Avaliação do desempenho de gerenciadores de bancos de dados multi modelo em aplicações com persistência poliglota

Fábio Roberto Oliveira, Luis Mariano del Val Cura
Faculdade Campo Limpo Paulista (FACCAMP)
Campo Limpo Paulista – SP – Brasil

foliveira@ymail.com , delval@cc.faccamp.br

Abstract. *Polyglot persistence application is a term recently introduced for applications that use several data models to represent and stores their data. Usually, those applications became very complex because they are implemented using simultaneously several database management systems. Multimodel database management systems were introduced to offer in a single system several data models. This paper presents a performance evaluation of multimodel database management systems associated to polyglot persistence applications. In this paper, the performance of several multimodel data tests are compared using simultaneously single model and multimomodel database management systems.*

Resumo. *Aplicações com persistência poliglota são aquelas que precisam que seus dados sejam representados e armazenados segundo múltiplos modelos de bancos de dados. Em geral, estas aplicações utilizam vários sistemas de gerenciamento de bancos de dados, que implementem esses modelos, o que torna seu desenvolvimento mais complexo. Como uma solução a este problema, foram introduzidos os sistemas de gerenciamento de bancos de dados multi modelos, que implementam em um único sistema vários modelos de dados. Este artigo apresenta uma avaliação do desempenho de sistemas de gerenciamento multi modelo, com dados associados a uma aplicação com persistência poliglota. Para esta avaliação, é comparado o desempenho de um conjunto de testes de dados multi modelos, utilizando vários sistemas de gerenciamento de um único modelo de dados e sistemas de gerenciamento multi modelo.*

1. Introdução

A oferta de novos gerenciadores de bancos de dados do tipo NoSQL cresceu muito nos últimos anos, chegando a ultrapassar uma centena de opções. A adoção desses novos Sistemas de Gerenciamento de Bancos de Dados (SGBD) ainda é uma decisão complexa para arquitetos de softwares e administradores de bancos de dados, principalmente porque cada um deles atende um modelo específico de dados, com interfaces de acesso e linguagens de consultas não padronizadas, e com desempenho focado a um modelo específico de dados. A escolha errada de um gerenciador de banco de dados NoSQL pode gerar um desempenho inadequado conforme os dados crescem, e a troca de um SGBD deste porte não é uma tarefa trivial, porque não há uma

padronização nesses gerenciadores ainda. É comum que os requisitos de uma aplicação obriguem a adoção de vários gerenciadores de bancos de dados, cada um atendendo um tipo de modelo de dados, como por exemplo, modelos de documentos, chave-valor ou grafos. Quando uma aplicação trabalha simultaneamente com vários modelos de bancos de dados diferentes, dizemos que ela utiliza persistência poliglota [Fowler 2013]. A vantagem da adoção de múltiplos gerenciadores de bancos de dados, para atender os modelos de dados diferentes, basicamente se resume a um bom desempenho, porque cada um dos componentes da aplicação gerencia e armazena seus dados no SGBD mais apropriado para aquele modelo de dados. As desvantagens são inúmeras: replicação de dados entre SGBDs diferentes, dificuldade no gerenciamento de transações, alto custo total de propriedade (TCO) devido a administração de vários SGBDs diferentes, com interfaces e ferramentas diferentes. Para suprir a necessidade de aplicações que dependem de modelos de bancos de dados NoSQL distintos, surgiram gerenciadores de bancos de dados que integram vários modelos de bancos de dados NoSQL num mesmo SGBD. Essa nova geração de SGBDs do tipo NoSQL, são denominados multi modelo.

Este artigo apresenta uma comparação de desempenho entre gerenciadores de bancos de dados baseados em um único modelo de dados, e gerenciadores de bancos de dados multi modelo. Para realizar esta pesquisa, serão utilizados dados de aplicações com requisitos que requerem a persistência poliglota. Estes dados serão armazenados em dois ou mais SGBD mono modelos, e simultaneamente em um SGBD multi modelo. Sobre estes dados, serão realizados múltiplos testes envolvendo operações típicas de gerenciamento e consulta de bancos de dados. O restante do artigo está organizado da seguinte forma. A seção 2 descreve os principais tipos de gerenciadores de bancos de dados NoSQL, apresentando em seguida os novos gerenciadores de bancos de dados NoSQL multi modelo. A seção 3 descreve os trabalhos relacionados e a metodologia adotada para avaliar o desempenho dos SGBDs. Na sequência na seção 4 são apresentados os testes e resultados obtidos e por fim a seção 5 apresenta a conclusões, sugestões e melhorias em trabalhos futuros.

2. Gerenciadores de bancos com modelos de dados NoSQL

Os gerenciadores de bancos de dados do tipo NoSQL compartilham características em comum. Algumas dessas características são: flexibilidade no modelo de dados, escalabilidade horizontal, novas linguagens de consultas apropriadas aos modelos de dados e diversas interfaces para acesso aos dados. Apesar de compartilhar essas características, que foram necessidades que originaram essa nova geração de SGBDs denominados NoSQL, a maioria dos SGBDs trabalha com um modelo de dados único, sendo necessário adotar um SGBD específico para cada modelo. Nesta seção descrevemos esses tipos de modelos, além de apresentar os novos gerenciadores de banco de dados multi modelo.

2.1. Modelos de dados NoSQL

Modelo de dados Chave-valor: Este modelo armazena os dados como uma grande tabela hash. O banco de dados é composto por um conjunto de chaves, que estão associadas a um único valor. Este modelo é de fácil implementação, bem escalável, e

permite que os dados sejam rapidamente acessados pela chave. Isso contribui para aumentar a disponibilidade de acesso aos dados. As operações disponíveis para manipulação de dados são bem simples, como modificar e recuperar. A desvantagem deste modelo é que ele não permite a recuperação de objetos por meio de consultas mais complexas, porque os dados estão estruturados de forma simples. Um dos gerenciadores de bancos de dados de maior sucesso desse modelo é o Dynamo [DeCandia et al. 2007] criado pela Amazon.

Modelo de dados Orientado a Colunas: Este modelo é um pouco mais complexo que o modelo chave-valor. O conceito de registros ou tuplas do modelo relacional é alterado para orientação a atributos ou colunas. Neste modelo os dados são indexados por uma tripla (linha, coluna e timestamp), onde as linhas e colunas são identificadas por chaves, e o timestamp permite diferenciar múltiplas versões de um mesmo dado. As operações de leitura e escrita são atômicas, isto é, todos os valores associados a uma linha são considerados na execução destas operações, independentemente das colunas que estão sendo lidas ou escritas. Outro conceito associado ao modelo é o de família de colunas, que é usado com o intuito de agrupar colunas que armazenam o mesmo tipo de dados. Este modelo de dados surgiu com o BigTable [Chang et al. 2006] criado pelo Google, por isso é comum falar sobre o modelo de dados BigTable. Este modelo permite particionamento dos dados, além de oferecer forte consistência.

Modelo de dados Orientado a Documentos: Este modelo armazena coleções de documentos. Um documento em geral, é um objeto com um identificador único e um conjunto de campos, que podem ser strings, listas ou documentos aninhados. Estes campos se assemelham a estrutura chave-valor já mencionado anteriormente, mas permite encadeamento como no formato XML. No formato mais comum do modelo orientado a documentos, temos um conjunto de documentos, e em cada documento temos um conjunto de campos (chaves) e o valor deste campo, formato conhecido como JSON. Outra característica importante é que este modelo não depende de um esquema rígido, ou seja, não exige uma estrutura fixa como ocorre nos bancos relacionais. O formato do documento é livre, permitindo assim que ocorra uma atualização na estrutura do documento, com a adição de novos campos, sem causar problemas ao modelo. A flexibilidade é uma das grandes vantagens deste modelo, além da velocidade. O gerenciador mais utilizado deste modelo é o MongoDB [Suter 2012].

Modelo de dados Orientado a Grafos: O modelo orientado a grafos possui três componentes básicos. Os nós (vértices do grafo), os relacionamentos (arestas) e as propriedades (atributos) dos nós e relacionamentos. Neste caso, o banco de dados pode ser visto como um multigrafo rotulado e direcionado, onde cada par de nós pode ser conectado por mais de uma aresta. A vantagem de utilização do modelo baseado em grafos, fica bastante clara quando consultas complexas, com filtros em propriedades nas arestas, são exigidas pelo usuário. Comparado ao modelo relacional, que para estas situações pode ser muito custoso, o modelo orientado a grafos tem um ganho de desempenho significativo. Aplicações que precisam fazer operações de percorrimento em múltiplos níveis tiram vantagem desse modelo, podendo inclusive só ter um desempenho ótimo utilizando bancos de dados orientado a grafos. O gerenciador mais popular desse segmento é o Neo4j [Miller 2013]

2.2. Os novos gerenciadores de bancos de dados NoSQL multi modelo

Além da definição do conceito de SGBD Multi Modelo, é importante ressaltar mais detalhadamente sua necessidade, recursos e vantagens. A necessidade de múltiplos modelos de dados numa mesma aplicação, é amplamente explorada pelo renomado autor Martin Fowler no seu livro NoSQL Distilled [Fowler 2013], definindo este conceito como persistência poliglota. A primeira geração de gerenciadores de bancos de dados NoSQL eram dirigidas a um modelo único de dados, considerado como seu modelo nativo. Um modelo de dados pode ser adequado para certos tipos de dados, e não para outros. Por exemplo, o cadastro de produtos de uma livraria se adapta perfeitamente ao modelo de dados de documentos, mas as ligações entre muitos produtos semelhantes, geralmente denominado como sistemas de recomendação, se adapta melhor ao modelo de dados de grafos. Para resolver essa necessidade de múltiplos modelos de dados, fornecedores de SGBDs do tipo NoSQL estenderam seus produtos oferecendo novos modelos, permitindo a manipulação de documentos, grafos, chave-valor e até orientação a objeto em alguns produtos. Nos SGBDs multi modelo, o armazenamento padrão é no estilo documentos. O modelo de grafos foi criado sobre o modelo de documentos, com documentos especiais chamados eixos (arestas) fazendo a ligação entre documentos especiais que são os nós (vértices). Os algoritmos de grafos foram adaptados para trabalhar com esses documentos especiais, provendo operações em formatos diferentes de dados. Estes novos SGBDs permitem que os dados fiquem consolidados num único produto, oferecendo uma linguagem de consulta única para todos os modelos de dados. A escalabilidade também é um fator importante, suportando alta carga de acessos simultâneos, sendo possível também em alguns produtos particionar os dados entre vários servidores, recurso denominado *sharding*. A consistência dos dados também é garantida, inclusive oferecendo transações entre os modelos de documentos e grafos simultaneamente. Provavelmente a maior vantagem destes SGBDs, além do alto desempenho dos SGBDs NoSQL, é a produtividade no desenvolvimento, porque apenas uma API ou linguagem de consulta é utilizada.

3. Trabalhos relacionados e metodologia adotada

Existem trabalhos na literatura que avaliam o desempenho de SGBDs e modelos NoSQL. Em [Jouili & Vansteenberghe 2013], são avaliados vários gerenciadores de bancos de dados NoSQL orientados a grafos, simulando operações com cargas de trabalho e tamanho de grafos diferentes, utilizando a interface de acesso *TinkerPop*, que é uma iniciativa de padronização no acesso a SGBDs orientados a grafos. Este artigo serviu como base de aprendizado, principalmente na análise das métricas de avaliação de SGBDs orientados a grafos. Em [Kolomicenko 2013], também são avaliados vários gerenciadores de bancos de dados NoSQL orientados a grafos, através da interface TinkerPop, utilizando grafos sintéticos. Embora o foco deste artigo também foi apenas verificar o desempenho em SGBDs orientados a grafos, ele apresenta a ferramenta BlueBench para avaliar o desempenho.

Em [Henricsson 2011], são avaliados unicamente gerenciadores de bancos de dados orientados a documentos, simulando operações de leituras e escritas com cargas de trabalho diferentes, utilizando bibliotecas de acesso para a linguagem Python. Esta dissertação apresenta métricas de desempenho de um SGBD orientado a documentos.

Em [Narde 2013], também são avaliados gerenciadores de bancos de dados NoSQL orientados a documentos, mas com o foco em comparar para qual estrutura de Cloud os SGBDs apresentam melhor desempenho. Foi utilizada a ferramenta Yahoo Cloud Serving Benchmark para avaliar o desempenho. Em todos os estudos verificados, apenas SGBDs do mesmo modelo foram avaliados. Não foram encontrados na literatura trabalhos de avaliação de desempenho envolvendo SGDBs multi modelo.

3.1 Metodologia de testes adotada.

Esta pesquisa pretende contribuir a determinar se uma aplicação com requisitos de persistência poliglota, obtém melhor desempenho em suas operações utilizando dois gerenciadores de bancos de dados de modelos de dados diferentes, ou utilizando um único gerenciador multi modelo. Para poder avaliar o desempenho, foi necessário encontrar um banco de dados de testes com requisitos de persistência poliglota, isto é, com parte dos seus dados modelados como documentos, e outra parte modelada como grafos. Foi escolhido um dataset de grande porte [Stanford 2015] com dados reais de vendas fornecido pela empresa Amazon. Neste dataset, o cadastro dos livros com os atributos código, título, grupo, classificação e categorias, se adapta ao modelo de documentos. O produto similar, devido aos seus relacionamentos, se adapta melhor ao modelo de grafos.

O uso do modelo de grafos é necessário para efetuar buscas em profundidade, simulando um sistema de recomendação de produtos. É possível recuperar os produtos relacionados a um produto, e através dos produtos recuperados relacionados, recuperar também seus produtos relacionados, criando assim uma busca em profundidade, ideal para o modelo de grafos. Essa busca não é apropriada para o modelo de documentos, porque nesse modelo apenas criam-se vínculos entre documentos, não oferecendo operações de percorrimento em profundidade. Os testes foram executados usando o SGBD mono modelo orientado a documentos MongoDB e o SGBD orientado a grafos Neo4j. Como SGBDs multi modelos, foram escolhidos o OrientDB e ArangoDB, que implementam tanto o modelo de dados de documento, como de grafos. Todas as operações executadas utilizaram a interface de acesso REST providas pelos SGBDs testados que padroniza o acesso aos SGBDs através de serviços WEB. Considerando as metodologias dos trabalhos relacionados, foram estabelecidos o seguinte conjunto de testes representativos para os dois modelos de dados: testes de inserção e recuperação de documentos, teste de recuperação de vértices e criação de arestas, e teste de percorrimento no grafo em profundidade.

4. Experimentos e resultados

Para realizar os testes, foram desenvolvidas aplicações em Java para acessar aos bancos testados através da arquitetura REST. Os testes propostos foram executados num servidor com processador AMD Opteron com 8 núcleos de 2.3ghz, 32Gb de RAM, duas HD's de 2Tb magnéticas configuradas em Raid 1. No caso do MongoDB, como ele não oferece uma interface REST completa, foi utilizada a interface RestHeart**.**

4.1.Teste de inserção de documentos

Nesse primeiro teste foram inseridos os 542.684 produtos do dataset da Amazon, através da interface Webservice REST, no formato JSON. Como podemos observar na figura 1, o SGBD ArangoDB, apesar de ser um SGBD multi modelo, apresentou um desempenho um pouco melhor do que o MongoDB, que é um SGBD de documentos nativo. O SGBD multi modelo OrientDB teve um desempenho muito inferior. Este teste mostra que para operações de de inserção simples, em um SGBD multi modelo, dependendo da implementação, o desempenho pode ser até superior a um SGBD de documentos nativo.

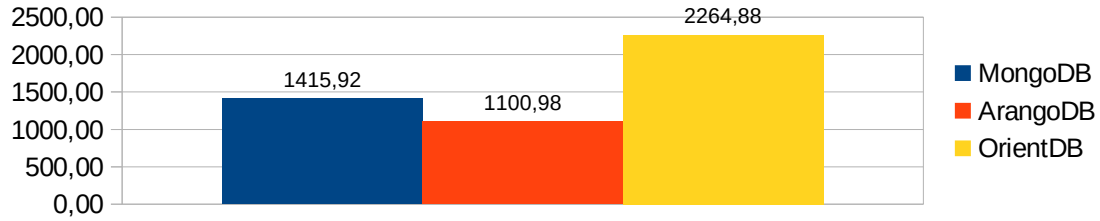

Figura 1. Tempo necessário para inserir 542.684 produtos (em segundos)

4.2. Teste de recuperação de documentos

Nesse segundo teste foram recuperados 1000 produtos na interface REST. O objetivo foi comparar o desempenho de leituras de coleções no formato JSON em um SGBD de formato nativo de documentos e em SGBDs do tipo multi modelo. Neste teste, com seus resultados apresentados na figura 2, ambos os SGBDs do tipo multi modelo superaram o SGBD de documentos nativo.

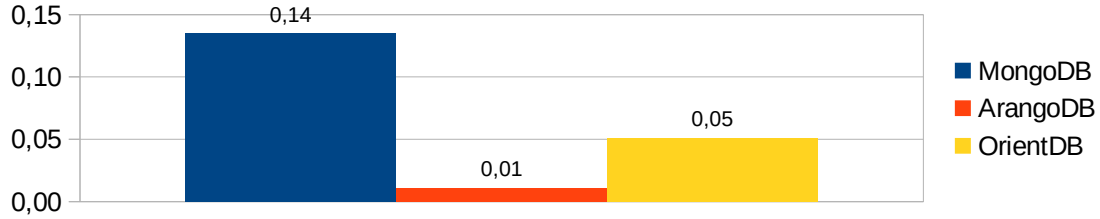

Figura 2. Tempo necessário para recuperar 1000 produtos (em segundos)

4.3. Teste de recuperação de vértices e criação de arestas

Neste terceiro teste, foi comparado o desempenho de uma operação de grafos nos bancos multi modelo,e em um SGBD de grafos nativo, como o Neo4j. Neste teste são recuperados dois vértices (produtos), e são criados arestas entre esses produtos, que representam os produtos relacionados. Podemos verificar na figura 3 que o SGBD de grafos nativo tem uma velocidade melhor nessas operações.

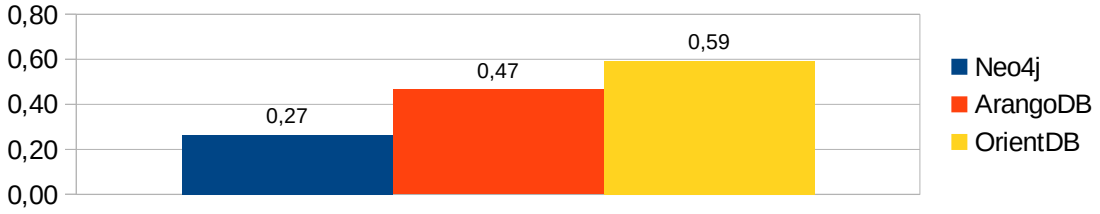

Figura 3. Tempo necessário para recuperar 2 produtos e criar um relacionamento entre eles (em segundos)

4.4. Teste de percorrimento no grafo em profundidade

Neste próximo teste executamos operações de percorrimento em profundidade de um grafo em vários níveis. Foram obtidos os produtos relacionados com um livro, e em seguida os produtos relacionados dos livros relacionados, e assim por diante, em vários níveis. A figura 4 mostra que o SGBD Neo4j teve uma leve vantagem em relação ao OrientDB, que também teve um bom desempenho. O SGBD multi modelo ArangoDB, se mostrou bem mais lento em operações de percorrimento em profundidade.

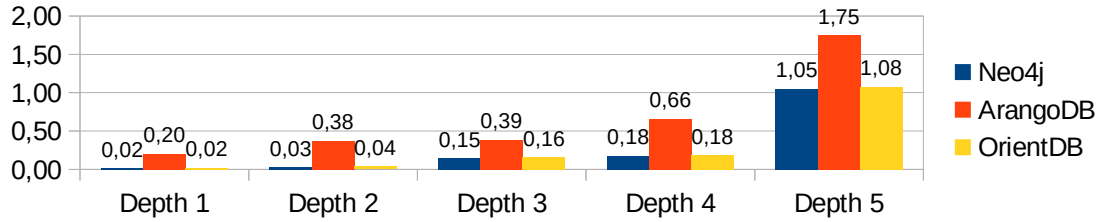

Figura 4. Tempo necessário para encontrar os produtos relacionados (em segundos)

4.5. Teste comparando múltiplos SGBDs mono modelo versus multi modelo

O último teste comparou o desempenho dos SGBDs com operações simples. Foram executadas três operações na sequência: gravar um livro, recuperar um livro e obter seus livros relacionados. Estas são duas operações típicas de documentos e uma operação de grafos. No caso da persistência poliglota, as operações foram executadas com os SGBDs MongoDB e Neo4j, ambos nativos e considerados os mais rápidos em cada modelo. Como vemos na figura 5, os SGBDs nativos MongoDB e Neo4j tiveram um desempenho levemente melhor que o SGBD multi modelo OrientDB.

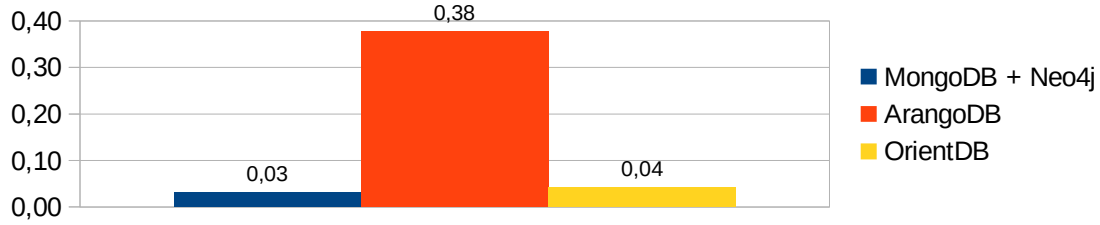

Figura 5. Tempo necessário para gravar um produto, recuperar o produto e seus livros relacionados (em segundos)

5. Conclusões

Com os resultados dos testes executados, concluímos que o uso da persistência poliglota em SGBDs mono modelo, ainda tem uma pequena vantagem de desempenho sobre os SGBDs multi modelo. Os SGBDs multi modelo testados apresentaram excelentes resultados em operações diferentes. Claramente o ArangoDB é muito rápido ao manipular documentos, sendo até mais rápido que um SGBD nativo de documentos, como o MongoDB. O OrientDB é muito rápido ao manipular grafos, sendo quase tão rápido quanto o Neo4j, que é um SGBD nativo orientado a grafos. Caso a aplicação necessite o melhor desempenho possível, não levando em conta a replicação de dados, dificuldade ao gerenciar transações e o TCO, a persistência poliglota com vários SGBDs mono modelo é indicada. Caso a aplicação queira concentrar os dados num único produto, o ArangoDB se mostrou mais apropriado para manipular documentos,

com bom desempenho em operações em grafos. O OrientDB se mostrou muito rápido em operações em grafos, com bom desempenho na manipulação de documentos. O modelo dos dados se mostrou muito importante na escolha de um SGBD multi modelo, em detrimento de outro. Como trabalho futuro, são necessários testes com mais datasets, além de testes de carga. Como os testes foram executados com operações simples, não paralelas, estes não simulam uma carga real de operações concorrentes nos gerenciadores de bancos de dados. Seria interessante um estudo onde uma aplicação de teste execute operações multi-threads, ou seja, disparando em cada thread uma operação diferente nos SGBDs, simulando o paralelismo nas diferentes operações.

Referências

Fowler, M.; Sadalage, P. (2013) "NoSQL Distilled", Addison-Wesley Professional, 1ª edição

Jouili, S.; Vansteenberghe, V. (2013) "An empirical comparison of graph databases", Social Computing 2013 International Conference, páginas 708-715 IEEE Computer Society.

Narde, R. (2013) "A Comparison of NoSQL systems", Tese de mestrado em Ciências da Computação. Rochester Institute of Technology.

DeCandia, G.; Hastorun, D.; Jampani, M.; Kakulapati, G.; Lakshman, A.; Pilchin, A.; Sivasubramanian, S.; Vosshall, P.; Vogels,W. (2007) "Dynamo: Amazon's Highly Available Key-value Store", ACM SIGOPS Operating Systems Review, 41:6, páginas 205-220

Chang, F.; Dean J.; Guernawat, S.; Hsieh, W.; Wallach, D.; Burrows, M.; Chandra, T.; Fikes, A.; Gruber, R. (2006) "Bigtable: A distributed Storage System for Structured Data", ACM Transactions on Computer Systems, volume 26 issue 2 artigo 4

Suter, R. (2012) "MongoDB An introduction and performance analysis", Seminário do mestrado em Ciências da Computação, HSR Hochschule für Technik Rapperswil

Miller, J.. (2013) "Graph Database Applications and Concepts with Neo4j", AIS Electronic Library, SAIS 2013 Proceedings paper 24.

Kolomicenko, V. (2013) "Analysis and Experimental Comparison of Graph Databases", Tese de mestrado em Ciências da Computação. Charles University , Prague

Henricsson, R. (2011) "Document Oriented NoSQL Databases", Tese de graduação em Ciências da Computação, Blekinge Institute of Technology

Stanford, University (2015) "Stanford Large Network Dataset Collection". Disponível em: <https://snap.stanford.edu/data/#amazon>. Acesso em: 23 fev. 2015.