Análise de Desempenho de Ferramentas para Persistência de Dados Ontológicos em Triplas: Experimentos e Resultados

Felipe Luzzardi da Rosa[1], **Roger da Silva Machado**[1], **Tiago Thompsen Primo**[1], **Adenauer Corrêa Yamin**[1], **Ana Marilza Pernas**[1]

[1]Universidade Federal de Pelotas (UFPel)

{fldrosa,rdsmachado,tiago.primo,adenauer,marilza}@inf.ufpel.edu.br

Abstract. *The use of ontologies for knowledge representation in several areas has become increasingly common. One of the reasons for the growing interest in using these structures is the capacity of knowledge representation and reasoning, especially when there is a high amount of data. However, few papers reffer to persistence and management of this kind of data, mostly focusing on modeling aspects. That being said, the motivation of this paper is to extend the research related to the persistence of this kind of data. For this, performance tests were made to compare the AllegroGraph, GraphDB, MarkLogic, Stardog, and Virtuoso tools using the WatDiv benchmark. In the tests performed, the Virtuoso tool presented the best performance in the vast majority of the queries, considering one, five, and ten milion triples.*

Resumo. *A utilização de ontologias para representação de conhecimento em diversas áreas vem sendo cada vez mais comum. Um dos motivos para o crescente interesse no uso dessas estruturas é a capacidade de se realizar inferência no conhecimento armazenado, derivando novo conhecimento, especialmente quando a quantidade de dados é alta. Entretanto, observa-se pouca atenção relacionada à forma de se persistir e gerenciar esses dados, sendo as pesquisas mais focadas em sua modelagem. Desta forma, este artigo tem como motivação ampliar a pesquisa relacionada a persistência destes dados. Para isso, foram realizados testes de desempenho visando comparar as ferramentas AllegroGraph, GraphDB, MarkLogic, Stardog e Virtuoso utilizando o benchmark WatDiv. Nos testes realizados a ferramenta Virtuoso apresentou o melhor desempenho na grande maioria das consultas considerando um, cinco e dez milhões de triplas.*

1. Introdução

Aplicações desenvolvidas em diferentes áreas, como Big Data, Internet das Coisas (IoT) e Web Semântica, utilizam fontes de informação contextual, obtidas pelas próprias aplicações, para tomadas de decisões. A capacidade de armazenar, correlacionar e produzir conhecimento a partir dessas fontes de informação é um tema cada vez mais relevante [Gray et al. 2014]. Uma estratégia que vem sendo amplamente empregada é o uso de ontologias.

Um dos diversos exemplos da utilização de ontologias para a modelagem de dados é a modelagem de contexto de alunos para auxiliar um ambiente de aprendizagem ubíqua (*u-learning*) [González et al. 2016]. Tais ambientes frequentemente fazem uso de

Objetos de Aprendizado[1], necessitando de um modelo ontológico para acompanhar os mesmos ao longo de seu ciclo de vida [Gluz and Vicari 2014]. Além disso, ontologias de metadados de Objetos de Aprendizado também são parte relevante nos projetos desta área [Behr et al. 2016].

Embora diversas propostas apresentadas neste âmbito façam uso de tais informações, poucas versam sobre a maneira de garantir a persistência das mesmas [Pierin and Sichman 2018, Lunardi et al. 2018], usualmente mantendo as mesmas em arquivos texto, sendo necessário carregar esses arquivos para a memória a cada manipulação.

Percebe-se ainda que alguns trabalhos utilizam repositórios relacionais para tal fim [Santos et al. 2018, Fujimmoto and Canedo 2018], os quais não são considerados modelos satisfatórios para persistência de dados ontológicos. Uma forma mais eficiente de manter a persistência destes dados é utilizando um modelo de armazenamento em triplas RDF (*Resource Description Framework*).

O modelo de armazenamento em triplas pode ser dividido em três categorias com base na arquitetura de sua implementação: em memória; nativo; e externo. No armazenamento em memória, todo o conjunto de triplas é mantido na memória principal do dispositivo para manipulação, o que o torna ineficiente quando o volume de dados é grande [Maharajan 2012].

O modelo de armazenamento nativo fornece persistência com o uso de uma base de dados projetada para esta funcionalidade. As ferramentas que implementam este modelo oferecem seus próprios recursos de manipulação da base de dados, utilizando linguagens como a SPARQL para acessá-lo. Na categoria de armazenamento externo, as triplas são persistidas, por exemplo, em bases relacionais, as quais não são apropriadas para o armazenamento de triplas [Can et al. 2017].

Nota-se que o armazenamento nativo de triplas possui um bom desempenho, melhorando a capacidade de raciocínio do sistema como um todo, e proporcionando assim uma base de dados apropriada para o modelo de triplas (RDF) [BioOntology 2011]. Apesar disso, poucos trabalhos presentes na literatura tratam efetivamente sobre a persistência e a manipulação dos dados provenientes de ontologias, sendo que a manipulação eficiente desses dados tem impacto direto nos serviços oferecidos, podendo tornar o processo de tomada de decisão mais eficiente.

Visando contribuir para a pesquisa na área, este artigo apresenta uma análise de desempenho de cinco ferramentas populares para armazenamento nativo de triplas: AllegroGraph, GraphDB, MarkLogic, Stardog e Virtuoso. Com base nesta análise será possível identificar aquela mais apta para ser utilizada em sistemas que dependem de informações provenientes de ontologias.

Para realizar a análise de desempenho foi utilizado o *benchmark* WatDiv [Aluç 2014], o qual disponibiliza um número diverso de consultas para teste, possibilitando uma análise criteriosa das ferramentas. Os experimentos foram realizados em duas

[1]Unidade de instrução/ensino reutilizável. Podem ser definidos por ”qualquer entidade, digital ou não digital, que possa ser utilizada, reutilizada ou referenciada durante o aprendizado suportado por tecnologias”.

dimensões, na primeira foi analisada a escalabilidade de cada ferramenta, considerando diferentes tamanhos da base de dados. Na segunda os desempenhos fornecidos pelas ferramentas foram comparados entre si. O objetivo do experimento é o de identificar a ferramenta que apresenta a melhor relação entre escalabilidade e desempenho.

O restante do artigo está dividido como segue. A Seção 2 apresenta trabalhos relacionados ao gerenciamento de dados no formato de triplas. A Seção 3 versa sobre as ferramentas de gerenciamento de dados nativas em formato de triplas, mostrando um breve resumo sobre as cinco ferramentas analisadas. A Seção 4 mostra o *benchmark* WatDiv, enquanto a Seção 5 apresenta a análise de desempenho das ferramentas com este *benchmark*. Por fim, são apresentadas as considerações finais e os trabalhos futuros na Seção 6.

2. Trabalhos relacionados

Os trabalhos de [Bizer and Schultz 2009] e [Aluç 2014] apresentam propostas de *benchmarks* para ferramentas de gerenciamento nativo em formato RDF, utilizando consultas na linguagem SPARQL para medir os tempos de execução de diversas ferramentas. Os dois trabalhos apresentam resultados de análises de desempenho utilizando os *benchmarks* propostos em diversas ferramentas de armazenamento em triplas. Porém, devido ao fato de que o principal objetivo destes trabalhos é apresentar os *benchmarks* propostos, é dado pouco foco à análise de desempenho em si, sendo dada mais atenção a apresentação das características técnicas dos *benchmarks*. Além disso, como os artigos em questão foram escritos na época do lançamento de seus *benchmarks*, diversas das ferramentas de armazenamento apresentadas neles estão atualmente obsoletas ou possuem versões mais atuais.

Em [Rajabi et al. 2015] e [Can et al. 2017] o armazenamento de dados em formato de triplas RDF é apontado como mais eficiente do que o armazenamento em um modelo relacional nos contextos tratados pelos trabalhos. Os artigos, além de comparar o desempenho de seus estudos de caso com armazenamento no formato RDF e no modelo relacional, discutem diferenças de suas vocações. Como o foco destes trabalhos é a inclusão do modelo de armazenamento em triplas em seus respectivos contextos, não é dada atenção especial às diversas ferramentas disponíveis para realizar tal armazenamento, apresentando análises de desempenho simples entre uma ferramenta que utiliza o modelo relacional e outra que utiliza o modelo de triplas.

Em [Gluz et al. 2017] são realizados experimentos para analisar ferramentas de gerenciamento de triplas RDF para testar e avaliar a eficácia e desempenho do mesmos. Foram analisadas as ferramentas RDF4J [RDF4J 2019] e Apache Jena [Jena 2019], utilizando a metodologia Berlin de *benchmarks*, a qual não obriga que os dados estejam conforme uma representação ontológica. Os testes foram realizados com base na simulação de clientes do banco de dados, executando consultas SPARQL para busca de informações. Nos testes analisados a ferramenta Jena obteve o melhor desempenho.

Dois dos trabalhos relacionados identificados, [Rajabi et al. 2015] e [Can et al. 2017], limitam-se a comparar o desempenho de uma determinada alternativa a este modelo com opções tradicionais baseadas no modelo relacional. Outros dois trabalhos, [Bizer and Schultz 2009] e [Aluç 2014], tratam efetivamente do desempenho de ferramentas de gerenciamento, porém encontram-se desatualizados e tendem a focar

mais na concepção de um *benchmark* do que nas análises de desempenho em si.

Já o trabalho de [Gluz et al. 2017] tem como foco uma avaliação baseada em desempenho de duas ferramentas de gerenciamento no formato de triplas. Apesar de realizar esta avaliação, o mesmo compara somente duas ferramentas, e, ainda, é analisado somente a execução de duas consultas SPARQL. Além disso, não são realizados nenhum tipo de teste estatístico para validação dos testes realizados.

Dito isso, este trabalho possui o diferencial de tratar especificamente de uma análise de desempenho de ferramentas de gerenciamento de triplas para suporte a raciocínio e inferência. Esta análise é baseada no tempo de execução de diferentes consultas SPARQL, sendo analisadas cinco ferramentas populares para gerenciamento de triplas. Ademais, é apresentado uma validação dos resultados com o uso de um teste estatístico.

3. Ferramentas de gerenciamento de dados nativo em triplas

Os bancos de dados de armazenamento nativo em triplas foram construídos para o armazenamento e recuperação de dados no formato RDF. Para a análise desenvolvida, foram escolhidas cinco ferramentas para armazenamento nativo de triplas, com base principalmente em sua popularidade. As cinco ferramentas escolhidas estão entre as dez primeiras no ranking da DB-Engine [DB-Engines 2019] e são descritas a seguir.

AllegroGraph [Allegrograph 2019] é uma estrutura de banco de dados e aplicação de alto desempenho para o armazenamento e consulta de dados no formato de triplas. Pode ser implantada como um servidor de banco de dados independente e oferece interfaces para acesso remoto, onde a comunicação entre os processos de servidor e cliente é realizada pela Web.

GraphDB (antigamente conhecido como OWLIM) [GraphDB 2019] é uma ferramenta para armazenamento de dados no formato de triplas. Possui a possibilidade de realização de inferência semântica, permitindo aos usuários criar novos fatos semânticos com base em dados existentes. Tanto as consultas como as inferências realizadas sobre os dados armazenados podem ser realizadas em tempo de execução.

MarkLogic [Marklogic 2019] é um sistema gerenciador de dados não relacional multimodelo, capaz de armazenar nativamente documentos JSON e triplas RDF. Além de possuir um modelo de dados flexível, o MarkLogic possui uma arquitetura distribuída que permite trabalhar com uma grande quantidade de dados, mantendo consistência entre as transações.

Stardog [Stardog 2019] é um sistema gerenciador de banco de dados gráfico semântico, implementado em Java. Ele possui suporte para RDF e OWL *Web Ontology Language*, fornecendo capacidades de raciocínio e utilizando SPARQL como linguagem de consulta. Ele possui disponibilidade de acesso aos dados utilizando a web, e *plugins* que possibilitam a utilização de outros *frameworks*.

Virtuoso [Virtuoso 2019] é um sistema gerenciador de banco de dados que combina a funcionalidade de um banco de dados relacional, XML e RDF, sendo projetado para tirar vantagem do suporte de segmentação do sistema operacional e múltiplos processadores. O armazenamento de dados é realizado utilizando uma quádrupla onde, além de armazenar a tripla básica (sujeito, predicado e objeto), também é armazenado o grafo

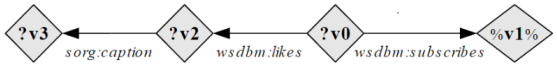
relacionado. Com isso o Virtuoso consegue trabalhar com múltiplos grafos de forma simultânea.

4. Benchmark utilizado

O *benchmark* escolhido para os testes desenvolvidos neste artigo foi o Waterloo SPARQL Diversity Test Suite (WatDiv), desenvolvido pela Universidade de Waterloo [Aluç 2014]. Dentre os principais critérios de escolha deste *benchmark*, destacam-se os diferentes tipos de consulta disponibilizados e o fato de ele ser baseado realmente em triplas. Desta forma, permitindo que o *benchmark* identifique problemas de desempenho sobre ferramentas de gerenciamento de triplas existentes que não são detectados por outros *benchmarks*. O WatDiv fornece um gerador de *datasets* com um fator de escala variável, permitindo assim a geração de *datasets* de diversos tamanhos.

Acompanha o pacote do *benchmark* um gerador de consultas SPARQL. As consultas são geradas em quatros grupos: "Lineares" (L), mais simples e diretas; "Estrela" (S) consultas que referenciam diversos nodos em um formato de estrela; "Floco de Neve"(F), consultas que referenciam vários nodos que por sua vez referenciam outros nodos; e "Complexas" (C), consultas que utilizam uma mistura dos formatos anteriores. Ao todo são disponibilizadas vinte consultas, as quais são divididas entre os grupos da seguinte forma: cinco Lineares; sete Estrelas; cinco Flocos de neve; e três Complexas.

A Figura 1 apresenta um exemplo de consulta do tipo Linear, mostrando a consulta L1 em seu formato grafo e em sua representação SPARQL. Já na Figura 2, é apresentado um exemplo de uma consulta do tipo Estrela, apresentando a consulta S1, também em suas formas de grafo e de consulta SPARQL. Por sua vez, a Figura 3 mostra um exemplo de consulta do tipo Floco de neve, mostrando a consulta F2 também em seu formato de grafo e representação SPARQL. Por fim, a Figura 4 apresenta um exemplo de consulta do tipo Complexa, mostrando a consulta C2 tanto em seu formato de grafo quanto em sua representação SPARQL.



(a) L1 em formato de grafo

```
#mapping v1 wsdbm:Website uniform
SELECT ?v0 ?v2 ?v3
 WHERE {
   ?v0 wsdbm:subscribes %v1% .
   ?v2 sorg:caption ?v3 .
   ?v0 wsdbm:likes ?v2 .
}
```

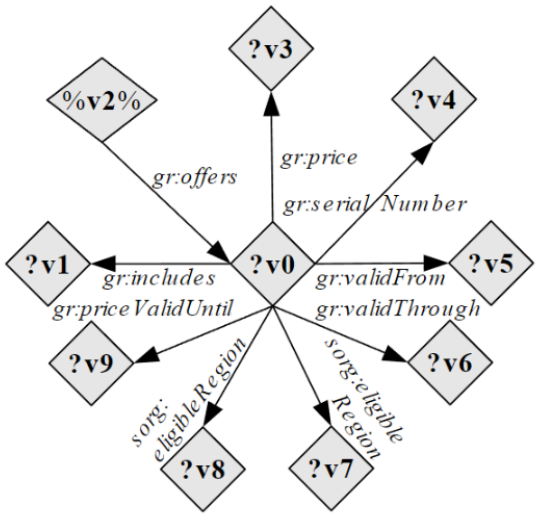
(b) L1 SPARQL

Figura 1. Exemplo de consulta do tipo Linear

É importante observar que variações no tamanho de uma base RDF não devem refletir significativamente no desempenho de seu acesso. O mesmo não é válido para as diferenças de tempo de acesso entre consultas pertencentes aos diferentes grupos.

5. Testes e resultados

Para realização da análise das ferramentas apresentadas na Seção 5, foram gerados *datasets* com 1, 5 e 10 milhões de triplas, os quais foram inseridos nas cinco ferramentas analisadas. Com o intuito de analisar o desempenho das ferramentas optou-se pela execução de doze consultas, sendo três de cada grupo.

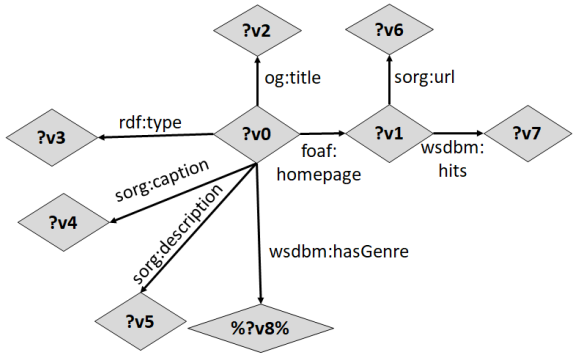


(a) S1 em formato de grafo

```
#mapping v2 wsdbm:Retailer uniform
SELECT ?v0 ?v1 ?v3 ?v4 ?v5 ?v6 ?v7 ?v8 ?v9
 WHERE {
  ?v0 gr:includes ?v1 .
  %v2% gr:offers ?v0 .
  ?v0 gr:price ?v3 .
  ?v0 gr:serialNumber ?v4 .
  ?v0 gr:validFrom ?v5 .
  ?v0 gr:validThrough ?v6 .
  ?v0 sorg:eligibleQuantity ?v7 .
  ?v0 sorg:eligibleRegion ?v8 .
  ?v0 sorg:priceValidUntil ?v9 .
 }
```

(b) S1 SPARQL

Figura 2. Exemplo de consulta do tipo Estrela



(a) F2 em formato de grafo

```
#mapping v8 wsdbm:SubGenre uniform
SELECT ?v0 ?v1 ?v2 ?v4 ?v5 ?v6 ?v7
 WHERE {
   ?v0 foaf:homepage ?v1 .
   ?v0 og:title ?v2 .
   ?v0 rdf:type ?v3 .
   ?v0 sorg:caption ?v4 .
   ?v0 sorg:description ?v5 .
   ?v1 sorg:url ?v6 .
   ?v1 wsdbm:hits ?v7 .
   ?v0 wsdbm:hasGenre %v8% .
 }
```

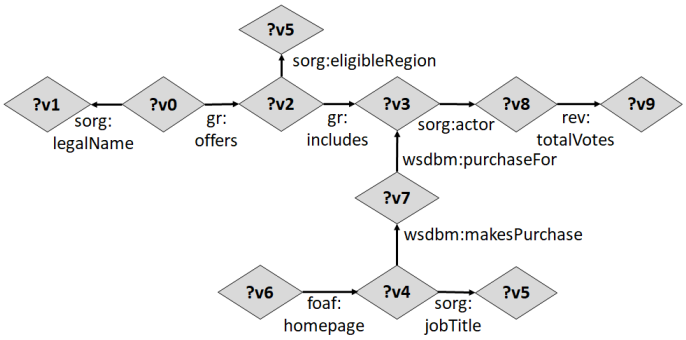
(b) F2 SPARQL

Figura 3. Exemplo de consulta do tipo Floco de neve

Desta forma, considerando todos os valores permitidos para cada variável, teria-se um conjunto de 180 estudos de caso, cada um representado por $P_i = (d, f, c)$, onde:

- d: informa o tamanho do dataset utilizado (1, 5 ou 10 milhões);
- f: identifica a ferramenta utilizada (Alegrograph, GraphDB, Stardog, Virtuoso e MarkLogic);
- c: informa a consulta utilizada, sendo três de cada grupo (Linear - L1, L4 e L5, Estrela - S2, S5 e S7, Floco de neve - F2, F4 e F5 e Complexa - C1, C2 e C3)

Devido a limitações de sua licença gratuita, os testes com 10 milhões de triplas não foram realizados na ferramenta AllegroGraph. Com isso, há duas possibilidades para d (1, 5), cinco para f e doze para c, mais os executáveis de quatro ferramentas para d com valor de 10 milhões. Desta forma, chega-se ao total de $\mathcal{P} = \{P_1, P_2, ...P_{168}\}$ executáveis. Cada combinação dos valores permitidos foi executada 30 vezes, desta forma, permitindo realizar a análise dos dados coletados com uma maior precisão. Assim, foram coletados $168 \times 30 = 5040$ tempos de execuções individuais.



(a) C2 em formato de grafo

```
SELECT ?v0 ?v3 ?v4 ?v8
 WHERE {
  ?v0 sorg:legalName ?v1 .
  ?v0 gr:offers ?v2 .
  ?v2 sorg:eligibleRegion
wsdbm:Country5 .
  ?v2 gr:includes ?v3 .
  ?v4 sorg:jobTitle ?v5 .
  ?v4 foaf:homepage ?v6 .
  ?v4 wsdbm:makesPurchase ?v7 .
  ?v7 wsdbm:purchaseFor ?v3 .
  ?v3 rev:hasReview ?v8 .
  ?v8 rev:totalVotes ?v9 .
 }
```

(b) C2 SPARQL

Figura 4. Exemplo de consulta do tipo Complexa

Os tempos médios de execução em segundos para cada tipo de consulta, “Lineares” (L), “Estrela” (S) ,“Floco de Neve”(F) e “Complexas” (C) são apresentados na respectivamente nas figuras 5, 6, 7 e 8.

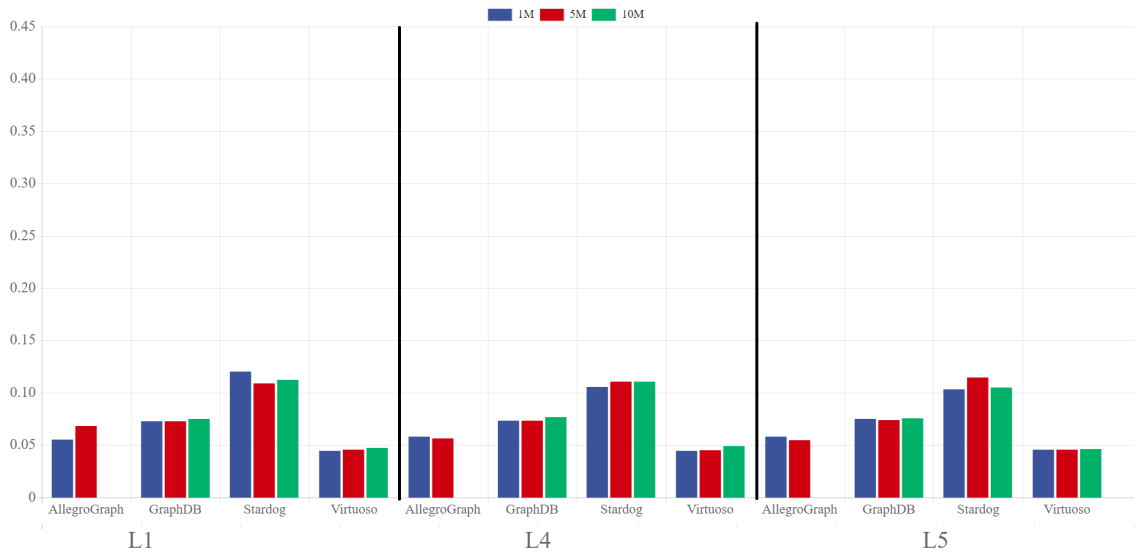


Figura 5. Tempo de execução das consultas do tipo L.

Os desempenhos com a ferramenta MarkLogic não são apresentados pois, além de ter um desempenho muito inferior às demais ferramentas, o desvio padrão se apresentou muito elevado.

Com os resultados apresentados, é possível concluir que a ferramenta Virtuoso apresentou os melhores resultados de desempenho na grande maioria das consultas. Também é possível observar que as ferramentas Virtuoso e Stardog apresentaram boa escalabilidade, com baixa variação de tempo de execução entre os diferentes tamanhos de base de dados, enquanto que as ferramentas AllegroGraph e GraphDB apresentaram uma escalabilidade também boa, porém inferior as duas ferramentas anteriormente mencionadas.

Esta conclusão foi validada aplicando o teste t de Student com intervalo de confiança de 95% entre as cinco ferramentas, em cada um dos 3 tamanhos e em cada

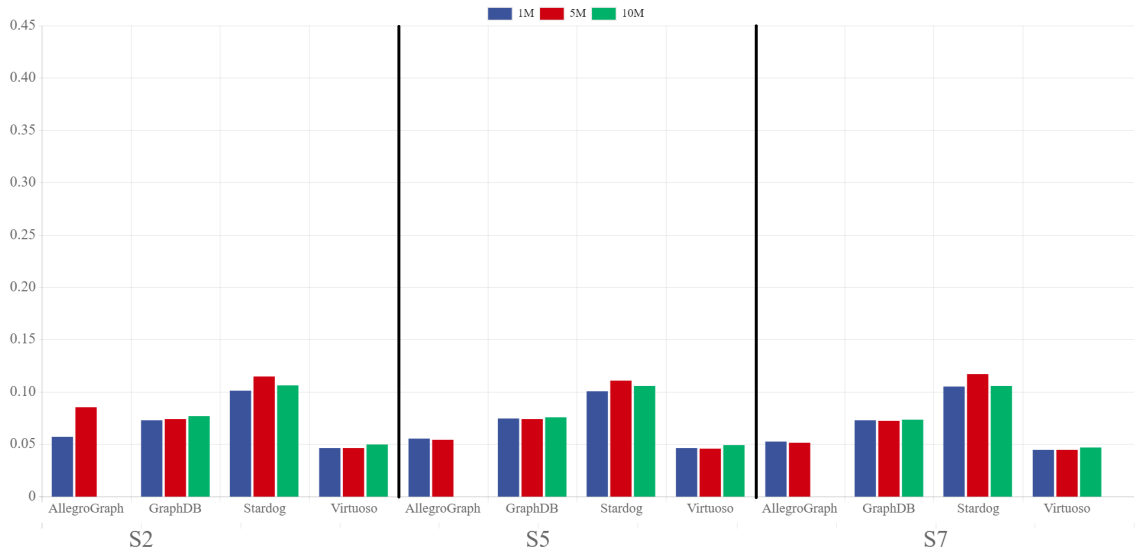


Figura 6. Tempo de execução das consultas do tipo S.

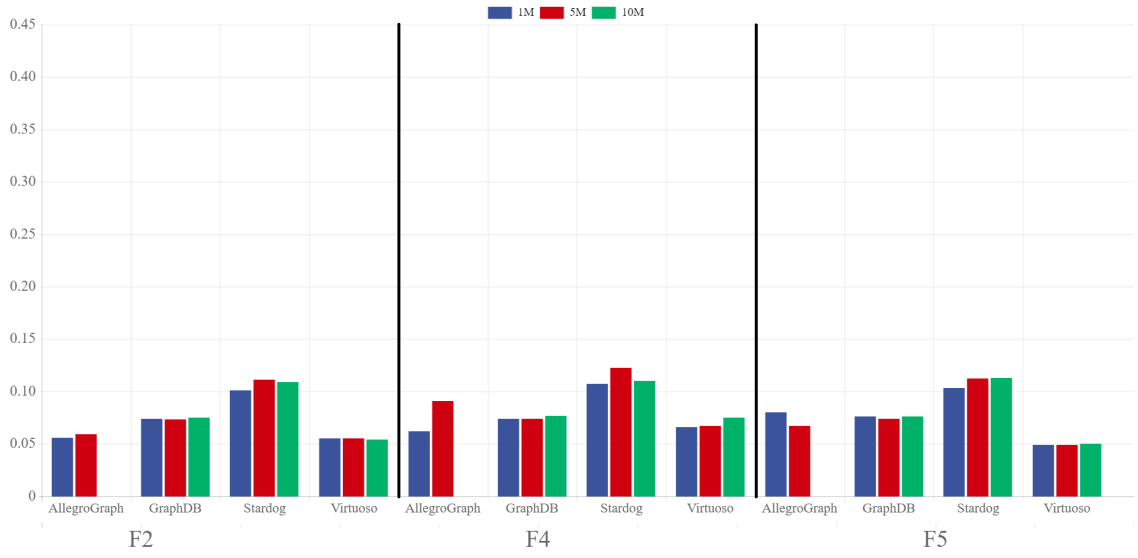


Figura 7. Tempo de execução das consultas do tipo F.

uma das 12 consultas. Como resultados destes testes foram detectados que os resultados da consulta C1 com o tamanho 5M entre Stardog e AllegroGraph e entre Stardog e GraphDB não obtiveram diferenças estatísticas significativas. Ainda, a consulta C3 para o tamanho 1M entre Stardog e GraphDB também não obteve diferenças estatísticas significativas. Os demais resultados apresentaram diferenças estatísticas significativas, assegurando os resultados obtidos e apresentados nas figuras 5, 6, 7 e 8.

Também pode ser notado que o Stardog obteve resultados muito similares em todos os testes realizados, tendo sido a ferramenta com menor alteração de desempenho entre tamanhos diferentes de base de dados. Além dos testes entre as ferramentas, foi analisado se o tamanho da base impactava no desempenho das mesmas. Para isso, foi realizado o teste t entre as médias de execução de cada ferramenta comparando o tamanho

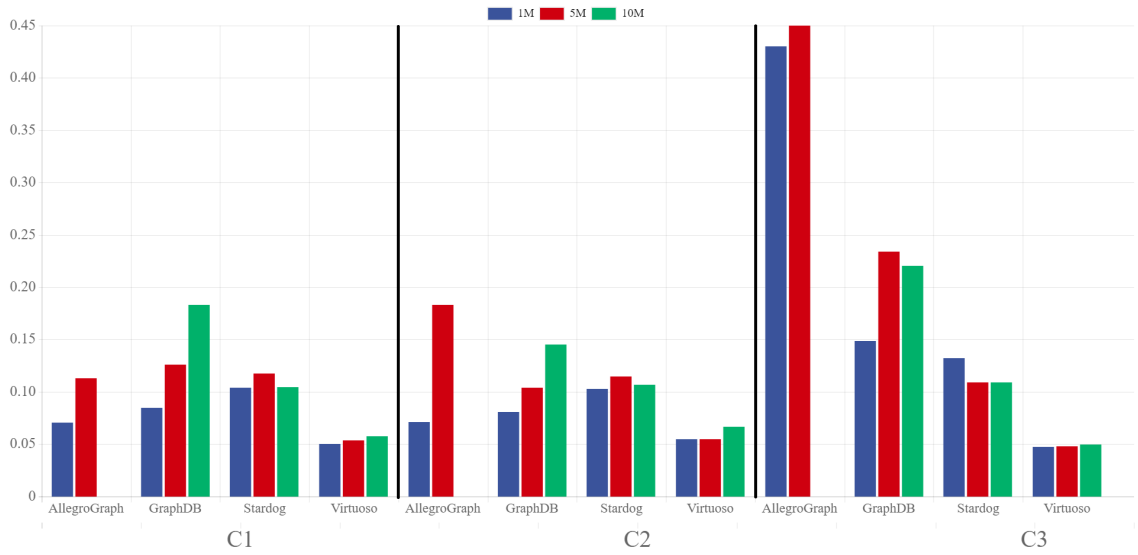


Figura 8. Tempo de execução das consultas do tipo C.

1M com 5M e 5M com 10M.

As consultas que não apresentaram diferenças significativas são apresentadas na Tabela 1. Pode-se observar que a ferramenta AllegroGraph foi a que apresentou menor número de consultas com diferenças não significativas. No entanto, relembra-se que nesta ferramenta não foi possível realizar consultas em uma base de tamanho de 10M. A ferramenta que apresentou maior número de diferenças não significativas foi a MarkLogic, pois além de apresentar tempos de execução altos, também apresentou valores muito altos para os desvios padrões anotados.

Embora o motivo de a ferramenta Virtuoso ter apresentado os melhores desempenhos em todos os testes não tenha sido um foco do trabalho, uma hipótese bastante provável para este fato é sua estratégia de armazenamento em quádrupla. Com essa estratégia, a ferramenta armazena não apenas a tripla básica (sujeito, predicado, objeto) mas também o grafo relacionado da mesma. O armazenamento deste grafo permite que o Virtuoso acesse o mesmo diretamente quando deseja realizar uma consulta, fazendo com que o tempo para a construção do mesmo, gasto pelas demais ferramentas, seja economizado.

Tabela 1. Consultas que não apresentaram diferenças significativas

Tamanho	AllegroGraph	GraphDB	MarkLogic	Stardog	Virtuoso
1M ≠ 5M	S5	L1, L4, L5, F4, S5, S7	C2, C3, L4, L5, F2, S2, S5, S7	C3, L1, F4, S7	C2, L5, F5, S7
5M ≠ 10M		L5	C2, C3, F2, F4, S2, S5, S7	C3, L1, L4, F2, F4, F5, S7	

Com a análise dos testes realizados, é possível constatar que a ferramenta que apresenta a melhor escalabilidade em relação tanto ao tamanho da base de dados quanto as diferentes consultas foi a Stardog. Porém, destaca-se que apesar da ferramenta Virtuoso variar um pouco mais que a Stardog, ela obteve os melhores resultados na grande maioria dos testes realizados, sofrendo pouca variação no desempenho com o aumento no tamanho

da base de dados.

6. Considerações finais

Este trabalho realizou, primariamente, uma identificação de ferramentas para armazenamento de dados em formato de triplas, um formato altamente apropriado para o armazenamento de dados ontológicos. Em conjunto, o trabalho também realizou uma comparação de desempenho relacionada ao tempo de execução de consultas SPARQL entre cinco ferramentas de armazenamento em formato de triplas: AllegroGraph, GraphDB, MarkLogic, Stardog e Virtuoso. Foi apresentada uma descrição de cada ferramenta analisada e também do *benchmark* utilizado para a execução dos testes, o WatDiv. Os testes consideraram o tempo de execução, em segundos, de doze diferentes consultas em três tamanhos de bases de dados.

Ao fim dos testes foi possível concluir que a ferramenta Virtuoso foi a que mostrou o melhor desempenho geral, apresentando o tempo de execução mais baixo na grande maioria das consultas, apesar de apresentar uma flutuação no desempenho um pouco maior que Stardog quando varia-se a quantidade de dados tratada.

Estes resultados foram importantes nas decisões de pesquisa a serem tomadas pelo grupo, pois forneceram embasamento para a escolha da ferramenta mais adequada para gerenciar o modelo de triplas implementado no *middleware* para Computação Ubíqua desenvolvido pelo grupo. Esta escolha impacta diretamente nos serviços cientes de contexto oferecidos, pois as aplicações enviam continuamente requisições de consulta aos contextos gerenciados pelo *middleware* para sua tomada de decisão, sendo importante se ter o menor tempo possível de acesso aos dados. Além disso, a identificação e a utilização da ferramenta de armazenamento adequada para as ontologias utilizadas pelo grupo afetam diretamente a capacidade de realizar inferência nas mesmas, facilitando e agilizando a tarefa.

Como trabalhos futuros destaca-se a continuidade da pesquisa ligada a provimento de serviços cientes de contexto, aplicando a ferramenta Virtuoso para acesso aos dados do modelo de triplas. Onde esta ferramenta será utilizada para gerenciar a persistência dos dados providos pela ontologia utilizada em um ambiente *u-learning*, utilizando também uma estratégia de aprendizado multimodelo.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001 e da FAPERGS (Programa Pesquisador Gaúcho - PqG).

Referências

Allegrograph (2019). Acesso em maio 2019. Disponível em: `franz.com/agraph/allegrograph/`.

Aluç, G. (2014). Diversified Stress Testing of RDF Data Management Systems. Master's thesis, David R. Cheriton School of Computer Science, Waterloo, ON, Canada.

Behr, A., Primo, T., and Viccari, R. (2016). Towards educational metadata interoperability on semantic web. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 27, page 1026.

BioOntology (2011). Comparison of Triple Stores. Disponível online em: <https://www.bioontology.org/wiki/images/6/6a/Triple_Stores.pdf>. acesso em novembro 2016.

Bizer, C. and Schultz, A. (2009). The berlin sparql benchmark.

Can, O., Sezer, E., Bursa, O., and Unalir, M. O. (2017). Comparing relational and ontological triple stores in healthcare domain. *Entropy*, 19(1):30.

DB-Engines (2019). DB-Engines Ranking of RDF Stores. Acesso em maio 2019. Disponível em: https://db-engines.com/en/ranking/rdf+store.

Fujimmoto, M. M. T. and Canedo, E. D. (2018). Modelo conceitual de dados baseado em ontologia: Estudode caso cgu. In *ONTOBRAS*.

Gluz, J. C., Machado, F., and Galão, M. C. (2017). Avaliação de tecnologias de bancos de dados semânticos para a construção de um sistema inteligente de gestão de conteúdos de aprendizagem: Experimentos e resultados. In *VI Congresso Brasileiro de Informática na Educação (CBIE 2017) - XXVIII Simpósio Brasileiro de Informática na Educação (SBIE 2017)*.

Gluz, J. C. and Vicari, R. M. (2014). Rumo a uma plataforma semântica de conteúdos educacionais digitais: o modelo ontológico. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 25, page 993.

González, G., Durán, E., and Amandi, A. (2016). Context ontologies in ubiquitous learning environments. In Montes y Gómez, M., Escalante, H. J., Segura, A., and Murillo, J. d. D., editors, *Advances in Artificial Intelligence - IBERAMIA 2016*, pages 391–403, Cham. Springer International Publishing.

GraphDB (2019). Acesso em maio 2019. Disponível em: http://graphdb.ontotext.com/.

Gray, A. J. G., Groth, P., Loizou, A., Askjaer, S., Brenninkmeijer, C., Burger, K., Chichester, C., Evelo, C. T., Goble, C., Harland, L., Pettifer, S., Thompson, M., Waagmeester, A., and Williams, A. J. (2014). Applying linked data approaches to pharmacology: Architectural decisions and implementation. *Semant. web*, 5(2):101–113.

Jena (2019). Acesso em maio 2019. Disponível em: http://jena.apache.org/.

Lunardi, G. M., Machado, G. M., Machado, A., and de Oliveira, J. P. M. (2018). Um modelo ontológico probabilístico para assistir pessoascom declínio cognitivo. In *ONTOBRAS*.

Maharajan, S. (2012). Performance of native SPARQL query processors. Master's thesis, Department of Information Technology, Uppsala University.

Marklogic (2019). Acesso em maio 2019. Disponível em: https://www.marklogic.com/product/marklogic-database-overview/database-features/semantics/.

Pierin, F. L. and Sichman, J. S. (2018). Integraweb: uma arquitetura baseada em mapeamentossemânticos. In *ONTOBRAS*.

Rajabi, E., Sicilia, M.-A., and Sanchez-Alonso, S. (2015). Interlinking educational resources to web of data through IEEE LOM. *Computer Science and Information Systems*, 12(1):233–255.

RDF4J (2019). Acesso em maio 2019. Disponível em: `http://rdf4j.org/`.

Santos, L. A., Miranda, G. M., Campos, S. L., Falbo, R. A., Barcellos, M. P., Souza, V. E. S., and Almeida, J. P. A. (2018). Using an ontology network for data integration: A case in the public security domain. In *ONTOBRAS*.

Stardog (2019). Acesso em maio 2019. Disponível em: `stardog.com/`.

Virtuoso (2019). Acesso em maio 2019. Disponível em: `http://virtuoso.openlinksw.com/`.