Análise de Interações e Padrões em Redes Sociais Acadêmicas

RELATÓRIO TÉCNICO No.5, Projeto Apoena

Financiado por CNPq/Brasil, projeto nro. 458400/2014-9.

Mariana O. S. Silva e Mirella M. Moro,
Universidade Federal de Minas Gerais, Belo Horizonte, Brasil
Contato: mariana.santos@dcc.ufmg.br, mirella@dcc.ufmg.br
URL: `http://www.dcc.ufmg.br/~mirella/projs/apoena`

25 Julho, 2017

Resumo

Devido à natureza inerente de redes sociais, acadêmicas ou não, extrair e analisar conhecimento relevante de tais redes apresentam diversos desafios para usuários, desenvolvedores e para a tecnologia. Para suportar as necessidades de aplicações específicas, bem como tarefas gerais relacionadas com a armazenamento, manutenção e análise de dados, é preciso definir uma infraestrutura para dados de redes sociais acadêmicas bem como explorá-las para obter maior conhecimento sobre as mesmas. Assim, este estudo visa especeficar e validar um modelo para as redes sociais acadêmicas, incluindo a definição de uma infraestrutura de banco de dados que permita armazenar e manter os dados das redes facilitando o uso de técnicas de análise de redes sociais. Para isso, foi escolhido um SGBD (Sistema de Gerenciamento de Bancos de Dados) próprio para dados em grafo, Neo4j, e realizada uma comparação de desempenho entre um dos SGBD relacionais mais populares, o MySQL.

Capítulo 1

Introdução

A análise de redes sociais tem se tornado um assunto extremamente abordado e relevante para a comunidade científica. Estudos focados na análise de interações entre pessoas ou organizações, bem como detectar padrões presentes nessas interações permitem prever o comportamento de uma rede e analisar diferentes aspectos da mesma. Em um contexto acadêmico, uma rede social acadêmica pode ser representada por colaborações científicas que possuem padrões e características desconhecidas em relação a interação dos indivíduos envolvidos. Tais indivíduos seriam representados por pesquisadores e os vínculos relacionais, os relacionamentos (colaborações) entre eles.

Tais redes apresentam uma estrutura grande e volumosa de dados, impossibilitando assim uma análise manual detalhada. Dessa forma, os usuários, desenvolvedores e a própria tecnologia, enfrentam diversos desafios para analisar e extrair informações ou conhecimentos relevantes de tais redes. Para os usuários, que necessitam obter conhecimento a partir de tais redes; Os desenvolvedores, que são responsáveis por criar metodologias capazes de adquirir e manipular as informações disponíveis; e para a tecnologia, que deve oferecer o suporte necessário para implementação dessas metodologias.

No contexto de Bancos de Dados (BD), desenvolver técnicas para a grande massa de dados de redes sociais é um desafio extremamente custoso. Tanto a academia quanto a indústria perceberam que as tecnologias atuais de BD não são eficientes para gerir esse grande volume de dados. Os famosos e populares modelos relacionais, embora sejam extremamente poderosos e flexíveis, começaram a enfrentar problemas de limitação de escalabilidade. Consequentemente, uma nova categoria de BD foi definida: NoSQL. Essa nova classe difere completamente dos tradicionais bancos relacionais pois não exige esquema pré-definido, relacionamento e chaves; e o mais importante: consultas com junções também não são suportadas (ou tão pouco necessárias) [2].

Considerando o contexto atual, esse trabalho propõe especificar e validar modelos para as redes sociais acadêmicas, incluindo a definição de uma infraestrutura de banco de dados que permita armazenar e manter os dados das redes bem como facilite a utilização de técnicas de análise de redes sociais. Devido à natureza inerente das redes sociais, a proposta inicial é que seja utilizado um SGBD (Sistema de Gerenciamento de Bancos de Dados) próprio para dados em grafo, utilizando e desenvolvendo tecnologias NoSQL.

O restante deste trabalho está organizado em 4 seções. A Seção 2 apresenta alguns conceitos básicos e discute trabalhos relacionados. A seção 3 detalha o modelo proposto. A seção 4 apresenta os resultados do estudo comparativo através de experimentos, enquanto a seção 5 conclui o trabalho.

Capítulo 2

Configuração Experimental

Todos os testes foram realizados em uma máquina configurada com um processador Intel Core i7 CPU @ 2.40Ghz e RAM de 16GB. Foi utilizada a versão 5.7.11 - MySQL Community Server (GPL) e a versão 1.1.8 do Neo4J 3.0.6. Para garantir a estabilidade do servidor de virtualização durante a execução dos testes, as consultas foram testadas primeiro no MySQL e depois no Neo4j.

Como o conjunto de dados utilizado nesses experimento era extremamente volumoso, foi utilizado o comando LIMIT (em ambos os bancos de dados) como um parâmetro de parada para que uma simples consulta não trave toda a aplicação. Os experimentos foram realizados utilizando-se quatro diferentes parâmetros de parada com tamanho crescente: o primeiro limitando em 10 registros, o segundo em 100 registros, o terceiro parâmetro em 1.000 registros e por último, 10.000 registros. O conjunto de dados foi coletado do banco de dados DBLP (*Digital Bibliography & Library Project*), um repositório bibliográfico de ciência da computação, no dia 16 de Setembro de 2016.

A DBLP teve seu início em 1993, evoluindo de um pequeno servidor web experimental para um serviço popular para a comunidade de ciência da Computação (Ley, 2009). Apesar de ter começado como uma pequena e limitada base de dados, esta se tornou uma biblioteca digital contendo trabalhos de quase todos os campos de estudo em computação, contando com mais de 3.5 milhões de registros. Para esse estudo, foram coletados essas publicações e seus detalhes associados. A Tabela 2.1 apresenta a estrutura do conjunto de dados e a quantidade de dados coletados.

Tabela 2.1: Descrição do conjunto de dados coletado.

Dados	**Número de registros**
Publicações em artigos	1.505.020
Autores	1.779.971
Publicações em proceedings	31.549
Publicações em inproceedings	1.861.226
Relação entre autores e publicações	9.707.161
Total	**14.884.927**

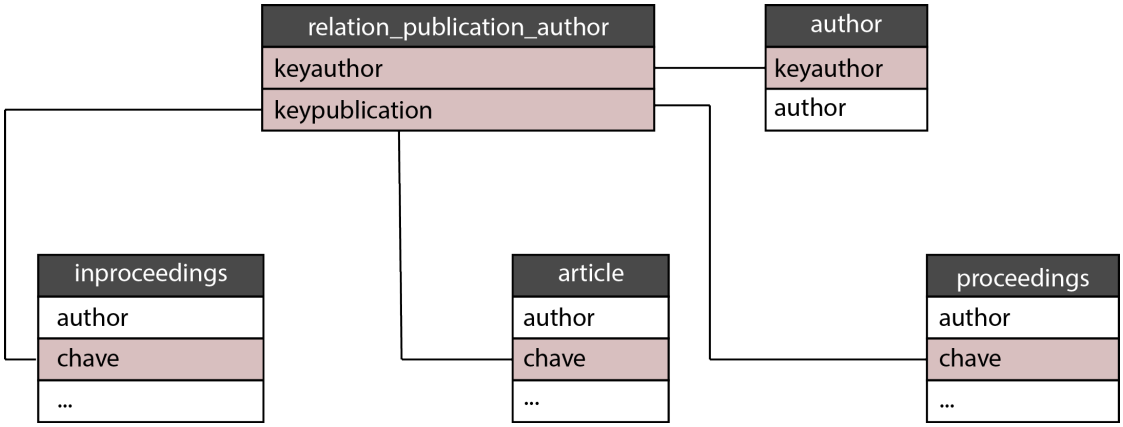


Figura 2.1: Esquema do banco de dados relacional.

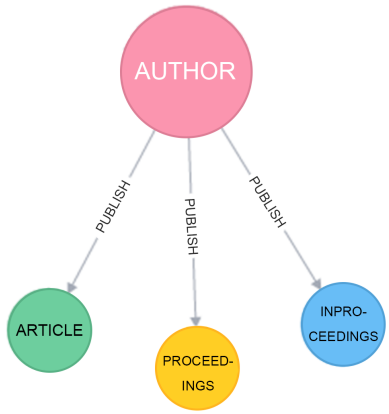


Figura 2.2: Esquema do banco de dados em grafo.

2.1 Banco de dados MySQL

O esquema de dados relacionais tinha 5 tabelas no total. O esquema conceitual da base de dados implementada é mostrado na Figura 2.1. Nenhum índice foi adicionado à implementação básica deste esquema de banco de dados com a finalidade de tentar evitar qualquer viés devido ao seu design ou implementação.

2.2 Banco de dados Neo4j

O banco de dados em grafo possui 4 tipos de nós e 1 tipo de relacionamento, *PUBLISH*. Como em bancos de dados próprios para grafos não existe um esquema pré-definido equivalente a tabelas relacionais, representamos o esquema como tipos de nó e tipos de relacionamento. Cada nó pode ter propriedades diferentes sem ter que cumprir com regras estabelecidas ou restrições de design. Além disso, cada instância de relacionamento deve ser direcional, ou seja, dependente dos nós que ele conecta. O esquema lógico do banco de dados implementado é mostrado na Figura 2.2.

2.3 Consultas

O sistema foi avaliado com 27 consultas, descritas na Tabela 2.2. No MySQL, as consultas foram implementadas com SQL enquanto no Neo4j, as consultas foram implementadas utilizando a linguagem Cypher (linguagem de consulta de gráfico declarativa para o banco de dados em grafo Neo4j). Os tempos de execução foram coletados após a execução das consultas e anotados em segundos. Cada consulta selecionada foi executada cinco vezes (com intervalos limitados de dados de 10, 100, 1000 e 10000 em alguns casos relevantes) em ambos os servidores de banco de dados. Depois, foram calculados os tempos médios das execuções para obtermos resultados mais exatos e justos. Na Seção 3, foram analisadas as consultas mais complexas dos experimentos. Essas consultas, geralmente, continham muitas operações de junção e de subconsultas. Dois exemplos de consultas complexas são a 8 e a 18, apresentadas nas Listagens 2.1 e 2.2 a seguir:

Listing 2.1: Código da consulta número 8 em SQL.

```
SELECT MAX(t1.Publications)
FROM (SELECT COUNT(*) AS Publications
FROM relation_publication_author rpa, inproceedings i
WHERE rpa.keypublication = i.chave AND YEAR(i.mdate) >= '2006') AS t1;
```

Listing 2.2: Código da consulta número 18 em SQL.

```
SELECT t1.author, MAX(t1.PUBLICATIONS)
FROM (SELECT COUNT(*) AS PUBLICATIONS, a.author
FROM relation_publication_author re, author a, inproceedings i
WHERE re.keyauthor = a.keyauthor AND i.chave = re.keypublication
GROUP BY re.keyauthor) as t1;
```

Tabela 2.2: Descrição das consultas criadas para o experimento.

Consulta	**Enunciado**
1	*Listar a quantidade de publicações por autores em inproceedings*
2	*Listar a quantidade de publicações por autores em articles*
3	*Listar a quantidade de autores por publicações em inproceedings*
4	*Listar a quantidade de autores por publicações em articles*
5	*Listar as publicações e o número de autores participantes, onde esse número é maior que dois, em articles*
6	*Listar as publicações e o número de autores participantes, onde esse número é maior que dois, em inproceedings*
7	*Listar a data da última publicação que dois autores publicaram juntos em articles*
8	*Listar o número de publicações nos últimos 10 anos em inproceedings*
9	*Listar o número de publicações nos últimos 10 anos em articles*
10	*Listar os autores que publicaram em articles em '2016'*
11	*Listar os autores que publicaram em inproceedings em '2016'*
12	*Listar a quantidade de publicações por autores em articles nos últimos 10 anos*
13	*Listar a quantidade de publicações por autores em inproceedings nos últimos 10 anos*
14	*Listar os autores que publicaram na mesma publicação X*
15	*Listar a data da primeira publicação de cada autor em articles*
16	*Listar a data da primeira publicação de cada autor em inproceedings*
17	*Listar o autor que mais publicou em articles*
18	*Listar o autor que mais publicou em inproceedings*
19	*Listar a quantidade de publicações em articles de journals que possuem IEEE no nome*
20	*Listar os autores que publicaram em inproceedings entre 2010 a 2015*
21	*Listar os autores que publicaram mais de 10 artigos em 2015*
22	*Listar todas as distintas datas de publicações em articles*
23	*Listar todas as distintas datas de publicações em inproceedings*
24	*Listar qual publicação teve mais coautores em sua publicação, em inproceedings*
25	*Listar qual publicação teve mais coautores em sua publicação, em articles*
26	*Listar a primeira publicação em articles e os seus autores*
27	*Listar a primeira publicação em inproceedings e os seus autores*

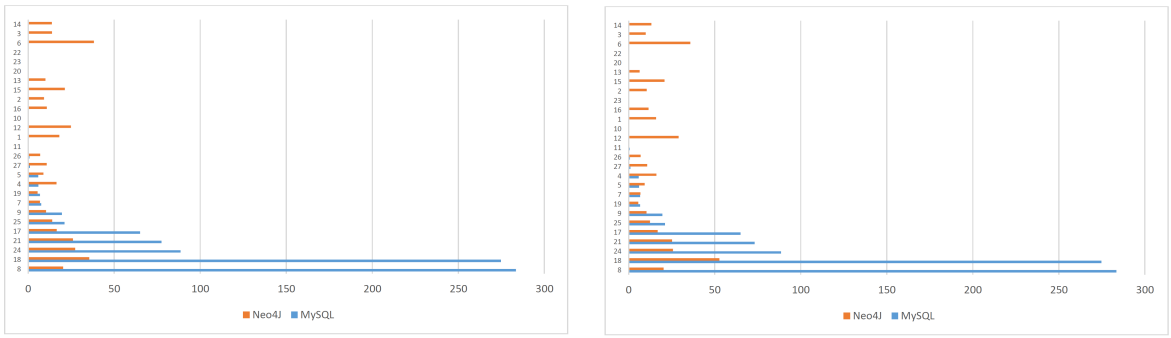
Capítulo 3

Resultados

Em seguida, serão apresentados os resultados obtidos no experimento. Pode ser facilmente observado, a partir dos valores recuperados na Tabela 3.1, que os tempos de execução das consultas rodadas no Neo4j são maiores do que no bancos de dados relacionais, na maioria dos casos. Além disso, para algumas consultas, o Neo4j apresenta um desempenho pior do que o MySQL, mesmo quando se comparam diferentes limites de dados. Por exemplo, na consulta de número 2, com um limite de 1.000 registros para o Neo4j e 10.000 para MySQL (ou seja, o MySQL às vezes supera o Neo4j mesmo se o tamanho dos dados for 10 ordens de magnitude maior).

As Figuras 3.1a, 3.1b, 3.2a e 3.2b mostram o tempo médio de resposta em segundos no MySQL e no Neo4j para cada uma das 27 consultas, nos quatro parâmetros utilizados. De modo geral, ambos os servidores de banco de dados operaram em tempos aceitáveis. Porém, podemos verificar que o banco de dados gráfico, Neo4j, retornou melhor resultado contra consultas mais complexas. Isso ocorre porque bancos de dados relacionais falham ao recuperar dados quando as aplicações exigem junções entre tabelas grandes. Em modelos relacionais, todos os dados são percorridos para localizar os registros que atendem aos critérios de pesquisa. Em consequência, quanto maior for o conjunto de dados, mais tempo será gasto para encontrar correspondências.

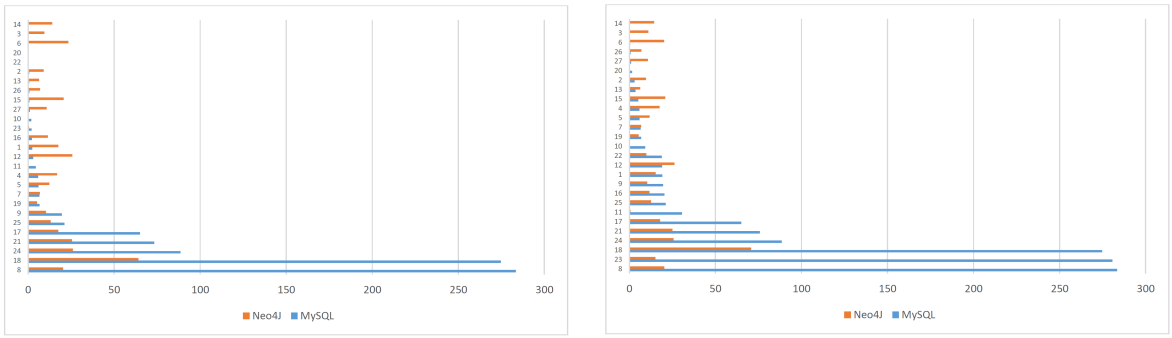
Enquanto em modelos relacionais a eficiência diminui quando o conjunto de dados é grande e as consultas são de caráter não-relacional, bancos de dados de grafos conseguem lidar facilmente com grande volume de



(a) Parâmetro limitador = 10

(b) Parâmetro limitador = 100

Figura 3.1: Desempenho das consultas em ambos os sistemas, ordenado em relação ao tempo de execução das consultas em SQL.



(a) Parâmetro limitador = 1.000

(b) Parâmetro limitador = 10.000

Figura 3.2: Desempenho das consultas em ambos os sistemas, ordenado em relação ao tempo de execução das consultas em SQL.

dados. Como eles são independentes de esquemas estáticos, eles são melhores para gerenciar dados com esquemas dinâmicos. Além disso, em modelos não relacionais orientados à grafos, não se verifica o grafo inteiro para localizar os nós que atendem aos critérios de pesquisa. Assim, quando ocorre um aumento do número de nós de cem para mil, o tempo de recuperação não sofre um aumento tão significativo, como pode ser visualizado nos gráficos abaixo.

Como foi dito anteriormente, em certas consultas, o desempenho do MySQL foi muito inferior do que o Neo4j. Mais especificamente, as consultas 8, 17, 18, 21, 24 e 25 podem ser consideradas as consultas mais divergentes. De acordo com a Tabela 2.2, essas eram consultas que demandavam operações de junção entre muitas tabelas, como também a utilização de subconsultas. Por exemplo, na Listagem 2.1 temos uma consulta utilizando subconsultas.

Para a consulta 8, o desempenho do MySQL é extremamente pior do que Neo4j, superando o Neo4j em mais de 1 ordem de magnitude. Logo, podemos dizer que esta foi provavelmente uma das consulta mais complexa do experimento, uma vez que requer a utilização de uma subconsulta contendo uma junção entre duas tabelas grandes e mais uma função *COUNT(*)*. Outra consulta que apresentou um desempenho semelhante à consulta 8 foi a 18ª consulta (Listagem 2.2). Esta também utilizou uma subconsulta contendo 2 junções e mais uma função de *COUNT(*)*.

Em resumo, se for necessário usar um volume médio de dados sem consultas muito complexas que exigem muitas junções entre relacionamentos, então o MySQL é mais adequado. Mas, se existirem muitos relacionamentos que exijam junções entre tabelas grandes, sua eficiência diminui e, muitas vezes, os bancos de dados relacionais falham ao tentar recuperar os dados.

Tabela 3.1: Resultados das consultas em segundos.

Consulta	Tempo MySQL (s)				Tempo Neo4j (s)			
	10	**100**	**1.000**	**10.000**	**10**	**100**	**1.000**	**10.000**
1	0,04814	0,29378	2,3567	19,10224	18,06	15,8778	17,4956	15,2272
2	0,01982	0,12224	0,62458	2,997	9,1838	10,3976	8,9842	9,576
3	0,0011	0,00248	0,02482	0,26588	13,8274	9,8536	9,4578	10,9494
4	5,90485	5,830153333	5,80226	5,80898	16,46415	16,0408	16,7848	17,4788
5	5,83824	6,02196	5,99906	5,92432	8,8124	9,2408	12,276	11,7392
6	0,0014	0,00338	0,02628	0,45056	38,1326	35,8132	23,3534	20,1594
7	7,49076	6,62928	6,50174	6,48058	6,8094	6,8094	6,8094	6,8094
8	283,40654	283,40654	283,40654	283,40654	20,2904	20,2904	20,2904	20,2904
9	19,51824	19,51824	19,51824	19,51824	10,337	10,337	10,337	10,337
10	0,03312	0,30228	1,77496	9,19758	0,0538	0,0298	0,0594	0,3812
11	0,05886	0,60354	4,4458	30,5377	0,0132	0,0168	0,066	0,6402
12	0,03662	0,36674	2,92878	18,99116	24,7794	28,9672	25,651	26,1602
13	0,00866	0,07538	0,65352	3,50066	9,9774	6,31	6,3262	6,2894
14	0,00054	0,00052	0,00054	0,00212	13,733	13,1662	14,0024	14,3198
15	0,00878	0,07932	0,73422	5,19714	21,2852	20,7702	20,631	20,7632
16	0,02206	0,22798	2,19358	20,3836	10,8172	11,5378	11,4624	11,6338
17	64,96422	64,96422	64,96422	64,96422	16,546	16,8108	17,5234	17,825
18	274,73484	274,73484	274,73484	274,73484	35,4496	52,7118	64,0558	70,7026
19	6,76454	6,64344	6,6344	6,83144	5,4402	5,5416	5,119	5,3598
20	0,00534	0,01948	0,14284	1,49508	0,014	0,013	0,0302	0,1978
21	77,42318	73,14518	73,2188	75,8049	25,9756	25,1276	25,3974	24,951
22	0,00248	0,01538	0,23618	18,81344	0,0114	0,0238	0,2644	9,8106
23	0,00338	0,14078	2,00796	280,66276	0,0096	0,0458	0,503	15,0604
24	88,527	88,527	88,527	88,527	27,2495	25,663	26,0085	25,64725
25	21,05124	21,05124	21,05124	21,05124	13,861	12,3445	13,05825	12,525
26	0,67194	0,74574	0,67656	0,67118	6,93225	6,93225	6,93225	6,93225
27	0,99894	1,0006	0,98658	0,9657	10,73725	10,73725	10,73725	10,73725

Capítulo 4

Trabalhos Relacionados

Bancos de dados relacionais têm oferecido eficiente suporte de armazenamento de dados por muitas décadas. Mas no contexto atual, esses sistemas relacionais não estão sendo mais tão eficazes quando é necessário a realização de muitas operações em grandes conjuntos de dados. Por exemplo, uma rede de hiperlinks conectando todas as páginas na World Wide Web (WWW) é altamente complexa e quase impossível de modelar de forma eficiente em um banco de dados relacional [1, 8]. Esses problemas também podem ser observados na modelagem de redes sociais. A implementação de tais problemas em bancos de dados relacionais envolve grande número de junções, o que é caro de ser calculado [5].

Com o aumento intenso do volume de dados, houve um claro desejo de um armazenamento adaptado às necessidades de dados em grafo. Os bancos de dados orientados a grafos processam com eficiência densos conjuntos de dados, e o seu design permite a construção de modelos preditivos e análise de correlações e padrões de dados. De acordo com Hwang et al. [4], este modelo, onde todos os nós estão ligados por relações, permite travessias rápidas entre os vértices ao longo das arestas. Ou seja, permite recuperar a informação a partir de um grafo rapidamente.

Enquanto os bancos de dados NoSQL têm a vantagem de velocidade e escalabilidade, há uma série de desvantagens em relação aos bancos de dados relacionais tradicionais [7, 6]. Segundo Leavitt, os bancos de dados NoSQL, embora apresentem um bom desempenho para consultas simples, eles consomem tempo para operações mais complexas. Além disso, consultas para tais operações complexas podem ser complicadas de implementar.

Em [7], foi comparado o desempenho de alguns bancos de dados NoSQL e SQL, utilizando operações de leitura, escrita, exclusão e instanciação em Banco de dados que trabalham no esquema chave/valor. De acordo com os pesquisadores, nem todos os bancos de dados NoSQL funcionam melhor do que os bancos de dados SQL. E para cada banco de dados, o desempenho varia com cada operação. Alguns são lentos para instanciar, mas rápido para ler, escrever e excluir. Outros são rápidos para instanciar, mas lento nas outras operações. Além disso, há pouca correlação entre o desempenho e o modelo de dados que cada banco de dados usa.

[1] fornecem uma análise comparativa mais específica, entre o Neo4j e o banco de dados mais difundido, o MySQL. Foi apontado que ambos os sistemas obtiveram bons resultados nos benchmarks objetivos e subjetivos. No entanto, o Neo4j obteve melhores resultados quando foram realizados testes objetivos. Ou seja, bancos de dados orientados à grafos recuperam os resultados do conjunto de consultas predefinidas mais rapidamente do que os bancos de dados relacionais. Além disso, os bancos de dados de grafos possuem

uma estrutura mais flexível, pois novos relacionamentos podem ser adicionados ao banco de dados sem a necessidade de reestruturar o esquema novamente.

Outra pesquisa comparativa [3], entre Neo4j e o MySQL, revelou que bancos de dados relacionais são melhores para aplicativos com um número pequeno e fixo de relacionamentos. Tais sistemas relacionais não suportam aplicações que exigem alterações contínuas do esquema de banco de dados, por exemplo, redes sociais. Outro ponto analisado por [3] foi que bancos de dados orientados por grafos podem facilmente lidar com grandes registros de dados, pois eles não exigem junções custosas. Como eles não dependem de um esquema estático, eles são melhores para gerenciar dados com modelagem dinâmica.

Nesta seção, foram discutidos alguns estudos comparativos entre bancos de dados relacionais e não relacionais. Cada um deles abrange contextos distintos, resultando em obeservações contrárias em alguns casos. Considerando o contexto desta pesquisa, análise de redes sociais acadêmicas, a revisão bibliográfica não apresentou estudos que fossem semelhantes ao cenário abordado. Portanto, nossa contribuição neste trabalho é uma abordagem prática na comparação dos bancos de dados relacional e não relacional (MySQL e Neo4j), dentro do contexto tratado na pesquisa.

Capítulo 5

Conclusão

Nessa pesquisa, foi comparado o desempenho de um banco de dados relacional (implementado no MySQL) e um banco de dados em grafos (implementado em Neo4j), no contexto de uma rede social de coautoria acadêmica. A comparação abrangeu 27 consultas e quatro parâmetros limitadores de dados. Os resultados dos experimentos indicam que o MySQL apresentou um melhor desempenho do que o Neo4j na maioria dos casos, mas o Neo4j supera o MySQL em 8 consultas que exigiam múltiplas operações de junção ou a utilização de subconsultas.

Foi possível concluir que o banco de dados relacional é mais indicado para aplicativos que têm um número pequeno e fixo de relacionamentos e não suporta aplicações que exigem alterações contínuas do esquema de banco de dados, por exemplo, gerenciamento de redes sociais. Por outro lado, os bancos de dados de grafos, especificamente o Neo4j, são uma opção melhor para a análise de redes sociais (acadêmicas ou não).

Pesquisas futuras incluem realizar a mesma comparação de desempenho utilizando uma rede do GitHub com diversos tipos de relacionamento, formando uma rede híbrida.

Agradecimentos. Trabalho parcialmente financiado por CNPq e FAPEMIG.

Referências Bibliográficas

[1] S. Batra and C. Tyagi. Comparative analysis of relational and graph databases. *International Journal of Soft Computing and Engineering (IJSCE)*, 2(2):509–512, 2012.

[2] R. Cattell. Scalable sql and nosql data stores. *Acm Sigmod Record*, 39(4):12–27, 2011.

[3] R. Fatima and A. Ahmed. Role of graph databases in social networking sites: A performance comparison between graph database neo4j and relational database mysql in social networking sites. *Journal of Independent Studies and Research*, 10(2):22, 2012.

[4] J. S. Hwang, S. Lee, Y. Lee, and S. Park. A selection method of database system in bigdata environment: A case study from smart education service in korea. *International Journal Advance Soft Computing Application*, 7(1):9–21, 2015.

[5] M. Kleppmann. Should you go beyond relational databases. *Retrieved March*, 2:2010, 2009.

[6] N. Leavitt. Will nosql databases live up to their promise? *Computer*, 43(2):12–14, 2010.

[7] Y. Li and S. Manoharan. A performance comparison of sql and nosql databases. In *Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on*, pages 15–19. IEEE, 2013.

[8] J. McKendrick. Make room for the monster databases. *Database trends and applications*, 15:12, 2002.