Avaliação de Desempenho de Sistemas Relacionais para Armazenamento de dados RDF

William Pereira[1], **Tiago Heinrich**[1], **Rebeca Schroeder**[1]

[1]Departamento de Ciências da Computação – Universidade do Estado de Santa Catarina
Centro de Ciências Tecnológicas – 89.219-710 – Joinville – SC – Brasil

{willpereira,tiagoheinrich1995}@gmail.com, rebeca.schroeder@udesc.br

Abstract. *Nowadays, an increasing amount of data are becoming available in RDF format. In order to provide suitable database systems to manage RDF data, there are approaches adapting relational database systems to process RDF, or using NoSQL systems to deal with the large volume of some RDF datasets. This paper presents an experimental study which compares two models of relational systems in this context. The first model, named triple-store, is a simple solution that converts an RDF dataset to a relation. The second model is based on the RDF structure to provide a more appropriate relation schema. These models are represented in the experiments by the systems Jena-TDB Fuseki and ntSQL, respectively. The results reported that the relational schema applied by ntSQL provides better response time in SPARQL queries than compared to the schema of a triple-store.*

Resumo. *Atualmente, observa-se um volume crescente de dados sendo publicado no formato RDF. Para prover sistemas de bancos de dados adequados para gerenciar este tipo de dados, as soluções partem de adaptações dos SGBDs relacionais para suportar RDF, assim como sistemas NoSQL para suprir a demanda do volume de algumas bases de dados deste tipo. Este artigo apresenta um estudo experimental que compara dois modelos de sistemas relacionais neste contexto. O primeiro destes modelos, conhecido como triple-store, é uma solução simples que transforma uma fonte RDF em uma tabela. O segundo modelo utiliza noções da estrutura RDF para propor um esquema relacional mais robusto. Estes modelos são representados nos experimentos pelos sistemas Jena-TDB Fuseki e o ntSQL, respectivamente. Os resultados obtidos demonstram que o uso de um esquema relacional mais robusto, como o obtido pelo ntSQL, confere um melhor desempenho em consultas SPARQL submetidas a estes repositórios.*

1. Introdução

RDF (*Resource Description Framework*) é hoje o modelo padrão para representação de dados na Web (Apache Jena 2016a). A partir da estrutura de identificadores da Web, o RDF permite definir semanticamente os relacionamentos entre dados através do uso de URIs (*Universal Resource Identifier*). Dados RDF são definidos através de triplas, compostas de um sujeito, um objeto e relacionados por um predicado. Com a padronização, diversas fontes passaram a produzir dados em RDF continuamente. Como resposta a esta realidade, bases de dados de diferentes tamanhos e características estão disponíveis neste

formato, em especial, na Web. Algumas destas fontes de dados estão disponíveis no site *Large Triple Stores*[1].

O crescente volume de dados no formato RDF criou a necessidade por sistemas de bancos de dados capazes de gerenciar dados neste modelo. Em geral, sistemas NoSQL ou repositórios de grande escala têm sido adotados para o armazenamento de fontes RDF em virtude do elevado volume de dados que algumas fontes apresentam (Zeng et al. 2013). Entretanto, o uso de sistemas deste tipo acrescentam uma maior complexidade ao desenvolvimento de aplicações pois, em geral, estes sistemas não apresentam algumas das características de um Sistema Gerenciador de Banco de Dados (SGBD) relacional (Arnaut et al. 2011). Desta forma, para repositórios com volumes de dados de dimensões convencionais o uso de SGBDs relacionais tem sido preferidos por alguns trabalhos.

No contexto de SGBDs Relacionais que suportam RDF, existem dois tipos de modelo aplicados. O primeiro, bastante simples, é conhecido como *triple-store*. Um *triple-store* define um banco RDF através de uma única relação composta pelos campos sujeito, predicado e objeto. Nesta relação, as tuplas correspondem a triplas RDF. Exemplos de sistemas que empregam este modelo são o Jena TDB (Apache Jena 2016a) e RDF-3X (Neumann and Weikum 2010). O segundo modelo corresponde à utilização de conhecimentos sobre a estrutura RDF para definição de um esquema relacional baseada em tipos. Neste caso, cada tipo representa uma relação da base de dados. O sistema ntSQL (Bayer et al. 2014) é um dos sistemas que emprega este modelo.

Existem diversos trabalhos, como J. Huang, D. Abadi 2011, Ravindra et al. 2011 e Papailiou et al. 2014, que provam a ineficiência de *triple-store*, especialmente no desempenho de consultas RDF mais complexas. Esta ineficiência é devida ao tamanho atingido pela relação do *triple-store*, e do custo das auto-junções necessárias para o desempenho de consultas RDF complexas. Segundo Bayer et al. 2014, a ausência de conhecimento sobre a estrutura dos dados RDF faz com que diversas soluções utilizem SGBDs relacionais em sua forma mais simples através de um *triple-store*. Entretanto, a ausência de um esquema acaba por sub-utilizar o modelo relacional ao criar relações baseadas apenas nas composições de triplas. Conforme apontado por Pham 2013, apesar de RDF constituir um modelo livre de esquema, é possível a extração de estruturas de dados a partir de diversas fontes RDF. Esta possibilidade viabiliza uma representação RDF mais adequada em SGBDs relacionais, bem como um melhor desempenho em consultas. Neste contexto, este trabalho visa investigar a diferença no desempenho em consultas de um *triple-store* com um banco equivalente que aplique o segundo modelo através do ntSQL.

Este artigo tem por objetivo apresentar um estudo experimental que compara o desempenho em consultas utilizando os dois tipos de modelos aplicados por SGBDs Relacionais para RDF. Este estudo compara o *triple-store* Jena TDB Fuseki (Apache Jena 2016a) como representante do primeiro modelo, e o ntSQL (Bayer et al. 2014) como representante do segundo modelo. Os experimentos foram baseados no *Belin SPARQL Benchmark* (Bizer and Schultz 2009) através de consultas SPARQL e seu gerador de bases de dados. Os resultados obtidos apontam um ganho significativo no desempenho de consultas RDF utilizando o modelo empregado pelo ntSQL, comparado ao *triple-store*.

[1] https://www.w3.org/wiki/LargeTripleStores

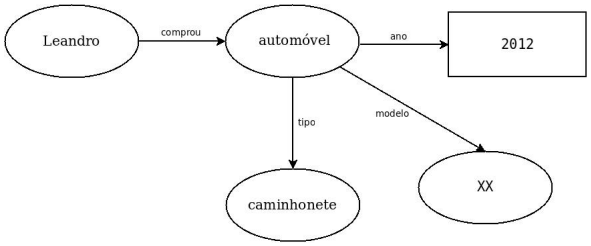


Figura 1. Representação gráfica de dados RDF.

Este trabalho está organizado em mais 5 seções. A Seção 2 apresenta o modelo RDF e sua linguagem de consulta denominada SPARQL. A seção seguinte introduz os modelos de sistemas relacionais para armazenamento de dados RDF. Adicionalmente, é apresentado o *triple-store* Jena-TDB, bem como o sistema ntSQL que é o representante dos sistemas que utilizam noções da estrutura de dados RDF. A Seção 4 apresenta a avaliação experimental realizada por este trabalho, que por sua vez compara o *triple-store* Jena TDB-Fuseki com o ntSQL. Os trabalho relacionados a este trabalho, no que se refere a outras avaliações similares, são apresentados pela Seção 5. A Seção 6 apresenta as conclusões deste trabalho, bem como as perspectivas de trabalhos futuros.

2. Conceitos Iniciais

Esta seção introduz conhecimentos necessários para a compreensão do estudo experimental a ser apresentado por este artigo. Para tanto, as seções a seguir apresentam o modelo RDF e sua linguagem de consulta SPARQL.

2.1. RDF

RDF é a sigla para *Resource Description Framework* e que, segundo a *World Wide Web Consortium*, é um *framework* para representação de informações na Web. Uma característica básica do modelo RDF é que os metadados utilizados para descrever características de um site, por exemplo, precisam seguir uma estrutura básica de organização. Esta estrutura é reconhecida como tripla, composta por *sujeito-predicado-objeto*.

Para demonstrar este aspecto, considere como exemplo a seguinte afirmação: "Leandro comprou um automóvel." Neste caso, *Leandro* é o sujeito, *comprou* é o predicado e *automóvel* é o objeto. Esta frase, ou tripla, pode ser representada com sua estrutura sujeito-predicado-objeto através de um grafo. A Figura 1 mostra um grafo composto por esta e outras triplas. Embora o exemplo fornecido omita este detalhe, observa-se que por ser um modelo de dados voltado à *Web*, as informações que denotam sujeitos e objetos são especificadas por identificadores de recursos na *Web* dados por URIs (*Uniform Resource Identifier*). Ou seja, um sujeito ou objeto poderia ser, ao invés de um nome, uma URI para um site que tenha informações sobre o dado. Na representação gráfica, as elipses representam sujeitos e objetos especificados por URIs, as setas representam predicados e os retângulos representam objetos que são do tipo literal.

Com base no exemplo da Figura 1 é possível extrair as seguintes triplas do grafo direcionado: "Leandro comprou automóvel", "automóvel tipo caminhonete", "automóvel ano 2012", "automóvel modelo XX".

2.2. SPARQL

SPARQL (**S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage) é uma linguagem de consulta sobre dados RDF. Ela define consultas através de padrões de triplas RDF na forma de *sujeito-predicado-objeto*. Com base na Figura 1, um exemplo de utilização da linguagem SPARQL é verificar o modelo e ano do automóvel comprado por Leandro através da seguinte consulta:

```
1       SELECT DISTINCT ?qualmodelo, ?qualano
2       WHERE {
3           Leandro comprou automovel
4           automovel modelo ?qualmodelo
5           automovel ano ?qualano
6       }
```

Observe que a estrutura de formação da consulta se dá por padrões de triplas. Os elementos das triplas podem ser especificados conforme um grafo RDF, ou serem definidos como variáveis utilizando o ? como prefixo. No exemplo os objetos que referem-se ao modelo e cor do automóvel são tratados como variáveis cujos valores serão obtidos e retornados pela consulta.

SPARQL suporta uma variedade de consultas. Entretanto, a linguagem possui limitações como, por exemplo, sub-expressões não são suportadas. A abrangência do SPARQL aumenta conforme a utilidade do RDF também aumenta, e com isso, a necessidade de *benchmarks* para testar as capacidades da linguagem. Um *benchmark* para este cenário é o *Berlin SPARQL Benchmark* ou BSBM. O objetivo do BSBM, segundo Bizer and Schultz 2009, é ajudar os desenvolvedores a encontrar a melhor arquitetura e o melhor sistema de banco de dados para suas necessidades.

O BSBM é baseado em um caso de uso de um sistema de *e-commerce*, onde uma lista de produtos é oferecida por vendedores e posteriormente avaliada por clientes através de revisões. Um exemplo de consulta SPARQL do BSBM é dada a seguir:

```
1
2       PREFIX rdf: <http://www.w3.org/.../22-rdf-syntax-ns#>
3       PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4       PREFIX bsbm:<http://www4.wiwiss.fu-berlin.de/.../>
5       SELECT ?product ?label
6       WHERE {
7           ?product rdfs:label ?label .
8           ?product rdf:type bsbm:Product .
9           FILTER regex(?label, "%word1%")
10      }
```

Esta consulta é uma das mais simples encontradas no BSBM, e visa obter produtos que tem como rótulo uma *string* específica. Processos de *benchmarking* utilizando o BSBM foram executados sobre diversos sistemas de armazenamento que suportam RDF. Como mencionado anteriormente, o foco do presente artigo está sobre sistemas relacionais para armazenamento RDF. Para tanto, a seção a seguir apresenta alguns modelos empregados por estes sistemas e suas principais características.

3. Sistemas Relacionais para Armazenamento RDF

Nesta seção são apresentados dois modelos de armazenamento de dados RDF em sistemas relacionais, bem como alguns dos sistemas que os implementam. Os sistemas apresentados são comparados na avaliação a ser apresentada pela Seção 4.

3.1. Triple-Stores

Um dos formatos para armazenamento RDF, o *Triple-Store* utiliza uma mistura do formato de um modelo de sistema gerenciador de banco de dados relacional com a facilidade de inferência de dados do modelo RDF. Em um *triple store*, dados RDF são armazenados como um conjunto de triplas em uma tabela Abadi et al. 2009. O diferencial deste formato é a facilidade de criação da base de dados, para tal só é necessário criar uma única tabela que irá conter três campos, esses campos serão respectivamente o campo do sujeito, do predicado e do objeto. Deste modo, as tuplas desta tabela correspondem às triplas de um grafo RDF.

Existem alguns exemplos de sistemas que utilizam este formato de armazenamento. Em geral, estes sistemas suportam outros modelos de armazenamento fornecendo portabilidade quanto a diferentes tipos de dados. Alguns exemplos de *triple stores* são o Virtuoso (W3C 2016), Jena TDB (Apache Jena 2016b) e o RDF3-X (Neumann and Weikum 2010). Dentre estes, apenas o RDF3-X (Neumann and Weikum 2010) suporta exclusivamente o modelo RDF. O Jena TDB Fuseki (Apache Jena 2016a) é uma extensão do Jena TDB com suporte exclusivo a consultas RDF.

O Jena TDB é um componente do projeto Jena que fornece armazenamento e consultas para modelos utilizados na Web Semântica, como OWL, RDF e XML. Foi desenvolvido para atuar como um repositório de dados para a Web Semântica de alto desempenho, mesmo em uma única máquina. Seu sistema de armazenamento utiliza o modelo de *triple-store* apresentado anteriormente. O processador de consultas SPARQL é servido pelo componente Jena Fuseki Apache Jena 2016a, que por sua vez é considerado um servidor SPARQL. Em conjunto com o TDB, o Fuseki provê um sistema de armazenamento persistente e transacional para RDF. O Jena Fuseki é um *framework* Java de código aberto.

3.2. ntSQL

O ntSQL é uma ferramenta para a conversão de bases de dados RDF para bases de dados relacionais. A ferramenta é composta de dois módulos. O primeiro módulo compreende um conversor de dados RDF em formato NT para *scripts* de criação de esquemas relacionais, bem como instruções para a inserção de dados no formato SQL Bayer et al. 2014. O segundo módulo da ferramenta corresponde ao mapeamento de consultas SPARQL para consultas SQL sobre o esquema relacional produzido pelo primeiro módulo. Embora em operação, este segundo módulo não se encontra disponível para publicação.

O mapeamento de dados RDF para o modelo relacional é baseado na extração da estrutura RDF a partir de suas fontes de dados. Em resumo, assume-se que sujeitos e objetos não-literais estão relacionados a seus respectivos tipos. No mapeamento, após identificados os tipos, relações para cada tipo são criadas tendo como tuplas os dados relacionados aos respectivos tipos no grafo RDF. Relacionamentos entre os tipos são também identificados para a devida criação de chaves estrangeiras entre as relações criadas.

$< Usuario1 > < type > < Pessoa >.$
$< Usuario1 > < nome > < Pedro >.$
$< Usuario2 > < type > < Pessoa >.$
$< Usuario2 > < nome > < Joao >.$
$< Usuario3 > < type > < Pessoa >.$
$< Usuario3 > < nome > < Maria >.$
$< Usuario1 > < responsavelPor > < Usuario2 >.$
$< Usuario1 > < responsavelPor > < Usuario3 >.$

Tabela 1. Triplas RDF

Tabela 2. Relação Pessoas

id	nome	responsavel
Usuario1	Pedro	
Usuario2	João	Usuario1
Usuario3	Maria	Usuario1

Como exemplo de mapeamento RDF-Relacional, considere o seguinte conjunto de triplas RDF dadas no formato NT pela Tabela 1. No formato NT cada tripla é representada por uma linha, sendo que sujeito, predicado e objeto são colocados entre $<>$. O mapeamento obtido pelo ntSQL para este conjunto de triplas pode ser representada pela relação da Tabela 2. Como pode ser observado nas triplas apresentadas, os usuários são todos do tipo *Pessoa*, e podem estar relacionados entre si através do predicado *responsavelPor*. Neste caso, o mapeamento para relacional é dado pela criação de uma relação para este tipo *Pessoa*, sendo que o campo *id* pode ser definido como a chave primária da relação, e o campo *responsavel* como uma chave estrangeira representando o relacionamento entre os usuários.

Embora o exemplo apresentado seja simples, é possível perceber que a extração de estruturas de dados a partir de tipos RDF viabiliza uma representação RDF mais adequada em SGBDs relacionais, se comparada ao modelo *triple-store*. A comparação entre estes dois modelos é estabelecida pelos experimentos da próxima seção.

4. Avaliação Experimental

Nesta seção é apresentada uma avaliação experimental que compara o desempenho de um sistema do tipo *triple-store* com o ntSQL, isto é, dois sistemas relacionais destinados ao armazenamento de dados RDF. Os sistemas comparados correspondem aos sistemas apresentados pelas Seções 3.1 e 3.2 deste artigo, isto é, TDB (Fuseki) e o ntSQL respectivamente. A métrica utilizada por esta avaliação refere-se ao desempenho destes sistemas dado pelo tempo de resposta em consultas SPARQL. As bases utilizadas por este experimento, bem como as consultas, foram extraídas do *benchmark* Berlin SPARQL Benchmark (BSBM) introduzido pela Seção 2.2.

Para a escolha dos sistemas de armazenamento comparados, foi escolhido o sistema TDB (Fuseki) por ser o triple-store do projeto Jena, que por sua vez aparece no topo do *ranking* do site DB-Engines (DB-Engines 2016) como sistema de armazenamento mais popular e destinado ao modelo RDF. Em princípio o sistema RDF-3X também havia sido escolhido devido a sua evidência como sistema de referência em diversos artigos. No entanto, observou-se que o sistema foi descontinuado e a versão mais recente disponível apresentava alguns *bugs*, bem como alguns resultados inconsistentes. Quanto ao ntSQL, sua escolha se deu pelo caráter inovador de seu modelo de armazenamento, se comparado aos *triple-stores*. O sistema de gerenciamento de banco de dados utilizado pelo ntSQL foi o MySQL. Recomenda-se a leitura da Seção 3.2, para uma melhor compreensão do modelo aplicado pelo ntSQL.

As configurações aplicadas no experimento, bem como os resultados obtidos, são apresentados pelas seções a seguir.

Tabela 3. Número de triplas RDF por quantidade de produtos

Triplas RDF	Produtos
100	40382
200	75555
400	156054
800	297721
1600	585208
2000	725310

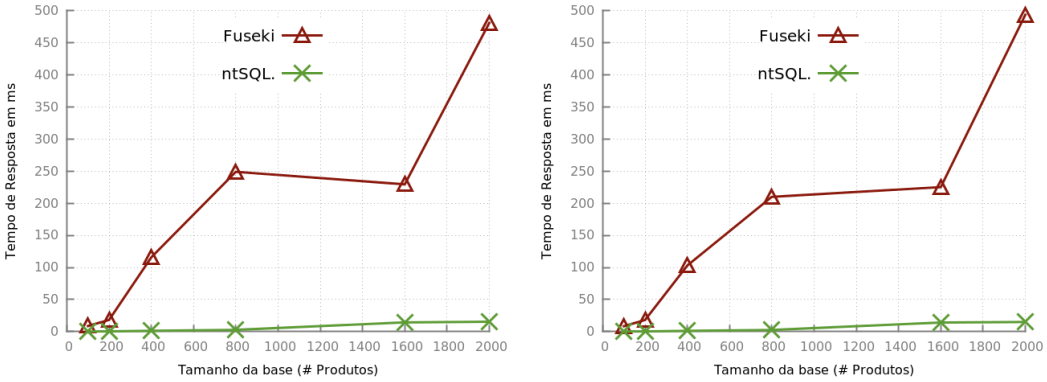
4.1. Configurações do Experimento

A máquina utilizada para os testes possui 4G de memória RAM, um processador AMD phenom II x4 e o sistema operacional linux-ubuntu 14.04. Para a realização do experimento, foram utilizadas as consultas 1, 6 e 9 do BSBM. Entre as 11 consultas SPARQL disponibilizadas pelo BSBM escolheu-se estas 3 pois representam tamanhos diferentes de consultas em termos da quantidade de padrões de triplas apresentadas por cada um, bem como da quantidade de resultados retornados. Desta forma, acredita-se conferir uma abrangência representativa do BSBM na comparação dos sistemas e seus resultados. Para verificar a escalabilidade dos sistemas comparados utilizaram-se bases com tamanhos variados. No BSBM o fator de escala da base corresponde a quantidade de produtos do sistema de *e-commerce*. No caso do experimento foram utilizadas bases de 100, 200, 400, 800, 1600 e 2000 produtos. Uma base com 2000 produtos possui 725310 triplas de RDF, como pode ser observado na Tabela 3. A comparação entre os 3 sistemas escolhidos é apresentada na seção a seguir com base na métrica do tempo de resposta para as consultas do BSBM.

4.2. Resultados obtidos

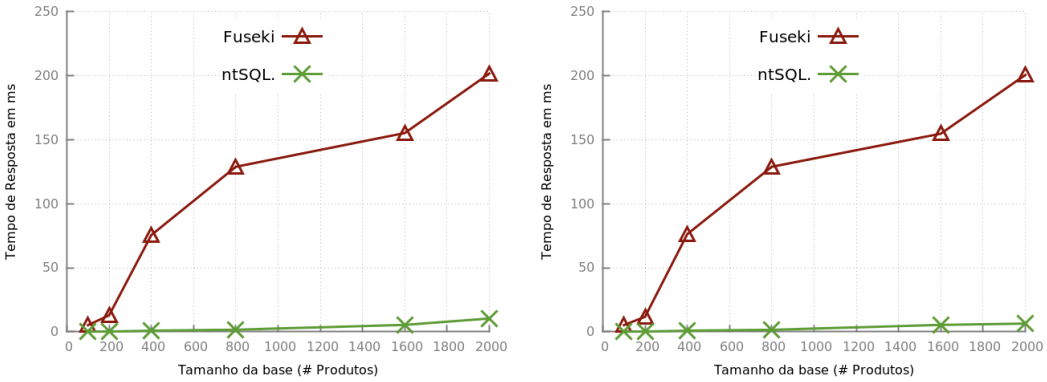
Como mencionado na seção anterior, as três consultas escolhidas do BSBM foram utilizadas para os testes em seis tamanhos de base diferentes. Cada consulta foi executada dez vezes para cada tamanho de base, dos quais foram retirados para cada base a média e mediana. Os tempos de resposta dos 2 sistemas comparados com relação as consultas 1, 3 e 9 do BSBM são apresentados pelas Figuras 2, 3 e 4, respectivamente.

Observa-se em todas as consultas que o Fuseki apresenta um desempenho muito inferior em comparação com o ntSQL. O ntSQL provou ter um melhor tempo para efetuar as consultas, demonstrando um desempenho com um crescimento quase constante, não possuindo nenhuma variação drástica com o crescimento do tamanho do banco. Em relação ao Fuseki seu crescimento apresenta picos de acordo com o crescimento do tamanho dos bancos, com aumento do tempo de resposta muito superior ao ntSQL. Acredita-se que esta diferença possa ser explicada pelo modelo de armazenamento utilizado por *triple-stores* ao processar consultas com diversos padrões de triplas. Como por exemplo a consulta 1 apresenta duas auto-junções para o ntSQL e quatro para o Fuseki, a consulta 6 não apresenta nenhuma junção para o ntSQL e duas junções para o Fuseki, e para a consulta 9 não é utilizado nenhuma junção. Um motivo para diferença de tempo entre os dois modelos seria o tamanho das tabelas, a tabela utilizada pelo ntSQL é significantemente menor. Ao colocar todas as triplas em uma mesma tabela, além de gerar grandes arquivos para as relações, surge a necessidade da execução de diversas auto-junções para a recuperação dos diversos padrões de triplas das consultas. O modelo empregado pelo ntSQL mostra-se mais adequado neste sentido por distribuir os dados em diferentes relações, agrupando-os de acordo com seus tipos.



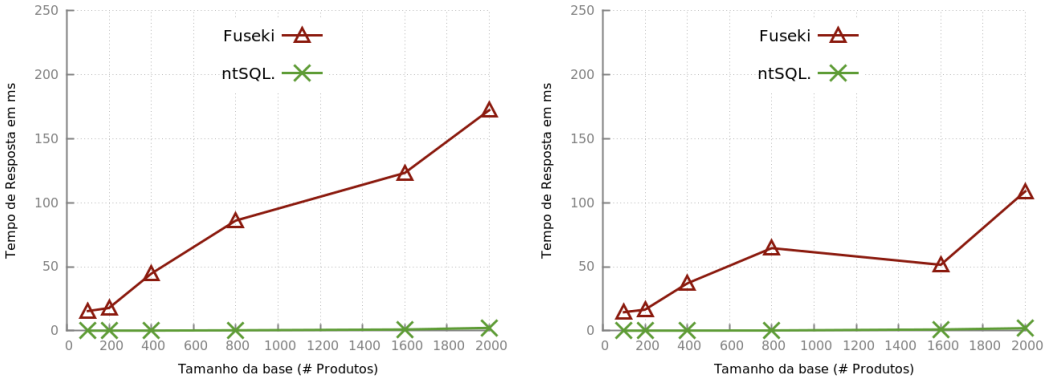
(a) Média (b) Mediana

Figura 2. Desempenho dos Sistemas - Consulta 1 do BSBM



(a) Média (b) Mediana

Figura 3. Desempenho dos Sistemas - Consulta 3 do BSBM



(a) Média (b) Mediana

Figura 4. Desempenho dos Sistemas - Consulta 9 do BSBM

5. Trabalhos Relacionados

Nesta seção será discutido os resultados de outros trabalhos que apresentam avaliações referentes a sistemas para RDF, mostrando assim o desempenho médio de bancos triple-store quando comparados entre si e também entre outros modelos de armazenamento. O primeiro trabalho escolhido é o *TDB Results for Berlin SPARQL Benchmark* (Bizer and Schultz 2009), que utiliza o *benchmark* do Berlin para avaliar o Jena-TDB. Foram feitos testes com 50k, 250k, 1M, 5M, 25M e 100M triplas do Berlin SPARQL Benchmark (BSBM). O TDB utilizado por este trabalho difere do que foi utilizado pelo estudo experimental apresentado pela Seção 4 por utilizar um outro processador de consultas SPARQL diferente do Fuseki, além de sua avaliação se tratar de um ambiente distribuído. Apesar das diferenças, observou-se que os tempos apresentados pelo presente trabalho são proporcionais aos obtidos nas respectivas consultas avaliadas por este trabalho.

Além deste, outro trabalho relacionado ainda com o BSBM é o *BSBM with Triples and Mapped Relational Data* (Orri Erling 2016). Este trabalho tem como objetivo demostrar que um esquema relacional mapeado a partir de um RDF tem um poder de processamento melhor, com respostas mais rápidas para as consultas do que um *triple-store*. O banco utilizado para os testes foi o OpenLink Virtuoso, que é considerado o *triple-store* mais rápido dentre os disponíveis no mercado (Morsey et al. 2011). Para uma base de 100M triplas no sistema, considerando o conjunto de consultas (*query mix*) original do BSBM, o *triple-store* avaliado por este trabalho conseguiu executar 5746 QMpH (*Query Mix Per Hour*), enquanto que para o repositório relacional mapeado do RDF o total foi de 7525 QMpH. Este resultado por si só mostra uma superioridade do repositório relacional mapeado, assim como constatado pelo presente trabalho através do ntSQL.

Outro trabalho que vale destaque é o *DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data* (Morsey et al. 2011), que tem como objetivo criar e testar um novo *benchmark* com um caso de uso real, dados reais e aplicabilidade já testadas na Web Semântica. Além deste caso real, os testes foram realizados em três sistemas RDF de evidência no cenário atual, ou seja, que estão bem rankeados no DB-Engines 2016. Eles são o Virtuoso, Sesame e BigOWLIM. Os dados carregados no sistema são extraídos da DBpedia, uma grande diferença em relação aos outros *benchmarks* que possuem dados sintéticos. O *dataset* escolhido por Morsey et al. 2011 possui 153.737.776 triplas. Os resultados do trabalho mostram que o Virtuoso tem o melhor desempenho entre os três em mais de 90% dos casos, e o segundo melhor sistema é o BigOWLIM seguido do Sesame. Esses dois últimos tem resultados bem próximos, porém o BigOWLIM tem vantagem nos maiores *datasets*, enquanto o Sesame tem vantagem nos menores.

6. Conclusão

Este artigo apresentou um estudo experimental que compara dois modelos de armazenamento de dados RDF em sistemas relacionais através dos sistemas Jena-TDB Fuseki e ntSQL. Como esperado, o Jena-TDB Fuseki apresentou um desempenho inferior ao ntSQL, atestando as desvantagens do uso de *triple-stores* já apontadas por outros trabalhos. O melhor desempenho do ntSQL pode ser atribuído ao uso de um esquema relacional mais robusto do que os utilizados por *triple-stores*. Como trabalho futuro, pretende-se envolver outros sistemas na avaliação e outros *benchmarks*. Além disto, pretende-se de-

senvolver uma análise mais detalhada que possa justificar o desempenho dos sistemas através de características de consultas e do esquema de banco de dados.

Agradecimentos: Este trabalho foi parcialmente suportado pelos programas de iniciação científica PIC&DTI e PIPES da Universidade do Estado de Santa Catarina.

Referências

aaaa aaaa.

Abadi et al. 2009 Abadi, D. J., Marcus, A., Madden, S. R., and Hollenbach, K. (2009). SW-Store: A Vertically Partitioned DBMS for Semantic Web Data Management. *The VLDB Journal*, 18(2):385–406.

Apache Jena 2016b Apache Jena (Acesso em Fevereiro de 2016b). Apache Jena TDB. https://jena.apache.org/documentation/tdb/index.html.

Apache Jena 2016a Apache Jena (Acesso em Janeiro de 2016a). Apache Jena Fuseki. https://jena.apache.org/documentation/fuseki2/.

Arnaut et al. 2011 Arnaut, D., Schroeder, R., and Hara, C. (2011). Phoenix: A Relational Storage Component for the Cloud. In *IEEE International Conference on Cloud Computing (CLOUD)*, pages 684–691.

Bayer et al. 2014 Bayer, F. R., Nesi, L. L., and Schroeder, R. (2014). ntSQL: Um Conversor de Documentos RDF para SQL. In *Anais da Escola Regional de Banco de Dados*. SBC.

Bizer and Schultz 2009 Bizer, C. and Schultz, A. (2009). The Berlin SPARQL Benchmark. In *International Journal on Semantic Web & Information Systems*.

DB-Engines 2016 DB-Engines (Acesso em Janeiro de 2016). DB-Engines Ranking of RDF Stores. http://db-engines.com/en/ranking/rdf+store.

J. Huang, D. Abadi 2011 J. Huang, D. Abadi, K. R. (2011). Scalable SPARQL Querying of Large RDF Graphs. *PVLDB*, 4(11):1123–1134.

Morsey et al. 2011 Morsey, M., Lehmann, J., Auer, S., and Ngomo, A.-C. N. (2011). DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data. In *International Semantic Web Conference*.

Neumann and Weikum 2010 Neumann, T. and Weikum, G. (2010). The rdf-3x engine for scalable management of rdf data. *The VLDB Journal*, 19(1):91–113.

Orri Erling 2016 Orri Erling (Acesso em Janeiro de 2016). BSBM with Triples and Mapped Relational Data. http://www.openlinksw.com/dataspace/doc/oerling/weblog/Orri

Papailiou et al. 2014 Papailiou, N., Tsoumakos, D., Konstantinou, I., Karras, P., and Koziris, N. (2014). H2rdf+: An efficient data management system for big rdf graphs. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 909–912. ACM.

Pham 2013 Pham, M. (2013). Self-organizing structured RDF in MonetDB. In *IEEE 29th International Conference on Data Engineering Workshops (ICDEW)*, pages 310–313.

Ravindra et al. 2011 Ravindra, P., Hong, S., Kim, H., and Anyanwu, K. (2011). Efficient processing of rdf graph pattern matching on mapreduce platforms. In *Proceedings of the Second International Workshop on Data Intensive Computing in the Clouds*, DataCloud-SC '11, pages 13–20. ACM.

W3C 2016 W3C (Acesso em Janeiro de 2016). OpenLink Virtuoso. https://www.w3.org/2001/sw/wiki/OpenLink_Virtuoso.

Zeng et al. 2013 Zeng, K., Yang, J., Wang, H., Shao, B., and Wang, Z. (2013). A Distributed Graph Engine for Web Scale RDF data. *Proceedings of the VLDB Endowment*, 6(4):265–276.