Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Análise de Projetos de Banco de Dados: Modelo Relacional vs. Modelo em Grafos

Thales Paim Fachinelli

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Bruno Rabello Monteiro

COORIENTAÇÃO:
Alexandre Magno de Sousa

Julho, 2019
João Monlevade–MG

Thales Paim Fachinelli

Análise de Projetos de Banco de Dados: Modelo Relacional vs. Modelo em Grafos

Orientador: Bruno Rabello Monteiro

Coorientador: Alexandre Magno de Sousa

Monografia apresentada ao curso de Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Julho de 2019

F139a Fachinelli, Thales Paim.
Análise de projetos de banco de dados [manuscrito]: modelo relacional vs. modelo em grafos / Thales Paim Fachinelli. - 2019.

63f.:

Orientador: Prof. MSc. Bruno Rabello Monteiro.
Coorientador: Prof. MSc. Alexandre Magno de Souza.

Monografia (Graduação). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Departamento de Computação e Sistemas de Informação.

2. Grafos de ligação . 3. Modelagem. 4. Banco de dados relacionais. 5. Projeto de banco de dados. I. Monteiro, Bruno Rabello. II. Souza, Alexandre Magno de. III. Universidade Federal de Ouro Preto. IV. Titulo.

CDU: 004.652

Catalogação: ficha.sisbin@ufop.edu.br

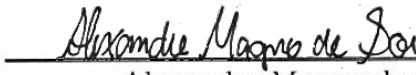
FOLHA DE APROVAÇÃO DA BANCA EXAMINADORA

Análise de Projetos de Banco de Dados: Modelo Relacional vs. Modelo em Grafos

Thales Paim Fachinelli

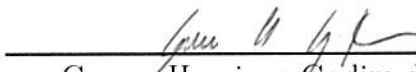
Monografia apresentada ao Instituto de Ciências Exatas e Aplicadas da Universidade Federal de Ouro Preto como requisito parcial da disciplina CSI499 – Trabalho de Conclusão de Curso II do curso de Bacharelado em Sistemas de Informação e aprovada pela Banca Examinadora abaixo assinada:

Bruno Rabello Monteiro
Me.
DECSI - UFOP



Alexandre Magno de Sousa
Me.
DECSI - UFOP

Rafael Frederico Alexandre
Dr.
Examinador
DECSI - UFOP



George Henrique Godim da Fonseca
Dr.
Examinador
DECSI - UFOP

João Monlevade, 10 de julho de 2019

Este trabalho é dedicado primeiramente a minha família. A meus pais Ademir e Elba e a minha irmã Vittória, vocês sempre foram e sempre serão meu maior tesouro. Dedico este trabalho também a minha namorada Amanda e sua família que sempre me deram apoio e me guardaram com muito zelo e carinho. Não menos importante dedico o trabalho aos meus orientadores Bruno e Alexandre Magno que sempre acreditaram no meu potencial e direcionaram da melhor forma possível.

Agradecimentos

Agradeço a Deus por todas a bençãos.

A minha família pelo apoio incondicional.

A minha namorada e sua família por todo carinho e zelo.

Aos meus amigos pelo companheirismo.

Aos meus orientadores pelo conhecimento compartilhado.

"Alea iacta est"

— Gaius Julius Caesar (January 10th, 49 B.C.),
as he led his army across the Rubicon River, Northern Italy.

Resumo

Este trabalho objetiva apresentar as diferenças no processo de modelagem de bancos de dados relacionais e de bancos de dados orientados a grafos (BDGs). Não obstante deste propósito, há também o desígnio de propor um processo de modelagem mais estruturado para os BDGs, bem como, identificar possíveis características apresentadas na etapa conceitual de modelagem de problemas, que poderiam direcionar a escolha do projetista com relação a qual dos dois modelos adotar para o seu desenvolvimento. A metodologia proposta para cumprir o objetivo foi uma extensa revisão bibliográfica, a fim de compreender os conceitos necessários, além da modelagem conceitual de três casos de uso de BDGs. Ademais, os conceitos e características de cada modelo são explorados com a finalidade de determinar as vantagens e desvantagens do modelo relacional e do modelo orientado a grafos.

Palavras-chaves: Banco de dados. Grafos. Modelagem. NoSQL.

Abstract

This paper aims to present the differences in the process of modeling relational databases and graph-oriented databases. Notwithstanding this purpose, there is also the purpose of proposing a more structured modeling process for these databases, as well, as identifying possible features presented in the conceptual problem modeling stage, which could direct the designer's choice as to which of the two models to adopt. The methodology proposed to accomplish the objective was an extensive literature review, to understand the necessary concepts, in addition to the conceptual modeling of three use cases. Moreover, the concepts and characteristics of each model are explored in order to determine the advantages and disadvantages of the relational model and the graph oriented model.

Key-words: NoSQL. Graphs. Databases. Modelling.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

BD Banco de Dados

BDs Bancos de Dados

BDD Banco de Dados Distribuído

BDDs Bancos de Dados Distribuídos

SGBD Sitema Gerenciador de Banco de Dados

SGBDR Sitema Gerenciador de Banco de Dados Relacional

SGBDG Sitema Gerenciador de Banco de Dados orientado a Grafos

SQL Structured Query Language

NoSQL Not SQL

NoAM NoSQL Abstract Model

Sumário

1 Introdução

Segundo Abadi, Madden e Hachem (2008), bancos de dados (BDs) são coleções de dados relacionados e estão presentes constantemente no dia a dia das pessoas. Podem ser vistos desde operações bancárias, passando pelas redes sociais, e até em jogos *online*. Tal definição é corroborada por outros autores como Silberschatz, Korth e Sudarshan (2012), que descrevem um BD como "uma coleção de dados inter-relacionados, representando informações sobre um domínio específico". O principal papel dos BDs é o armazenamento de dados de forma persistente e correlacionada. Bancos de dados são controlados por sistemas de gerenciamento de banco de dados (SGBD), que possuem recursos capazes de manipular e consultar informações, além de garantir restrições impostas ao BD.

Recentemente, o volume de dados gerados, e que precisam ser analisados, chegou a um tamanho crítico. Dados não estruturados surgem de dados estruturados devido a liberdade de produção de conteúdo fornecida pela Internet, o que desafia os métodos mais tradicionais de armazenamento de dados (VIRGILIO; MACCIONI; TORLONE, 2013). Esse cenário motivou os projetistas a buscarem novas soluções em bancos de dados e impulsionou a criação de modelos não relacionais, chamados de NoSQL (*Not Only* SQL), em outras palavras, modelos que não utilizam o modelo relacional e a linguagem SQL. Dentre estes modelos, encontra-se o modelo orientado a grafos (MoG), que almeja trazer as vantagens da utilização da teoria dos grafos na abordagem de problemas de armazenamento de dados.

O projeto de sistemas de bancos de dados relacionais está consolidado, tanto profissionalmente, quanto academicamente, sendo abordado amplamente por autores como Heuser (2008), Silberschatz, Korth e Sudarshan (2012), Date (2004) e Elmasri e Navathe (2010). Entretanto, a modelagem e execução de projetos de bancos de dados não-relacionais ainda carece de amadurecimento e padronização (JACOBS, 2009).

Nos livros textos e complementares das disciplinas que cobrem o conteúdo de banco de dados, o material sobre modelos não relacionais, mais especificamente, sobre o MoG e os banco de dados orientado a grafos/banco de dados em grafos (BDG), não contém uma análise aprofundada do assunto.

Na literatura é possível encontrar trabalhos que abordam os BDGs. Miller (2013) apresenta um estudo da viabilidade dos BDGs em relação ao modelo de dados relacional e como problemas tais quais as redes sociais ou áreas como a química, a biologia e a semântica *web*, são melhor modelados usando grafos. Já Lopes (2014) e Batra e Tyagi (2012) propõem análises voltadas às diferenças de desempenho entre sistemas que utilizam o modelo relacional e sistemas que usam o MoG. Ciglan, Averbuch e Ladialav (2012), por

sua vez, realizaram estudos de tempos gastos para percorrer bases de dados em grafos.

Jouili e Vansteenberghe (2013) fizeram comparações entre os SGBDs que utilizam o MoG. Foram comparados o desempenho dos principais SGBDs em grafos: Neo4j[1], Orient DB[2], Titan[3] and DEX[4].

Além disso, os banco de dados em grafos, são utilizados por grandes empresas como: Walmart[5], em seu sistema de recomendação de compras; Lockheed Martin Space[6], em suas pesquisas sobre o ciclo de vida de seus produtos; E-bay[7], nas sua aplicação de *e-commerce*; UBS[8], na administração de seu banco de dados de conhecimento; Monsanto[9], em pesquisas de bancos de dados genômicos; e a Catterpilar[10], no armazenamento e administração da documentação relacionada a seu maquinário.

A principal motivação deste trabalho é descobrir se existem características que possam ser vistas já na fase conceitual do projeto de banco de dados, que indiquem que o modelo orientado a grafos seja mais indicado que o modelo relacional para um determinado problema em questão. Para isso, primeiramente foi percebido a necessidade de diferenciar o processo de modelagem entre esses dois tipos de bancos de dados. Não obstante disso, também foi observada a falta de um processo bem estruturado para a modelagem de dados no MoG, o que instigou os autores a realização de um estudo aprofundado, afim de propôr processo de modelagem mais estruturado.

1.1 Objetivos

O objetivo principal é delimitar as diferenças entre os processos de modelagem de bancos de dados relacionais e bancos de dados orientados a grafos, além de buscar estabelecer um processo mais sistematizado de projeto conceitual e lógico de BDGs.

Especificamente, pretende-se classificar e tipificar os problemas nos quais cada modelo de dados é mais indicado. Além disso, busca-se elucidar as diferenças entre o modelo relacional e modelo orientados a grafos, com foco nas etapas do projeto conceitual.

[1] http://neo4j.com/
[2] http://orientdb.com/
[3] http://titan.thinkaurelius.com/
[4] http://www.sparsity-technologies.com/
[5] https://walmart.com.br/
[6] https://www.lockheedmartin.com/en-us/capabilities/space.html
[7] https://www.ebay.com/
[8] https://www.ubs.com/br/en.html
[9] https://www.monsantoglobal.com/global/br/Pages/default.aspx
[10] https://www.cat.com/

1.2 Organização do Trabalho

Esse trabalho divide-se em cinco capítulos. Este primeiro fez uma introdução do assunto e contextualizou a motivação e os objetivos do trabalho. O capítulo 2 apresenta o estado da arte nas tecnologias de bancos de dados. Nele é apresentada a revisão bibliográfica com os principais conceitos e os trabalhos correlatos encontrados na literatura.

No capítulo 3 são modelados casos de uso, a fim de cumprir o objetivo de encontrar características do projeto conceitual que apontam para a utilização do modelo relacional ou do MoG, bem como é introduzida a proposta de estruturação do processo de modelagem para BDGs.

Os resultados são expostos no capítulo 4, enumerando as características encontradas no processo e descrevendo detalhadamente a proposta de metodologia de modelagem para BDs em grafos. Por fim, no capítulo 5 apresenta as considerações finais e propostas de trabalhos futuros.

2 Estado da Arte

Este capítulo contempla a revisão bibliográfica realizada e objetiva levar o leitor à exploração de conceitos importantes em bancos de dados, dando ênfase ao modelo orientado a grafos. O modelo relacional é brevemente apresentado, bem como outros modelos NoSQL.

O modelo relacional é apresentado na seção 2.1. Em seguida, o modelo orientado a grafos é abordado, com mais profundidade na seção 2.2. No seção 2.3 serão abordados alguns conceitos de bancos de dados não-relacionais em geral, bem como os outros modelos NoSQL. Também será apresentado o modelo *Abstract Model* (NoAM), que visa a introdução de uma etapa conceitual no projeto de BDs não-relacionais. Por fim, a seção 2.4 apresenta alguns trabalhos existentes de comparação entre BDs relacionais e orientados a grafos.

2.1 Modelo Relacional

O modelo relacional foi proposto por Codd (1970) e usa o conceito de relação matemática. Nesse modelo, o banco de dados é visto como uma coleção de relações e cada tupla da relação representa um fato. Outra nomenclatura para relações e tuplas é tabelas e linhas, respectivamente. Heuser (2008) define o modelo relacional como uma descrição dos tipos de informações que estão armazenadas em um banco de dados. Em um banco de dados relacional, os dados são armazenados em tabelas. Estas têm uma estrutura que se repetem a cada linha, de forma muito similar a uma planilha. Os relacionamentos entre as tabelas são feitos através da utilização de chaves.

Ainda segundo Heuser (2008), em um banco de dados relacional, há ao menos dois tipos de chaves a serem considerados: a chave primária, e a chave estrangeira. Uma chave primária é uma coluna ou uma combinação de colunas cujos valores distinguem uma linha (geralmente chamada de tupla) das demais dentro de uma tabela. Ou seja, eles são atributos únicos a cada uma das tuplas de cada uma das diferentes tabelas. A chave estrangeira é uma coluna ou uma combinação de colunas, cujos valores aparecem necessariamente na chave primária de uma tabela e em tabelas diferentes como atributo com a finalidade de formar uma espécie de par entre seu valor e o valor da chave primária desta outra tabela. Estes pares de chaves primárias e estrangeiras compõe um relacionamento entre as tuplas em questão.

Há ainda outros conceitos importantes pertinentes aos BDs relacionais como: restrições do modelo relacional; propriedades ACID; controle de transações; controles de concorrência; dentre outros. Esses conceitos fogem ao escopo deste trabalho, e podem ser

encontrados Heuser (2008), Elmasri e Navathe (2010) e Date (2004).

O processo de modelagem de sistemas de banco de dados relacionais é composto pelo projeto conceitual, com a construção de diagramas de Entidades-Relacionamentos (ER), o projeto lógico, com o mapeamento dos digramas ER para o modelo relacional de dados e no projeto físico, com a implementação do banco de dados com o uso de um SGBD. A seção 2.1.1 detalha o processo de modelagem de bancos de dados relacionais.

2.1.1 Projeto de Bancos de Dados Relacionais

O processo de modelagem de BDs relacionais pode ser dividido em três etapas (HEUSER, 2008). Na primeira, chamada conceitual, são especificadas as entidades e os relacionamentos entre elas. Na etapa lógica, a segunda, são projetados os esquemas relacionais, com a representação de tabelas, atributos e relacionamentos entre si. A fase final do processo é o projeto físico, com o uso de um SGBD relacional.

Uma das formas de executar o projeto conceitual de banco de dados é usando o modelo Entidade-Relacionamento (ER), proposto por Chen (1976). Tal modelo busca uma visão mais unificada dos problemas e de seus dados correlatos. O modelo Entidade-Relacional foi estendido por Teorey, Yang e Fry (1986) a fim de cobrir características dos dados não previamente abordadas no trabalho de Chen.

O modelo ER usa dois conceitos básicos: entidades e relacionamentos entre essas entidades. A representação desses conceitos é feita de forma visual. Em um diagrama ER, as entidades são vistas em retângulos e o relacionamento por losangos. Além disso, é possível expressar restrições de participação no relacionamento, via indicação de cardinalidades. As cardinalidades podem ser tanto máximas, quanto mínimas. Na Figura 1 tem-se duas entidades: *EMPREGADO* e *MESA*, ligados pelo relacionamento *ALOCAÇÃO*. Os números entre parênteses indicam que uma instância de empregado pode estar relacionada a no mínimo 1 e no máximo uma instância de mesa, enquanto uma instância de mesa pode ter zero ou um empregado alocado.

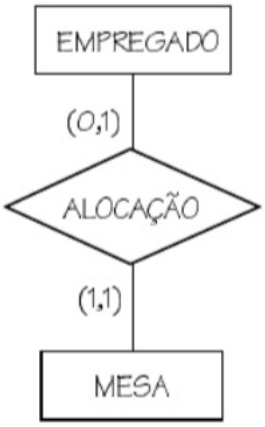


Figura 1 – Exemplo de Diagrama ER com cardinalidades.

Fonte: Retirado de Heuser (2008).

A Figura 2 mostra outros conceitos e representações visuais utilizadas em um diagrama ER.

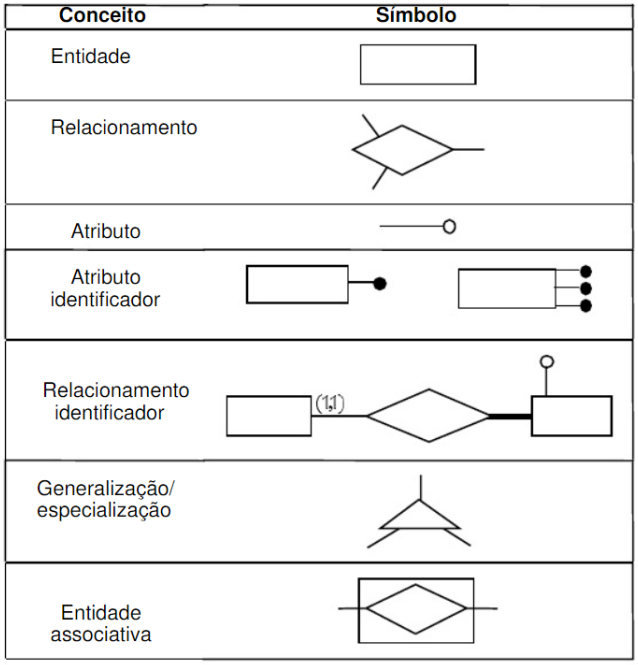


Figura 2 – Representações gráficas do modelo ER.

Fonte: Retirado de Heuser (2008).

A etapa de projeto lógico é realizada com o mapeamento do esquema ER para o modelo relacional. Em resumo, entidades e relacionamentos são mapeados para tabelas e/ou chaves estrangeiras. A Figura 3 apresenta as regras de mapeamento utilizadas para

os relacionamentos. Um detalhamento maior sobre a etapa de mapeamento ER-Relacional pode ser visto em (ELMASRI; NAVATHE, 2010; DATE, 2004; HEUSER, 2008).

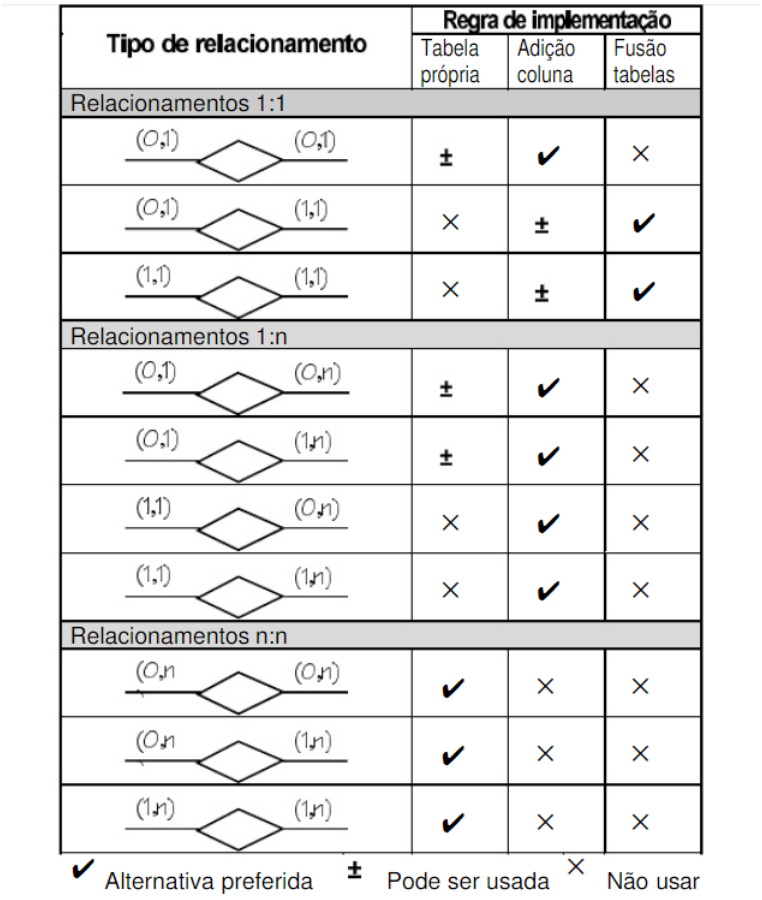


Figura 3 – Regras de mapeamento ER-Relacional.

Fonte: Retirado de Heuser (2008).

Dentro do processo de modelagem de BDs relacionais é válida a observação do conceito de normalização, que apesar de ter sua concepção oriunda da ambição de se ser uma forma de modelagem formal de banco de dados relacionais (DATE, 2004), também pode ser vista como uma sub-etapa da fase lógica. A normalização objetiva a eliminação da redundância dos dados através da utilização de regras que atendam a cada uma das formas normais. De modo resumido, pode-se dizer que a normalização ocasiona a criação de novas tabelas. Isso implica na obtenção de maior qualidade do projeto lógico e isso se traduz não apenas em uma compreensão mais clara dos projetistas com relação aos dados e seus relacionamentos, mas também em uma melhor utilização do armazenamento do BD visto a eliminação de dados redundantes.

A última etapa da modelagem corresponde a construir fisicamente o banco de dados. Essa etapa corresponde à utilizar um SGBD relacional e a linguagem SQL para implementação das tabelas descritas no modelo lógico. Como foi esclarecido na introdução 1 o detalhamento desta fase foge ao escopo desse trabalho.

2.1.2 Desvantagens do Modelo Relacional

O modelo relacional apesar de muito amadurecido e utilizado, tanto comercialmente quanto academicamente, apresenta algumas desvantagens, dentre elas as principais são: a escalabilidade; a operação de junção (*join*); e o uso com dados hierárquicos.

Para Vaish (2013), a escalabilidade é definida como a habilidade de um sistema aumentar seu *throughput*, com a adição de recursos para contrabalancear o aumento da carga demandada dele. Isso pode ocorrer de duas formas: a vertical, que consiste em disponibilizar um único recurso poderoso o suficiente para lidar com a carga; e a escalabilidade horizontal, que envolve dividir a carga entre *clusters* de sistemas mais rentáveis.

Escalabilidade é uma questão enfática se tratando da crescente utilização de sistemas na *web*. Aplicações *web*, além de lidar com um volume muito grande de dados, necessitam suportar inúmeras operações de leitura e escrita em disco, bem como, precisam manter tempos de resposta o mais baixo possível, sem contar a preocupação em manter a disponibilidade dos dados a todo momento.

O modelo relacional lida relativamente bem com esse tipo de problema, mas notou-se a necessidade de propor modelos novos, dado ao fato de que BDs relacionais são mais escaláveis verticalmente do que horizontalmente o que pode ser custoso para as organizações.

Existem soluções para mitigar esse problema no modelo relacional. A principal delas é o uso de bancos de dados distribuídos, onde se faz a distribuição da carga das consultas entre servidores físicos diferentes. O maior problema de utilizar esta solução é a replicação de dados. As informações tendem a ser replicadas entre os diferentes servidores por questão de consistência.

Outra desvantagem do modelo relacional é o custo da operação de junção. Segundo Eich e Mishra (1992), a junção é uma das operações fundamentais para as consultas em BDs relacionais. Ela permite associar tuplas de tabelas diferentes, permitindo a recuperação de dados correlacionados presentes em mais de uma tabela. A implementação dessa operação foi baseada no produto cartesiano entre tabelas, e é uma operação custosa para os SGBDs.

A junção é usada de forma constante em SGBDs relacionais. Para exemplificar a junção, as Tabelas 1 e 2 apresentam um exemplo de tabelas de *Aluno* e *Curso* respectivamente, enquanto a Tabela 3 mostra o resultado da junção, com a condição *A(idcurso) = B(idcurso)*.

Soluções implementadas através da utilização do modelo relacional tendem a gerar diagramas elegantes, com tabelas normalizadas, a fim de padronizar os dados e eliminar redundância. O problema nessas soluções é a incapacidade dos projetistas de prever todas mudanças que irão ocorrer durante a operação de um sistema. No caso destas mudanças

Tabela 1 – Tabela de Aluno

Aluno	**id_curso**
Thales	1
Amanda	2
Bruno	3
Isabela	1

Fonte: Elaborada pelo próprio autor.

Tabela 2 – Tabela de Curso

id_curso	**Curso**
1	Sistemas de Informação
2	Logística
3	Engenharia de Computação

Fonte: Elaborada pelo próprio autor.

Tabela 3 – Resultado da Junção entre as tabelas Aluno e Curso

Aluno	**id_curso**	**id_curso**	**Curso**
Thales	1	1	Sistemas de Informação
Amanda	2	2	Logística
Bruno	3	3	Engenharia de Computação
Isabela	1	1	Sistemas de Informação

Fonte: Elaborada pelo próprio autor.

afetarem a própria estrutura dos dados, isso irá causar a desnormalização das tabelas e, consequentemente, uma degradação considerável de desempenho das consultas e a duplicação de dados.

Alguns problemas podem ser melhor modelados através da utilização diagramas que usualmente não são muito diferentes de árvores ou dígrafos (TSICHRITZIS, 1976). Ao modelar os dados de um problema, pode haver um caso especial, em que a estrutura tem forma similar a uma árvore, em que as arestas no diagrama partem da raiz e seguem sempre na direção oposta dela. Além disso, a cada nível da árvore um nó possui um nó pai único.

A este caso muito particular é dado o nome de dados hierárquicos como definido em Tsichritzis (1976). Essa definição pode ser observada através do diagrama exemplo da Figura 4 que modela a hierarquia entre as academias licenciadas a promover cursos de certificação da empresa Cisco. É possível observar que os dados estão em forma de árvore

e que cada nó filho tem apenas um nó pai, na raiz. E no topo da hierarquia está a própria empresa Cisco, abaixo dela está a unidade da empresa responsável pelas academias de treinamento, no caso a *Cisco Academy Training Center*. Subordinadas a essa unidade estão as academias regionais e por fim na parte mais baixa da hierarquia estão as academias locais que são subordinadas as academias regionais.

Figura 4 – Exemplo de modelagem de Hierarquia.

Fonte: Retirado de Salgado, Fonseca e Times (2016).

Dados muito hierarquizados são um potencial problema para o modelo relacional, pois a cada nível de hierarquia, uma junção precisa se realizada para recuperar as informações relevantes as consultas realizadas sobre elas.

Como é possível observar, apesar de amplamente utilizados e de muito amadurecidos, os BDs relacionais possuem pontos fracos. A abordagem de cientistas de dados no tratamento destes problemas difere ao passo que, enquanto alguns buscam mitigar esses problemas através da melhoria dos BDs relacionais, outros criaram modelos baseados em ideias diferentes, objetivando não ter de lidar diretamente com essas barreiras.

A seção 2.2 apresenta os bancos de dados orientados a grafos, com seus conceitos e o processo de modelagem proposto na literatura.

2.2 Modelo Orientado a Grafos

O modelo de bancos de dados orientado a grafos, é baseado na ideia fundamental de estruturar um banco de dados como um grafo, portanto para melhor compreender BDs em grafos é necessário inicialmente abordar conceitos de grafos.

Segundo Harris, Hirst e Mossinghoff (2008) um grafo é um par G = (V, E), onde V representa um conjunto de vértices e E um conjunto de arestas. Vértices e arestas podem ser separados em tipos através da utilização de rótulos (*labels*). Em termos de representação, os rótulos são nomes dados aos elementos do grafo.

A Figura 5 exemplifica os conceitos básicos de um grafo, nela os círculos são os vértices e as linhas conectando os vértices são as arestas. O grafo do lado esquerdo da figura possui tanto vértices quanto arestas rotuladas. O grafo do lado direito apresenta o mesmo grafo sem rotulação.

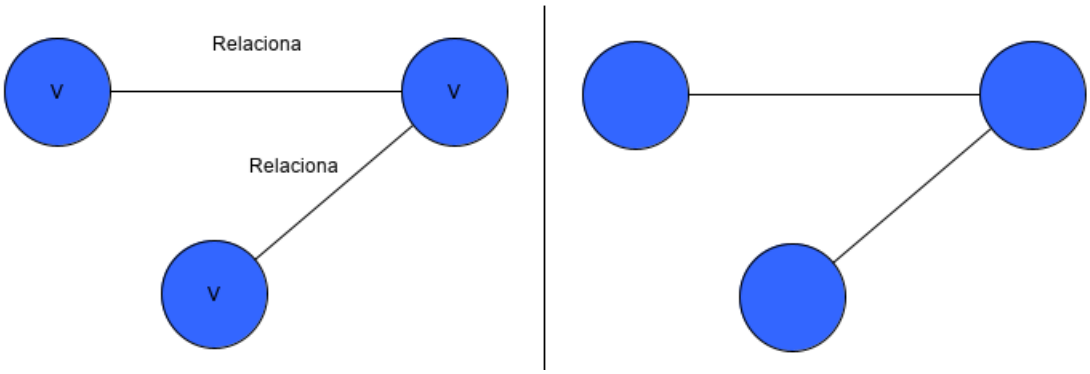


Figura 5 – Grafos rotulados e não-rotulados.

Fonte: Adaptado de Szwarcfiter (1986)

Para os grafos, os rótulos de vértices e arestas é opcional. Porém, para os BDs de grafos, esses rótulos servem para tipificar os elementos e dados. Nesse sentido, a rotulação dos vértices e arestas facilita a compreensão do modelo e acarreta no aumento de desempenho das consultas, como poderá ser visto na seção 2.4.

As arestas de um grafo podem ou não ter uma direção indicada, nesse caso o grafo é dito ser orientado. Conforme Harris, Hirst e Mossinghoff (2008) um grafo é orientado, caso os pares de vértices que o compõe sejam ordenados, isso implica no fato de que as sua arestas são direcionadas e que sua representação seja feita através da utilização de setas e não de linhas.

O conceito de grafo direcionado é exemplificado através da comparação contida na Figura 6, do lado direito há um grafo não direcionado, nele há relacionamentos entre os vértices representados por arestas rotuladas como por exemplo as arestas com o rótulo *Ensina*. Apesar de estabelecer o relacionamento não é possível saber quem ensina quem. Já o grafo do lado esquerdo é direcionado, e é possível fazer a diferenciação os papéis dos vértices nos relacionamentos.

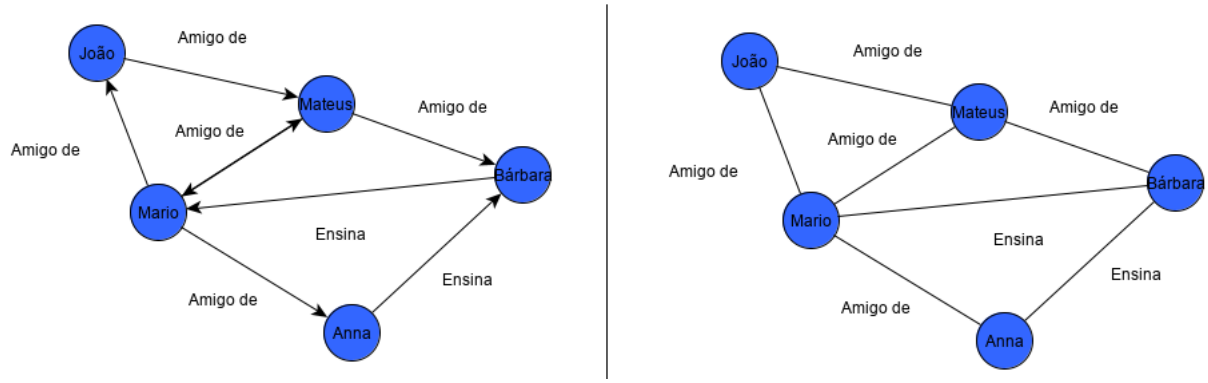


Figura 6 – Exemplo de grafo direcionado.

Fonte: Próprio autor.

Para que o conceito de grafo possa ser utilizado em programas de computador, é utilizar algum das formas existentes de implementação: matriz de incidência; lista de adjacências; e/ou matriz de adjacência.

Na é a matriz de adjacências, Figura 7, um grafo é representado por uma matriz de 0's e 1's com colunas e linhas indexadas pelos vértices. Quando existir uma aresta entre dois vértices o valor da matriz, referenciada pelo índice de linha e pelo índice de coluna, corresponderá a 1, se for um vértice de origem, e -1 seja um vértice de destino. No caso de não haver aresta entre os dois vértices o valor da matriz será 0.

Embasado nesta definição, é possível inferir que um rótulo é implementado através da criação de um vértice de adjacências. Uma descrição mais detalhada do modelo utilizado para banco de dados em grafos pode ser encontrada em Angles (2018).

$$\widetilde{M}_{i,j} = \begin{cases} 1 & \text{se } i = j \\ -1 & \text{se } (i,j) \in E \\ 0 & \text{caso contrário} \end{cases}$$

Figura 7 – Definição da matriz de adjacência de um dígrafo.

Fonte: Adaptado de Szwarcfiter (1986)

O conceito de matriz de incidência é similar a matriz de adjacência, mas que ao invés de as duas dimensões da matriz serem indexadas utilizando vértices, a indexação é feita em apenas uma das dimensões com os identificadores dos vértices e a outra dimensão é indexada com os identificadores das arestas. Já as listas de adjacências são estruturas que armazenam, para cada vértice, os identificadores de todos os vértices que possuem relacionamentos com ele. Mais detalhes sobre as representações computacionais de grafos podem ser encontrados em Szwarcfiter (1986).

No trabalho de Engels (1990) é discutido o uso de grafos para representação dos estados de um banco de dados, traçando uma relação inicial sobre os dois conceitos, já em Robinson, Webber e Eifrem (2015) é explicitado que bancos de dados em grafos são implementados utilizando grafos direcionados.

Não existe uma definição aceita como padrão, até o momento para BDs orientados a grafos. A mais utilizada na literatura é a apresentada por Pokorny (2016). Nessa definição, um banco de dados em grafo G = (V, E, N, Σ, ρ, A, Att) é um multigrafo dirigido, rotulado e com atributos. Onde V é um conjunto finito de vértices com rótulos desenhados de um alfabeto infinito N, E é um conjunto de arestas, ρ é uma função incidente do mapeamento E para $V \times V$. Os rótulos de arestas são retirados do conjunto finito de símbolos Σ, e α é a função de mapeamento de rotulagem E para Σ, Att é uma atribuição de mapeamento para cada nó e aresta de um subconjunto que provavelmente será vazio de atributos de A.

Basicamente em um BD em grafos, o grafo mapeia os dados em vértices e arestas. Os vértices representam os dados, e as arestas representam os relacionamentos existentes entre eles. Vale relembrar que bancos de dados de grafos priorizam os relacionamentos entre os dados (GOONETILLEKE et al., 2019).

Diferentemente dos bancos de dados relacionais, o banco de dados orientado a grafos baseia-se em nos conceitos de NoSQL. O MoG, também não é orientado a esquemas ou modelos conceituais, e busca flexibilizar a consistência dos dados simplificando o processo de modelagem e evitando custosas como as junções. Além disso, a priorização de relacionamentos advém da possibilidade de fazer uso de algoritmos de grafos na implementação de consultas.

2.2.1 Modelagem de Bancos de Dados Orientados a Grafos

Problemas que utilizam bancos de dados relacionais como soluções, primeiramente geram um diagrama ER que é mapeado para o modelo relacional. O equivalente ao modelo ER, quando deseja-se usar o MoG, é o processo de *whiteboarding* (ROBINSON; WEBBER; EIFREM, 2015). O processo de modelagem com o *whiteboarding* inicia-se diretamente desenhando um grafo com algumas instâncias para representar seus elementos e relacionamentos, onde as entidades serão representadas pelos vértices do grafo e os relacionamentos entre eles serão indicados por arestas direcionadas.

Os vértices podem ter rótulos, a fim de agrupar os mesmos em subgrafos, reduzindo consideravelmente o tempo de resposta a consultas sobre os mesmos. As arestas também podem possuir rótulos, que definirão qual a relação que as mesmas estão indicando entre os vértices.

Por isso, a próxima fase do processo, naturalmente consiste-se do enriquecimento da visão inicial, buscando através do aprimoramento do grafo previamente concebido, a

garantia que para cada entidade no domínio do problema sejam definidos seus papéis relevantes através desses rótulos.

Os atributos dessas entidades são descritos como propriedades dos vértices e as arestas do grafo representarão as relações entre as entidades. A Figura 8 mostra um exemplo de modelagem de um banco de dados em grafos. A imagem mostra dois nós: *Pessoa* e *Departamento* com uma aresta indicando a relação de uma *Pessoa Lotar* um *Departamento*.

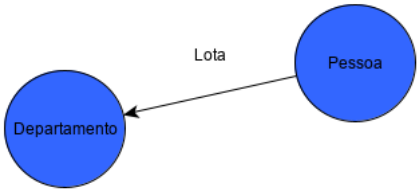


Figura 8 – Exemplo de modelagem em grafos.

Fonte: Próprio autor.

Segundo Robinson, Webber e Eifrem (2015), a abordagem orientada a grafos é ideal para modelagem de problemas que exigem estruturas mais flexíveis para o banco de dados e/ou ou cujo volume de dados tende a crescer exponencialmente justamente por não se ater a rigidez estrutural. A Figura 9 mostra a comparação de diagramas dos modelos relacionais e de grafos para um problema de *data centers*. É possível observar como, em um primeiro momento, o processo de *whiteboarding* é feito via instâncias particulares de elementos, enquanto na modelagem relacional, cada elemento particular já é agrupado em uma entidade.

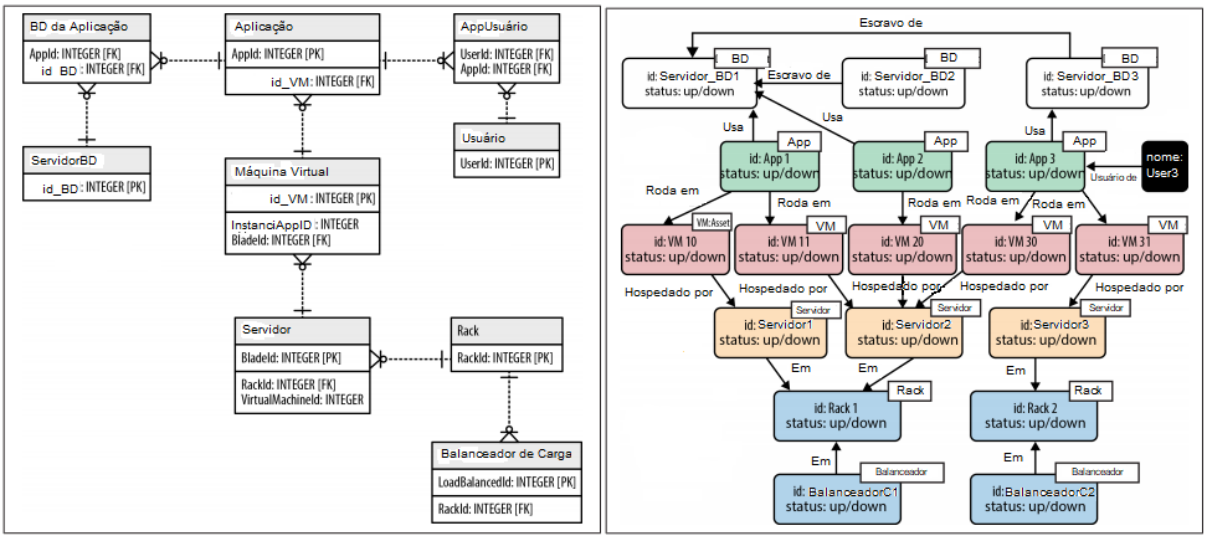


Figura 9 – Comparativo entre diagramas

Fonte: Adaptado de Robinson, Webber e Eifrem (2015).

A próxima etapa do processo de modelagem é a implementação do banco de dados fisicamente utilizando um SGBD e uma linguagem específica. Este trabalho não aborda o projeto físico de BDs. Cabe observar que, diferentemente do que ocorre com BDs relacionais, e principalmente com a SQL, não existe uma linguagem padrão para SGBDs orientados a grafos. As duas linguagens mais utilizadas são a Cypher, que é uma linguagem prioritária do Neo4J; e a Gremlin, utilizada em alguns SGBDs como o JanusGraph e o TitanDB. Mais informações sobre a Cypher pode ser encontradas em Robinson, Webber e Eifrem (2015) e na seção do site correspondente do Neo4J [1]. Já com relação a Gremlin, mais informações podem ser encontradas na página do JanusGraph [2], e na página guia da *Apache Tinker Pop*[3]

2.2.2 Desvantagens de Bancos de Dados Orientado a Grafos

O principal problema com relação ao MoG é a grande falta de padronização que predomina nas implementações dos SGBDs de grafos. Não apenas a questão da linguagem utilizada nas consultas não é padronizada, mas também a própria implementação de BDGs pode ser diferente entre os SGBDs. Um exemplo é o SGBD JanusGraph, que permite a implementação de um BD no formato *Resource Description Framework* (RDF), onde cada vértice do grafo armazena todas informações da entidade que modela, bem como um *link* para os vértices com os quais se relaciona. Deste modo a implementação tanto dos vértices quanto das arestas do grafo assemelham muito aos documentos em BDs orientados a documentos. Essa implementação é muito diferente do modo matricial, que é o modo mais comum de implementação de BDs no MoG, (ANGLES, 2012).

A falta de padronização se traduz em outro problema, que é a imaturidade das ferramentas de BDGs. Sem padronização, cada empresa tenta desenvolver e aprimorar sua própria forma de implementação. Isso representa um obstáculo frente ao avanço tecnológico, que seria consideravelmente maior, caso as pesquisas fossem mais focadas e embasadas em padrões.

Além disso, soma-se o fato de que o MoG é um conceito recente em termos de tecnologia, o que aumenta ainda mais sua deficiência em termos de maturidade frente ao modelo relacional.

Os dois problemas supracitados geram um terceiro, que é a falta de mão de obra especializada em BDs orientados a grafos. A despeito da utilização de SGBD de grafos por grandes empresas, ainda é difícil encontrar profissionais qualificados e aptos a realizar a implementação e manutenção essas soluções.

Por fim, o mais notável problema das tecnologias disponíveis no MoG é a completa

[1] https://neo4j.com/developer/cypher-query-language/
[2] https://docs.janusgraph.org/latest/gremlin.html
[3] https://kelvinlawrence.net/book/Gremlin-Graph-Guide.

ausência da implementação de mecanismos de segurança. Nos SGDBs orientados a grafos, sequer são implementados níveis de acesso e grupos de segurança, portanto qualquer usuário tem acesso a todas informações desde que tenha o conhecimento para realizar a consulta que retorne as informações desejadas. Isso é algo muito grave pois fere diversas certificações e procedimentos organizacionais já bem estabelecidos no mercado.

Corroborando com Robinson, Webber e Eifrem (2015), alguns casos de uso do MoG são citados na literatura, e em todos eles tanto a característica de estrutura flexível, quanto de grande volumes de dados está presente. Conforme Fowler e Sadalage (2012), os casos de uso do MoG consistem-se nos seguintes: aplicações do domínio social como as redes sociais; aplicações de domínio espacial, como os problemas de roteamento e transporte; e de comercio, como os motores de recomendação de uma plataforma de *e-commerce*.

Em Huang et al. (2002), é descrita uma aproximação baseada em grafos para um sistema de recomendação de livros para uma biblioteca digital. O sistema é dividido em duas camadas, sendo uma delas a de clientes e a outra de livros. Na camada de livros, os vértices são usados para representar os livros e as arestas indicam similaridade entre seus conteúdos. Na camada de clientes, os vértices indicam clientes e as arestas indicam similaridades demográficas. Arestas entre os clientes e livros, indicam as compras realizadas pelos mesmos. Muitos outros sistemas de recomendação funcionam de forma similar a essa.

2.3 NoSQL

Os modelos NoSQL surgiram não apenas dos problemas do modelo relacional, mas pela necessidade de uma nova perspectiva em termos de armazenamento de dados, cujo volume era cada vez maior, o crescimento cada vez mais rápido e as informações cada vez mais distribuídas. A maior motivação por trás de modelos não relacionais é a escalabilidade horizontal, SGBDs não relacionais são projetados para uso em sistemas de forma distribuída.

A abordagem NoSQL objetiva mitigar os problemas do modelo relacional. Vicknair et al. (2010) cita quatro características de problemas que apontam para o uso dos modelos NoSQL: (1) tabelas com muitas colunas; (2) existência de várias tabelas representando relacionamentos *n:m*; (3) dados com características hierárquicas ou de árvores e; (4) alterações frequentes na estrutura do BD.

Outra característica dos modelos NoSQL é o fato de serem modelos sem esquemas pré-definidos (*schemaless*) (SULLIVAN, 2015). Isso permite que as estruturas utilizadas nos diferentes modelos sejam mais flexíveis que o modelo relacional, especialmente com relação aos seus atributos.

É possível perceber, com isso, que alguns autores como Robinson, Webber e Eifrem

(2015) e Vaish (2013), acabem por omitir a etapa de projeto conceitual do BD, iniciando o projeto do banco de dados diretamente pela etapa de projeto lógico.

Bancos de dados NoSQL se baseiam nas propriedades denominadas de BASE (*Basically Available, Soft state, Eventually consistent*) (WEBER, 2010). Essas propriedades indicam que não há a exigência de disponibilidade a toda hora, ou seja, que os dados podem estar apenas parcialmente disponíveis; que uma informação pode ser sobrescrita eventualmente com uma versão própria mais recente e que, o BD precisa ter apenas uma consistência eventual, isso significa que podem haver momentos em que o BD não está num estado consistente rígido

Esse conceito é complementado pelo teorema CAP que discute os aspectos de consistência, disponibilidade e tolerância a particionamento. A consistência diz respeito à replicação dos dados nos servidores; a disponibilidade refere-se a acessibilidade dos dados; e a tolerância a partição é a qualidade do banco de dados continuar operacional mesmo sob uma falha parcial. O teorema CAP afirma que apenas duas dessas características podem ser oferecidas ao mesmo tempo, o que obriga os projetistas a direcionar quais delas irão ser prioridades para sua aplicação (GILBERT; LYNCH, 2012).

Bancos de dados NoSQL ainda seguem mais dois princípios fundamentais, o de desnormalização de tabelas e o de agregação de dados. O processo de desnormalização de tabelas é formalmente definido em Yoo, Lee e Jeon (2018) como a unificação de tabelas com base nos relacionamentos existentes entre elas. Esse processo é realizado com o intuito e eliminar a tabela cujo conteúdo está replicado no banco de dados.

Um dos benefícios da desnormalização é a redução da quantidade de junções necessárias para a recuperação da informação. Em contrapartida, esse processo insere redundância de dados nas tabelas, bem como, aumenta o tamanho das mesmas. É necessário observar que a introdução de redundância de dados, bem como o aumento da quantidade de atributos nas tabelas gerado pelo processo, tornam as operações de atualizações e inserção mais caras computacionalmente (WEBER, 2010).

Já agregação de dados é o processo de agrupar informações, ao ponto de algumas vezes ser capaz de expressá-las de forma sumária (YOO; LEE; JEON, 2018). Em BDs relacionais esse agrupamento é feito com a cláusula GROUP BY da SQL, que agrupa dados dado um valor de atributo. Outros exemplos são as as funções de agregação existentes na SQL: COUNT, MAX, MIN, AVG e SUM.

Todas as funções supracitadas agrupam dados idênticos ou de similaridade considerável, a fim de facilitar a recuperação de informação pelas consultas dos usuários. Agregação nos bancos de dados NoSQL tem os papéis não apenas de facilitar a recuperação de informações, mas também aumentar o desempenho do processamento das consultas.

Na subseção 2.3.1 serão abordados, brevemente, os outros três modelos de bancos de

dados não relacionais existentes: (1) modelo chave-valor, (2) modelo orientado a documentos (3) modelo de famílias de colunas.

2.3.1 Outros Modelos NoSQL

No modelo chave-valor existem apenas dois elementos: uma chave e um valor. Cada chave é um identificador que estará associado a um valor, que representará um dado no BD. A modelagem deste tipo de banco de dados limita-se a construção dos padrões das chaves (VAISH, 2013).

Para exemplificar, conforme mostrado na Figura 10, considerando o projeto das matrículas (chaves) para *ALUNOS* em uma universidade. Geralmente, deseja-se saber o nome do aluno, o curso em que o mesmo está matriculado e o campus onde ele irá participar das aulas. Para cada item a ser armazenado, cria-se uma chave concatenando o número da entidade, no caso o número sequencial correspondente ao índice numérico interno do banco de dados, o nome da entidade bem como do atributo ao qual a mesma estará associada.

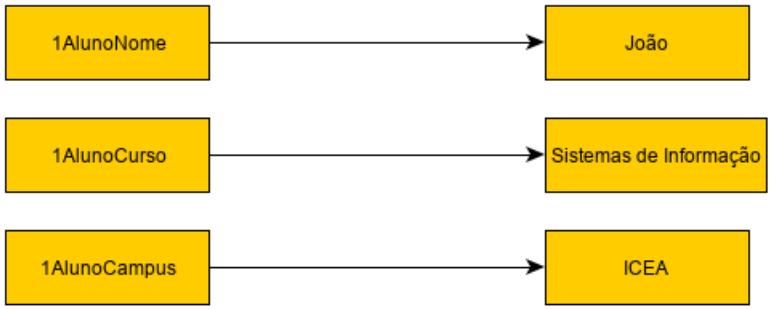


Figura 10 – Exemplos de pares chave-valor.

Fonte: Próprio autor.

No exemplo considera-se a matrícula de um hipotético primeiro *ALUNO* do campus ICEA, com o nome de *João* e curso a ser matriculado é o de *Sistemas de Informação*. De acordo com essas especificações foram geradas as chaves: *1AlunoNome* que aponta para o valor do atributo *Nome* do *ALUNO*. *1AlunoCurso* que aponta para o valor do atributo *Curso* que é *Sistemas de Informação* e *1AlunoCampus* que aponta para o valor do atributo *Campus* onde o discente foi matriculado cujo valor é *ICEA*. Mais detalhes podem ser encontrados em Nayak, Poriya e Poojary (2013).

O modelo orientado a documentos baseia-se na estruturação dos dados em *strings*, ou nas representações binárias das mesmas (SULLIVAN, 2015). Esse modelo funciona de um modo similar ao modelo chave-valor. Os documentos, em geral, são representadas por arquivos do formato *Extensible Markup Language* (XML) e *JavaScript Object Notation* (JSON) e lidam com dados ditos semi-estruturados. A principal diferença entre o modelo

orientado à documentos e o modelo chave-valor é que, ao invés de associar uma chave a um valor de atributo, uma única chave é associada a múltiplos atributos. A Figura 11 contém um exemplo de informações de um aluno armazenadas no formato de um JSON. Bancos de dados orientados a documentos são abordados mais detalhadamente em Vaish (2013).

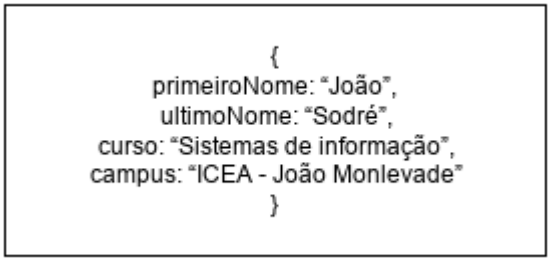


Figura 11 – Entidade Aluno no modelo orientados a documentos.

Fonte: Próprio Autor.

O modelo baseados em famílias de colunas é apresentado em Fowler e Sadalage (2012) e baseia-se na construção de blocos de estruturas para representar os dados. Este modelo tem duas estruturas básicas: as linhas ou famílias de colunas e as colunas propriamente ditas, que são a unidade básica de armazenamento. A ideia que compõe as colunas é um par entre uma chave de coluna e o valor do conteúdo da coluna. Bancos de dados orientados à famílias de colunas são compostos por conjuntos de linhas, um conjunto de colunas compõe uma linha, sendo que linhas podem possuir, ou não, as mesmas colunas. O valor de cada atributo de coluna é modelado como uma par de chave-valor

Na Figura 12, é exemplificada a modelagem desse tipo de banco de dados. Foi utilizado o mesmo exemplo da figura 10 onde está sendo modelado um *ALUNO* do campus *ICEA* chamado *João* que cursa *Sistemas de Informação*. Em BDs orientados a famílias de colunas cada um dos atributos de *ALUNO* é modelado como uma coluna diferente, os valores destes atributos são guardados internamente como pares de chave-valor. O agrupamento das colunas dos atributos de *João* formam sua respectiva linha. Ainda é possível agrupar diferentes linhas em estruturas de tabelas muito semelhantes as tabelas do modelo relacional. Neste caso as tabelas tipificam as linhas, o que no exemplo ocorre com *João* que é modelado como uma linha do tipo um *ALUNOS*. Outros detalhes sobre essa abordagem podem ser encontrados em Fowler e Sadalage (2012).

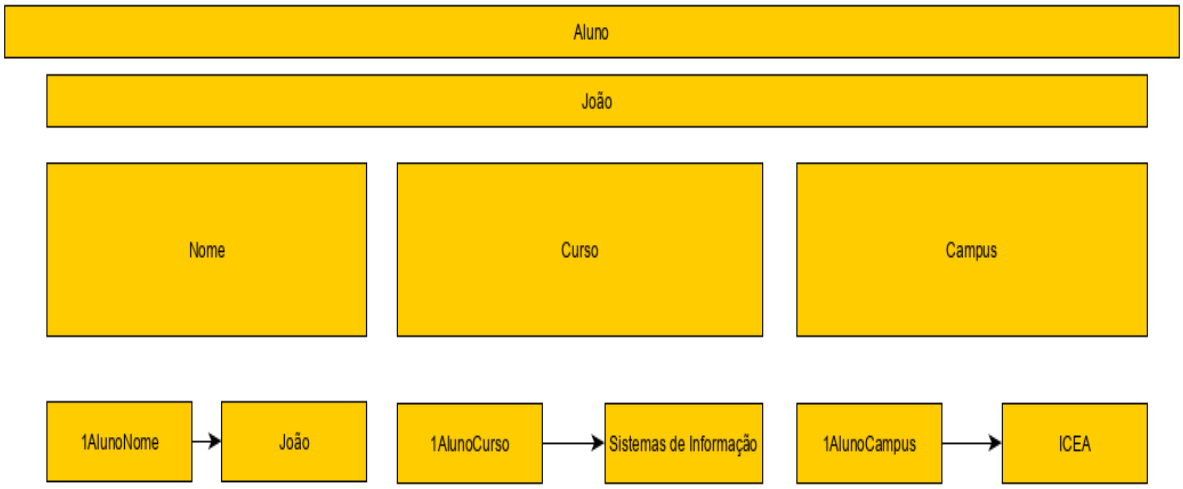


Figura 12 – Diagrama exemplo da modelagem de uma coluna em BDs orientados a famílias de colunas.

Fonte: Próprio Autor.

A subseção (2.3.2) aborda um modelo para a fase conceitual do projeto de bancos de dados NoSQL. Este modelo visa abranger todos os tipos de BDs não relacionais, de forma a ser flexível suficiente para ser adotado pelos projetistas.

2.3.2 Modelagem Conceitual NoSQL - NoAM

Bancos de dados NoSQL são vistos como independentes de esquemas, sem uma estrutura mais rígida e definida, como ocorre nos BDs relacionais.

Entretanto, por mais liberdade que tenham os modelos não relacionais, os problemas a serem modelados quase sempre possuem um certo grau de estruturação. Independentemente do modelo de banco de dados escolhido, os dados terão de ser mapeados em elementos de modelagem: como tabelas; conjuntos; vértices; documentos, entre outros. Isto significa que o projeto de bancos de dados NoSQL exige decisões de projeto cruciais que irão impactar diretamente requisitos de qualidade como o desempenho, a consistência e a escalabilidade.

A abordagem NoAM, inicialmente proposta em Bugiotti et al. (2014), e depois revisada em Bugiotti, Cabibbo e Torlone (2010), foi concebida visando não apenas identificar as características em comum compartilhadas pelos diferentes tipos de banco de dados NoSQL, mas também utilizar os mesmos a fim de organizar o processo de modelagem de dados.

O NoAM realiza uma modelagem estruturalmente mais consistente e, logicamente mais organizada dos dados, em um nível mais alto. Desta forma, o processo é aplicável a qualquer um dos mais de 50 SGBDs não relacionais disponíveis no mercado.

O processo NoAM baseia-se nas seguintes fases:

- Modelagem conceitual de dados e projeto de agrupamentos: Identificar as várias classes de objetos agregados necessários para a aplicação. Essa fase buscar atender os requisitos funcionais dos usuários, assim como, as restrições de escalabilidade e de consistência;
- Particionamento de agrupamentos: Consiste em dividir os agrupamentos em elementos menores. Essa fase visa atender aos requisitos de desempenho;
- Projeto do BD em alto nível: Cada agrupamento é mapeado para o modelo intermediário de dados NoAM;
- Implementação: Fase que mapeia os dados previamente modelados nos elementos específicos modelo escolhido e do SGBD.

Em Bugiotti, Cabibbo e Torlone (2010), observa-se inicialmente que todos modelos de bancos de dados NoSQL baseiam-se no conceito de entidades individuais atômicas, fundamentais para a sua modelagem. Cada entidade pode ser dos mais variados tipos como documentos, vértices e tuplas. Essas entidades serão a representação individual e indivisível e são denominadas de "unidade de acesso aos dados".

O principal objetivo dos BDs NoSQL é garantir o acesso, a persistência e a capacidade de manipulação dessas unidades de dados de forma eficiente. O agrupamento de "unidades de acesso aos dados"configura uma "unidade de distribuição"que são usualmente distribuídas entre diferentes servidores de um *cluster*.

Ruiz, Morales e Molina (2015) argumentam que bancos de dados NoSQL foram desenvolvidos para lidar exatamente com essas unidades de dados com ênfase nas unidades de distribuição. Portanto, todos os modelos NoSQL são modelos orientados a agregação, visto que todos lidam com a manipulação de dados agregados em grupos de unidades individuais.

Basicamente no NoAM, as entidades que populam o banco são modeladas através de um conjunto coleções. Cada coleção possui um nome distinto e é é representada por um conjunto de blocos, na Figura 13 existem duas coleções: *Player* e *Games*.

O modelo NoAM trata cada unidade de dados por um bloco, que representa a maior unidade de dados. É sobre o bloco que são garantidas as operações de acesso atômicas, eficientes e escaláveis. Acesso a blocos individuais ocorre de modo eficiente, enquanto o acesso simultâneo a múltiplos blocos é mais custoso, por equivaler operação de junção do modelo relacional. Cada bloco é identificado por uma chave de bloco, que é única dentro de uma coleção. Um bloco é um conjunto não vazio de entradas, e cada entrada é um par entre uma chave de entrada única dentro de seu bloco e seu valor.

Cada um dos jogadores, no caso *mary* e *rick*, é modelado como um bloco e eles, por sua vez, possuem blocos menores que armazenam os atributos e os valores desses atributos (Figura 13). Por exemplo, a *Player mary* tem os seguintes atributos simples: *username* = "mary"; *firstName* = "Mary"; e *lastName* = "Wilson". O bloco com o nome *games* seguido pelos [ ] indicam um relacionamento com outra entidade, no caso, com a entidade *games[0]*.

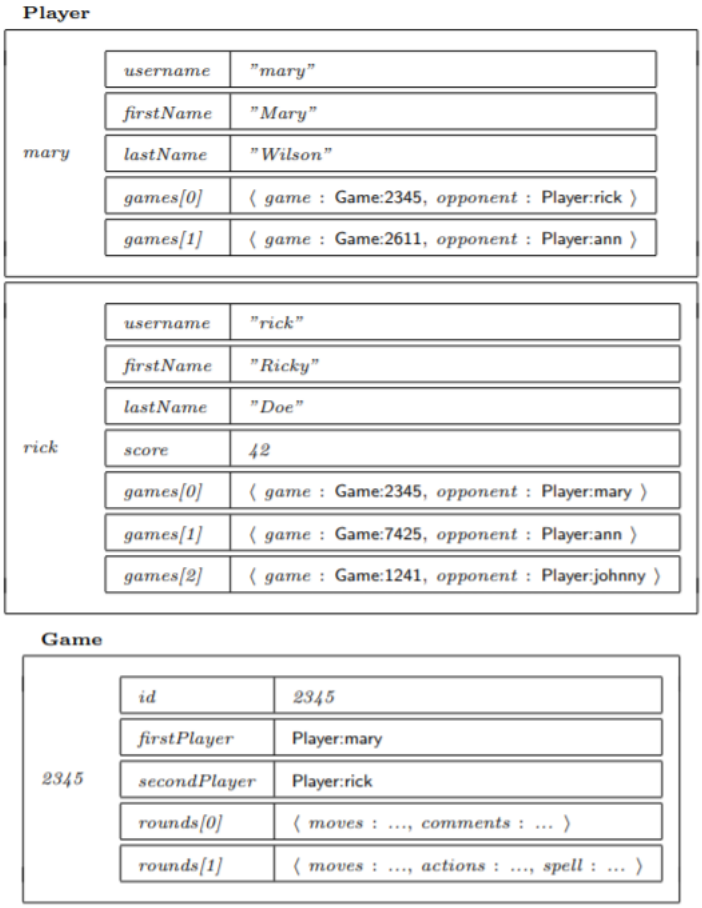


Figura 13 – Entidades de jogador e jogo modelada em blocos. A entidade de jogador representada pelo bloco "Player"e a entidade de jogos pelo bloco "Game".

Fonte: Retirado de Bugiotti, Cabibbo e Torlone (2010).

O primeiro passo da metodologia é a construção da representação conceitual dos dados através de seu mapeamento em entidades, relacionamentos e atributos. Em seguida, cada classe de agrupamentos deve ser mapeada a uma coleção diferente, sendo que cada uma dessas classes será visualmente representada por um bloco. No exemplo anterior, existem as duas classes de agrupamentos: *Player* e *Games*. O nome de cada entidade serve, dentro de uma coleção, como chave da coleção. Com as coleções e blocos todos definidos, mapeia-se o modelo NoAM para o modelo NoSQL desejado.

Na seção 2.4 serão feitas as considerações finais do capítulo de estado da arte, nela

serão citados alguns trabalhos que realizaram comparações entre os diferentes modelos de bancos de dados existentes.

2.4 Considerações Finais

Em Batra e Tyagi (2012), é realizada uma comparação entre o SGDB relacional MySQL e o SGBDs em grafos Neo4J. Os parâmetros de comparação foram: o nível de suporte e de maturidade que diz respeito a quão bem um sistema já foi testado e qual nível de suporte oferecido; a segurança; e a flexibilidade em termos de modificação do modelo estrutural. O trabalho destaca que os SGBDs relacionais são sistemas mais amadurecidos, dado o maior tempo de mercado e o nível de suporte oferecido. Quanto à segurança, foi mostrado que enquanto o MySQL oferece amplo suporte, tanto na questão pura de segurança quanto nos níveis de acesso aos usuários, o Neo4J não oferece qualquer recurso deste tipo. Por fim, via experimentação mostrou-se que a alteração da estrutura do banco de dados no Neo4J foi muito mais flexível.

Vicknair et al. (2010) comparam o desempenho de execução com dois tipos de consultas, tanto usando um banco de dados relacional quanto um banco de dados em grafo. Os bancos de dados foram criados com um mesmo número de tuplas/vértices, respectivamente. Tanto a quantidade de tuplas/vértices, quanto o tipo dos dados foram variados para cada experimento. Verificou-se, entre outros pontos que: (1) bancos de dados em grafos dispendem de mais espaço de armazenamento para um mesmo tipo de dados; (2) bancos de dados em grafos obtém melhores desempenhos em consultas estruturais que os BDs relacionais; (3) bancos de dados relacionais possuem melhor desempenho em consultas com dados numéricos; e (4) os BDs em grafos apresentam um desempenho melhor de consultas sobre caracteres.

Considerando os tópicos abordados neste capítulo, a Tabela 4 apresenta um resumo comparativo entre os modelos relacional e o modelo orientado a grafos.

No capítulo 3 é realizada a modelagem conceitual e lógica de problemas da literatura. Esta etapa do trabalho almejou não apenas comparar o processo em si, mas também identificar características particulares dos problemas que favorecem o uso do MoG.

Tabela 4 – Resumo Comparativo entre os Modelos: Relacional vs MoG.

	Modelo Relacional	Modelo Orientados a Grafos
Flexibilidade Estrutural		X
Segurança (Grupos e Níveis de Segurança)	X	
Melhor Performance em Consultas Estruturais		X
Melhor Performance em Consultas de Dados	X	
Maior Maturidade das Soluções	X	
Entidades mais Detalhadas	X	
Escalabilidade Horizontal		X
Possibilidade de Implementar Propriedades ACID	X	X

Fonte: Elaborada pelo próprio autor.

3 Bancos de Dados de Grafos

Com o objetivo de encontrar características que indiquem que tipo de problemas são melhor resolvidos com o uso de bancos de dados em grafos, realizou-se a projeto conceitual de problemas usualmente atribuídos a soluções usando o MoG.

Para cada problema foram realizado o projeto conceitual e o projeto lógico, usando tanto o modelo relacional, quanto o modelo em grafos. Tal tarefa visou comparar o processo de modelagem, bem como, buscou-se corroborar com as características apresentadas em Vicknair et al. (2010). Os problemas modelados foram encontrados e adaptados da literatura, embasados nos casos de uso apresentados em Fowler e Sadalage (2012).

As três próximas seções apresentam cada um dos problemas selecionados: Redes Sociais, Roteamento de Mercadorias e Sistemas de Recomendação.

3.1 Problema 1: Redes Sociais

A formulação de uma rede é algo peculiar do ser humano, que cria vínculo com indivíduos semelhantes e que possuem interesses em comum como: trabalho, *hobbies*, entre outros. Estas relações vão sendo expandidas pela pessoa que vai tomando decisões e rumos na rede, (TOMAEL; ALCARA; CHIARA, 2004).

Pode-se especificar o problema de uma rede social do seguinte modo:

- A rede social deve armazenar de usuários, suas amizades, suas postagens e as reações a estas postagens. Para cada usuário armazena-se um identificador, o nome completo, um e-mail e a sua data de nascimento.
- Usuários podem fazer nenhuma ou várias postagens. Cada postagem tem obrigatoriamente um texto e deve ter sido feito exatamente por um usuário. Os usuários podem ter apenas uma reação a cada postagem. Essa reação pode ser uma expressão ou um comentário.
- É possível ainda para cada usuário cadastrar um local de estudo, um local de trabalho e um endereço, mas o registro dessas informações não é obrigatório. Além disso, usuários podem indicar amizades com outros usuários.

Considerando esses requisitos, foi gerado o diagrama ER, apresentado na Figura 14. Para a realização da modelagem ER de todos os problemas foi utilizada a ferramenta educacional brModelo[1]. Cabe destacar a modelagem de *POSTAGEM* como uma entidade

[1] http://www.sis4.com/brModelo/

fraca de *USUARIO*, uma vez que não pode existir uma postagem sem usuário, e o auto relacionamento *Amigo de* para representar as amizades de cada usuário.

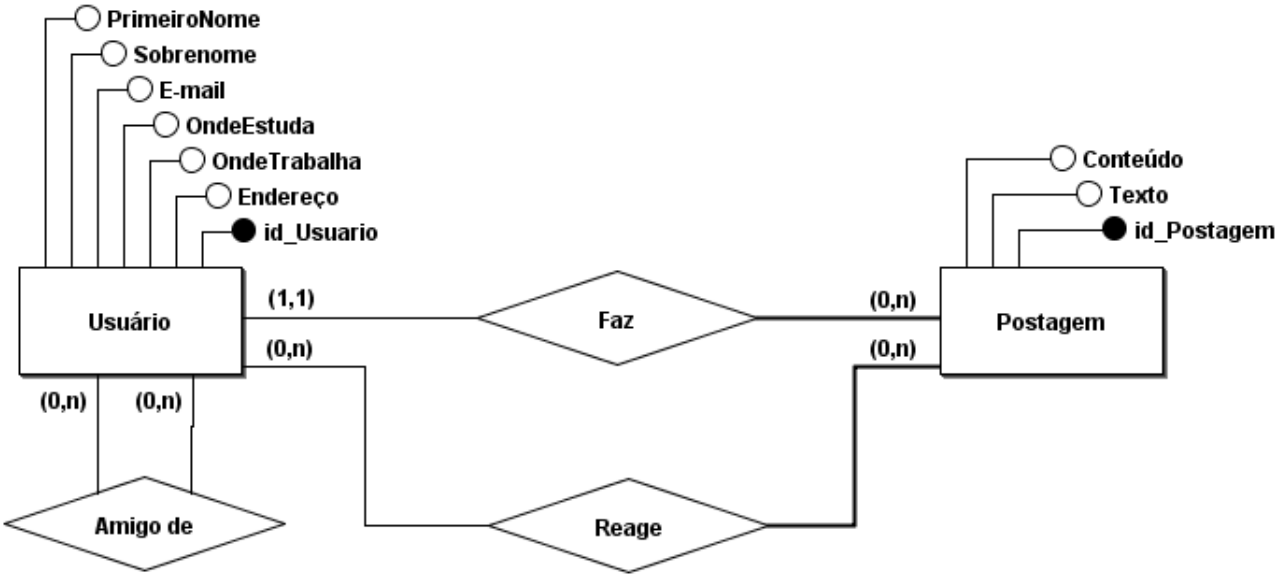


Figura 14 – Diagrama ER de uma rede social.

Fonte: Próprio autor.

O mapeamento do diagrama ER para o modelo relacional é mostrado na Figura 15. Para realização do mapeamento foram seguida as regras vistas em (HEUSER, 2008; ELMASRI; NAVATHE, 2010; DATE, 2004; SILBERSCHATZ; KORTH; SUDARSHAN, 2012). Foi utilizada a ferramenta MySQL Workbench[2] para construção dos diagramas relacionais.

[2] https://www.mysql.com/products/workbench/

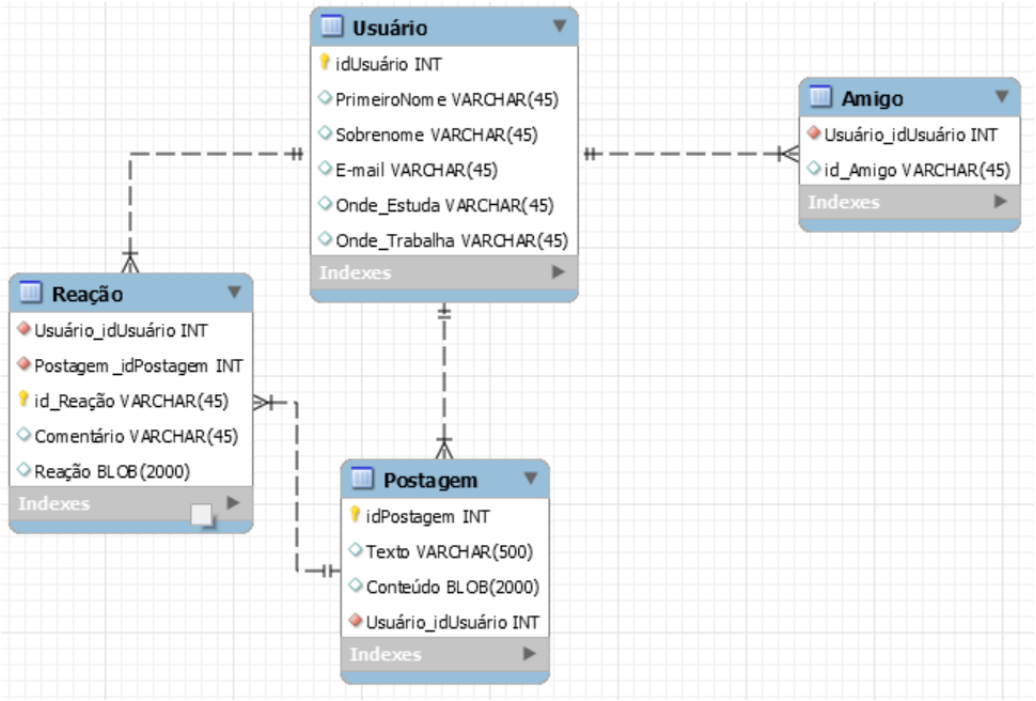


Figura 15 – Modelo lógico relacional de uma rede social similar ao Facebook.

Fonte: Próprio autor.

É possível observar que o auto relacionamento existente na entidade *USUARIO* é mapeada para uma tabela que armazena as amizades. Isso se traduz na replicação de dados, o que pode vir a tornar-se um problema sério, visto que o volume de dados duplicados aumentaria quase que proporcionalmente a quantidade de usuários

Outras abordagens para implementar esse relacionamento de amizade seria utilizar uma tabela de amigos para cada usuário que deveria ser implementada na aplicação, ou a utilização de uma tabela com todos os relacionamentos de amizade. A segunda alternativa seria inviável, o custo computacional para realização de consultas sobre ela e pelo volume de dados. Tais problemas também apareceriam nas tabelas de *POSTAGEM* e *REAÇÃO*.

Um exemplo de modelagem de redes sociais usando grafos pode ser vista em (RECUERO, 2009). Nesse trabalho, os dois elementos principais são o *ATOR/USUÁRIO* que podem ser representados como um vértice, e é responsável pelas interações e que vão constituir as ligações sociais; e as *INTERAÇÕES* que definem as conexões entre os entre os usuários. Outros conceitos do problema que podem ser modelados como vértices são as postagens.

Usando o processo de *Whiteboarding* o diagrama da Figura 16 foi gerado. Em um primeiro momento, cada instância seria mapeada para um vértice próprio. No caso, visualizou-se 4 usuários e 1 postagem, também representada por um vértice. As interações entre os vértices representam as amizades entre usuários, a criação da postagem e a reação as postagens, respectivamente.

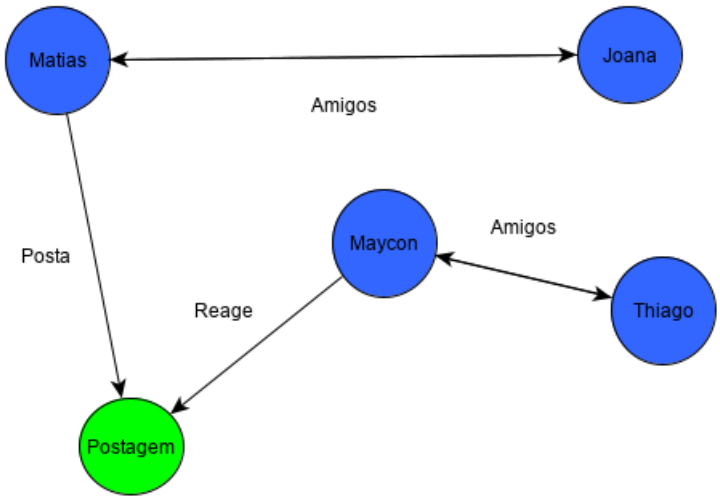


Figura 16 – Processo de Whiteboarding para uma rede social.

Fonte: Próprio Autor.

A Figura 17 apresenta a próxima etapa do processo de *Whiteboarding*. Essa etapa consiste na utilização de rótulos dos vértices.

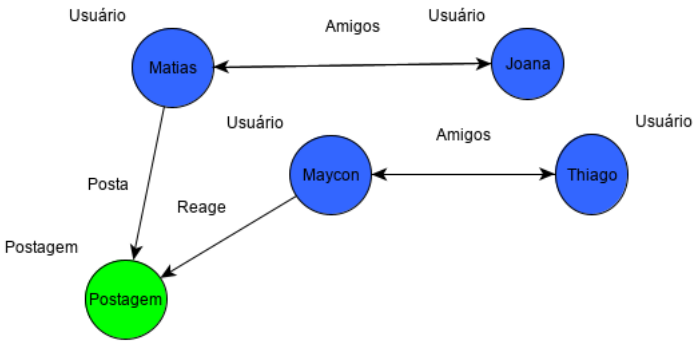


Figura 17 – Processo de Whiteboarding, com rótulos, para uma rede social.

Fonte: Próprio Autor.

O relacionamento de amizade modelado por arestas evita a replicação de dados e agiliza consultas do tipo "Quem são os amigos de Joana?".

Por fim, a Figura 18 mostra uma etapa não prevista no processo original de *Whiteboarding*. Este trabalho propõe esta etapa como forma de respeitar a estrutura do problema, ainda que permita e flexibilização do MoG.

Essa etapa consiste no agrupamento de vértices de mesmo rótulo, eliminando do projeto conceitual de banco de dados em grafos as instâncias individuais do problema. É possível observar que o diagrama mostrado é um grafo muito mais simples que os mostrados nas Figuras 16 e 17;

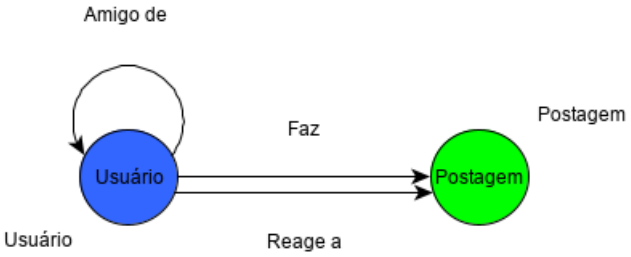


Figura 18 – Modelo Lógico do Problema de Rede Social no Modelo Orientado a Grafos.

Fonte: Próprio Autor.

3.2 Problema 2: Roteamento de Mercadorias

Muitas empresas dos mais diversos setores necessitam determinar o melhor roteiro para a entrega de suas mercadorias tanto em pontos de revenda como na residência dos clientes. Este problema é conhecido como o problema do Caixeiro Viajante (SZWARCFITER, 1986).

O problema pode ser especificado, de modo adaptado, da seguinte forma (HEUSER, 2008):

- O sistema deve armazenar com relação aos produtos, seu lote, seu fabricante, bem como o fornecedor de cada lote. Para cada produto em si, armazena-se um identificador, um código, um identificador de lote e um identificador. Cada produto é fabricado por um único fabricante, que pode fabricar vários produtos. Um produto pertence a um único lote, que pode ter vários produtos;
- Fornecedores e fabricantes devem ter seu identificador, nome, telefone, CNPJ e um contato armazenados. Cada lote guarda um identificador, a data do lote e seu número;
- Um fornecedor fornecer nenhum ou vários lotes e cada lote é fornecido obrigatoriamente por apenas um único fornecedor;
- Um fabricante pode ter seus produtos comercializados por nenhum ou vários fornecedores, já um fornecedor deve comercializar os produtos de no mínimo um até diversos fabricantes distintos;
- Um produto pode ser registrado em nenhuma ou várias vendas, já em cada venda é registrado no mínimo um produto. Para cada venda armazena-se um identificador, o identificador do cliente para qual ela foi realizada e um código;

- Pode ser realizadas várias vendas para um cliente e uma venda é feita a um único cliente. Clientes devem ter seu identificador, seu nome, CPF e endereço persistidos. Cada cliente mora em um único endereço e em cada endereço podem morar desde um único até vários clientes;
- Para cada endereço deve-se guardar um identificador e uma descrição de endereço.

Considerando os requisitos supracitados foi gerado o diagrama ER apresentado na Figura 19. Cabe destacar que como o problema visa calcular a melhor rota, baseado na proximidade entre endereços, eles serão modelados como um entidade própria possuindo um identificador e uma descrição.

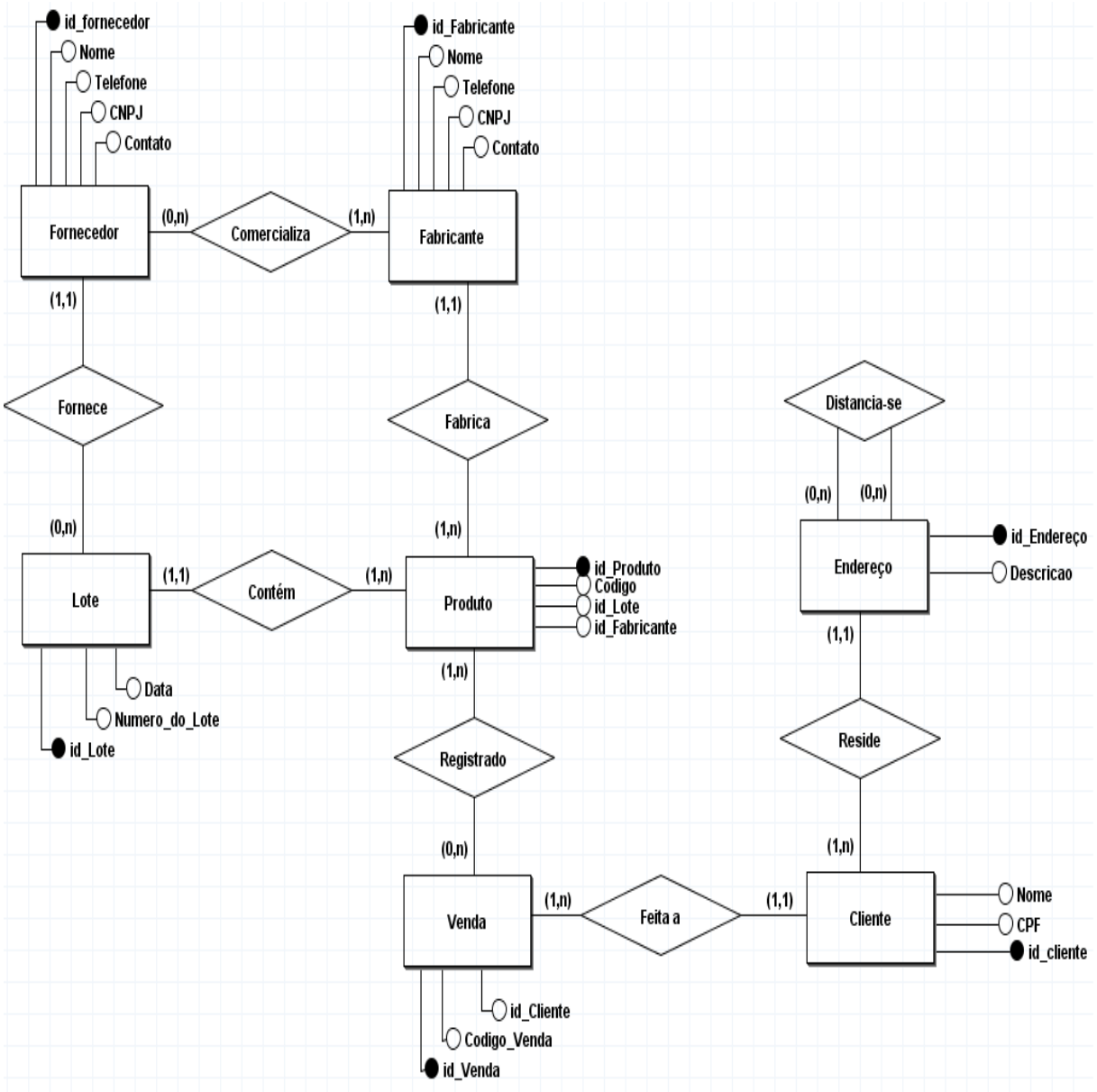


Figura 19 – Diagrama ER do problema de roteamento de produtos.

Fonte: Próprio Autor.

Na Figura 20 pode ser visto o mapeamento do diagrama ER para o modelo relacional.

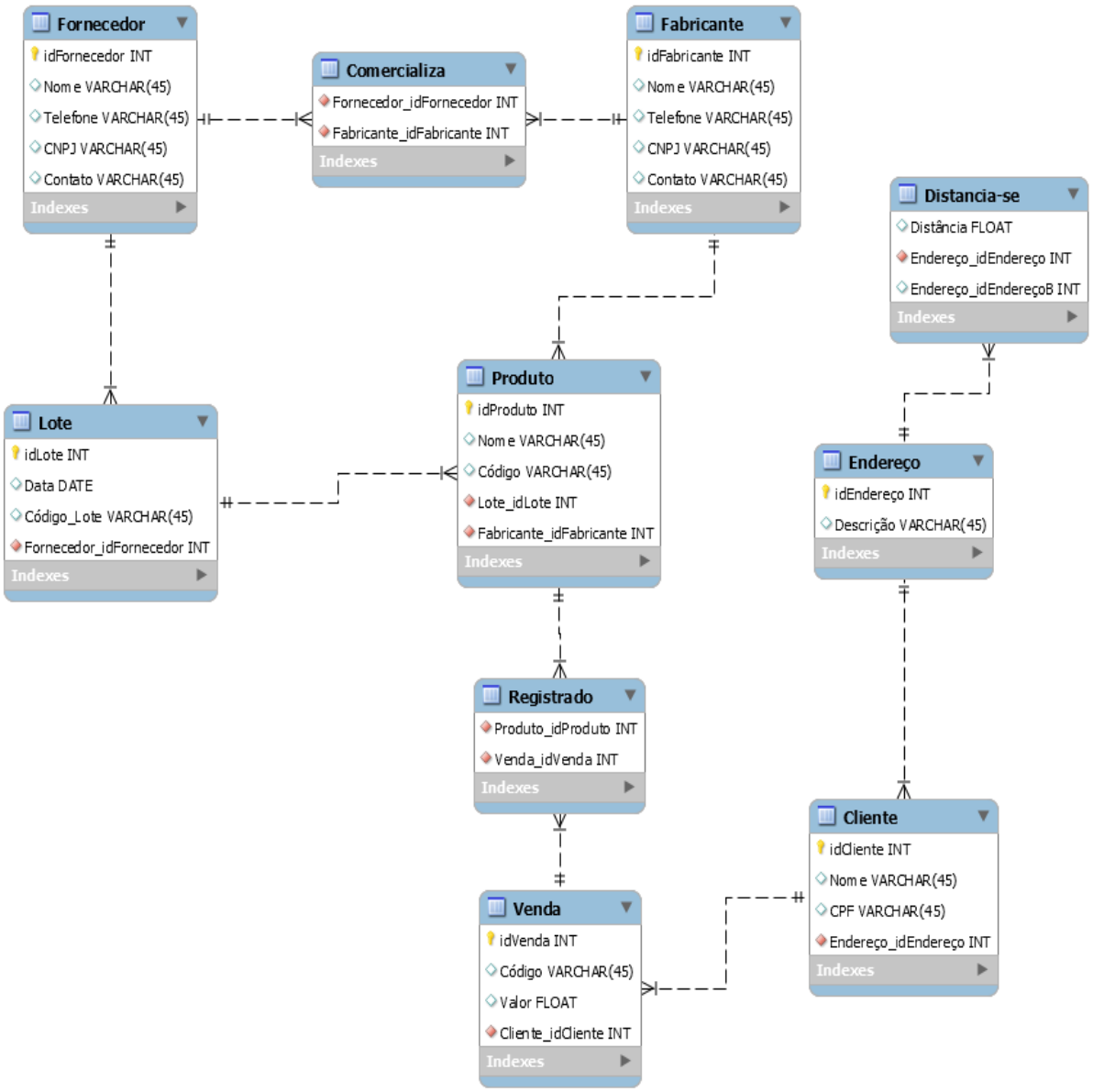


Figura 20 – Diagrama relacional do problema de roteamento de produtos.

Fonte: Próprio Autor.

Cada uma das entidades: *PRODUTOS*; *FORNECEDORES*; *LOTES*; *FABRICANTES*; *CLIENTES*; e *VENDAS* são mapeados como tabelas próprias, contendo seus respectivos atributos. Os relacionamentos *n:m* também geraram suas próprias tabelas.

A questão começa a tomar um rumo diferente quando se observa a parte estendida do problema: os *ENDEREÇOS*, necessários para o cálculo da melhor rota. Na modelagem ER, foi necessário um auto relacionamento na própria entidade *ENDEREÇO*, isso traduziu-se em uma tabela no modelo relacional. Nesta tabela são armazenados os identificadores

dos *ENDEREÇOS* de destino e origem, e a distância entre eles. Os cálculos de melhor rota seriam realizados pela aplicação.

Alguns trabalhos, já citam o problema de roteamento como pertencente a teoria dos grafos (HARRIS; HIRST; MOSSINGHOFF, 2008; SZWARCFITER, 1986). Além disso, a abordagem relacional incorreria em duas outras dificuldades. A primeira seria a replicação de dados em disco, visto que para cada par de endereços replicam-se os identificadores dos mesmos na tabela de distância. A segunda dificuldade teria relações com o desempenho de consultas sobre essa tabela. Os atributos opcionais também iriam onerar o armazenamento.

Com o processo de *whiteboarding* o diagrama da Figura 21 foi gerado. Cada instância foi mapeada para um vértice próprio. Neste caso, visualizou-se 2 *Clientes*, 2 *Endereços*, 2 *Produtos*, 1 *Lote*, 1 *Venda*, 1 *Fabricante* e 1 *Fornecedor*. Cada instância de entidade é representada por um vértices e as arestas entre os vértices representam os diferentes tipos de relacionamentos entres as entidades como por exemplo um *CLIENTE* reside em *ENDEREÇO*, um *FABRICANTE* fabrica *PRODUTO*.

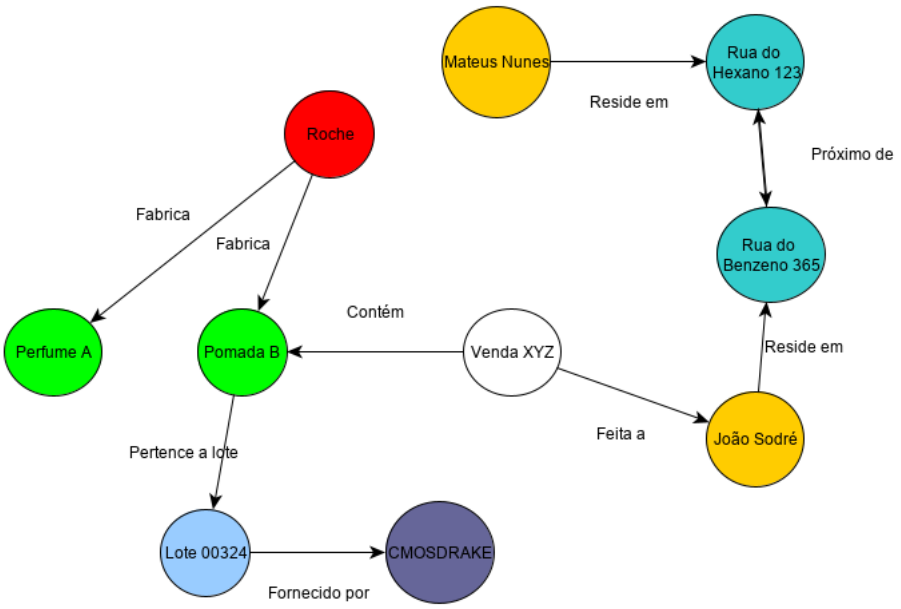


Figura 21 – Processo de Whiteboarding para o problema de roteamento de produtos.

Fonte: Próprio Autor.

O próximo passo do processo seria a rotulação dos vértices como demonstrado na Figura 22:

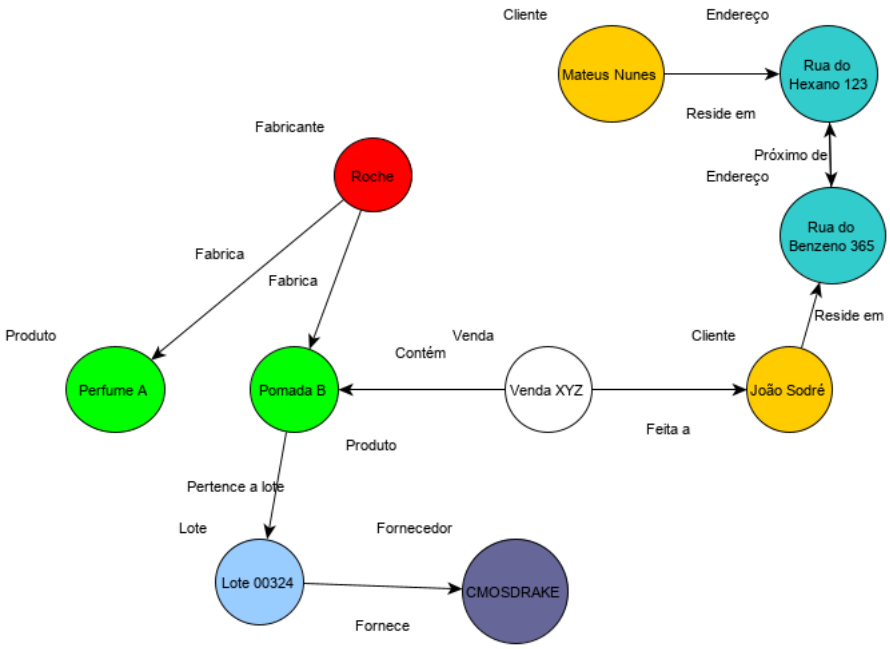


Figura 22 – Processo de Whiteboarding, com rótulos, para o problema de roteamento de produtos.

Fonte: Próprio Autor.

Considerando a etapa extra proposta, obtem-se o grafo da Figura 23.

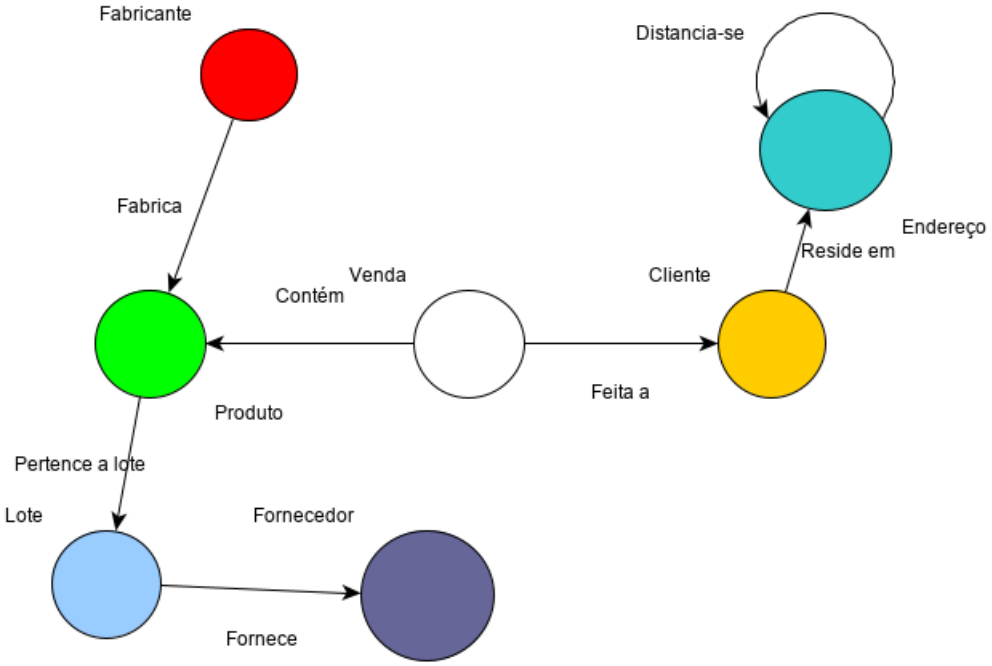


Figura 23 – Modelo Lógico do Problema de Roteamento de Produtos no MoG.

Fonte: Próprio Autor.

3.3 Problema 3: Sistemas de Recomendação

Sistemas de recomendação usam a informação de consumo dos próprios clientes para divulgar anúncios de produtos similares que possam lhes interessar (LEVITT, 2006). O terceiro problema modelado foi embasado no trabalho de Huang et al. (2002) e pode ser especificado da seguinte forma:

- O sistema deve armazenar as informações de usuários e livros. Para cada usuário deve ser armazenado seu identificador, nome, idade, e-mail, endereço e opcionalmente tópicos de interesse;
- Para um livro deve ser guardada o identificador, o título, a editora, o volume, o autor e a série, sendo a série uma característica não obrigatória;
- Usuários compram ou não livros, que podem ser comprados por nenhum ou vários usuários;
- Usuários podem ser similares demograficamente com ninguém ou múltiplos usuários;
- Livros podem ter um conteúdo similar a nenhum ou vários outros livros.

Considerando os requisitos previamente apontados foi gerado o diagrama ER apresentado na Figura 24. É pertinente ressaltar o auto relacionamento das duas entidades modeladas.

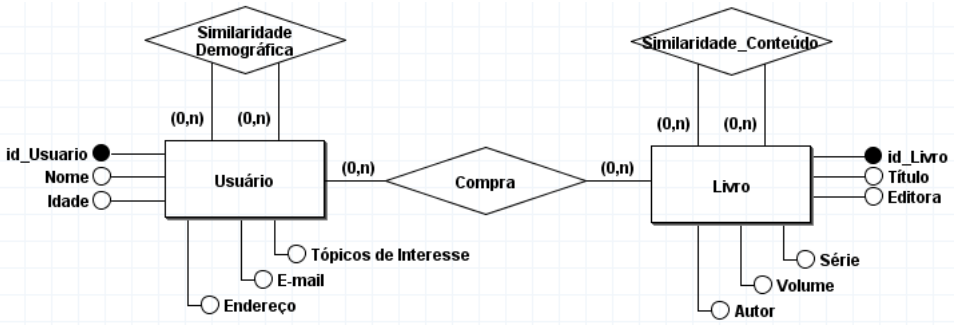


Figura 24 – Diagrama ER para o problema de sistemas de recomendação.

Fonte: Próprio Autor.

O mapeamento do diagrama ER para o modelo relacional é mostrado na Figura 25.

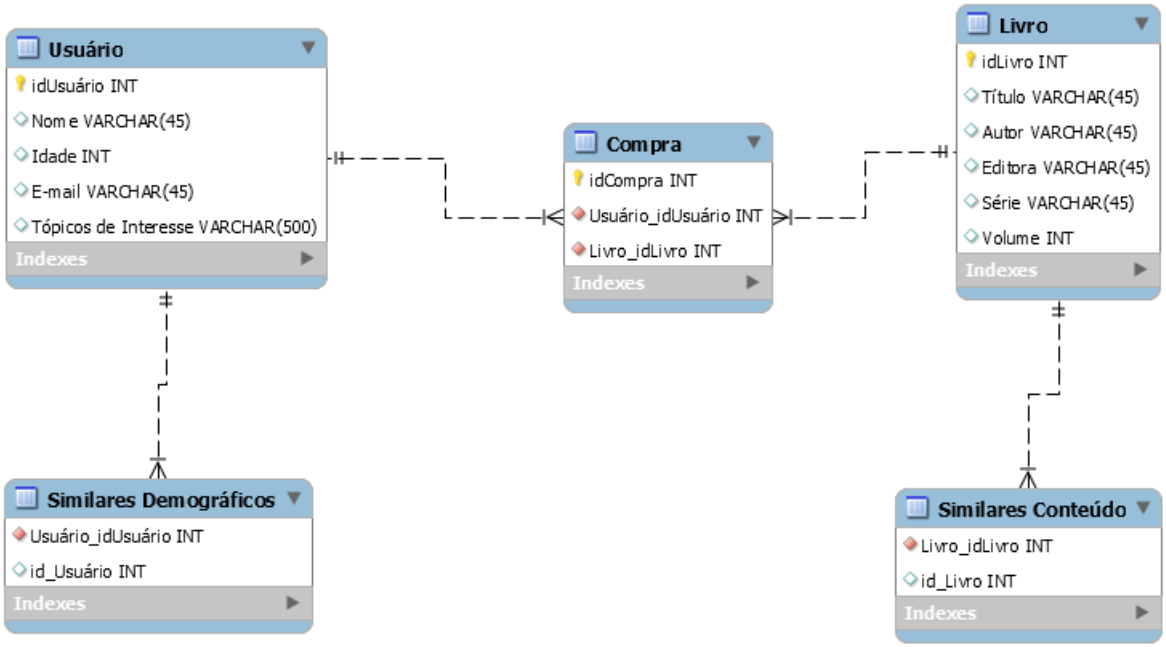


Figura 25 – Diagrama lógico para o problema de sistemas de recomendação.

Fonte: Próprio Autor.

As entidades *Usuário* e *Livro* são mapeadas em tabelas próprias que armazenam seus respectivos atributos. Além disso, a tabela de *Compra* armazena seu próprio identificador, o identificador do *Usuário* que realizou a compra e o identificador do *Livro* comprado. No modelo relacional ainda são necessárias duas tabelas: uma para similaridade demográfica entre os *Usuários* e outra para similaridade de conteúdo entre *Livros*.

Essa forma de implementação tende a trazer alguns problemas. Um deles é causado pelos atributos opcionais de *Usuário* e *Livro*, que caso sejam pouco utilizados, ocasionaria muitos valores do tipo *NULL*. Outro problema é que uma junção entre essas duas tabelas poderia implicar em um estouro de memória. Além disso, cada uma das tabelas geradas a partir dos relacionamentos *n:m* ocuparia espaço em disco, com dados redundantes.

Usando o processo de *whiteboarding* o diagrama da Figura 26 foi gerado. Num primeiro momento, cada instância seria mapeada para um vértice próprio, sendo 2 *Livros* e 3 *Clientes*, apenas para iniciar.

Cada instância de entidade é representada por um vértice e as arestas entre os vértices representam os relacionamentos. Na figura, um *Marcos* comprou o livro *Jogo dos Tronos* e o livro *A Volta dos Que Não Foram* tem um conteúdo similar ao livro *As Tranças do Careca*.

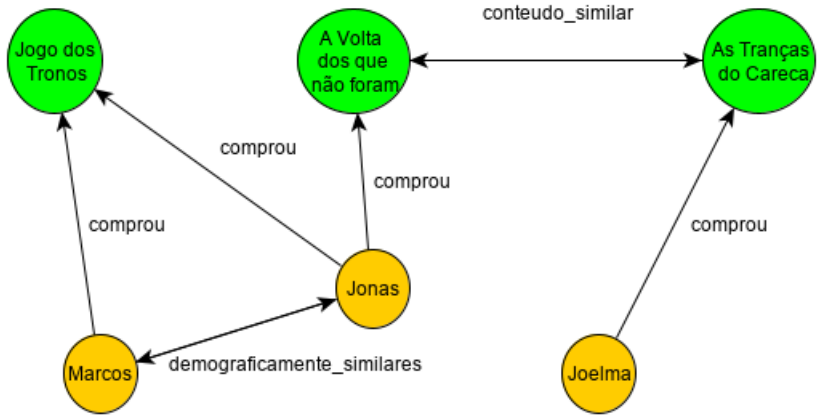


Figura 26 – Processo de Whiteboarding para o problema de sistemas de recomendação.

Fonte: Próprio Autor.

A Figura 27 apresenta a etapa de rotulação dos vértices.

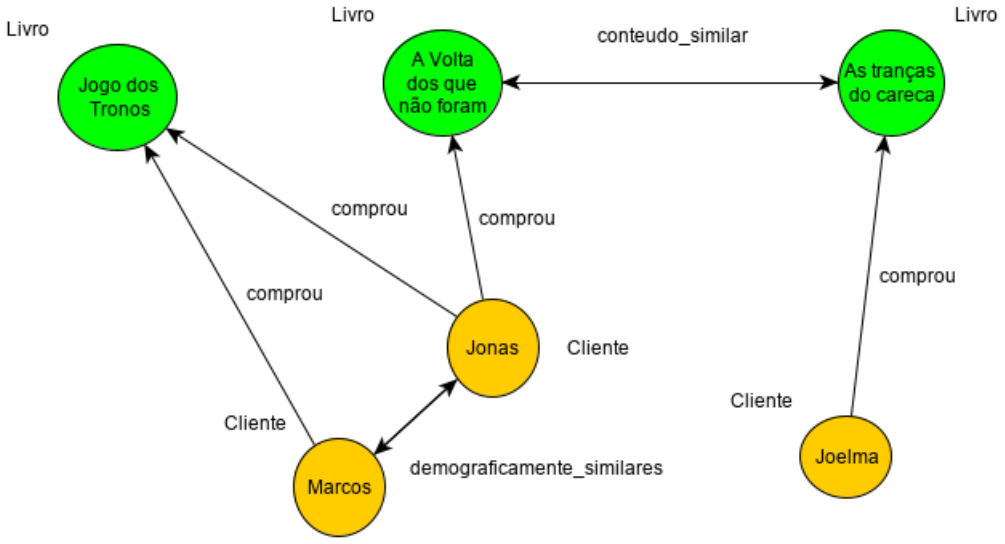


Figura 27 – Processo de Whiteboarding, com rótulos, para o problema de sistemas de recomendação.

Fonte: Próprio Autor.

Por fim, a Figura 28 mostra o agrupamento de vértices de mesmo rótulo, visando a substituição das instâncias individuais do projeto conceitual de banco de dados em grafos pelos vértices tipificados.

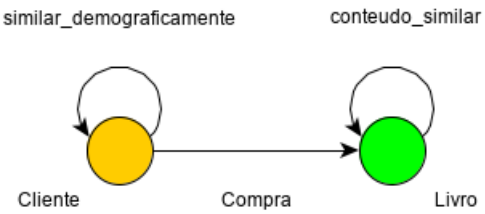


Figura 28 – Agrupamento de vértices de mesmo rótulo para o problema de sistemas de recomendação.

Fonte: Próprio Autor.

Os três problemas escolhidos são citados na literatura como de domínios onde o banco de dados orientado a grafos são uma escolha melhor que o banco de dados relacional. Com a modelagem ER desses casos de uso, foram verificadas 3 (três) características comuns presentes nos diagramas: (1) Existência de auto relacionamentos em alguma das entidades; (2) Quantidade significativa de relacionamentos *n:m*; (3) Existência de atributos opcionais nas entidades.

Além de elucidar essas características, um dos outros objetivos deste trabalho é propor um processo de modelagem dentro do MoG, mais estruturada que o *whiteboarding*, que é a metodologia atualmente utilizada. Apesar de existirem trabalhos desenvolvidos propondo metodologias, técnicas e processos de modelagem, os mesmos tendem a abordar modelos NoSQL de forma mais geral ou outros modelos que não modelo orientado a grafos.

A ideia inicial foi mapear o modelo NoAM para o modelo em grafos, durante o processo, o autor percebeu a falta de algumas características no mesmo, que são indispensáveis de serem consideradas ao modelar um banco de dados em grafos. Na tentativa de mitigar esse problema é proposta uma extensão do modelo NoAM além do seu mapeamento para o MoG. Naturalmente o resultado será discutido detalhadamente na seção do capítulo de resultados junto a proposta de características no modelo ER que podem indicar que o MoG seja mais apropriado para um problema em questão.

O Capítulo 4 apresenta a proposta de modelagem com mais detalhes, bem como explicita as características levantadas com a modelagem dos problemas.

4 Contribuições

Neste capítulo são apresentadas as características levantadas, após a modelagem dos casos de uso do Capítulo 3. Tais características indicam que a utilização do modelo orientado a grafos pode ser a mais apropriada. A estruturação do processo de modelagem de BDs do MoG também é retratada neste capítulo. Por fim, é realizada a modelagem de um único problema em todos os modelos de BDs a fim de comparação.

4.1 Proposta de Processo de Modelagem para o MoG

Uma das contribuições deste trabalho é a proposição de uma estruturação para a modelagem dos bancos de dados em grafos, visto que o *Whiteboarding*, sacrifica essa estruturação para prover a liberdade na modelagem dos dados.

Apesar da flexibilidade ser uma qualidade importante na modelagem de dados, e existirem problemas que demandem pouca estruturação nos dados, são comuns problemas apresentarem uma estrutura por si só, mesmo que mínima.

O processo de modelagem proposto engloba a utilização do modelo NoAM adaptado, na fase conceitual. E para realização do projeto lógico, é sugerido o uso do mapeamento para o MoG, com base no trabalho de Pokorny (2016).

No modelo orientado a grafos, os dados são estruturados em vértices e arestas. Logo, é interessante buscar o particionamento do grafo principal em subgrafos, utilizando-se da semelhanças existente entre vértices. Com isso em mente, o modelo NoAM se torna um candidato de ser usado na etapa de projeto conceitual.

O modelo NoAM utiliza o conceito de blocos para realizar esse agrupamento, como visto na seção 2.3.2. Essa característica do modelo NoAM tende a direcionar o projetista ao agrupamento, ou no caso a agregação dos dados.

Durante a modelagem conceitual é deve-se considerar as seguintes recomendações: (1) Representar por uma única entrada os agrupamentos pequenos de dados que são acessados/modificados juntos; e (2) Dividir os agrupamentos, caso o agrupamento seja grande e tenha apenas partes acessadas/modificadas com frequência. Além disso, dois ou mais elementos devem pertencer a mesma entrada se são frequentemente acessados/modificados juntos e devem pertencer a entradas distintas caso contrário.

Deste modo, a primeira etapa do processo é a mesma do modelo NoAM. Encontrar os conceitos do problema e agrupá-los em blocos. A partir do diagrama de blocos do modelo NoAM pronto, é necessário iniciar o mapeamento para o MoG.

A primeira unidade do NoAM a ser mapeada para o modelo orientado a grafos é o bloco. Um bloco contém componentes que caracterizam a entidade que o bloco representa. A Figura 29 mostra um bloco identificado por *João*. Os retângulos (blocos simples) dentro do bloco trazem informações sobre essa entidade em particular, como por exemplo seu *Nome* e *Endereço*.

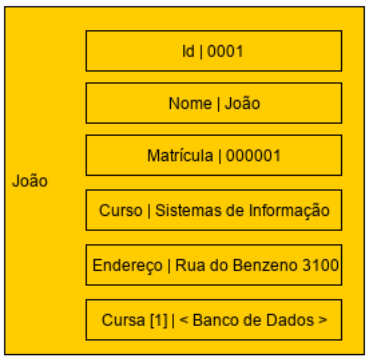


Figura 29 – Bloco *ESTUDANTE*: João

Fonte: Próprio autor.

O nome *João*, dado ao bloco, equivale a um identificador do bloco, e as informações dentro dos retângulos correspondem as características dessa entidade. Com os conceitos do MoG, o bloco apresenta as características de um vértice, com identificador igual a *João* e com as características do bloco sendo os atributos do vértice. A Figura 30 mostra o resultado desse mapeamento.

Figura 30 – Mapeamento do bloco NoAM em um vértice do MoG.

Fonte: Próprio autor.

Observa-se que o atributo *Cursa* do bloco *João* não foi mapeado. Pela notação usada pelo NoAM, esse atributo representa uma associação entre blocos. Deste modo, o valor do atributo é mapeado para um novo vértice, *Banco de Dados*, e o nome do atributo, *Cursa*, é mapeado como uma aresta rotulada, que parte do vértice *João* e chega no vértice *Banco de Dados* (Figura 31). Não é possível inferir se há algum relacionamento que parte do vértice *Banco de Dados* com destino ao vértice *João*, visto que isso não é representado no bloco.

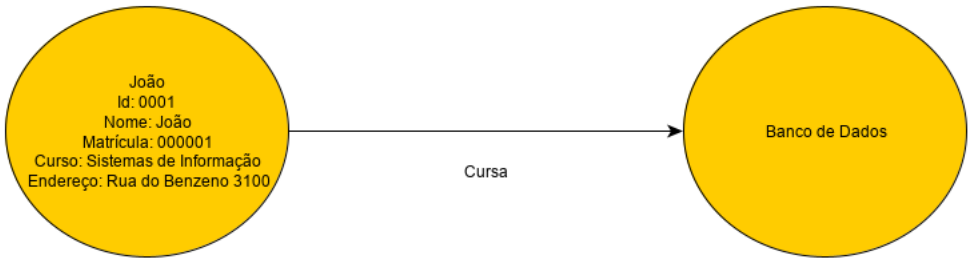


Figura 31 – Mapeamento dos blocos NoAM em vértices do MoG.

Fonte: Próprio autor.

A seção 2.3.2 apresentou outros elementos que precisam ser mapeados para o MoG. Para exemplificar esses elementos, considere a Figura 32. São mostrados dois agrupamentos *ALUNO* e *DISCIPLINA*. Cada agrupamento possui 2 blocos. Os blocos seriam mapeados conforme indicado anteriormente. O detalhe maior ficaria na criação dos vértices *Banco de Dados* e *Computação Gráfica*. Eles seriam criados pela regra de mapeamento do atributo *Cursa*, porém teriam acrescidos atributos como: *Nome*, *Campus*, *Código*, *Duração* e *Sala*, visto que existem características dos blocos correspondentes.

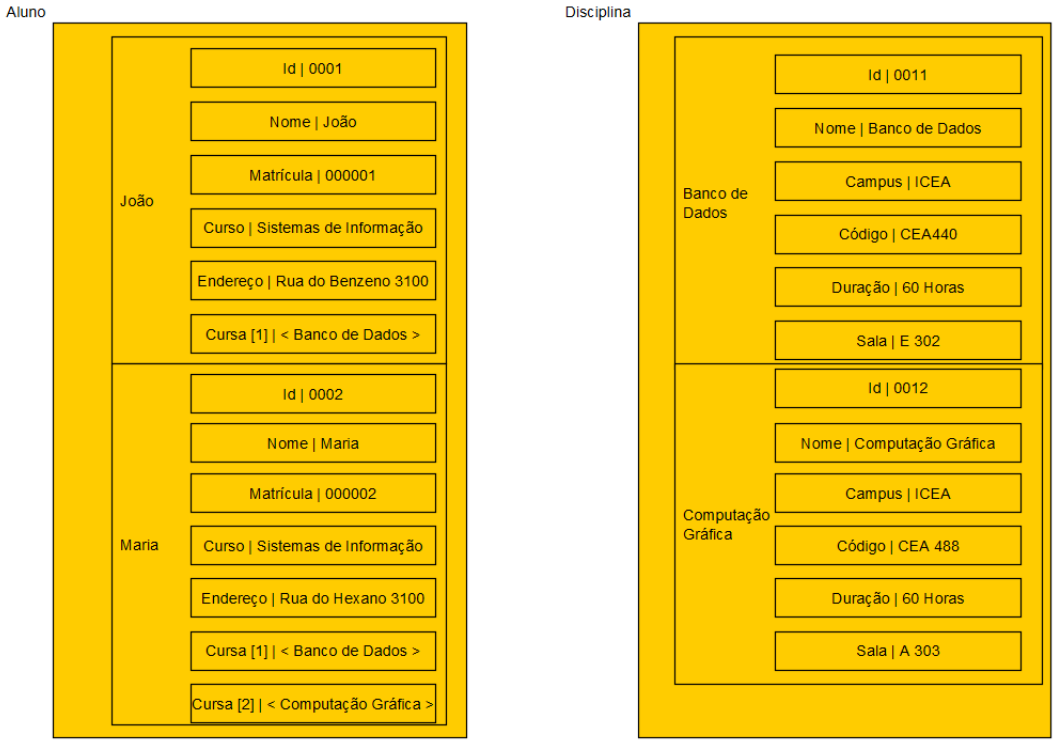


Figura 32 – Agrupamentos *ALUNO* e *DISCIPLINA*.

Fonte: Próprio autor.

A Figura 33 mostra o resultado do mapeamento do diagrama NoAM da Figura 32.

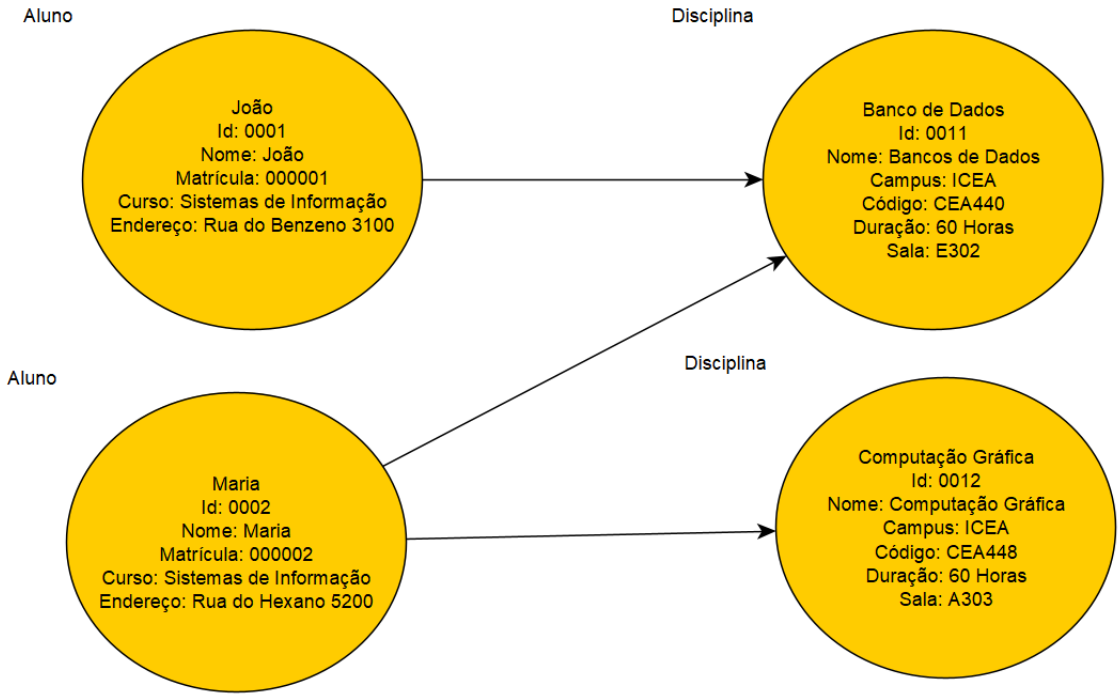


Figura 33 – Mapeamento dos agrupamentos *ALUNO* e *DISCIPLINA* para o MoG.

Fonte: Próprio autor.

No modelo NoAM, blocos individuais são agrupados em blocos maiores. Isso faz com que os blocos mais internos passem a serem considerados um agrupamento/agregado, ou seja, um grupo de blocos que representam entidades similares. Na Figura 32 o primeiro macro-bloco à esquerda, tem um identificador *ALUNO*, que serve como nomenclatura para aquele grupo de blocos. De modo similar, ao lado direito o macro-bloco tem o identificador igual a *DISCIPLINA*.

No mapeamento para o MoG, o identificador *ALUNO*, seria equivalente a um rótulo, isso porque o papel dos rótulos é agrupar vértices em um mesmo tipo. Vale lembrar que a rotulação de vértices auxilia o desempenho do banco de dados posteriormente. BDs em grafos, sem os rótulos, exigiriam que o SGBD percorresse todo o grafo em busca do resultado de todas as consultas executadas sobre ele, o que seria muito ineficiente.

Considerando o modelo NoAM, todos seus elementos foram mapeados para o MoG. Se faz necessário ainda a observância de algumas regras ao modelar um banco de dados em grafos.

A primeira delas é que, no caso de ao modelar conceitos do problema, for verificado que apenas parte dos atributos serão manipulados com frequência, deve-se: modelar os atributos, com mais acessos, como blocos; e os atributos com menor acesso, como blocos auxiliares. Essa modelagem visa o desempenho futuro do banco, dado um vez que o tamanho do vértice implica diretamente no tempo de consulta.

A segunda regra a ser observada é a diferenciação entre rótulos e atributos. Rótulos agrupam vértices com características semelhantes em uma unidade agregada, já os atributos caracterizam os propriamente vértices. A melhor forma de definir essa modelagem é pensando nas consultas que o sistema realizará. Consultas estruturais necessitam de rótulos, já consultas sobre os dados necessitam de atributos. Para exemplificar, no exemplo anterior *Curso* foi modelado como um atributo, porém, se houvesse muitas consultas do tipo "Qual é o curso do aluno X?", seria mais vantajoso modelar *Curso* como um rótulo.

O outro fator a ser levado em conta, quanto a utilização de rótulos, é no excesso de uso. Administradores de banco de dado e projetistas podem ser perder em um emaranhado de rótulos. Além disso, como cada rótulo é implementado como um vetor dinâmico, que armazena os índices dos vértices, pode ocorrer a redundância de dados (GOONETILLEKE et al., 2019).

A terceira regra é manter o tamanho do vértice o menor possível. Isso deve ser feito, seja modelando o vértice com poucos atributos, ou com atributos de tamanho pequeno. Uma boa prática é modelar atributos que são campos maiores como *BLOBs* e/ou *CLOBs* e atributos, cujo tamanho sejam 250 caracteres alfanuméricos, como vértices fracos de seus vértices correspondentes.

Deve-se ainda observar os conceito de agregados atômicos, de caminhos materializados e de conjuntos aninhados. Agregados atômicos, conforme definido por Yoo, Lee e Jeon (2018), é uma técnica de modelagem que consiste em modelar entidades, que normalmente estariam divididas em múltiplas tabelas em um BD relacional normalizado, em um único vértice. Desta forma, as operações sobre os dados ocorrem em um lugar único e de forma atômica. É possível ainda estender esse conceito através da utilização dos rótulos.

Caminhos materializados são uma técnica para ajudar a evitar percorrer de forma recursiva, estruturas similares a árvores (NATH, 2016). A ideia é armazenar identificadores dos pais e/ou filhos de cada nó como atributos dos vértices. Essa técnica pode ser utilizada de forma mais eficiente no MoG, novamente com o uso de rótulos equivalentes.

Por fim, é valido a utilização de conjuntos aninhados, ou seja, armazenar os nós folhas de árvores em um vetor, e mapear cada nó não-folha usando índices de início e fim. No modelo orientado a grafos, isso seria feito utilizando rótulos para definir para onde um vértice não-filho iria "apontar"(NATH, 2016).

4.2 Características de Problemas para Uso de BDs em Grafos

Uma das contribuições deste trabalho, é elucidar características vistas, na etapa conceitual da modelagem dos bancos de dado, que apontem para o uso de BDs em grafos como solução.

Para isso, além de embasamento na modelagem dos três casos de uso, realizados no Capítulo 3, também foi considerada o estado da arte, apresentado no 2.

Dentre as características observadas estão: (1) existência de auto-relacionamentos nos diagramas ER; (2) existência de muitos relacionamentos *n:m*; (3) entidades com atributos parcialmente utilizados ou atributos opcionais entre instâncias de uma mesma entidade; e (4) problemas notoriamente já associados ao domínio da teoria dos grafos.

Apesar de resultarem em diagramas ER relativamente simples, os três casos de uso demonstram uma característica em comum entre eles: os auto-relacionamentos. A ocorrência frequente de auto-relacionamentos, tanto em número de entidades, quanto na quantidade que ocorrem em cada entidade, pode implicar em uma potencial redundância de dados.

Uma quantidade significativa de relacionamentos *n:m* entre entidades também pode ser um indicativo de que é melhor adotar o MoG. Como dito antes, relacionamentos *n:m* geram tabelas no modelo relacional, acarretando no aumento do número de junções e/ou no aumento da redundância de dados.

Atributos opcionais também são outro indicativo. Isso porque cada atributo opcional de uma tupla, terá o valor *NULL*, desperdiçando espaço e dificultando as consultas. No MoG esse problema é resolvido ignorando o atributo no vértice particular. Isso porque, nos bancos de dados em grafos, vértices de mesmo tipo não precisam ter os mesmos atributos (ROBINSON; WEBBER; EIFREM, 2015).

Por fim, a característica mais abstrata entre elas, é a do problema estar naturalmente associado ao domínio da teoria dos grafos. Pode-se sugerir que esses problemas serão beneficiados com o uso do utilização do MoG. Isso porque a documentação sobre o assunto é abundante, o que é de grande valor para os desenvolvedores.

Juntamente a isto, quando da realização do projeto conceitual, com o modelo ER, forem ponderados os tipos de consultas que incidirão sobre os dados modelados. Se o tipo predominante de consulta for o estrutural, também há uma indicação que BDs em grafos podem ser a melhor opção (VICKNAIR et al., 2010).

Nenhuma das características apresentadas neste trabalho inviabiliza o uso do modelo relacional, e sim, meramente apontam para possíveis vantagens em termos de tempos de consulta e utilização de recursos, caso o modelo orientado a grafos seja utilizado.

O modelo relacional, por sua vez, tem vantagem ao modelar entidades que exijam maior detalhamento (mais atributos e/ou atributos de campos grandes). Outro ponto positivo, é que BDs relacionais ocupam, em geral, menos espaço em disco que os BDs em grafos. Isso porque bancos de dados orientados a grafos precisam armazenar as estruturas matriciais, além dos dados dos vértices. Naturalmente que devem haver cuidados para evitar a replicação de dados, caso ela não traga algum tipo de benefício.

Não menos importante, o conceito e a teoria dos bancos de dados relacionais são mais desenvolvidos e consolidados. E, o processo de modelagem é bem definido e estruturado.

4.3 Considerações Finais

Neste capítulo foram apresentadas possíveis características que indiquem que o problema possa ser do domínio de problemas do MoG, bem como a estruturação do processo de modelagem de bancos de dados orientados a grafos e algumas vantagens do modelo relacional. O próximo capítulo apresenta as conclusões do trabalho e sugestões de possíveis trabalhos futuros.

5 Conclusão e trabalhos futuros

5.1 Considerações Finais

Neste trabalho, através da revisão bibliográfica e da modelagem de problemas, foi possível apontar as características que os problemas melhor resolvidos com o MoG possuem.

Também foi proposto o uso do modelo NoAM de modo adaptado para realização de um processo de modelagem mais estruturado para o MoG.

Este estudo indica que não há uma solução ideal para todos os casos e todos os problemas. Independente das características do problema, é a qualidade do projeto do banco de dados que de fato será crucial para se ter os melhores resultados. Consequentemente, a qualidade de um BD é mais dependente das habilidades de seu projetista do que do modelo que ele adota.

5.2 Pontos Fortes do Trabalho

Um dos pontos fortes deste estudo exploratório foi a comparação entre os modelo relacional e o MoG, não em termos de desempenho, e sim com relação ao seu processo de modelagem.

Poucos trabalhos podem ser encontrados sobre este assunto, e nenhum deles dá ênfase na comparação entre os bancos de dados relacionais e os banco s de dados orientados a grafos.

A revisão bibliográfica, não apenas reúne conceitos muito dispersos na literatura, mas também apresenta toda a fundamentação teórica para o cumprimento dos objetivos propostos pelo trabalho.

A modelagem de casos de uso do MoG possibilitou a identificação das características de problemas já citadas e auxiliou a definição e estruturação do processo de modelagem proposto.

5.3 Pontos Fracos do Trabalho

O volume de problemas modelados, 3, foi relativamente baixo. Um tempo maior para realização do trabalho permitiria a realização de mais casos de uso.

O uso do modelo NoAM, já na fase de projeto conceitual ainda não é uma abordagem

muito difundida. Esse fator, somado a falta de padronização do domínio de BDs em grafos, limitou o trabalho em alguns dos aspectos.

5.4 Trabalhos Futuros

Como trabalhos futuros pode vislumbrar a modelagem de um número maior de problemas para verificação das características já encontradas.

Outros trabalhos podem objetivar uma experimentação, similar a de Vicknair et al. (2010), que serviria para corroborar ou não com a noção de que bancos de dados específicos possuem vantagens em tipos particulares de consultas.

Outro trabalho poderia realizar a comparação entre o processo de modelagem proposto e o *Whiteboarding*, na modelagem de um problema em escala comercial. Isso seria de grande contribuição, pois poderia ajudar a consolidar o processo proposto como uma forma mais estruturada de modelagem no MoG.

Referências

ABADI, D. J.; MADDEN, S. R.; HACHEM, N. Column-stores vs. row-stores: How different are they really? In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, 2008. (SIGMOD '08), p. 967–980. ISBN 978-1-60558-102-6. Citado na página 14.

ANGLES, R. A comparison of current graph database models. In: *2012 IEEE 28th International Conference on Data Engineering Workshops*. [S.l.: s.n.], 2012. p. 171–177. Citado na página 28.

ANGLES, R. The property graph database model. In: *Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management*. [S.l.: s.n.], 2018. Citado na página 25.

BATRA, S.; TYAGI, C. Comparative analysis of relational and graph databases. *International Journal of Soft Computing and Engineering (IJSCE)*, v. 2, n. 2, p. 1–4, 2012. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.649.9417&rep=rep1&type=pdf>. Citado 2 vezes nas páginas 14 e 36.

BUGIOTTI, F. et al. Database design for nosql systems. In: *Conceptual Modeling - 33rd International Conference, ER 2014, Atlanta, GA, USA, October 27-29, 2014. Proceedings*. [S.l.: s.n.], 2014. p. 223–231. Citado na página 33.

BUGIOTTI, F.; CABIBBO, L.; TORLONE, P. A. R. Data modeling in the nosql world. In: *Hal Archives Ouvertes, 2016*. [S.l.: s.n.], 2010. p. 967–980. Citado 3 vezes nas páginas 33, 34 e 35.

CHEN, P. P. The entity-relationship model - toward a unified view of data. *ACM Trans. Database Syst.*, v. 1, n. 1, p. 9–36, 1976. Disponível em: <https://doi.org/10.1145/320434.320440>. Citado na página 18.

CIGLAN, M.; AVERBUCH, A.; LADIALAV, H. Benchmarking traversal operations over graph databases. *IEEE 28th International Conference on Data Engineering Workshops*, v. 28, n. 1, p. 1–4, 2012. Disponível em: <http://ikt.ui.sav.sk/archive/vega/AEC0312_SCOPUS_ciglan_gdm12.pdf>. Citado na página 14.

CODD, E. F. A relational model of data for large shared data banks. *Commun. ACM*, ACM, New York, NY, USA, v. 13, n. 6, p. 377–387, jun. 1970. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/362384.362685>. Citado na página 17.

DATE, C. J. *Introdução a Sistemas de Bancos de Dados*. 8. ed. [S.l.]: Campus, 2004. ISBN 8535212736. Citado 4 vezes nas páginas 14, 18, 20 e 39.

EICH, M. H.; MISHRA, P. Join processing in relational databases. *ACM Computmg Surveys*, v. 24, n. 1, p. 1–51, 1992. Disponível em: <https://www.csd.uoc.gr/~hy460/pdf/p63-mishra.pdf>. Citado na página 21.

ELMASRI, R.; NAVATHE, S. B. *Sistemas de Banco de Dados*. 6. ed. [S.l.]: Pearson Brasil, 2010. ISBN 978-8579360855. Citado 4 vezes nas páginas 14, 18, 20 e 39.

ENGELS, G. Elementary actions on an extended entity-relationship database. In: *Proceedings of the 4th International Workshop on Graph-Grammars and Their Application to Computer Science*. [S.l.: s.n.], 1990. p. 344–362. Citado na página 26.

FOWLER, M.; SADALAGE, P. J. *NoSQL Distilled*. [S.l.]: Addison-Wesley, 2012. ISBN 978-0-387-79710-6. Citado 3 vezes nas páginas 29, 32 e 38.

GILBERT, S.; LYNCH, N. A. Perspectives on the cap theorem. *IEEE Computer*, v. 45, n. 2, p. 30–36, 2012. ISSN 0018-9162. Citado na página 30.

GOONETILLEKE, O. et al. On effective and efficient graph edge labeling. *Distributed and Parallel Databases*, v. 37, n. 1, p. 5–38, 2019. Disponível em: <https://doi.org/10.1007/s10619-018-7234-4>. Citado 2 vezes nas páginas 26 e 55.

HARRIS, J. M.; HIRST, J. L.; MOSSINGHOFF, M. J. *Combinatorics and Graph Theory, Second Edition*. [S.l.]: Springer, 2008. (Undergraduate Texts in Mathematics). ISBN 978-0-387-79710-6. Citado 3 vezes nas páginas 23, 24 e 45.

HEUSER, C. A. *Projeto de Banco de Dados*. 6. ed. [S.l.]: Bookman, 2008. ISBN 9788577803828. Citado 7 vezes nas páginas 14, 17, 18, 19, 20, 39 e 42.

HUANG, Z. et al. A graph-based recommender system for digital library. In: *ACM/IEEE Joint Conference on Digital Libraries, JCDL 2002, Portland, Oregon, USA, June 14-18, 2002, Proceedings*. [S.l.: s.n.], 2002. p. 65–73. Citado 2 vezes nas páginas 29 e 47.

JACOBS, A. The pathologies of big data. *Queue*, ACM, New York, NY, USA, v. 7, n. 6, p. 10:10–10:19, jul. 2009. ISSN 1542-7730. Disponível em: <http://doi.acm.org/10.1145/1563821.1563874>. Citado na página 14.

JOUILI, S.; VANSTEENBERGHE, V. An empirical comparison of graph databases. In: *International Conference on Social Computing, SocialCom 2013, SocialCom/PASSAT/BigData/EconCom/BioMedCom 2013, Washington, DC, USA, 8-14 September, 2013*. [S.l.: s.n.], 2013. p. 708–715. Citado na página 15.

LEVITT, N. Das redes sociais à inovação. *Computer*, v. 39, n. 5, p. 13–16, 2006. Citado na página 47.

LOPES, J. M. *Um estudo comparativo entre bancos de dados considerando as abordagens relacional e orientada ao grafo*. 90 p. Monografia (Bacharelado) — Universidade Federal de Santa Catarina, 2014. Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/131868/TCC-Jeiel_final_BU.pdf?sequence=4&isAllowed=y>. Citado na página 14.

MILLER, J. J. Graph database applications and concepts with neo4j. In: *Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA*. [S.l.: s.n.], 2013. v. 2324, n. S 36. Citado na página 14.

NATH, K. G. A. Nosql database: An advanced way to store, analyze and extract results from big data. In: *International Journal of Advance Research in Computer Science and Management Studies*. [S.l.: s.n.], 2016. p. 203–221. Citado na página 55.

NAYAK, A.; PORIYA, A.; POOJARY, D. Type of nosql databases and its comparison with relational databases. In: *International Journal of Applied Information Systems*. [S.l.: s.n.], 2013. p. 16–19. Citado na página 31.

POKORNY, J. Modelling of graph databases. *Journal of Advanced Engineering and Computation (JAEC)*, v. 1, n. 1, p. 4–14, 2016. Disponível em: <http://jaec.vn/index.php/JAEC/article/view/44>. Citado 2 vezes nas páginas 26 e 51.

RECUERO, R. *Redes Sociais na Internet*. [S.l.]: Editora Meridional, 2009. Citado na página 40.

ROBINSON, I.; WEBBER, J.; EIFREM, E. *Graph Databases New Opportunities for Connected Data*. 2. ed. [S.l.]: O'Reilly Media Inc., 2015. Citado 6 vezes nas páginas 26, 27, 28, 29, 30 e 56.

RUIZ, D. S.; MORALES, S. F.; MOLINA, J. G. Inferring versioned schemas from nosql databases and its applications. In: *Conceptual Modeling - 34th International Conference, ER 2015, Stockholm, Sweden, October 19-22, 2015, Proceedings*. [S.l.: s.n.], 2015. p. 467–480. Citado na página 34.

SALGADO, A. C.; FONSECA, F.; TIMES, V. 2016. Disponível em: <http://fabioadev.blogspot.com/2016/03/aula-de-modelo-hierarquico-de-banco-de.html>. Citado na página 23.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. *Sistema de Banco de Dados*. 6. ed. Elsevier, 2012. ISBN 9788535245356. Disponível em: <https://books.google.com.br/books?id=Bn7_GAAACAAJ>. Citado 2 vezes nas páginas 14 e 39.

SULLIVAN, D. *NoSQL for Mere Mortals*. [S.l.]: Addison-Wesley, 2015. (For Mere Mortals). ISBN 978-0-387-79710-6. Citado 2 vezes nas páginas 29 e 31.

SZWARCFITER, J. L. *Grafos e Algoritmos Computacionais*. [S.l.]: Campus, 1986. ISBN 978-0-387-79710-6. Citado 4 vezes nas páginas 24, 25, 42 e 45.

TEOREY, T. J.; YANG, D.; FRY, J. P. A logical design methodology for relational databases using the extended entity-relationship model. *ACM Comput. Surv.*, v. 18, n. 2, p. 197–222, 1986. Disponível em: <https://doi.org/10.1145/7474.7475>. Citado na página 18.

TOMAEL, M. I.; ALCARA, A. R.; CHIARA, I. G. D. Das redes sociais à inovação. *Ci. Inf.*, v. 34, n. 2, p. 93–104, 2004. Citado na página 38.

TSICHRITZIS, F. H. L. D. Hierarchical data-base management: A survey. *ACM Comput. Surv.*, v. 8, n. 1, p. 105–123, 1976. Disponível em: <https://doi.org/10.1145/356662.356667>. Citado na página 22.

VAISH, G. *Professional NoSQL*. [S.l.]: Packt, 2013. ISBN 978-0-387-79710-6. Citado 4 vezes nas páginas 21, 30, 31 e 32.

VICKNAIR, C. et al. A comparison of a graph database and a relational database: a data provenance perspective. In: *Proceedings of the 48th Annual Southeast Regional Conference*. [S.l.: s.n.], 2010. p. 42. Citado 5 vezes nas páginas 29, 36, 38, 56 e 59.

VIRGILIO, R. D.; MACCIONI, A.; TORLONE, R. Converting relational to graph databases. In: *First International Workshop on Graph Data Management Experiences and Systems*. New York, NY, USA: ACM, 2013. (GRADES '13), p. 1:1–1:6. ISBN

978-1-4503-2188-4. Disponível em: <http://doi.acm.org/10.1145/2484425.2484426>. Citado na página 14.

WEBER, S. Nosql databases. In: . [S.l.: s.n.], 2010. p. 967–980. Citado na página 30.

YOO, J.; LEE, K.; JEON, Y. Migration from RDBMS to nosql using column-level denormalization and atomic aggregates. *J. Inf. Sci. Eng.*, v. 34, n. 1, p. 243–259, 2018. Disponível em: <http://jise.iis.sinica.edu.tw/JISESearch/pages/View/PaperView.jsf?keyId=160_2116>. Citado 2 vezes nas páginas 30 e 55.

TERMO DE RESPONSABILIDADE

Eu, **Thales Paim Fachinelli** declaro que o texto do trabalho de conclusão de curso intitulado "***Análise de Projetos de Banco de Dados: Modelo Relacional vs. Modelo em Grafos***" é de minha inteira responsabilidade e que não há utilização de texto, material fotográfico, código fonte de programa ou qualquer outro material pertencente a terceiros sem as devidas referências ou consentimento dos respectivos autores.

João Monlevade, 17 de julho de 2019

Thales Paim Fachinelli

Thales Paim Fachinelli

DECLARAÇÃO DE CONFORMIDADE

Certifico que o(a) aluno(a) **Thales Paim Fachinelli**, autor do trabalho de conclusão de curso intitulado "***Análise de Projetos de Banco de Dados: Modelo Relacional vs. Modelo em Grafos***" efetuou as correções sugeridas pela banca examinadora e que estou de acordo com a versão final do trabalho.

João Monlevade, 25 de julho de 2019.

Bruno Rabello Monteiro