Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Curso de Ciência da Computação

AITAN VIEGAS PONTES

SISTEMA DE RECOMENDAÇÃO BASEADO EM BANCO DE DADOS ORIENTADO A GRAFOS

São Luís
2017

AITAN VIEGAS PONTES

SISTEMA DE RECOMENDAÇÃO BASEADO EM BANCO DE DADOS ORIENTADO A GRAFOS

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Profº Me. Carlos Eduardo Portela Serra de Castro

São Luís

2017

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Pontes, Aitan Viegas.
Sistema de recomendação baseado em banco de dados orientado a grafos / Aitan Viegas Pontes. – 2017.
41 f.

Coorientador(a): Steve Ataky Tsham Mpinda.
Orientador(a): Carlos Eduardo Portela Serra de Castro.
Monografia (Graduação) – Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís, 2017.

1. Banco de dados grafos. 2. Cypher. 3. Linguagem de consulta. 4. Neo4j. 5. Sistema de recomendação. I. Castro, Carlos Eduardo Portela Serra de. II. Mpinda, Steve Ataky Tsham. III. Título.

AITAN VIEGAS PONTES

SISTEMA DE RECOMENDAÇÃO BASEADO EM BANCO DE DADOS ORIENTADO A GRAFOS

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 01 de Fevereiro de 2017

BANCA EXAMINADORA

Profº Me. Carlos Eduardo Portela Serra de Castro (Orientador)
Universidade Federal do Maranhão

Profº Me. Steve Ataky Tsham Mpinda
(Coorientador)
Unidade de Ensino Superior Dom Bosco

Profº Dr. Samyr Beliche Vale
Universidade Federal do Maranhão

Profº Dr. Carlos de Salles Soares Neto
Universidade Federal do Maranhão

São Luís
2017

A minha mãe Ildeci, com toda minha gratidão, por tudo que fez por mim ao longo da minha vida.

Ao meu eterno amor, Flaviane, pelo incentivo, paciência e dedicação comigo.

Agradecimentos

Agradeço primeiramente a Deus pelo dom da vida, saúde e sabedoria para chegar até aqui;

Aos meus pais, José Ribamar(in memorian) e Ildeci Viegas, por sempre acreditarem em mim e me mostrarem que a educação é a ferramenta transformadora da vida e realização de sonhos. Em especial a minha mãe que nunca mediu esforços para me oferecer um ensino de qualidade e por sempre me perguntar pela minha formatura, espero agora poder lhe dar a resposta que tanto aguarda;

A minha esposa, Flaviane, por nunca deixar eu desistir, por ouvir minhas angústias e sempre responder com palavras confortadoras e de incentivo que me motivaram a continuar, além das incontáveis correções textuais;

A minha filha, Alice, por ser minha fonte de alegria e motivação com seus sorrisos, brincadeiras e falando coisas que ninguém entende.

Ao meu irmão, Igor, pelo apoio, por todo companheirismo, sempre tentando me ajudar e por ser alguém que sempre posso contar.

Ao meu orientador, Carlos Eduardo Portela, pela oportunidade concedida, pela paciência em corrigir meus textos, por responder sempre com celeridade a todos meus questionamentos e por todo o apoio dado;

Ao meu coorientador e grande amigo, Steve Ataky, pela indicação do tema que deu origem a este trabalho e por se colocar sempre à diposição.

À todos os meus amigos que estiveram ao meu lado durante esta jornada, me apoiando e me proporcionando momentos de descontração e diversão. Um agradecimento especial aos companheiros da connection por sempre inventarem um motivo para comemoração;

Aos amigos e colegas de curso, verdadeiros parceiros, e à todos os professores que fizeram parte de toda essa trajetória.

RESUMO

Sistemas de recomendações são sistemas computacionais que tem o objetivo de auxiliar os usuários a encontrarem as informações que realmente desejam, pois com a popularização da internet os dados tem crescido exponencialmente e a tarefa de encontrar informações relevantes, em meio a tantos dados, se torna árdua. O grande volume de dados produzidos tem se tornado um problema até mesmo para os sistemas computacionais, que precisam processar imensas quantidades de dados e transforma-las em informações relevantes aos usuários em um tempo hábil.

Os bancos de dados grafos começaram a ser mais explorados recentemente por apresentarem bons desempenho no processamento de grandes volumes de dados conectados. Desta forma, uma abordagem utilizando banco de dados grafos para o desenvolvimento de sistemas de recomendação parece promissora. Este trabalho apresenta os principais conceitos a respeito de sistemas de recomendações e banco de dados grafos, assim como, propõe técnicas de recomendação utilizando banco de dados grafos que são testadas em um estudo de caso de recomendação de filmes utilizando a linguagem de consulta Cypher e o SGBD Neo4j.

Palavras-chave: Sistema de Recomendação, Banco de Dados Grafos, Linguagem de Consulta, Neo4J, Cypher.

ABSTRACT

Recommendation systems are computer systems that aim at helping users finding the desired information, since with the popularization of the Internet, data have grown exponentially and the task of finding relevant information within so much data becomes over and over again hard. The huge volume of data produced has become a problem even for computing systems, which need to process huge amounts of data and turn it into relevant information to users in a timely manner.

Graph databases have risen and started to be more exploited recently for they present a noteworthy performance in processing large volumes of connected data. Thereby, an approach using graph databases for the development of recommendation systems seems promising. This work presents the main concepts apropos of recommendations systems and graph databases, as well as proposes techniques of recommendation using graph databases, which are validated by means of a case study vis-a-vis recommendation of movies using the Cypher query language and the DBMS Neo4j.

Keywords: Recommendation Systems, Graph Database, Query Language, Neo4j, Cypher.

Lista de ilustrações

Lista de listagens

Lista de tabelas

Lista de abreviaturas e siglas

SR	Sistema de Recomendação
FBC	Filtragem Baseada em Conteúdo
FC	Filtragem Colaborativa
BD	Banco de Dados
BDG	Banco de Dados Grafos
SGBD	Sistemas Gerenciadores de Banco de Dados
SQL	Structured Query Language
NoSQL	Not Only SQL

Sumário

1 Introdução

O crescimento exponencial dos dados da internet geram uma sobrecarga de informação para os usuários, que encaram um verdadeiro desafio para encontrar aquilo que realmente buscam. Até mesmo a escolha de um filme para assistir no fim de semana pode se tornar uma árdua tarefa, considerando por exemplo que o aplicativo de filmes *online* Netflix possui cerca de 17000 filmes disponíveis.

As pessoas já se utilizam de recomendações há muito tempo para apoiar suas decisões e descoberta de novos materiais. Normalmente quando alguém deseja obter um produto ou serviço discute com familiares e amigos ou consulta a opnião de especialistas, podemos citar como exemplo quando alguém vai a biblioteca e pede uma recomendação ao bibliotecário. Mas esse método de recomendação tem seus limites, especialmente para a descoberta de novos conteúdos. Pode existir um livro que a pessoa gostaria de ler, mas ninguém em seu círculo tem conhecimento sobre o mesmo.

Para (EKSTRAND; KONSTAN, 2010), sistemas computacionais fornecem a oportunidade de expandir o círculo de pessoas do qual o usuário pode receber recomendações. Eles também possibilitam a análise do histórico do usuário e a definição de padrões de preferências que nem os usuários e nem seus conhecidos conseguem identificar, fornecendo uma experiência de seleção mais bem afinada. Esses são os chamados sistemas de recomendação.

Segundo (CUNG; JEDIDI, 2014), um modelo de dados em grafo, apesar de ser algo relativamente novo, possui muitas soluções para melhorar aplicações que tenham uma enorme quantidade de dados com um elevado grau de correlação. A principal vantagem dos modelos de dados grafos é que o relacionamento/ligação entre os elementos/entidades é o núcleo do modelo. Grafos também tem um elevado grau de flexibilidade e escalabilidade. Isso significa que, em uma era, onde a quantidade de dados é tão grande e onde a internet está enfrentando uma sobrecarga, bases de dados grafos propõem algumas soluções para gerenciar e armazenar esses dados.

O modelo de dados grafo demonstra ser uma abordagem promissora para implementação de aplicações de sistema de recomendação, tanto por sua flexibilidade e escalabilidade quanto pela sua capacidade de descrever relacionamentos, elementos essenciais nesse tipo de sistema. Assim este trabalho visa demonstrar as vantagens da utilização de banco de dados grafos na implementação de um sistema de recomendação.

1.1 Objetivos

O objetivo geral desta monografia é propor duas técnicas de recomendação baseada em banco de dados orientados a grafos. Os objetivos específicos são:

- Realizar uma revisão sistemática da literatura concernente ao estado da arte de banco de dados orientado a grafos, linguagem Cypher e sistema de recomendação;
- Apresentar os principais conceitos sobre sistemas de recomendação e de modelagem orientada a grafos: construção, relacionamentos, propriedades, utilização, armazenamento e gerenciamento;
- Realizar um estudo de caso de um sistema de recomendação de filmes como forma de consolidar os conhecimentos adquiridos.

1.2 Organização do Trabalho

Este trabalho está organizado da seguinte forma: O Capítulo 2 apresenta os principais conceitos e tecnologias, a cerca de sistemas de recomendação, banco de dados grafos e linguagem de consulta a grafos. O capítulo 3 apresenta as duas técnicas propostas para recomendação utilizando filtragem colaborativa. O Capítulo 4 apresenta um estudo de caso sobre sistema de recomendação de filmes utilizando as metodologias propostas. O Capítulo 5 apresenta as conclusões sobre o trabalho apresentado e os trabalhos futuros.

2 Referencial Teórico

Neste capítulo são descritos conceitos sobre sistemas de recomendação, grafos, sistemas de banco de dados grafos e linguagem de consulta a grafos.

2.1 Sistema de recomendação

Sistemas de recomendação tem mudado a forma como as pessoas encontram produtos, informações e até mesmo outras pessoas. Eles são sistemas de apoio a decisão e descoberta de novos conteúdos, o qual através do estudo de padrões de comportamentos descobrem o que alguém vai preferir dentre uma coleção de dados pela qual ele nunca experimentou.

Recursos computacionais que indicam livros-filmes-músicas (itens) a potenciais leitores-espectadores-ouvintes (usuários) e sugerem relações de amizades em redes sociais são exemplos de sistemas de recomendação populares atualmente (GOLDSCHMIDT; BEZERRA, 2015).

O primeiro sistema de recomendação, Tapestry, foi desenvolvido pela Xerox Palo Alto Research Center no início dos anos 90. A motivação para o desenvolvimento do Tapestry foi o aumento do uso de correio eletrônico, que resultou em usuários sendo inundados por um enorme fluxo de entrada de documentos (GOLDBERG DAVID NICHOLS; TERRY, 1992).

Na tentativa de solucionar o problema, os proponentes do Tapestry indicaram algumas abordagens, na mais promissora, cada usuário definia um filtro pessoal, recebendo assim apenas os *emails* que fossem capturados por este filtro, além disso, os usuários poderiam ajudar outros usuários a fazer a filtragem, gravando suas impressões aos documentos lidos, o que ficou conhecido como filtragem colaborativa (COSTA, 2009). Devido a isso vários pesquisadores adotaram o termo filtragem colaborativa para designar os sistemas de recomendação que foram surgindo. A terminologia como é conhecida atualmente surgiu pela primeira vez no trabalho (RESNICK, 1997) de Resnick, onde ele defendia que sistemas de recomendação era um termo mais adequado para definir estes tipos de sistema, já que os mesmos poderiam existir sem a colaboração humana.

No final dos anos 90, os primeiros sistemas de recomendação comerciais começam a surgir. O mais conhecido foi o sistema de recomendação da Amazon.com, baseado em dados de histórico de compras, histórico de navegação e itens em visualização corrente. Outros sistemas de *e-commerce* rapidamente seguiram a nova

tendência (EKSTRAND; KONSTAN, 2010). Desde então, os sistemas de recomendação se tornaram essenciais em lojas virtuais, pois eles podem impulsionar as vendas; promover artigos impopulares que podem ser altamente interessantes a determinados usuários; aumentar a satisfação do usuário; ajudar a compreender melhor os desejos do usuário e fidelizar clientes (RICCI LIOR ROKACH; KANTOR, 2011).

Uma explosão de pesquisas nesta área começaram a surgir pelo aumento da disponibilidade de dados e interesses comerciais. Um evento crítico foi o anúncio do prêmio Netflix. Em 2006, a empresa lançou o prêmio de 1 milhão de dólares para o melhor algoritmo de filtragem colaborativa, o qual deveria realizar melhores recomendações que o sistema utilizado por ela. Cerca de 2000 times submeteram seus sistemas de recomendação (SR) e 650 obtiveram resultados que ultrapassaram até 5 por cento a precisão do programa do Netflix. O vencedor foi o time BellKor's Pragmatic Chaos o qual superou em 10.6 por cento o algoritmo da maior provedora de filmes e série de televisão via streamming (SKRASEK, 2015).

2.1.1 Problema da Recomendação

Informalmente, o problema da recomendação pode ser reduzido a adversidade de estimar classificações para itens que ainda não foram vistos pelo usuário. Intuitivamente, esta estimativa é usualmente baseada em classificações dadas por este usuário a outros itens (ADOMAVICIUS; TUZHILIN, 2005).

Segundo (SARWAR GEORGE KARYPIS; RIEDL, 2001), a tarefa da *recomendação* é gerar uma lista com os itens melhores avaliados para o usuário. Enquanto que, a tarefa individual de estimar avaliação para um único item é conhecida como *predição*.

(ADOMAVICIUS; TUZHILIN, 2005) propôs uma definição formal para o problema de recomendação que é apresentada a seguir:

> Definição: Seja C o conjunto de todos os usuários, e seja S o conjunto de todos os possíveis itens que podem ser recomendados, tais como livros, filmes ou restaurantes. O espaço S dos possíveis itens pode ser muito grande, contendo centenas de milhares ou até mesmo milhões de itens em algumas aplicações. Similarmente o espaço dos usuários também pode ser muito grande. Sendo u uma função de utilidade que mede a utilidade de um item s para um usuário $c, i.e, u : CxS \rightarrow R$, onde R é um conjunto totalmente ordenado. Então para cada usuário $c \in C$, nós queremos escolher o item $s' \in S$ que maximiza a utilidade do usuário. Mais formalmente:
>
> $$\forall c \in C, s'_c = \underset{s \in S}{\operatorname{argmax}} u(c, s) \tag{2.1}$$

Em sistemas de recomendação, a utilidade de um item normalmente é representada por uma nota. No site Netflix, os usuários podem classificar os filmes com notas de um (ruim) a cinco (bom) estrelas (MEDEIROS, 2013).

Os dados de um sistema de recomendação podem ser representados como uma matriz de utilidade X com n usuários e m itens, onde cada célula $X_{c,s}$ corresponde a avaliação dada por um usuário a um determinado item. Assume-se que a matriz é esparsa, o que significa que a maioria das entradas são desconhecidas, afinal os usuários avaliam apenas um pequeno subconjunto de itens. Desta forma, o problema central dos sistemas de recomendação é extrapolar a função de utilidade para todo o conjunto CxS.

Na Figura 1, temos um exemplo de matriz de utilidade que representa avaliações de usuários a determinados filmes(itens), com notas variando de 1 a 5. Os filmes são representados por F1 a F5, enquanto que os usuários pelas letras de A a D. Podemos observar na matriz que algumas células estão em branco, tendo como significado o fato que o usuário não avaliou esse item. O objetivo de um sistema de recomendação é predizer os espaços em branco da matriz de utilidade.

		Itens				
		F1	**F2**	**F3**	**F4**	**F5**
Usuários	**A**	4		5		
	B		5		3	
	C	2		4		5
	D	1		3		4

Figura 1 – Matriz de avaliações de usuários a filmes. Cada célula representa a nota dada pelo usuário ao item

Fonte: Elaborado pelo autor

Vários métodos foram propostos na literatura para estimar notas para itens não avaliados pelo usuário, ou seja, para preencher os espaços em branco da matriz de utilidade. Os sistemas de recomendação são usualmente classificados de acordo com a abordagem utilizada para realizar essas estimativas. Na próxima sessão veremos algumas dessas classificações.

2.1.2 Classificação dos Sistemas de Recomendação

Segundo (LESKOVEC; ULLMAN, 2010), sistemas de recomendação utilizam um vasto número de diferentes tecnologias. Pondendo ser classificados em três grupos de abordagens.

2.1.2.1 Filtragem Baseada em Conteúdo

Em métodos de recomendação baseados em conteúdo, a utilidade $u(c, s)$ do item s para o usuário c é estimada baseada nas utilidades $u(c, s_i)$ atribuídas pelo usuário c para itens $s_i \in S$ que são similar ao item s (ADOMAVICIUS; TUZHILIN, 2005). Esse método analisa as propriedades dos itens e tenta achar semelhanças entre os classificados com maiores notas pelo usuário, desta forma criando uma espécie de perfil que é atualizado a cada *feedback* do usuário a um item. Por exemplo, se um usuário do Netflix assiste e dá boas notas a muitos filmes de faroeste, então o sistema irá recomendar filmes da sua base de dados classificados com o gênero faroeste(LESKOVEC; ULLMAN, 2010).

As propriedades do item ou perfil do item é um conjunto de atributos que o caracterizam comumente representado como um vetor de palavras chaves que podem descrevê-lo. A extração de características dos itens pode ser realizada automaticamente utilizando técnicas de recuperação de informação, porém essas técnicas funcionam bem quando utilizadas para extração de características em documentos textuais e apresenta problemas com outros domínios, como quando aplicado em dados multimídias.

Um problema apresentado nos sistemas baseados em conteúdo é a super especialização, pois como o sistema apenas recomenda aqueles itens que se encaixam no perfil do usuário, ele tende a recomendar itens muito similares aos já avaliados. No exemplo do filme citado anteriormente, será recomendado apenas filmes de faroeste, mesmo o usuário gostando também de comédia. A novidade é algo desejável em sistemas de recomendação e alguns estudos propõe formas de solucionar este problema (SHETH; MAES, 1993) (ZHANG; MINKA, 2002).

Um outro problema enfrentado por esta abordagem é a precisão da recomendação para novos usuários, pois os mesmos precisam avaliar uma quantidade significativa de itens para que o sistema possa aprender suas preferências e assim recomendar itens relevantes. Com poucos itens avaliados o sistema fica propenso a recomendar itens inadequados ao gosto dos usuários.

2.1.2.2 Filtragem Colaborativa

Filtragem colaborativa é uma das técnicas mais utilizadas para algoritmos de recomendação, ela baseia suas predições e recomendações em avaliações ou comportamentos de outros usuários no sistema. Intuitivamente, o sistema assume que se os usuários concordam com a qualidade e relevância dos mesmos itens, ou seja, avaliam de forma parecida os mesmos itens, então eles irão concorda também em outros itens(EKSTRAND; KONSTAN, 2010). Por exemplo, se Maria gosta dos mesmos itens que um um grupo de usuários, então para aqueles itens que ela ainda não viu,

mas que o grupo de usuários avaliou positivamente, o sistema assume que ela também irá gostar.

Mais formalmente, (ADOMAVICIUS; TUZHILIN, 2005) afirma que a utilidade $u(c, s)$ do item s para o usuário c é estimada baseada em utilidades $u(c_j, s)$ atribuídas para o item s por aqueles usuários $c_j \in C$ que são similares ao usuário c.

O sistema monitora os usuários para encontrar padrões de comportamentos que contribuam para o cálculo de similaridade entre eles. Embora isso seja realizado apenas para fazer recomendações, nem todos os usuários se sentem confortável com o monitoramento de seus históricos de navegação e acessam o sistema como usuários não registrados ou desabilitam a execução de cookies, desta forma, dificultando a extração de informações de comportamentos importantes para a precisão das predições (HU, 2012).

Nesta abordagem possui algumas limitações, uma delas são os novos usuários, que assim como no FBC precisam avaliar uma quantidade substancial de itens para que o sistema possa aprender suas preferências. E o outro problema é dos novos itens, pois na filtragem colaborativa o sistema baseia-se unicamente nas preferências dos usuários para fazer recomendações. Portanto, até que o novo item seja avaliado por um número significativo de usuários, ele não será recomendado pelo sistema(ADOMAVICIUS; TUZHILIN, 2005).

Esses problemas podem ser resolvidos utilizando uma abordagem híbrida que combina as técnicas de filtragem baseada em conteúdo e a colaborativa.

2.1.2.3 Filtragem Híbrida

Afim de evitar os problemas causados pelas limitações das abordagem FBC e FC, vários sistemas de recomendação utilizam a abordagem híbrida que é a combinação das duas anteriores, desta forma, uma complementa a outra.

De várias formas, as abordagens colaborativas e baseadas em conteúdo fornecem capacidades complementares. Métodos colaborativos são melhores para recomendar itens razoavelmente bem conhecidos para usuários em comunidades de gostos similares. Os métodos baseado em conteúdo são bons para recomendações de itens não populares para usuários de gostos únicos(MOONEY; ROY, 2000).

2.2 Modelo de Dados Grafos

Modelagem é uma atividade de abstração motivada por uma necessidade particular ou objetivo. modela-se a fim de representar um domínio em um espaço onde ele pode ser estruturado e manipulado. Não existem representações naturais

do mundo tal como ele realmente é, apenas várias formas de abstrações e simplificações, algumas das quais são mais utilizadas que outras para satisfazer um objetivo particular(ELMASRI; NAVATHE, 2011).

Um modelo de banco de dados é uma coleção de conceitos que podem ser usados para descrever a estrutura de banco de dados. Com estrutura de banco de dados, queremos dizer os tipos, restrições e um conjunto de operações básicas que se aplicam aos dados (ELMASRI; NAVATHE, 2011).

Ao longo do tempo os modelos de banco de dado sofreram evoluções e vários deles foram propostos com os avanços das pesquisas nesta área. Cada modelo de BD é baseado em algum princípio teoríco e esses princípios desempenham um papel importante no desenvolvimento dos modelos. Antes dos modelos de banco de dados relacionais, o principal foco era o sistema de arquivo(KALIYAR, 2015).

O modelo de banco de dados relacional foi introduzido por Codd (CODD, 1970) (CODD, 1983) e destaca o conceito de abstração de níveis apresentado a ideia de separação entre os níveis físico e lógico. É baseado nos conceitos de conjuntos e relações (ANGLES; GUTIERREZ, 2008).

O modelo de BD orientado a objetos foi apresentado por Kim (KIM, 1990) em 1990 e é baseado no paradigma orientado a objetos, tendo como objetivo representar dados como coleção de objetos. Modelos de banco de dados grafos surgiram na mesma época (KALIYAR, 2015). Esses modelos tentam superar as limitações impostas pelo modelo tradicional, principalmente no que diz respeito a captura de estruturas de grafos inerentes aos dados presentes em aplicações como *hypertext* ou sistemas de informações geográficas, onde a interconectividade dos dados é um aspecto importante (ANGLES; GUTIERREZ, 2008).

Uma grande vantagem da Representação em grafos em relação as demais técnicas de modelagem de dados é a estreita afinidade entre os modelos lógicos e físicos. A representação de um domínio na modelagem relacional possui um considerável distanciamento da linguagem natural, pois primeiro é realizada a representação em um modelo lógico que depois é convertido para um modelo físico. Essas transformações introduzem dissonância semântica entre a conceitualização do mundo e a instância desse modelo no banco de dados. Com banco de dados grafos, essa lacuna é reduzida consideravelmente(ROBINSON; EIFREM, 2015).

Não é necessário ser um profundo conhecedor das teorias dos grafos para obter as vantagens dos banco de dados grafos, apenas saber o que é um grafo já é suficiente. A próxima sessão mostra de forma sucinta os principais conceitos de grafos.

2.2.1 Conceito de Grafos

Um grafo $G(V, E)$ é definido pelo par de conjuntos V e E, onde V é um conjunto de vértices e E um conjunto de arestas, nas quais são pares dos vértices em V. Se dois vértices a e b são pontos finais de uma aresta, dize-se então que eles são adjacentes(vizinhos). Se o vértice a é um dos pontos finais de uma aresta e, dize-se que e incide em a. O nível de um vértice é o número de arestas incidentes sobre ele (ZADEH, 2016).

O tamanho do grafo é dado pelo número de vértices que possui. Usualmente denota-se o número de vértices com n e o número de arestas com m.

Para a modelagem de dados considera-se que os vértices ou nós dos grafos representam as entidades do mundo real e as arestas que ligam esses vértices como relacionamentos.

Na figura 2, é apresentado um grafo que possui como vértices A, B, C e D. As arestas são representadas pelas linhas rotuladas com a combinação das letras em minúsculo dos vértices que elas conectam. Pode-se afirmar que o grafo é de tamanho 4, pois possui 4 vértices.

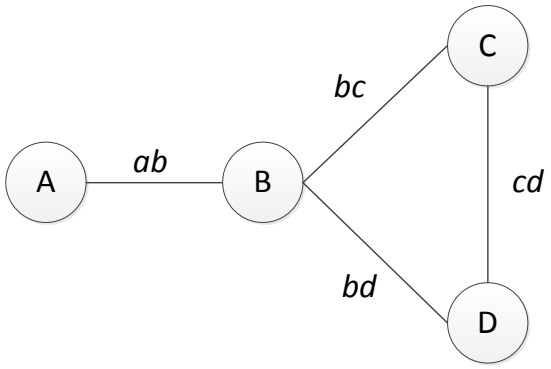

Figura 2 – Grafo de vértices (A,B,C e D) e arestas (ab, bc, bd e cd)

Fonte: Elaborada pelo autor

Existem variadas formas de representação de um grafo. Para a modelagem de dados a mais popular é a de Grafo Proprietário Rotulado do inglês *Labeled Property Graph Model*, de forma reduzida Grafo Proprietário ou *Property Graph*, que será melhor discutida na sessão a seguir.

2.2.1.1 Grafo Proprietário

Segundo (ROBINSON; EIFREM, 2015), esse modelo de grafo é composto de nós, relacionamentos, propriedades e rótulos.

- Os nós contém propriedades que são armazenadas na forma de chave-valor, algo semelhante aos atributos de uma classe em java.
- O nós podem ser marcados com um ou mais rótulos, grupos de rótulos agrupam os nós e indicam os papéis que eles desempenham dentro do conjunto de dados.
- Relacionamentos conectam nós e estrutura o grafo. Um relacionamento sempre tem uma direção, um único nome, um nó inicial e final. Direção e nome em relacionametos acrescentam clareza semântica.
- Assim como os nós os relacionamentos também podem possuir propriedades. A capacidade de adicionar propriedades aos relacionamentos é particularmente útil para fornecimento de metadados adicionais para algoritmos de grafos, acrescentando semântica adicional as relações(como qualidade e peso), e a fim de restringir consultas em tempo de execução.

A Figura 3 é um exemplo de grafo proprietário, que é utilizado para descrever uma pequena rede social. Cada nó possui o rótulo Usuário que indica seu papel no modelo e o nome do usuário é armazenado na propriedade *nome*, em cada nó. Os relacionamentos conectam dois nós e são nomeados com o rótulo AMIGO_DE, indicando que um usuário é amigo de outro na rede social. O relacionamento ainda possui a propriedade *desde*, que armazena o início da amizade. Pode-se notar nesta pequena rede social que Alice e Bob são amigos mutuamente, assim como, Bob e Bia. Mas não há reciprocidade na amizade de Bia por Alice, pois Bia é amiga de Alice, mas Alice ainda não é amiga de Bia.

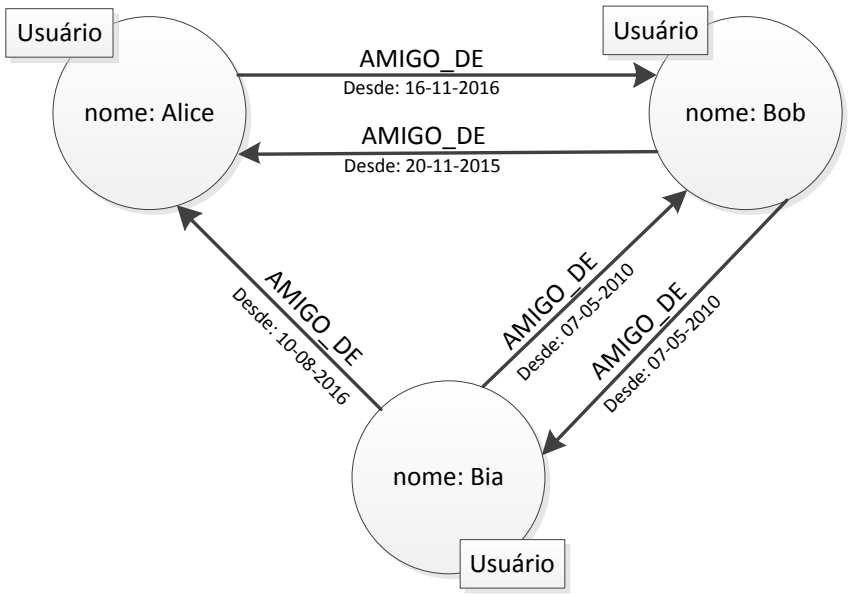

Figura 3 – Um pequeno grafo social

Fonte: Robinson e Eifrem (2015), adaptado pelo autor

Muitas pessoas acham o modelo de grafo proprietário intuitivo e fácil de entender. Embora simples, ele pode ser usado para descrever a grande maioria dos problemas, criando modelos sofisticados e ricos semanticamente (ROBINSON; EIFREM, 2015).

2.2.2 Banco de Dados Grafos

Um *sistema gerenciador de banco de dados grafos* (comumente chamados, *banco de dados grafos*-BDG) é um sistema gerenciador de banco de dados NOSQL que fornece os métodos *Create*, *Read*, *Update*, *Delete* para serem utilizados em um modelo de dados grafos, além de uma estrutura de armazenamendo de grafos especial. BDG são geralmente construídos para uso com sistema trasacionais (OLTP) e foram projetados pensando na integridade transacional e disponibilidade operacional (KALIYAR, 2015).

No BDG, diferentemente dos convencionais, os dados não são mantidos em tabelas. A estrutura de dados utilizada é o grafo, que possui todos os vértices e arestas diretamente conectadas a outro vértice, o que dispensa operações de junções. Os grafos armazenam os dados em nós os quais possuem alguns relacionamentos. Estes bancos de dados seguem o modelo de dados grafo proprietário.

BDG utiliza *index free adjacency* que é importante para alcançar alta performance ao percorrer o grafo. No banco de dados todo nó mantém referência direta para

os nós adjacentes. Ele é referenciado como um micro index para outros nós e é mais barato que usar indexes globais. Isso significa que o tempo de consulta é independente do tamanho total do grafo e simplesmente diretamente proporcional ao tamanho do subgrafo pesquisado. Banco de dados grafos produzem resultados muito rápido em termos de tempo de consulta e também podem armazenar grandes quantidades de dados (KALIYAR, 2015).

Relacionamentos são cidadãos de primeira classe no modelo de dados grafos. Este não é o caso em outros SGBD's, onde precisa-se inferir conexões entre as entidades utilizando artfícios, tais como, chave estrangeira ou processamento fora de banda como o *map-reduce*. BDG permite construir arbitrariamente modelos sofisticados mais próximos ao domínio do problema, apenas pela simples abstração de nós e relacionamentos dentro de estruturas conectadas. Os modelos resultantes são mais simples e ao mesmo tempo mais expressivos que aqueles produzidos usando banco de dados relacionais tradicionais e outros NOSQL (Not Only SQL) (ROBINSON; EIFREM, 2015).

Em teoria BDG são mais rápidos que os bancos de dados relacionais ao lidarem com relacionamentos e domínios altamente conectados. Por exemplo encontrar em um rede social todos os amigos dos amigos de um determinado usuário. Afim de demostrar isso na prática (PARTNER; WATT, 2014) realiza um experimento interessante. Eles construíram essa consulta em um banco de dados relacional e em um BDG, mas eles foram além do 2º nível de amizade (amigos dos amigos), chegando até o 5º nível. Eles utilizaram o SGBD relacional MySQL e o SGBD grafo Neo4j, com uma base de dados contendo 1,000,000 de usuários. Os resultados do experimento são apresentados na Tabela 1, o tempo de execução é dado em segundos.

Tabela 1 – Comparativo entre os tempos de execução do MySQL e Neo4j

Nível	**Tempo de execução em seg.- MySQL**	**Tempo de execução em seg. - Neo4j**
2	0.016	0.010
3	30.267	0.168
4	1,543.505	1.359
5	sem conclusão em 1 Hora	2.132

Fonte: Partner e Watt (2014)

No nível 2 os SGBD's realizam a consulta de forma boa o suficiente para serem utilizados em sistemas *online*. Embora o Neo4j seja 60% mais rápido que o MySQL, um usuário final não notaria essa diferença de milissegundos entre eles. Já no 3º nível (amigos dos amigos dos amigos), a diferença se torna bem maior e o Neo4j é 180 vezes mais rápido que o MySQL que possui um tempo de 30 segundos, inviável para um sistema *online*. No 4º nível a diferença aumenta e o Neo4j é 1,135 vezes mais

rápido. Por fim no 5º nível o MySQL nem consegue finalizar a consulta em uma hora de execução.

Baseado no que foi apresentado até aqui acredita-se que a modelagem de dados em grafo casa perfeitamente com aplicações de sistema de recomendação, pois nesses sistemas os dados são altamente conectados e os relacionamentos entre eles são fundamentais para predições. Os SGBD's Grafos abraçam relacionamentos, conseguindo processa-los de forma rápida e eficiente.

2.2.2.1 Neo4j

O Neo4j, foi o banco de dados grafo escolhido para ser utilizado neste trabalho, por ser bastante popular e possuir licença gratuita. O Neo4j permite que dados sejam percorridos da mesma forma que os grafos mantendo algumas características que se tornaram bastante comuns em banco de dados relacionais, como controle de transações e suporte completo as propriedades ACID por ter na memória logs de transações e gerenciador de bloqueio (ALMEIDA, 2011). Além de suporte a cluster, tornando-o adequado para usar dados grafos em produção.

Neo4j é um banco de dados grafo NoSql criado pela empresa Neo Technology, com o desenvolvimento iniciado em 2003 e publicado em 2007. O código do projeto pode ser visualizado no Github com suporte no StackOverflow e Google Groups, utilizado por centenas de milhares de companias e organizações em quase todos os mercados. Casos de uso incluem gerenciamento de rede, softwares analiticos, pesquisa cientifica, roteamento, gestão organizacional e de projeto, recomendações, redes sociais, entre outros. (EDUARDO, 2015).

Neo4j gerencia todas as operações que modificam dados em uma transação. Nele tanto nós quanto relacionamentos podem conter propriedades. Neo4j é um banco de dados otimizado para estrutura de grafos e segue o modelo de grafo proprietário. Há diversas maneiras de realizar consultas no Neo4j, por exemplo a API traverser nativa ou a linguagem de consulta Cypher (ROBINSON; EIFREM, 2015).

Segundo (EDUARDO, 2015), algumas características particulares fazem o Neo4j ser muito popular entre os usuários, desenvolvedores e adminstradores de banco de dados, sendo elas:

- Todos os relacionamentos em Neo4j são igualmente importantes e rápido , fazendo o possível para materializar e usar novas relações , mais tarde, para "atalho" e acelerar os dados de domínio quando surgem novas necessidades.
- Armazenamento compacto e cache de memória para grafos , resultando em aumento de escala e eficiente bilhões de nós em um banco de dados em hardware

moderado.

- Escrito em Java, rodando em JVM(Java Virtual Machine)

2.2.2.2 Linguagem de consulta Cypher

Optou-se pela utilização da linguagem de consulta a BDG Cypher por sua estreita afinidade com o hábito humano de representar grafos em forma de diagramas, tornando-o ideal para descrição dos mesmos. Além disso Cypher é uma liguagem de consulta de fácil aprendizado que serve como base para o apredizado de grafos. Depois de entender Cypher, torna-se muito mais fácil aprender outras linguagens de consultas.

Pela proximidade com a forma de representação intuitiva que descreve-se grafos, o Cypher é facilmente lido e compreendido por desenvolvedores, profissionais *database* e *stakeholders*. Um exemplo dessa proximidade de representação é apresentado por (ROBINSON; EIFREM, 2015), onde é apresentado a representação de um modelo simples de grafo em Cypher, a partir da representação de um modelo intuitivo. A Figura 4 apresenta o modelo elaborado para o exemplo.

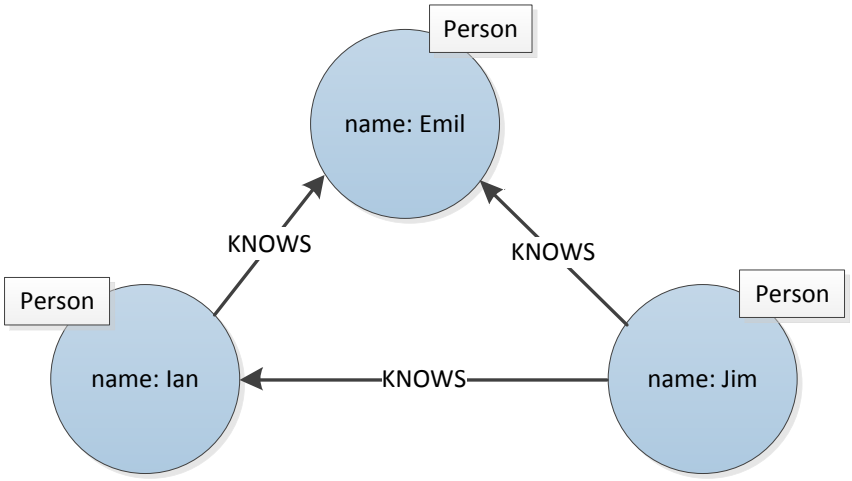

Figura 4 – Modelo de grafo simples, expressado na forma de diagrama

Fonte: Robinson e Eifrem (2015)

O Cypher representa os grafos em forma de desenhos utilizando ASCII, o modelo apresentado na figura é representado na linguagem da seguinte forma:

Listagem 1 – Exemplo de contrução de grafo na linguagem Cypher.

```
1 (emil)<-[:KNOWS]-(jim)-[:KNOWS]->(ian)-[:KNOWS]->(emil)
```

Fonte: Robinson e Eifrem (2015)

O padrão descreve caminhos que conectam jim a emil e ian assim como ian a emil. A sintaxe do Cypher segue muito naturalmente a forma como descreve-se grafos em lousas ou nas folhas de papel.

Para demonstrar como os banco de dados grafos abraçam os relacionamentos, ao contrário dos relacionais, (ROBINSON; EIFREM, 2015) apresentam o resultado de uma consulta a uma base de dados relacional, novamente citando o exemplo de amigos dos amigos. A modelagem do banco de dados é apresentada na Figura 5, extraída do livro.

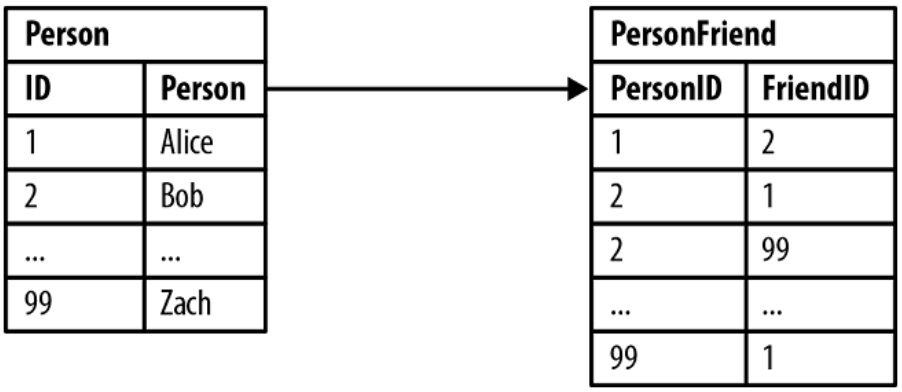

Figura 5 – Modelo de banco de dados relacional para armazenamento de amizades

Fonte: Robinson e Eifrem (2015)

A consulta a essa base de dados para a pergunta “quem são os amigos dos amigos de Alice” ficaria da seguinte forma:

Listagem 2 – Consulta em SQL para amigos dos amigos de Alice.

```
1 SELECT p1.Person AS PERSON, p2.Person AS FRIEND_OF_FRIEND
2 FROM PersonFriend pf1 JOIN Person p1
3 ON pf1.PersonID = p1.ID
4 JOIN PersonFriend pf2
5 ON pf2.PersonID = pf1.FriendID
6 JOIN Person p2
7 ON pf2.FriendID = p2.ID
8 WHERE p1.Person = "Alice" AND pf2.FriendID <> p1.ID
```

Fonte: Robinson e Eifrem (2015)

Essa é uma consulta complexa computacionalmente e de difícil compreensão até mesmo para desenvolvedores, a coisa piora ao aprofundar os níveis de amizades.

A mesma consulta em banco de dados grafos, utilizando a linguagem Cypher ficaria da seguinte forma:

Listagem 3 – Consulta em Cypher para amigos dos amigos de Alice.

```
1 MATCH(a:Person)-[:KNOWS]->(b)-[:KNOWS]->(c)
2 WHERE a.Person = "Alice"
3 RETURN b, c
```

Fonte: Robinson e Eifrem (2015)

Com o Cypher a consulta ficou muito mais compreensível e até mesmo intuitiva, dessa forma tornando mais simples até mesmo a implementação de sistemas mais complexos. A atividade de manutenção dos sistemas também é beneficiada, pois ficará mais simples compreender o papel de cada consulta e como ela funciona.

3 Técnicas Propostas

Neste capítulo são apresentadas duas propostas de técnica de recomendação utilizando sistemas de banco de dados grafos. A filtragem de recomendação abordada será a colaborativa, por ser a mais utilizada atualmente. As técnicas apresentadas são baseadas nas estratégias de recomendação demonstradas em (KERNIX, 2016), que apresenta alguns passos para a elaboração de um sistema de recomendação.

3.1 Definição dos Passos das Técnicas

Neste estudo são propostas duas técnicas para recomendação por filtragem colaborativa, desta forma, foca-se no passado do usuário, como o que ele viu e como avaliou, para encontrar subconjuntos de usuários com gostos similares ao usuário alvo. Para realizar as recomendações, as características de todos os usuários similares devem ser estudadas(KERNIX, 2016). utiliza-se os seguintes passos:

- Definição de cálculo de similaridade entre dois usuários
- Retenção dos usuários que são mais similares ao usuário alvo das recomendações
- Definição de cálculo para ranqueamento dos itens em comum entre os usuários similares
- Recomendação dos itens melhores ranqueados que ainda não foram vistos pelo usuário alvo.

3.1.1 Técnica 1: Os itens mais vistos por usuários similares

Esta técnica consiste em considerar a similaridade entre dois usuários u_1 e u_2 utilizando o índice de similaridade de Sorensen-Dice, que é um coeficiente que pode ser utilizado para comparação entre simples conjuntos de dados e é expresso pela razão entre o dobro do número de elementos em comum entre dois conjuntos pela soma da quantidade de elementos em cada conjunto(ZHAO; CHEN, 2013).

Em relação aos itens avaliados pelos usuários, eles serão classificados a partir de sua popularidade do mais popular para o menos popular. Este método pode ser aplicado mesmo sem ter o conhecimento das notas dadas pelos usuários aos itens.

Dado R_{ui}, o conjunto de itens avaliados por u_i, a similaridade entre dois usuários u_1 e u_i pode ser expressa por:

$$sim(u_1, u_i) = \frac{2 * R_{u1} \bigcap R_{ui}}{R_{u1} \bigcup R_{u_i}} \tag{3.1}$$

O resultado da função de similaridade é o grau de semelhança entre os dois usuários. A partir do grau de semelhança é possível realizar a junção por similaridade, que é a tarefa de utilizar funções de similaridade com o objetivo de encontrar pares de objetos que tenham um grau de semelhança acima de um limite estipulado (SIDNEY, 2014).

Realizando a junção por similaridade iremos encontrar o conjunto U_{sim} formados pelos usuários com grau de semelhança acima do limiar t. Encontramos U_{sim} da seguinte forma:

$$U_{sim} = \bigcup_{sim(u_1,u_i)>t} u_i \tag{3.2}$$

O valor do *threshold* t pode ser alterado para otimizar a seletividade do subconjunto de usuários similares.

O conjunto de itens avaliados pelos usuários similares mas não por u_1, pode ser expresso como:

$$M_{sim} = [\bigcup_{u_i \in U_{sim}} R_{ui}] \bigcap \overline{R_{u1}} \tag{3.3}$$

O ranqueamento de cada item m_j de M_{sim} é então computado como a proporção de usuários similares que também avaliaram/visitaram m_j. Neste caso, os itens serão ranqueados do mais visitado para o menos visitado entre os usuários similares.

Sendo Q_{mj} o número de usuários que visitaram um determinado item m_j, temos:

$$Q_{mj} = \bigcup_{\substack{u_i \in U_{sim} \\ m_j \in R(u_i)}} u_i \tag{3.4}$$

Logo a nota para cada item m_j é dada por:

$$s(m_j) = \frac{Q_{m_j}}{U_{sim}} \tag{3.5}$$

3.1.2 Técnica 2: Os itens que foram visitados e bem raqueados por usuários similares

Nesta técnica, ocorre um refinamento na maneira como os itens são classificados, explorando as avaliações dadas pelo subconjunto de usuários semelhantes.

Os conjuntos U_sim e M_sim são encontrados da mesma forma como foram encontrados na técnica anterior, utilizando as equações 3.1, 3.2 e 3.3.

A pontuação dada a um determinado item nesta técnica é definida como a soma das avaliações atribuídas a m_j pelos usuários semelhantes. Seja $r(m_j, u_i)$ a classificação dada pelo usuário u_i ao item m_j. O ranking do item m_j é determinado como:

$$s(m_j) = \sum_{\substack{u_i \in U_{sim} \\ m_j \in R(u_i)}} r(m_j, u_i) \tag{3.6}$$

Desta forma, espera-se uma melhor recomendação, já que os itens a serem indicados serão aqueles melhores ranqueados pelos usuários similares ao alvo.

4 Estudo de Caso

Neste capítulo é realizado um estudo de caso sobre um sistema de recomendação de filmes utilizando as técnicas propostas no Capítulo 3.

4.1 Conjunto de Dados

Neste trabalho buscou-se utilizar um conjunto de dados clássico na literatura. O conjunto de dados escolhido foi o MovieLens, disponibilizado pelo projeto de pesquisa GroupLens Research Project da University of Minnesota, que desenvove ferramentas experimentais e interfaces para exploração de dados e recomendação (RESEARCH, 2016).

Várias versões do conjunto de dados são disponibilizadas no site da MovieLens, a versão escolhida foi a ML-100K que possui 100 mil avaliações para 1.682 filmes por 943 usuários. Os dados deste conjunto foram coletados a partir do site da MovieLens durante o período de 19 de Setembro de 1997 a 22 de abril de 1998. Estes dados foram tratados e usuários com menos de 20 avaliações foram removidos do conjunto (HARPER; KONSTAN, 2015). Os dados disponíveis neste conjunto são listados a seguir:

- 100 mil avaliações, variando de 1 a 5, a 1.682 filmes por 943 usuários
- Cada usuário avaliou pelo menos 20 filmes
- Informações demográficas simplificadas sobre os usuários (idade, sexo, ocupação, CEP e um id)
- Informações sobre o filme, como título, data de lançamento, IMDb URL, gênero (pode ser vários ao mesmo tempo), além de um identificador único

Neste trabalho foram utilizadas apenas as informações referentes as avaliações dos usuários, título dos filmes com seu respectivo ID e ID do usuário. Desta forma o conjunto de dados pode ser representado na forma de um simples grafo com:

- Dois tipos de nós:
  - User
  - Movie

- Um tipo de relacionamento:

  Has_rated: se o usuário u_i avaliou o filme m_j, então u_i está explicitamente conectado a m_j, com uma relação do tipo Has_rated. A avaliação $r(m_j, u_i)$ é armazenada como uma propriedade do relacionamento e seu valor varia entre 1 e 5.

Um modelo do grafo extraído do conjunto de dados é apresentado na Figura 6, que demonstra parte do grafo, selecionando apenas o usuário 100 e os respectivos filmes avaliados por ele.

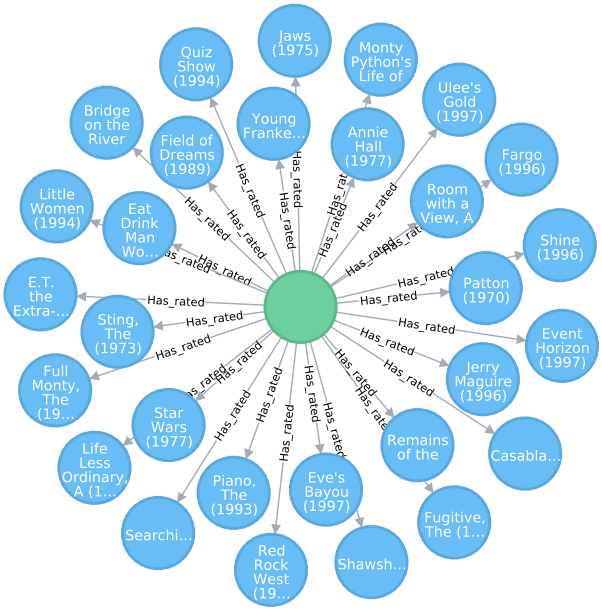

Figura 6 – Usuário 100 e todos os filmes que ele avaliou

Fonte: *Print Screen* do Neo4j

4.2 Algoritmo

Nesta sessão é demonstrado como converter as equações e passos apresentados no Capítulo 3 em linhas de código para a linguagem de consulta Cypher. Neste estudo escolheu-se aleatoriamente o usuário com id 60 para ser o usuário alvo(u_1) das recomendações, em um caso prático seria como se o usuário 60 estivesse logado no Netflix e o sistema tentará encontrar os melhores filmes para ele.

4.2.1 Algoritmo para Técnica 1

Nesta subseção é apresentado o algoritmo que traduz os passos apresentados na Subseção 3.1.1. A princípio conta-se a quantidade de filmes assistidos pelo usuário alvo(u_1) e o resultado é armazenado em um variável para posterior utilização.

Listagem 4 – Contando a quantidade de filmes ranqueados por u_i.

```
1 MATCH (u1:User{user_id: 60})-[:Has_rated]->(m1:Movie)
2 WITH count(m1) as countm
```

Fonte: Elaborado pelo autor

Em seguida encontra-se os usuários que tenham ao menos um filme em comum com o usuário o alvo, linhas 3, 4 e 5. Depois, nas linhas 9 e 10, esses usuários são submetidos a função de similaridade que dá um peso para cada usuário a partir deste peso, na linha 13 são filtrados apenas aqueles usuários que possuem um peso superior ao limiar. Assim encontra-se dentre os usuários que possuem filmes em comum com o alvo aqueles que são mais similares a ele.

Após encontrar os usuários similiares é realizada na linha 16 da Listagem 5 a contagem deles e o valor armazenado na variável countu, esse valor será utilizado para o cálculo de ranqueamento dos filmes que serão recomendados.

Listagem 5 – Contando a quantidade de usuários similares a u_i.

```
1 /* Encontra usuários que tenham assitido filmes em  */
2 /* comum com o alvo e conta a quantidade desses filmes */
3 MATCH (u1:User{user_id:60})-[:Has_rated]->(m1:Movie)
4 MATCH (m1)<-[r:Has_rated]-(u2:User)
5 WITH u2, countm, count(r) as countr
6
7 /* Econtra quantidade de filmes assistidos pelos usuários */
8 /* que tem filmes em comum com o alvo e computa similaridade*/
9 MATCH (u2)-[r2:Has_rated]->(m:Movie)
10 WITH u2, countm, 2*tofloat(countr)/(count(r2)+countm) as sim
11
12 /* Mantém os usuários u2 que possuem similaridade superior ao limiar */
13 WHERE sim>0.5
14
15 /* Conta o número de usuários similares e armazena em countu */
16 WITH count(u2) as countu, countm
```

Fonte: Elaborado pelo autor

Este passo é quase uma repetição do código apresentado na Listagem 5, tendo que ser realizado para que se econtre o conjunto de usuários similares ao alvo, pois o Cypher não possibilita contar a quantidade de um conjunto e armazena-lo ao mesmo

tempo. Na listagem 5 apenas contou-se a quantidade de usuários similares ao alvo, na listagem 6 armazena-se um vetor com cada um desses usuários.

Listagem 6 – Armazenamento do subconjunto de usuários similares a u_1

```
1 /* Encontra usuários que tenham assitido filmes em  */
2 /* comum com o alvo e conta a quantidade desses filmes */
3 MATCH (u1:User{user_id:60})-[:Has_rated]->(m1:Movie)<-[r:Has_rated]-(u2:
    User)
4 WHERE NOT u2=u1
5 WITH u2, countu, countm, count(r) as countr
6
7 /* Econtra quantidade de filmes assistidos pelos usuários */
8 /* que tem filmes em comum com o alvo e computa similaridade*/
9 MATCH (u2)-[r2:Has_rated]->(m:Movie)
10 WITH u2, countm, countu, 2*tofloat(countr)/(count(r2)+countm) as sim
11
12 /* Mantém os usuários u2 que possuem similaridade superior ao limiar */
13 WHERE sim>0.5
```

Fonte: Elaborado pelo autor

Agora inicia-se o processo de recomendação em si, após encontrar-se o subconjunto de usuários que são mais similares ao alvo(u_1) serão encontrados os filmes que esses usuário assistiram mas que o usuário alvo não assistiu, linhas 3 e 4 da Listagem 7. Por fim é computada a nota para cada filme e apresentada uma lista de sugestão de filmes ordenados pela nota de cada um, linhas 7,8 e 9.

Listagem 7 – Retorno dos 20 filmes mais recomendados para o usuário u_1

```
1 /* Recupera filmes m que foram assistidos pelos */
2 /* similares do usuário, mas não pelo usuário */
3 MATCH (m:Movie)<-[r:Has_rated]-(u2)
4 WHERE NOT (m)<-[:Has_rated]-(:User{user_id:60})
5
6 /* Computa nota/score e retorna a lista de sugestão ordenado de forma
    decrescente por nota/score  */
7 RETURN DISTINCT m.movie_id, m.title, tofloat(count(r))/countu as score
    ORDER BY score DESC LIMIT 10
8 tofloat()/countm as sim
```

Fonte: Elaborado pelo autor

4.2.2 Algoritmo para Técnica 2

Para este algoritmo ocorreram poucas mudanças em relação ao apresentado na Subseção 4.2.1, a diferença entre os dois está no fato deste agoritmo levar em consideração as notas dadas pelos usuários similares.

Uma sutíl diferença é apresentada na filtragem dos usuários similares, que neste algoritmo inclui a condição de que a diferença entre as notas dadas pelos usuários similares e o alvo para um mesmo filme não podem divergir em mais de um ponto, que pode ser visto na linha 15 da Listagem 8. Desta forma qualificando um pouco mais a seleção dos usuários afim ao alvo.

A outra diferença está na forma de calcular as notas para os filmes recomendados, agora sendo levando em conta as notas dadas por todos os usuários similares, esse cálculo pode ser observado na linha 28 da Listagem 8. Com isso houve também redução no número de linhas de código, pois não há necessidade de calcular a quantidade de usuários similares.

Abaixo é apresentado o algoritmo para a metodologia apresentada na Subseção 3.1.2.

Listagem 8 – Algoritmo para Metodologia 2

```
1  /* Encontra a quantidade de filmes assitidos  */
2  /* pelo usuário alvo */
3  MATCH (u1:User {user_id:60})-[:Has_rated]->(m1:Movie)
4  WITH count(m1) as countm
5  /* Encontra usuários que tenham assitido filmes em comum */
6  /* com o alvo e conta a quantidade desses filmes */
7  MATCH (u1:User{user_id:60})-[:Has_rated]->(m1:Movie)<-[r:Has_rated]-(u2:
       User)
8  WITH u2, countm, count(r) as countr
9  /*Econtra quantidade de filmes em comum assistidos por cada usuários*/
10 /* e computa similaridade */
11 MATCH (u2)-[r2:Has_rated]->(m:Movie)<-[r1:Has_rated]-(u1:User{user_id
       :60})
12 WHERE (NOT u2=u1) AND (abs(r2.rating - r1.rating) <= 1)
13 WITH u1, u2, countm, 2*tofloat(countr)/(count(r2)+countm) as sim
14 /* Mantém os usuários u2 que possuem similaridade superior ao limiar */
15 WHERE sim>0.5
16 /* Recupera filmes m que foram assistidos pelos similares do usuário, */
17 /* mas não pelo usuário */
18 MATCH (m:Movie)<-[r:Has_rated]-(u2)
19 WHERE NOT (m)<-[:Has_rated]-(u1)
20 /* Computa nota/score e retorna a lista de sugestão ordenado de forma */
21 /* decrescente por nota/score  */
22 RETURN DISTINCT m,tofloat(sum(r.rating)) as score ORDER BY score DESC
```

Fonte: Elaborado pelo autor

4.3 Resultados

Para verificar a qualidade dos resultados obtidos com cada técnica proposta, foi utilizado 40 % da base para realização das predições e o restante para o teste. Dentre os 943 usuários do banco foram escolhidos 20 aleatoriamente para servirem de amostragem para os testes. Estes 20 usuários foram submetidos as predições da técnica 1 e 2, após isso os resultados das predições de cada técnica foram comparados com os resultados obtidos fazendo a consulta nos 60 % restante da base.

Com os resultados obtidos de cada técnica foi elaborada uma tabela comparativa para avaliar as mesmas. Para a elaboração dessa tabela levou-se em consideração o percentual das notas dadas pelos usuários aos filmes após a predição. A tabela comparativa é apresentada abaixo.

Tabela 2 – Comparativo entre a técnica 1 e 2, considerando os filmes preditos e as notas dadas pelos usuários

	Estrelas dadas aos filmes preditos			
	5	**4**	**3**	**Não vistos**
Técnica 1	45%	30%	18%	7%
Técnica 2	52%	35%	10%	3%

Fonte: Elaborado pelo autor

Analisando a Tabela 2, pode-se observar que houve uma pequena melhora do resultados obtidos pela técnica 2 em relação a 1, possivelmente impulsionados pelo cálculo de ranqueamento que utiliza as notas dadas pelos usuários similares.

O número de filmes preditos pela técnica 2 que foram avaliados com nota 5 pelos usuários foi 52% sendo uma melhoria de 7% em relação a técnica 1. Para os filmes avaliados com nota 4 a técnica 2 apresentou uma melhoria de 5% em relação a 1, nos filmes avaliados com nota 3 a superioridade da técnica 2 foi de 8%. Os percetuais de filmes preditos mas que não foram visualizados pelos usuários foram de 7% na técnica 1 e de 3% na técnica 2.

Mesmo utilizando algoritmos simples foi possível realizar recomendações de filmes satisfatórias nas duas técnicas propostas, além disso foi possível notar que a estrutura de grafo permite recuperar o conjunto de usuários similares ao alvo, sem navegar por todo o conjunto de usuário, e que o conjunto de filmes assistidos pelos usuários similares mas que não foram vistos pelo alvo pôde ser recuperado de forma muito simples e direta, comprovando que a utilização de base dados grafos é uma boa abordagem para aplicação em sistemas de recomendação.

5 Conclusão

A sobrecarga de informação gerada atualmente, grande parte impulsionada pelo desenvolvimento das tecnologias, tem gerado problemas para os usuários encontrarem aquilo que realmente buscam. Por isso foram realizadas diversas pesquisas a fim de auxiliar os usuários a encontrarem as informações que realmente desejam em um tempo hábil. Assim surgiram os sistemas de recomendação, que tem como objetivo sugerir conteúdos baseando-se no perfil do usuário.

Para a implementação de um sistema de recomendação foi sugerido a utilização de um banco de dados orientado a grafos, pois este apresenta características que se mostram promissoras para a construção desse tipo de sistemas, principalmente o fato de trabalharem bem com relacionamentos.

O maior desafio encontrado para atingir os objetivos deste trabalho está relacionado ao fato de banco de dados grafos ser uma tecnologia que começou a ser melhor explorada recentemente, o que torna difícil encontrar bons materiais de estudo, principalmente em português.

O objetivo geral deste trabalho foi atingido, uma vez que são propostas, no Capítulo 3 duas técnicas para recomendação que foram aplicadas em banco de dados orientados a grafos no Capítulo 4. As técnicas demonstradas apresentaram resultados satisfatórios, já que foram desenvolvidas de forma simples e ainda assim indicaram percentual expressivo de aceitação dos usuários aos itens recomendados.

Os objetivos específicos deste trabalho foram atingidos, uma vez que, são apresentados os principais conceitos sobre sistemas de recomendação e modelagem de dados grafos no Capítulo 2 e é realizado um estudo de caso de sistema de recomendação de filmes aplicando as técnicas propostas neste trabalho no Capítulo 4.

5.1 Trabalhos Futuros

Os resultados obtidos com as duas técnicas propostas foram satisfatórios para este trabalho, mas podem ser melhorados. Desta forma, como trabalhos futuros propõe-se a melhoraria dos resultados obtidos quer seja pela forma de recuperar o conjunto de usuários similares ou pelo cálculo de ranqueamento dos itens recomendados.

Também é sugerida a expansão das técnicas propostas para a utilização da filtragem híbrida, desta forma, utilizando tanta a filtragem colaborativa quanto a baseada em conteúdo para realizar as recomendações.

Referências

ADOMAVICIUS, G.; TUZHILIN, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. In: *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*. [S.l.]: IEEE, 2005. v. 17, n. 63. Citado 3 vezes nas páginas 16, 18 e 19.

ALMEIDA, A. *Trabalhando com Relacionamentos: bancos de dados baseados em grafos e o Neo4j*. [S.l.]: Caelum, 2011. <http://blog.caelum.com.br/trabalhando-com-relacionamentos-bancos-de-dados-baseados-em-grafos-e-o-neo4j/>. Citado na página 25.

ANGLES, R.; GUTIERREZ, C. Survey of graph database models. In: *ACM Computing Surveys*. [S.l.]: ACM, 2008. v. 40, n. 1. Citado na página 20.

CODD, E. F. A relational model of data for large shared data banks. In: *Communications of the ACM*. [S.l.]: ACM, 1970. v. 13, n. 6, p. 377–387. Citado na página 20.

CODD, E. F. A relational model of data for large shared data banks. In: *Communications of the ACM*. [S.l.]: ACM, 1983. v. 26, n. 1, p. 64–69. Citado na página 20.

COSTA, R. M. de R. *From Tapestry to SVD: A Survey of the Algorithms That Power Recommender Systems*. Tese (Doutorado) — Haverford College Department of Computer Science, 2009. Citado na página 15.

CUNG, H.-Q.; JEDIDI, M. Implementing a recommender system with graph database. *Universite de Fribourg*, Universite de Fribourg, p. 3–24, 2014. Citado na página 13.

EDUARDO, N. *Bem vindo ao Neo4j*. [S.l.]: Nicholasess, 2015. <http://nicholasess.com.br/neo4j-2/bem-vindo-ao-neo4j/>. Citado na página 25.

EKSTRAND, J. T. R. M. D.; KONSTAN, J. A. Collaborative filtering recommender systems. In: *Foundations and Trends in Human–Computer Interaction*. [S.l.]: now publishers, Inc., 2010. v. 4, n. 2, p. 81–173. Citado 3 vezes nas páginas 13, 16 e 18.

ELMASRI, R.; NAVATHE, S. B. *Sistemas de banco de dados*. Sexta. [S.l.]: Pearson Education, 2011. Citado na página 20.

GOLDBERG DAVID NICHOLS, B. M. O. D.; TERRY, D. Using collaborative filtering to weave an information tapestry. In: *Communications of the ACM - Special issue on information filtering*. [S.l.]: ACM, 1992. v. 35, n. 12, p. 61–70. Citado na página 15.

GOLDSCHMIDT, R.; BEZERRA, E. *Data mining : conceitos, técnicas, algoritmos, orientações e aplicações*. [S.l.]: Elsevier Brasil, 2015. Citado na página 15.

HARPER, F. M.; KONSTAN, J. A. The movielens datasets: History and context. In: *ACM Transactions on Interactive Intelligent Systems (TiiS)*. [S.l.]: ACM, 2015. v. 4, n. 19. Citado na página 32.

HU, H. An initial study of recommender systems. Linkoping University, 2012. Citado na página 19.

KALIYAR, R. kumar. Graph databases: A survey. *International Conference on Computing, Communication and Automation*, ACM, 2015. Citado 3 vezes nas páginas 20, 23 e 24.

KERNIX. *An efficient recommender system based on graph database*. [S.l.]: kernix.com, 2016. <http://www.kernix.com/blog/an-efficient-recommender-system-based-on-graph-database_p9>. Citado na página 29.

KIM, W. Object-oriented databases: Definition and research directions. *IEEE Transactions on knowledge and Data Engineering 2.3*, IEEE, 1990. Citado na página 20.

LESKOVEC, A. R. J.; ULLMAN, J. D. *Mining of Massive Datasets*. [S.l.]: Stanford, 2010. Citado 2 vezes nas páginas 17 e 18.

MEDEIROS, I. R. G. Estudo sobre sistemas de recomendação colaborativos. UFPE, 2013. Citado na página 16.

MOONEY, R. J.; ROY, L. Content-based book recommending using learning for text categorizatio. In: *Proceedings of the fifth ACM conference on Digital libraries*. [S.l.]: ACM, 2000. p. 195–204. Citado na página 19.

PARTNER, A. V. J.; WATT, N. *Neo4j in Action*. [S.l.]: Manning Publications, 2014. Citado na página 24.

RESEARCH, G. *Movielens*. 2016. Acessado em: 2016-10-06. Disponível em: <https://movielens.org/>. Citado na página 32.

RESNICK, P. Recommender systems. In: *Communications of the ACM*. [S.l.]: ACM, 1997. v. 40, n. 3, p. 56–58. Citado na página 15.

RICCI LIOR ROKACH, B. S. F.; KANTOR, P. *Recommender Systems Handbook*. [S.l.]: Springer, 2011. Citado na página 16.

ROBINSON, J. W. I.; EIFREM, E. *Graph Databases*. Second. [S.l.]: O'Reilly Media, 2015. Citado 8 vezes nas páginas 20, 21, 23, 24, 25, 26, 27 e 28.

SARWAR GEORGE KARYPIS, J. K. B.; RIEDL, J. Item-based collaborative filtering recommendation algorithms. *Proceedings of the 10th international conference on World Wide Web*, ACM, p. 285–295, 2001. Citado na página 16.

SHETH, B.; MAES, P. Evolving agents for personalized information filterings. *Proc. Ninth IEEE Conf. Artificial Intelligence for Applications*, IEEE, 1993. Citado na página 18.

SIDNEY, C. F. *Predição de desempenho para junções por similaridade baseadas em conjuntos*. [S.l.]: UFLA, 2014. Citado na página 30.

SKRASEK, J. *Social Netword Recommendation using Graph Databases*. [S.l.]: Masaryk University, 2015. Citado na página 16.

ZADEH, R. *Basic Definitions and Concepts in Graph Theory*. 2016. Acessado em: 2016-10-20. Disponível em: <https://stanford.edu/~rezab/discrete/Notes/2.pdf>. Citado na página 21.

ZHANG, J. C. Y.; MINKA, T. Novelty and redundancy detection in adaptive filtering. *Proc 25th Ann Int'l ACM SIGIR Conf*, ACM, 2002. Citado na página 18.

ZHAO, R.; CHEN, L. String similarity metrics comparison for name-matching task. Carnegie Mellon University, 2013. Citado na página 29.