Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

MDG-NoSQL: Modelo de Dados para Bancos NoSQL Baseados em Grafos

Gustavo C. Galvão Van Erven

Dissertação apresentada como requisito parcial
para conclusão do Mestrado Profissional em Computação Aplicada

Orientadora
Prof.ª Dr.ª Maristela Terto de Holanda

Coorientador
Prof. Dr. Rommel Novaes Carvalho

Brasília
2015

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado Profissional em Computação Aplicada

Coordenador: Prof. Dr. Marcelo Ladeira

Banca examinadora composta por:

Prof.ª Dr.ª Maristela Terto de Holanda (Orientadora) — CIC/UnB
Prof.ª Dr.ª Genaína Nunes Rodrigues — CIC/UnB
Prof. Dr. Sérgio Lifschitz — DI/PUC-Rio

CIP — Catalogação Internacional na Publicação

Erven, Gustavo C. Galvão Van.

MDG-NoSQL: Modelo de Dados para Bancos NoSQL Baseados em Grafos / Gustavo C. Galvão Van Erven. Brasília : UnB, 2015.
105 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2015.

1. Banco de Dados, 2. Grafos, 3. Governo Federal, 4. Relacionamento, 5. Controle da Corrupção

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

MDG-NoSQL: Modelo de Dados para Bancos NoSQL Baseados em Grafos

Gustavo Cordeiro Galvão van Erven

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Prof.ª Dr.ª Maristela Terto de Holanda (Orientador)
CIC/UnB

Prof. Dr. Genaina Nunes Rodrigues
CIC/UnB

Prof. Dr. Sérgio Lifschitz
PUC-Rio

Prof. Dr. Marcelo Ladeira
Coordenador do Programa de Pós-graduação em Computação Aplicada

Brasília, 13 de março de 2015

Dedicatória

Aos que cultivam nossa criatividade e tornam sonhos em realidades.

Agradecimentos

Agradeço a minha família, amigos e professores pelo apoio, orientação e paciência, assim como a Controladoria-Geral da União que, por meio da Diretoria de Informações Estratégicas, apoiou e colaborou com o desenvolvimento desse trabalho.

Resumo

Os bancos de dados em grafo vêm se tornando populares juntamente com as demais iniciativas *NoSQL*. Porém, os bancos de dados em grafos não possuem uma notação padrão. Sendo assim, o presente trabalho agrupa diversas notações e propostas de modelagem em grafos, construindo um novo modelo, chamado de Modelo de Dados para Bancos NoSQL Baseados em Grafos (MDG-NoSQL), com recursos para agrupar em uma notação as características dos bancos de dados em grafos. Esse modelo foi validado utilizando a implementação de um banco de dados de vínculos societários de empresas, combinados com os relacionamentos dessas pessoas jurídicas com os processos de compras, também chamadas de licitações, junto ao Governo Federal (Banco de Vínculos Simplificado para Licitações e Sociedades). Esse estudo de caso foi direcionado para auxiliar na detecção de fraudes em processos licitatórios que, além de ser diretamente aplicável a vários órgãos de controle, perícia e inteligência em âmbito nacional, permite extrair fundamentos extensíveis a outros problemas com modelagens de vínculos.

Palavras-chave: Banco de Dados, Grafos, Governo Federal, Relacionamento, Controle da Corrupção

Abstract

Currently, Graph Databases are becoming popular along with other NoSQL initiatives. Thus, researchers and companies have been developing data models to manipulate information as graphs However, there is no standard notation involves several features and structures of the graph databases. This model introduces several notations and concepts for building a new graph data model, called Data Model for NoSQL Graph Databases (GDM-NoSQL). The GDM-NoSQL was verified through a database for investigate relationships between companies and people as well as information about the procurements that these companies participated in with the Federal Government. This database was designed to facilitate the process of searching for frauds and to be used on multiple contexts, i.e. several different national agencies. Finally, the designed model was implemented in both a relational and a graph database in order to validate the hypothesis that writing relationships is simpler and more efficient in graph models than relational databases.

Keywords: Databases, Graphs, Federal Government, Relationship, Corruption Control

Sumário

Lista de Figuras

Capítulo 1

Introdução

Grafos são estruturas que permitem evidenciar relacionamentos entre objetos, sejam eles pessoas, ativos de rede ou mesmo cidades. Eles estão presentes em aplicações como os de mapas em celulares ou para controle de rotas em dispositivos na Internet, mas apenas recentemente os grafos vem ganhando popularidade como uma forma nativa de armazenamento de dados. Essa tecnologia emergente traz oportunidades de buscar soluções para problemas onde a informação já possui uma estrutura típica de grafos, assim como a necessidade de se criar um formalismo para descrever o domínio modelado como já é realizado nos bancos relacionais. Esse capítulo traz uma contextualização sobre os bancos de dados em grafos e quais os objetivos desse trabalho, no âmbito tanto da modelagem de dados, como na avaliação de soluções.

1.1 Contextualização

O aparecimento dos bancos de dados NoSQL (*Not Only SQL*) [31] trouxe alternativas para o gerenciamento dos dados além do tradicional banco de dados baseado no modelo relacional. Dentre estes novos sistemas, tem-se os bancos de dados em grafos, especializados em armazenar e recuperar informações utilizando estruturas de vértices e arestas.

Os bancos de dados baseados em grafos vem despertando interesses tanto em diversas áreas de pesquisa como do mercado, principalmente no uso de aplicações onde enfatizar relacionamentos entre as entidades do modelo de dados é tão importante quanto seus atributos. Pode-se citar como exemplo as interações entre átomos de proteínas, padrões de interconexão entre circuitos elétricos, relacionamentos entre dispositivos de uma rede de computadores [35], assim como também situações envolvendo relacionamentos entre pessoas são sugeridas como boas aplicações para implementação em bancos de grafos [4, 18, 33]. Porém, ainda existe uma lacuna sobre como modelar e implementar domínios

utilizando essas tecnologias, pois esses sistemas ainda não possuem uma maturidade no aspecto de modelagem e implementação como o modelo relacional.

A modelagem é uma etapa fundamental na organização dos dados que devem ser armazenados em um banco de dados. Sua importância é claramente percebida na modelagem de bancos relacionais, se interpondo a qualquer fase de implementação ou carga de sistemas mais robustos. Assim como na modelagem tradicional, as técnicas e ferramentas de modelagem em grafos também precisam evoluir, e são movidas principalmente pela necessidade de avaliar antecipadamente se um modelo representa adequadamente o domínio e as necessidade do negócio.

O Modelo de Dados para Bancos NoSQL Baseados em Grafo (MDG-NoSQL), proposto nesse trabalho, foi elaborado para tentar suprir essa lacuna com uma notação para modelar diagramas mais simples às estruturas de grafos mais complexas. Para validar o modelo, foi construído um banco de vínculos entre entidades do poder executivo envolvidas no processo licitatório que podem ser utilizados em investigações de fraudes [17]. Apesar da maioria dessas informações serem armazenadas em bancos relacionais, estas são analisadas utilizando-se ferramentas de estrutura predominantemente em grafos, o que justifica uma comparação entre a forma de consulta e armazenamento atual com um banco de dados em grafos. Para isso, foram selecionadas algumas questões comuns que utilizam esses dados como matéria prima, traduzidas em consultas que serviram para avaliar os impactos na mudança de tecnologia.

1.2 Objetivos

O objetivo geral dessa pesquisa consiste em preencher a lacuna existente de uma notação formal de modelo de dados que atenda aos bancos de dados NoSQL baseados em grafos, através da construção de uma notação generalizada que consolide diversas características presentes nas atuais iniciativas.

Os objetivos específicos dessa pesquisa:

- Proposta de uma notação para diagramar modelos de dados baseados em grafo;
- Implementação de um banco de vínculos simplificado para licitações e sociedades utilizando o modelo de dados em grafos;
- Comparação entre o modelo de dados em grafos e o relacional.

1.3 Estrutura do Trabalho

Este documento está dividido nos capítulos apresentados a seguir:

- Capítulo 2: Apresenta a teoria utilizada para elaboração do Modelo de Dados para Bancos NoSQL Baseados em Grafos (MDG-NoSQL), assim como algumas considerações e diferenças em comparação com a modelagem relacional;
- Capítulo 3: A notação para o MDG-NoSQL é descrito e exemplos de seu uso em alguns casos mais simples são apresentados;
- Capítulo 4: O banco de vínculos é modelado com a notação do MDG-NoSQL e são realizadas análises e comparações sobre a modelagem;
- Capítulo 5: Apresenta um comparativo de consultas sobre o banco de vínculos entre os modelos relacional e de grafos a partir das implementações em, respectivamente, MySQL e o NoSQL Neo4j.

Capítulo 2

Banco de Dados em Grafos

Este capítulo apresenta uma síntese sobre os bancos de dados em grafos. Inicialmente, a estrutura de grafos é formalmente apresentada. Em seguida, são apresentados alguns aspectos históricos com as principais iniciativas das décadas de 80 e 90. Por fim, a Seção 2.3 descreve a retomada dos bancos de dados em grafos nos dias atuais, integrando um dos grupos dos bancos de dados NoSQL [13].

2.1 Definições de Grafos

Os bancos de dados em grafos utilizam diversos estruturas para organizar seus dados. Esta seção apresenta um resumo das definições utilizadas nesses bancos de dados que também serviram para classificar as implementações e facilitar a construção do modelo.

Um grafo $G(V, E)$ consiste em um conjunto finito não vazio de vértices, ou nós, `V`, e uma relação $E \subseteq V \times V$. Ou seja, dado $V = \{v_1, v_2, ..., v_n\}$, um conjunto de relações possível seria $E = \{(v_1, v_2), (v_1, v_4), (v_2, v_6), ..., (v_{n-1}, v_n)\}$ onde cada subconjunto é chamada de aresta ou arco [8].

O número de vértices de um grafo é chamado de ordem. O tamanho do grafo é o número de arestas. Ou seja, $|V| = ordem$ e $|E| = tamanho$. A quantidade de arestas de um vértice é chamada de grau do vértice [6].

Também é possível representar um grafo de forma gráfica, onde os vértices são pontos e as arestas linhas ligando-os como mostra a Figura 2.1. No exemplo, os vértices representando pessoas distintas são ligadas por arestas que representam os relacionamentos de amizade entre elas. O exemplo ilustra também como os vértices podem ser apresentados com rótulo para dar significado a eles no grafo, assim como às arestas, indicando um tipo de relação ou mesmo o peso de percorrê-la. Para o grafo de amizade, as arestas não fazem distinção no sentido do relacionamento e portando são simétricas (bidirecionais), ou seja $(v_i, v_j) = (v_j, v_i)$.

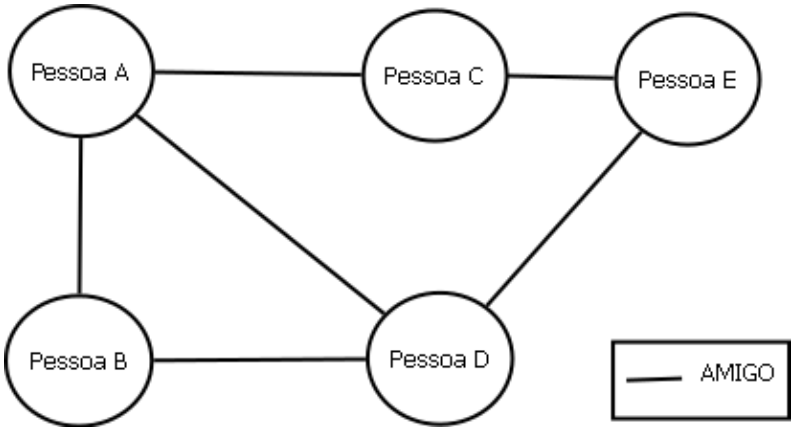


Figura 2.1: Exemplo de grafo.

Quando as arestas fazem distinção entre o sentido da relação, o grafo é chamado de direcionado ou digrafo [8]. Pode-se citar como exemplo a Figura 2.2 onde se indica que a `Pessoa A` está seguindo as publicações da `Pessoa B`, como o que acontece em sítios de redes sociais como Facebook e Twitter.

Isso também implica que para indicar uma relação simétrica, como a de amizade, seriam necessárias duas arestas ligando os vértices dos indivíduos no qual se quer representar a amizade. Poder-se-ia desenhar uma reta com setas em ambos os sentidos para simplificar o desenho, mas deve-se lembrar que ainda são duas arestas, uma para cada par ordenado do conjunto `E`. O grau de cada vértice também pode ser separado em grau de entrada, com arestas que chegam no vértice, e grau de saída, com arestas saindo do nó.

Figura 2.2: Exemplo de grafo direcionado.

Um grafo que permite a existência de várias arestas para um par de vértices é chamado de multigrafo [8]. Pode-se considerar um multigrafo como uma sobreposição de grafos que compartilham os mesmos vértices, mas cada qual com seu conjunto de arestas. Uma aplicação possível de multigrafo seria representar as diversas estradas que ligam um grupo de cidades, como apresentado na Figura 2.3 onde a `Cidade C` pode ser alcançada por dois caminhos a partir da `Cidade A`.

Pode-se generalizar os grafos para permitir que uma mesma aresta relacione mais de dois vértices. Nesse caso, cada arco, chamado de hiperaresta, teria um número variado de nós indicando a mesma relação entre eles, esses grafos são chamados de hipergrafos [3]. Assim sendo, o grafo passa a ser da forma $G(V, H)$, onde V é o conjunto de vértices

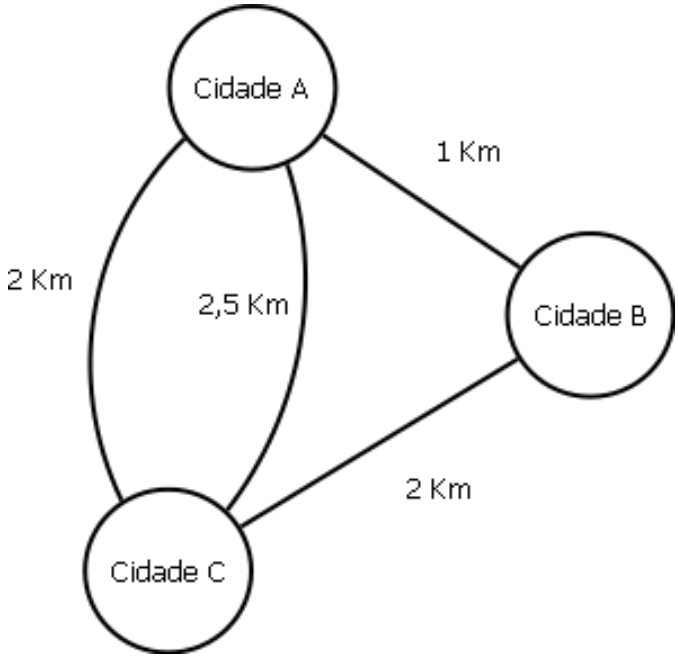


Figura 2.3: Exemplo de multigrafo não direcionado.

$V = \{v_1, v_2, ..., v_n\}$ e H o conjunto de hiperarestas $H = \{E_1, E_2, ..., E_m\}$ onde cada elemento E_m é um conjunto de vértices $E_m = \{v_x, ..., v_z\}$ podendo variar de dois elementos de V até `|V|`. A Figura 2.4 apresenta um exemplo de hipergrafo com duas hiperarestas em uma notação mais comum de ser encontrada para diagramas de instância, embora mais complicada de desenhar. A `Empresa A` participa de duas licitações que são indicadas pela hiperaresta `E1`. A `Empresa B`, participa apenas da `Licitação Y` sendo a hiperaresta que indica esse relacionamento composta apenas de dois elementos de vértices.

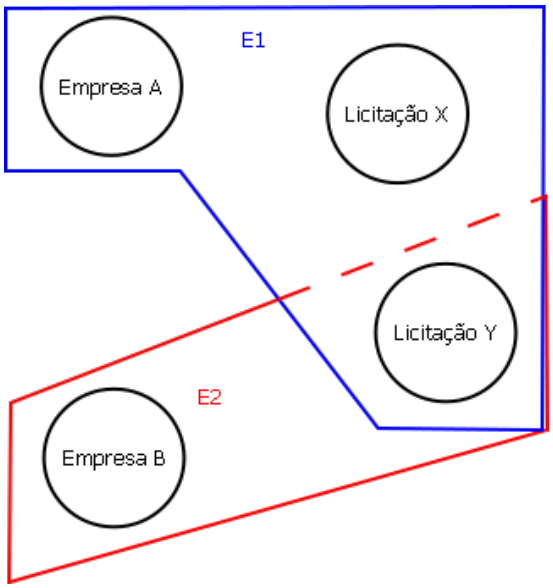


Figura 2.4: Exemplo de hipergrafo com hiperarestas E1 e E2.

Por fim, como foi feito para as arestas, pode-se imaginar um agrupamento de diversos grafos, ou subgrafos, que se relacionam entre si como se fossem vértices. Um grafo agrupado se torna um hipernó, sendo tratado como um vértice comum, e o grafo passa a ser chamado de aninhado [3], ou seja, grafos dentro de grafos. A Figura 2.5 apresenta um exemplo onde as informações de sociedade de uma empresa são agrupadas para formar um hipernó de `Quadro Societário de 2014`.

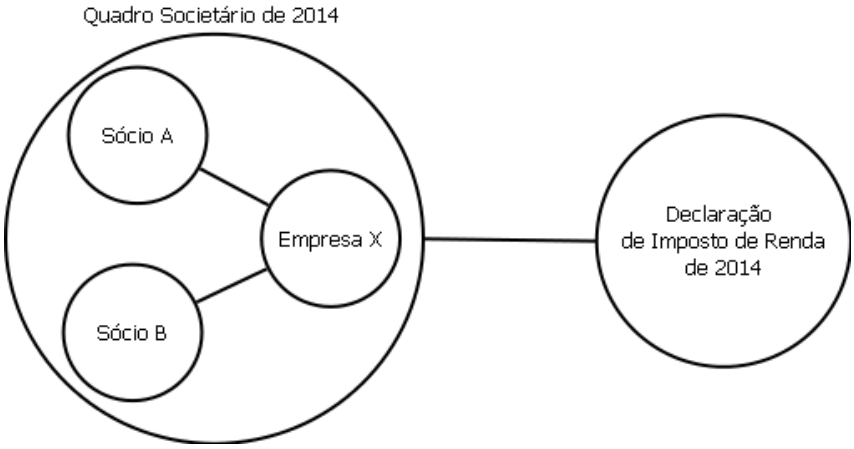


Figura 2.5: Exemplo de hipernó em um grafo aninhado.

O hipernó desse exemplo pode se vincular a outros como um vértice qualquer do grafo aninhado, como apresentado na relação entre o subgrafo com sócios de uma `Empresa` e sua `Declaração de Imposto de Renda`. O hipernó no exemplo indica o quadro societário do ano de competência 2014, ou seja, como uma fotografia de quem atuava na empresa como sócio no ano em questão, e permite que esse subgrafo seja ligado a outro vértice, como a declaração de imposto de renda da empresa para o mesmo período.

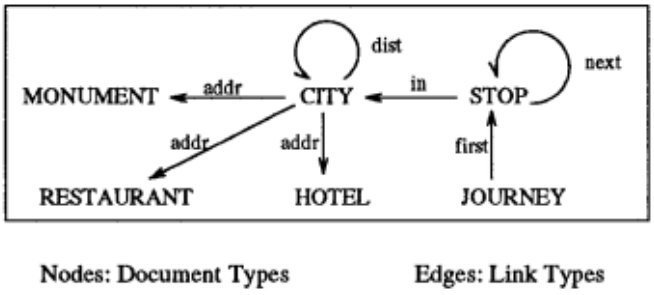
2.2 Evolução dos Bancos de Dados em Grafos

Os bancos de dados baseados em grafos não são tecnologias recentes, vários trabalhos foram realizados ao longo das últimas décadas, embora muito pouco tenha sido produzido durante a primeira década do século XXI [2, 3]. Logo após esse período, com a necessidade de manipular dados de natureza típica de grafos, como as redes sociais, e os bancos NoSQL, para armazenar grandes volumes de dados, ocorreu uma revitalização nessa área, ampliando o interesse e aplicação desses sistemas. Esta seção apresenta um breve evolução dos bancos de dados em grafos, dando ênfase nas características necessárias para esse trabalho.

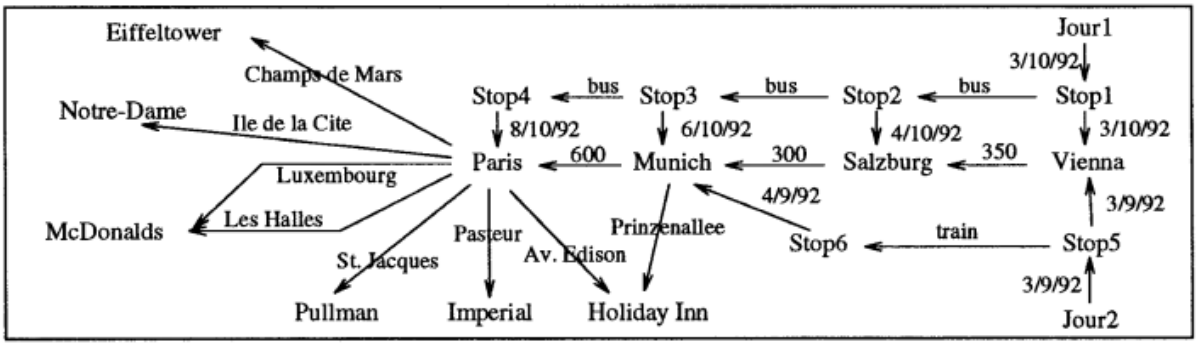
2.2.1 Diversidade de Modelos

Vários trabalhos tratando de modelos de dados utilizando grafos foram desenvolvidos desde de 1975. Entretanto, após o ápice na década de 1990, a produção de pesquisa diminuiu nesse tema até o surgimento dos bancos de dados NoSQL [3]. Essas pesquisas propunham uma forma de organizar a informação, assim como uma notação para o modelo utilizado e, muitas vezes, a forma de consultar como, por exemplo, o GOOD [12], Gram [1], Simatic-XT [27], LDM [22], GOAL [15], Hypernode [24], GROOVY [25] e GDM [16]. Dessas iniciativas, destacam-se para esse trabalho as notações utilizadas nos modelos, das quais serviram como base para a construção do MDG-NoSQL, enfatizando-se o GOAL, Gram, GDM, Simatec-XT e Hypernode.

O modelo da Figura 2.6, o *Gram - Graph Data Model and Query Language* [1], utiliza um grafo para o esquema onde os vértices são rótulos com o tipo de entidade, assim como os relacionamentos, que são representados pelas arestas também rotuladas. Sua estrutura é simples e permite que valores iguais em vértices sejam compartilhados entre as entidades. No exemplo de instâncias, cada vértice recebeu um valor de domínio da entidade, inclusive as arestas, apresentando informações como endereço ou meio de transporte para se chegar aos destinos desejados.



(a) Esquema.

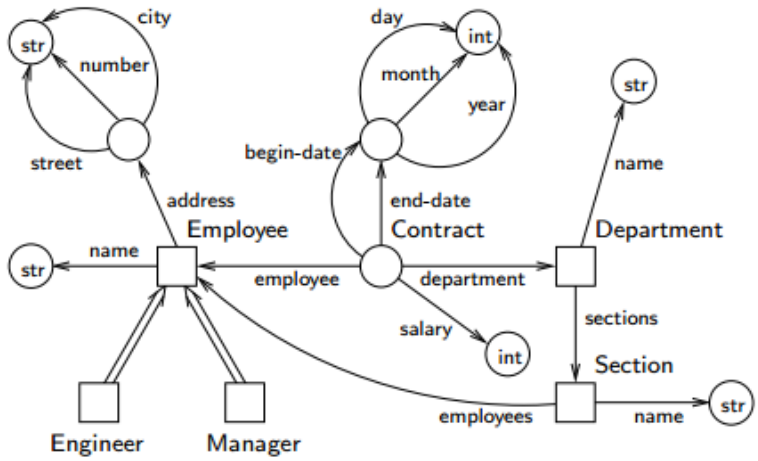


(b) Instância.

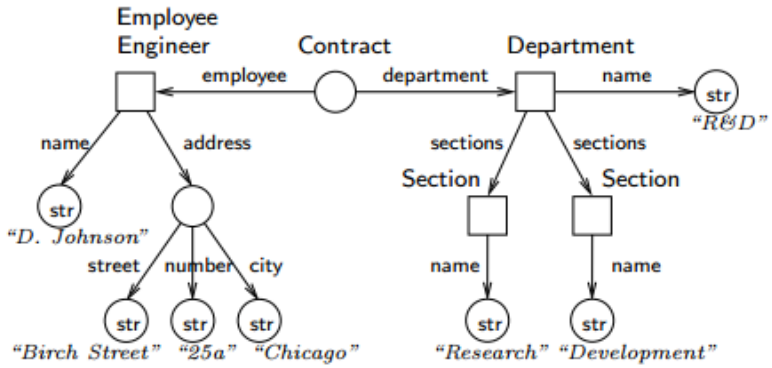
Figura 2.6: Exemplo de diagramas: Gram. Adaptado de [1].

Em outros modelos, foram criadas notações com formas diferentes para entidades

e atributos, como o *Graph Data Model* [16] apresentado na Figura 2.7. As entidade são representadas por um quadrado, sendo possível especializá-las utilizando setas mais larga, como no caso de `Manager` e `Employee`. Já os atributos são indicados pelas arestas ligadas a círculos com o tipo escrito no interior do círculo, como `int` e `str`. Na ocorrência de valores complexos, como o exemplo de `Contract` que liga `Employee` e `Department`, é utilizado um círculo vazio e um rótulo separado, com as arestas indicando os atributos, de forma semelhante ao das entidades. Esse relacionamento é apresentado na instância, onde o engenheiro `D. Johnson` trabalha no departamento `R&D`.



(a) Esquema.

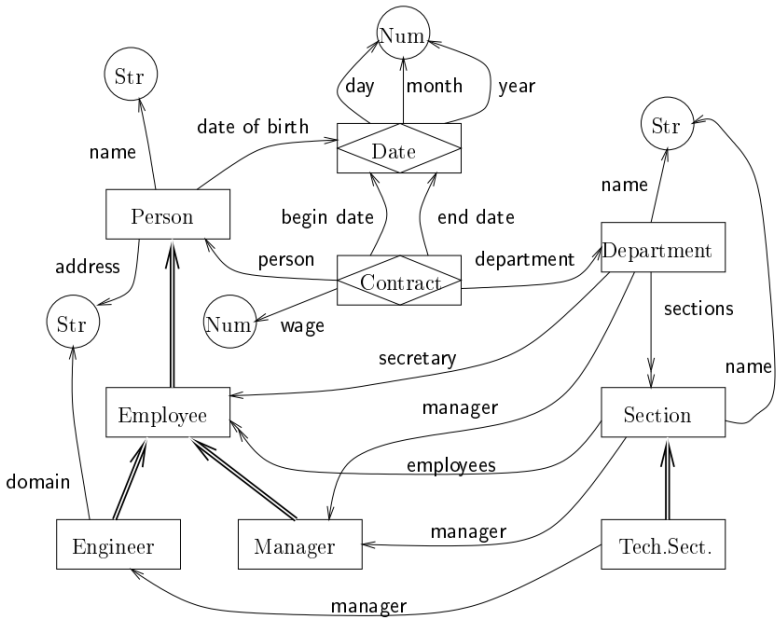


(b) Instância.

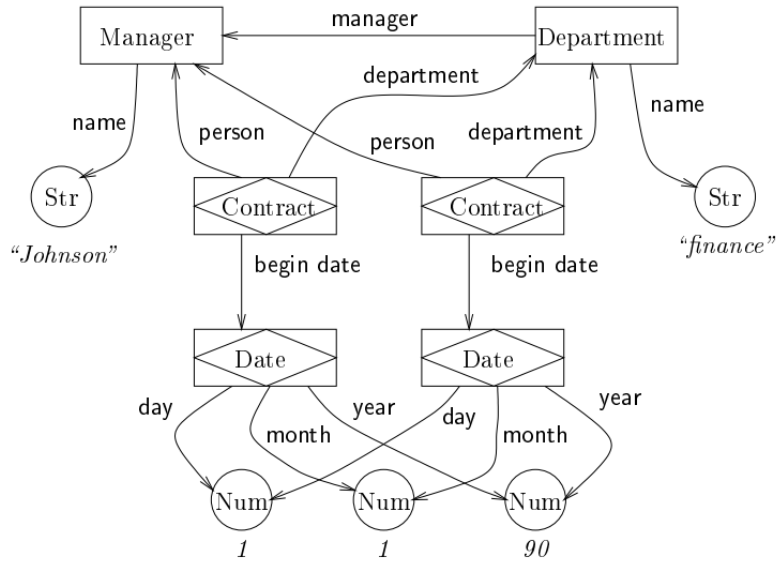
Figura 2.7: Exemplo de diagramas: GDM. Adaptado de [16].

Esse uso diferenciado de notação para atributo e entidade também é apresentado em no *GOAL - Graph-based Object and Association Language* apresentado na Figura 2.8. Este também utiliza as arestas e círculos com tipo para indicar atributos, mas os objetos complexos, como `Contract`, são representados com retângulos compostos por um losango interno. As arestas duplas (ou largas) também indicam a herança entre entidades, mas o GOAL traz também a aresta com seta dupla utilizada para representar atributos mul-

tivalorados, derivada de modelos mais antigos [12]. Um exemplo desse tipo de aresta é apresentado no relacionamento entre `Department` e `Section`.



(a) Esquema.



(b) Instância.

Figura 2.8: Exemplo de diagramas: GOAL. Adaptado de [15].

Generalizando o conceito de nós, tem-se primeiramente o *Simatic-XT* [27], representado na Figura 2.9, trabalhando com níveis de abstração. Ele possui o conceito de *Master Nodes* e *Master Edges* que permite agrupar um subgrafo para ser utilizado como vértice no nível de abstração superior. Apesar da notação não indicar um esquema dos dados, ela permite idealizar uma forma de representação para agrupamentos de subgrafos, já que

o primeiro nível de abstração, como o grafo `George`, `P1` e `Jones`, pode ser agrupado para formar os vértices do segundo nível, como `PERSON_1` para o exemplo.

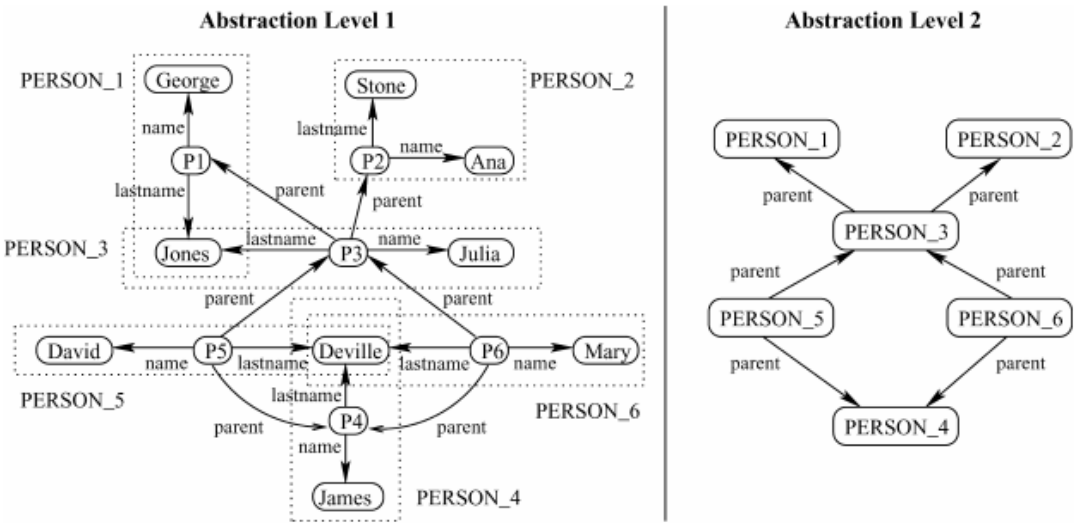


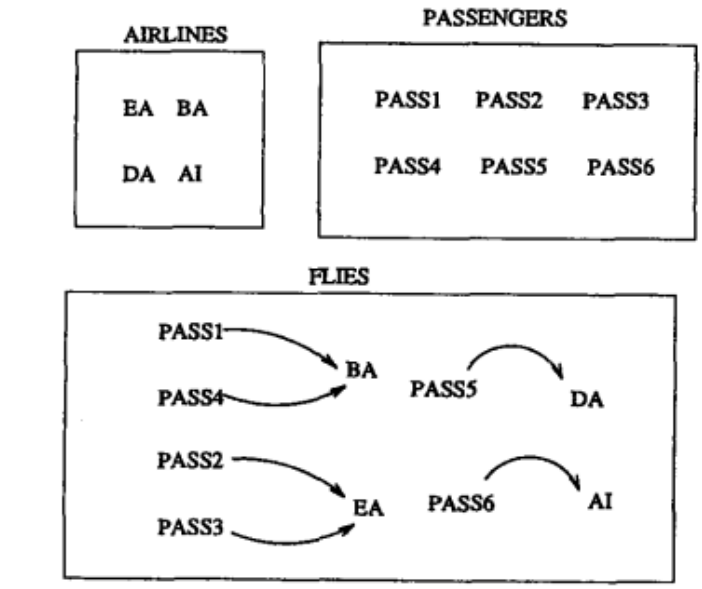
Figura 2.9: Exemplo de diagramas: Simatic-XT. Adaptado de [3].

Por fim, a Figura 2.10 apresenta o *Hypernode Model* [24]. Neste, os hipernós são construídos utilizando um rótulo, nomes de atributos, que inicial com "$", e valores atômicos entre aspas. Os arcos definem os relacionamentos do grafo no hipernó.

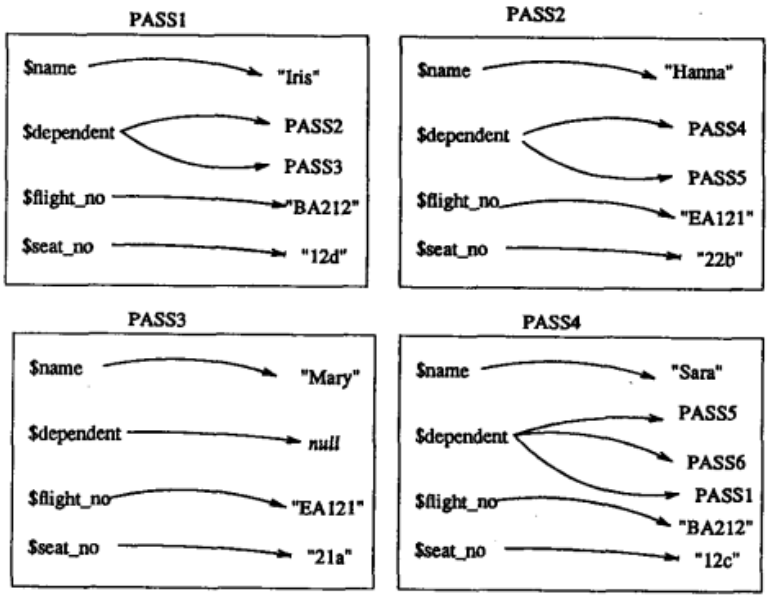
Os passageiros, por exemplo, são definidos com uma série de atributos ligados a seus valores (atômicos ou outros rótulos de hipernós), enquanto os voos são apresentados por um hipernó com seu grafo vinculando os passageiros com linhas aéreas. Sendo assim, pode-se construir hipernós mais complexos, que também poderão integrar outros grafos em outro hipernó, caracterizando-os como aninhados.

Apesar desses modelos serem os principais que serviram de base para notação, diversos outros foram desenvolvidos, principalmente durante a década de 1990 [3], e que já traziam conceitos que vêm sendo utilizados atualmente, como hipergrafos [25]. Com o surgimento dos bancos de dados NoSQL, os modelos de dados utilizando grafos voltaram a se destacar como uma solução para problemas de natureza típica dessa estrutura, onde relacionamentos são tão importantes quanto as entidades envolvidas [35]. Sendo que dessas novas iniciativas algumas características se destacaram e permitiram classificar os bancos de grafos de acordo com propriedades dos vértices, arestas ou mesmo da estrutura do grafo em si [2].

No entanto, esses modelos de grafos ainda focam apenas em suas estruturas ao apresentarem um modelo de dados, como em Robinson *et al.* [32], trabalho voltado para uma solução específica de banco de dados em grafos que traz alguns conceitos de modelagem, uma comparação com o banco relacional e exemplos de implementação. Estes últimos



(a) Voos com passageiros e companhias.



(b) Passageiros.

Figura 2.10: Exemplo de diagramas: Hypernode Model. Adaptado de [24].

utilizam um misto de diagramas de instâncias em grafo, juntamente com uma modelagem relacional para definir o modelo de dados.

Considerando a lacuna atual na modelagem para bancos de dados em grafos, utilizou-se como base os diagramas dessas iniciativas em bancos de grafos para desenvolver uma notação generalizada que cobrisse as características atuais desse tipo de banco NoSQL.

2.3 Bancos de Dados NoSQL em Grafos

O crescente volume de dados em estruturas distintas e a necessidade de gerar informação rapidamente vem forçando o estudo de novos modelos de dados além do relacional para atender as necessidades da academia e do mercado. Novas especializações, como cientista de dados, vêm se desenvolvendo para trabalhar no tratamento, manipulação e mineração de informações, seja de menor porte até as rotuladas como *Big Data* [18].

Nesse contexto, os chamados bancos NoSQL, muitos derivados dos trabalhos da Amazon [11] e Google [7], vêm ganhando espaço como novas propostas de modelos e sistemas de bancos de dados para o gerenciamento dessas informações. Pela sua crescente importância, essas novas formas de armazenamento e manipulação de informação já vêm sendo tema de diversos trabalhos, como Stonebraker [36], Leavitt [23], Hecht e Jablonski [13], Kaur e Rani [21] e Chen et al [9].

Considerando as diversas propostas de bancos de dados, uma das mais distintas tem como especialidade tratar os relacionamentos entre entidades. Os bancos de dados em grafos, apesar de descritos como um dos tipos de *NoSQL*, já são temas de trabalhos mais antigos que seus pares, como apresentado na Seção 2.2.1.

Todavia, apenas recentemente estes tipos de bancos começaram a se popularizar, principalmente pela crescente demanda por formas mais eficientes de manipular os dados estruturados em grafos, como em aplicações de química, biologia, web semântica e mineração conforme Srinivasa [35]. Em especial, uma das aplicações adequadas para a utilização de grafos consiste em tratar redes de relacionamentos. A visualização de relacionamentos e uso de teorias e algoritmos aplicados às redes de relacionamento auxiliam o trabalho de, por exemplo, montagem e estudo de cenários para compreensão, prevenção e combate às ações criminosas como apresentados em McIllwain [29], Xu e Chan [38] e Hulst [17].

Com esse novo impulso sobre a pesquisa e utilização dos bancos de dados em grafos, diversos conceitos, novos e antigos, vêm se materializando em implementações, cada uma utilizando combinações de características diferentes dos grafos, e que variam desde uma API até sistemas cliente/servidor. Essas iniciativas vêm sendo apresentadas em diversos trabalhos pela comunidade como em Angles [2], com ênfase na estrutura de dados dos bancos de grafos, e o do mesmo autor sobre performance e *benchmark* [4]. Também tratam de desempenho os artigos de Kalmegh [20], Jouili [19] e McColl [28].

Os bancos de dados NoSQL trouxeram abordagens diferenciadas do modelo relacional para organizar os dados. Atualmente, esses bancos se dividem em quatro grupos baseados na estratégia de armazenamento: chave/valor, colunar, documento e grafos [13].

Os bancos de chave/valor recebem esse nome devido aos dados serem armazenados utilizando uma chave única no banco, e um respectivo valor. Esse tipo de NoSQL foi muito influenciado pelo *Dynamo* [11], banco desenvolvido pela Amazon para armazena-

mento distribuído desse tipo de estrutura. Já os bancos colunares, influenciados pelo *BigTable* [7] do Google, alteram a forma tradicional de organizar os dados de linhas para colunas, mantendo o valor agrupado, juntamente com o identificador das linhas das quais ele pertence. Por fim, os NoSQL de documentos são bancos que permitem armazenar estruturas mais complexas[9], muitas vezes em formato *JSON* (*JavaScript Object Notation*), chamados de documentos, que podem ir desde uma chave/valor até várias informações encadeadas.

Encerrando os grupos de NoSQL, tem-se os bancos de dados baseados em grafos, ou simplesmente banco de grafos, que utilizam os conceitos de vértice e aresta para organizar dados altamente conectados, facilitando consultas típicas dessa estrutura [31]. Alguns desses bancos são simplesmente uma API, como o Sparksee (ou DEX)[1] e o HypergraphDB[2]. Outros sistemas, como o Neo4j[3] e o OrientDB[4], possuem a opção de executarem como um servidor com interface REST (*REpresentational State Tranfer*), além de disponibilizarem também a API.

Ao se trazer os grafos para os sistemas de bancos *NoSQL*, cada sistema gerenciador construiu sua implementação da forma que considerou mais adequada, mas, ainda que nos detalhes eles possam variar bastante, certas características se mantiveram semelhantes. O Neo4j e o OrienteDB, por exemplo, possuem o conceito de vértices e arestas com atributos, embora o OrientDB também implemente um modelo de dados dos NoSQL de documentos. Conforme Angles [2], os bancos NoSQL de grafos podem ser caracterizados pelos seus tipos de vértice, arestas e também pela natureza do grafo, resumidamente especificados da seguinte forma:

Pela natureza do grafo:

- Grafo simples: Os vértices e arestas possuem apenas rótulos ou um valor e todas as informações, inclusive atributos de uma entidade, são representadas por nós e arcos;
- Hipergrafo: Grafos que permitem que suas arestas possuam mais de dois vértices;
- Grafo com atributos: Esses permitem que seus vértices tenham atributos, essa característica simplifica o acessos às informações do nó [4], já que os atributos podem ser armazenados juntos com os vértices;
- Grafo aninhado e hipernó: São grafos que suportam outros grafos como nós, ou seja, todo um grafo pode ser vinculado com outro, como se fosse um vértice.

Quanto ao tipo de vértice:

[1] http://www.sparsity-technologies.com/
[2] http://www.hypergraphdb.org/
[3] http://neo4j.com/
[4] http://www.orientechnologies.com/orientdb/

- Nó rotulado: O nó aceita apenas um rótulo, que também pode ser o ID do vértice;
- Nó com atributo: O nó pode ter atributos como pares de chave–valor.

E quanto ao tipo de aresta:

- Aresta rotulada (direcionada ou não): A aresta possui um rótulo para expressar uma informação, como o relacionamento;
- Aresta com atributos (direcionada ou não): A aresta também pode conter atributos adicionais sobre o vínculo.

Em relação ao grafo simples, este considera tudo como vértice e aresta, ou seja, mesmo dados que seriam atributos dentro de tabelas serão vértices no modelo de dados. Essa característica justifica o nome da estrutura do grafo, já que os vértices e arestas são estruturas simples, restritas a um valor para identificá-los unicamente.

Já os hipergrafos e grafos alinhados podem ser utilizados para vincular de forma mais eficiente grupos de informações, como conjuntos de documentos a autores, pois as relações estariam disponíveis na mesma aresta ou agrupadas em um grafo como um hipernó [2].

Quanto aos grafos com atributos, esses permitem maior velocidade para recuperar informações relacionadas ao vértice ou aresta específica, mas ao se adicionar uma informação como atributo ao invés de um vértice, prejudica-se a possibilidade de identificação de um vínculo compartilhado, já que o mesmo será repetido em outra instância e sua identificação dependerá de uma consulta parecida com a de um banco relacional.

Apesar dos bancos de dados em grafos estarem novamente em destaque, faltam pesquisas atuais na área de modelagem de dados em grafos, até onde foi verificado, assim como uma notação formal que cubra as diversas características desses bancos, tema que esse trabalho aborda com o objetivo de preencher essa lacuna.

2.4 Conclusão

Foram apresentados nesse capítulo alguns exemplo dos modelos de dados encontrados na literatura que resumem melhor as características esperadas de um modelo de grafos, como definições de vértice ou suporte a atributos. Apesar de alguns modelos de grafos existirem desde a década de 80 [3], cada um deles foi utilizado em seu próprio contexto para problemas específicos. Entretanto, percebe-se que ainda não se elaborou um modelo de dados padrão consistente como o existente no modelo relacional e que permita trabalhar com cada uma dessas estruturas.

Muito das notações apresentadas até o momento foram propostas até meados de 2002 e estas já incorporavam estruturas que estão despontando nos atuais bancos *NoSQL* baseados em grafos[2]. As iniciativas de bancos de dados em grafos atuais também propõem

alguns modelos de dados, mas como os da literatura, focam apenas em cobrir os próprios sistemas, incorrendo no mesmo problema da não existência de um modelo padrão consolidado.

As estruturas comuns para classificação viabilizam construir um modelo consolidado cobrindo as características descritas na Seção 2.3, aplicável a diversos sistemas NoSQL de grafos. Pode-se somar à notação também aspectos externos que não foram explorados nos trabalhos resumidos aqui, como a cardinalidade entre as entidades do Modelo Entidade Relacionamento. A partir dessas ideias, foi criado o Modelo de Dados para Bancos NoSQL Baseados em Grafos, MDG-NoSQL, que é detalhado no Capítulo 3.

Capítulo 3

MDG-NoSQL: Modelagem de Dados para Bancos NoSQL Baseados em Grafos

Este capítulo apresenta uma notação para diagramação dos modelos de dados dos quatro tipos de estrutura de grafos: simples, de atributos, hipergrafo e aninhado. São apresentados detalhes da notação e alguns exemplos de utilização para ilustrar cada elemento do modelo. No fim do capítulo, é apresentado um quadro resumo da notação consolidada.

3.1 Definição do Modelo

Neste trabalho, o Modelo de Dados para Bancos NoSQL Baseados em Grafos (MDG-NoSQL) é proposto, com o objetivo de cobrir os diversos objetos utilizáveis nas estruturas de grafo. A construção da notação foi especificada para ser o mais simples possível facilitando o processo de entendimento e modelagem de um domínio.

Entre suas diversas características, pode-se citar a propriedade já consolidada de cardinalidade incorporada do modelo dos bancos de dados relacionais, permitindo um maior detalhamento da natureza dos relacionamentos entre os vértices, materializadas nas arestas.

O modelo MDG-NoSQL foi proposto para uso geral, suportando a modelagem dos principais tipos de grafos, sendo estes o grafo simples, de atributos, hipergrafo e aninhado (hipervértice). Ele pode ser utilizado em qualquer domínio que se deseje modelar os dados em grafos, sendo utilizado nesse trabalho para modelar o banco de vínculos como estudo de caso, apresentando como esse detalhes de modelagem para cada uma das quatro estruturas, inclusive com algumas variações.

3.2 Vértices Simples e com Atributos

Vértices são comuns de serem representados utilizando-se pontos, retângulos ou figuras circulares. Dessa forma, para definir um modelo de dados em grafos, é importante diferenciar o que é a representação de uma entidade e o que é instância, principalmente porque o MDG-NoSQL é apresentado como um grafo, mas descrevendo um esquema com entidades e relacionamentos, ao invés de simplesmente instâncias.

Como o objetivo é representar as entidades que existirão no banco de dados em grafos, foi utilizado um retângulo de bordas arredondadas, similar à caixa de entidade do modelo ER, o que remete a um conceito de entidade. Já para as instâncias, foi utilizado o formato circular/elíptico. Dessa forma, como apresentado na Figura 3.1, fica claro quando se está tratando de um vértice de instância ou de entidade. O nó do lado esquerdo da Figura 3.1 representa a entidade `Pessoa`, no lado esquerdo da figura tem-se as instâncias de Alice e Bob.

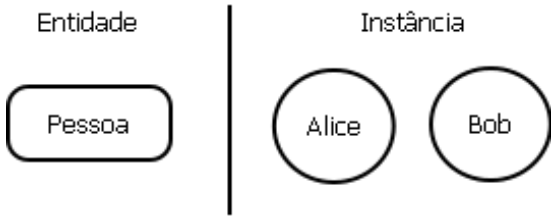


Figura 3.1: Diferença na notação de instância e entidade para `Pessoa`.

Após definir a forma de instância e entidade, pode-se estender a notação do quadrado de bordas aredondadas para que a entidade possua as informações necessárias para descrever mais precisamente suas instâncias. A notação para vértices é apresentada na Figura 3.2, utilizando obrigatoriamente rótulo, além de notações opcionais são incluídas dentro dos símbolos "[" e "]", como indicado por "Tipo" e "– Atributo: Tipo", quando necessário descrever melhor as características da entidade.

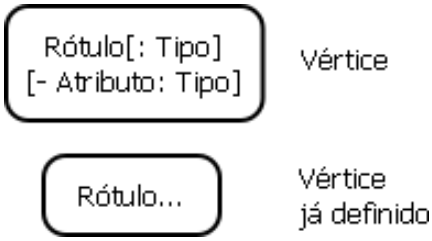


Figura 3.2: Notação para representar vértices.

O rótulo dentro da caixa possui duas funções. Primeiramente, e mais fundamental, é identificar uma entidade, como `Pessoa`, o que permite que alguns bancos de dados em grafos utilizem o rótulo para filtrar os vértices em consultas. `Alice`, por exemplo, poderá ter

um rótulo em sua implementação que remeterá à entidade `Pessoa`. A Figura 3.3 apresenta outros exemplos de entidades, indicadas pelos seus rótulos. Como previamente definido, não se está apresentando uma instância específica de `Pessoa`, `Empresa`, `Sociedade`, `Data` ou `Nome`, mas a representação da existência de um tipo de vértice relacionado com a entidade indicada no rótulo. A segunda função é definir o domínio de valores que podem identificar um vértice unicamente, caso o rótulo seja o único atributo permitido no nó. Para dar suporte a essa característica, o MDG-NoSQL faz uso da notação opcional de `Tipo` junto ao rótulo do vértice.

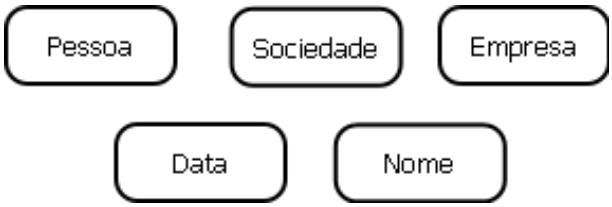


Figura 3.3: Uso da notação com diversas entidades.

O uso do tipo no rótulo pode ser uma questão de estrutura, como no caso do grafo simples, onde os atributos também são vértices e cada entidade precisa apontar para os diversos nós de atributo. A Figura 3.4 demonstra um exemplo para a entidade `Nome`, sendo seus vértices compartilhados entre `Pessoa` e `Empresa`. Isso implica que instâncias de ambas as entidades apontarão para algum vértice do tipo `Nome` que armazenará seu valor, inclusive podendo `Empresa` e `Pessoa` terem o mesmo nome, ainda que incomum. Porém, nessa situação, encontra-se o problema de como uma entidade será identificada unicamente no modelo.

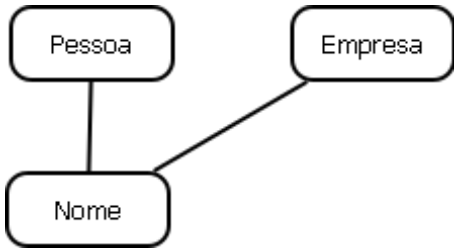


Figura 3.4: Uso da notação com diversas entidades.

A estrutura de grafo simples é um exemplo onde o rótulo do vértice também funciona como um identificador. Uma `Pessoa`, por exemplo, poderia ser identificada por vértices com números sequenciais ou um nome único, como o *login*. Sendo assim, é importante especificar qual o tipo de valor que o rótulo aceitará. Fazendo um paralelo com orientação a objetos, o rótulo funcionaria como o nome da classe e o tipo especificaria qual o domínio que um atributo com função de identificador teria dentro da classe.

A Figura 3.5 apresenta um exemplo do uso do tipo no rótulo para `Pessoa`, `Nome` e `Empresa`. A entidade `Pessoa` é indicada como inteiro grande (`long`). Esse tipo implica que as instâncias de pessoas serão identificadas por um número como, por exemplo, o Cadastro de Pessoa Física (CPF). As outras duas entidades, `Empresa` e `Nome`, são apresentados com tipos diferentes, indicando que uma instância de `Nome` será identificada por uma *string*, como o próprio nome, e `Empresa` por um char(14), que é uma forma comum de armazenar o Cadastro Nacional de Pessoa Jurídica (CNPJ) utilizado para identificar empresas no Governo Federal.

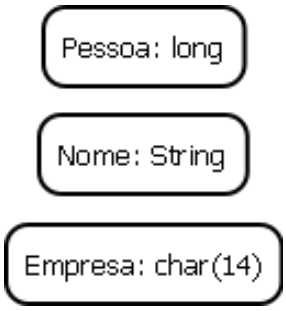


Figura 3.5: Vértices com tipo no rótulo.

Já para modelos de grafos que permitem atributos, utiliza-se a caixa do vértice com o rótulo da entidade e os demais atributos listados abaixo com os tipos correspondentes, conforme apresentado na Figura 3.6. No exemplo, `Pessoa` e `Empresa` são definidos como vértices com atributos, permitindo que o nome de ambas as entidades seja colocado como um campo do nó ao invés de tratá-lo como uma entidade a parte.

A abordagem de atributos incorporada no MDG-NoSQL, que foi derivada do modelo entidade relacionamento, representa diversos atributos dentro da notação de entidade. Ela também permite simplificar o diagrama por não ser necessário desenhar todos os atributos como vértices. Porém, deve-se lembrar que um dos benefícios do grafo é enfatizar os relacionamentos e colocar todos os atributos em um vértice pode prejudicar essa característica, sendo necessário avaliar o impacto sobre o modelo do que deve ser ou não evidenciado. Essa discussão é retomada com mais detalhes no estudo de caso do Capítulo 4 utilizando-se alguns exemplo do Banco de Vínculos Simplificado para Licitações e Sociedades.

Por fim, o modelo pode crescer e dificultar a ligação entre diversas entidades, como não foi verificado na literatura uma forma de diminuir a sobreposição de arestas, foi elaborada para esta notação uma forma de se referenciar as entidades já descritas, utilizando o símbolo de "...", junto ao rótulo do vértice. Ela permite fazer uma ligação com a entidade já definida, sem precisar passar um arco para outro ponto distante do diagrama, como apresentado na Figura 3.7. No exemplo, o relacionamento entre `Órgão` e `Pessoa` pode ser

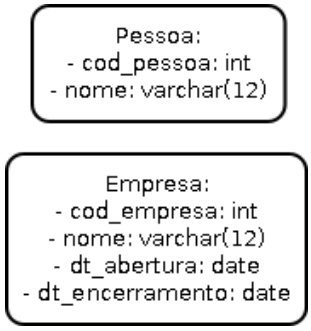


Figura 3.6: Vértice com atributos.

expressa sem ter que repetir a notação completa da entidade `Pessoa` no lado oposto do diagrama. Ao se utilizar essa notação, não se está definindo uma nova entidade, apenas evitando que um arco cruze todo o diagrama.

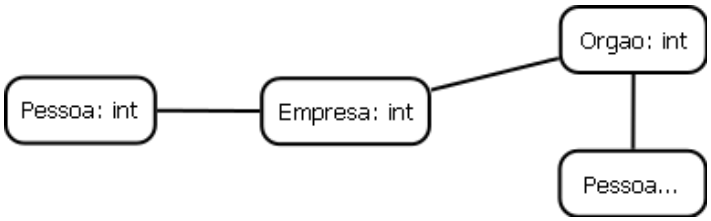


Figura 3.7: `Empresa` ligada a uma referência da entidade `Pessoa`.

Apesar do modelo apresentado na Figura 3.7 ser simples, se o modelo começar a crescer e tratar diversas entidades, o uso de um vértice como referência a uma entidade já definida pode simplificar o modelo, principalmente se os vértices possuírem atributos, e pode evitar uma poluição de arestas no diagrama.

3.3 Hipervértices (Hipernós) do Grafo Aninhado

Os hipervértices, ou hipernós, permitem a criação de grafos aninhados, já que são vértices construídos a partir de outros grafos. Eles consistem em indicar grafos ou subgrafos dentro de uma fronteira retangular. Esse tipo de notação é similar a já utilizada no *Hypernode Model* da Figura 2.10, assim como o uso de limites para agrupar objetos, presente no modelo Simatic-XT, da Figura 2.9. Porém, esses modelos foram estendidos para incorporar um tipo de rótulo e um valor para a quantidade esperada de subgrafos que formarão o hipernó, como descrito mais detalhadamente nesta seção.

O hipervértice é um grafo ao mesmo tempo que um nó. Como vértice, ele pode compor outros grafos com níveis de abstração mais alto, como no exemplo do Simatic-XT, da Figura 2.9. Já como um grafo, pode-se imaginar que um conjunto de vértices e arestas foi "marcado" para indicar que são parte de uma estrutura maior, o hipervértice, e

que possui seus próprios atributos. A notação para hipernós é apresentada na Figura 3.8 e foi diferenciada dos vértices simples pela borda reta do retângulo como o do *Hypernode Model* [24]. Dentro do hipervértice, deve ser colocado o modelo de grafo que ele fará referência.

Opcionalmente, pode-se detalhar os hipervértices, adicionando-se um indicador para o número de grafos que se espera dentro de cada hipernó, o tipo do rótulo, para situações similares aos vértices simples, e quais seus atributos, caso existam, destacados dentro do retângulo por um símbolo UML (*Unified Modeling Language*) [30] de nota, diferenciado dos demais desenhos utilizados para instâncias e entidades.

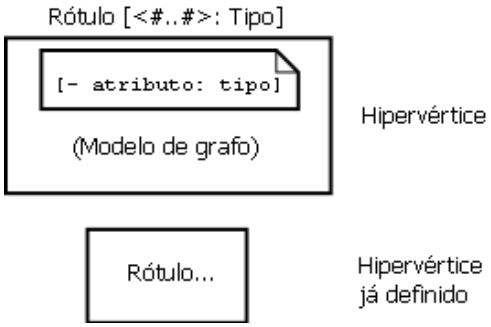


Figura 3.8: Notação de hipervértice.

Para melhor compreensão do hipernó, pode-se imaginar um cenário onde uma cidade é segmentada em regiões. Cada região define uma comunidade, com grupos de pessoas que vivem e possuem uma função específica dentro dela, seja de morador até líder da associação dos moradores. Supondo que as informações de moradores sejam um grafo relacionando os vértices do tipo `Pessoa`, `Função`, `Endereço` e `Região`, pode-se agrupar as diversas regiões em hipernós do tipo `Comunidade`. A Figura 3.9 apresenta um modelo para essa situação.

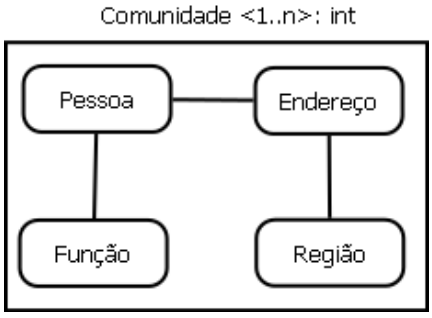


Figura 3.9: Exemplo de Hipernó.

Os valores entre os símbolos "<>" indicam que existem vários subgrafos em uma comunidade com a topologia de entidades do interior do hipernó. Sendo assim, o subgrafo do exemplo consiste em uma `Pessoa` em determinada `Função`, ligado a um `Endereço` de

uma `Região`. Como diversas pessoas residem na mesma `Região`, tem-se vários subgrafos com essa topologia e esse conjunto forma o hipernó de `Comunidade`. A Figura 3.10 destaca um exemplo de instância para `Comunidade` com dois subgrafos da topologia descrita no modelo, um em azul e outro em vermelho, onde o vértice `Região X` é compartilhado entre os dois, justificando o porque de estarem no mesmo hipervértice.

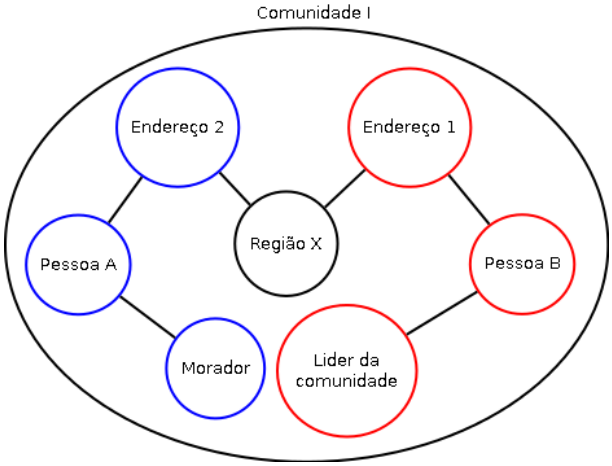


Figura 3.10: Exemplo de instância para `Comunidade`.

A indicação do número de subgrafos ajuda a distinguir quando se trata de coleções ou apenas um subgrafo. Supondo, por exemplo, que as comunidades interajam com as prefeituras de suas cidades, pode-se querer modelá-las também como um hipernó `Prefeitura`. De forma simples, pode-se imaginar a `Prefeitura` com um `Gabinete`, com o `Prefeito` e seus `Assessores`, e uma coleção de departamentos modelados também como um hipernó `Departamentos`. Esse hipervértice possuiria todos os departamentos da `Prefeitura` como apresentado na Figura 3.11. Dessa forma, uma instância do hipernó de `Prefeitura` terá apenas um subgrafo com o `Gabinete` do prefeito e um hipernó de `Departamentos`, diferente de `Comunidade` que possui uma coleção de subgrafos da topologia definida na notação.

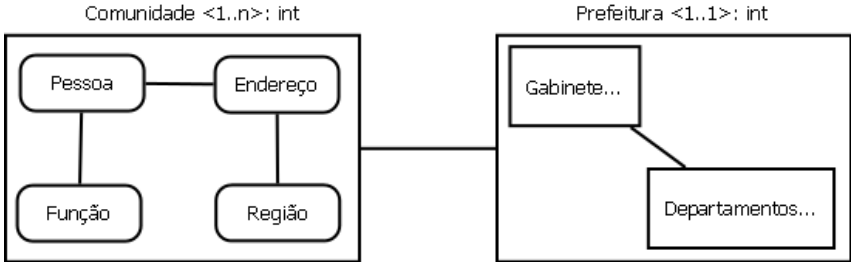


Figura 3.11: Hipervértice de `Comunidade` e `Prefeitura` se relacionando.

A possibilidade exemplificada na `Prefeitura` de incluir outros hipervértices justifica a natureza aninhada dessa estrutura de grafos. Sendo assim, pode-se ter instâncias dentro de

instâncias, como apresentado na Figura 3.12. Nesse exemplo, como definido no modelo de `Prefeitura`, tem-se apenas um objeto de cada hipervértice `Gabinete` e `Departamentos`, e cada um deles possui seu próprio subgrafo.

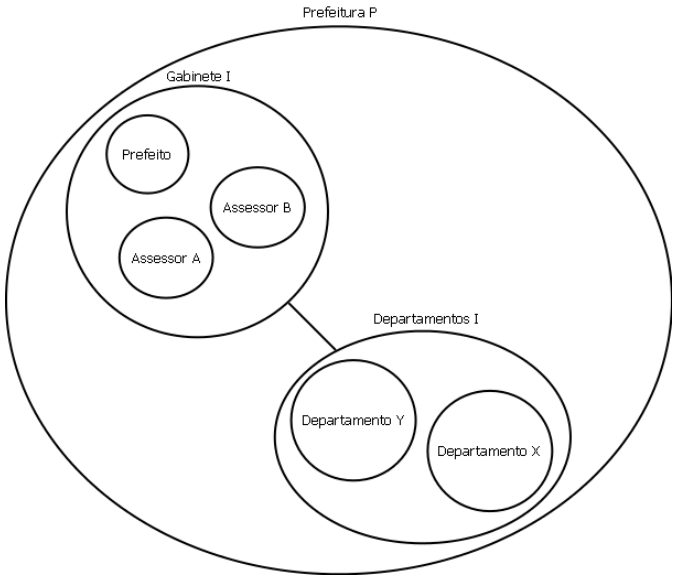


Figura 3.12: Exemplo de instância para `Prefeitura`.

Como o hipernó em si também é considerado um vértice no grafo aninhado, este pode se ligar com vértices ou hipervértices e também possuir atributos próprios da entidade que representa. No exemplo de `Prefeitura`, um atributo, como o número de CNPJ, pode ser adicionado ao hipernó, como mostrado na Figura 3.13. Nota-se que esse atributo não está relacionado com as entidades que compõem o hipernó, mas com o subgrafo como um todo que forma a `Prefeitura`.

A Figura 3.13 apresenta o relacionamento entre `Convênio` e `Prefeitura`, já que essa entidade pode firmar convênios para realização de programas voltados para as comunidades, exemplificando como um hipervértice pode se relacionar com um vértice. Já o hipervértice `Departamentos` foi detalhado para ilustrar sua composição, assim como o relacionamento de `Licitação` com `Departamento`. Nessa modelagem, a entidade `Licitação` foi representada como um vértice para demonstrar que o relacionamento de um hipervértice pode acontecer tanto com um vértice como com outro hipervértice.

Por fim, como nos vértices comuns, o hipervértice também pode utilizar o símbolo "..." para indicar que a entidade já foi definida em outro local. Isso é exemplificado em `Gabinete` dentro de `Prefeitura`, supondo que esse já foi descrito em outro local.

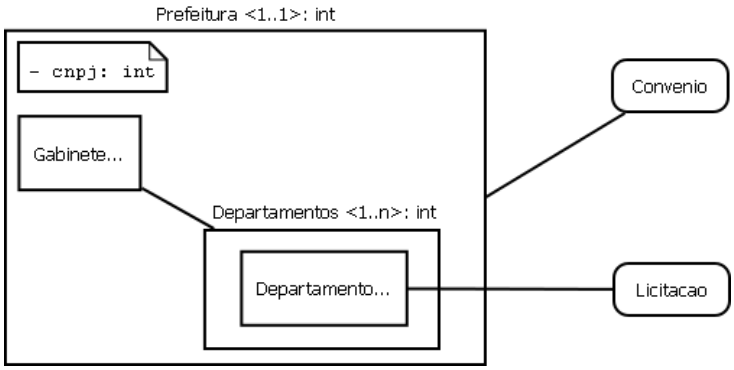


Figura 3.13: Uso de atributos e relacionamentos entre hipernós e vértices em `Prefeitura`.

3.4 Arestas Simples e de Atributos

Concluído a notação para vértices e hipervértices, é necessário realizar a ligação entre eles utilizando as arestas. Cada aresta deve ter seu rótulo, utilizado para indicar qual o relacionamento entre os nós. A Figura 3.14 apresenta dois tipos de arestas, direcionada e não direcionada, para indicar o sentido dos relacionamentos, além da notação do rótulo, podendo conter apenas o texto caracterizando o relacionamento, ou envolvendo outras características como cardinalidade, peso, tipos e atributo que serão detalhados ao longo desta seção.

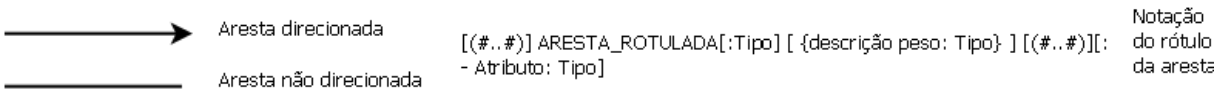


Figura 3.14: Notação para arestas.

Inicialmente, uma forma simples de representar um relacionamento seria utilizar a aresta não direcionada com um rótulo simples. A Figura 3.15 apresenta um exemplo do relacionamento de amizade. Como esse relacionamento é simétrico (ou bidirecional), já que as pessoas são amigas entre si, não é necessário indicar o sentido no segmento de reta utilizando uma seta, ou, se o grafo for direcionado, deve-se ter duas setas para indicar a bidirecionalidade do relacionamento. No esquema, por esse relacionamento envolver instâncias de `Pessoa`, a aresta é desenhada ligando a entidade nela própria, indicando que sai de uma instância de `Pessoa` e chega a outra da qual ela é amiga. Essa estrutura de esquema já foi utilizada em outros modelos, como o Gram, ilustrado na Figura 2.6.

Pode acontecer de duas entidades se relacionarem por mais de uma forma, como duas datas para um processo de licitação. Nesse caso, pode-se ter mais de um arco entre as entidades, mas cada um indicando um relacionamento diferente, como apresentado no esquema da Figura 3.16, onde a data de publicação e homologação geram duas arestas no modelo. O exemplo de instâncias também é apresentado para duas licitações demons-

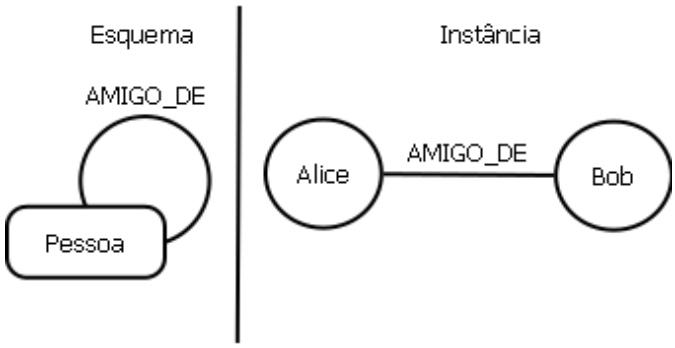


Figura 3.15: Exemplo de aresta não direcionada.

trando as arestas indicando os relacionamentos com as datas. Conforme a modelagem no esquema, a instância do processo com número iniciando em "1" possui as duas arestas para instâncias da entidade `Data`, uma de abertura e outra de homologação. Entretanto, o segundo processo de licitação ainda não foi homologado e apenas se relaciona com a data de abertura, nesse caso igual à data de homologação do primeiro processo. Observa-se que, mesmo se uma instância pudesse ter a mesma data de publicação e homologação, seriam duas arestas ligando uma instância de `Licitação` a uma de `Data`, cada qual com um rótulo de relacionamento diferente.

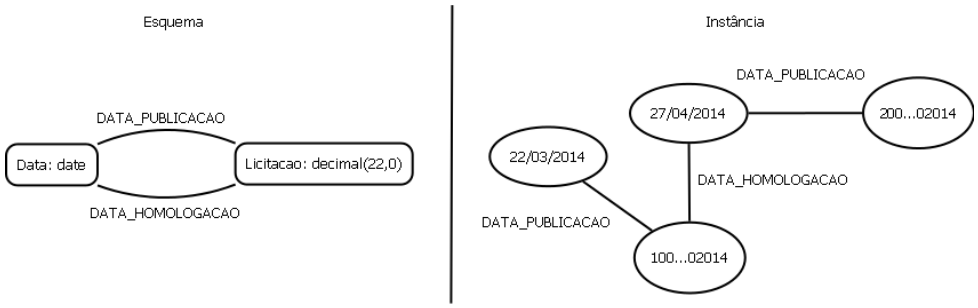


Figura 3.16: Exemplo de aresta não direcionada.

Utilizando um outro exemplo, poder-se-ia representar qual `Livro` determinada `Pessoa` leu utilizando-se uma aresta direcionada, como apresentado na Figura 3.17, já que o relacionamento é assimétrico (uma `Pessoa` leu um `Livro`, mas um `Livro` não pode ler uma `Pessoa`). O esquema do MDG-NoSQL modela como as entidades se relacionam, nesse caso de forma direcionada, e a instância apresenta os dois livros que `Alice` já leu. Ressalta-se que grafos direcionados são comumente desenhados através de setas e o modelo apenas definiu qual o tipo de seta será utilizado.

Entretanto, no modelo, não fica claro quantas pessoas ou quantos livros podem estar envolvidos no relacionamento, sendo necessário acrescentar ao rótulo essa informação. Para isso, pode-se utilizar o conceito de cardinalidade já presente no modelo entidade relacionamento. Na modelagem entidade relacionamento, a cardinalidade indica a quantidade

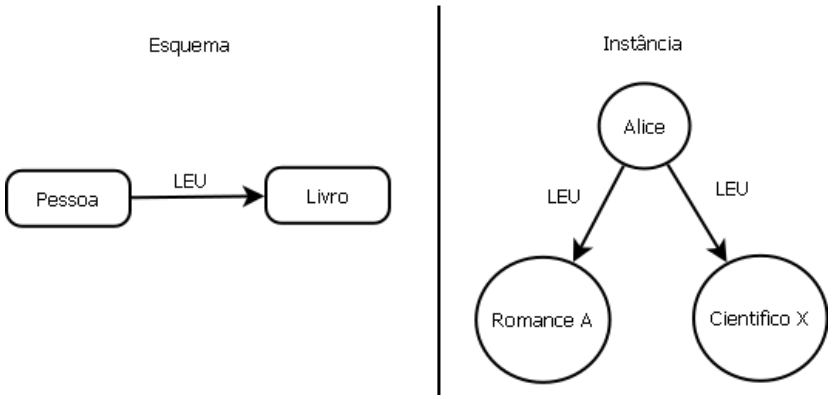


Figura 3.17: Exemplo de aresta direcionada.

de instâncias de uma entidade que pode se relacionar de alguma forma com as tuplas de outras [14]. Existem três possibilidades de cardinalidade para um relacionamento:

- 1:1 – Cada entidade pode participar de apenas uma instância de relacionamento com uma outra entidade, como, por exemplo, um departamento que possui apenas um gerente responsável por ele;
- 1:N – Cada entidade pode participar de uma ou mais instâncias de relacionamento com outra entidade, mas esta segunda participa apenas de uma, como um departamento e empregado, onde um departamento pode ter vários empregados, mas um empregado trabalha apenas em um departamento;
- M:N – Cada entidade pode participar de uma ou várias instâncias de relacionamento com outra entidade, mas diferente de 1:N, a segunda entidade pode participar de muitas instâncias também, como, por exemplo, projeto e empregado, onde um projeto possui vários empregados alocados, e estes participam de vários projetos.

Utilizando a cardinalidade da mesma forma que no modelo entidade relacionamento, um exemplo mais elaborado com livros é apresentado na Figura 3.18, onde ao invés de registrar a leitura, são registradas as compras dos livros por clientes do tipo `Pessoa`, sendo necessário indicar quantas compras foram realizadas por um cliente e quais livros foram comprados. Nesse exemplo, utiliza-se uma notação de cardinalidade, onde os valores dentro dos símbolos de "(" e ")" próximos aos vértices indicam as quantidades esperadas de cada entidade na relação, de forma semelhante ao existente no modelo relacional.

No exemplo da Figura 3.18, uma `Pessoa` pode ter de zero (0) até qualquer quantidade de pedidos de `Compra` (n). Já quando a `Compra` é iniciada, ela deve ter obrigatoriamente um dono apenas, e por isso a cardinalidade de (1..1) no relacionamento `REALIZA`, indicada junto ao vértice da entidade Pessoa. A `Compra` também deve ter no mínimo um item, podendo ser adicionados diversos livros, o que implica na cardinalidade de (1..n). Como

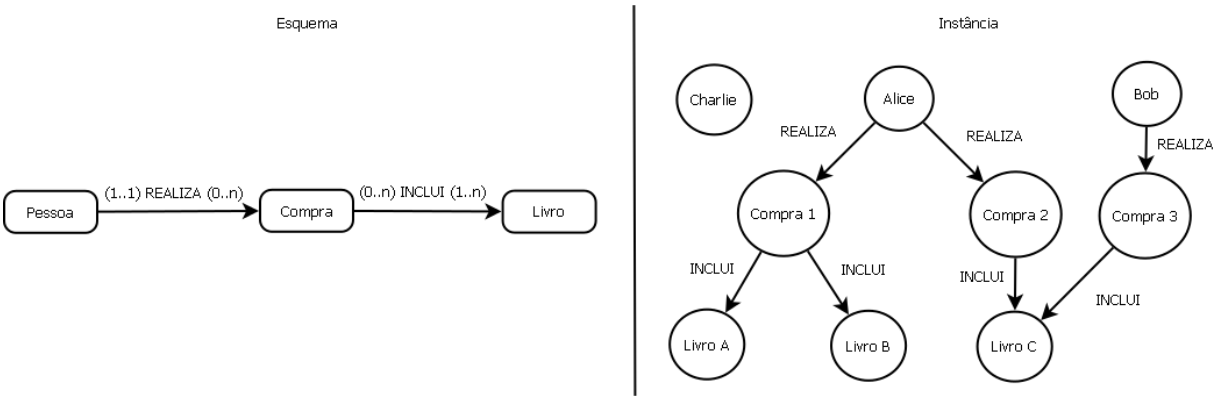


Figura 3.18: Exemplo de aresta com cardinalidade.

diversas pessoas podem comprar o mesmo título, a cardinalidade para eles é de (0..n), o que implica também que um `Livro` pode não ter sido comprado ainda.

Caso o relacionamento seja apresentado de forma diferente da horizontal, deixar a cardinalidade junto ao rótulo da aresta pode ficar confuso, ainda mais em um diagrama complexo. Sendo assim, para facilitar a leitura, pode-se também desvincular a notação de cardinalidade do rótulo e colocá-las junto à entidade, como no MER. A Figura 3.19 apresenta um exemplo para o caso de venda de livros. Uma vantagem dessa notação desvinculada do rótulo do arco é a possibilidade de utilizá-la com clareza mesmo em arestas não direcionadas.

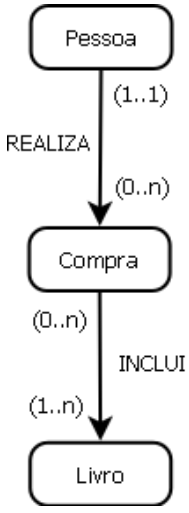


Figura 3.19: Cardinalidade no rótulo.

Importante ressaltar que, apesar da semelhança com o MER, pode-se tratar a cardinalidade como a quantidade de arestas que se espera para uma instância da entidade. Sendo assim, o relacionamento de `Compra` com `Livro` implica que poderão existir de um até diversas arestas saindo de uma instância de `Compra` com rótulo `INCLUI`, assim como para diversas arestas desse tipo chegando em uma instância de `Livro`, como foi apresen-

tado na Figura 3.18, onde `Charlie` ainda não realizou qualquer aquisição, mas `Alice` já possui duas compras de livros, sendo um deles o mesmo título que `Bob` adicionou em sua única compra.

Outra característica importante a ser acrescentada nas arestas é um tipo, assim como nos vértices. Novamente, o rótulo passa a se assemelhar com uma classe, sendo o tipo o indicador de que valores são permitidos para as instâncias. A Figura 3.20 apresenta um exemplo de esquema onde a relação de sociedade é dada por uma aresta indicando a qualificação do sócio (administrador, cotista, acionista, etc.) por uma cadeia de caracteres indicado logo após o símbolo de ":". A instância apresenta uma situação em que `Alice` se relaciona com duas empresas, sendo qualificada como diretora em uma e cotista em outra. Já `Bob` possui um relacionamento na `Empresa A` como acionista.

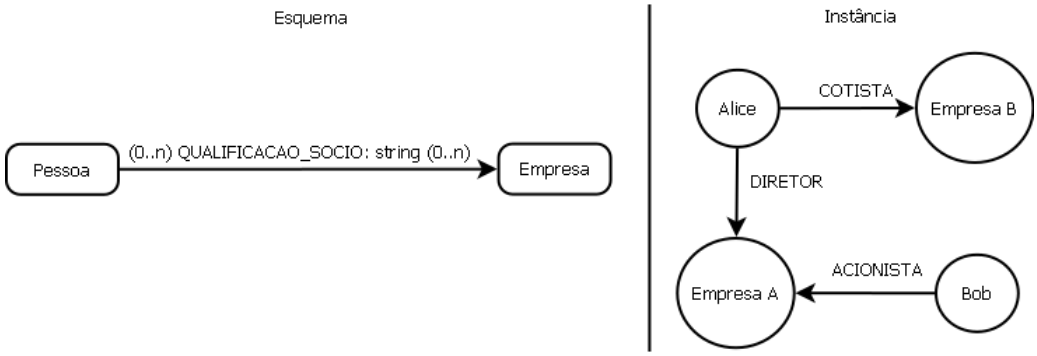


Figura 3.20: Exemplo de aresta com tipo.

Além do tipo do rótulo, as arestas também podem indicar um valor que tenha significado no relacionamento, muitas vezes chamado de peso. Alguns exemplos são: a distância em quilômetros entre cidades, a resistência entre dois terminais elétricos, o número de ligações telefônicas entre duas pessoas, ou mesmo o número de exemplares do mesmo tipo na compra de livros. A Figura 3.21 apresenta um exemplo do uso de peso na aresta, onde no relacionamento entre `Pessoa` e `Empresa`, cada sócio entra com uma cota de sociedade, podendo ir de qualquer valor acima de zero até 100%. Na instância, `Alice` possui 10% da `Empresa B` e 38,4% da `Empresa A`, onde `Bob` possui 20% de participação.

Apesar de serem parecidos, a notação de peso e tipo no rótulo da aresta indicam propriedades diferentes. O rótulo utilizando tipo, como no exemplo da qualificação na Figura 3.20, implica que o valor desse pode variar conforme o domínio definido pelo tipo. Ou seja, `Pessoa` pode ter uma aresta com rótulo de `DIRETOR` para uma sociedade e de `COTISTA` para outra conforme apresentado para o caso de `Alice`.

Para o peso, o rótulo pode até ser o mesmo, mas cada aresta possui um valor, dentro do domínio de seu tipo, referente ao que o arco representa, como distância ou cota. No exemplo da Figura 3.21, o rótulo entre `Pessoa` e `Empresa` é a `string` constante "SOCIO_DE",

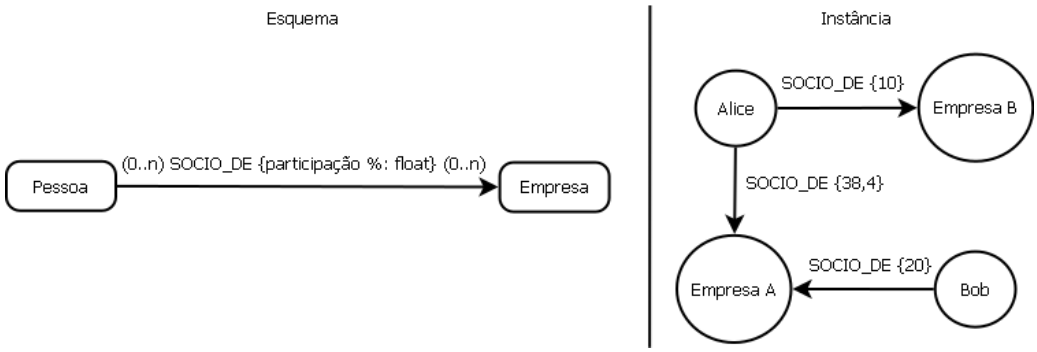


Figura 3.21: Exemplo de aresta com peso.

não precisando da indicação de tipo no rótulo, mas cada relacionamento possui um valor do tipo `float`, referente à porcentagem de participação na sociedade, como o de `Alice` com a `Empresa A`.

No MDG-NoSQL, os componentes da notação podem ser utilizados de forma independente desde que sejam claros e atendam as necessidades do modelo. Se todos os relacionamentos, por exemplo, são iguais, indicar apenas peso já seria suficiente.

Por fim, algumas arestas podem permitir o uso de atributos, assim como os vértices. A Figura 3.22 apresenta um exemplo de aresta com atributos. O relacionamento de sociedade entre `Pessoa` e `Empresa` nesse caso utiliza dois atributos na aresta para indicar qual a data de entrada e saída do sócio no quadro da empresa. Esse tipo de caracterização pode ser encontrado de forma semelhante em algumas documentações de bancos de dados em grafos com atributos, como do Neo4j, mas em geral como diagramas de instância.

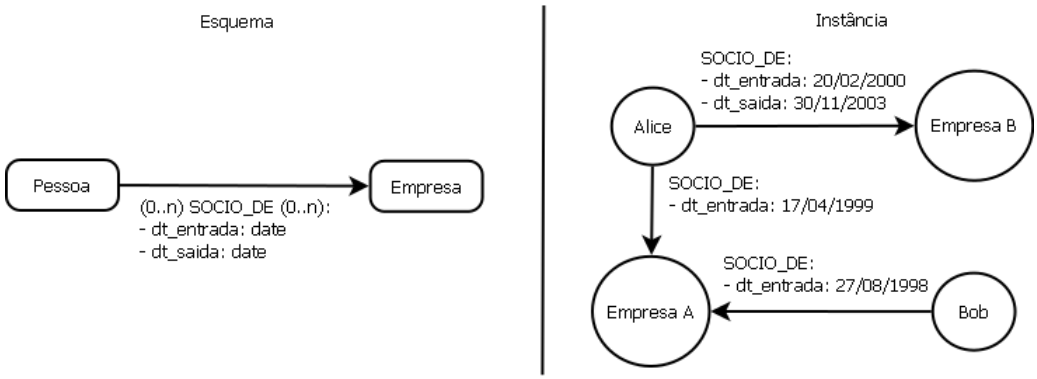


Figura 3.22: Exemplo de aresta com atributos.

Na Figura 3.22, justifica-se colocar os atributos `dt_entrada` e `dt_saida` na aresta por serem específicos do relacionamento `SOCIO_DE`. Essa característica pode ser muito útil para simplificar relacionamentos n-ários entre vértices como é apresentado na Seção 3.6. Uma característica a ser enfatizada é o fato de que o peso pode ser colocado como um

atributo da aresta, caso seja possível utilizar atributos, mas devido a sua importância nos grafos, o peso da aresta ganhou uma notação específica no MDG-NoSQL.

3.5 Hiperarestas (Hiperarco) de Hipergrafos

Até o momento foram apresentadas apenas arestas que correspondem a pares de nós, ou seja, para um par v e u de vértices quaisquer de um grafo *G(V,E)*, pode existir uma aresta e pertencente ao conjunto de arestas E, de forma que $e = \{v, u\}$. Entretanto, pode-se generalizar a ideia de aresta para um conjunto de dois ou mais vértices permitindo que $e = \{v, ..., u\}$, transformando o grafo em hipergrafo [2]. Isso permite relacionar várias informações em uma mesma aresta, que passa a ser chamada de hiperaresta ou hiperarco. De forma menos formal, uma hiperaresta é um conjunto com vários elementos. Esses elementos podem ser apenas vértices, quando a hiperaresta não for direcionada, ou ser formado por dois subconjuntos de vértices, um de origem e outro de destino, quando direcionado. A Figura 3.23 apresenta o diagrama de um hipergrafo com cinco vértice e três hiperarestas que relacionam as instâncias de entidades do tipo `Pessoa` e `Empresa`.

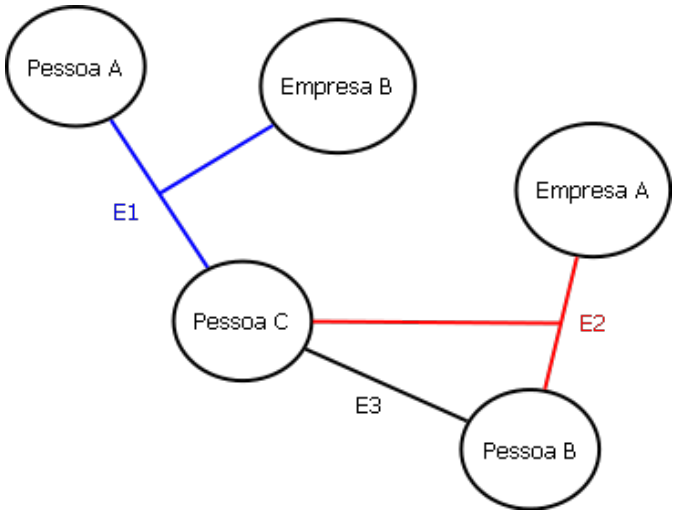


Figura 3.23: Exemplo de hipergrafo.

A hiperaresta `E1`, em azul, indica que a `Empresa B` possui a `Pessoa A` e `C` em seu quadro societário. Já a hiperaresta vermelha `E2` descreve o relacionamento societário entre a `Empresa A` e os sócios `Pessoa B` e `Pessoa C`. Por fim, a hiperaresta preta, `E3`, agrupa as `Pessoas C` e `B` por outro tipo de relacionamento, como, por exemplo, parentesco.

Embora seja possível apresentar os conjuntos de vértices das hiperarestas pelo diagrama de instâncias da Figura 3.23, essa representação não abstrai as entidades e relacionamentos, forçando a visualização de todos os objetos do grafo, que podem ficar ilegíveis com o aumento do número de elementos. Dessa forma, foi elaborada uma notação para

a modelagem do esquema utilizando hiperarestas no MDG-NoSQL. Estas utilizam linhas semelhantes às apresentadas em arestas simples, mas permitindo bifurcações no segmento de reta, característica construída para esse modelo, e com símbolos nas extremidades os arcos para diferenciá-las, indicando a presença de conjuntos com mais de um elemento da entidade na hiperaresta.

A Figura 3.24 apresenta as notações utilizadas para hiperarestas. Os rótulos não são apresentados, pois se aplicam as mesmas regras e notações utilizadas no rótulo de arestas. Ressalta-se que a notação utilizada no MDG-NoSQL não foi encontrada em outros modelos que utilizam hiperarestas, como o GROOVY [25], e, portanto, buscou-se símbolos que remetem a ideia de agregação no modelo UML [30] para construir uma notação que permitisse diferenciar o que é aresta e o que é hiperaresta.

Figura 3.24: Notação de Hiperaresta.

A hiperaresta se diferencia da aresta tradicional na notação pelo diamante e pela seta sem preenchimento. Ambas as notações são utilizadas para indicar que o conjunto da hiperaresta aceita mais de uma instância da entidade relacionada em que toca. Entretanto, a seta sem preenchimento é utilizada apenas no caso da hiperaresta possuir direção, indicando que podem existir múltiplas instâncias da entidade no conjunto de destino. No caso de se tratar de vértices de origem ou da hiperaresta não ser direciona, a ponta que possuir múltiplas instâncias no conjunto receberá um diamante. Iniciando com um exemplo mais geral de hiperaresta, a Figura 3.25 apresenta um modelo para `Empresa` participante de `Licitação` com hiperaresta não direcional.

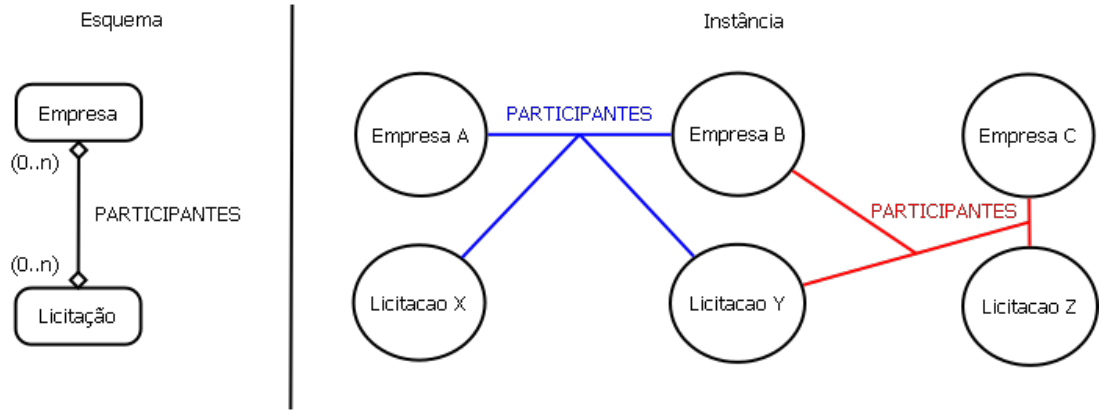


Figura 3.25: Exemplo de hiperaresta não direcionada.

Sendo assim, para o relacionamento M:N da Figura 3.25, ao invés de se ter diversos arcos compostos de pares de `Empresa`/`Licitação` comuns dos grafos, ao se utilizar hiperarestas, estas serão compostas de grupos de empresas e licitações em comuns entre elas, como apresentado na instância do hipergrafo. Dessa forma, para o conjunto de vértices V = *{Empresa A, Empresa B, Empresa C, Licitação X, Licitação Y, Licitação Z}*, tem-se duas hiperarestas com rótulo `PARTICIPANTES` no hipergrafo, sendo h_1 = *{Empresa A, Empresa B, Licitação X, Licitação Y}*, colorida em azul e h_2 = *{Empresa C, Empresa B, Licitação Y, Licitação Z}*, destacada em vermelho. Supondo que uma quarta empresa viesse a participar apenas da `Licitação Z`, esta não poderia entrar na hiperaresta `h_2`, pois a mesma não participa também da `Licitação Y`, sendo necessário criar uma nova hiperaresta com a empresa e a `Licitação Z`.

Ainda que as hiperarestas permitam diversas instâncias de várias entidades no seu conjunto, essa característica torna mais complexa a manipulação dos dados quando o conjunto de vértices e relacionamentos cresce, pois é necessário verificar as várias hiperarestas para identificar se existe alguma onde se possa encaixar um novo vértice, ou se é necessário criar uma nova hiperaresta. Sendo assim, pode-se limitar a quantidade de instâncias em alguns relacionamentos para uma unidade, forçando a simplificação. A Figura 3.26, que não utiliza o diamante na ligação com `Empresa`, apresenta um exemplo parecido com o da Figura 3.25, porém, indicando que o conjunto de cada hiperaresta terá apenas uma instância de `Empresa`. Esse hiperarco poderá incluir qualquer número de instâncias de `Licitação`, ou não existirá se a empresa nunca tiver participado de um processo licitatório. Nota-se que isso é equivalente a manter uma lista de licitações para cada empresa, de forma diferente de manter os diversos subconjuntos existentes no exemplo anterior.

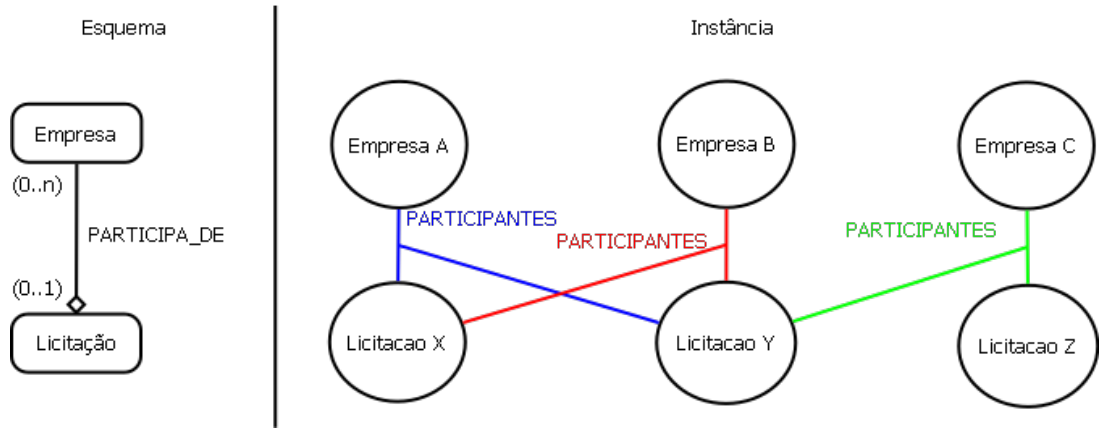


Figura 3.26: Hiperaresta com arco simples no lado de `Empresa`.

Nesse exemplo específico de hiperaresta na Figura 3.26, percebe-se uma similaridade com a notação de associação da UML. Entretanto, apesar do uso de um diamante ter-se baseado no indicador de agregação dessa linguagem, facilitando a correlação, as necessi-

dades específicas do MDG-NoSQL forçou uma dissociação entre eles. Primeiramente o símbolo de diamante pode ser utilizado em ambas as pontas, sendo esse caso uma situação específica. Segundo, as hiperarestas podem bifurcar, o que não acontece na associação. Por fim, o diamante não se localiza na UML ao lado da entidade agregada, mas na que contém a agregação. Isso acontece no MDG-NoSQL para permitir que a entidade na outra ponta, sem diamante, seja representada como participante na hiperaresta com uma unidade apenas. Assim, se houverem apenas duas entidades e ambas forem resumidas a uma instância no conjunto, a hiperaresta colapsa para uma aresta, com apenas dois vértices de cada entidade.

Considerando a cardinalidade do mesmo exemplo, nota-se que cada instância pode estar de nenhum (0) a vários (`N`) relacionamentos do tipo `PARTICIPANTES`, isso é o que indica a cardinalidade na hiperaresta. Porém, para enfatizar que uma aresta pode conter várias instâncias de quaisquer das entidades `Empresa` e `Licitação`, além de ser não direcionada, utiliza-se a notação de diamante nas extremidades que tocam cada entidade da hiperaresta. Nota-se que existe nessa notação de hiperaresta uma semelhança com o MER com a representação dos relacionamentos M:N, entretanto, o dimante no MDG-NoSQL é utilizado para indicar múltiplas instâncias da entidade no conjunto da hiperaresta e não em quantas instâncias de relacionamento a entidade pode aparecer. Dessa forma, o papel de indicar a cardinalidade continua sobre o símbolo de "(#..#)".

Um exemplo para justificar o uso em separado da notação do diamante e da cardinalidade é apresentado na Figura 3.27. Cada `Licitação` possui de uma até diversos itens, mas cada `Item_Licitação` está presente apenas em uma `Licitação`. Em um modelo utilizando uma aresta simples, ter-se-ia uma a cardinalidade (1..1) do lado de `Licitação` e (1..n) para `Item_Licitação`, conforme apresentado no exemplo. Entretanto, como as hiperarestas permitem vários elementos, pode-se colocar todos os itens de uma licitação em apenas uma hiperaresta, indicado no segundo esquema da Figura 3.27 pela ponta simples ligada a `Licitação`, e o uso do diamante junto ao `Item_Licitação`. Sendo assim, existirá apenas uma hiperaresta para cada `Licitação`, onde estarão todos as suas instâncias de `Item_Licitação`, exclusivas para esta, passando a cardinalidade de ambas as entidades para (1..1), ou seja, uma `Licitação` estará apenas em uma hiperaresta, assim como também um `Item_Licitação`.

Outro uso para hiperarestas é agrupar entidades diferentes como apresentado na Figura 3.28a, onde a hiperaresta se bifurca para ligar as diversas entidades envolvidas no relacionamento. Nesse exemplo, supondo que cada licitação tenha vários contratos para a empresa vencedora, pode-se agrupar esse relacionamento em uma hiperaresta utilizando a `Empresa`, a `Licitação`, e o `Contrato` assinado. Novamente, como cada hiperaresta possui apenas uma `Empresa`, omite-se o símbolo do diamante para esta entidade. Já seu uso em

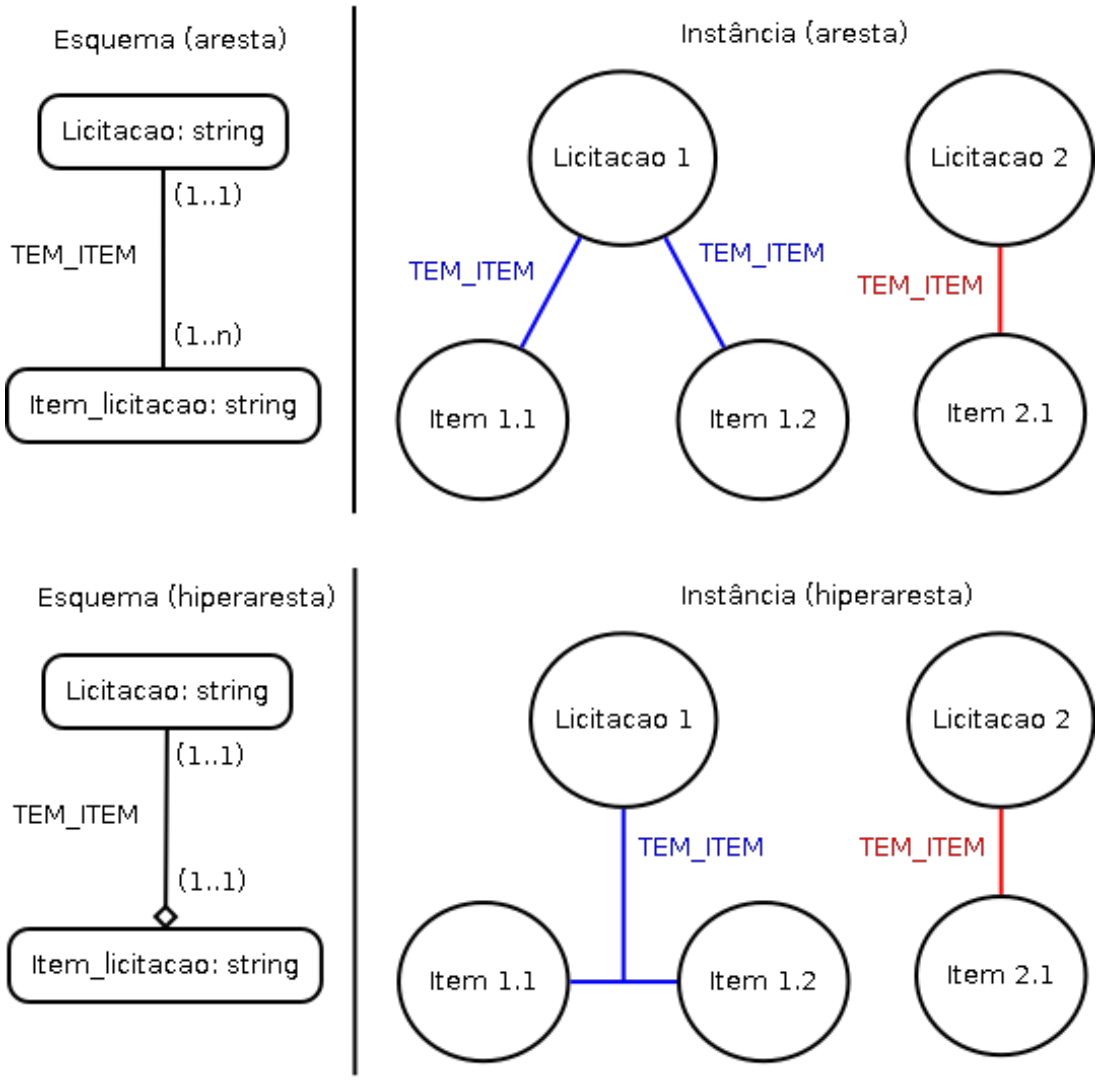
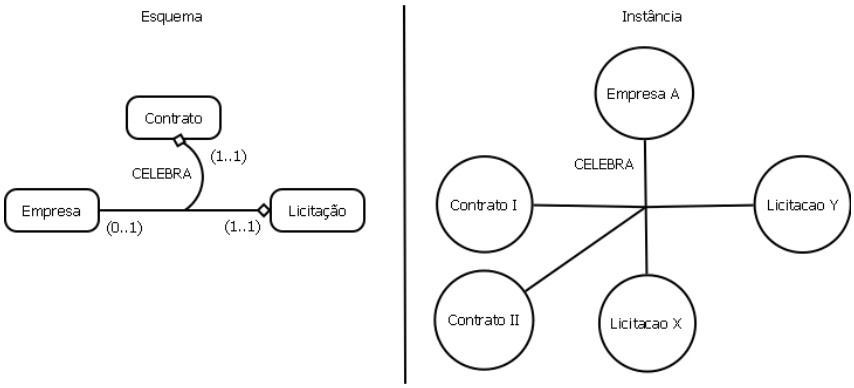


Figura 3.27: Exemplo de hiperaresta com a cardinalidade contrastando com a notação de diamante.

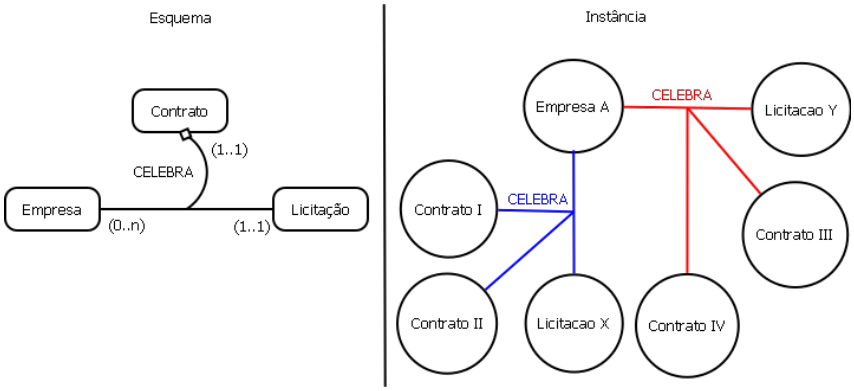
`Contrato` e `Licitação` indica que o conjunto da hiperaresta pode ter vários contratos e licitações. A instância apresenta um exemplo onde a `Empresa A` mantém `Contrato I` e `II` devido a `Licitação X` e `Y`. Embora as informações de `Empresa`, `Licitação` e `Contrato` estejam agrupadas, da forma como foi modelado não se pode indicar qual contrato se liga a qual licitação, não sendo adequado para um processo de auditoria, por exemplo.

Dessa forma, para especificar a qual `Licitação` os diversos contratos estão relacionados, pode-se limitar também a entidade `Licitação` para uma instância. Isso faz com que a hiperaresta relacione o par `Empresa`/`Licitação` com a lista de contratos gerados por ele, como apresentado na Figura 3.28b.

Nota-se que a cardinalidade das entidades permanece como no MER, onde a cardinalidade de uma entidade é indicada pelo quantidade de instâncias do relacionamento dela com o conjunto das demais. Sendo assim, cada instância de `Contrato` se relacionará ape-



(a) Hiperaresta modelada com múltiplas instâncias de `Contrato` e `Licitação` para uma `Empresa`.



(b) Hiperaresta com par de `Empresa`/`Licitação` e múltiplas instâncias de `Contrato`.

Figura 3.28: Hiperarestas vinculando várias entidades.

nas a um par de `Empresa`/`Licitação`, assim como a `Licitação` para `Empresa`/`Contrato`. Já para `Empresa` existem duas possibilidades, uma para cada modelo. Na Figura 3.28b, a cardinalidade será de (0..1), pois a hiperaresta possui todos as licitações e contratos assinados pela empresa vencedora, caso ela consiga vencer algum. No segundo modelo, da Figura 3.28b, a limitação de uma instância de `Licitação` impõem a necessidade de criar várias hiperarestas por `Empresa`, alterando a cardinalidade para (0..n), pois uma `Empresa` poderá estar em mais de um relacionamentos com `Licitação`/`Contrato`.

Assim como as arestas, hiperarestas também podem ser direcionadas. Nesse caso, a hiperaresta será um conjunto com dois subconjuntos como elementos, um deles agirá como origem e os outro será o destinos. Observando inicialmente o caso onde se tem apenas um elemento na origem/destino, pode-se ter uma situação como da Figura 3.29, onde uma `Empresa` é sócia de outra, possuindo também uma `Qualificação`.

Nessa situação, a origem e o destino ajudam a indicar qual `Empresa` é a sócia. Caso contrário, haveria duas instâncias de `Empresa` na aresta e não seria possível essa distinção. Basta imaginar a instância no exemplo sem as setas e apenas o rótulo do relacionamento.

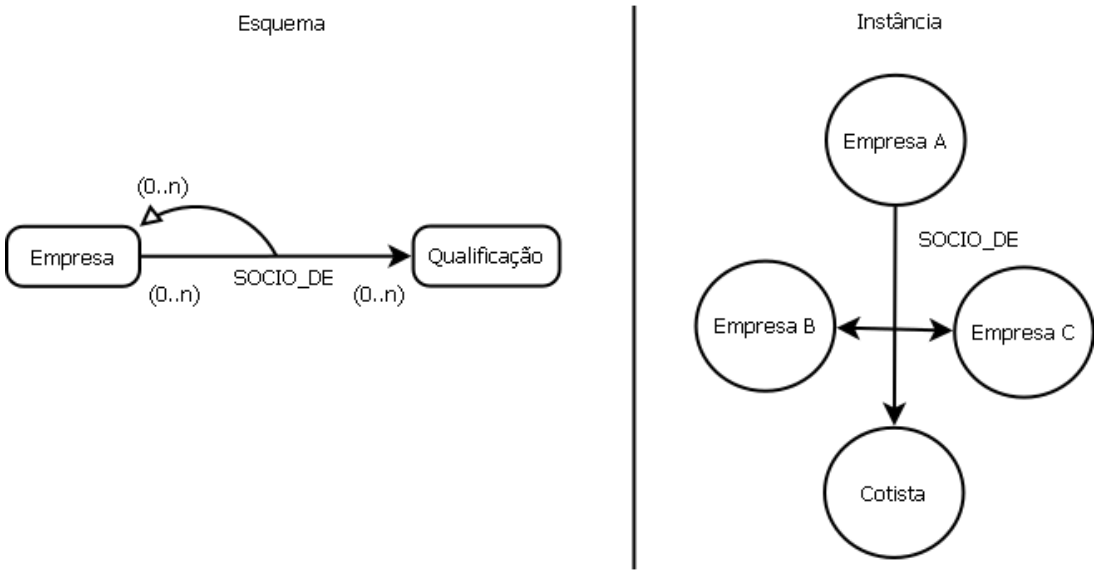


Figura 3.29: Exemplo de hiperaresta direcionada na origem.

Observa-se no esquema da Figura 3.29 que a hiperaresta possui uma origem com uma instância (`Empresa`) e dois tipos de seta, uma para `Qualificação`, preenchida como nas arestas, e outra voltando para a `Empresa`, com a seta sem preenchimento. Isso implica, como apresentado na instância do hipergrafo, que a hiperaresta terá apenas uma `Empresa` na origem, mas, no conjunto de destino, existirão várias empresas das quais a origem participa como sócio, assim como sua `Qualificação` comum para essas sociedades. Isso também implica na possibilidade de uma `Empresa` ter várias hiperarestas, com a lista das empresas comuns para cada `Qualificação` de sociedade.

Poder-se-ia utilizar um destino único também, como na Figura 3.30, para identificar qual `Cliente` comprou junto com outros os mesmos títulos de `Livro` em uma determinada `Loja`, agrupando clientes com interesses comuns e provável proximidade geográfica. Pela definição dessa hiperaresta, se tem apenas uma loja no conjunto, mas um grupo de livros e clientes, não necessariamente separando qual livro foi comprado por qual cliente. Novamente é utilizado o diamante nas pontas das origens para indicar a presença de múltiplas instâncias no conjunto da hiperaresta, mas uma seta com preenchimento, já que existirá apenas uma `Loja` para cada hiperaresta. Na instância, são apresentados os livros que `Alice` e `Bob` compraram em conjunto na Loja 1. Pela modelagem, `Alice` poderia aparecer novamente em uma hiperaresta, como, por exemplo, agrupando os livros em comum com outro cliente que não tenha títulos em conjunto com `Bob`, ou mesmo com compras em uma segunda loja. O motivo para usar a seta preenchida para indicar apenas uma instância da entidade consiste em permitir que a notação de hiperarestas, quando limitadas a uma instância tanto em origem como no destino, se iguale à notação da aresta simples.

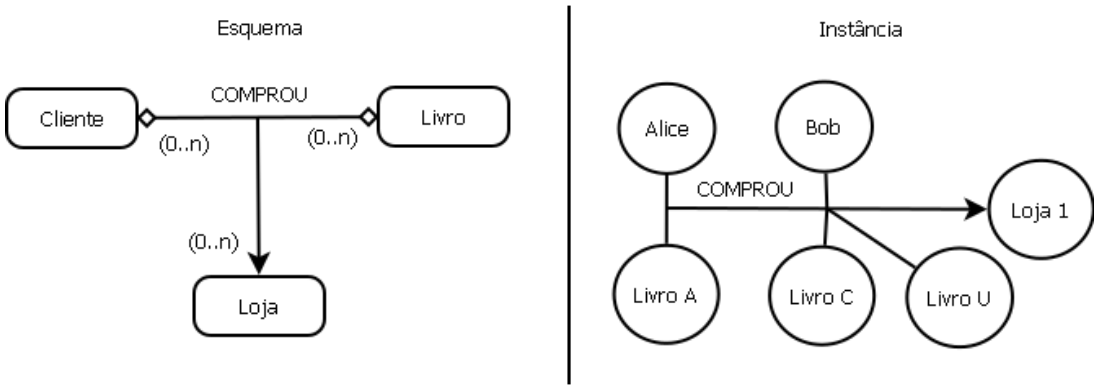


Figura 3.30: Exemplo de hiperaresta direcionada no destino.

Por fim, pode-se estender a ideia de origem e destino para uma hiperaresta direcionada, composta de um subconjunto de origem e outro de destino. Sendo assim, ter-se-ia um grupo de vértices apontando para outro grupo dentro da hiperaresta como apresentado na Figura 3.31. Como nesse modelo não há limitação nos conjuntos de origem e destino, pode-se ter vários sócios para várias empresas em comum.

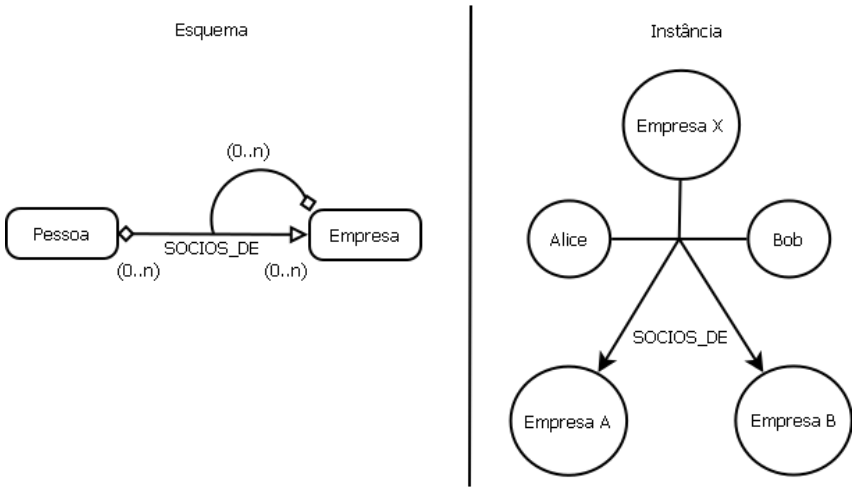


Figura 3.31: Exemplo de hiperaresta direcionada com conjuntos de origem e destino.

Sendo assim, as diversas instâncias de `Pessoa` e `Empresa` que são sócias em comuns do grupo de instâncias da entidade `Empresa` no destino são colocadas juntas na origem. Como cada entidade participa com um conjunto de vértices, utiliza-se os símbolos de diamante nas entidades de origem, e a seta não preenchida nas de destino, na construção da hiperaresta. A instância apresenta um exemplo onde `Alice`, `Bob` e a `Empresa X` são sócios em comum da `Empresa A` e `Empresa B`. Consequentemente, para um novo sócio ingressar no conjunto de origem dessa hiperaresta, deverá fazer parte dessas sociedades também.

3.6 Relações N-árias em grafos

Em certas ocasiões, os grafos precisam utilizar um nó intermediário para indicar o relacionamento entre mais de uma entidade, ou seja, n-ário. Como esse tipo de relacionamento é único apenas quando envolve as diversas instâncias, utiliza-se uma técnica já presente para grafos que consiste em criar um vértice intermediário para agrupar os relacionamentos [32]. Cada vértice de agrupamento indicará uma situação, como exemplificado na Figura 3.32 para a posse em um cargo público, onde uma `Pessoa` pode assumir um `Cargo` em determinado `Órgão`.

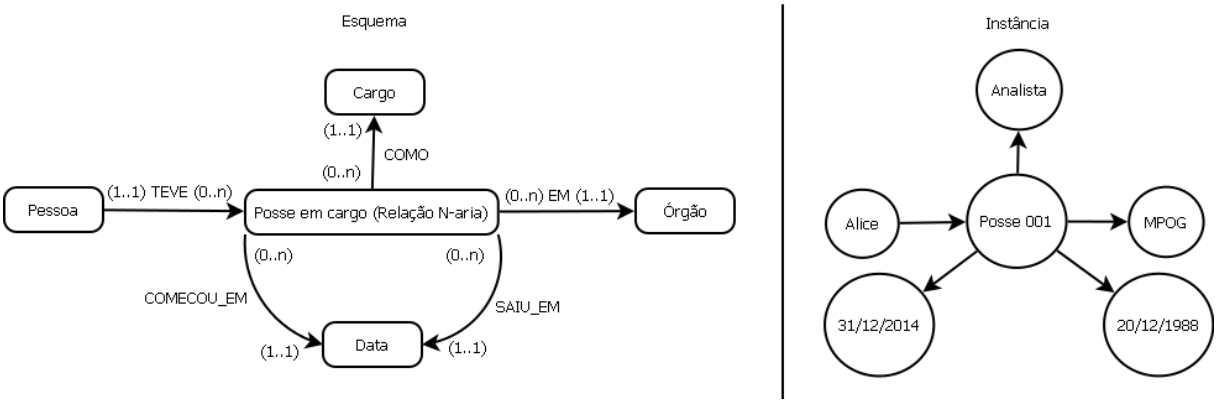


Figura 3.32: Exemplo de relação n-ária.

Enfatizando também a `Data`, como apresentado, ela surge como mais uma dimensão no relacionamento n-ário. Para essa situação, o vértice central cumpre o papel de indicar o ato da `Posse`, agrupando as informações necessárias para representar o evento, que ligadas diretamente seriam insuficientes.

No caso de hipergrafos, o fato de uma aresta poder apontar para vários vértices permite juntar as informações de forma que o vértice de agrupamento pode ser evitado. Alguns exemplos de relação n-ária já foram apresentados na Seção 3.5 e a Figura 3.33 apresenta o exemplo da posse em cargo público onde cada hiperaresta inclui todos as instâncias de `Pessoa` em um `Órgão` que tomaram posse no mesmo `Cargo` em determinada `Data`. A instância também apresenta um exemplo onde `Alice`, `Bob` e `Charlie` começaram a trabalhar como analistas no Ministério do Planejamento, Orçamento e Gestão (MPOG).

Esses agrupamentos também poderiam ser realizados em grafos aninhados, criando-se hipernós compostos por entidades que participem do relacionamento. A Figura 3.34 apresenta novamente o exemplo de `Posse`, mas agora utilizando hipernós. Foram juntados `Cargo`, `Órgão` e `Data` em um hipervértice para vincular o trio a uma `Pessoa`. No esquema, cada trio representa uma `Posse`, realizada pelo `Órgão` para o `Cargo` em uma `Data` específica. As posses se conectam com as instâncias de `Pessoa`, que também pode estar em diversas outras posses, caso tenha assumido outros cargos. A instância apresenta

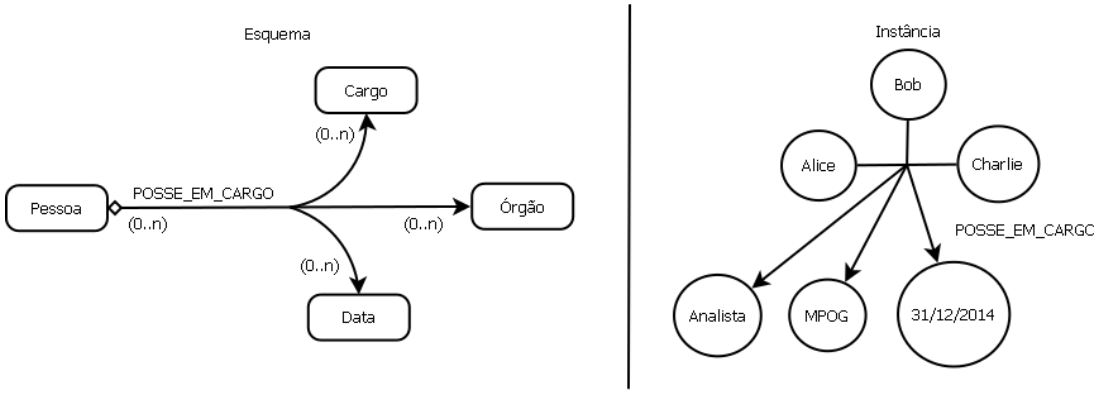


Figura 3.33: Exemplo relacionamento n-ário com hipergrafo.

o exemplo anterior, mas agora utilizando o hipernó com identificador 1 para a posse de `Alice`, `Bob` e `Charlie`.

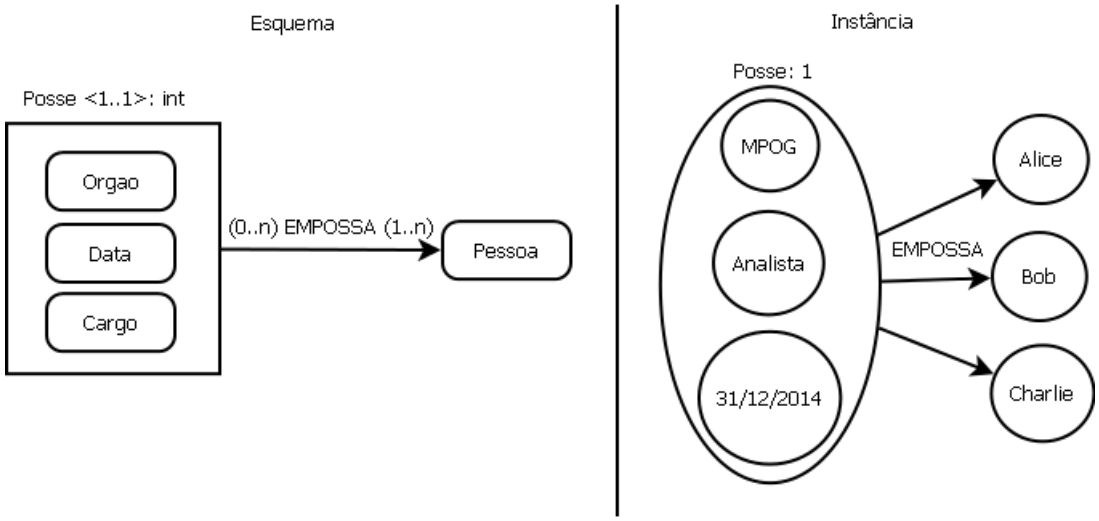


Figura 3.34: Exemplo relacionamento n-ário com hipernó.

Atributos também podem simplificar relacionamentos a ponto de se evitar a criação de vértices de agrupamento. A Figura 3.35 apresenta o exemplo de sociedade para `Empresa` utilizando a `Qualificação` e `Data` como atributos da aresta. Como ambas estão vinculadas ao relacionamento, são colocadas como atributo da aresta.

Nesse caso, não se dá ênfase no relacionamento de `Empresa` com `Qualificação` e `Data` ao colocá-los como atributos, mas se evita aumentar a complexidade do relacionamento pela utilização de um vértice de agrupamento. Isso pode ser verificado na instância, onde a ligação de sociedade entre a `Empresa X` e `Y` permanece direta (sem vértice intermediário) pelo uso dos atributos.

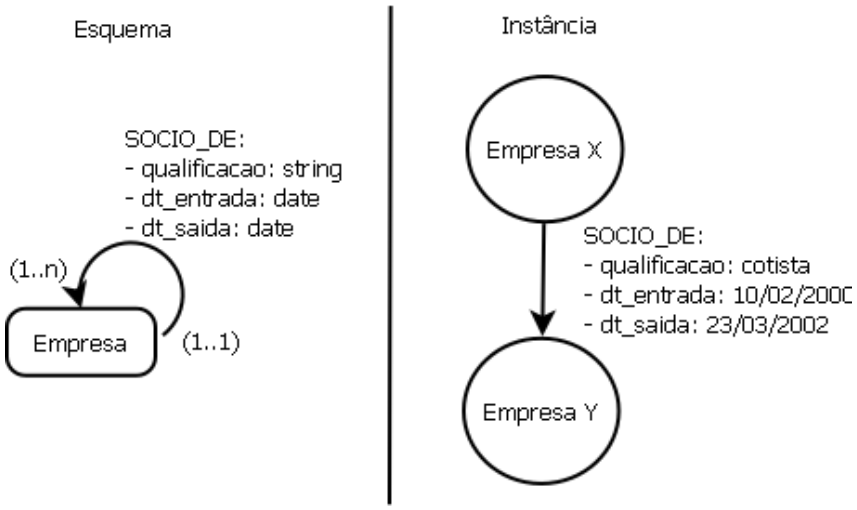


Figura 3.35: Simplificando relações com atributos na aresta.

3.7 Quadro Resumo do MDG-NoSQL

A notação proposta para esse estudo do MDG-NoSQL é resumida na Figura 3.36. Os elementos definidos de vértice, hipervértice, aresta e hiperaresta são apresentados em conjunto, assim como as informações opcionais entre os símbolos de "[" e "]".

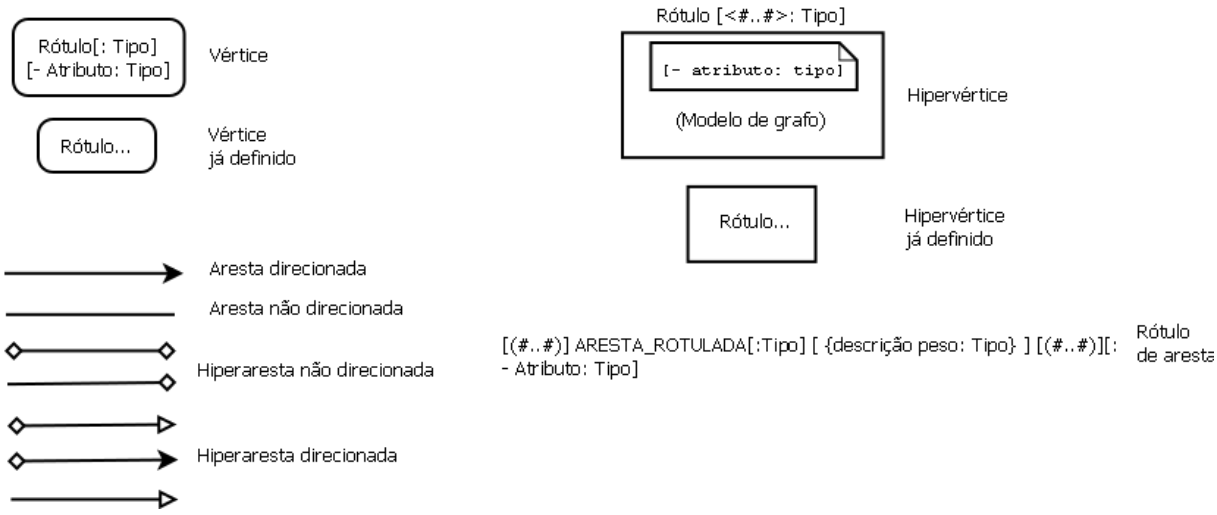


Figura 3.36: Notação consolidada para o Modelo de Dados em Grafos.

Como já comentado, cada componente foi definido para atender a modelagem das diversas estruturas de grafo. Dessa forma, pode-se construir diagramas desde grafos simples até hipergrafo aninhados com atributos. No Capítulo 4 é apresentado a implementação do Banco de Vínculos Simplificado para Licitações e Sociedades que utilizará o MDG-NoSQL para descrever o domínio proposto em um grafo de atributo e apresentando o relacionamento de sociedades em cada uma das outras três estruturas de grafos descritas no Capítulo 2. Isso porque o grafo de atributos possui uma flexibilidade que pode levá-lo

de um grafo simples a uma estrutura similar ao modelo relacional, através da forma de organização dos seus atributos, sendo sua avaliação um bom ponto de partida para a construção de considerações sobre um modelo de dados baseado em grafos.

Capítulo 4

Banco de Vínculos Simplificado para Licitações e Sociedades

Este capítulo apresenta o estudo de caso utilizado nessa pesquisa para comparar os modelos de grafo e relacional, assim como avaliar diversos diagramas utilizando o MDG-NoSQL. As seções seguintes apresentam uma descrição do domínio, a modelagem relacional, e os modelos de grafo para o relacionamento de sociedades utilizando suas várias estruturas de grafo. No Capítulo 5 são apresentados análises da implementação para algumas consultas pertinentes ao tema de auditoria.

4.1 Auditoria e o Processo de Licitações

A CGU, órgão do Governo Federal, além dos diversos componentes de sua estrutura hierárquica, possui uma diretoria ligada diretamente à Secretaria Executiva que auxilia na prevenção e combate à corrupção. A Diretoria de Pesquisas e Informações Estratégicas (DIE) tem como competência subsidiar com informações de inteligência as diversas áreas da CGU, colaborando com informações para uma melhor tomada de decisão por parte dos clientes de seus serviços. Uma das atividades da diretoria é tentar identificar indícios de fraude em processos de licitação. Para isso a DIE reúne e processa dados para gerar informações e construir relacionamentos que permitam avaliar como diversos participantes desses processos se relacionam. Essas informações podem estar nas bases internas do governo no qual seja garantido acesso aos auditores do órgão, ou mesmo públicas, nos diversos sítios disponíveis na Internet.

Pessoas ou empresas podem ser utilizadas como entrepostos para mascarar relacionamentos diretos que seriam, no mínimo, conflitantes, prejudicando a licitude dos contratos firmados com a administração pública. Por isso as informações de relacionamento permitem identificar indícios de fraudes em processos como de licitações. Um caso hipotético

seria de duas empresas que participam do mesmo processo de compra e possuem sócios com algum parentesco ou relação indireta. Essa seria, no mínimo, uma situação de risco para a lisura das licitações no órgão, pois essa relação permitiria que o preço fosse acertado entre as duas empresas para tentar influenciar no resultado.

Atualmente, para dar suporte a essa atividade, a diretoria agrupa os dados em bancos relacionais e realiza pesquisas através de ferramentas de *Business Intelligence* (BI) [26] ou com consultas diretas às bases. As pesquisas correspondem à busca de informações principalmente de vínculos, como sociedades, participações conjuntas de licitação ou posse em cargo público. Algumas dessas são detalhadas e exploradas no Capítulo 5.

Para o capítulo presente, são avaliados os modelos de grafo e relacional em um modelo simplificado do processo de licitação, devido à relevância do tema para o Governo Federal e grande interesse dos órgãos de Controle. O processo de licitação é utilizado pelo Governo Federal para realizar as aquisições de bens e serviços. O principal dispositivo legal é a Lei № 8.666/93 que descreve as diversas modalidades de licitação, mas para esse trabalho apenas uma pequena parte será utilizada, pois o objetivo principal não é elaborar um sistema para o processo de licitação, mas a aplicação do modelo em uma análise dos relacionamentos derivados nesse cenário.

Quando algum órgão do governo necessita realizar uma aquisição, este inicia um processo de licitação, descrevendo o que se deseja adquirir assim como as demais informações legais necessárias para garantir a lisura do processo. Como o processo é um tanto oneroso, vários itens podem ser licitados em conjunto, formando um documento único que deve ser amplamente publicado. Dessa forma, mesmo que uma licitação relacione vários itens, cada um será adquirido com certa independência e permitindo que diferentes empresas se inscrevam para concorrer a cada um deles.

Uma aquisição, por exemplo, de computadores e ativos de rede, pode ser dividida em dois itens para contemplar cada um dos tipos de equipamento. A empresa interessada em participar pode se inscrever por um item ou ambos, se tiver os requisitos necessários para ofertar o que é desejado. O objetivo é garantir a concorrência entre as empresas e promover a melhor compra pelo poder público. No entanto isso nem sempre acontece, às vezes por questões técnicas ou de mercado, mas certas vezes por uso de mecanismos fraudulentos que comprometem o processo licitatório.

Por essa razão a análise dos relacionamentos entre empresas, sócios e os processos de licitação se torna importante. Ela pode contribuir para identificar relacionamentos suspeitos que coloquem em risco uma aquisição. A partir desses vínculos, pode-se também aprofundar uma investigação buscando indícios e mesmo provas de que um determinado processo foi fraudado ou mesmo detectar quadrilhas que atuem nesse tipo de crime.

A análise de redes de relacionamentos, visualizando-os como um grafo, já é utilizado no

combate ao crime organizado [29, 34]. Nessas redes são colocadas diversas pessoas suspeitas de participarem de esquemas criminosos além de informações que possam correlacioná-las, como telefones, empresas, contas bancárias, cargos públicos, entre outros [38]. Após a construção da rede, analistas podem analisá-la buscando indícios baseados nas interações, ou pontos fortes e fracos da rede com o propósito de desarticulá-la [17]. Apesar da natureza de grafo, a maior parte das informações estão armazenadas em bancos de dados relacionais e precisam ser reorganizados para a análise.

4.2 Modelo de Dados Relacional para o Banco de Vínculos Simplificado para Licitações e Sociedades

A modelagem simplificada para o banco de vínculos no processo de licitação é apresentado na Figura 4.1. Começar pelo modelo relacional permite partir de uma base sólida e bem conhecida para então propor e elaborar o modelo de grafos, avaliando suas diferenças e características para o mesmo estudo de caso.

O diagrama apresenta o relacionamento entre entidades envolvidas nos processos de licitações, detalhando também as informações de sociedade. No Brasil, uma `Empresa` pode ter várias `Pessoas` ou mesmo outras `Empresas` como sócios. Sendo assim, as sociedades são representados por um relacionamento de M:N materializada na tabela associativa `PessoasSocios`. Como as `Empresas` também podem ser sócias, o auto relacionamento M:N dessa entidade é estabelecido pela tabela associativa de `EmpresasSocios`.

Junto aos relacionamentos de sócios estão as informações de qualificação do sócio da tabela `QualificaçõesSocio`. O objetivo é identificar qual o papel do sócio na empresa, tais como administrador, cotista, acionista ou diretor, por exemplo.

A informação de `Telefones` também possui um papel importante, pois permite identificar `Empresas` e `Pessoas` vinculadas, e por isso foi colocada em uma tabela separada, ao invés de compor o grupo de atributos dessas duas entidades. Como cada instância pode possuir vários números e esses podem ser compartilhados, o relacionamento de `Telefones` com `Pessoas` e `Empresas` possui cardinalidade M:N para ambos, fechando o bloco de relações societárias.

A ligação das `Empresas` com `Licitações` é feita pela participação das mesmas nos `Itens_Licitações`. Para promover a concorrência, várias `Empresas` podem participar de vários itens de licitação, seja do mesmo processo ou não, em um relacionamento M:N. Já `Itens_Licitaçõe` estão relacionados hierarquicamente com `Licitações`, sendo único para o processo de licitação, o qual pode possuir vários itens em um relacionamento 1:N.

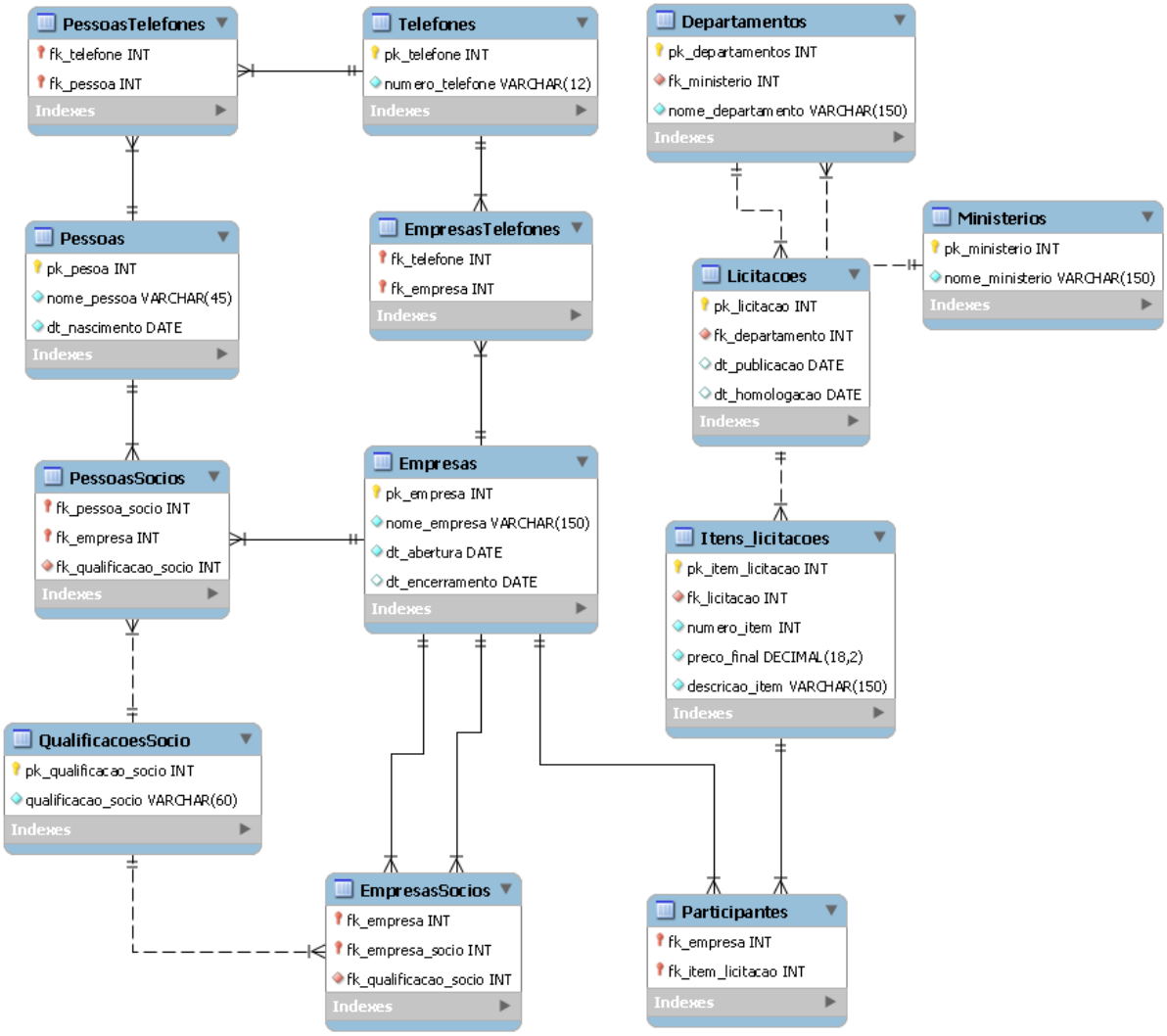


Figura 4.1: Modelo relacional para o estudo de caso com sociedades e licitações.

Cada processo de licitação deve ser vinculado a uma instância de `Departamentos`, responsável pela aquisição, e o mesmo está vinculado a uma instância de `Ministérios`, sendo os dois relacionamentos também 1:N, conforme representado no diagrama.

Depois de realizada a licitação, a melhor proposta é homologada e a empresa ganhadora assina o contrato com a administração pública. Essa homologação pode ser assinada por um servidor diferente do responsável pela licitação. Essas relações serão incluídas posteriormente para se verificar a flexibilidade na alteração dos modelos, assim como a de contadores e seu histórico com as empresas.

4.3 Modelos de Dados Baseado em Grafos para o Banco de Vínculos Simplificado para Licitações e Sociedades

Nesta seção é aplicada o MDG-NoSQL para modelar as relações de sociedades nas estruturas de grafos simples, de atributo, hipergrafo e aninhado, mas apresentando o modelo completo apenas para o grafo de atributos. Isso porque, como o grafo de atributos não é tão complexo quanto o hipergrafo e o grafo aninhado, mas também não tão rígido quanto o grafo simples, se apresenta como um bom ponto de partida para se elaborar considerações sobre a modelagem de dados em grafos, assim como uma comparação com o modelo relacional, do qual ele se aproxima, dependendo de como os atributos são tratados.

4.3.1 Grafo Simples

Os bancos de grafos simples utilizam apenas vértices e arestas cabendo aos rótulos armazenar os valores identificadores das instâncias. Dessa forma, os atributos se transformam também em nós, como se fossem entidades. O MDG-NoSQL, quando utilizada para grafos simples, considera entidade, representada pelo seu atributo identificador, e os demais atributos, como vértices. A Figura 4.2 apresenta o relacionamento de sociedades do estudo de caso, modelado em um grafo simples.

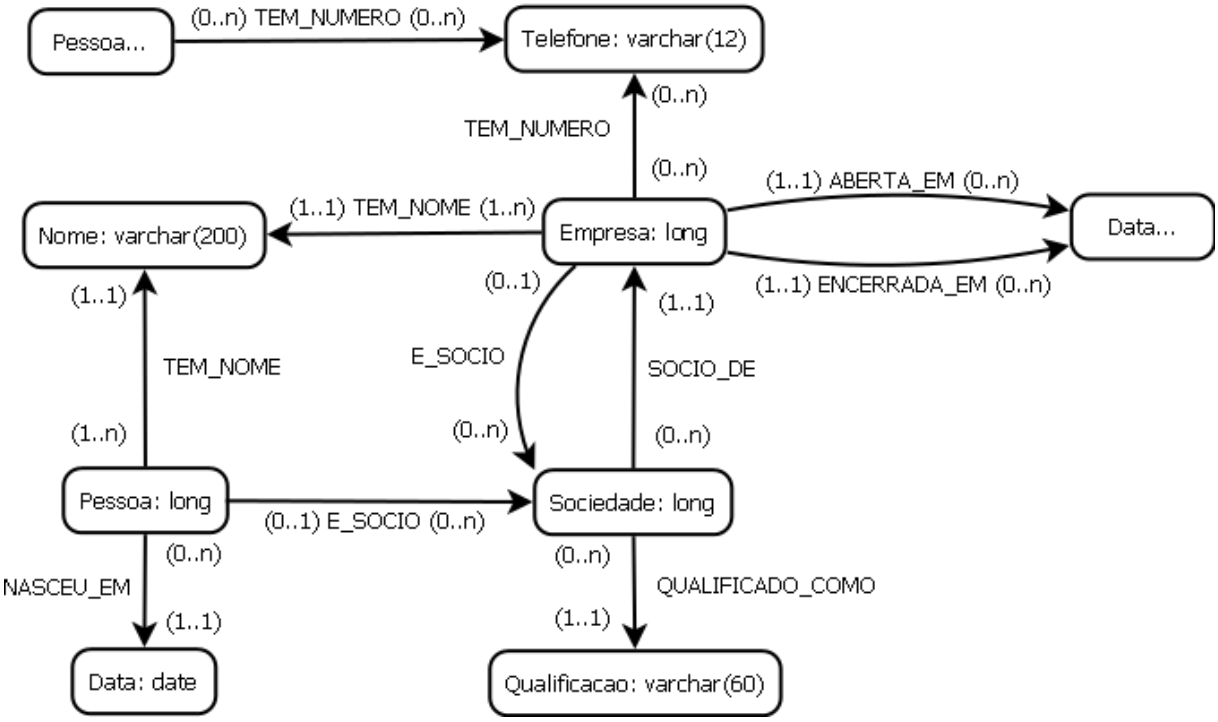


Figura 4.2: Estudo de caso modelado em um grafo simples.

Nota-se no modelo diversos atributos compartilhados como `Nome` e `Data`. Os relacionamentos entre de `Pessoa` e `Empresa` com `Nome` tem a mesma natureza e por isso usam o mesmo rótulo. Já a `Data` se relaciona de forma distinta, indicando o nascimento de uma `Pessoa`, ou as datas de abertura e encerramento de uma `Empresa`.

Já o relacionamento entre sócios e empresas, por envolver também as qualificações, é representado de forma n-ária pelo vértice `Sociedade`, que identifica cada sócio, juntamente com sua `Qualificação`. Como uma `Empresa` pode ter sócios do tipo `Pessoa` ou `Empresa`, essas entidades podem aparecer ligadas a várias `Sociedades`, mas cada `Sociedade` terá apenas um sócio, seja `Pessoa` ou `Empresa`.

As relações M:N aparecem de forma simplificada no modelo, como no caso de `Telefone` com `Pessoa` e `Empresa`. Como pode-se ligar vários vértices entre si, não é necessário qualquer entidade entre eles para estabelecer esse relacionamento, contrastando com o modelo relacional, onde seria necessário uma tabela associativa, como apresentado na Figura 4.1.

Algumas consequências de se ter todos os atributos também como vértices é que o modelo se torna mais normalizado, pois qualquer atributo que poderia ser duplicado é compartilhado entre as demais entidades, mantendo-se apenas uma instância. Essa característica também aumenta a quantidade de conexões entre elas por enfatizar os relacionamentos com valores de atributos em comum. Em contra partida, a quantidade de elementos aumenta juntamente com o número de atributos e valores distintos no modelo, deixando as consultas mais onerosas, pois deve-se buscar os vértices ligados a uma instância quando for necessário verificar algum atributo.

4.3.2 Grafo de Atributos

Os grafos de atributos permitem que tanto os vértices como as arestas armazenem informações de atributos, como um par de chave e valor. Essa característica permite que certas informações sejam armazenadas próximas aos nós, facilitando a recuperação do dado, além de simplificar a diagramação do modelo, pois são necessários menos vértices, caso alguns atributos não sejam utilizados para expressar relacionamentos entre as instâncias.

A Figura 4.3 apresenta o estudo de caso completo utilizando um grafo de atributos. Percebe-se uma diminuição nos vértices de entidades, simplificando o modelo. Cada nó internaliza seus atributos e indica seus tipos. Sendo assim, informações como `Nome` e `Data` passam a integrar a estrutura do vértice ao invés de se tornarem outro nó.

Em relação às entidades no processos de licitação, observa-se o encadeamento da hierarquia entre eles como em `Ministério` e `Departamento`.

Um `Ministério` com várias instâncias de `Departamento` que pode publicar diversos processos de `Licitação`, composta por uma ou mais instância de `Item_Licitação`. Já nessa última entidade, o relacionamento entre ela e `Empresa` merece um destaque, comparado com o modelo relacional.

No modelo relacional, o relacionamento M:N de `Empresa` com `Item_Licitação` é modelado utilizando uma tabela associativa onde são armazenadas as diversas combinações dessas duas entidades. Já no grafo, esse relacionamento binário é implementado simplesmente pelo uso das arestas, sem necessidade de algum tipo de vértice intermediário. Essa característica apresenta uma maior flexibilidade nos relacionamentos utilizando grafos, além de simplificar o modelo.

Já os relacionamentos entre os vértices `Pessoa`, `Empresa`, `Qualificação` e `Sociedade` continuam parecidos com a do modelo simples, apesar de agora alguns nós estarem mais semelhantes com tuplas de tabelas. De fato, o uso de atributos pode trazer o diagrama para mais próximo de um modelo relacional.

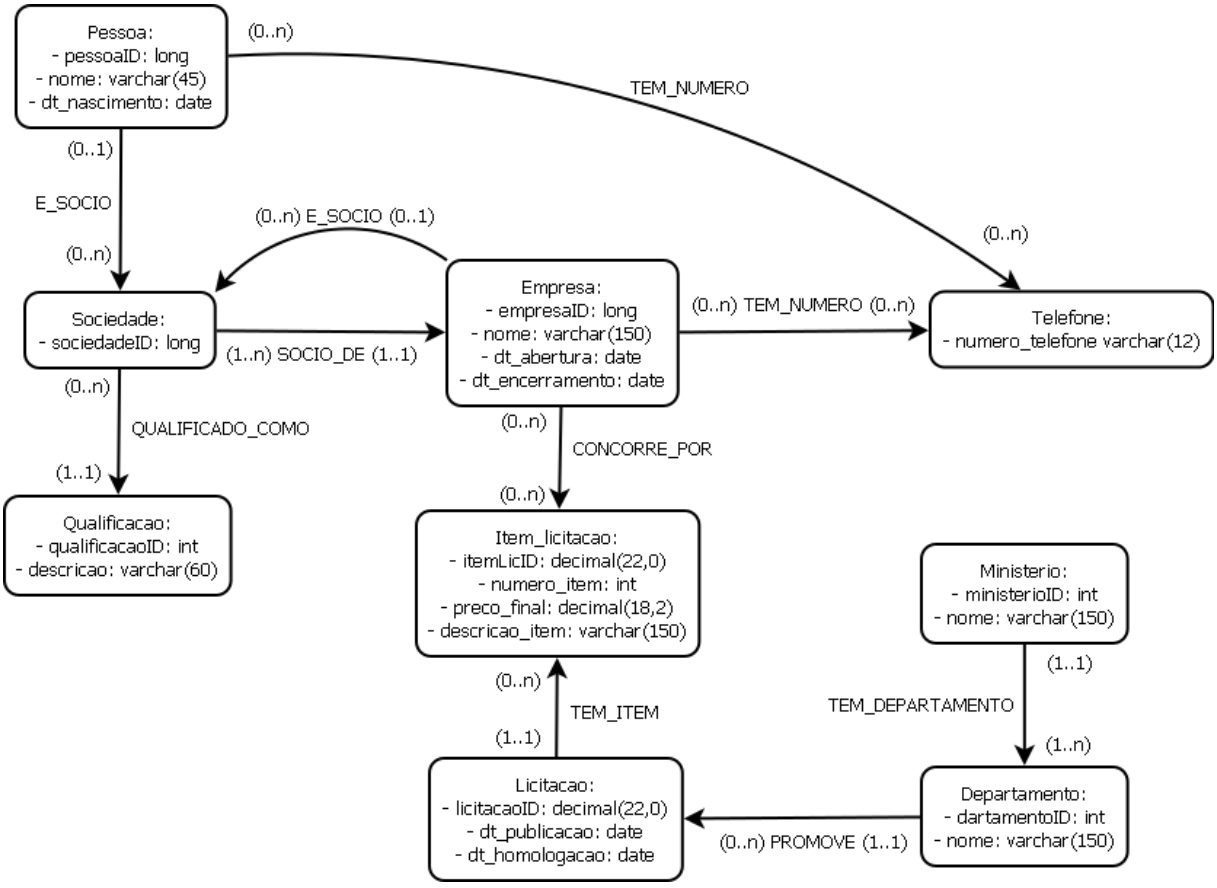


Figura 4.3: Estudo de caso modelado em um grafo de atributos.

Essa semelhança gera uma armadilha ao se modelar utilizando grafos de atributos. Isso porque, se todos os atributos forem mantidos juntos aos nós ou arestas, pode-se

perder o benefício de enfatizar os relacionamentos. O `Nome`, por exemplo, que costumam ser colocados como campos nas tabelas, em determinada situação poderiam servir para identificar pessoas homônimas ou mesmo parentesco utilizando os nomes de família. Nesse caso, mantê-los como vértices seria mais vantajoso que como atributo, pois não seria necessário varrer todos os vértices em busca dos nomes semelhantes, mas apenas localizando o nó indexado com o nome desejado e recuperando seus vértices adjacentes. Tratá-lo como vértice também facilita criar novos vínculos baseados em nomes de família, pois seria necessário apenas adicionar uma aresta entre dois nomes similares.

Em outro cenário, onde cada `Empresa` inicialmente tivesse apenas um `Telefone`, poder-se-ia mantê-los juntos aos vértices, mas em uma auditoria, ao se perguntar quais empresas declararam o mesmo telefone, seria necessário varrer os nós como se fossem tuplas em uma tabela de um banco relacional. A compreensão das necessidades do negócio no momento de decidir quais atributos devem ser mantidos nos vértices ou explicitados como vértices é fundamental para um melhor proveito da estrutura de grafos.

Para apresentar um cenário diferente aos casos de `Nome` e `Telefone`, onde manter o atributo em uma aresta é melhor que torná-la um vértice, basta analisar a entidade `Qualificação` do ponto de vista do trabalho de auditoria. Ela é uma situação onde trazer atributos, ou até mesmo entidades, para dentro de vértices e arestas, pode beneficiar as consultas.

O relacionamento de sociedade foi construído de forma n-ária devido à necessidade de indicar qual a `Qualificação` do sócio na `Empresa`. Entretanto, do ponto de vista do contexto de identificação de fraudes, e observando-se os caminhos possíveis para o grafo, percebe-se que a `Qualificação` não contribui muito no modelo, estando separada mais por uma questão de normalização. A Figura 4.4 apresenta um conjunto de instâncias para duas sociedades distintas que possuem a mesma `Qualificação`. Para esse exemplo, será considerada uma situação com caminhos bidirecionais.

Observa-se na Figura 4.4 que existe um caminho entre a `Pessoa A` e `Pessoa B` através da `Qualificação` Cotista. O mesmo acontece para a `Empresa X` e a `Empresa Y` e novamente entre a `Pessoa A` com `Empresa Y` e `Pessoa B` com `Empresa X`. Esse exemplo está de acordo com o modelo que evidencia o relacionamento entre sócios que possuem a mesma qualificação e simplifica uma busca pelas sociedades, como as do tipo `Cotista`. Porém, a forma como foi modelado o problema aumenta a complexidade do grafo com um maior quantidade de vértices intermediários e que não traz uma informação realmente valiosa para o objetivo da busca de fraudes.

Ao invés disso, essa modelagem pode inundar com dados irrelevantes quem procurar por vínculos entre empresas e pessoas. Isso porque o fato de duas pessoas serem sócias de empresas diferentes como cotistas é algo muito comum. Dificilmente ambas as pessoas

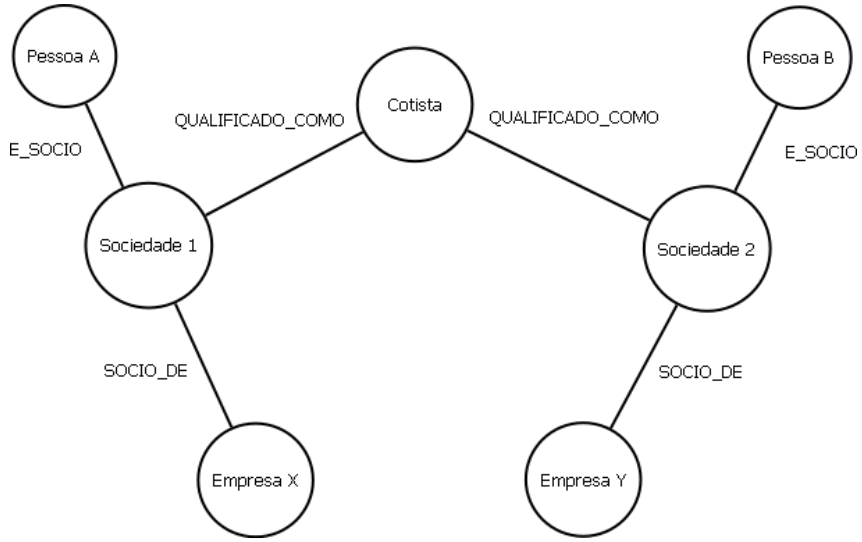


Figura 4.4: Estudo de caso modelado em um grafo de atributos.

serão conhecidas uma da outra e, portanto, dificilmente estarão em conluio para fraudar um processo de licitação. Da forma como se está modelado, apesar de separar a entidade `Qualificação` e evidenciar seus relacionamentos, se forem consultados os caminhos entre `Pessoa` ou `Empresa` para identificar vínculos, muitos relacionamentos serão retornados para os casos em que os pares consultados terão apenas a `Qualificação` em comum, e este ainda poderá ser o menor caminho entre eles. Isso significa que se for realizada uma consulta apenas pelos caminhos mais curtos, serão apresentados somente os que passam por `Qualificação`, deixando de fora outros mais relevantes, como `Sócio`, por exemplo. Além disso, provavelmente não serão comuns consultas como recuperar todas as empresas com determinada qualificação, mas sim para verificar as qualificações dos sócios de uma empresa. Dessa forma, a informação de `Qualificação` é mais importante para se filtrar uma `Sociedade` do que para localizar um vínculo entre instâncias distintas de `Pessoa` ou `Empresa`.

Essa característica da `Qualificação` sugere que será mais eficiente, visando as consultas por vínculos na identificação de fraudes, mover essa informação para um atributo em uma aresta de sociedade, ao invés de manter o vértice intermediário entre sócio e empresa. Essa alteração é apresentada na Figura 4.5, onde o modelo de grafos de atributos da Figura 4.3 foi alterado para remover o vértice do relacionamento n-ário por meio da inclusão da `Qualificação` como atributo da aresta `SOCIO_DE`.

Observa-se, no novo modelo, que agora existe uma repetição das qualificações para as arestas `SOCIO_DE`, mas o relacionamento entre os sócios e empresas tornou-se direto, sendo necessários menos saltos para se chegar de um sócio a outro pelos relacionamentos realmente relevantes. Também foi removido o caminho entre `Pessoa` e `Empresa` criado por `Qualificação`, que não trazia uma informação de vínculo útil, deixando apenas os

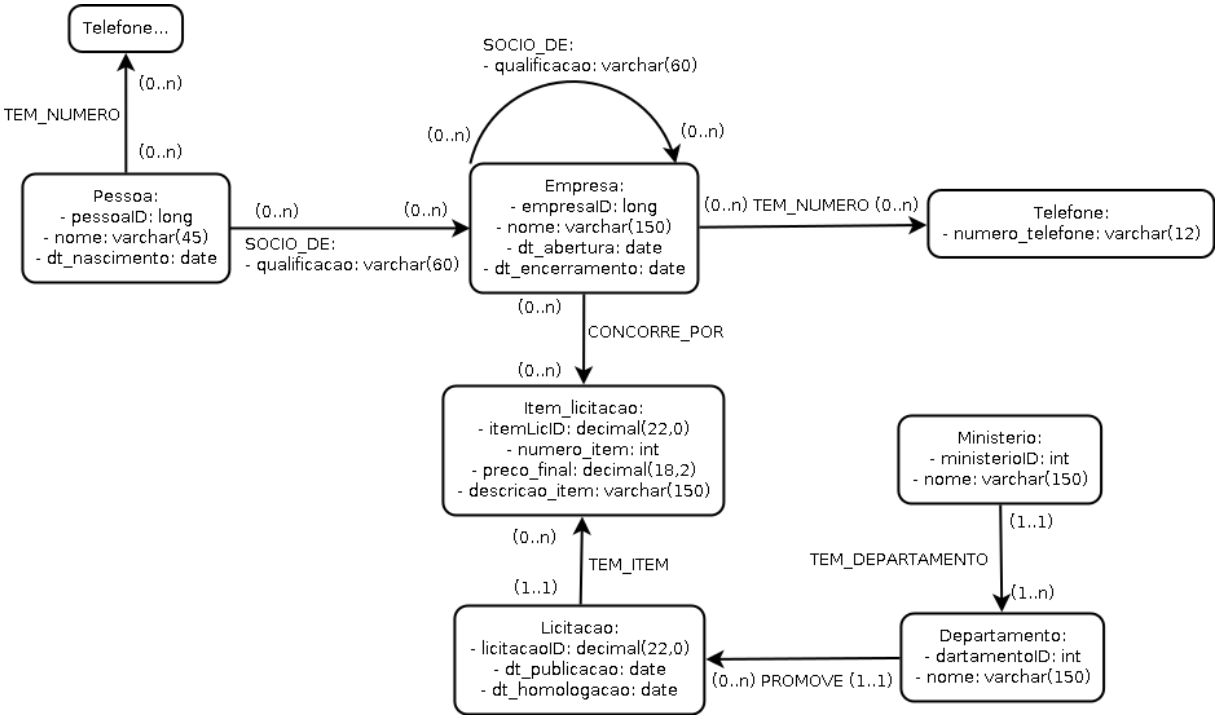


Figura 4.5: Estudo de caso modelado em um grafo de atributos com Qualificação como atributo.

caminhos que realmente podem levar a um relacionamento suspeito entre essas duas entidades.

Por fim, no modelo relacional não se pode realizar essa alteração porque o relacionamento M:N de `Pessoa` e `Empresa` impõe a tabela intermediaria similar ao vértice n-ário de `Sociedade`, mas no grafo ele pode ser evitado caso não exista uma terceira entidade. Outro ponto a ressaltar é que um vértice a mais no grafo pode gerar novos caminhos, já no caso da tabela `QualificaçõesSocio` no modelo relacional, ela apenas normalizou a informação dentro da relação de sociedade, mantendo como caminho ainda as tabelas `PessoasSocios` e `EmpresasSocios`.

4.3.3 Hipergrafos e Hiperarestas

Um hipergrafo permite que suas arestas liguem mais de dois vértices. Ou seja, a relação entre seus vértices não precisa ser binária. Isso permite que em certas situações onde o mesmo relacionamento ocorre entre várias instâncias seja utilizado apenas um arco. A Figura 4.6 apresenta os relacionamentos de sociedade utilizando um hipergrafo simples, ou seja, sem atributos.

Percebe-se de imediato no modelo que o relacionamento de sociedade no hipergrafo foi internalizado na hiperaresta `SOCIO_DE`. Isso porque, como foi visto no Capítulo 3, alguns vínculos de natureza n-ária podem ser representados com a hiperaresta, como

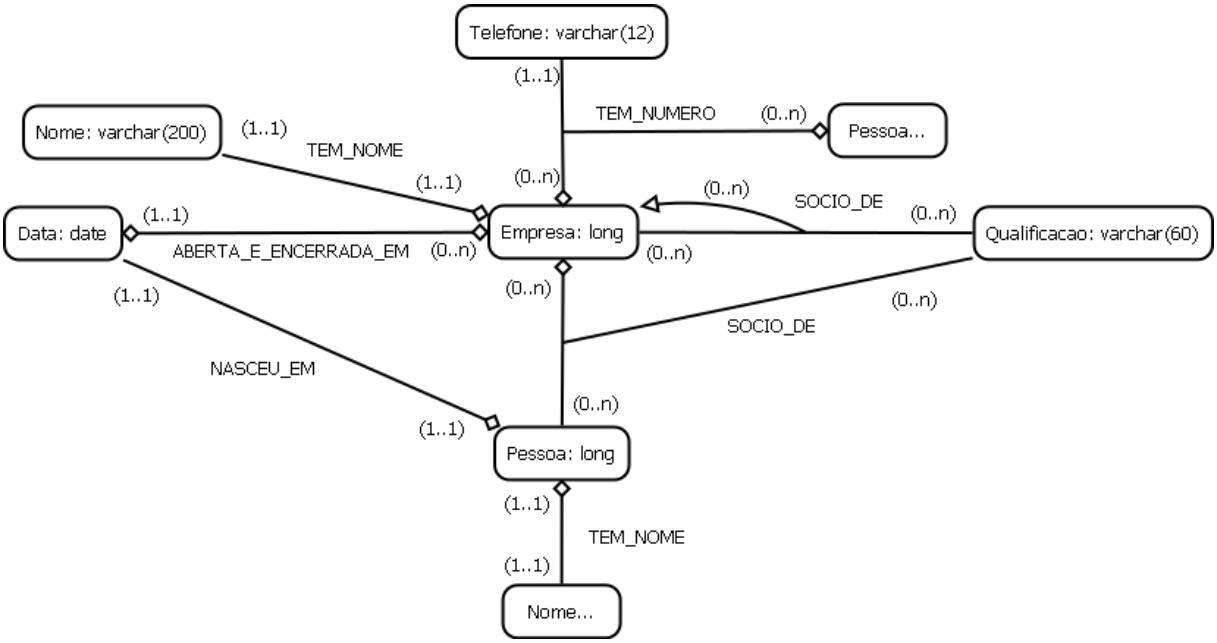


Figura 4.6: Estudo de caso modelado em um hipergrafo sem atributos.

no caso do relacionamento de sociedades. Entretanto, a hiperaresta de sociedades entre as instâncias de `Empresa` precisam ser direcionada para distinguir o sócio na origem de suas empresas com a mesma `Qualificação`. Dessa forma, se uma `Empresa` possuir seis empresas distintas, onde em quatro delas ela é qualificada como administrador e nas outras duas como cotista, existirão duas hiperarestas no banco de dados em grafos. A primeira terá a `Empresa` sócia na origem junto com a `Qualificação` de administrador e as quatro empresas no conjunto de destino. Já na segunda hiperaresta, a origem conterá a sócia novamente, mas com a `Qualificação` de cotista, além de duas outras empresas no conjunto de destino. Ressalta-se que a hiperaresta direcionada não é necessária para sócios do tipo `Pessoa`, já que é possível separar quem é o sócio e quais são as empresas.

Outro relacionamento que pode parecer incomum é o apresentado para `Telefone`, `Pessoa` e `Empresa`. Nesse caso, como o `Telefone` é relevante para consultar vínculos entre essas duas entidades, a hiperaresta foi modelada para aceitar apenas um `Telefone`, e as diversas instâncias de `Pessoa` e `Empresa` que compartilhem o mesmo número. Sendo assim, cada número conterá apenas uma hiperaresta com todas as pessoas e empresas que o utilizem.

Já o vértice da entidade `Nome`, de forma diferente do `Telefone`, aparece em hiperarestas separadas, apesar de ser compartilhado por `Pessoa` e `Empresa`. Isso é motivado porque os nomes de empresas dificilmente são iguais aos das pessoas, e por isso a maioria das interseções seriam vazias. Ainda assim, cada `Nome` mantém em sua hiperaresta a referência de todos os vértices homônimos, e uma `Pessoa` ou `Empresa` estará indicada em apenas

um relacionamento com `Nome`. Esse caso também mostra a diferença entre o diamante, indicando múltiplas instâncias de `Empresa` ou `Pessoa` na hiperaresta, e a cardinalidade, indicada como (1..1) junto às entidades, pois cada instância delas só podem aparecer em uma hiperaresta desse relacionamento.

Para concluir os vínculos das entidades de sociedade, tem-se o relacionamento entre `Data` e `Empresa`, que é apresentado no modelo como uma hiperaresta sem direção com múltiplas instâncias em ambas as entidades. Ela é possível nesse modelo, sem precisar utilizar uma hiperaresta direcionada para separar as datas de abertura e encerramento, porque ambas tratam de eventos sequenciais. Não se pode ter uma data de encerramento antes da abertura da `Empresa`. Sendo assim, o modelo apresenta um conjunto na hiperaresta com todas as instâncias de `Empresa` que possuem datas em comum, e possível de serem separadas, sabendo-se que a menor data se refere à abertura.

Por fim, o fato de existirem hiperarestas não exclui a possibilidade da estrutura também possuir atributos. A Figura 4.7 apresenta uma modificação no modelo para exemplificar um hipergrafo que possua atributos. Para ilustrar os atributos no relacionamento, o modelo também incluiu a `Qualificação` na hiperaresta `SOCIO_DE`, como já apresentado no grafo de atributos.

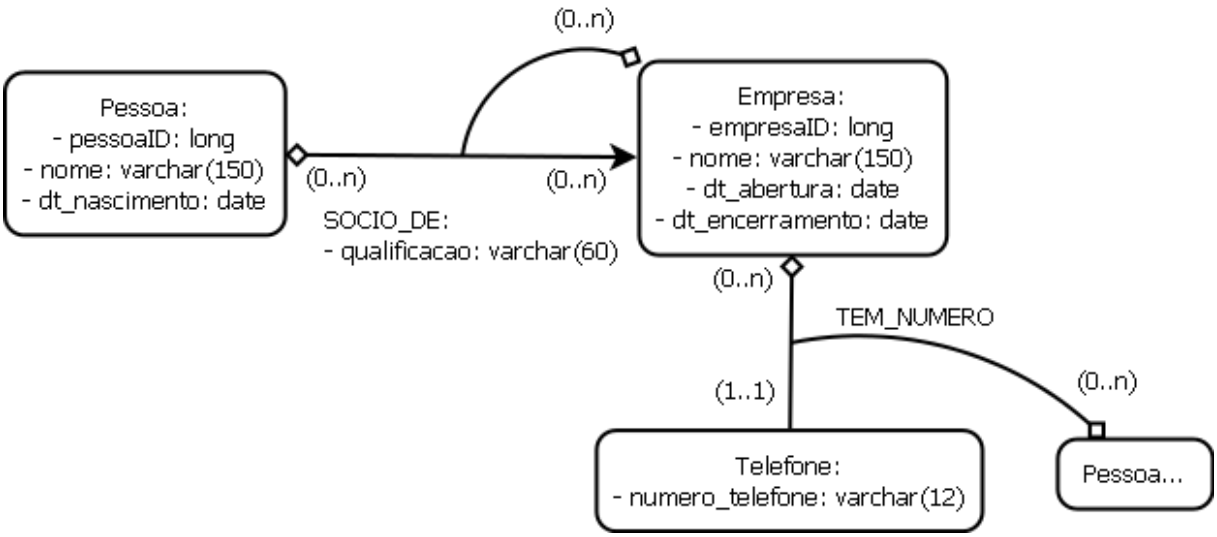


Figura 4.7: Estudo de caso modelado em um hipergrafo com atributos.

Novamente, o uso de atributos simplifica a estrutura do modelo, mas deve-se realizar a análise de quais atributos precisam ser enfatizados ou não, de forma parecida com a realizada no grafo de atributos, além de como serão organizados os subconjuntos das hiperarestas. O modelo da Figura 4.7, por exemplo, foi alterado para que a hiperaresta `SOCIO_DE` contenha uma origem com todos os sócios, `Empresa` ou `Pessoa`, com a mesma `Qualificação` e uma instância apenas de `Empresa` no conjunto de destino.

4.3.4 Grafo Aninhado e Hipervértice

Mais complexo que os demais, o grafo aninhado utiliza a estrutura de hipervértice que representa um subgrafo. Dessa forma, pode-se agrupar vários vértices e hipervértices, dando a eles significados mais amplos, como `Pessoa`, `QuadroSocietario` e `Empresa`, conforme é apresentado na Figura 4.8.

Para esse modelo, `Pessoa` e `Empresa` são construídos como hipervértices. No caso de `Pessoa`, foram utilizados os vértices de `Nome` e `Data`, referente ao nascimento. Esse hipervértice indica que apenas uma subgrafo comporá sua estrutura, e o rótulo define que o tipo que identificará o hipervértice será um inteiro. Sendo assim, uma instância de `Pessoa` será um hipervértice, com `Nome` e `Data`, identificado por um valor inteiro.

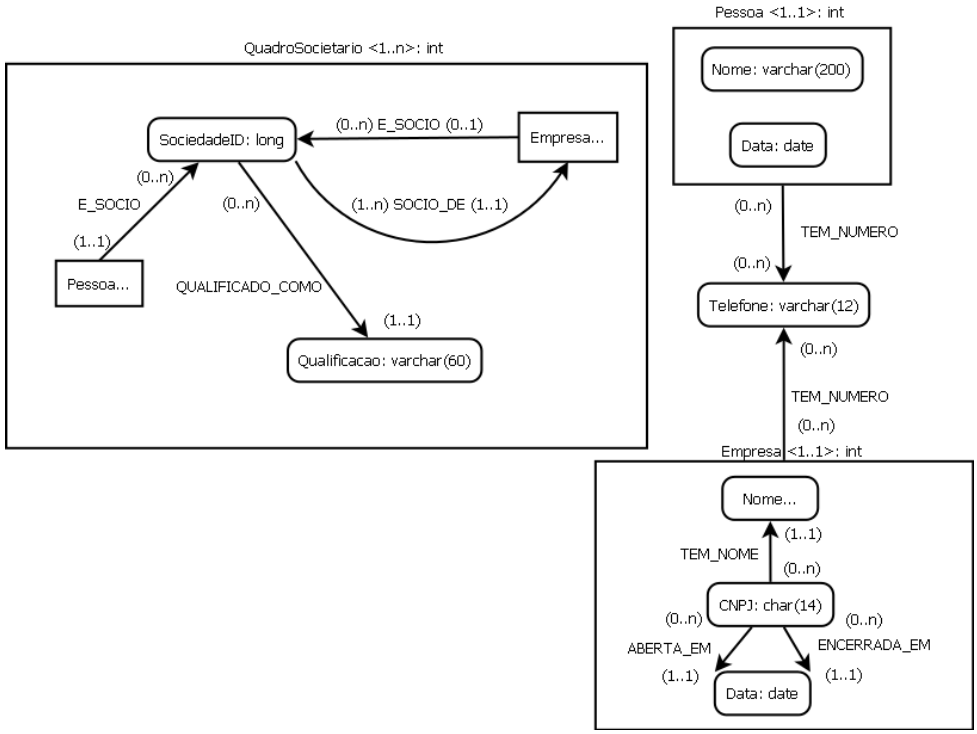


Figura 4.8: Estudo de caso modelado em um grafo aninhado sem atributos.

Para a entidade `Empresa`, o hipervértice é construído a partir do número de `CNPJ`, ligados ao `Nome` e `Data`, para indicar as datas de abertura e encerramento. O hipervértice também utiliza apenas um subgrafo com essa topologia, assim como um número inteiro como identificador.

Com as estruturas de hipervértice para `Pessoa` e `Empresa` definidas, pode-se ligá-las como se fossem vértices comuns de um grafo. Sendo assim, ambas as entidades foram vinculadas ao vértice de `Telefone`, novamente permitindo que se identifique pessoas e empresas com o mesmo número. O relacionamento de sociedade também pode

ser construído utilizando essas estruturas, como mostra a topologia do hipervértice de `QuadroSocietario`.

A função do `QuadroSocietario` é agrupar todos os subgrafos que descrevem um relacionamento de sociedade vinculados a mesma `Empresa`, indicado pelo valor do agrupamento no rótulo dessa entidade. Dessa forma, a instância de `QuadroSocietario` terá um identificador único e todos os sócios, sejam de `Pessoa` ou `Empresa`, que compõem o quadro societário da companhia descrita.

4.4 Análise do Modelo de Dados para Bancos NoSQL Baseado em Grafos

O MDG-NoSQL foi idealizado para cobrir uma lacuna na modelagem de bancos em grafos, que apesar de possuir diversas propostas na literatura, nenhuma buscou atender a abstração dos conceitos de grafo, mas focou em soluções específicas de suas iniciativas. Dessa forma, o MDG-NoSQL foi elaborado com o objetivo de ser simples, generalista, abrangente e construído a partir de símbolos encontrados em softwares especializados em diagramas, como o DIA[1], este, inclusive, sendo uma ferramenta livre e utilizado para desenhar os modelos deste trabalho. Resumidamente, pode-se descrever o MDG-NoSQL pelo seguinte conjunto de características:

- Possui uma notação para uso em grafos de características simples, com atributos, hipergrafo e grafo aninhado (hipervértice);
- Entidades e relacionamentos indicadas por rótulos nos vértices e arestas;
- Suporte à tipagem para atributos e rótulos nos vértices e arestas, além de notação específica para peso;
- Suporte à cardinalidade.

Assim como no modelo relacional, o MDG-NoSQL permite avaliar diversos modelos antes de iniciar uma implementação, dando ênfase aos esquemas dos dados ao invés de instâncias. Ao se definir um esquema inicial, pode-se questionar quais as consequências de organizar a informação da forma modelada e buscar ajustar o modelo para chegar a uma decisão de projeto final para ser apresentada.

Embora a notação seja voltada para esquemas, isso não exclui a possibilidade de se exemplificar utilizando grafos com instância, como utilizado diversas vezes no detalhamento da notação do Capítulo 3. Mas a ênfase no esquema é importante para abstrair o modelo para um nível mais general, e não apenas uma situação específica.

[1] http://dia-installer.de/

Dessa forma, apesar dos diagramas serem grafos, o MDG-NoSQL pode ser considerado um "metagrafo", ou seja, um grafo que fala sobre outro grafo, sendo o primeiro a abstração descrita com o MDG-NoSQL, e o segundo sua implementação em um banco de dados em grafo. Isso significa que os vértices e arestas da notação definirão as regras do que será inserido no banco físico, o que inclui tipagem dos dados, tanto dos rótulos como de atributos, quando existentes. Ele também permite entender como os caminhos no grafo poderão se formar, já que especifica quais entidades estão conectadas e em qual direção.

Outra característica do MDG-NoSQL é o uso da cardinalidade já utilizada no MER. Porém, nos grafos, ela indica qual a quantidade de arestas se espera que uma instância tenha para um relacionamento específico. Ou, detalhando de outra forma, pode-se imaginar a cardinalidade como um número esperado de instâncias de arestas de certo relacionamento com o identificador (ou chave primária) de uma determinada instância de vértice.

A construção do MDG-NoSQL, assim como sua aplicação sobre o estudo de caso, permitiu a elaboração de algumas considerações, principalmente em comparação com a modelagem relacional, que podem influenciar decisões importantes durante a elaboração do modelo de dados.

4.4.1 Vértices, Não Tabelas

No modelo relacional, as entidades são mapeadas em tabelas que agrupam seus atributos. Um desses atributos, a chave primária (*primary key* - *PK*), é um identificador único e não nulo que existe como atributo de uma entidade. Quando implementada, cada tupla pode ser identificada pela sua chave primária. Quando se fala de uma entidade `Pessoa`, por exemplo, tem-se a ideia, no modelo relacional, de uma tabela que conterá uma chave única, um nome e endereço em cada instância. Ao definir uma outra entidade chamada `Empresa`, essa também poderá materializar uma tabela com atributos semelhantes. Mesmo que o domínio das chaves seja o mesmo, não ocorrerá colisões, pois cada chave pertence a uma relação diferente.

Em um grafo, os tipos principais das instâncias armazenadas são nós ou arestas. Sendo assim, antes de uma instância no banco de grafos ser `Pessoa` ou `Empresa`, será um vértice ou aresta. Isso significa que deve-se ter um identificador único para cada instância, independente da entidade ao qual ela pertença, ou pode ser necessário adicionar mecanismos para garantir unicidade nos nós do grafo a ser modelado. A Figura 4.9 exemplifica essa ideia entre os dois tipos de modelagem. Observa-se que as instâncias no modelo relacional possuem o mesmo identificador, mas a forma de organizar os dados por tabela permite distingui-las.

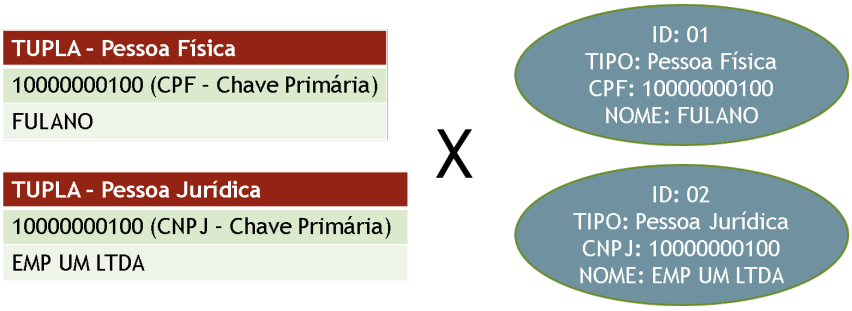


Figura 4.9: Comparativo entre tuplas de tabelas e vértices.

Na Figura 4.9, cada vértice é identificado pelo campo `ID` e não se pode trazer os dados simplesmente aproveitando as chaves para esse campo, pois as mesmas entrariam em conflito. Dessa forma, os valores de `ID` podem ser criados de forma sequencial, por exemplo, e colocar os campos que eram chave como atributos indexados dentro dos vértices.

Alguns bancos de grafos permitem utilizar os rótulos para separar os vértices como se fossem tabelas, fazendo com que um rótulo `Pessoa` em um vértice indique que este está no grupo dos nós de pessoas. Outros podem utilizar classes da linguagem orientada a objetos. De qualquer forma, ao se implementar um modelo, deve-se levar em consideração como as instâncias serão identificadas unicamente dentro do conjunto de vértices armazenadas no banco de grafos.

4.4.2 Grafos Enfatizam Relacionamentos

Grafos permitem modelar diversos problemas como ruas em cidades, fronteiras de países ou parentescos entre pessoas. Ligar dois vértices por uma aresta significa indicar esses relacionamentos. Porém, algumas vezes, esses relacionamentos podem não ficar explícitos devido ao modelo construído.

A Figura 4.10 apresenta um esquema do MDG-NoSQL e duas instâncias da entidade `PessoaJuridica`, onde o ano e o mês de abertura das empresas aparecem como um atributo. Estruturas de grafos, como a de atributo, permitem que outras informações além de uma chave ou rótulo sejam armazenadas junto com os vértices, mas algumas informações podem ser mais úteis se forem evidenciadas.

Por exemplo, para verificar as empresas que são abertas no mesmo ano e mês, pode-se varrer os vértices de forma similar ao que é feito nas linhas de uma tabela, mas com isso se desperdiça uma vantagem dos grafos de enfatizar esses relacionamentos.

Sendo assim, o modelo pode ser modificado para remover o atributo ano e mês, criando um vértice independente de `AnoMes`, que será ligado a `PessoaJuridica` por uma aresta indicando seu relacionamento, como apresentado na Figura 4.11. Isso faz com que a

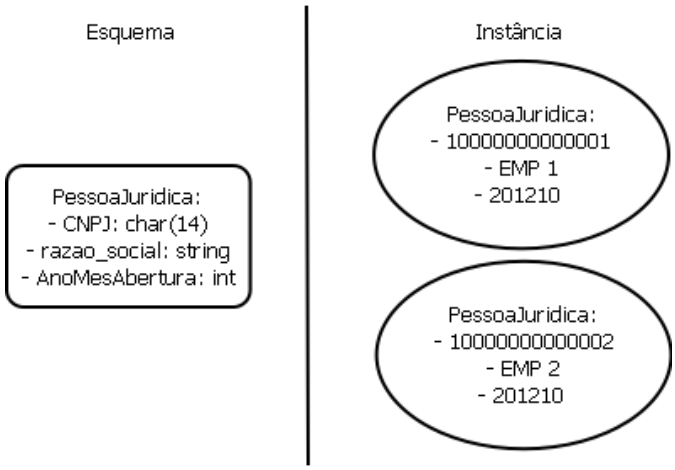


Figura 4.10: Exemplo de vértices com ano e mês da abertura como atributo.

ligação entre as duas instâncias pela data seja evidenciado, formando um caminho entre elas.

Figura 4.11: Grafos evidenciam relacionamentos utilizando atributos em comum.

Pode parecer estranho ter um conjunto de vértices no modelo exclusivo para informações como `Data`, ou mesmo `Nome`, já que o mais comum é que essas informações sejam armazenadas como atributos. Entretanto, como uma das vantagens dos grafos é enfatizar relacionamentos, pode ser importante evidenciar os atributos em comuns entre diversas outras entidades, além de diminuir a redundância de dados.

Dessa forma, ao se procurar por empresas que foram abertas nesse período, pode-se buscar o vértice de ano e mês desejado, assim como todas as empresas que possuem arestas apontando para esse vértice de `AnoMes`. Se for necessário verificar se existe um relacionamento entre elas por meio da data, também se pode caminhar entre elas por esse vértices (transformando o atributo em uma ponte). Porém, existem situações onde pode ser interessante realizar a operação inversa, ou seja, remover relacionamentos tirando a ênfase de alguns vínculos, como o realizado na Subseção 4.3.2.

4.4.3 Relações Binárias de M:N em Grafos são Diretas

Uma vez que os vértices podem se conectar a diversos outros nós, um grafo pode apresentar diretamente um relacionamento binário de M:N entre entidades. A Figura 4.12 apresenta um exemplo de relacionamento M:N entre `Empresa` e `Pessoa` representada por um grafo.

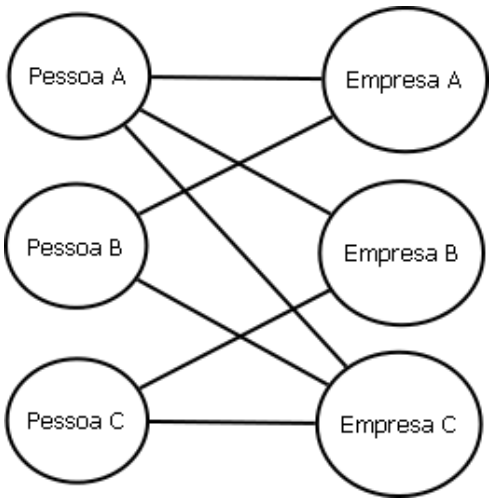


Figura 4.12: Exemplo de instâncias com relacionamento M:N em um grafo.

Nesse exemplo de instâncias para sociedade, várias pessoas podem possuir múltiplas empresas e estas também podem ter diversos sócios, como a `Pessoa A` que é sócia das empresas `A`, `B` e `C`. Já a `Empresa C` possui mais dois sócios, a `Pessoa B` e `Pessoa C`, enquanto a `Empresa B` tem como sócio a `Pessoa A` e `C`. Ou seja, os relacionamentos já estão representados pelo conjunto de arestas entre as instâncias.

Essa característica flexibiliza uma mudança na estrutura do esquema do banco de dados em grafos, caso um relacionamento passe de 1:N para M:N, pois significará apenas a permissão para que mais de uma aresta do mesmo tipo de relacionamento se conecte a outra entidade. Diferentemente do modelo relacional onde seria necessário criar mais uma tabela para representar o relacionamento M:N.

4.4.4 Grafos facilitam caminhadas entre os vértices

Por fim, entre as vantagens de se utilizar estruturas de grafos, percorrer caminhos entre as interligações das instâncias é uma das principais. Muito do estudo de grafos introduz diversos algoritmos e aplicações para realizar "essa caminhada" [10]. Como é apresentado na análise de resultados no Capítulo 5, é bem mais simples percorrer relacionamentos em um banco de dados em grafos do que no modelo relacional. Ao se construir uma

modelagem em grafos, a ideia de como os vértices podem ser percorridos deve sempre ser considerada.

Capítulo 5

Implementação do Modelo do Estudo de Caso

O presente capítulo utiliza uma implementação para análise comparativa entre consultas de bancos de grafo e bancos de dados relacionais utilizando-se de linguagens nativas de cada implementação. Além do modelo do Capítulo 4, foi implementado o de grafos de atributos da Figura 4.5 (Subseção 4.3.2). Após a execução e comparação dessas duas implementações, foi construído um modelo relacional para descrever um grafo de atributos utilizando tabelas, ou seja, um banco de dados relacional com uma tabela para armazenar vértices e outra para arestas, além de duas tabelas utilizadas para manter os atributos. Esse terceiro modelo permitiu criar uma abstração para implementar a parte de sociedades do modelo de grafos no banco de dados relacional e verificar o impacto sobre a consulta mais complexa de utilizar essa abstração.

5.1 Escopo

Em relação à escolha do modelo de grafos, a estrutura de grafo de atributos foi escolhidos por: atenderem às necessidades descritas no banco de vínculos de relacionamentos de sociedades e licitações; permitirem uma comparação mais direta com o modelo relacional; além de já existir um Sistema Gerenciador de Banco de Dados para grafos de atributos chamado Neo4j, que implementa o *Cypher*, uma linguagem declarativa especializada que pode ser comparada com SQL.

Dessa forma, foi preparado um ambiente composto por uma máquina de 64GB de RAM, 12 núcleos de 24 *threads* e 5TB de disco, utilizando o sistema operacional Linux CentOS 6 para a execução dos dois SGDBs, alternadamente. Para o modelo relacional, foi utilizado o banco MySQL e a linguagem padrão SQL. Para o banco de grafos, foi

escolhido o Neo4j, com uma linguagem também declarativa chamada Cypher orientada para consultas em grafos.

Como o objetivo do banco é identificar relacionamentos que evidenciem fraudes no processo de licitação, foram realizadas três consultas em cada um dos bancos. Como as fraudes nesses processos dependem de coluio entre empresas por meio dos seus sócios, as questões foram elaboradas para tentar identificar esses vínculos, conforme destacado em seguida.

- Questão 1: Quais empresas possuem o mesmo telefone?
- Questão 2: Quais itens de licitação duas empresas concorreram juntas?
- Questão 3: Como duas empresas se relacionam a partir de suas sociedades até o terceiro nível de empresa?

A primeira questão procura verificar se o contrato de telefone não está sendo compartilhado por mais de uma empresa. Caso seja, provavelmente os sócios das empresas se conhecem, ou são os mesmos, e podem utilizar ambas as empresas para forjar uma concorrência em uma licitação.

A Questão 2 procura identificar se duas empresas concorreram juntas em licitações. Essa consulta é uma continuação da questão anterior em uma investigação, pois se foram identificados sócios em comum, pode-se verificar se realmente estão participando de licitações em conjunto.

A última questão procura identificar relacionamentos entre os sócios. Caso não seja encontrado um vínculo por telefone, os sócios de uma empresa ainda podem estar relacionados a partir de empresas e sociedades intermediárias entre duas concorrentes em uma licitação. Esse tipo de mecanismo é utilizado para dificultar a identificação da prática de combinação de lances e preços antes do processo.

Por fim, após as consultas nos bancos com as modelagens selecionadas, foi implementado no banco relacional o modelo com tabelas de vértices e aresta para verificar como a terceira consulta, mais complexa, se comportaria em um modelo de grafos de atributos implementado sobre um banco de dados relacional. Por fim, foram feitas algumas considerações sobre a implementação e carga dos modelos utilizando dados reais.

5.2 Consultando Relacionamentos

O modelo relacional é consultado a partir da linguagem SQL (*Structured Query Language*). Porém, tanto a modelagem como a linguagem SQL foram idealizadas para trabalhar sobre tabelas. A questão que surge ao tentar realizar consultas típicas de grafos é se

ambos se comportam de forma aceitável para a manipulação dos dados. Pode-se tratar as duas primeiras consultas do escopo como travessias com apenas dois saltos da origem. Um relacionamento entre duas empresas a partir de um nó, como telefone. A terceira consulta já é uma travessia mais complexa entre os nós do grafo e busca atingir, a partir de certa empresa, outra pelas relações de sócios.

Para realizar as consultas, foram gerados dados fictícios para alimentar o modelo a partir de uma série de *scripts* implementados em SQL. Foram criadas 10 empresas e 10 pessoas assim como algumas centenas de itens de licitação. As relações foram em sua maioria randomizadas e apenas algumas delas foram inseridas para garantir a existência de alguns relacionamentos, como empresas ligadas até o terceiro nível de sociedade por empresas.

Para o comparativo com um banco de grafos, foi escolhido Neo4j. Ele é um banco que utiliza a estrutura de atributos e possui uma linguagem de consulta chamada *Cypher* [32]. Por possuir as características necessárias para a comparação com o modelo relacional, foi decidido por utilizá-lo para avaliar como uma consulta especializada em grafos pode beneficiar a busca de relacionamentos.

5.2.1 Questão 1: Quais empresas possuem o mesmo telefone?

Para obter esse relacionamento foi feita a junção (*JOIN*) entre a tabela associativa `EmpresasTelefones` e as duas tabelas de domínio de `Empresas` e `Telefones`. Pode-se observar que na Consulta 5.1 foram necessários alias nas tabelas de `Empresas`, definindo qual está a direita da projeção na consulta, e qual está a esquerda, ligadas pelo relacionamento de `Telefones`.

Consulta 5.1: Consulta de empresas que possuem o mesmo telefone.

```
1  SELECT
2    CL.nome_empresa as empresaA,
3    T.numero_telefone,
4    CR.nome_empresa as empresaB
5  FROM EmpresasTelefones AS CT01
6    INNER JOIN Empresas AS CL
7      ON CL.pk_empresa = CT01.fk_empresa
8    INNER JOIN Telefones AS T
9      ON T.pk_telefone = CT01.fk_telefone
10   INNER JOIN EmpresasTelefones AS CT02
11     ON CT02.fk_telefone = CT01.fk_telefone
12   INNER JOIN Empresas AS CR
13     ON CR.pk_empresa = CT02.fk_company
14   WHERE CT01.fk_empresa <> CT02.fk_empresa;
```

Também foi necessário um alias para juntar os `Telefones` em comum na linha 10. Isso porque a primeira referência para essa tabela já estava definida para uma empresa específica, ligando `EmpresaA–EmpresaTelefonesCT01–Telefone`. Esse alias liga `EmpresaTelefones-CT01` com `EmpresaTelefones-CT02` pelo número telefônico e permite ligar `EmpresaTelefones-CT02` com a `EmpresaB`, porém como é a mesma tabela, aparecerão linhas onde a empresa da esquerda será igual a da direita, o que pode ser evitado com a cláusula da linha 14. O resultado da consulta é apresentado na Figura 5.1.

Result Grid | Filter Rows: | Expo

company_name	telephone_number	company_name
COMPANY A	55550001	COMPANY G
COMPANY G	55550001	COMPANY A
COMPANY B	55550002	COMPANY D
COMPANY D	55550002	COMPANY B
COMPANY E	55550004	COMPANY H
COMPANY H	55550004	COMPANY E

Figura 5.1: Resultado da consulta de telefones no MR.

Observa-se que as relações estão duplicadas devido ao produto cartesiano de `Empresas` X `Empresas`. Quando a junção parte do lado esquerdo ela localiza o telefone em comum para a `Empresa A` com `G`, por exemplo. A medida que as outras empresas vão sendo avaliadas o relacionamento de `G` com `A` também aparece, duplicando a informação. Essas duplicações desaparecem se forem especificados a origem e o destino, conforme será apresentado na Subsecção 5.2.2.

Já no modelo de grafos, foi realizada a Consulta 5.2. Utilizando a linguagem Cypher, desenvolvida para o Neo4j, basta especificar a origem, nesse caso uma Empresa qualquer `empresaO`, e o destino também do tipo `Empresa`, `empresaD`, passando pelo relacionamento `TEM_NUMERO`.

No Cypher, se for necessário especificar o sentido do vínculo, basta utilizar os símbolos < e > como setas junto ao hífen no relacionamento. Ou seja, para a consulta dos telefones de uma `Empresa`, tem-se pelo modelo um relacionamento com sentido de `Empresa` para `Telefone`, e por isso utiliza-se a estrutura empresaO: Empresas -[r:TEM_NUMERO*1..2]-> telefoneD: Telefone. Isso implica em buscar esse padrão de grafo (`Empresa->Telefone`) e retornar as instâncias correspondentes. Caso fosse escrito com o sentido oposto `empresaO: Empresas <-[r:TEM_NUMERO*1..2]- telefoneD: Telefone`, não seria retornado qualquer `Telefone`, pois pelo modelo implementado da Figura 4.3, não existem arestas no sentido `Telefone->Empresa`.

Porém, o Cypher permite que se omita os símbolos de direção da aresta, considerando ambos os sentidos do relacionamento. Isso permite utilizar de forma simples o `Telefone` como uma ponte para se chegar a outra `Empresa` que o compartilhe, transformando o relacionamento em bidirecional.

Consulta 5.2: Busca de empresas relacionadas por telefone utilizando o Cypher.

```
MATCH (empresaO:Empresas) -[r:TEM_NUMERO*1..2]- (empresaD:Empresas)
RETURN empresaO, r, empresaD
```

No relacionamento também foi especificado o número de saltos (*hops*) que poderiam ser avaliados para se chegar a uma empresa. Como temos apenas telefone entre elas para esse vínculo, serão dois saltos entre uma empresa e outra, indicada na consulta por *1..2.

Por fim, o resultado é retornado através da cláusula *RETURN*, indicando que devem ser obtidos as duas empresas e os relacionamentos entre elas. O resultado final é apresentado na Figura 5.2.

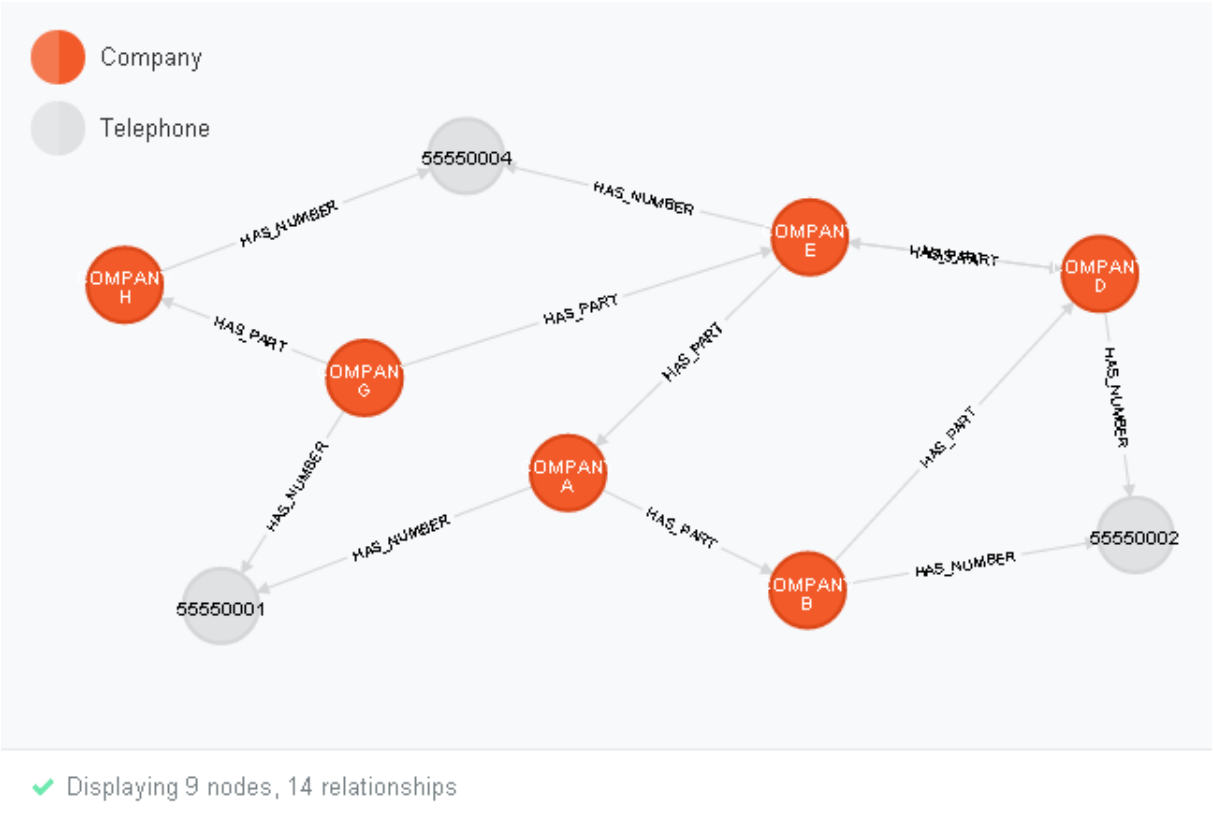


Figura 5.2: Resultado da consulta para empresas por telefones no Neo4j.

Como foi retornado o grafo de empresas que se relacionam por telefones, a interface apresenta tanto os vínculos de empresas por telefone, como as sociedades que existem entre

essas empresas. Isso porque o banco retornou para a interface de consulta, as empresas e telefones correspondentes ao padrão especificado (`Empresa->Telefone<-Empresa`), assim como os demais relacionamentos que essas instâncias possuem entre si, como no caso de sociedade. Isso não significa que existam apenas esses sócios, mas que para essas empresas retornadas, algumas são sócias e compartilham o telefone também.

Como os bancos de grafos já trabalham com essa estrutura de vértices e arestas por padrão, buscas voltadas para encontrar caminhos nos relacionamentos se tornam mais simples. Esse exemplo já demonstra como a consulta especializada em grafos pode facilitar o processo de busca de vínculos como os entre empresas.

5.2.2 Questão 2: Quais itens de licitação duas empresas concorreram juntas?

A Consulta 5.3 é parecida com a Consulta 5.2. Porém, nesse caso, deseja-se todos os itens no qual duas instâncias de `Empresas` participaram na `Licitação`. Essa consulta é útil porque se elas tiverem algum relacionamento como por telefone ou sócios, a participação em conjunto pode indicar uma tentativa de fraudar a licitação utilizando preços previamente acertados, prejudicando a concorrência e a lisura do processo.

Limitando para duas empresas específicas, estabelece-se uma origem e destino. Essa restrição é apresentada na linha 18 vindo após a clausula *WHERE*. Como foi definido o sentido da relação, as linhas da tabela não serão duplicadas com o caminho inverso.

Na Consulta 5.3 seria possível apenas realizar a junção entre as relações utilizando a tabela associativa, ordenando-a por `Licitação` e `Item_Licitação` para observar quais empresas aparecem juntas. Porém, para uma maior quantidade de dados, esse trabalho seria oneroso e algum processamento adicional seria necessário para entregar a informação final.

Como o objetivo nessa etapa é utilizar o modelo relacional para apresentar os vínculos das entidades, as consultas foram direcionadas na tentativa de entregar um resultado o mais próximo possível do desejado com a menor necessidade de processamento adicional dos dados obtidos.

Consulta 5.3: Consulta de empresas que concorreram a licitações.

```
1  SELECT
2    CL.nome_empresa as empresaB,
3    pk_item_licitacao,
4    fk_licitacao,
5    numero_item,
6    descricao_item,
7    CR.nome_empresa as empresaC
```

```
8  FROM Participantes AS A01
9  INNER JOIN Empresas AS CL
10     ON CL.pk_empresa = A01.fk_empresa
11   INNER JOIN Itens_licitacoes AS PI
12     ON PI.pk_item_licitacao = A01.fk_item_licitacao
13   INNER JOIN Participantes AS A02
14     ON A02.fk_item_licitacao = A01.fk_item_licitacao
15   INNER JOIN Empresas AS CR
16    ON CR.pk_empresa = A02.fk_empresa
17   WHERE A01.fk_empresa <> A02.fk_empresa
18    AND CL.nome_empresa = 'EMPRESA B' AND CR.nome_empresa = 'EMPRESA C';
```

A Figura 5.3 apresenta o resultado parcial da consulta para a `Empresa B` e `C`. Como definimos a origem e destino, o caminho é feito da `Empresa B` para a `C` identificando quais itens de licitação ambas participaram.

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

company_name	pk_procurement_item	fk_procurement	num_item	final_price	item_description	company_name
COMPANY B	3	1	3	83969.93	ITEM 49	COMPANY C
COMPANY B	4	2	1	8687.10	ITEM 28	COMPANY C
COMPANY B	10	4	1	44499.09	ITEM 7	COMPANY C
COMPANY B	12	4	3	77229.11	ITEM 28	COMPANY C
COMPANY B	13	5	1	18945.80	ITEM 20	COMPANY C
COMPANY B	25	9	1	34408.87	ITEM 50	COMPANY C
COMPANY B	27	9	3	45625.18	ITEM 46	COMPANY C
COMPANY B	28	10	1	6983.86	ITEM 32	COMPANY C
COMPANY B	32	11	2	2759.65	ITEM 7	COMPANY C
COMPANY B	35	12	2	76943.44	ITEM 27	COMPANY C
COMPANY B	40	14	1	75180.36	ITEM 37	COMPANY C
COMPANY B	42	14	3	36507.87	ITEM 1	COMPANY C
COMPANY B	43	15	1	71626.77	ITEM 49	COMPANY C
COMPANY B	47	16	2	70231.13	ITEM 1	COMPANY C
COMPANY B	52	18	1	41359.54	ITEM 11	COMPANY C
COMPANY B	53	18	2	60167.86	ITEM 36	COMPANY C

Figura 5.3: Resultado da consulta de licitações no MR.

Partindo para o modelo de grafos, similarmente ao que foi visto na Questão 1 para `Telefone`, temos um conjunto de elementos que podem vincular duas outras instâncias diretamente. Utilizando novamente a `Empresa B` e `C`, deseja-se verificar quais caminhos existem entre elas, passando por um item de licitação. Novamente, os relacionamentos possuem direção para a entidade intermediária, `Empresa->Item_Licitação<-Empresa`, e por essa razão o sentido utilizado foi bidirecional, conforme apresentado na Consulta 5.4

Consulta 5.4: Busca de empresas relacionadas por itens utilizando o Cypher.

```
    MATCH (empresaB {nome: 'EMPRESA B'}) -[r:CONCORRE_POR*1..2]- (empresaC {nome: '
EMPRESA C'})
    RETURN empresaB, r ,empresaC
```

O resultado é uma nuvem de itens com os diversos vínculos entre as duas empresas como apresentado na Figura 5.4. Ambas as participantes foram colocadas nas extremidades opostas do grafo para ressaltá-las, assim como também os itens que elas disputaram. Sendo assim, ambas as empresas concorreram por 27 itens, o que pode ser um indício de acordo de preço, caso seja confirmado os vínculos societários entre elas. Da mesma forma que a Consulta 5.2, utilizar um banco de grafos simplificou consideravelmente obter o resultado da pesquisa em comparação com o modelo relacional.

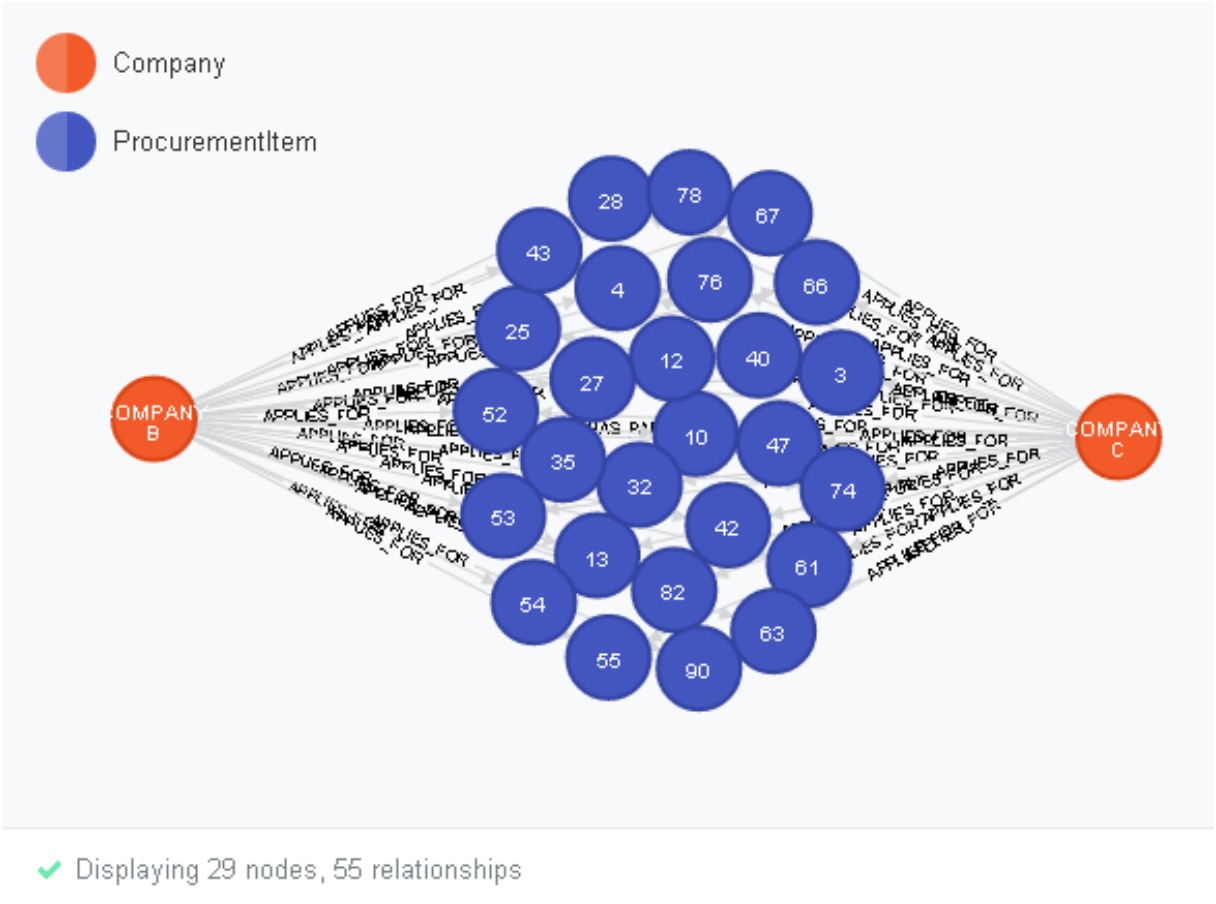


Figura 5.4: Resultado da consulta para empresas por itens de licitação no Neo4j.

Na realidade, o Cypher procura um padrão no grafo que atenda a declaração dada, e por isso retorna vários vértices e arestas que atendam o padrão, assim como os demais relacionamentos entre eles. Outra forma de consultar os casos onde se tem origem e

destino, é dizer ao banco para procurar pelos menores caminhos entre os dois nós ao invés de procurar um padrão. Isso pode melhorar o tempo de resposta, já que o banco utilizará um algoritmo de grafos específico para essa busca, ao invés de buscar por um padrão de grafo. A resposta pode ser ainda mais rápida se a necessidade for de localizar apenas um caminho ao invés de todos os menores caminhos.

A Consulta 5.5 realiza a busca pelos menores caminhos e obtém o mesmo resultado da Figura 5.4, já que todos os caminhos para ambas as consultas tem o mesmo tamanho de 2 saltos.

Consulta 5.5: Busca de empresas relacionadas por itens utilizando menor caminho.

```
MATCH p = allShortestPaths((o:Empresas{nome:"EMPRESA B"})-[r:CONCORRE_POR*..2]-(d
:Empresas{nome:"EMPRESA C"})) RETURN NODES(p)
```

Para questões de apenas um salto, percebe-se que não existem muitas dificuldades em verificar relacionamentos nos bancos relacionais, porém, quando se aumenta a profundidade da pesquisa nos relacionamentos, a vantagem de se utilizar grafos fica mais evidente, como é apresentado na Questão 3.

5.2.3 Questão 3: Como duas empresas se relacionam a partir de suas sociedades até o terceiro nível de empresa?

O objetivo dessa consulta é identificar se duas empresas estão relacionadas até um certo nível de profundidade nos relacionamentos de sociedade. Existe uma certa similaridade dessa consulta e outras da literatura como a de amigos de amigos[32], porém essa leva em consideração mais de uma entidade para o mesmo tipo de relacionamento (`Pessoa-Empresa` e `Empresa-Empresa`).

Como uma `Empresa` pode ter sócios tanto das entidades de `Pessoa` como `Empresa`, ambas as sociedades devem ser utilizadas. Dessa forma, buscamos, a partir de uma das empresas, seus sócios para então localizar as demais empresas no qual eles também são sócios. Se uma delas for igual a outra empresa alvo, esse relacionamento deve ser retornado.

Foi definido que a procura seria feita até o terceiro nível de `Empresa` para permitir uma melhor avaliação do problema, mas sem se tornar complexo demais para executar. Realizar uma busca até o terceiro nível de `Empresa` significa que se partirá de uma das empresas alvo, considerada nível raiz, buscando a cada nível de sociedade abaixo da raiz a segunda empresa alvo até que seja encontrada ou atingido a terceira `Empresa` na cadeia de sócios/empresas.

A Figura 5.5 apresenta uma árvore com as possibilidades de desdobramentos para essa questão. Uma empresa alvo é colocada no topo como raiz. Essa pode possuir um sócio do tipo Pessoa, no ramo esquerdo, ou uma outra Empresa, no ramo direito. Como uma `Pessoa` possui relacionamento de sociedade apenas com empresas, o próximo nível do ramo esquerdo será obrigatoriamente uma `Empresa`. Já para o ramo direito, uma `Empresa` pode ter relacionamento de sociedade com `Pessoa` ou `Empresa`, bifurcando novamente o caminho. A árvore então segue até que cada caminho atinja três empresas abaixo da raiz.

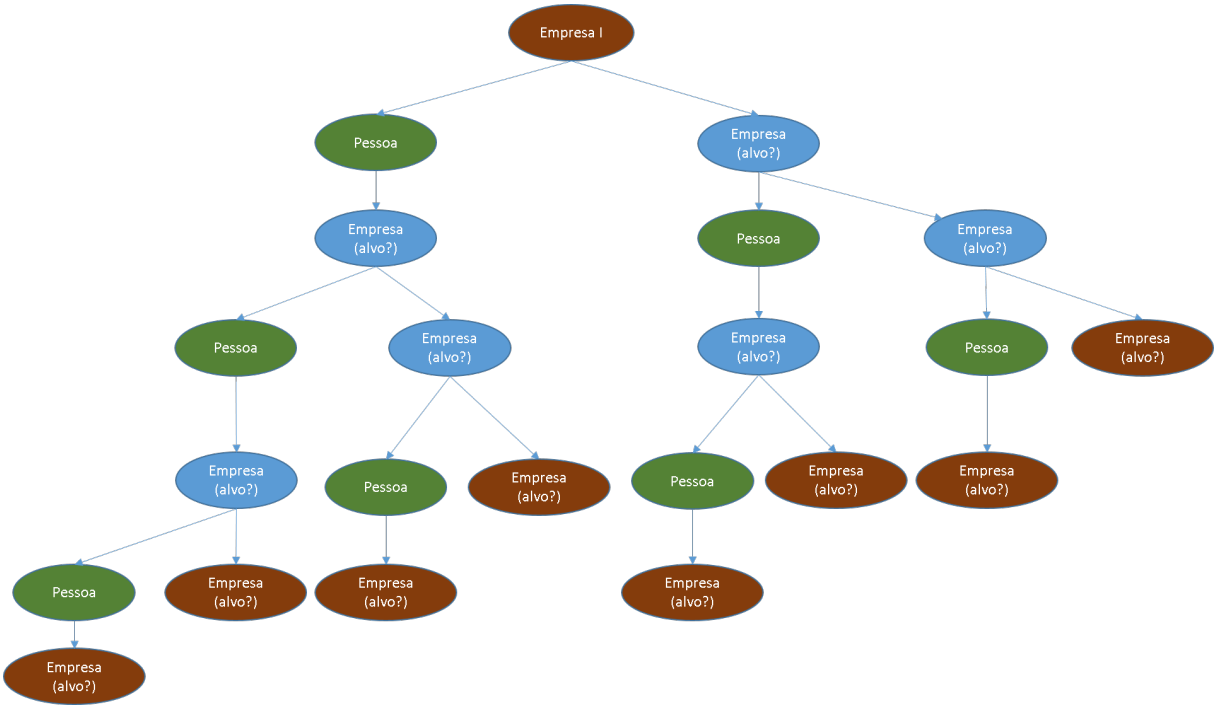


Figura 5.5: Árvore de consulta para duas empresas por sócios até a terceira empresa.

A árvore dessa questão demonstra como a busca no banco deverá ser mais complexa que as outras duas, pois será necessário cobrir vários níveis com alternâncias e junções diferentes, assim como suas projeções. Observando a Figura 5.6, a parte mais a esquerda da árvore foi destacada para ilustrar a maior profundidade que a busca pode chegar. Apesar de serem necessários apenas três empresas após a raiz, esse maior ramo da árvore demonstra a necessidade de ir a uma profundidade de seis nós devido a alternância entre os sócios do tipo Pessoa e Empresa.

Esse ramo mais profundo recupera todos os sócios do tipo Pessoa e as demais Empresas nas quais eles são sócios. Segue-se dessa forma até atingir a terceira empresa ou os seis níveis de profundidade. Apenas esses ramos podem ser obtido a partir da Consulta 5.6.

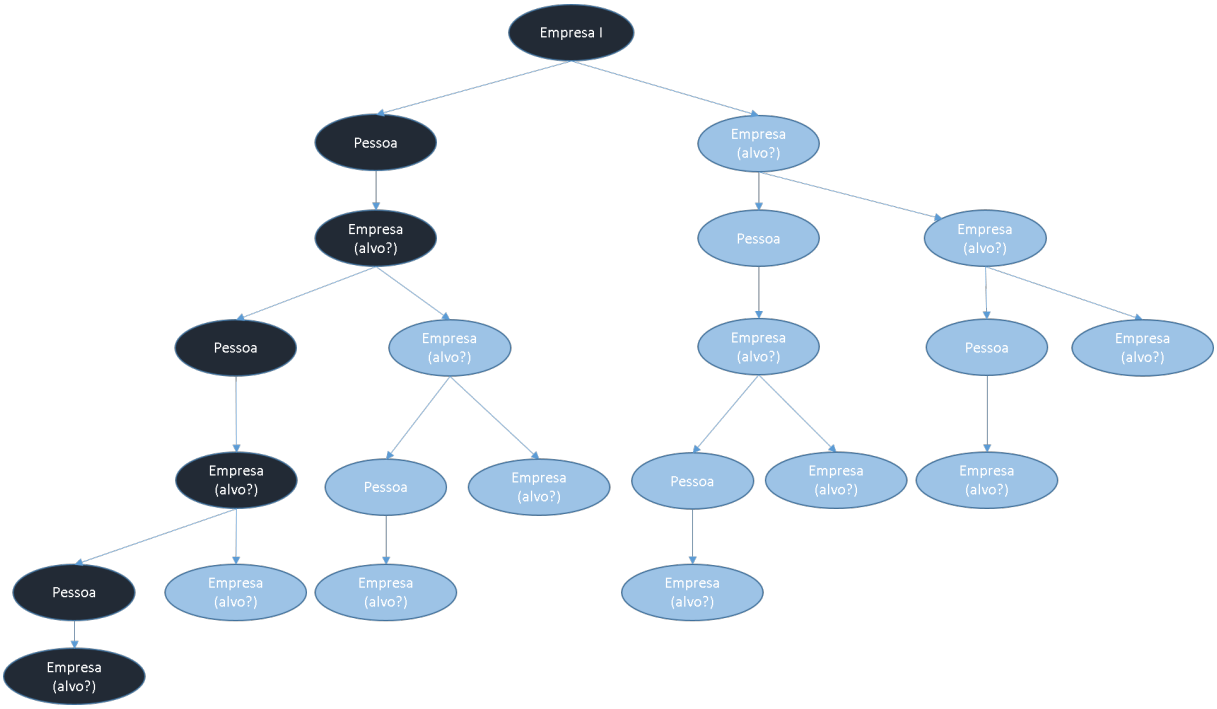


Figura 5.6: Destaque do primeiro ramo da árvore de consulta para duas empresas por sócios.

Consulta 5.6: Consulta de empresas que concorreram a licitações.

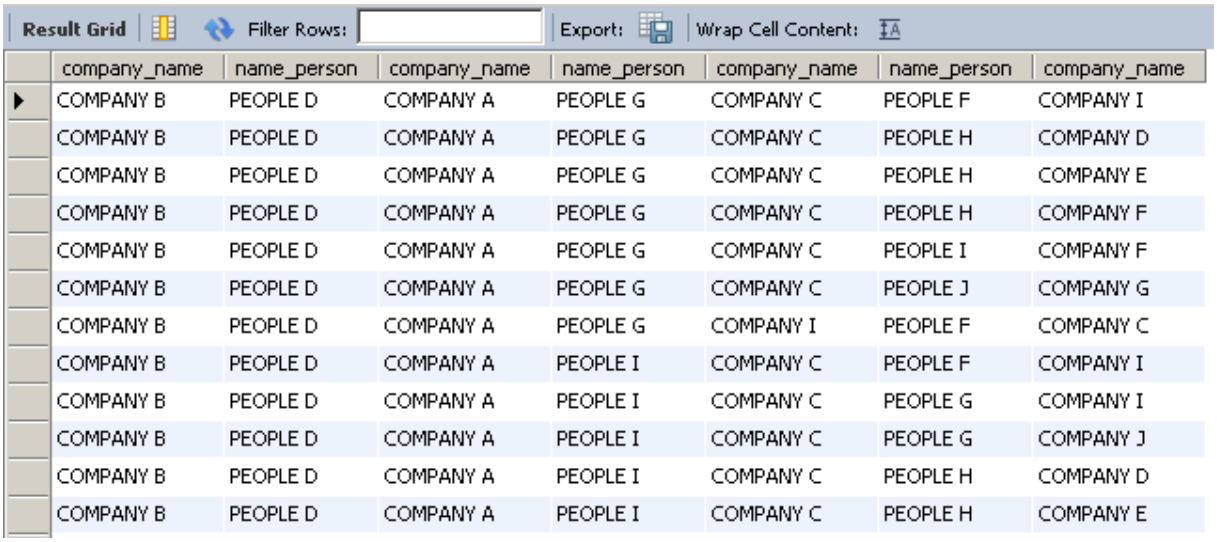
```
##############################BRANCH 01##########################################
#EMPRESA - PESSOA - EMPRESA - PESSOA - EMPRESA - PESSOA - EMPRESA

SELECT
  C01.nome_empresa,
  P01.nome_pessoa,
  C02.nome_empresa,
  P02.nome_pessoa,
  C03.nome_empresa,
  P03.nome_pessoa,
  C04.nome_empresa
FROM PessoasSocios AS PP01
  INNER JOIN Empresas AS C01 --Inicio das 12 juncoes.
    ON C01.pk_empresa = PP01.fk_empresa
  INNER JOIN Pessoas AS P01
    ON P01.pk_pessoas = PP01.fk_pessoa_socio
  LEFT JOIN PessoasSocios AS PP02
    ON PP02.fk_pessoa_socio = P01.pk_pessoa
  LEFT JOIN Empresas AS C02
    ON C02.pk_empresa = PP02.fk_empresa
  LEFT JOIN PessoasSocios AS PP03
    ON PP03.fk_empresa = C02.pk_empresa
  LEFT JOIN Pessoas AS P02
    ON P02.pk_pessoa = PP03.fk_pessoa_socio
  LEFT JOIN PessoasSocios AS PP04
    ON PP04.fk_pessoa_socio = P02.pk_pessoa
  LEFT JOIN Empresas AS C03
```

```
25      ON C03.pk_empresa = PP04.fk_empresa
26    LEFT JOIN PessoasSocios AS PP05
27      ON PP05.fk_empresa = C03.pk_empresa
28    LEFT JOIN Pessoas AS P03
29      ON P03.pk_pessoas = PP05.fk_pessoa_socio
30    LEFT JOIN PessoasSocios AS PP06
31      ON PP06.fk_pessoa_socio = P03.pk_pessoa
32    LEFT JOIN Empresas AS C04
33      ON C04.pk_empresa = PP06.fk_empresa --Fim das 12 juncoes
34 WHERE C02.pk_empresa <> C01.pk_empresa
35    AND C02.pk_empresa <> C03.pk_empresa
36    AND C01.pk_empresa <> C03.pk_empresa
37    AND C04.pk_empresa <> C01.pk_empresa
38    AND C04.pk_empresa <> C02.pk_empresa
39    AND C04.pk_empresa <> C03.pk_empresa
40    AND P02.pk_pessoa <> P01.pk_pessoa
41    AND P03.pk_pessoa <> P01.pk_pessoa
42    AND P03.pk_pessoa <> P02.pk_pessoa
43    AND ((C01.nome_empresa = 'EMPRESA B' AND C02.nome_empresa = 'EMPRESA C')
44    OR (C01.nome_empresa = 'EMPRESA B' AND C03.nome_empresa = 'EMPRESA C')
45    OR (C01.nome_empresa = 'EMPRESA B' AND C04.nome_empresa = 'EMPRESA C'));
```

Essa consulta permite verificar a complexidade envolvida em caminhar nos relacionamentos recursivamente a partir de um banco de dados relacional. Indo até seis níveis, foram necessárias 12 junções entre as tabelas de empresas, pessoas e sociedades. O resultado dessa consulta é apresentado na Figura 5.7, com a `Empresa B` como raiz no primeiro campo da projeção da consulta, e a `Empresa C` podendo aparecer em quaisquer dos outros campos retornados. Ressalta-se que este é apenas um dos ramos para verificar se existe um vínculo de sociedade entre duas empresas. Para se obter as informações completas, é necessário realizar outras consultas cobrindo os demais ramos e consolidá-las. A consulta completa para os outros ramos são apresentados em blocos no Anexo I, considerando os ramos da esquerda para a direita.

Trazendo para o modelo de grafos, a Consulta 5.7 apresenta a forma de relacionar duas empresas pelos seus sócios, independente de serem pessoas ou outras empresas. Da mesma forma que as demais consultas, sociedade e empresa seguem um sentido, e por isso foi utilizada na consulta a indicação que se deseja observar os vínculos bidirecionalmente.

Como foi apresentado na árvore das Figuras 5.5 e 5.6, pode-se descer até o sexto nível da árvore, e por isso foi especificado esse nível utilizando-se a restrição `E_SOCIO*1..6` na Consulta 5.7. Diferente do modelo relacional, a busca em um grafo permite utilizar apenas uma consulta para passar por todos os caminhos desejados, retornando sequencias de sócios independente do tipo.



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

company_name	name_person	company_name	name_person	company_name	name_person	company_name
COMPANY B	PEOPLE D	COMPANY A	PEOPLE G	COMPANY C	PEOPLE F	COMPANY I
COMPANY B	PEOPLE D	COMPANY A	PEOPLE G	COMPANY C	PEOPLE H	COMPANY D
COMPANY B	PEOPLE D	COMPANY A	PEOPLE G	COMPANY C	PEOPLE H	COMPANY E
COMPANY B	PEOPLE D	COMPANY A	PEOPLE G	COMPANY C	PEOPLE H	COMPANY F
COMPANY B	PEOPLE D	COMPANY A	PEOPLE G	COMPANY C	PEOPLE I	COMPANY F
COMPANY B	PEOPLE D	COMPANY A	PEOPLE G	COMPANY C	PEOPLE J	COMPANY G
COMPANY B	PEOPLE D	COMPANY A	PEOPLE G	COMPANY I	PEOPLE F	COMPANY C
COMPANY B	PEOPLE D	COMPANY A	PEOPLE I	COMPANY C	PEOPLE F	COMPANY I
COMPANY B	PEOPLE D	COMPANY A	PEOPLE I	COMPANY C	PEOPLE G	COMPANY I
COMPANY B	PEOPLE D	COMPANY A	PEOPLE I	COMPANY C	PEOPLE G	COMPANY J
COMPANY B	PEOPLE D	COMPANY A	PEOPLE I	COMPANY C	PEOPLE H	COMPANY D
COMPANY B	PEOPLE D	COMPANY A	PEOPLE I	COMPANY C	PEOPLE H	COMPANY E

Figura 5.7: Resultado da consulta de sócios para o primeiro ramo no MR.

Consulta 5.7: Busca relacionamentos entre duas empresas por sócio utilizando o Cypher.

```
    MATCH (empresaB { nome:'EMPRESA B' }) -[r:E_SOCIO*1..6]- (empresaC { nome:'
EMPRESA C' })
    RETURN empresaB, r, empresaC;
```

O resultado da consulta é apresentado na Figura 5.8. As duas empresas verificadas foram postas em lados opostos para facilitar a leitura. Pode-se ver no resultado os caminhos que passam por empresas encadeadas, da mesma forma que alternados entre as instâncias de `Pessoa` e `Empresa`, além das suas variações.

Esse resultado é devido à busca pelo padrão de grafo em que estejam as duas empresas em até seis saltos de distância entre elas, capturando todos os vértices dos caminhos entre elas. Embora o resultado traga diversas informações sobre quem está relacionado entre elas, ao se aumentar o tamanho do grafo pode ser tão trabalhoso para o banco de grafos trazer esse padrão com todos seus vértices quanto para o relacional realizando as junções.

Porém, pode-se tirar proveito da estrutura de grafos, assim como apresentado na Consulta 5.5. Ao invés de se consultar pelo padrão do grafo, pode-se buscar pelos menores caminhos entre as `Empresas B` e `C`, para identificar como elas se relacionam sem necessariamente trazer todos os caminhos possíveis (`allShortestPaths`). A Consulta 5.8 demonstra a utilização do menor caminho para o caso de sociedade.

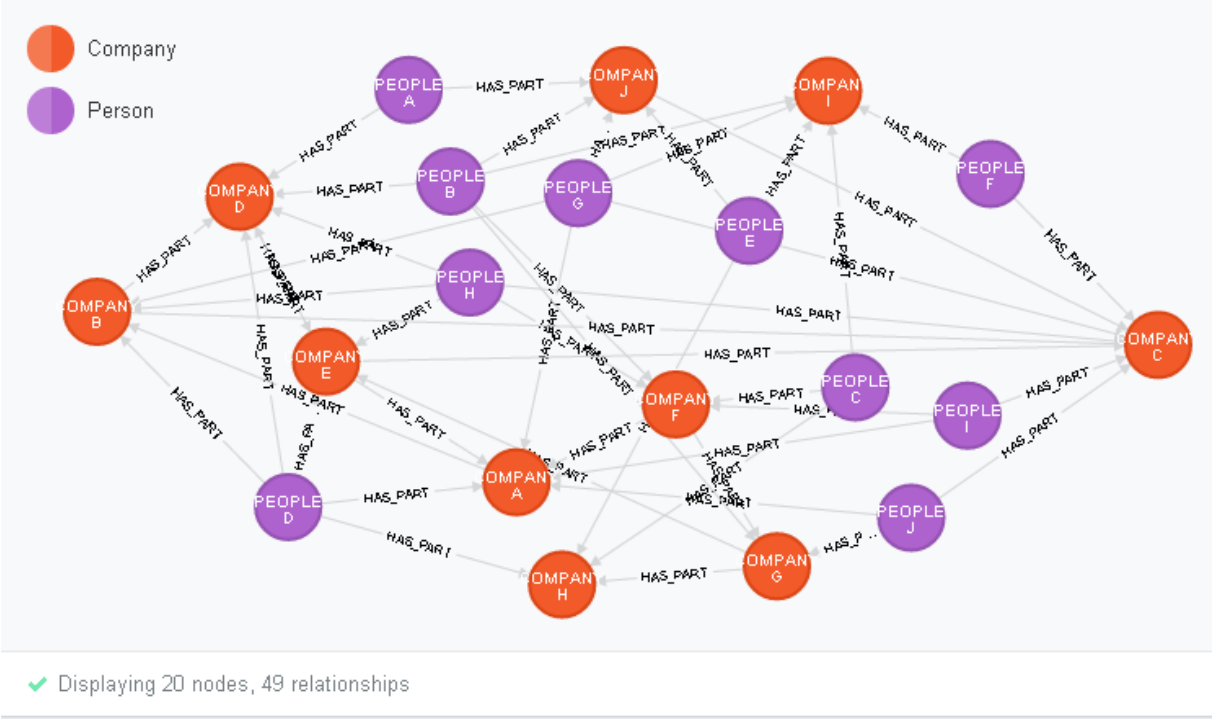


Figura 5.8: Resultado da consulta para sócios de empresas no Neo4j.

Consulta 5.8: Busca de vínculos entre duas empresas por sócio utilizando menores caminhos.

```
MATCH p = allShortestPaths((o:Empresas{nome:"EMPRESA B"})-[r:E_SOCIO*..6]-(d:
Empresas{nome:"EMPRESA C"})) RETURN NODES(p)
```

O resultado da Consulta 5.5 é apresentado na Figura 5.9. Desta vez, diferente da consulta de `Licitações`, o grafo retornado foi simplificado a dois nós. Isso porque existe apenas um menor caminho entre a `Empresa B` e `C`, ou seja, são conectados diretamente pois possuem um relacionamento de sociedade. Como só existe um caminho com essa distância, apenas ele foi retornado, destacando a sociedade direta entre as empresas, que junto com o resultado da consulta de participação por `Item_Licitação`, apresenta um forte indício de que esse processos foram fraudados.

Como comentado anteriormente, a possibilidade de utilizar esse tipo de função é uma vantagem para o banco de grafos. Não existe uma função equivalente na linguagem SQL e dessa forma seria necessário construir uma função ou procedimento que implementasse essa procura entre as tabelas. Já o banco de grafos pode tratar esse tipo de operação como algo natural de sua estrutura.

Das consultas apresentadas, a por sociedade é a que permite visualizar o maior contraste entre utilizar o banco relacional com SQL e um banco de grafos com uma linguagem

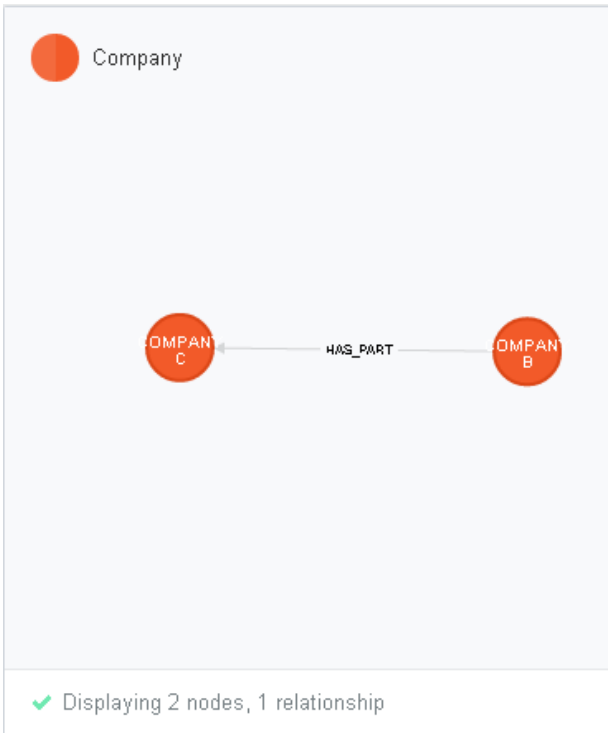


Figura 5.9: Resultado da consulta para sócios de empresas no Neo4j com menor caminho.

especializada. Essa comparação demonstra como a utilização dessa ferramenta especializada pode contribuir com a busca de vínculos entre entidade de forma muito mais flexível que o modelo relacional. Ainda que não considerando as questões de forma de armazenamento e indexação, nas camadas lógicas e de consulta, as vantagens ficam evidenciadas.

5.3 Modelo Relacional Descrevendo um Grafo

Nesse cenário, foi proposto utilizar uma modelagem relacional para abstrair o conceito de grafo, ou seja, utilizando tabelas para armazenar vértices e arestas, supondo que a abstração das entidades possa simplificar a consulta. Trabalhos como o de Vicknair [37], Ruflin et al [33] e Batra [5] fizeram comparativos entre bancos de grafos e relacionais, mas esses não tentaram abstrair em tabelas o modelo de grafos.

Considerando a questão de sociedades elaborada para o comparativo das seções anteriores, foi construído uma abstração dos grafos de atributos, com a mesma estrutura utilizada no banco de grafos selecionado. Sendo assim, ao invés de criar as relações baseadas nas entidades do domínio de `Pessoas` e `Empresas`, o modelo relacional terá apenas tabelas de vértices e arestas, além das relações para atributos destes.

O objetivo na construção desse modelo é verificar se ele pode simplificar as buscas por relacionamentos. Entretanto, foi realizada apenas a consulta para a terceira questão, pois

considerou-se que esta, por ser mais complexa, já seria suficiente para avaliar a adaptação.

A Figura 5.10 apresenta o modelo relacional proposto. Ele possui uma tabela para vértices e outra para as arestas. Ambas possuem uma chave além de um campo para rótulo. A tabela de aresta mantém o relacionamento entre os vértice pelas chaves estrangeiras, podendo inclusive possuir duas conexões para o mesmo par, considerando a necessidade de vínculos de natureza diferente. Os atributos são armazenados em tabelas separadas, tanto dos vértices como das arestas, sendo permitido vários atributos para ambos, desde que tenham chaves diferentes.

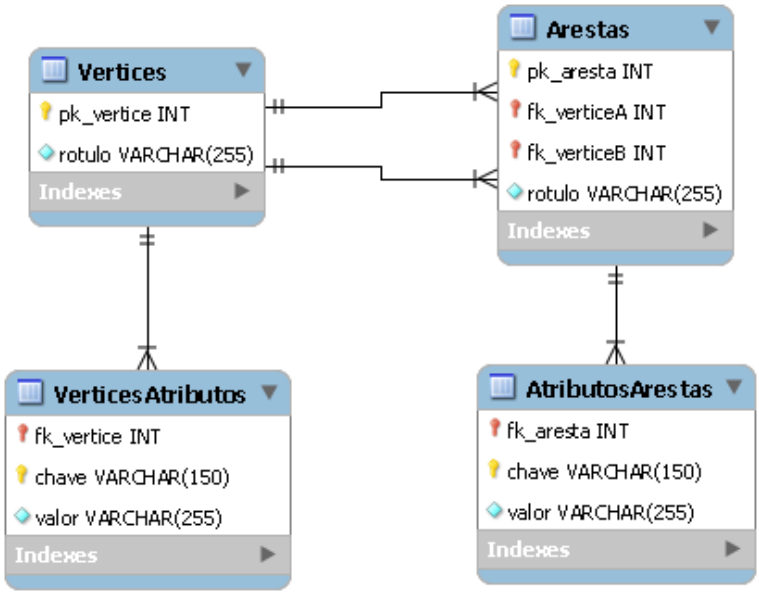


Figura 5.10: Modelagem relacional simplificada de um grafo de atributos.

Para esse modelo, quando se desejar incluir uma `Empresa`, essa deve ser inserida na tabela de vértices, com o atributo de nome na relação de atributos, ligando-as pela chave estrangeira. Já na aresta, caso seja necessário especificar o tipo de vinculo, além do rótulo, como sócio administrado, pode-se utilizar os atributos sem necessidade de criar uma tabela de qualificação. Apesar dos atributos poderem ficar em uma tabela apenas utilizando um campo de tipo para separar quais são de vértices e quais são de arestas, foi preferido manter os conceitos separados para garantir maior clareza e simplicidade.

Após a construção do modelo, o banco de dados adaptado foi implementado no MySQL e realizada a Consulta 5.9. Observa-se que para essa consulta foi necessário utilizar o artifício das linhas 19 e 23 na junção por conta da bilateralidade de sociedade. Se o sócio está sempre no atributos `fk_vertexA`, não teremos pessoas em `fk_vertexB`, sendo assim, na passagem do segundo grupo de junções para o terceiro ocorrerá uma exclusão do encadeamento do tipo `Pessoa-Empresa-Pessoa`, restando apenas os sócios `Empresa`.

Consulta 5.9: Consulta de sócios utilizando um grafo modelado em banco relacional.

```
1  SELECT DISTINCT
2    companyStart.pk_vertice AS empresaInicialPk,
3    companyStart.rotulo AS rotuloEmpresaInicial,
4    'E_SOCIO' AS edgeLabel01,
5    VT01.pk_vertice  AS alvo01,
6    VT01.rotulo  AS rotulo01,
7    E01.rotulo AS arestaRotulo01,
8    VT02.pk_vertice  AS alvo02,
9    VT02.rotulo  AS rotulo02,
10   E02.rotulo AS arestaRotulo02,
11   VT03.pk_vertice  AS alvo03,
12   VT03.rotulo  AS rotulo02
13 FROM vertices AS empresaInicial
14   INNER JOIN arestas AS E00 ON empresaInicial.pk_vertice = E00.fk_verticeB
15   INNER JOIN verticesatributos AS empresaInicialAtr
16         ON empresaInicialAtr.fk_vertice = empresaInicial.pk_vertice
17   LEFT JOIN vertices AS VT01
18     ON VT01.pk_vertice = E00.fk_verticeA
19   LEFT JOIN verticesatributos AS VATTT01
20     ON VATTT01.fk_vertice = VT01.pk_vertice
21
22   LEFT JOIN arestas AS E01
23     ON (VT01.pk_vertice=E01.fk_verticeA OR VT01.pk_vertice=E01.fk_verticeB)
24   LEFT JOIN vertices AS VT02
25     ON (VT02.pk_vertice=E01.fk_verticeA OR VT02.pk_vertice=E01.fk_verticeB)
26   LEFT JOIN verticesatributos AS VATTT02
27     ON VATTT02.fk_vertice = VT02.pk_vertice
28
29   LEFT JOIN arestas AS E02
30     ON (VT02.pk_vertice=E02.fk_verticeA OR VT02.pk_vertice=E02.fk_verticeB)
31   LEFT JOIN vertices AS VT03
32     ON (VT03.pk_vertice=E02.fk_verticeA OR VT03.pk_vertice=E02.fk_verticeB)
33   LEFT JOIN verticesatributos AS VATTT03
34     ON VATTT03.fk_vertice = VT03.pk_vertice
35
36 WHERE empresaInicial.rotulo = 'EMPRESAS'
37   AND empresaInicialAtr.valor = 'EMPRESA B'
38   AND (VATTT01.valor = 'EMPRESA C'
39     OR VATTT02.valor = 'EMPRESA C' OR VATTT03.valor = 'EMPRESA C')
40   AND VT02.pk_vertice <> VT01.pk_vertice
41   AND VT03.pk_vertice <> VT01.pk_vertice
42   AND VT03.pk_vertice <> VT02.pk_vertice;
```

Se for utilizada uma junção após o segundo grupo para pessoas, a consulta perderá sua generalização para se aproximar da Consulta 5.6, onde uma forma de se evitar essa construção é duplicar as linhas invertendo-se o sentido dos relacionamentos com consultas

bidirecionais, o que seria equivalente a manter duas arestas para representar a bidirecionalidade de um grafo direcionado. Ainda assim, seriam necessárias as junções, mesmo que nessa situação não se precise criar uma consulta para cada ramo da Figura 5.5.

Pode-se então concluir que a navegabilidade de um grafo ainda é impactada pela necessidade de inúmeras junções recursivas, mesmo abstraindo o conceito de grafo para armazená-lo como tabelas, além da necessidade de manter dois registros para indicar a bilateralidade de um relacionamento. Essa lacuna deixada pelo modelo relacional justifica e motiva o estudo e desenvolvimento na área de bancos de dados em grafos para permitir uma modelagem e implementação mais adequada e vantajosa para domínios que possam ser apresentados naturalmente como estruturas de grafos.

5.4 Consulta em Base com Dados Reais

Considerando a necessidade de implementação em ambiente real deste banco de dados na CGU, foi verificada a performance da consulta de sócios e empresas da Subseção 5.2.3, mas utilizando apenas a Consulta 5.6, referente ao ramo mais longo da árvore ilustrada na Figura 5.6.

Foram instalados em uma máquina provisória com 12 cores e 64 GB de RAM, o sistema de gerenciamento de bancos de dados relacional (SGDB) MySQL e o SGDB baseado em grafos Neo4j, utilizados alternadamente para evitar concorrências de recursos entre eles. Em cada um deles foram carregados 7.754.989 registros de empresas e 14.190.151 de Sócios com 20.903.480 relacionamentos de vínculos societários.

Para ambos os sistemas, foram extraídas as informações de sociedade do banco de dados de origem e colocadas em tabelas temporárias de um banco intermediário. A carga no MySQL foi realizada utilizando um programa proprietário para ETL de dados. Ela foi executada diretamente entre as tabelas temporárias e o MySQL sem maiores dificuldades. Já para o Neo4j, foi necessário exportar os dados de sociedades para um arquivo e quebrá-lo em diversos outros para permitir o carregamento.

O Neo4j necessitou do desenvolvimento de um programa para inserção dos dados, já que as ferramentas que o acompanham não suportaram bem a inserção de grandes volumes de dados. O sistema desenvolvido consiste em um programa[1] para ler um XML e carregar o mapeamento de vértices e arestas de um arquivo CSV ou consulta SQL. No caso de arquivos, pode-se também ler uma coleção de arquivos com a mesma estrutura, diminuindo o uso da memória. Através desse sistema foi possível carregar as informações de sociedade utilizada no comparativo. Um exemplo do arquivo XML é apresentado na Figura 5.11. Ele descreve a estrutura de grafo que será carregada, definindo, além da

[1]Disponível em: https://github.com/gvanerven/graphrepo

origem e destino das informações, que campos que comporão a estrutura dos vértices e arestas. Os relacionamentos são indicados nas arestas a partir dos rótulos dos vértices de origem e destino. Dessa forma, cada campo de uma linha carregada é utilizada na estrutura de um vértice ou aresta.

```
<?xml version='1.0' encoding='utf-8'?>
<job description="JOB SOCIO EMPRESAS">
        <destination id="graphProcurements" type = "NEO4J" method="update">/data/base02/neo4j/data/graph.db/</destination>

        <source id="sociosEmpresas"
            type="Filetable"
            delimiter=";"
            textSurroundedChar='"'
            header="false"
            multiFiles="true"
            userHeader="cod_sociedade;cnpj_empresa;cnpj_socio;dt_entrada_socio;dt_saida_socio;cod_qualificacao;tipo_socio;cod_carga">
                /data/base01/sociedades_pj-.*
            </source>

        <vertex label="Empresa" aliasFor="PessoaJuridica">
                <map fieldName="cnpj_empresa" rename="cnpj" key="true" type="long"/>
        </vertex>

        <vertex label="PessoaJuridica">
                <map fieldName="cnpj_socio" rename="cnpj" key="true" type="long"/>
        </vertex>

        <edge id="SocioSociedade" vertexSrc="PessoaJuridica" label="SOCIO_DE" vertexDst="Empresa">
                <map fieldName="dt_entrada_socio" type="string"/>
                <map fieldName="dt_saida_socio" type="string"/>
                <map fieldName="cod_qualificacao" key="true" type="int"/>
                <map fieldName="cod_carga" type="int"/>
        </edge>

</job>
```

Figura 5.11: Exemplo de XML para carga de sociedade entre empresas.

O programa foi desenvolvido para facilitar a carga e funciona com o conceito de tarefa. Cada tarefa é definida em um arquivo XML. O objetivo do arquivo é mapear os campos do CSV ou consulta com os vértices, arestas e atributos. Dentro do XML se destacam quatro *tags*: destino (`destination`), fonte de dados (`source`), vértice (`vertex`) e aresta (`edge`). A *tag* de destino indica para qual banco serão transferidas as informações, a ideia é permitir a carga em diferentes bancos de dados em grafo. Para o Neo4j, deve-se colocar o diretório que se encontra os arquivos do banco de dados. Já para a *tag* de fonte de dados (`source`), são detalhados as informações de como deve ser lida ou consultada as informações. Como a fonte pode possuir muitos registros e não permitir a carga de uma vez por questões de memória física da máquina, o arquivo pode ser criado sem cabeçalho e desmembrado em diversos outros arquivos com a mesma estrutura. Isso permite o uso das opções de múltiplos arquivos (`multiFiles`) e cabeçalho (`header`) indicando, respectivamente, que deverão ser lidos todos os arquivos do diretório com o mesmo padrão de nome e utilizado o cabeçalho indicado em todos eles.

Para o mapeamento dos nós, a *tag* vértice (`vertex`) é utilizada, já indicando seu rótulo e quais serão os atributos. Caso se tenha que definir dois vértices da mesma entidade, como apresentado na Figura 5.11, o rótulo deverá assumir outro valor, referenciado posteriormente para as arestas, utilizando-se a opção `aliasFor` para indicar o rótulo real da entidade. Já para os arcos, a *tag* arestas (`edge`) mapeia quais são os relacionamentos

entre os vértices, utilizando a opção `label` para indicar o relacionamento, assim como `vertexSrc` `vertexDst` para a origem e destino, respectivamente. Tanto para os vértices como para as arestas, a *tag* mapa (`map`) é utilizada para especificar os atributos, que podem ser um inteiro, ponto flutuante ou cadeia de caracteres, até a presente implementação.

Depois dos dados carregados em ambos os bancos, foi realizada a consulta com o banco "frio", ou seja, logo após a inicialização do SGDB para evitar influência do cache. Foram escolhidas duas empresas que apresentaram uma quantidade aproximada de 400 vínculos societários para realizar a consulta utilizando-as como origem e destino.

A consulta no banco MySQL durou aproximadamente 20 horas (**72.608,949 s**) e retornou um relacionamento com seis saltos de profundidade. Ele foi o único vínculo alternando entre os sócios `Pessoa` e `Empresa`, utilizando-se até seis saltos. Sendo assim, a busca no banco de dados em grafo deve retornar ao menos esse resultado quando consultado.

Quando realizada a consulta de sociedades no banco Neo4j, utilizando a busca pelo padrão do grafo como na Consulta 5.7, o mesmo não conseguiu realizá-la retornando erro na alocação de espaço de memória após aproximadamente 15 horas de execução. Provavelmente o Neo4j não conseguiu manipular todos os vértices e arestas ao tentar localizar todas as ligações possíveis que podem estar entre as duas empresas (`Empresa`-*-`Empresa`). Entretanto, como ressaltado anteriormente, o objetivo principal é identificar se existe relação entre duas empresas em até seis saltos, não sendo necessário trazer todo o grafo que envolvem as duas Empresas. Pode-se, alternativamente, buscar apenas os menores caminhos entre elas. Dessa forma, foi realizada a pesquisa por menor caminho obtendo-se o resultado da Figura 5.12, referente ao mesmo vínculo retornado no banco relacional, mas com um tempo de **369 ms**, satisfatório para uma aplicação *online* de busca de vínculos.

Percebe-se, então, que ao se mudar a tecnologia utilizada para obter informações, a maneira como se pesquisa pode impactar consideravelmente o resultado da busca. Nota-se também que pesquisas que seriam complexas de se realizar em uma estrutura podem ser muito beneficiadas ao alterar a forma de armazenamento e manipulação, como nesse caso do uso de grafos.

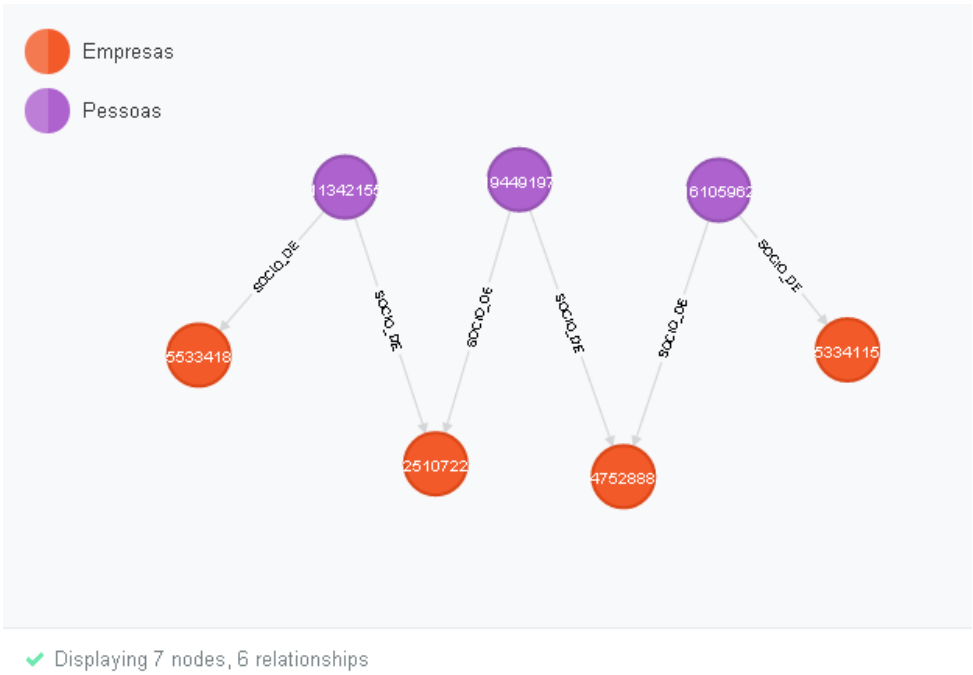


Figura 5.12: Resultado da consulta de sócios no Neo4j.

Capítulo 6

Conclusão

Esta pesquisa apresentou o Modelo de Dados para Bancos NoSQL Baseados em Grafos (MDG-NoSQL), consolidando em uma notação as diversas características e estruturas de dados presentes em Bancos de Dados em Grafos. O MDG-NoSQL especifica a notação que atende o modelo de dados para grafos simples, de atributos, hipergrafos e aninhados. Para isso, ele define uma notação principal para os elementos de vértice e aresta, além de suas generalizações de hipervértice e hipearesta, e complementa com diversas características opcionais ao modelo, como tipagem de rótulo e atributos.

O MDG-NoSQL foi desenhado a partir de notações simples, encontradas em programas de diagramação, sendo que, para a construção dos diagramas desse trabalho, foi utilizada a ferramenta livre DIA. A medida que as características do MDG-NoSQL foram sendo apresentadas, diagramas ilustrando seu uso em situações específicas ajudaram um melhor entendimento.

Após sua apresentação, o MDG-NoSQL foi utilizado para modelar Banco de Vínculos Simplificado para Licitações e Sociedades juntamente com o modelo relacional. A comparação das duas abordagens permitiu verificar como cada modelo se comporta a alguns cenários de mudanças, gerar considerações sobre a modelagem de dados em grafos e evidenciar também algumas diferenças entre a modelagem em grafos e relacional.

Utilizando dados simulados construídos a partir de *scripts* SQL, foram implementados e carregados dois sistemas gerenciadores de bancos de dados (SGDB), um relacional e outro em grafos. Esses dados permitiram avaliar, qualitativamente, como podem ser realizadas pesquisas utilizando as linguagens nativas desses dois sistemas. As consultas foram limitadas a três situações comuns no contexto do estudo de caso. Após a análise dessas consultas, foi possível perceber a vantagem do banco de dados em grafos para buscas que envolvem a navegação entre os relacionamentos. Consultas complexas, como buscar vários sócios entre duas empresas, ficaram mais simples no ambiente de grafos, além de se beneficiarem dos algoritmos específicos para essa estrutura de dados.

Também foram avaliadas algumas características de carga e consulta sobre as implementações de grafo e relacional, utilizando uma massa de dados reais. Apesar do processo de carga ter sido simples no banco relacional, para o banco de dados em grafo foi necessário o desenvolvimento de um programa para facilitar a carga dos dados, devido a falta de ferramentas para inserção quantidades de dados na ordem de milhões de registros. O programa utiliza o conceito de tarefa, descrita em um arquivo XML, que mapeia vértices e arestas para o banco de grafos utilizado na implementação.

Finalizada a carga dos dados, foi possível verificar algumas diferenças de desempenho entre as duas abordagens, assim como na forma de construção das consultas para busca de vínculos e como um problema de natureza típica de grafo, como relacionamentos entre sócios, pode tirar proveito de uma estrutura de dados especializada.

Por fim, percebe-se que diversas pesquisas ainda podem ser realizadas a partir do término dessa trabalho. No MDG-NoSQL, pode-se avaliar outros domínios, inclusive de problemas típicos, como Sistemas de Gerenciamento de Alunos em Cursos ou Compras *Online*, buscando lacunas para expandir o modelo, ou mesmo aperfeiçoar técnicas de modelagem que tirem o melhor proveito da estrutura de grafos em situações recorrentes, gerando um conjunto de padrões para modelagem. Saindo do escopo do modelo de dados e alternando para proposta dos bancos NoSQL de tratar quantidades de dados cada vez maiores, pode-se elaborar pesquisas para expandir e otimizar a capacidade de manipular volumes cada vez maiores de informações, que no contexto de grafos, se reflete em vértices e arestas. Do ponto de vista da implementação, exitem várias iniciativas que desenvolvem soluções de bancos de dados em grafos, justificando suas classificações, comparações e estudos das estruturas e técnicas utilizadas internamente a elas, permitindo a elaboração de propostas que contribuam para a evolução desses sistemas e a construção de novas tecnologias.

Capítulo 7

Anexos

7.1 Anexo I

Consulta 7.1: Consulta de empresas que concorreram a licitações.

```
##############################BRANCH 02#########################################
#COMPANY - PERSON - COMPANY - PERSON - COMPANY - COMPANY

SELECT
  C01.company_name,
  P01.name_person,
  C02.company_name,
  P02.name_person,
  C03.company_name,
  C04.company_name
FROM people_partners AS PP01
  INNER JOIN companies AS C01
    ON C01.pk_company = PP01.fk_company
  INNER JOIN people AS P01
    ON P01.pk_person = PP01.fk_person_partner
  INNER JOIN people_partners AS PP02
    ON PP02.fk_person_partner = P01.pk_person
  INNER JOIN companies AS C02
    ON C02.pk_company = PP02.fk_company
  LEFT JOIN people_partners AS PP03
    ON PP03.fk_company = C02.pk_company
  LEFT JOIN people AS P02
    ON P02.pk_person = PP03.fk_person_partner
  LEFT JOIN people_partners AS PP04
    ON PP04.fk_person_partner = P02.pk_person
  LEFT JOIN companies AS C03
    ON C03.pk_company = PP04.fk_company
  LEFT JOIN companies_partners AS CP05
    ON CP05.fk_company_partner = C03.pk_company
  LEFT JOIN companies AS C04
    ON C04.pk_company = CP05.fk_company
WHERE C02.pk_company <> C01.pk_company
  AND C02.pk_company <> C03.pk_company
```

```
31    AND C01.pk_company <> C03.pk_company
32    AND C04.pk_company <> C01.pk_company
33    AND C04.pk_company <> C02.pk_company
34    AND C04.pk_company <> C03.pk_company
35    AND P02.pk_person <> P01.pk_person
36    AND ((C01.company_name = 'COMPANY B' AND C02.company_name = 'COMPANY C')
37      OR (C01.company_name = 'COMPANY B' AND C03.company_name = 'COMPANY C')
38      OR (C01.company_name = 'COMPANY B' AND C04.company_name = 'COMPANY C'));

################################################################################

##############################BRANCH 03#########################################
#COMPANY - PERSON - COMPANY - COMPANY - PERSON - COMPANY

1   SELECT
2     C01.company_name,
3     P01.name_person,
4     C02.company_name,
5     C03.company_name,
6     P02.name_person,
7     C04.company_name
8   FROM people_partners AS PP01
9     INNER JOIN companies AS C01
10      ON C01.pk_company = PP01.fk_company
11    INNER JOIN people AS P01
12      ON P01.pk_person = PP01.fk_person_partner
13    LEFT JOIN people_partners AS PP02
14      ON PP02.fk_person_partner = P01.pk_person
15    LEFT JOIN companies AS C02
16      ON C02.pk_company = PP02.fk_company
17    LEFT JOIN companies_partners AS CP03
18      ON CP03.fk_company_partner = C02.pk_company
19    LEFT JOIN companies AS C03
20      ON C03.pk_company = CP03.fk_company
21    LEFT JOIN people_partners AS PP04
22      ON PP04.fk_company = C03.pk_company
23    LEFT JOIN people AS P02
24      ON P02.pk_person = PP04.fk_person_partner
25    LEFT JOIN people_partners AS PP05
26      ON PP05.fk_person_partner = P02.pk_person
27    LEFT JOIN companies AS C04
28      ON C04.pk_company = PP05.fk_company
29  WHERE C02.pk_company <> C01.pk_company
30    AND C02.pk_company <> C03.pk_company
31    AND C01.pk_company <> C03.pk_company
32    AND C04.pk_company <> C01.pk_company
33    AND C04.pk_company <> C02.pk_company
34    AND C04.pk_company <> C03.pk_company
35    AND P02.pk_person <> P01.pk_person
36    AND ((C01.company_name = 'COMPANY B' AND C02.company_name = 'COMPANY C')
37      OR (C01.company_name = 'COMPANY B' AND C03.company_name = 'COMPANY C')
38      OR (C01.company_name = 'COMPANY B' AND C04.company_name = 'COMPANY C'));

################################################################################

##############################BRANCH 04#########################################
#COMPANY - PERSON - COMPANY - COMPANY - COMPANY
```

```
SELECT
  C01.company_name,
  P01.name_person,
  C02.company_name,
  C03.company_name,
  C04.company_name
FROM people_partners AS PP01
  INNER JOIN companies AS C01
    ON C01.pk_company = PP01.fk_company
  INNER JOIN people AS P01
    ON P01.pk_person = PP01.fk_person_partner
  LEFT JOIN people_partners AS PP02
    ON PP02.fk_person_partner = P01.pk_person
  LEFT JOIN companies AS C02
    ON C02.pk_company = PP02.fk_company
  LEFT JOIN companies_partners AS CP03
    ON CP03.fk_company_partner = C02.pk_company
  LEFT JOIN companies AS C03
    ON C03.pk_company = CP03.fk_company
  LEFT JOIN companies_partners AS CP04
    ON CP04.fk_company_partner = C03.pk_company
  LEFT JOIN companies AS C04
    ON C04.pk_company = CP04.fk_company
WHERE C02.pk_company <> C01.pk_company
  AND C02.pk_company <> C03.pk_company
  AND C01.pk_company <> C03.pk_company
  AND C04.pk_company <> C01.pk_company
  AND C04.pk_company <> C02.pk_company
  AND C04.pk_company <> C03.pk_company
  AND ((C01.company_name = 'COMPANY B' AND C02.company_name = 'COMPANY C')
    OR (C01.company_name = 'COMPANY B' AND C03.company_name = 'COMPANY C')
    OR (C01.company_name = 'COMPANY B' AND C04.company_name = 'COMPANY C'));

################################################################################

##############################BRANCH 05#########################################
#COMPANY - COMPANY - PERSON - COMPANY - PERSON - COMPANY

SELECT
  C01.company_name,
  C02.company_name,
  P01.name_person,
  C03.company_name,
  P02.name_person,
  C04.company_name
FROM companies_partners AS CP01
  INNER JOIN companies AS C01
    ON C01.pk_company = CP01.fk_company_partner
  LEFT JOIN companies AS C02
    ON C02.pk_company = CP01.fk_company
  LEFT JOIN people_partners AS PP02
    ON PP02.fk_company = C02.pk_company
  INNER JOIN people AS P01
    ON P01.pk_person = PP02.fk_person_partner
  LEFT JOIN people_partners AS PP03
    ON PP03.fk_person_partner = P01.pk_person
```

```
  LEFT JOIN companies AS C03
    ON C03.pk_company = PP03.fk_company
  LEFT JOIN people_partners AS PP04
    ON PP04.fk_company = C03.pk_company
  LEFT JOIN people AS P02
    ON P02.pk_person = PP04.fk_person_partner
  LEFT JOIN people_partners AS PP05
    ON PP05.fk_person_partner = P02.pk_person
  LEFT JOIN companies AS C04
    ON C04.pk_company = PP05.fk_company
WHERE C02.pk_company <> C01.pk_company
  AND C02.pk_company <> C03.pk_company
  AND C01.pk_company <> C03.pk_company
  AND C04.pk_company <> C01.pk_company
  AND C04.pk_company <> C02.pk_company
  AND C04.pk_company <> C03.pk_company
  AND P02.pk_person <> P01.pk_person
  AND ((C01.company_name = 'COMPANY B' AND C02.company_name = 'COMPANY C')
    OR (C01.company_name = 'COMPANY B' AND C03.company_name = 'COMPANY C')
    OR (C01.company_name = 'COMPANY B' AND C04.company_name = 'COMPANY C'));

################################################################################

##############################BRANCH 06##########################################
#COMPANY - COMPANY - PERSON - COMPANY - COMPANY

SELECT
  C01.company_name,
  C02.company_name,
  P01.name_person,
  C03.company_name,
  C04.company_name
FROM companies_partners AS CP01
  INNER JOIN companies AS C01
    ON C01.pk_company = CP01.fk_company_partner
  LEFT JOIN companies AS C02
    ON C02.pk_company = CP01.fk_company
  LEFT JOIN people_partners AS PP02
    ON PP02.fk_company = C02.pk_company
  INNER JOIN people AS P01
    ON P01.pk_person = PP02.fk_person_partner
  LEFT JOIN people_partners AS PP03
    ON PP03.fk_person_partner = P01.pk_person
  LEFT JOIN companies AS C03
    ON C03.pk_company = PP03.fk_company
  LEFT JOIN companies_partners AS CP04
    ON CP04.fk_company_partner = C03.pk_company
  LEFT JOIN companies AS C04
    ON C04.pk_company = CP04.fk_company
WHERE C02.pk_company <> C01.pk_company
  AND C02.pk_company <> C03.pk_company
  AND C01.pk_company <> C03.pk_company
  AND C04.pk_company <> C01.pk_company
  AND C04.pk_company <> C02.pk_company
  AND C04.pk_company <> C03.pk_company
  AND ((C01.company_name = 'COMPANY B' AND C02.company_name = 'COMPANY C')
    OR (C01.company_name = 'COMPANY B' AND C03.company_name = 'COMPANY C')
```

```
32      OR (C01.company_name = 'COMPANY B' AND C04.company_name = 'COMPANY C'));

################################################################################

################################BRANCH 07#######################################
#COMPANY - COMPANY - COMPANY - PERSON - COMPANY

1  SELECT
2    C01.company_name,
3    C02.company_name,
4    C03.company_name,
5    P01.name_person,
6    C04.company_name
7  FROM companies_partners AS CP01
8    INNER JOIN companies AS C01
9      ON C01.pk_company = CP01.fk_company_partner
10   INNER JOIN companies AS C02
11     ON C02.pk_company = CP01.fk_company
12   LEFT JOIN companies_partners AS CP02
13     ON CP02.fk_company_partner = C02.pk_company
14   LEFT JOIN companies AS C03
15     ON C03.pk_company = CP02.fk_company
16   LEFT JOIN people_partners AS PP03
17     ON PP03.fk_company = C03.pk_company
18   LEFT JOIN people AS P01
19     ON P01.pk_person = PP03.fk_person_partner
20   LEFT JOIN people_partners AS PP04
21     ON PP04.fk_person_partner = P01.pk_person
22   LEFT JOIN companies AS C04
23     ON C04.pk_company = PP04.fk_company
24 WHERE C02.pk_company <> C01.pk_company
25   AND C02.pk_company <> C03.pk_company
26   AND C01.pk_company <> C03.pk_company
27   AND C04.pk_company <> C01.pk_company
28   AND C04.pk_company <> C02.pk_company
29   AND C04.pk_company <> C03.pk_company
30   AND ((C01.company_name = 'COMPANY B' AND C02.company_name = 'COMPANY C')
31     OR (C01.company_name = 'COMPANY B' AND C03.company_name = 'COMPANY C')
32     OR (C01.company_name = 'COMPANY B' AND C04.company_name = 'COMPANY C'));

################################################################################

################################BRANCH 08#######################################
#COMPANY - COMPANY - COMPANY - COMPANY

1  SELECT
2    C01.company_name,
3    C02.company_name,
4    C03.company_name,
5    C04.company_name
6  FROM companies_partners AS CP01
7    INNER JOIN companies AS C01
8      ON C01.pk_company = CP01.fk_company_partner
9    INNER JOIN companies AS C02
10     ON C02.pk_company = CP01.fk_company
11   LEFT JOIN companies_partners AS CP02
12     ON CP02.fk_company_partner = C02.pk_company
```

```
13    LEFT JOIN companies AS C03
14      ON C03.pk_company = CP02.fk_company
15    LEFT JOIN companies_partners AS CP03
16      ON CP03.fk_company_partner = C03.pk_company
17    LEFT JOIN companies AS C04
18      ON C04.pk_company = CP03.fk_company
19 WHERE C02.pk_company <> C01.pk_company
20    AND C02.pk_company <> C03.pk_company
21    AND C01.pk_company <> C03.pk_company
22    AND C04.pk_company <> C01.pk_company
23    AND C04.pk_company <> C02.pk_company
24    AND C04.pk_company <> C03.pk_company
25    AND ((C01.company_name = 'COMPANY B' AND C02.company_name = 'COMPANY C')
26      OR (C01.company_name = 'COMPANY B' AND C03.company_name = 'COMPANY C')
27      OR (C01.company_name = 'COMPANY B' AND C04.company_name = 'COMPANY C'));

################################################################################
```

Referências

[1] Bernd Amann and Michel Scholl. Gram: a graph data model and query languages. In *Proceedings of the ACM conference on Hypertext*, pages 201–211. ACM, 1992. xi, 8

[2] R. Angles. A Comparison of Current Graph Database Models. In *2012 IEEE 28th International Conference on Data Engineering Workshops (ICDEW)*, pages 171–177, April 2012. 7, 11, 13, 14, 15, 31

[3] Renzo Angles and Claudio Gutierrez. Survey of Graph Database Models. *ACM Comput. Surv.*, 40(1):1:1–1:39, 2008. xi, 5, 7, 8, 11, 15

[4] Renzo Angles, Arnau Prat-Pérez, David Dominguez-Sal, and Josep-Lluis Larriba-Pey. Benchmarking Database Systems for Social Network Applications. In *First International Workshop on Graph Data Management Experiences and Systems*, '13, pages 15:1–15:7, New York, NY, USA, 2013. ACM. 1, 13, 14

[5] Shalini Batra and Charu Tyagi. Comparative analysis of relational and graph databases. *International Journal of Soft Computing and Engineering (IJSCE)*, 2(2), 2012. 76

[6] Deepayan Chakrabarti and Christos Faloutsos. *Graph Mining: Laws, Tools, and Case Studies*. Morgan & Claypool Publishers, October 2012. 4

[7] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data. *ACM Trans. Comput. Syst.*, 26(2):4:1–4:26, June 2008. 13, 14

[8] Gary Chartrand. *Introductory Graph Theory*. Dover Publications, New York, unabridged edition edition, December 1984. 4, 5

[9] Min Chen, Shiwen Mao, and Yunhao Liu. Big Data: A Survey. *Mobile Networks and Applications*, 19(2):171–209, April 2014. 13, 14

[10] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, Mass, third edition edition edition, July 2009. 60

[11] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall,

and Werner Vogels. Dynamo: Amazon's Highly Available Key-value Store. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*, '07, pages 205–220, New York, NY, USA, 2007. ACM. 13

[12] Marc Gyssens, Jan Paredaens, and Dirk van Gucht. A Graph-oriented Object Database Model. In *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, '90, pages 417–424, New York, NY, USA, 1990. ACM. 8, 10

[13] Robin Hecht and Stefan Jablonski. evaluation: A use case oriented survey. In *2011 International Conference on Cloud and Service Computing (CSC)*, pages 336–341, 2011. 4, 13

[14] Carlos Alberto Heuser. *Projeto de banco de dados*. Sagra Luzzatto, 1998. 27

[15] J. Hidders and J. Paredaens. Goal, a Graph-Based Object and Association Language. In J. Paredaens and L. Tenenbaum, editors, *Advances in Database Systems*, number 347 in International Centre for Mechanical Sciences, pages 247–265. Springer Vienna, January 1994. xi, 8, 10

[16] Jan Hidders. Typing Graph-Manipulation Operations. In Diego Calvanese, Maurizio Lenzerini, and Rajeev Motwani, editors, *Database Theory — ICDT 2003*, number 2572 in Lecture Notes in Computer Science, pages 394–409. Springer Berlin Heidelberg, 2003. xi, 8, 9

[17] Renée C. van der Hulst. Introduction to Social Network Analysis (SNA) as an investigative tool. *Trends in Organized Crime*, 12(2):101–121, June 2009. 2, 13, 45

[18] O'Reilly Media Inc. *Big Data Now: 2012 Edition*. O'Reilly Media, 2 edition edition, October 2012. 1, 13

[19] Salim Jouili and Valentin Vansteenberghe. An empirical comparison of graph databases. In *Social Computing (SocialCom), 2013 International Conference on*, pages 708–715. IEEE, 2013. 13

[20] P. Kalmegh and S.B. Navathe. Graph Database Design Challenges Using HPC Platforms. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, pages 1306–1309, November 2012. 13

[21] Karamjit Kaur and Rinkle Rani. Modeling and querying data in NoSQL databases. pages 1–7. IEEE, October 2013. 13

[22] Gabriel M. Kuper and Moshe Y. Vardi. The logical data model. *ACM Transactions on Database Systems (TODS)*, 18(3):379–413, 1993. 8

[23] Neal Leavitt. Will NoSQL Databases Live Up to Their Promise? *Computer*, 43(2):12–14, February 2010. 13

[24] M. Levene and G. Loizou. A graph-based data model and its ramifications. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):809–823, October 1995. xi, 8, 11, 12, 22

[25] M. Levene and A. Poulovassilis. An object-oriented data model formalised through hypergraphs. *Data & Knowledge Engineering*, 6(3):205–224, May 1991. 8, 11, 32

[26] David Loshin. *Business Intelligence: The Savvy Manager's Guide*. Morgan Kaufmann, Amsterdam ; Boston, 1 edition edition, July 2003. 44

[27] M. Mainguenaud and X. T. Simatic. A data model to deal with multi-scaled networks. *Computers, Environment and Urban Systems*, 16(4):281–288, July 1992. 8, 10

[28] Robert Campbell McColl, David Ediger, Jason Poovey, Dan Campbell, and David A. Bader. A Performance Evaluation of Open Source Graph Databases. In *Proceedings of the First Workshop on Parallel Programming for Analytics Applications*, '14, pages 11–18, New York, NY, USA, 2014. ACM. 13

[29] Jeffrey Scott McIllwain. Organized crime: A social network approach. *Crime, Law and Social Change*, 32(4):301–323, December 1999. 13, 45

[30] Russ Miles and Kim Hamilton. *Learning UML 2.0*. O'Reilly Media, Beijing ; Sebastopol, CA, 1 edition edition, May 2006. 22, 32

[31] Eric Redmond and Jim R. Wilson. *Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement*. Pragmatic Bookshelf, Dallas, Tex, 1 edition edition, May 2012. 1, 14

[32] Ian Robinson, Jim Webber, and Emil Eifrem. *Graph Databases*. O'Reilly Media, Sebastopol, Calif., 1 edition edition, June 2013. 11, 39, 64, 70

[33] Nicolas Ruflin, Helmar Burkhart, and Sven Rizzotti. Social-data Storage-systems. In *Databases and Social Networks*, '11, pages 7–12, New York, NY, USA, 2011. ACM. 1, 76

[34] Malcolm K. Sparrow. The application of network analysis to criminal intelligence: An assessment of the prospects. *Social Networks*, 13(3):251–274, 1991. 45

[35] Srinath Srinivasa. Data, Storage and Index Models for Graph Databases. In Sherif Sakr and Eric Pardede, editors, *Graph Data Management*, pages 47–70. IGI Global, 2011. 1, 11, 13

[36] Michael Stonebraker. Databases V. NoSQL Databases. *Commun. ACM*, 53(4):10–11, 2010. 13

[37] Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, and Dawn Wilkins. A Comparison of a Graph Database and a Relational Database: A Data Provenance Perspective. In *Proceedings of the 48th Annual Southeast Regional Conference*, SE '10, pages 42:1–42:6, New York, NY, USA, 2010. ACM. 76

[38] Jennifer Xu and Hsinchun Chen. Criminal Network Analysis and Visualization. *Commun. ACM*, 48(6):100–107, June 2005. 13, 45