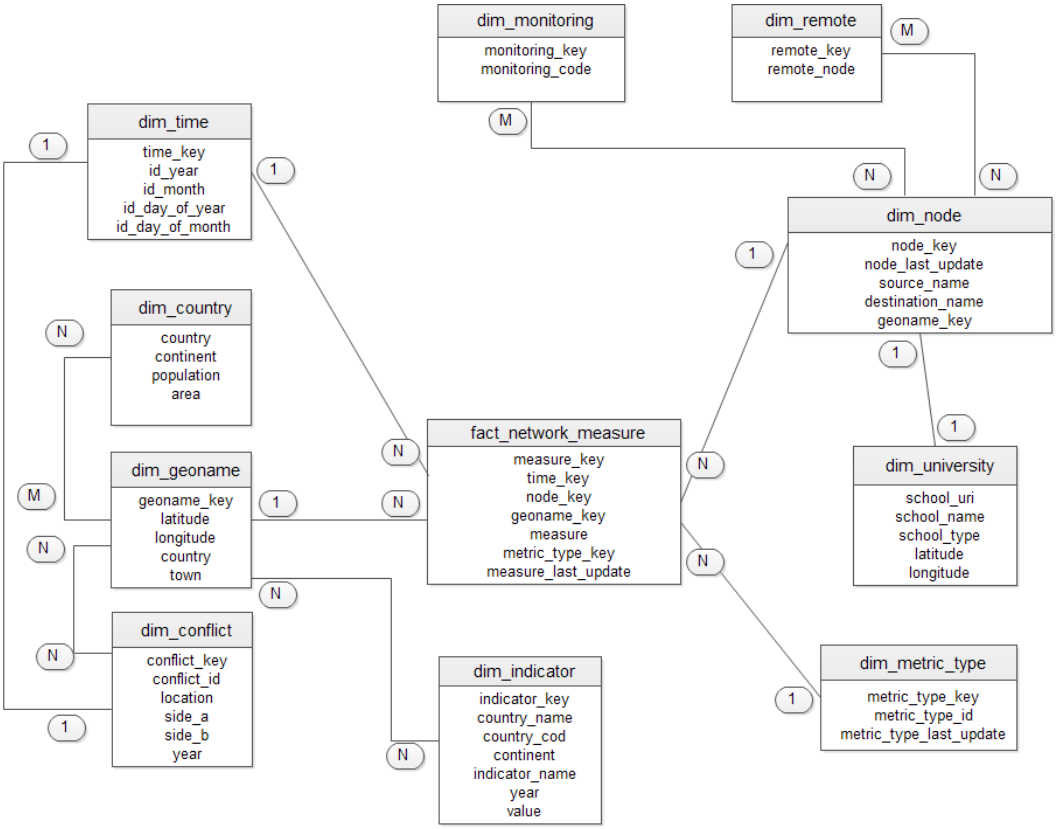

Figura 2 — Modelo Multidimensional do PingER e fontes de dados complementares, baseado no esquema floco de neve (Snow Flake).

De acordo com o modelo multidimensional, foi elaborado um glossário com as definições dos atributos das tabelas e seus relacionamentos, bem como os termos utilizados durante as etapas de análise e visualização dos dados. O glossário está descrito no Anexo C.

4.3 PROCESSOS DE ETL

Para o começo do desenvolvimento do fluxo, consideraram-se os arquivos extraídos manualmente do diretório do SLAC em formato texto (.txt), e, com base na modelagem do esquema estrela inicial, aplicamos as operações de transformação, para, somente então, realizar a carga no modelo físico. A partir do modelo construído no banco de dados,

elaboramos as dimensões auxiliares, até a conclusão do modelo multidimensional completo no esquema floco de neve. De modo resumido, a partir dos arquivos de origem que estavam em formato .txt, .csv ou .xls, desdobramos as etapas para carregar os dados em um banco de dados relacional, através do processo de ETL para a modelagem multidimensional, até chegarmos à análise desses dados em uma ferramenta web. Com base no modelo relacional físico elaborado no MySQL, foi realizada a carga dos mesmos dados de origem em um novo banco de dados MySQL e também no Cassandra. Ambas estruturas seguem o modelo da Figura 2, e para construí-lo, foi necessário desenvolver diversas etapas internas de denormalização de dados que podem ser construídas a partir de steps padrões do Kettle. Inicialmente, utilizamos apenas ferramentas open source durante o ETL. Porém, na fase da análise, foi necessário utilizar versões comerciais de teste dessas ferramentas, visto que são as únicas que possuem funcionalidades que atendem ao objetivo deste trabalho.

No início deste projeto, foi considerada a carga de dados com informações até outubro/2014. Ao longo do desenvolvimento, estabeleceu-se que cargas incrementais poderiam ocorrer com uma frequência mensal. Entre novembro/2014 e abril/2015, nenhuma das métricas foi atualizada no diretório web do SLAC. Em maio/2015, quando já havia sido concluída a fase de experimentos, observamos um incremento de arquivos no diretório, com a inclusão de arquivos de todas as métricas até o dia 10/05.

4.3.1 Processo de tratamento e carga dos dados no banco de dados relacional

Como a etapa de ETL é totalmente realizada através do Pentaho, foi necessário definir os próximos passos como forma de *workflow* de transformações desse processo.

Com isso, atribuímos um diretório padrão de armazenamento dos arquivos do PingER para realizar iterações a partir de cada um deles. Observamos que cada arquivo segue uma estrutura interna similar, então foi possível executar rotinas personalizadas que atenderam a cada um individualmente. O PingER utiliza o padrão AAAA-MM, que se aplica à nomenclatura de cada arquivo, e o padrão MmmDD foi adotado como cabeçalho para as colunas das medições dentro de cada arquivo (ex.: 2013-01, Jan12), sendo dias, meses e anos iguais a DD, MM/Mmm e AAAA, respectivamente.

O padrão de cada arquivo respeita a seguinte classificação: Coluna 1 = Nó Monitor, Coluna 2 = Nó Monitorado, Coluna 3 = Mmm01, Coluna 4 = Mmm02, e assim por diante, até o último dia do mês. Cabe lembrar que as duas últimas colunas são iguais às duas primeiras colunas de cada arquivo. Portanto, elas são ignoradas no momento da seleção dos dados carregados no banco de dados.

Em seguida, criamos uma rotina para salvar em um arquivo no formato .xls, os nomes de todos os arquivos que foram carregados na base de dados utilizando o step ‘Get File Name’. Com essa informação salva, conseguimos agilizar o procedimento de carga feito para cada um dos arquivos selecionados.

A primeira informação armazenada foi o tipo da métrica obtida no momento inicial da leitura do arquivo, através de um step de modificação Java Script, em que selecionamos apenas uma parte do nome de cada arquivo lido. Tanto as informações do nó origem, quanto as do nó destino, foram encontradas pela seleção das duas primeiras colunas de cada uma das linhas.

Uma vez encontrada parte da data de medição (mês/ano), através do nome do arquivo, restou apenas identificar o dia em que cada medição ocorreu e concatenar, para obter a data completa. Para isso, realizamos uma sequência de steps que coletam dados de medição de todos os dias para cada par de nós, e salva em uma tabela temporária, que é truncada a cada arquivo lido novamente. Finalmente, fizemos uma consulta, que retornou os valores de medição para cada dia referente a cada par de nós do arquivo, e acrescentamos às outras informações obtidas. Portanto, conseguimos normalizar todas as informações no banco de dados relacional ao final do processo de carga.

A leitura para todos os arquivos iniciais, desde 1998 até outubro/2014, consumiu a maior fatia de tempo de processamento, considerando todas as etapas deste trabalho. Isso se deu não somente pela escolha da granularidade temporal em nível de dia para o extenso intervalo de anos, mas também pela quantidade média de medições mensal. Na maioria dos casos, a quantidade de par de nós, correspondentes a uma medição mensal de uma métrica, ultrapassava os 1.000 pares. Porém, em diversos outros arquivos, observamos uma quantidade bem maior, de cerca de 10.000 pares de nós. Efetuando um simples cálculo, a partir disso, teríamos, então, cerca de 300.000 linhas processadas para este arquivo em questão, considerando que, para cada par de nós, temos, em média, 30 dias de medições.

O tempo total de processamento de carga foi avaliado em torno de 35 horas, para todos os 1.982 arquivos, relativos a todas as 10 métricas de rede, resultando, em média, a 1 arquivo lido por minuto.

a) Processo de carga das informações de localização dos nós

Nesta etapa, foi utilizado um código em Java desenvolvido durante o projeto MultiLOD. Este código executa uma rotina para cada entrada de um nó, com base no arquivo NodeDetails (formato JSON), também utilizado pelo projeto PingER. O arquivo possui o

detalhamento de todos os nós, com informações (Latitude, Longitude, NodeName, IP, etc.) encontradas na base do Geonames. Todas as informações geográficas encontradas neste arquivo são de grande importância para a maioria das análises deste projeto.

Portanto, para recuperar as informações referentes à localização dos nós, utilizamos apenas um step de entrada, que recebe o arquivo .xls de entrada gerado pelo código mencionado, e, logo em seguida, criamos um step que carrega as informações na tabela dim_geonames.

A base de dados com as informações armazenadas até a etapa corrente, foi confrontada com o modelo utilizado, para confirmar todos os processos de carga das informações citadas anteriormente. Alguns pequenos ajustes estruturais e de nomenclatura foram feitos nas transformações, até a validação completa de todo o modelo conceitual. Nas etapas a seguir, definimos os processos de carga que suportam as dimensões auxiliares do modelo multidimensional da Figura 2.

b) Processo de carga das informações de conflitos

Para recuperar dados sobre conflitos, realizamos diversas tentativas de busca através da base do DBPedia. Porém, não foram encontradas informações suficientes sobre a situação dos países, devido à forma de categorização de eventos de crise fornecer poucos detalhes, dificultando o objetivo de construção de uma base de dados confiável.

Decidimos consultar outras fontes de dados, até que as informações sobre a situação de conflitos (guerras ou eventos de crises) dos países, nos diversos períodos de tempo, fossem encontradas de fato. Entre as diversas fontes pesquisadas, encontramos resultados mais acurados no UCDP (Uppsala Conflict Data Program), que é um programa de coleta de dados desenvolvido no departamento de paz e conflito da Universidade de Uppsala. Recuperamos as informações desejadas extraindo um arquivo com o conjunto de dados de conflitos ano a ano sobre conflitos armados, com pelo menos um participante sendo o governo do país (UCDP/PRIO Armed Conflict Dataset). De forma complementar, obtivemos também um conjunto de dados com informações de ataques intencionais sobre civis a partir de governos e organizações de outros grupos armados.

O conjunto de dados tem diversas informações de natureza irrelevante ao projeto. Desse modo, extraímos apenas um intervalo de dados e as colunas necessárias, e carregamos na tabela dim_conflitos do modelo multidimensional.

c) Processo de carga das informações de universidades

A consulta em SPARQL utilizada no trabalho de Souza (2013), Apêndice D, não retornou resultados através do site de consultas em SPARQL utilizando o endpoint sobre a DBPedia,[19] como podemos ver na Figura 3:

Figura 3 — Consulta em SPARQL utilizando o endpoint sobre a DBPedia.

Para recuperar a base com as informações das métricas das universidades, originada do DBPedia, foi utilizado o arquivo *dump* do RDF, do Laboratório SLAC, disponível no site http://pingerlod.slac.stanford.edu/.

De forma a possibilitar a realização da consulta utilizando a métrica de todas as universidades, foi criado, então, um repositório, para a importação do arquivo *dump* com conteúdo RDF das informações de todo o projeto PingER LOD até dezembro/2013. Para criar o repositório, foi necessário instalar o Sesame, que é um framework Java open source para armazenamento e consulta de dados em BD RDF. Assim, foi possível inserir e manipular dados em arquivos no formato RDF através de consultas básicas, usando SPARQL para viabilizar uma forma de comunicação e um acesso aos componentes internos desse tipo de arquivo.

No entanto, após carregar a base no repositório e ajustar suas configurações de diretório através do link de acesso (http://localhost:8080/openrdf-workbench/), foi utilizado um plugin (openRDF for PDI) criado para o Kettle, de forma que a consulta em SPARQL pudesse acessar todos os seus dados além de possibilitar a conexão com outros formatos de dados de saída. O plugin recebe de entrada um arquivo armazenado em um repositório, e possui um campo no qual podemos inserir uma consulta em SPARQL, executada sobre os dados desse arquivo. Em seguida, o plugin retorna os campos selecionados de acordo com todos os parâmetros da consulta.

Foi criada, então, uma transformação para retornar o resultado em formato .xls. E, a partir deste arquivo, realizamos a carga no modelo multidimensional na tabela dim_universidades. A consulta detalhada em SPARQL para recuperar os dados do arquivo *dump* pode ser encontrada no Apêndice A.

[19] http://dbpedia.org.

d) Processo de carga das informações de indicadores que são representados em função da porcentagem do PIB (Produto Interno Bruto) de diversos países, porém neste trabalho será utilizado apenas o indicador do investimento que um determinado País aplica emdesenvolvimento e pesquisa)

Em adição aos estudos já realizados, decidiu-se explorar ainda mais o relacionamento entre as diversas fontes de dados, para complementar o modelo com informações do contexto econômico de todos os países. As informações obtidas atendem a algumas das consultas selecionadas previamente, cujo objetivo é investigar a relação entre a qualidade de rede e o investimento dos países (em relação ao % do seu PIB) em pesquisa e desenvolvimento tecnológico. Como já citamos, essas informações são originadas do World Bank, e foram acumuladas ao longo dos anos.

O BD RDF do World Bank (2.3.2.d) contém todas as informações necessárias sobre os investimentos do PIB dos países. Foi preciso unir os dados do PingER com os dados do BD RDF, para realizar o cruzamento entre as bases, e, para isso, recuperamos os dados do World Bank e carregamos no banco de dados do PingER LOD. Os dados se encontram no site do World Bank,[20] no arquivo WDI.xlsx. Após o download do arquivo, alguns ajustes foram feitos para diminuir o tamanho do conjunto de dados, visto que precisamos apenas de um indicador econômico para atender ao objetivo proposto.

Coletamos apenas um conjunto de dados para os anos do escopo do PingER, e, em seguida, carregamos todos os dados em uma tabela, que representa uma das dimensões do modelo multidimensional elaborado.

e) Processo de carga das informações de países

O conteúdo desta tabela é de baixa relevância para o projeto, ao comparar com o restante das informações, porém, permite um maior detalhamento sobre a localização dos nós, uma vez que relaciona cada país com seu continente e com informações sobre cada país. Resolvemos utilizar apenas as colunas relativas aos continentes, países e cidades, além de outras informações, como população e área, que podem ser importantes futuramente. Diversas outras informações sociais também podem ser encontradas na origem desta fonte de dados. Supondo que esta tabela será atualizada apenas uma única vez, retiramos todos os indicadores sociais que podem se alterar com o tempo.

[20] http://data.worldbank.org/data-catalog/world-development-indicators.

f) Processo de carga das informações de nós monitores e monitorados

Por fim, consolidamos todas as informações geográficas e econômicas, assim como as informações sobre conflitos e universidades, e finalmente efetuamos o processo de carga dos nós de rede monitores[21] e monitorados.[22] Através destes arquivos obtemos então, todas as informações de nós a partir dos dados da tabela fato construída. Nesta etapa podemos ainda encontrar diversos nós desativados, para complementar as buscas com as informações mais atuais. Caso a busca pela medição para um nó específico não retorne nenhum resultado, realizamos uma nova busca a partir dos campos origem_name e destino_name, na tabela dim_no_rede. Esta dimensão armazena as informações históricas das rotas entre todos os nós desde o início em que as medições ocorreram.

Todos os processos de carga listados foram executados automaticamente no Kettle, e alguns deles podem ocorrer em paralelo, principalmente caso o processamento seja feito a partir de fontes de dados distintas. Com isso, terminamos de popular nossa base carga_ping_dwh, e temos uma base ROLAP (*Relational Online Analytical Processing*) que está preparada para ser analisada por qualquer ferramenta OLAP (*Online Analytical Processing*) que entenda essa estrutura. Construímos assim o modelo físico baseado no esquema multidimensional no banco relacional e, através dele, dispomos de todos os insumos para carregar todos os dados em um banco de dados do Cassandra (vide Apêndice B). Em seguida, o objetivo foi utilizar as informações das fontes de dados que compõem as dimensões auxiliares do modelo para popular o banco de dados NoSQL.

4.3.2 Processo de tratamento e carga dos dados no Cassandra

Como já organizamos os dados no modelo multidimensional, então precisamos apenas criar as estruturas de armazenamento no Cassandra e selecionar todas as bases do modelo relacional para carregar em uma base de dados do Cassandra.

Primeiro, acessamos o aplicativo cliente do Cassandra encontrado no diretório local C:\Cassandra\bin\, e abrimos o arquivo cassandra-cli.bat. Criamos um Keyspace para armazenar as tabelas do banco de dados do Cassandra, através do comando ‘CREATE KEYSPACE tblmedicao’. Com a criação do Keyspace, foi possível criar as tabelas, utilizando o comando ‘CREATE COLUMN FAMILY’, para cada uma das tabelas. Para nomear as famílias de colunas que caracterizam as tabelas do modelo físico e lógico, foi utilizado o prefixo “cf” acompanhado da tabela correspondente do modelo multidimensional. O script

[21] http://www-wanmon.slac.stanford.edu/cgi-wrap/dbprac.pl?alias=all.
[22] http://www-wanmon.slac.stanford.edu/cgi-wrap/dbprac.pl?monalias=all.

completo com o desenvolvimento do modelo físico no Cassandra referente ao modelo multidimensional pode ser visto com mais detalhes no Anexo D.

A partir das estruturas criadas, desenvolvemos as transformações no Kettle, para carregar todos os dados. Neste passo, as transformações foram criadas em apenas 2 steps, pois temos todas as informações na base de dados estruturada no modelo multidimensional. Com isso, foi preciso apenas selecionar todos os campos de cada tabela dimensão e da tabela fato e carregar no Cassandra, através do step Cassandra Input.

As configurações necessárias de carga deste step incluíam apenas o nome do Keyspace, da família de colunas, do campo identificador que é a chave da família de colunas e opções de marcação para as configurações de esquema (criar automaticamente a família de colunas, truncar a família de colunas e inserir campos que não estão na família de colunas). Essas configurações foram feitas para garantir a carga dos dados de todas as tabelas.

Após a execução de todas as transformações, armazenamos os dados no Cassandra no modelo multidimensional. Podemos confirmar se todas as famílias de colunas foram armazenadas corretamente no Cassandra através de uma consulta no CQLSH, que é o terminal de comando shell para executar consultas em linguagem CQL do Cassandra.

As primeiras tentativas para carregar dados em famílias de colunas do Cassandra não retornaram dados de acordo com o esperado, ou seja, alguns erros de duplicidade de dados ou falhas na conexão foram observados diversas vezes. Alguns pequenos ajustes foram feitos no step de Input do Cassandra; por exemplo, o tamanho do batch e o campo identificador que deve ser utilizado como chave da família de colunas. Garantimos, com isso, que os dados inseridos eram exatamente os mesmos armazenados em ambos os ambientes de testes, para que fosse desenvolvido o processo de análise e visualização, respectivamente.

O conjunto de ferramentas do Cassandra não possui uma ferramenta nativa de visualização da análise de toda a base de dados. E, através do terminal de consultas do Cassandra, conseguimos acesso a um conjunto de informações bastante modesto, que não satisfaz completamente os nossos objetivos.

A Plataforma BI da Pentaho recomenda a utilização de um aplicativo contido em seu próprio conjunto de ferramentas de análise (Instaview) para a manipulação de dados Big Data em bancos não relacionais, tais como o Cassandra, no qual podemos importar os dados e explorá-los com muito pouco esforço. Segundo o manual do aplicativo, isso pode ser feito apenas apontando o Instaview para o cluster Cassandra, selecionando o Keyspace desejado e, depois, inserindo consultas em MDX, para realizar operações OLAP (Slice and Dice) por sobre os dados do Cassandra.

Foi observada uma limitação de número de linhas de entrada para o banco de dados do Cassandra no Instaview, pois o conector do Cassandra no Instaview não suporta a carga para uma amostra de mais de 1 milhão de linhas, e também não suporta queries que utilizem NULL values, o que seria necessário para a etapa de análise. A Figura 4 demonstra o erro encontrado ao executar as consultas que selecionam as tabelas do Cassandra na última etapa de armazenamento em cache dos dados.

Figura 4 — Descrição da falha ao carregar mais de 1.000.000 de linhas no Instaview.

Diversas pessoas também relataram esta restrição, e em nenhum dos casos uma solução foi totalmente esclarecida. Na maioria dos casos, o erro foi resolvido apenas ao diminuir a quantidade de linhas da amostra. Entretanto, gostaríamos de evitar esta alternativa, pela caracterização do grande volume de dados a ser utilizado no trabalho. Algumas configurações internas no modelo e na conexão com o Cassandra foram feitas, na tentativa de solucionar o problema, porém, todas as tentativas se mostraram falhas para o objetivo de carregar uma base com mais de 1 milhão de linhas. Paralelamente, foi tentado contato com o suporte técnico da empresa fornecedora, isto devido às limitações, pelo fato da ferramenta ser uma versão de teste da versão comercial (Pentaho Enterprise Edition).

A partir destes fatos, decidimos, então, descartar a utilização da ferramenta do Pentaho como ferramenta de análise e visualização dos dados necessários. Efetuamos diversas pesquisas por possíveis soluções através de ferramentas similares, e que possuíam as mesmas funcionalidades principais entre si, para abordar essa etapa da solução. Para selecionar outra plataforma que tornasse possível a abordagem, fizemos uma pesquisa em diversos artigos, até que a ferramenta JasperReports Server Pro (Jaspersoft),[23] da companhia TIBCO, foi selecionada, para que pudesse ser feita a análise dos dados e a execução das consultas previamente definidas.

[23] http://community.jaspersoft.com/project/jasperreports-server.

Conforme relatado em um estudo feito por Rita (2014), foi observado que o Jaspersoft se comporta com mais eficiência em relação ao reporte e análise de dados, enquanto que o Pentaho possui um foco maior na integração de dados, no ETL e no workflow de dados. Durante os experimentos deste trabalho, também encontramos as mesmas características da conclusão do artigo, assim como em resultados de outros artigos que influenciaram a escolha pelo Jaspersoft como ferramenta de análise e visualização de dados. A ferramenta oferece suporte para a conexão com bases de dados NoSQL através de um conector nativo. Essa etapa pôde ser desenvolvida com bastante eficiência, considerando a praticidade da manipulação das diversas tabelas do modelo multidimensional, incluindo toda a configuração de metadados e os relacionamentos entre as tabelas.

Assim como a ferramenta Instaview do Pentaho, a TIBCO também possui, na sua versão comercial, a única alternativa para realizar as análises, já que esta versão é a única que inclui visualizações ad-hoc a partir de uma fonte de dados não estruturada.

A princípio, tentamos utilizar a tabela fato completa, que possui aproximadamente 277 milhões de linhas, uma para cada medição de um tipo de métrica diferente, em um determinado dia do ano, e relativa a um par de nós específico. Porém, durante a realização das análises, o tempo para elaborar cada item da consulta, assim como o resultado geral, mostrou-se superior à expectativa inicial. Cada consulta levava mais de 1 hora para retornar seu resultado, considerando a base com todos os dados. Com isso, o universo de dados da tabela fato foi reduzido para cerca de 16 milhões de linhas, já que o elevado tempo para a conclusão dos experimentos inviabilizaria a etapa de análise. Ainda considerando o tamanho da base reduzida, entende-se que o conceito de Big Data continua a existir, e todas as suas características e funcionalidades podem ser aplicadas em termos gerais.

Não deixamos de retornar resultados bastante precisos, já que a amostra aleatória considera todo o intervalo de tempo na ordenação aleatória, e a quantidade de registros ainda continua alta. Além disso, as medições são calculadas através de valores médios para as funções de agregação, e possuem um desvio padrão baixo para o alto volume de dados, ocorrendo, assim, um pequeno impacto no resultado das consultas.

Ao limitar a quantidade de linhas da massa de dados que será utilizada posteriormente, foi estabelecida, então, uma nova tabela, cujo índice foi escolhido randomicamente, para que não interferisse no resultado de nenhuma das consultas. Nessa nova tabela, consideramos apenas os registros que possuem geoname_key não nulo, ou seja, registros cuja localidade pode ser definida com clareza, de forma a aumentar o escopo do resultado de todas as

análises, visto que a maioria das consultas depende de informações tais como país e continente.

Com isso, executamos a consulta que gera a amostra de dados no banco relacional, e fizemos uma nova carga no Cassandra, apenas da tabela fato. Tal restrição de linhas reduziu não somente o tempo de execução da transformação de carga de dados (de cerca de 9 horas para apenas 30 minutos), como também o tempo de execução das consultas. Com essa modificação, conseguimos reduzir o tempo médio de execução das consultas para cerca de 3 minutos, ao limitarmos o volume de dados da tabela fato.

Poderíamos estimar o tempo de execução das consultas caso utilizássemos a base completa. Se considerarmos uma relação linear entre a quantidade de linhas da base e o tempo de execução das consultas, teremos um tempo médio de execução em torno de 20 (vinte) vezes maior. Ou seja, algo em torno de 60 minutos para cada linha, fato que inviabilizaria a realização das análises, considerando a plataforma computacional escolhida, incluindo a arquitetura de hardware e o sistema operacional utilizado.

Abaixo, listamos o exemplo da consulta gerada para selecionar, aleatoriamente, 19 milhões de registros da tabela fato completa.

```
SELECT
  measure_key
, network_node_key
, time_key
, geoname_key
, metric_type_key
, measure
FROM fact_network_measure, (
      SELECT measure_key AS sid
      FROM fact_network_measure
      WHERE geoname_key IS NOT NULL
      ORDER BY RAND( )
      LIMIT 19000000) tmp
WHERE fact_network_measure. measure_key = tmp.sid
ORDER BY measure_key
```

Pelo fato do tipo de métrica Directivity não estar presente no diretório dos dados do SLAC, conforme mencionado anteriormente, e dele ser derivado através de outro indicador (Minimum Round Trip Delay), entendemos que seria mais eficiente calcular esta métrica apenas para a amostra de dados de 19 milhões de linhas, para não impactar inicialmente no tempo total de carga dos arquivos.

Este indicador define a diretividade da conexão entre um par de nós cujas localizações são conhecidas previamente. Sendo assim, valores do indicador próximos de um significam que o caminho entre os hosts é percorrido através de uma rota aproximadamente circular, enquanto que valores muito menores do que 1 significam que o caminho entre eles é bem indireto. A derivação da métrica depende do coeficiente de diretividade, e sua fórmula de cálculo é feita a partir de diversos cálculos mais complexos. Desse modo, utilizamos, para seu cálculo, a fórmula de Haversine,[24] que inclui a métrica Minimum Round Trip Delay e também a latitude e longitude, tanto do nó monitor quanto do nó que está sendo monitorado.

Ao incluirmos o cálculo da métrica Directivity sobre os dados da amostra e inserirmos os seus resultados, integrando-os na tabela fato, finalmente chegamos à amostra total, de aproximadamente 16 milhões de linhas, carregada no banco de dados do Cassandra.

Antes de configurar o domínio utilizado para realizar as consultas, foi selecionada a fonte de dados armazenados no Cassandra. Acessamos, então, a página inicial do JasperReports Server Pro, através do link,[25] para obter todos os privilégios de manipulação dos dados no modo de administrador. Em seguida, criamos uma fonte de dados Cassandra e selecionamos os parâmetros da conexão para se conectar com o Keyspace tblmedicao. Testamos a conexão e, em seguida, salvamos.

Ao tentar criar o domínio sobre a fonte de dados existente, descobrimos que as tabelas do Cassandra não foram identificadas. Com isso, após uma pesquisa sobre o comportamento das fontes de dados do Cassandra no Jaspersoft, notamos que as fontes de dados do Cassandra não podem ser utilizadas diretamente através de domínios. Para isso, a solução encontrada foi a criação de uma fonte de dados virtual, a partir da fonte de dados do Cassandra,[26] para possibilitar a realização das análises sobre as consultas.

4.4 CONSULTAS ANALÍTICAS

Uma vez estabelecida a conexão entre os dados definimos o domínio cujas consultas serão realizadas através do entendimento de todas as fontes de dados carregadas. Com base neste entendimento, elaboramos o esquema com todas as tabelas, cruzamentos, atributos, pré-filtros e campos calculados, que serão necessários para os resultados. Estas consultas contemplam todo o domínio do PingER, porém, como foi descrito anteriormente, outras

[24] http://en.wikipedia.org/wiki/Haversine_formula
[25] http://localhost:8080/jasperserver-pro
[26] http://community.jaspersoft.com/documentation/jasperreports-server-administration-guide/v56/cassandra-data-sources

fontes de dados importantes também foram adicionadas, de forma a oferecer um maior poder de combinações de dados, para enriquecer as consultas, e, sobretudo, reforçar as bases do estudo do desempenho.

De acordo com o esquema multidimensional, selecionamos as tabelas que serão utilizadas para elaborar as consultas, conforme as tabelas carregadas pelo Cassandra.

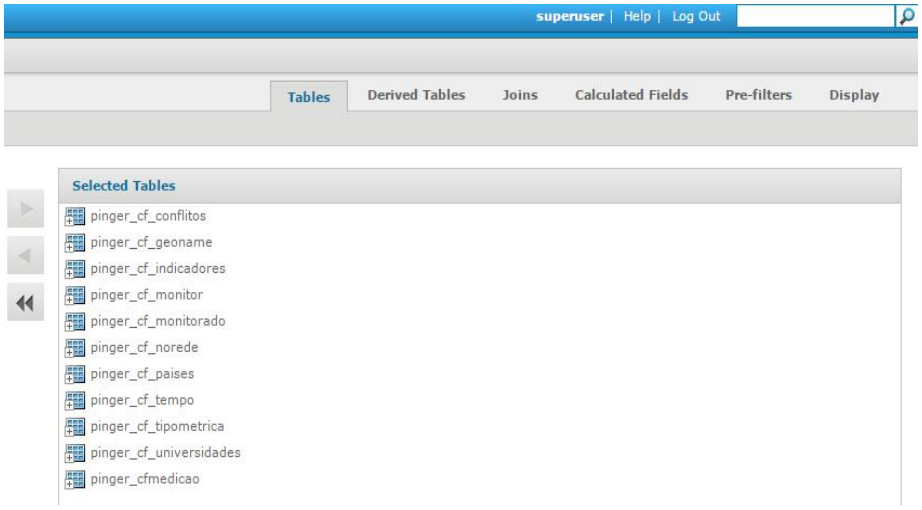

Figura 5 — Tabelas utilizadas para elaborar as consultas.

Para conectar todas as fontes relacionadas às dimensões do modelo, realizamos os cruzamentos entre as tabelas, de acordo com o modelo desenvolvido, e ajustamos o tipo de dados de cada campo, pois o Jaspersoft interpreta campos numéricos como medidas — com isso, diversos campos de chave primária e outros campos, que deveriam ser tratados como campos de atributos de tabelas dimensão, são representados incorretamente.

Aplicamos todas as configurações necessárias, e, a partir delas, podemos criar visualizações ad-hoc sobre o domínio criado, uma para cada consulta desejada. É importante avaliar o resultado de todas as consultas e comparar com o resultado na base de dados estruturada do MySQL, antes de realizar as métricas de avaliação necessárias.

Para auxiliar na análise dos resultados dos testes, as consultas foram classificadas de acordo com as características relacionadas aos operadores relacionais de ordenação e agregação, por exemplo.

Características	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Operador distinct			√				√	√		
Operador de Somatório										
Operador de Média			√				√	√	√	
Operador de Mínimo	√									
Operador de Máximo	√	√								
Mashup							√	√	√	√

4.4.1 Consultas analíticas realizadas

Nesta etapa, descrevemos todo o procedimento para a elaboração de cada uma das consultas analíticas que são usadas pelo projeto, assim como o detalhamento dos objetivos e o que elas medem, após a especificação de todos os parâmetros. Para cada execução, exibimos, como exemplo, apenas as dez primeiras ocorrências do resultado da consulta, considerando a amostra da base de dados completa. Isto foi feito para que o formato de apresentação de cada consulta se tornasse simplificado; por consequência, a tela com a evidência de cada consulta foi exibida completamente. Por último, confrontamos o resultado com o que foi apresentado pela base de dados relacional, para constatar a consistência e relevância dos dados pelo Cassandra, além da integridade de todos os dados retornados.

O desempenho das consultas pode ser medido a partir do log de execução disponível nos itens de configuração do Jaspersoft. Desse modo, todos os desvios de tradução das análises são suprimidos neste cálculo, fazendo com que o desempenho das consultas não seja prejudicado pela execução do software em si. É importante notar que o registro de execuções funciona tanto para a conexão no MySQL quanto para o Cassandra. Estas informações são armazenadas em cache na ferramenta, ou seja, enquanto o usuário estiver logado o resultado da execução referente a uma consulta específica vai estar armazenado no sistema. Com isso apenas o primeiro resultado de execução pode ser considerado, pois as outras execuções desta consulta já estarão calculadas.

Consulta 1: Verificar quais foram os nós que tiveram o mínimo de perda de pacotes de rede e maior vazão (Throughput), em um determinado período de tempo.

a) Para selecionar os nós, escolhemos o campo destino_name da tabela cf_norede, filtramos o campo instantiation_name com o valor ‘packet_loss’, e, depois, definimos o intervalo de tempo selecionando, o campo ano da tabela cf_tempo. Por

último, selecionamos o campo da métrica de rede (medida) da tabela cfmedicao e ordenamos de forma crescente.

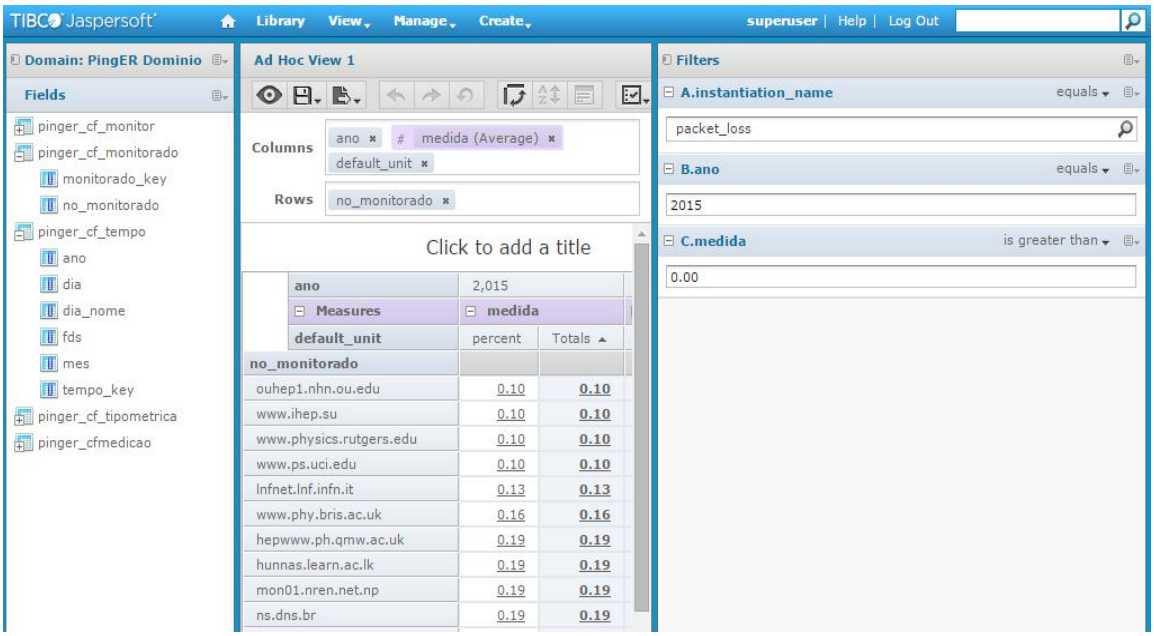

Figura 6 — Resultado da execução da consulta 1 (parte 1).

b) Para selecionar os nós, escolhemos o campo no_monitorado da tabela cf_monitorado, filtramos o campo instantiation_name com o valor ‘throughput’. Depois, definimos o intervalo de tempo, selecionando o campo ano e/ou mes e/ou ano da tabela cf_tempo. Por último, selecionamos o campo da métrica de rede (medida) da tabela cfmedicao e ordenamos de forma decrescente.

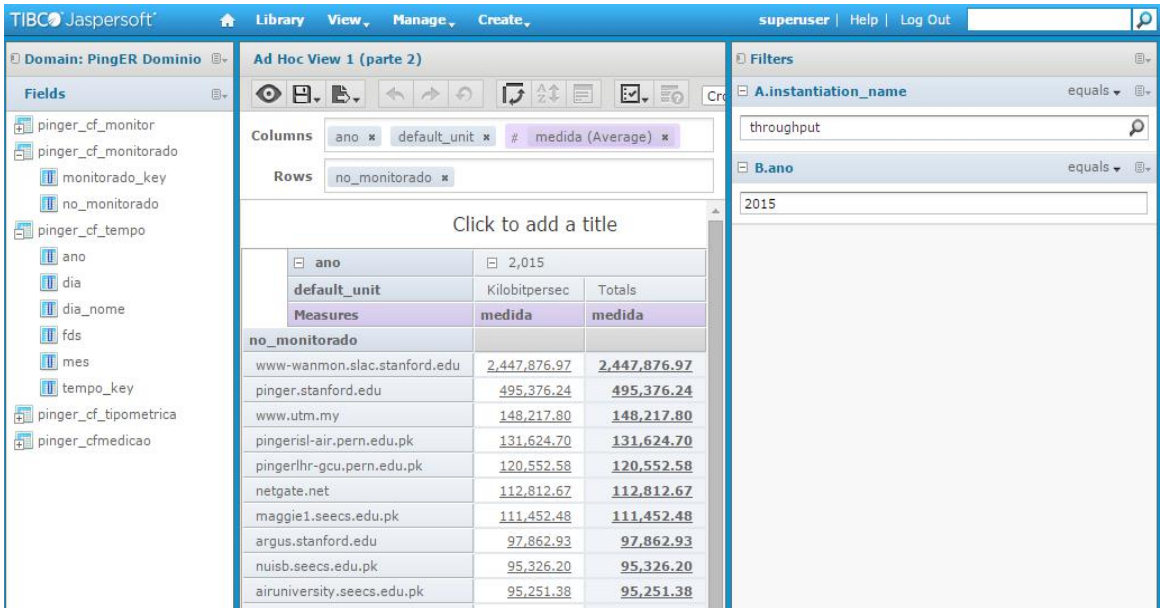

Figura 7 — Resultado da execução da consulta 1 (parte 2).

Consulta 2: Verificar quais foram os nós que tiveram o máximo de perda de pacotes de rede, em um determinado período de tempo.

Para selecionar os nós que tiveram o máximo de perda de pacotes de rede, escolhemos o campo no_monitorado da tabela cf_monitorado, filtramos o campo instantiation_name com o valor ‘packet_loss’, e, depois, filtramos o campo ano e/ou mes e/ou ano da tabela cf_tempo, definindo o intervalo de tempo. Por último, selecionamos o campo da métrica de rede (medida) da tabela cfmedicao e ordenamos de forma decrescente.

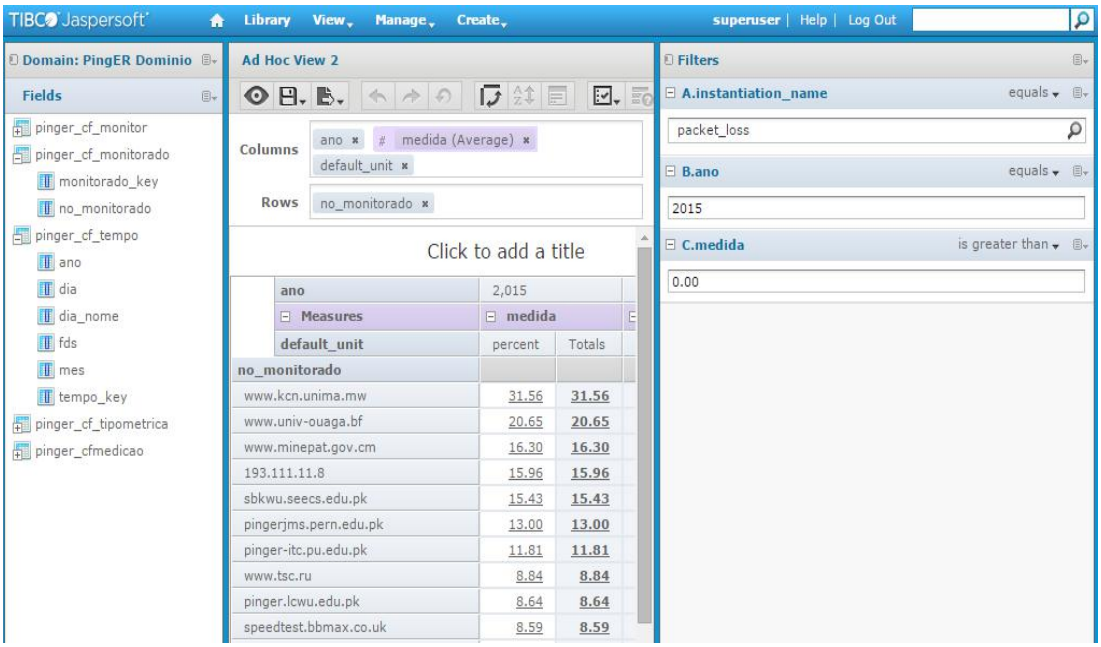

Figura 8 — Resultado da execução da consulta 2.

Consulta 3: Recuperar as métricas de rede e seus valores médios em um determinado intervalo de tempo, para avaliar a medida de qualidade de rede de nós em vários países ao longo de um período de tempo.

Para recuperar as métricas de rede e seus valores médios em um determinado intervalo de tempo, escolhemos os países selecionando o campo country da tabela cf_geoname, filtramos o campo ano e/ou mes e/ou ano da tabela cf_tempo, definindo o intervalo de tempo. Depois, selecionamos o campo instantiation_name da tabela cf_tipometrica, para representar os tipos métricas de rede. Por fim, selecionamos o campo da métrica de rede (medida) da tabela cfmedicao e ordenamos os países por ordem alfabética, de A a Z.

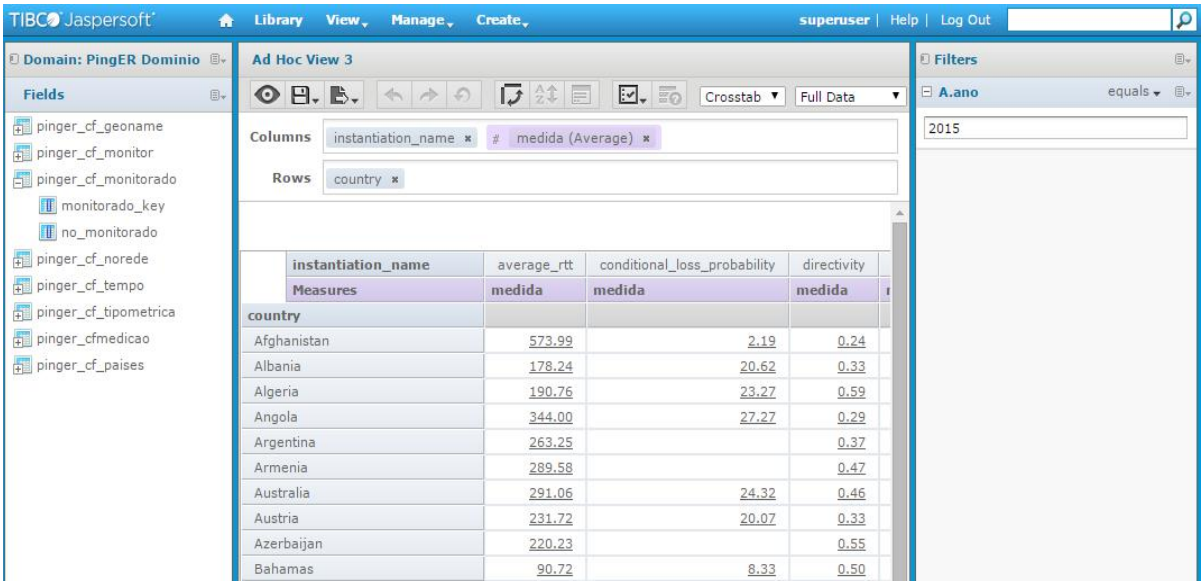

Figura 9 — Resultado da execução da consulta 3.

Consulta 4: Recuperar os primeiros 10 países que tiveram maiores perdas de pacotes de rede, em um determinado período de tempo.

Para recuperar os 10 primeiros países que tiveram maiores perdas de pacotes de rede, escolhemos os países selecionando o campo country da tabela cf_geoname, filtramos o campo ano e/ou mes e/ou ano da tabela cf_tempo, definindo o intervalo de tempo, e, depois, selecionamos o campo instantiation_name da tabela cf_tipometrica, para representar os tipos métricas de rede. Por fim, selecionamos o campo da métrica de rede (medida) da tabela cfmedicao e ordenamos os países em ordem decrescente de perda de pacote.

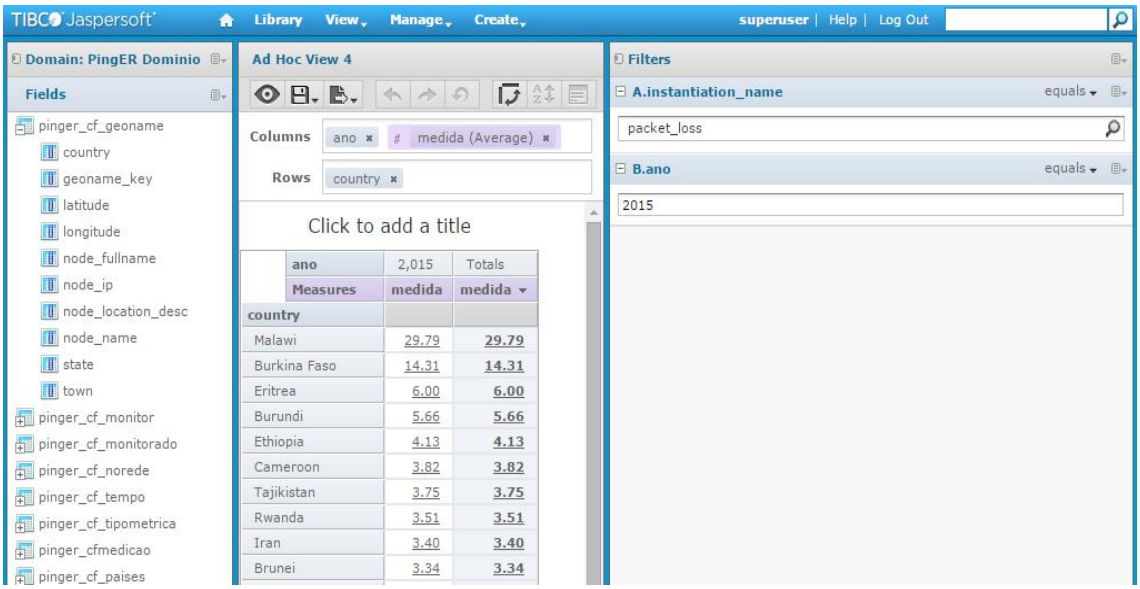

Figura 10 — Resultado da execução da consulta 4.

Consulta 5: Comparar a quantidade de nós inacessíveis (unreachability) nos diferentes continentes do planeta, em um determinado período de tempo, ordenando de forma decrescente.

Para comparar a quantidade de nós inacessíveis, escolhemos os continentes selecionando o campo continente da tabela cf_indicadores, bem como os países, selecionando o campo country da tabela cf_geoname, filtramos o campo ano e/ou mes e/ou ano da tabela cf_tempo, definindo o intervalo de tempo. Depois, selecionamos o campo instantiation_name da tabela cf_tipometrica, para representar os tipos de métricas de rede. E, finalmente, selecionamos o campo métrica de rede (medida) da tabela cfmedicao, e ordenamos em ordem decrescente, conforme solicitado.

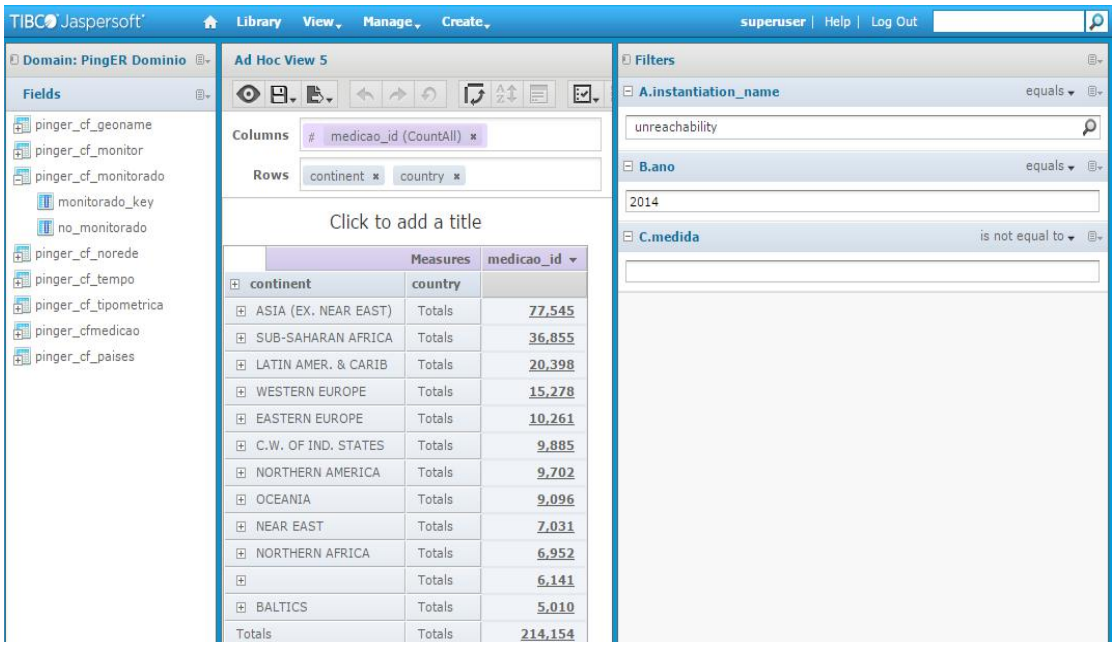

Figura 11 — Resultado da execução da consulta 5.

Consulta 6: Investigar a relação do continente com maior inacessibilidade dos nós com a situação dos seus países, em um período de tempo com guerras ou eventos de crises. Utilizar a DBPedia caso seja possível.

Conforme mencionado anteriormente, a DBPedia não forneceu informações suficientes para que a realização da consulta fosse possível. Sendo assim, utilizamos a base através do site da UCDP, com as informações necessárias.

Para investigar a relação do continente com maior inacessibilidade dos nós, escolhemos os continentes através do campo continente da tabela cf_indicadores e seus respectivos países, através do campo country da tabela cf_geoname, e seus territórios, aqui

representados por lado A e lado B (campos side_a e side_b) da tabela cf_geoname. Filtramos o campo ano e/ou mes e/ou ano da tabela cf_tempo, definindo o intervalo de tempo e o campo instantiation_name da tabela cf_tipometrica com o valor ‘unreachability’. Finalizando, selecionamos o campo da métrica de rede (medida) da tabela cfmedicao.

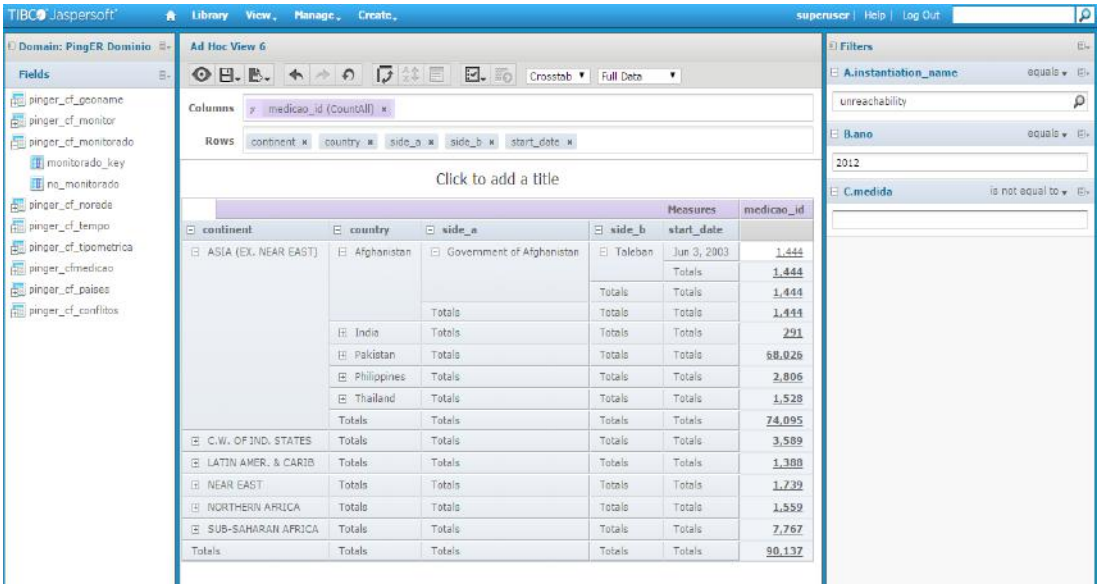

Figura 12 — Resultado da execução da consulta 6.

Consulta 7: Verificar qual a relação entre a qualidade da rede das universidades e a qualidade da universidade em um período de tempo. Para tal, deve se recuperar a métrica da universidade (número de alunos, fundos arrecadados, etc.), originada da DBPedia, bem como a métrica de rede (ex.: Throughput, Packet Loss, RTT, etc.), originada do laboratório SLAC.

Para verificar qual a relação entre a qualidade da rede das universidades e a qualidade das universidades em um período de tempo, escolhemos o nome da universidade selecionando o campo school_name da tabela universidades, bem como o campo school_facultysize, que define o tamanho da universidade, o campo school_numberofstudents, que vem a ser o número total de estudantes da universidade (graduados e em graduação), e o campo school_endowment, que representa o valor total de fundos arrecadados. Filtramos o campo instantiation_name com o tipo da métrica desejada e depois definimos o intervalo de tempo, selecionando o campo ano e/ou mes e/ou ano da tabela cftempo. Por último, selecionamos o campo da métrica de rede (medida) da tabela cfmedicao e ordenamos.

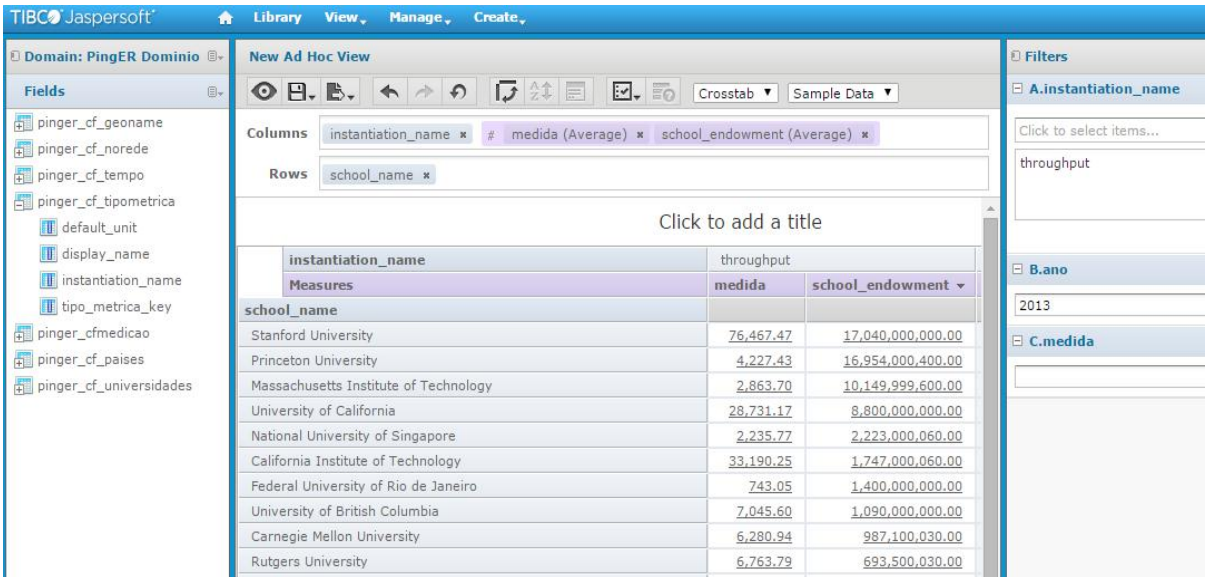

Figura 13 — Resultado da execução da consulta 7.

Consulta 8: Investigar a relação entre a qualidade de rede e o investimento dos países (% do PIB) em pesquisa e desenvolvimento tecnológico, originado do World Bank, ao longo dos anos. A versão dessa consulta é dividida em outras duas consultas (uma para cada endpoint) paralelas, que ao final se juntam (operação junção) para responder à questão descrita.

Para investigar a relação entre a qualidade de rede e o investimento dos países em pesquisa e desenvolvimento tecnológico ao longo dos anos, escolhemos os países selecionando o campo country da tabela cf_geoname, filtramos o campo ano e/ou mes e/ou ano definindo o intervalo de tempo e, depois, selecionamos o campo instantiation_name com o tipo da métrica desejada da tabela cf_tipometrica, para representar os tipos métricas de rede. Por fim, selecionamos o campo da métrica de rede (medida) da tabela cfmedicao e ordenamos em ordem alfabética por país, de A a Z.

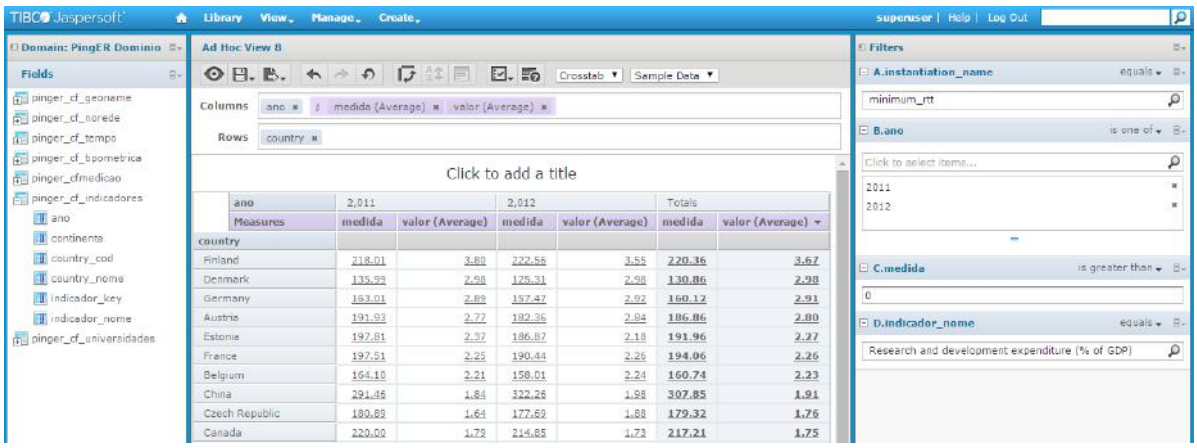

Figura 14 — Resultado da execução da consulta 8.

Consulta 9: Investigar a relação entre a qualidade de rede e o investimento dos países (% do PIB) em pesquisa e desenvolvimento tecnológico, originado do World Bank, ao longo dos anos. A versão dessa consulta é para testar como cada sistema se comporta ao lidar com a mesma consulta anterior de forma federada, ou seja, a mesma consulta é distribuída para a busca de informações em múltiplas fontes de dados, desde que se conheça previamente a localização, o vocabulário heterogêneo e como combinar os dados dessas fontes.

Entendemos que esta consulta não se aplica ao trabalho, visto que utilizamos apenas uma base de dados conjunta e integrada em um único modelo físico. Como a versão da consulta envolve junções de dados fornecidos por múltiplas fontes em diversos formatos, então não conseguiríamos realizar buscas em outras fontes de dados se não a principal.

Consulta 10: Analisar a qualidade de rede com a qualidade das universidades e o investimento dos países na pesquisa e desenvolvimento tecnológico em uma única consulta (Pinger + DBPedia + World Bank).

Para analisar a qualidade de rede com a qualidade das universidades e o investimento dos países na pesquisa e desenvolvimento tecnológico, escolhemos os países selecionando o campo country da tabela cf_geoname e suas respectivas universidades, através do campo school_name da tabela cf_universidades; como exemplo, filtramos o campo ano e/ou mes e/ou ano da tabela cftempo. Selecionamos o campo instantiation_name com o tipo da métrica desejada da tabela cf_tipometrica. Por fim, selecionamos o campo da métrica de rede (medida) da tabela cfmedicao, o campo da métrica da universidade (school_endowment) e o campo da métrica de investimento dos países (valor) que representa o percentual do PIB de investimento em pesquisa e desenvolvimento de cada país.

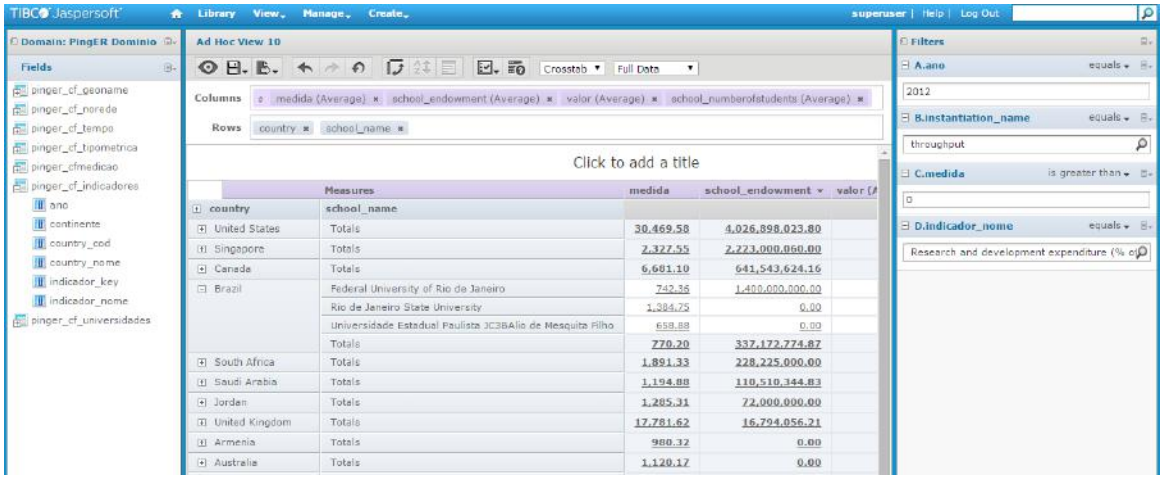

Figura 15 — Resultado da execução da consulta 10.

Normalmente, a maneira mais fácil de se configurar um cluster de Cassandra é através de várias máquinas. Como foi considerada apenas uma máquina neste trabalho, somente poderíamos fazer isso através de duas alternativas. Uma delas seria utilizar uma única máquina para definir várias instâncias (máquinas virtuais), executando todas simultaneamente. Mas essa prática pode exigir muitos requisitos de hardware, pois a máquina virtual demanda diversos recursos computacionais, apesar de cada vez mais os fabricantes desenvolverem tecnologias que possibilitam aos seus chips um trabalho aprimorado em soluções de máquinas virtuais.

A segunda maneira de se obter vários nós em um cluster do Cassandra, pode ser realizada executando-se várias vezes o servidor do Cassandra no mesmo sistema, armazenando os dados em vários lugares e monitorando portas diferentes em hosts distintos. Isso pode ser feito através de ajustes nos arquivos de configuração Cassandra para aplicar duas (ou mais) configurações diferentes e iniciá-las.

Optamos por utilizar a segunda opção para a execução do Cassandra, através de dois nós em uma única máquina. Em um exemplo hipotético, caso o desempenho do cluster Cassandra seja realmente um fator crítico, é aconselhável que os nós sejam separados em máquinas distintas, para que o Cassandra possa ser usado principalmente como fonte de armazenamento de dados, e não precisar lidar com as solicitações em tempo real. Em seguida, executando consultas em cada nó, é esperada a melhora do desempenho e utilização de hardware.

4.4.2 Comparação do desempenho nas consultas analíticas

4.4.2.1 Métricas de avaliação

Como forma de avaliar o desempenho das consultas, definimos os seguintes indicadores, que foram propostos no projeto MultiLOD de acordo com as seguintes métricas de avaliação utilizadas:

- Tempo de carga: tempo usado para carregar o conjunto de dados selecionado do arquivo fonte até o sistema de armazenamento. Isso inclui a construção das estruturas iniciais de índices e a geração de estatísticas sobre a otimização da consulta.
- Consultas por segundo (CpS): número de consultas de um tipo específico que foram respondidas pelo sistema em um segundo. A métrica é calculada através da divisão do número de consultas de um tipo específico pelo tempo de execução total

dessas consultas. Deve ser considerado o tamanho do banco de dados e o número de nós que estão sendo usados de forma concorrente.

Para o cálculo desta métrica, executamos as consultas de 1 a 10 (exceto a consulta 9 que não se aplica), duas do mesmo tipo por vez. Ao mensurar a métrica da consulta de número 1 devemos executar duas consultas número 1 e aguardar até o final de suas execuções para calcular a média de tempo de execução entre elas para somente então computar o resultado. Por exemplo, com o término da execução das consultas, calculamos esta medida através da divisão dessas duas consultas pelo tempo total de execução das duas consultas somadas. Considerou-se então o tamanho do banco de dados da amostra da tabela fato (cerca de 20 milhões de linhas) e a execução de duas consultas concorrentes de um tipo específico.

- Mixes de consultas por hora (MCpH): número de consultas misturadas aleatoriamente que foram respondidas completamente pelo sistema em uma hora. Deve ser considerado o tamanho do banco de dados e o número de nós que estão sendo usados de forma concorrente.

Para calcular esta métrica, sorteamos os números de 1 a 10 e executamos as consultas em uma ordem aleatória (exceto a consulta 9 que não se aplica). Assim que a execução de todo o conjunto de consultas for concluída, realizamos uma limpeza do cache das execuções, e repetimos novamente o processo entre todas as consultas, e assim por diante, até completar uma hora. A limpeza do cache foi feita a cada rodada, pois assim que executamos cada consulta, o seu resultado era armazenado e indexado, e, caso fôssemos executar novamente esta consulta, ela seria exibida quase que imediatamente. Também desconsideramos o tempo de limpeza do cache entre as rodadas de execução das consultas. Desse modo, incluímos na regra de cálculo apenas o tempo da execução das consultas em si e o tempo de *fetch* (tempo para retornar todas as linhas de cada resultado) de cada uma delas. E em relação ao escopo desta métrica, foi considerado apenas o banco de dados da amostra da tabela fato (aproximadamente 20 milhões de linhas) e a execução de todo o conjunto de consultas simultaneamente.

4.4.2.2 Metodologia

Antes de gerar os resultados sobre o desempenho de cada abordagem de sistema de armazenamento, deve ser testado se cada um dos resultados apresentados na execução de cada

consulta corresponde aos resultados esperados. Uma vez que a consulta esteja correta, os testes podem prosseguir de acordo com as métricas pré-estabelecidas.

Para cada tamanho gerado de banco de dados, devem ser aplicados os seguintes passos:

1. Carga do conjunto de dados: o desempenho da carga de dados considera que todos os dados estão em formato de família de colunas do Cassandra.
2. Banco de dados deve estar “frio”: o sistema de armazenamento deve ser desligado após a carga, a memória cache deve ser limpa e o sistema deve ser reiniciado. Com isso, considera-se que o banco de dados esteja em seu estado “frio”.
3. Aquecimento do banco de dados: para que o sistema de armazenamento atinja o seu estado normal de condições de trabalho, o conjunto de consultas precisa ser executado, de forma aleatória, por um determinado período de tempo a ser medido conforme o trabalho do laboratório.
4. Executar as consultas em múltiplos nós: avaliar a execução do conjunto de consultas misturadas e distribuídas pelo número de nós disponíveis. Inicialmente, com um nó, e, posteriormente, com dois nós.

4.4.2.3 Resultados dos experimentos

Tempo de carga

MySQL

- Tempo de carga dos arquivos do PingER no banco de dados relacional, considerando todo o conjunto de transformações no Kettle: cerca de 35 horas.
- Tempo de carga de todo o banco relacional do PingER na tabela fato e de todas as tabelas de dimensões auxiliares no modelo multidimensional: cerca de 10 horas.

Cassandra

- Tempo de carga da base de dados da tabela fato completa: aproximadamente 9 horas.
- Tempo de carga da base de dados da amostra de cerca de 16 milhões de linhas: aproximadamente 30 minutos.
- Tabela fato: aproximadamente 30 minutos (vide abaixo).

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	carrega-fato-medicao-amostra	0	0	16197978	16197978	0	0	0	0	Finished	29mn 37s	9,116
2	Cassandra Output	0	16197978	0	0	0	0	0	0	Finished	29mn 38s	9,109

Figura 16 — Tempo de carga da tabela fato.

- Tabelas dimensão: aproximadamente 20 segundos (vide abaixo).

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
9	carrega-universidades	0	0	264	264	0	0	0	0	Finished	0.09999999999999999s	2,467
3	carrega-tipo-metricas	0	0	11	11	0	0	0	0	Finished	0.09999999999999999s	139
7	carrega-tempo	0	0	40000	40000	0	0	0	0	Finished	15.1s	2,650
5	carrega-paises	0	0	227	227	0	0	0	0	Finished	0.09999999999999999s	2,270
10	carrega-no-rede	0	0	41854	41854	0	0	0	0	Finished	9.6s	4,372
8	carrega-monitorado	0	0	751	751	0	0	0	0	Finished	0.0s	83,444
4	carrega-monitor	0	0	69	69	0	0	0	0	Finished	0.09999999999999999s	734
2	carrega-indicadores	0	0	54808	54808	0	0	0	0	Finished	8.6s	6,377
6	carrega-geoname	0	0	1105	1105	0	0	0	0	Finished	0.09999999999999999s	15,563
1	carrega-conflitos	0	0	574	574	0	0	0	0	Finished	0.09999999999999999s	6,449
17	Cassandra Output 7	0	264	0	0	0	0	0	0	Finished	10.1s	26
15	Cassandra Output 5 3	0	751	0	0	0	0	0	0	Finished	13.0s	58
14	Cassandra Output 5 2	0	69	0	0	0	0	0	0	Finished	13.6s	5
13	Cassandra Output 5	0	40000	0	0	0	0	0	0	Finished	16.2s	2,475
18	Cassandra Output 4 4	0	227	0	0	0	0	0	0	Finished	9.1s	25
16	Cassandra Output 4 2	0	574	0	0	0	0	0	0	Finished	1.9s	307
12	Cassandra Output 4	0	54808	0	0	0	0	0	0	Finished	9.7999999999999999s	5,600
19	Cassandra Output 3	0	41854	0	0	0	0	0	0	Finished	10.7s	3,903
20	Cassandra Output 2	0	11	0	0	0	0	0	0	Finished	4.3999999999999995s	2
11	Cassandra Output	0	1105	0	0	0	0	0	0	Finished	7.8s	142

Figura 17 — Tempos de carga das tabelas dimensão no Cassandra.

- Consultas por segundo (CpS).

Consulta 1 (parte 1)				
Sistema de armazenamento	1.000.000	5.000.000	10.000.000	~ 20.000.000
Cassandra	0,104	0,051	0,009	0,004
MySQL	0,043	0,031	0,022	0,011
Consulta 1 (parte 2)				
Sistema de armazenamento	1.000.000	5.000.000	10.000.000	~ 20.000.000
Cassandra	0,103	0,044	0,009	0,004
MySQL	0,078	0,049	0,011	0,006
Consulta 2				
Sistema de armazenamento	1.000.000	5.000.000	10.000.000	~ 20.000.000
Cassandra	0,109	0,039	0,009	0,004
MySQL	0,119	0,053	0,023	0,012
Consulta 3				
Sistema de armazenamento	1.000.000	5.000.000	10.000.000	~ 20.000.000
Cassandra	0,109	0,037	0,008	0,005
MySQL	0,010	0,009	0,007	0,005
Consulta 4				
Sistema de armazenamento	1.000.000	5.000.000	10.000.000	~ 20.000.000
Cassandra	0,114	0,033	0,010	0,005
MySQL	0,009	0,008	0,007	0,006
Consulta 5				
Sistema de armazenamento	1.000.000	5.000.000	10.000.000	~ 20.000.000
Cassandra	0,104	0,040	0,009	0,004
MySQL	0,009	0,008	0,007	0,005
Consulta 6				
Sistema de armazenamento	1.000.000	5.000.000	10.000.000	~ 20.000.000
Cassandra	0,107	0,042	0,009	0,004
MySQL	0,024	0,020	0,017	0,013
Consulta 7				
Sistema de armazenamento	1.000.000	5.000.000	10.000.000	~ 20.000.000
Cassandra	0,113	0,035	0,009	0,004
MySQL	0,054	0,044	0,039	0,029
Consulta 8				
Sistema de armazenamento	1.000.000	5.000.000	10.000.000	~ 20.000.000
Cassandra	0,112	0,043	0,010	0,005
MySQL	0,005	0,005	0,004	0,003
Consulta 10				
Sistema de armazenamento	1.000.000	5.000.000	10.000.000	~ 20.000.000
Cassandra	0,113	0,038	0,010	0,005
MySQL	0,032	0,029	0,028	0,023

Figura 18 — Exemplo de execução — detalhamento da consulta 5.

- Mixes de consultas por hora (MCpH).

Execução de Consultas Aleatórias				
Sistema de armazenamento	1.000.000	5.000.000	10.000.000	~ 20.000.000
Cassandra	363	157	109	66
MySQL	115	93	81	59

4.5 ANÁLISE DOS RESULTADOS E CONSIDERAÇÕES GERAIS

Observamos que no Cassandra a execução das consultas utilizando um nó, em relação ao cluster de dois nós, se mostrou um pouco mais eficiente ao avaliarmos o custo de processamento por nó. Pelo fato de não ser possível utilizar outras máquinas de desempenho similar para a elaboração do cluster, então não conseguimos avaliar o particionamento do banco em diferentes máquinas.

É interessante notar que isto era esperado, pois apesar de cada nó acessar os dados que estão distribuídos em diferentes diretórios, ele utiliza o mesmo processamento de quando há um nó pelo fato dos dois nós estarem na mesma máquina.

Os fatores que devem ser utilizados como critérios de comparação, passam por diversas questões quantitativas de desempenho durante as etapas de ETL e execução das consultas, além de questões qualitativas relacionadas à facilidade de uso e escolha da linguagem de consulta. A comparação realizada neste trabalho foi baseada principalmente em quatro critérios: tempo de carga e desempenho por consulta executada, escalabilidade do sistema, consistência dos dados e usabilidade.

Utilizamos a premissa da escolha do mesmo modelo lógico e plataforma computacional para a execução dos testes comparativos do Cassandra e MySQL. As consultas selecionadas para os testes foram apresentadas e caracterizadas no item anterior, sendo assim, elas simulam operações analíticas de um usuário comum de um sistema de Business Intelligence a partir de diversas tabelas e cruzamentos da base de dados.

Em operações de escrita, ou seja, no momento da carga dos dados, verificamos que o Cassandra completou a alocação de todas as amostras de modo visivelmente mais eficiente que o armazenamento no MySQL. Observamos que na execução utilizando uma amostra de um milhão de linhas, o Cassandra possui um desempenho superior ao do MySQL, para todas as consultas. Tanto em relação ao conjunto de 5, quanto ao de 10 milhões de linhas, percebemos uma ligeira vantagem do desempenho das consultas no Cassandra, ocasionado pela quantidade do número de linhas da base que foram processadas; enquanto que no MySQL houve uma queda sensível de desempenho de forma linear.

Por último, foi selecionada a amostra completa, com aproximadamente 20 milhões de linhas. Percebemos, na maioria das consultas, uma diferença a favor do Cassandra para quase todas as consultas. Acreditamos que isto se deve principalmente pela utilização de um modelo de dados simples e denormalizado. Além disso, o banco de dados em ambos os casos é estático e isto favorece o Cassandra para que possa apenas executar as consultas durante a execução dos testes.

Apesar do Cassandra não utilizar todo o poder computacional proporcionado por suas propriedades de particionamento e escalabilidade é importante notar que mesmo pela ausência do aproveitamento de sua arquitetura distribuída ainda são observadas as vantagens do seu modelo de dados bastante flexível. O MySQL (que utiliza o mecanismo de armazenamento InnoDB) usa uma estrutura tradicional de árvore B+. Isto significa que podem haver múltiplos acessos aleatórios em disco enquanto realizam operações de escrita. Entretanto, o Cassandra usa uma estrutura de dados LSM (log-structured merge trees) para uma formatação on-disk, o que significa que todas as escritas são feitas sequencialmente (o banco de dados é o log). Independentemente do tamanho dos dados, isto implica em uma baixa latência de escritas e operações de leitura aleatórias, que podem ocorrer a partir da memória, ou então através de uma rápida busca em disco.

Já em relação ao cluster de dois nós, o Cassandra é capaz de atingir uma alta escalabilidade de escrita, ao particionar o Keyspace de um modo que cada máquina seja responsável apenas por uma parte das famílias de colunas de armazenamento. Isto implica em uma vazão de escritas maior, pois mais operações podem ser feitas paralelamente. Para fazer

o mesmo no MySQL, é necessário recorrer ao particionamento (tanto através do cluster no MySQL, quanto a nível de aplicação). E, para justificar o particionamento, deve ser considerado que o volume de dados é grande o suficiente.

É interessante notar que a vantagem a favor do MySQL foi observada apenas em consultas que possuem uma grande quantidade de agregações, e o MySQL é otimizado para o desempenho com operações JOIN, como era esperado. No Cassandra, a utilização de operações de junções entre os dados, ainda que pontuais, pode tornar-se bastante custosa. Apesar dos bancos NoSQL serem projetados para não depender das junções de forma nativa, ocorreram consultas que necessitaram de operações dessa natureza, ainda que não no mesmo nível que se faz necessário em sistemas relacionais comuns.

Pelo fato do Cassandra também não possuir instruções de ordenação como no SQL (cláusula Order By), utilizada em diversas consultas, é recomendado que os dados sejam inseridos no Cassandra de forma já ordenada. Neste trabalho, a tradução das instruções de ordenação dos dados foi feita internamente pela própria aplicação, antes da execução das consultas analíticas no Cassandra. Esta operação consome grande parte do tempo, contribuindo para a piora do desempenho, como foi observado.

Adicionalmente, não é permitido filtrar valores nulos no Cassandra da mesma forma que em uma base relacional, pois o Cassandra requer que todos os campos da cláusula WHERE sejam parte da chave primária. Desse modo, não será permitido que exista algum valor nulo em alguma parte da chave primária determinada. Para as consultas que exijam filtros com valores nulos, a aplicação novamente deve tratar internamente, e apresentar o resultado, para que possa ser executada a consulta no Cassandra através da conexão interna. E tal fato consome uma fatia de tempo, e contribui ainda mais para a queda de desempenho.

Um fator importante a ser considerado, e que não está relacionado ao desempenho, é que, para uma análise detalhada da expressão de uma consulta em CQL (Cassandra Query Language), existe uma dificuldade em traduzir a consulta analítica para uma linguagem de consulta, como é o caso do SQL para os SGBDs relacionais. Não existe, nesta abordagem do Cassandra, uma aproximação da simplicidade e expressividade oferecida pelo SQL. Sendo assim, perdem-se todas as funcionalidades oferecidas pela linguagem, tais como funções, rotinas, agregações, etc. Além disso, deixa-se de utilizar a mais simples restrição de integridade sobre o banco, o que pode tornar a aplicação mais pesada, ainda mais neste trabalho, em que utilizamos apenas uma máquina.

De forma complementar, devemos notar que a escalabilidade em alto grau sugerida pelo Cassandra se torna necessária apenas em bancos de grande porte, nos quais a alta

disponibilidade é imprescindível. Utilizar o Cassandra como solução para bancos de dados nos quais a disponibilidade não seja um fator imprescindível, ainda é uma abordagem discutível. Esta propriedade não é relevante neste trabalho, pois, apesar da importância dos dados utilizados, os mesmos não são críticos, visto que o impacto ocasionado pela perda deles é pequeno, ao levarmos em consideração que a frequência de atualização da tabela fato é mensal e a ausência de parte dos dados não traz graves consequências financeiras para seus usuários.

5 CONCLUSÃO

5.1 CONSIDERAÇÕES FINAIS

Neste trabalho de conclusão de curso, abordamos diversas técnicas de manipulação e análise de dados em um tipo de banco NoSQL (Cassandra), através do processo de publicação de dados de natureza multidimensional. Também abordamos uma aplicação prática das informações retiradas desses dados utilizando medidas de desempenho de rede (projeto PingER), operado pelo SLAC Stanford Linear Accelerator Laboratory.

Além disso, conseguimos relacionar estas informações com outros tipos de fontes de dados, para tornar o problema cada vez mais interessante do ponto de vista analítico em um cenário real. Cabe acrescentar que outros tipos de análises também podem ser feitos, conforme os resultados das consultas anteriores, ou de alguma outra necessidade mais específica.

No capítulo 2, descrevemos o domínio dos dados utilizados nos experimentos durante todo o processo de publicação de dados em um banco não-estruturado. E também reunimos todas as informações pertinentes para caracterizar os termos que englobam as nomenclaturas dos dados e seus possíveis relacionamentos.

No capítulo 3, definimos os conceitos de banco de dados através da análise comparativa entre o modelo relacional e não relacional, apontando suas vantagens e desvantagens, assim como em quais casos devemos utilizar cada um. Também apresentamos os diversos tipos de bancos não relacionais, e argumentamos todas as suas caraterísticas que consideramos mais relevantes. Por fim, abordamos o modelo de dados e a arquitetura do banco de dados escolhido para este trabalho de publicação de dados (Cassandra).

No capítulo 4, definimos o domínio do problema e construímos o modelo lógico de representação dos dados, para conectar todas as fases do projeto. Nesta etapa, o ETL aplica transformações sobre os dados, preparando para a carga na base do Cassandra. Concluída a carga no banco, os dados são armazenados e preparados para a ligação com o ambiente web de consultas. A partir deste ponto, é possível fazer as análises, consultas e visualizações. Realizamos, então, toda a modelagem da representação multidimensional, o processamento das consultas, e a medição das métricas de avaliação que oferecem parâmetros de desempenho do sistema como um todo. Também listamos todas as dificuldades enfrentadas ao longo do projeto, e as decisões tomadas para encontrar outros meios de atender aos objetivos propostos sem prejudicar as fases dos processos.

Resumindo, toda a evolução desta linha de estudo escolhida do projeto MultiLOD foi apresentada, desde a fase inicial, de concepção das teorias necessárias antes da abordagem prática, até o final, com as aplicações analíticas sobre os dados e o desempenho das configurações dos sistemas que compreendem os dados obtidos.

Portanto, o desenvolvimento desse trabalho permitiu aprofundar nossos estudos e experimentar na prática deferentes tipos de abordagens e tecnologias de tratamento de dados. Como forma complementar, o trabalho se mostrou uma alternativa aos experimentos do projeto PingER, em que o obstáculo gerado pelo grande volume de dados proporcionou um grande benefício em termos do acesso eficiente e prático às informações, através de consultas em uma interface bastante amigável.

Contudo, é importante salientar que o desempenho para realizar toda a aplicação em um conjunto de dados tão volumoso, deve ser considerado, e, para isso, deve ser considerada uma plataforma computacional cujo hardware atenda a diferentes requisitos. Caso outras máquinas pudessem ser incluídas no cluster, seria possível evidenciar as vantagens do Cassandra sobre o MySQL em um sistema distribuído. Como consequência, por considerarmos um alto consumo de dados gerado por esse projeto, foram encontradas diversas falhas de conexão e gargalos entre diversas transferências de dados na fase de ETL dos dados. Isto poderia ser corrigido pela padronização dos arquivos de origem antes do ETL. Porém, como todos os dados são de domínio público, entendemos que uma alteração do formato deles é praticamente inviável.

5.2 TRABALHOS FUTUROS

Entendemos que diversas pesquisas correlatas podem ser feitas para complementar os experimentos realizados durante este trabalho. Optamos por não utilizar um formato de disponibilização de consultas de forma gráfica pela dificuldade em representar as escalas em uma tela de exibição.

Como alternativa, poderíamos exibir os resultados em um mapa, e disponibilizá-los sob o formato de pontos que representam os nós ou países, e associá-los à suas informações. A ferramenta não oferece suporte a esse tipo de visualização interno da consulta. Porém, existe outra ferramenta do pacote (iReport) que é somente disponibilizada comercialmente. Acreditamos que esta ferramenta pode se tornar interessante em trabalhos futuros, para ser associada às consultas analíticas, mas o custo da licença deve ser planejado e devidamente considerado.

Para que possamos também comparar o desempenho do banco Cassandra em relação a outros tipos de bancos de dados, recomendamos a instalação do sistema de banco de dados HBase, já que ele também é um banco de dados colunar, pelo fato de armazenar dados em família de colunas nas tabelas. Assim, após sua instalação devemos aplicar as configurações e ajustes de carga no banco, e, em seguida, realizamos as mesmas consultas utilizadas neste projeto. Com isso, conseguiríamos avaliar as métricas de desempenho entre esses dois tipos de bancos.

Caso fosse considerado o uso de outras máquinas durante o desenvolvimento do projeto, seria recomendado utilizá-las como nós do Cassandra, e avaliar o comportamento em relação à consistência dos dados e tempo de execução das análises, em relação ao custo por processamento e particionamento.

E, por fim, mas não menos importante, poderíamos também incluir na pesquisa outras fontes de dados, para enriquecer ainda mais as análises e complementar os resultados do PingER, como, por exemplo, bases a partir de dados de indicadores sociais, socioeconômicos, geográficos, ambientais entre outras. Além de fornecer um direcionamento mais eficaz, as análises podem auxiliar na descoberta de correlações entre indicadores de diferentes áreas, através de uma visão mais ampla do mercado, sendo possível analisar o cenário como um todo e, até mesmo, conhecer melhor as diferentes ações causadas pelos agentes de transformação, para que possamos tomar uma decisão estratégica mais assertiva, com base em fatores externos.

REFERÊNCIAS

ATOL CONSEILS et Développements. **LIVRE BLANC - Les ETL Open Source**. Disponível em: http://www.atolcd.com. Acesso em: 13 jul. 2015.

BERNERS-LEE, T. **An attempt to give a high-level plan of the architecture of the Semantic WWW**. 1998. Disponível em: http://www.w3.org/DesignIssues/Semantic.html. Acesso em: 09 set. 2014.

CAMPOS, M. L. M. **Notas e apresentações de aula**. Instituto de Matemática, Departamento de Ciência da Computação. Universidade Federal do Rio de Janeiro, 2015.

CARDOSO, O. N. P. **Banco de dados**. Lavras: UFLA - FAEPE, 2003. 102 p.

CODD, E. F. A Relational Model of Data for Large Shared Data Banks. **Communications of the ACM**, New York, v. 13, n. 6, p. 377- 387, 1970.

COREY, M. et al. **Oracle 8i Data Warehouse**. Rio de Janeiro: Campus, 2001.

COTTRELL, L. **Internet End-to-end Performance Monitoring**. 2001. Disponível em: http://www-iepm.slac.stanford.edu. Acesso em: 13 set. 2014.

_____. **PingER Beacon Site.** Disponível em: http://www-iepm.slac.stanford.edu/pinger/beacon.html. Acesso em: 09 out. 2014.

_____. **Tutorial on Internet Monitoring and PingER at SLAC**. Disponível em: http://www.slac.stanford.edu/comp/net/wan-mon/tutorial.html. Acesso em: 03 set. 2014.

COTTRELL, L; MATTHEWS, W. The PingER Project: Active Internet Performance Monitoring for the HENP Community. **IEEE Communications Magazine**, New York, v. 38, n. 5, p. 130-136, 2000.

DEVELOPER WORKS, IBM. **Software, Open Source, SOA, Innovation, Open Standards, Trends**. Disponível em: https://www.ibm.com/developerworks/mydeveloperworks/blogs/ctaurion/entry/open-source_que_aconteceu_de_2004_para_hoje?lang=pt_br. Acesso em: 04 set. 2014.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. 4. ed. São Paulo: Addison-Wesley, 2005.

HILBERT, M. Big Data for Development: A Review of Promises and Challenges. **Development Policy Review**, London, v. 34, n. 1, p. 135-174, 2016.

INMON, W. H. **Building The Data Warehouse**. 4. ed. New York: John Wiley, 2005.

KANAZAWA, S. 'First, kill all the economists…'† : the insufficiency of microeconomics and the need for evolutionary psychology in the study of management. **Managerial and Decision Economics**, Chichester, v. 27, n. 2-3, p. 95-101, 2006.

KIMBALL, R.; CASERTA, J. Delivering Fact Table. In: KIMBALL, R.; CASERTA, J. **The Data Warehouse ETL Toolkit**: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. Indianopolis: Wiley, 2004. p. 209-254.

LEARN DATA MODELING. **Star Schema**: General information. 2012. Disponível em: http://www.learndatamodeling.com/star.php#.UogBUsTUDIF. Acesso em: 04 set. 2015.

LÓSCIO, B. F.; OLIVEIRA, H. R.; PONTES, J. C. S. NoSQL no desenvolvimento de aplicações Web colaborativas. In: SIMPÓSIO BRASILEIRO DE SISTEMAS COLABORATIVOS, 8., 2011, Paraty. **Anais [...]**. [S.l.: s. n.], 2011.

MACHADO, F. N. R. **Tecnologia e Projeto de Data Warehouse**: uma visão multidimensional. 4 ed. São Paulo: Érica, 2008.

MONTASSIER NETO, T. C. **Avaliação das ferramentas ETL Open-Source Talend e Kettle para projetos de data warehouse em empresas de pequeno porte**. 2012. 123 f. Trabalho de Conclusão de Curso (Bacharelado em Sistema de Informações) - União Metropolitana de Educação e Cultura, Lauro de Freitas, 2012.

PRITCHETT, Dan. **BASE**: an ACID Alternative. 2008. Disponível em: http://queue.acm.org/detail.cfm?id=1394128. Acesso em: 20 maio 2015.

RITA, L. S. **Magic Quadrant for Business Intelligence and Analytics Platforms**. 2014. Disponível em: http://guidedanalytics.co.uk/page1.php. Acesso em: 13 mar. 2015.

SHETH, A. P. Changing Focus on Interoperability in Information Systems: from system, syntax, structure to semantics. In: GOODCHILD, Michael… [et al.] (Ed.). **Interoperating Geographic Information Systems**. Boston: Springer, 1999. p. 5-29.

SOUZA, Renan Francisco Santos. **Processo de Publicação de Dados Abertos Interligados e Aplicação a Dados de Desempenho de Rede**. 2013. 135 f. Monografia (Graduação em Ciência da Computação) - Instituto de Matemática, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2013.

TALEND, **Open Integration Solutions**. Disponível em: http://www.talend.com. Acesso em: 04 out. 2014.

TIWARI, Shashank. **Professional NoSQL**. New York: John Wiley& Sons, 2011. 386 p.

WORLD WIDE WEB CONSORTIUM. **Internet Media Type registration, consistency of use**. 2002. Disponível em: http://www.w3.org/2001/tag/2002/0129-mime. Acesso em: 04 nov. 2015.

_____. **Big Data & Analytics Hub**. Disponível em: http://www.ibmbigdatahub.com/infographic/four-vs-big-data. Acesso em: 15 out. 2014a.

_____. **Big data showdown**: Cassandra vs. HBase. Disponível em: http://www.javaworld.com/article/2140805/big-data/big-data-showdown-cassandra-vs-hbase.html. Acesso em: 08 out. 2014b.

_____. **Cassandra Official Home Page**. Disponível em: http://cassandra.apache.org/. Acesso em: 07 out. 2014c.

_____. **Cassandra NoSQL Database**, Part 3: Clustering. Disponível em: https://msdn.microsoft.com/en-us/magazine/jj721601.aspx. Acesso em: 08 out. 2014d.

_____. **Data Integration - Kettle.** Disponível em: http://community.pentaho.com/projects/data-integration/. Acesso em: 08 out. 2014e.

_____. **PENTAHO, Open Source Business Intelligence.** Disponível em: http://kettle.pentaho.com. Acesso em: 09 out. 2014f.

_____. **SPARQL protocol for RDF**. 2008. Disponível em: http://www.w3.org/TR/rdf-sparql-protocol/. Acesso em: 10 dez. 2014.

_____. **Technical report development process**: Maturity level for work in progress. 2005. Disponível em: http://www.w3.org/2005/10/Process-20051014/tr.html#q73. Acesso em: 04 nov. 2014.

APÊNDICE A - CONSULTA SPARQL PARA GERAR O MAPA DE MÉTRICA DE REDE X MÉTRICA DE UNIVERSIDADE

```
PREFIX : <http://www-iepm.slac.stanford.edu/pinger/lod/resource#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX MU: <http://www-iepm.slac.stanford.edu/pinger/lod/ontology/MomentUnits.owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbp-rsrc: <http://dbpedia.org/resource/>
PREFIX dbp-prop: <http://dbpedia.org/property/>
PREFIX fb: <http://rdf.freebase.com/ns/>
PREFIX pos: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX gn: <http://sws.geonames.org/>
PREFIX dbp-owl: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX time: <http://www.w3.org/2006/time#>
PREFIX Units: <http://www-iepm.slac.stanford.edu/pinger/lod/ontology/Units.owl/#>
PREFIX MGC: <http://www-
iepm.slac.stanford.edu/pinger/lod/ontology/MomentGeneralConcepts.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX MD: <http://www-
iepm.slac.stanford.edu/pinger/lod/ontology/MomentDataV2.owl#>
PREFIX gn-ont: <http://www.geonames.org/ontology#>


SELECT DISTINCT ?SchoolURI ?SchoolName ?SchoolType ?SchoolNumberOfStudents
?Lat ?Long ?DBPediaLink
 ?SchoolFacultySize ?SchoolNumberOfGradStudents ?SchoolNumberOfUgradStudents
?SchoolEndowment
WHERE {

        ?SchoolURI MGC:SchoolPingerName ?PingerName .
        ?SchoolURI MGC:DBPediaLink ?DBPediaLink .
        OPTIONAL {
                ?SchoolURI MGC:SchoolName ?SchoolName .
        }
   OPTIONAL {
                ?SchoolURI pos:lat ?Lat .
                ?SchoolURI pos:long ?Long .
        }
        OPTIONAL {
        ?SchoolURI MGC:SchoolNumberOfStudents ?SchoolNumberOfStudents .
   }
        OPTIONAL {
        ?SchoolURI MGC:SchoolNumberOfStudents ?SchoolNumberOfStudents .
```

```
    }
    OPTIONAL {
      ?SchoolURI MGC:SchoolNumberOfStudents ?SchoolNumberOfStudents .
        }
    OPTIONAL {
            ?SchoolURI MGC:SchoolType ?SchoolType .
        }
        OPTIONAL {
            ?schooltown MGC:GeoCountry ?CountryName .
            ?schooltown gn-ont:name ?TownName .
        }
        OPTIONAL {
            ?SchoolURI MGC:SchoolFacultySize ?SchoolFacultySize .
        }
        OPTIONAL {
      ?SchoolURI MGC:SchoolNumberOfGradStudents ?SchoolNumberOfGradStudents .
        }
        OPTIONAL {
            ?SchoolURI MGC:SchoolNumberOfUgradStudents
?SchoolNumberOfUgradStudents .
        }
        OPTIONAL {
            ?SchoolURI MGC:SchoolEndowment ?SchoolEndowment .
        }
}
```

APÊNDICE B - SCRIPT DE CRIAÇÃO DAS TABELAS DO MODELO MULTIDIMENSIONAL NO MYSQL

```
CREATE TABLE `dim_sample` (
`sample_key` int(11) NULL DEFAULT NULL ,
`sample` int(11) NULL DEFAULT NULL
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci;

CREATE TABLE `dim_conflict` (
`conflict_key` int(11) NULL DEFAULT NULL ,
`conflict_id` tinytext CHARACTER SET utf8 COLLATE utf8_general_ci NULL ,
`location` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
`side_a` tinytext CHARACTER SET utf8 COLLATE utf8_general_ci NULL ,
`side_b` tinytext CHARACTER SET utf8 COLLATE utf8_general_ci NULL ,
`incompatibility` double NULL DEFAULT NULL ,
`territory_name` tinytext CHARACTER SET utf8 COLLATE utf8_general_ci NULL ,
`year` int(11) NULL DEFAULT NULL ,
`intensity_level` double NULL DEFAULT NULL ,
`cumulative_intensity` double NULL DEFAULT NULL ,
`type_of_conflict` double NULL DEFAULT NULL ,
`start_date` datetime NULL DEFAULT NULL ,
`end_date` tinytext CHARACTER SET utf8 COLLATE utf8_general_ci NULL
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci;

CREATE TABLE `dim_geoname` (
`geoname_key` int(11) NULL DEFAULT NULL ,
`latitude` decimal(20,2) NULL DEFAULT NULL ,
`longitude` decimal(20,2) NULL DEFAULT NULL ,
`country` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
`state` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
NULL ,
`town` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
NULL ,
`nodename` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
`nodeip` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
`node_fullname` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
```

```
`node_locationdesc` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci
NULL DEFAULT NULL
)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci;

CREATE TABLE `dim_indicator` (
`indicator_key` int(11) NULL DEFAULT NULL ,
`country_name` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
`country_cod` varchar(5) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
`continent` varchar(30) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
`indicator_name` varchar(80) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
`year` int(11) NULL DEFAULT NULL ,
`value` double NULL DEFAULT NULL)
ENGINE=MyISAM
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci;

CREATE TABLE `dim_monitoring` (
`monitoring_key` int(11) NULL DEFAULT NULL ,
`monitoring_node` tinytext CHARACTER SET utf8 COLLATE utf8_general_ci NULL )
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci;

CREATE TABLE `dim_remote` (
`remote_key` int(11) NULL DEFAULT NULL ,
`remote_node` tinytext CHARACTER SET utf8 COLLATE utf8_general_ci NULL )
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci;

CREATE TABLE `dim_node` (
`node_ key` int(11) NOT NULL AUTO_INCREMENT ,
`node_ last_update` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' ,
`source_name` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
`destination_name` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci
NULL DEFAULT NULL ,
`geoname_key` int(11) NULL DEFAULT NULL ,
PRIMARY KEY (`node_key`))
ENGINE=MyISAM
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci
AUTO_INCREMENT=41923;
```

```
CREATE TABLE `dim_country` (
`country` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
`continent` tinytext CHARACTER SET utf8 COLLATE utf8_general_ci NULL,
`population` double NULL DEFAULT NULL ,
`area` double NULL DEFAULT NULL )
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci;

CREATE TABLE `dim_time` (
`date_key` int(11) NULL DEFAULT NULL ,
`id_year` int(11) NULL DEFAULT NULL ,
`id_month` int(11) NULL DEFAULT NULL ,
`id_day_of_year` int(11) NULL DEFAULT NULL ,
`id_day_of_month` int(11) NULL DEFAULT NULL ,
`id_day_of_week` int(11) NULL DEFAULT NULL ,
`id_week_of_year` int(11) NULL DEFAULT NULL ,
`nm_day` varchar(30) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
`nm_month` varchar(30) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
`id_trimester` int(11) NULL DEFAULT NULL ,
`nm_trimester` varchar(2) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
`nm_trimester_of_year` varchar(32) CHARACTER SET utf8 COLLATE utf8_general_ci
NULL DEFAULT NULL ,
`in_weekend` char(1) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
`nr_days_on_month` int(11) NULL DEFAULT NULL ,
`nm_date` tinytext CHARACTER SET utf8 COLLATE utf8_general_ci NULL ,
`dt_date` datetime NULL DEFAULT NULL ,
`nm_week` varchar(32) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
`week_of_month_number` int(11) NULL DEFAULT NULL ,
`week_of_month_name` tinytext CHARACTER SET utf8 COLLATE utf8_general_ci
NULL ,
`year_sk` int(11) NULL DEFAULT NULL ,
`month_sk` int(11) NULL DEFAULT NULL ,
`quarter_sk` int(11) NULL DEFAULT NULL ,
`day_of_week_sort_name` varchar(60) CHARACTER SET utf8 COLLATE utf8_general_ci
NULL DEFAULT NULL ,
`year_sort_number` varchar(4) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL )
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci;
```

```
CREATE TABLE `dim_metric_type` (
`metric_type_key` int(11) NOT NULL ,
`metric_type_id` int(11) NULL DEFAULT NULL ,
`metric_type_last_update` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' ,
`display_name` varchar(30) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
`instantiation_name` varchar(30) CHARACTER SET utf8 COLLATE utf8_general_ci
NULL DEFAULT NULL ,
`default_unit` varchar(30) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
PRIMARY KEY (`metric_type_key `))
ENGINE=MyISAM
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci;

CREATE TABLE `dim_university` (
`school_uri` tinytext CHARACTER SET utf8 COLLATE utf8_general_ci NULL,
`school_name` tinytext CHARACTER SET utf8 COLLATE utf8_general_ci NULL ,
`school_type` tinytext CHARACTER SET utf8 COLLATE utf8_general_ci NULL ,
`latitude` double NULL DEFAULT NULL ,
`longitude` double NULL DEFAULT NULL ,
`dbpedialink` tinytext CHARACTER SET utf8 COLLATE utf8_general_ci NULL ,
`school_facultysize` double NULL DEFAULT NULL ,
`school_numberofstudents` double NULL DEFAULT NULL ,
`school_numberofgradstudents` double NULL DEFAULT NULL ,
`school_numberofugradstudents` double NULL DEFAULT NULL ,
`school_endowment` double NULL DEFAULT NULL ,
`nodename` tinytext CHARACTER SET utf8 COLLATE utf8_general_ci NULL)
ENGINE=InnoDB
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci;

CREATE TABLE `fact_network_measure` (
`measure_key` int(11) NOT NULL DEFAULT 0 ,
`time_key` int(11) NULL DEFAULT NULL ,
` node_key` int(11) NULL DEFAULT NULL ,
`geoname_key` int(11) NULL DEFAULT NULL ,
`measure` decimal(20,6) NULL DEFAULT NULL ,
`metric_type_key` varchar(20) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL ,
`measure_last_update` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' ,
PRIMARY KEY (`measure_key `))
ENGINE=MyISAM
DEFAULT CHARACTER SET=utf8 COLLATE=utf8_general_ci;
```

ANEXO A - DEFINIÇÃO DAS ETAPAS DO PROCESSO DE ETL

O Processo de ETL

O processo de ETL (extração, transformação e carga de dados, em inglês *Extract, Transform and Load*) é a fase mais crítica, complexa e demorada — ocupa boa parte do tempo em projetos DW (data warehouse), algo que varia entre 60% até 80% do tempo total gasto em um projeto (KIMBALL, 2004) — na construção de um DW, pois faz a aglutinação dos dados de múltiplas origens e entrega em um formato padronizado e consistente para ser carregado no DW. Normalmente, a origem dos dados provém dos sistemas transacionais, também conhecidos como OLTP (*Online Transactional Processing*); porém, outros tipos de fontes podem ser considerados, conforme a necessidade e disponibilidade das informações. Esses dados são entregues formatados em dimensões estruturadas, com o objetivo de facilitar aos sistemas de consultas, ou OLAP (*Online Analytical Processing*), disponibilizando, assim, a inteligência empresarial, para ser explorada pelos usuários finais (analistas, gerentes, diretoria). Entende-se por "captura e entrega" de dados um complexo trabalho de congregar fontes de dados de diversas origens com formatos e critérios variados em um único ambiente consistente e organizado. Uma implementação inconsistente ou equivocada no processo ETL tornará as informações armazenadas no DW não confiáveis para a tomada de decisão.

Logo após as etapas de levantamento das necessidades de informações, requisitos do projeto e definição da modelagem dos dados a serem apresentados no DW, o passo seguinte é identificar a origem dos dados, local onde são processados nos sistemas transacionais da organização. Os dados podem estar dispersos ou centralizados em um único local, assim como as características de plataformas de hardware e sistemas operacionais irão influenciar na complexidade do projeto.

Desta forma, inicia-se a etapa de ETL, que é uma das mais críticas no projeto de DW, pois é a etapa que envolve a movimentação de dados entre os ambientes transacionais e o ambiente de consulta analítica.

Etapas do Processo de ETL

Basicamente, as atividades que envolvem a etapa de ETL estão agrupadas em três grandes processos. O primeiro é representado pela (E) extração dos dados de origem, que pode envolver conexões com fontes de diversas origens, como os SGBDs, normalmente de sistemas transacionais (OLTP), estes também de diferentes fabricantes, planilhas eletrônicas, arquivos XML, arquivo texto, arquivos não estruturados, entre outros. É importante ter uma

definição concisa dos mapeamentos dos dados de origem que serão conduzidos para a *staging area* (área de transição), onde são convertidos para um único formato. A conversão se faz necessária devido a heterogeneidade existente nas informações oriundas desses sistemas, sendo essencial a conformação prévia para o tratamento adequado.

(T) Transformação dos dados. É o trabalho que concentra o maior esforço de análise; consequentemente, maior tempo é despendido, embora o grau de dificuldade dependa diretamente de fatores que são determinados pela qualidade dos dados de origem, das regras de negócios a serem aplicadas e da disponibilidade ou não de documentação atualizada das bases de dados de origem.

(L) Carga — L do inglês *Load* — requer do desenvolvedor o conhecimento das estratégias para alimentação das tabelas para o ambiente do DW, de acordo com o esquema multidimensional adotado. Além das estratégias de carregamento e atualização das tabelas de dimensão e fato, também inclui a periodicidade de carregamento dos dados. Algumas tabelas podem exigir carregamento a cada hora, outras diariamente ou semanalmente. Além disso, é imprescindível que cada etapa seja iniciada somente após o término com sucesso da etapa anterior (COREY, 2001).

ANEXO B - GLOSSÁRIO DOS DADOS DO PROJETO PINGER

Mean Opinion Score (MOS)

É a mais conhecida medida da qualidade de voz. É um método subjetivo de teste de qualidade.

Existem dois métodos de teste:

- teste de opinião de conversação;
- teste de opinião de escuta.

Os usuários de teste julgam a qualidade da transmissão da voz ao conversarem ou ao ouvirem amostras de voz. Após estes testes, os usuários irão qualificar a qualidade da voz de acordo com a seguinte escala:

MOS	Qualidade
5	Excelente
4	Bom
3	Razoável
2	Pobre
1	Mau

Directivity

Esta é uma métrica que identifica a diretividade da conexão entre 2 nós de localização conhecida. A sua unidade é Alpha. Alpha de valor 1 significa que o caminho entre os nós segue uma rota mais direta. Valores menores do que 1 indicam que o caminho vai ficando cada vez mais indireto (passando por outros nós).

Duplicate Packets

Mede a quantidade de respostas duplicadas ao comando de Ping. É um bom indicativo do ambiente da rede.

Packet Delay Variation (PDV)

Em redes de computadores, variação de atraso de pacotes (PDV) é a diferença do atraso ponto-a-ponto em um sentido único, entre pacotes selecionados em um fluxo com quaisquer pacotes perdidos sendo ignorada.

Round-Trip Delay Time (RTT)

RTT, no contexto de rede computacional, é conhecido como Ping Time, ou seja enviado um pacote de dados (comando de Ping), é o tempo para atingir o seu destino em outro nó ou site.

Maximum Round Trip Delay

É o valar máximo de uma amostra de RTT medida num intervalo determinado de tempo, normalmente de 1 hora.

Minimum Packet Loss

Perda do Pacote ocorre quando um ou mais pacotes de dados através da rede falham em atingir o seu destino. Packet Loss é fortemente associado à Qualidade do Serviço (QoS), e está relacionada com a unidade de medida Erlang.

TCP Throughput

Medida da taxa de transferência ou rendimento (throughput) utilizando TCP como protocolo de transporte. A unidade utilizada no PingER e kbits/s.

Unreachability

Taxa que identifica se um host não está sendo alcançado.

Se nenhuma resposta é recebida para 30 minutos de comandos de então o host remoto (nó) é considerado inalcançável (Unreach).

Zero Packet Loss Frequency

Também conhecido como **Quiescent** Network Frequency. Quando temos um Packet Loss zero, referimos à rede como sendo quiescente, ou não ocupada.

Podemos então medir a Zero Packet Loss Frequency como com que frequência o caminho de rede ficou quiescente. Um percentual alto é indicativo de uma boa rede (quiescente ou não muito sobrecarregada).

InterQuartile Range (IQR)

Em estatística, Amplitude Interquartil é dada pela diferença entre o quartil superior e o quartil inferior, ou seja, Q3 – Q1.

Exemplo: na seguinte série {0,0,0,0,1,1,1,1,1,1,2,2,2,2,2,2,3,3,3} o primeiro quartil é representado pelo quinto valor, ou seja, 1 e o terceiro quartil é representado pelo décimo-quinto valor, ou seja 2. A amplitude inter-quartis é 2 – 1 = 1.

Out of Order Packets

Para cada amostra de 10 pacotes, é observado se as sequências de respostas foram recebidas na mesma ordem em que foram enviadas. Para cada diferente, a amostra é marcada com a quantidade de diferenças observadas. Para um dado intervalo (por exemplo um mês), o valor reportado é a fração das amostras que foram marcadas como fora de ordem. Desde que pacotes de dados enviados (pings) são emitidos no intervalo de 1 segundo, é esperado que a fração fora de ordem desses seja muito pequena, caso contrário deve ser investigado o motivo.

Ping Unpredictability

A métrica Unpredictability é derivada do cálculo baseado na variabilidade do Packet Loss e do RTT (Round Trip Time). Se um caminho da rede perde muito mais pacotes num intervalo de tempo do que em outro, ou se o RTT flutua fortemente entre duas medidas, então o cálculo fornece um valor de Unpredictability mais alto para esse caminho. Essa métrica é uma inovação útil do projeto do PingER (GÜNTHER, 2002).

ANEXO C - GLOSSÁRIO DOS TERMOS UTILIZADOS NA MODELAGEM

Tabela Fato (fact_network_measure):

- measure_key: chave de identificação de cada medição executada.
- time_key: identificação do dia, mês e ano que a medição ocorreu.
- node_key: identifica o par de nós (monitor e monitorado) envolvidos em uma medição.
- geoname_key: define através de uma chave identificadora a localização única do nó que foi monitorado.
- measure: valor da medição de rede entre um par de nós em um determinado momento para um tipo de métrica.
- metric_type_key: identifica a qual tipo de métrica a medição está relacionada.
- medicao_last_update: registro da data completa que a medição foi salva na tabela.

Tabelas Dimensão:

a) dim_time

- time_key: chave de identificação da data que a medição ocorreu.
- id_year: caracteriza o ano em que a medição foi realizada
- id_month: caracteriza o mês em que a medição foi realizada
- id_day_of_year: identifica o dia do ano em que houve a medição em relação ao total de dias no ano.
- id_day_of_month: identifica o dia do mês em que houve a medição em relação ao total de dias no mês.
- id_day_of_week: registra o dia da semana de domingo a sábado (1 a 7).
- id_week_of_year: é a identificação da semana do ano no momento que a medição ocorreu.
- nm_day: : representa o dia da medição através do nome completo (segunda-feira, terça-feira, quarta-feira, quinta-feira, sexta-feira, sábado e domingo).
- nm_month: é utilizada para registrar o nome completo do mês da medição (Janeiro a Dezembro).
- in_weekend: variável utilizada para caracterizar se a data pertence a um final de semana (Y) ou não (N).
- nm_date: representa uma data no formato dia/mês/ano que pode variar entre 01/01/1998 e 08/07/2017.

b) dim_metric_type

- metric_type_key: representa a identificação de cada um dos tipos de métrica utilizados.
- metric_type _id:
- metric_type _last_update: data de registro do tipo da métrica.
- display_name: descreve o nome da métrica completo por extenso.
- instantiation_name: representa o nome da métrica conforme a nomenclatura dos arquivos de origem.

c) dim_node

- node_key: representa a chave de identificação entre um par de nós (monitor e monitorado) específicos.
- node_last_update: é o registro da data completa em que um par de nós foi salvo na tabela.
- source_name: caracteriza um nó de origem de onde parte a medição, através da descrição de seu domínio.
- destination_name: caracteriza um nó de destino no qual a medição é recebida, através da descrição de seu domínio.
- geoname_key: caracteriza a localização (latitude e longitude) do nó que foi monitorado, caso ela seja encontrada.

d) dim_monitoring

- monitoring_key: chave identificadora de cada nó monitor do PingER.
- monitoring_node: para cada chave identificadora é definido um site monitor específico.

e) dim_remote

- remote_key chave identificadora de cada nó que pode ser monitorado por um nó monitor do PingER.
- monitoring_node: para cada chave identificadora é definido um site remoto que pode específico.

f) dim_geoname

- geoname_key: define através de uma chave identificadora a localização única de um nó.
- latitude: representa a latitude de um nó.
- longitude: representa a longitude de um nó.
- country: refere-se ao país de um nó específico, caso seja possível localizá-lo.
- state: caracteriza o estado referente a um nó determinado, caso seja possível localizá-lo.
- town: caracteriza a cidade de origem para um nó específico, caso seja possível localizá-la.
- nodename: define um site remoto que está diretamente relacionado com a geoname_key, ou seja, contêm a mesma informação que o campo monitoring_node.

g) dim_country

- country: descreve o nome de um país entre uma lista de diversos países do mundo.
- continente: caracteriza o continente no qual o país determinado pertence.
- population: define a quantidade de pessoas que pertencem a um país, desde a última atualização da tabela.
- área: expressa a área total que um país ocupa atualmente.

h) dim_indicator

- indicator_key: chave que identifica cada um dos indicadores de cunho social, econômico ou político para um país e ano específicos.
- country_name: nome de um país cujo indicador está relacionado.
- country_cod: representa o código de um país, através de 3 letras, cujo indicador está relacionado.
- continente: define o continente no qual o país determinado pertence.
- indicator_name: seleciona um tipo de indicador que é medido de forma anual para um país específico.
- year: representa o ano em que o indicador vai ser medido para um país.
- value: caracteriza a medição propriamente dita do indicador selecionado.

i) dim_university

- school_uri: chave de identificação do endereço de uma faculdade na Internet.
- school_name: nome completo da faculdade.
- school_type: classificação do tipo de faculdade, ou seja, caracteriza se a faculdade é pública, privada, estadual, entre outras classificações.
- latitude: representa a latitude em que uma faculdade está localizada, caso a informação seja encontrada.
- longitude: latitude: representa a latitude em que uma faculdade está localizada, caso a informação seja encontrada.
- dbpedialink: link com as diversas informações relevantes sobre a faculdade no DBPedia.
- school_facultysize: quantidade total de membros que pertencem ao staff da faculdade em si.
- school_numberofstudents: quantidade total de estudantes que pertencem à faculdade.
- school_numberofgradstudents: quantidade total de estudantes que estão graduados.
- school_numberofugradstudents: número de estudantes que ainda está cursando algum curso de graduação na faculdade.
- school_endowment: quantidade de recursos alocados através de doações governamentais ou alguma outra instituição para possibilitarem investimentos na faculdade em questão.
- nodename: define um site remoto que está diretamente relacionado com a faculdade, caso ela possa ser associada a um nó remoto monitorado.

j) dim_conflict

- conflict_key: chave de identificação para cada conflito ocorrido em um período determinado.
- location:nome(s) do País ou países cujo governo reivindica em primeiro lugar o motivo principal da disputa em questão.
- side_a: uma das partes envolvidas no conflito, geralmente a parte que motivou o conflito, caso seja um conflito interno será um grupo interno dentro do País caso contrário ele será representado por um País.

- side_b: uma das partes envolvidas no conflito, geralmente a parte que foi motivada a entrar no conflito, caso seja um conflito interno será um grupo interno dentro do País caso contrário ele será representado por um País.
- incompatibility: número que representa a categoria no qual o conflito pertence, podendo ser dividida de 1 a 3. Sendo 1 igual a um conflito governamental, 2 referente a um conflito territorial e 3 caso seja de cunho governamental e territorial ao mesmo tempo.
- territory_name: representa em qual território o conflito realmente ocorreu levando em consideração que o conflito foi ocasionado por motivos territoriais.
- year: caracteriza um ano determinado dentro do período que o conflito ocorreu. Caso o conflito perdure por vários anos então para cada ano do conflito ele será registrado separadamente.
- intensity_level: nível de intensidade de um conflito (de 1 a 2) registrado em um ano. Caso o conflito envolva um número menor do que 1000 mortes este valor será igual a um (1), caso seja igual ou superior será igual a dois (2).
- cumulative_intensity: representa a intensidade total do conflito, considerando-se todo o seu histórico e não apenas um ano específico.
- type_of_conflict: classifica os tipos de conflito em quatro diferentes tipos, extra sistêmico (entre o governo de um País e um grupo não governamental fora do seu território), interestadual (entre os governos de dois ou mais países), interno (entre o governo e um ou mais grupos de oposição, sem intervenção de outro País) ou internacionalizado (entre o governo de um País e um ou mais grupos de oposição do território aonde o conflito ocorre, com intervenção de um ou mais países). Estas categorias são registradas através de números, de 1 a 4 respectivamente.
- start_date: data de início do conflito.
- end_date: data de término do conflito, caso ele não tenha terminado o campo permanece vazio.

ANEXO D - SCRIPT DE CRIAÇÃO DO MODELO FÍSICO NO CASSANDRA

```
CREATE KEYSPACE tbl_network WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : 3};
USE tbl_medicao;
CREATE COLUMN FAMILY cf_fact_network_measure;
CREATE COLUMN FAMILY cf_time;
CREATE COLUMN FAMILY cf_metric_type;
CREATE COLUMN FAMILY cf_node;
CREATE COLUMN FAMILY cf_monitoring;
CREATE COLUMN FAMILY cf_remote;
CREATE COLUMN FAMILY cf_geoname;
CREATE COLUMN FAMILY cf_country;
CREATE COLUMN FAMILY cf_indicator;
CREATE COLUMN FAMILY cf_conflict;
```