UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

MODELAGEM PARTICIPATIVA
Uma Nova Abordagem para Modelar Bancos de Dados Voltados a Grafos

Luis Augusto Neumann

Itajubá, maio de 2017

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Luis Augusto Neumann

MODELAGEM PARTICIPATIVA
Uma Nova Abordagem para Modelar Bancos de Dados Voltados a Grafos

Dissertação submetida ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação como parte dos requisitos para obtenção do Título de Mestre em Ciência e Tecnologia da Computação

Área de Concentração: Área de Concentração

Orientador: Prof. Dr. Edmilson Marmo Moreira

Coorientador: Prof. Dr. Otávio Augusto Salgado Carpinteiro

maio de 2017
Itajubá - MG

Pai, o que é mais importante para você: eu ou esse seu trabalho de escola?

Sophia, 8 anos

Agradecimentos

Ao final de uma jornada de três anos mais do que agradecer às pessoas que de alguma maneira lhe apoiaram no trabalho, é importante dizer obrigado àqueles que perderam horas de convivência comigo e nesse sentido escrevo, com todas as letras, obrigado Tigrinha e Sol.

Agradeço ainda aos professores Edmilson, Otávio e Enzo pelo apoio durante o Mestrado e aos colegas Douglas Soares, Rafael, Alessandra e Phyllipe que caminharam juntos comigo.

E aos colegas do LNA, aqui representados pelo Bruno, Giuliana e Micheline, pelo apoio no dia a dia.

Resumo

Com a disseminação do uso dos computadores para quase todas as atividades humanas, cada vez mais dados de diferentes características precisam ser coletados, armazenados e pesquisados. Isso vem trazendo novos desafios para que esses dados gerem informação pois as formas tradicionais de bancos de dados vêm se mostrando ineficientes para atender essa nova demanda e assim novas concepções de armazenamento de dados tem sido apresentadas, dentre elas os bancos de dados voltados a grafos. Entretanto, não basta simplesmente armazenar os dados na forma de grafos pois é necessário saber como modelar esses dados para melhor explorar as características oferecidas ppor estes bancos de dados. Nesse sentido, esta dissertação analisa e compara alguns métodos propostos por diferentes pesquisadores para modelar dados na forma de grafos e, com base nessa análise, apresenta uma nova metodologia para modelar os dados como grafos, que produz um grafo menor com caminhos menores entre os vértices em relação às modelagens analisadas, com a incorporação de vértices por outros vértices ou mesmo arestas, facilitando, assim, a prospecção de dados. Os métodos analisados e proposto foram comparados, através da implantação de um base de teste comum.

Palavras chave: *Banco de Dados, NoSql, Modelagem, Grafo.*

Abstract

With the increased use of computers in most human activities, more and more data with different characteristics need to be collected, stored, and prospected. Infomation extraction from this new type of data is a challenge, and the traditional databases have proved to be inefficient to meet this new demand. As a result, new forms of databases have been presented, among them the graph databases. However, besides storing the data in a graph form, it is necessary to model the data properly so that the characteristics offered by these databases are better explored. In this context, this dissertation analyzes and compares some methods proposed by different researchers to model data in the form of graphs and, based on this analysis, presents a new methodology to model the data as graphs, which produces a smaller graph with smaller paths between the vertices in relation to the analyzed models, with the incorporation of vertices by other vertices or even edges, thus facilitating the prospecting of data. The analyzed and proposed methods were compared through the implementation of a common database test.

Keywords: *Database, NoSql, Modeling, Graph.*

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Listagens

Lista de Algoritmos

Glossário

ACID	*acrônimo para características necessárias para um banco de dados relacional: Atomicidade, Consistência, Integridade e Durabilidade*
BASE	*acrônimo para Basically Available, Soft state, Eventually consistent, em português Basicamente disponível, estado leve e Consistente em momento indeterminado*
Big Data	*Compreende o imenso volume de dados, estruturados e não estruturados, advindos das atividades do dia a dia, não somente os dados em si, mas a informação contida nestes dados.*
Cluster	*técnica que consiste em dividir fisicamente o banco de dados em diferentes servidores de forma que o servidor central sabe onde gravar ou buscar determinado tipo de dado, ficando isso transparente para as aplicações usuárias do banco*
Cypher	*é a linguagem de pesquisas em grafo do Neo4j, um sistema de banco de dados voltado a grafos.*
JSON	*um acrônimo para "JavaScript Object Notation", é um formato leve para intercâmbio de dados computacionais.*
NoSQL	*do inglês Not Only SQL, dá nome a uma nova corrente de bancos de dados que vão além dos bancosSQL*
SQL	*do inglês Structured Query Language, dá nome a linguagem criada para realizar consultas aos dados armaezenados em bancos de dados relacionais*
RDBMS	*do inglês a Relational DataBase Management System, sistema de gerenciamento de um banco de dados relacional*
Sharding	*Em bancos de dados, é o processo de armazenamento de registros em várias máquinas*
SGBD	*Sistemas Gerenciadores de Bancos de Dados*
SGBDG	*Sistemas Gerenciadores de Bancos de Dados de Grafos*
W3C	*Word Wide Web Consortium - consórcio de organizações que buscam padronizar conceitos, protocolos e padrões de estruturas para a Web, visando obter maior eficácia de seus recursos.*

1 Introdução

Com a explosão do uso da Internet aliada a equipamentos cada vez mais rápidos e com maior capacidade de armazenamento e replicação de dados, praticamente todas as atividades humanas estão sendo mapeadas e migradas para bancos de dados cada vez maiores e mais poderosos. É o fenômeno do chamado *Big Data*.

A maioria dos Sistemas de Gerenciamento de Bancos de Dados (SGBDs) segue uma concepção cujas bases foram lançadas na década de 70 do século passado e não têm atendido eficientemente o *Big Data*. Nesse sentido, novas concepções de bancos de dados, como os chamados bancos NoSQL, têm sido apresentadas e seu uso vem aumentando ano a ano, conforme mostra acompanhamento do *site* Db-engines (2017), ilustrado no gráfico da figura 1.1. O gráfico mostra a tendência histórica da popularidade das categorias de SGBDs, mês a mês e o que se observa, a partir de janeiro de 2014, é uma popularização acentuada do uso dos bancos NoSQL, principalmente dos bancos de dados voltados a grafos (SGBDGs) e uma estagnação dos bancos relacionais.

Já a figura 1.2 mostra que, apesar do crescimento do uso dos bancos NoSQL, o uso dos bancos relacionais ainda é dominante. Nesse sentido, praticamente todas as ferramentas e técnicas de modelagem ainda são voltadas para esse modelo de banco, o que vem dificultando uma maior expansão no uso de outros SGBDs.

Umas das características dos novos bancos é uma menor rigidez nas suas estruturas, permitindo, por exemplo, registros de uma mesma tabela com atributos diferentes, ou mesmo atributos de mesmo nome mas tipagem diferente.

Embora essa menor rigidez, ainda é preciso cuidado na modelagem desses bancos para possibilitar o melhor aproveitamento das suas características.

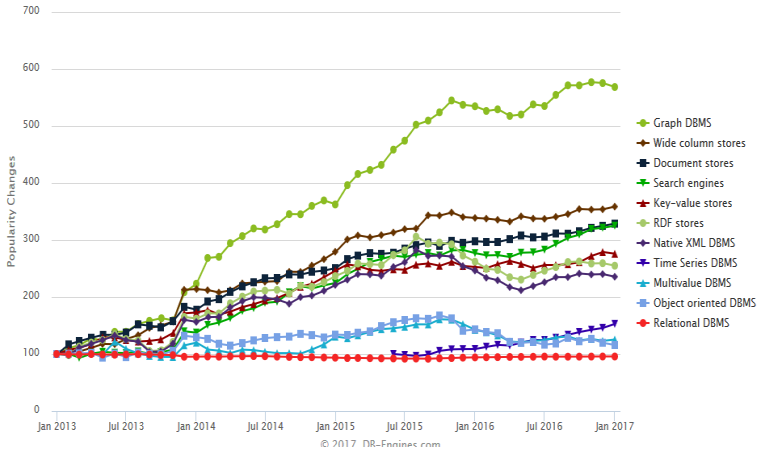


Figura 1.1: Evolução da Popularidade dos SGBDs de 2013 a janeiro de 2017 - Db-engines (2017)

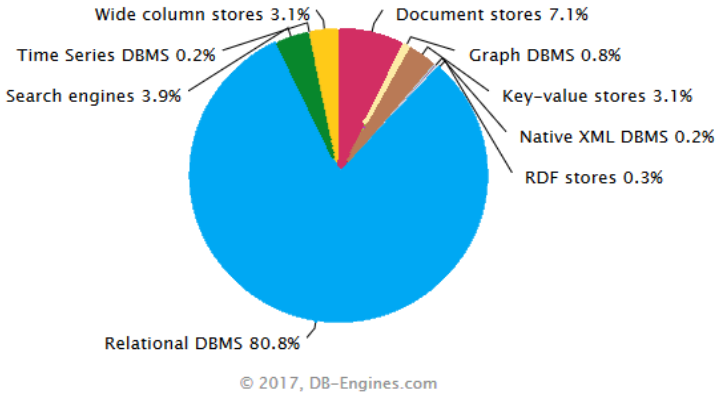


Figura 1.2: Ranking de Popularidade em Porcentagem dos SGBDs em janeiro de 2017 - Db-engines (2017)

1.1 Objetivo

Assim, o objetivo desse trabalho é analisar propostas de modelagem de bancos de dados voltados a grafos, comparar o seu desempenho a partir de uma mesma base original, modelada conforme cada uma das propostas, e, a partir dos pontos fortes de cada uma das modelagens, apresentar uma nova proposta de modelagem que explore as melhores características de cada uma delas.

1.2 Desenvolvimento

Este dissertação foi dividida em cinco capítulos, sendo o primeiro esta Introdução.

O capítulo 2, Fundamentação Teórica, busca contextualizar toda a parte teórica envolvida na modelagem dos bancos de dados voltados a grafos. Nesse sentido, parte de uma revisão das principais categorias de sistemas gerenciadores de bancos de dados desde a década de 50 do século passado até os bancos NoSQL e faz uma análise mais detalhada dos bancos de dados voltados a grafos.

Ainda são apresentadas quatro propostas de modelagem de dados para grafos: de Park et al. (2014) e Hunger (2015); Erven (2015), Lysenko et al. (2016) e Lampoltshammer e Wiegand (2015); Bordoloi e Kalita (2013b) e Virgilio, Maccioni e Torlone (2014), contendo seus pontos em comum e pontos divergentes.

Para melhor entendimento dessas propostas, é detalhada uma base de dados de teste, doravante chamada *BaseTeste*, que, a partir do seu modelo entidade relacionamento ou do modelo relacional, foi migrada para um modelo voltado a grafos, conforme as propostas de modelagem analisadas.

Posteriormente essas bases foram populadas com dados fictícios, e uma série de consultas foram aplicadas sobre elas e seus resultados analisados e comparados, permitindo determinar os pontos fortes e fracos de cada uma delas. A parte inicial do capítulo 4 traz esses testes.

O capítulo 3 discorre sobre a proposta objetivo desta dissertação, uma Modelagem Participativa para Bancos de Dados Voltados a Grafos, elaborada a partir das melhores características dos quatro métodos anteriormente analisados. Ainda nesse capítulo, será apresentado o resultado da migração da *BaseTeste* para um banco de dados orientado a grafos.

No final do capítulo 4, as mesmas consultas aplicadas às bases anteriores são agora executadas para o banco modelado na nova proposta e seu resultado comparado com os anteriores.

Em seguida, no capítulo 5 os resultados são expostos, possíveis trabalhos futuros são explanados e a conclusão final é apresentada.

Finalizando o trabalho, seguem dois apêndices, o primeiro com informações adicionais sobre os experimentos com a *BaseTeste* e o segundo com a modelagem e experimentos com um segunda base, a *BaseFilmes*.

2 Fundamentação Teórica

2.1 Modelo Entidade Relacionamento e Modelo Relacional

Muitas vezes confundidos como sinônimos, é preciso separar conceitualmente o que é modelo entidade relacionamento e o que é modelo relacional.

Chen (apud HEUSER, 2009) colocou que o projeto de um banco de dados, definição formal do que este conterá ou não e a forma como as informações serão armazenadas, usa dois níveis de abstração diferentes:

- **modelo conceitual** - mostra que dados serão guardados no banco de dados, os conjuntos de dados e os relacionamentos entre entre eles, independente do tipo de banco de dados onde será armazenado;
- **modelo lógico** - descreve a forma como estes dados serão armazenados, de acordo com o SGBD onde ele será implantado, isto é, define como o modelo conceitual será implementado.

Os dois modelos podem ser construídos de forma gráfica ou utilizando apenas texto. As figuras 2.1 e 2.2 trazem exemplos destes modelos nas duas formas.

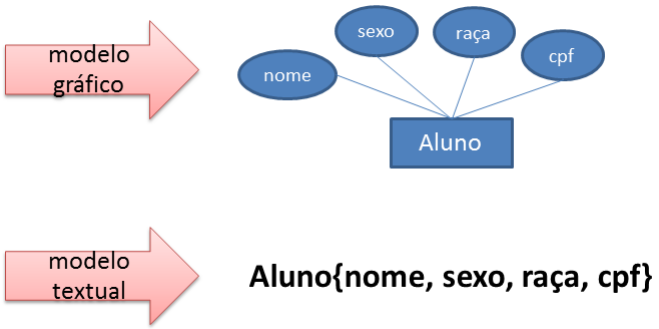


Figura 2.1: Modelo Conceitual - Exemplos

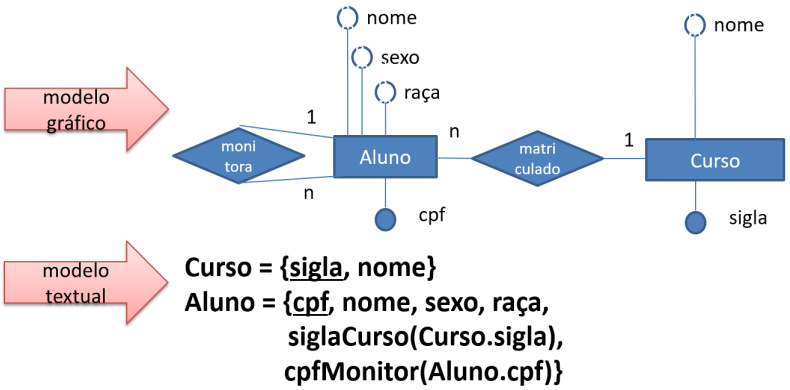


Figura 2.2: Modelo Lógico - Exemplos

Um exemplo de modelo conceitual é o modelo entidade relacionamento (MER) e um exemplo de modelo lógico é o modelo relacional (MR).

O primeiro, o modelo entidade relacionamento, pode ser conceituado como um grafo $G = E, R$, onde E, as entidades do MER, são os vértices do grafo e R, seus relacionamentos, são as arestas. Esse conceito é importante na análise, mais adiante, dos métodos de modelagem para bancos de dados voltados a grafos.

Já o modelo relacional é baseado no conceito matemático das relações, onde uma relação é o subconjunto do produto cartesiano entre dois ou mais conjuntos.

2.1.1 Modelo Entidade Relacionamento

Criado por Peter Chen em 1976 é muitas vezes considerado como um primeiro passo para a criação do modelo de um banco de dados relacional. Embora isso seja verdade, a partir do MER outros modelos lógicos também podem ser mapeados, como modelos para bancos de dados voltados a objetos e agora bancos de dados voltados a grafos.

Para Chen (1976), o mundo está cheio de coisas:

1. que podem ser reais ou abstratas;
2. que possuem características próprias e podem ser agrupadas em conjuntos;
3. que se relacionam entre si.

Com base nesse enunciado e sabendo que num modelo de banco de dados apenas devem ser moldadas as coisas (entidades), características (atributos) e relações entre elas (relacionamentos) que interessam ao modelo de negócio que será atendido, foi que Chen enunciou seu modelo conceitual, o Modelo Entidade Relacionamento (a).

O primeiro elemento de um *MER* é chamado `entidade`.

`Entidade`, conforme Chen e Heuser, é um conjunto de objetos do mundo real dos quais se deseja manter informações no banco de dados, e que pode ser agrupado por características comuns e distintas de outro objeto. Ele pode representar objetos concretos, como um aluno, ou objetos abstratos, como o curso em que este aluno está matriculado. No modelo de Chen, as entidades são representadas por um retângulo.

Conforme pode-se ver na figura 2.3, os alunos (objetos concretos) João, Maria, Mara e Yan serão agrupados na entidade `Aluno` e os cursos (objetos abstratos) serão agrupados na entidade `Curso`.

Figura 2.3: MER - Entidade

As entidades ainda podem ser classificadas em **fortes** e **fracas**. **Entidade forte** é uma entidade que existe por si mesma, independente de qualquer outra entidade. Por exemplo, a entidade `Aluno` é uma entidade forte. Já **entidade fraca** é uma entidade cuja existência depende de uma outra entidade, a entidade forte. Por exemplo, imagine uma entidade `Carro` que armazena os carros que serão utilizados pelo `Aluno` para entrar nos câmpus da universidade. Para existir carro autorizado pelo aluno precisa existir aluno, então `Carro` é uma entidade fraca. A figura 2.4 mostra estas entidades e seu relacionamento.

Figura 2.4: MER - Entidade Forte e Entidade Fraca

O segundo elemento é o `relacionamento`. O `relacionamento` conecta as entidades umas com as outras, da mesma maneira como ocorre no mundo real que esta sendo representado no *MER*. Por exemplo, pode-se querer saber quais são os alunos que estão matriculados em determinado curso, então, essa situação será representada pelo relacionamento `Aluno_Matriculado_Curso` (figura 2.5), simbolizado por um losango com linhas que une as entidades por ele conectadas. A exemplo das entidades, o relacionamento também pode ter atributos.

Figura 2.5: MER - Relacionamento

Uma característica importante dos relacionamentos é sua `cardinalidade`, isto é, a quantidade de ocorrências de uma entidade que podem estar relacionadas, naquela relação, com uma determinada ocorrência de outra entidade.

Para cada lado do relacionamento existe a cardinalidade mínima e máxima.

A `cardinalidade mínima` indica o menor número de ocorrências da entidade que participam do relacionamento. Pode ser:

- 0 (zero) - indica que o relacionamento não é obrigatório;
- 1 (um) - indica que o relacionamento é obrigatório e pelo menos uma ocorrência da entidade deve participar dele;
- n - indica que o relacionamento é opcional e pode existir mais de uma ocorrência da entidade participando do relacionamento.

A `cardinalidade máxima` indica o maior número de ocorrências da entidade que podem participar do relacionamento. Pode ser:

- 1 (um) - uma e apenas uma ocorrência da entidade participa do relacionamento;
- n - indica que mais de uma ocorrência da entidade pode participar do relacionamento.

Na prática é colocada apenas a cardinalidade máxima do lado da relação, sendo omitida a mínima. Assim, uma relação de cardinalidade mínima 0 (zero) e máxima 1 em vez de 0..1 é representada apenas por 1.

A figura 2.6 traz um exemplo de relação, onde o lado do `Aluno` no relacionamento `Matriculado`(1..1) indica que o aluno deve obrigatoriamente estar matriculado em um curso. Já o lado do `Curso` (0..n), indica que um curso pode ter quaisquer quantidade de alunos nele matriculado, inclusive nenhum.

Com base na `cardinalidade máxima` os relacionamentos podem ser classificados em:

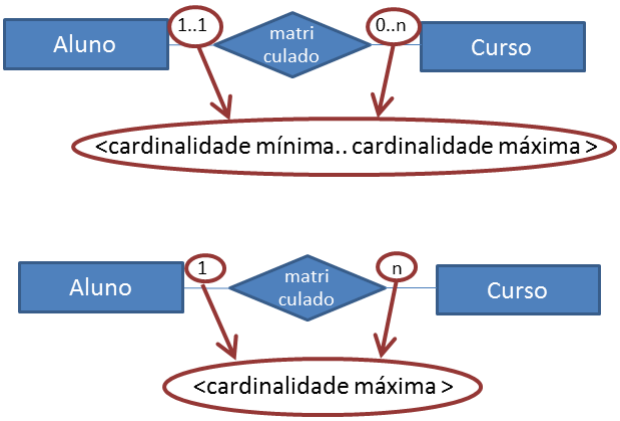


Figura 2.6: MER - Cardinalidade

- relacionamentos $1:1$ (um para um) - se a relação R entre as entidades X e Y é do tipo $1:1$, isso quer dizer que uma ocorrência de X está ligada a apenas uma ocorrência de Y e uma ocorrência de Y está ligada a apenas uma ocorrência de X;
- relacionamentos $1:n$ (um para muitos) - se a relação R entre as entidades X e Y é do tipo $1:n$, isso quer dizer que uma ocorrência de X pode estar ligada a uma ou mais ocorrências de Y, mas Y está ligado a apenas uma ocorrência de X;
- relacionamentos $m:n$ (muitos para muitos) - se a relação R entre as entidades X e Y é do tipo $m:n$, isso quer dizer que uma ocorrência de X pode estar relacionada com uma ou mais ocorrências de Y, e o mesmo se aplica a Y, uma ocorrência de Y pode estar ligada a várias ocorrências de X.

A figura 2.7 traz um exemplo da representação dos três tipos de relacionamentos:

- `Aluno TEM DNA` é do tipo $1:1$ porque 1 `Aluno` pode ter apenas 1 `DNA`, e 1 `DNA` é de apenas 1 `Aluno`;
- `Aluno ESTÁ COM Livro` é do tipo $1:n$ porque 1 `Aluno` pode estar com vários (n) `Livros` (da Biblioteca) mas 1 `Livro` pode estar com apenas 1 `Aluno`;
- `Aluno LEU Livro` é do tipo $m:n$ porque 1 `Aluno` pode ter lido vários `Livros` e 1 `Livro` pode ter sido lido por vários `Alunos`.

Os relacionamentos também podem ser classificados por grau. Ainda observando a figura 2.7, vê-se que os três relacionamentos, apesar de cardinalidades diferentes, têm uma característica em comum: todos conectam `duas` entidades. Esse é o `relacionamento binário` ou de `grau 2`, o mais comum dos tipos de relacionamento quando eles

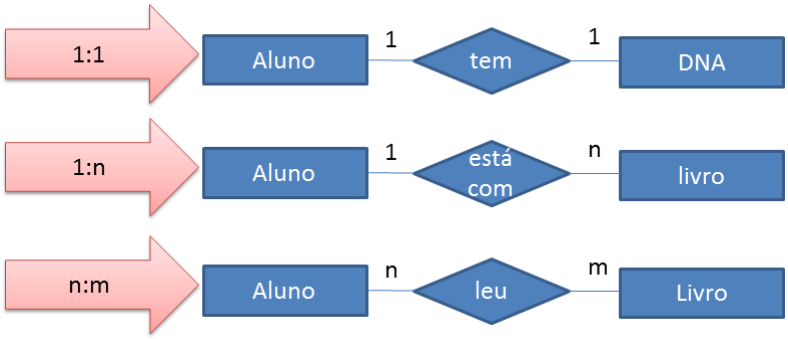


Figura 2.7: MER - Relacionamento

são classificados pelo número de entidades conectadas. No exemplo, o relacionamento `Tem` conecta `Aluno` e `DNA`, `Está com` e `Leu` conectam `Aluno` e `Livro`.

O próximo é o relacionamento de `grau 1`, ou `unário`, figura 2.8, aquele que tem apenas uma entidade envolvida, ou seja, uma ocorrência da entidade está relacionada com outra ocorrência da própria entidade.

Na figura vê-se três exemplos que envolvem apenas ocorrências dentro da entidade `Aluno`, um `Aluno` forma dupla com outro `Aluno`, um `Aluno` pode ser monitor de outros n `Alunos`, e `Alunos` estudam com diversos `Alunos` que também estudam com diversos `Alunos`.

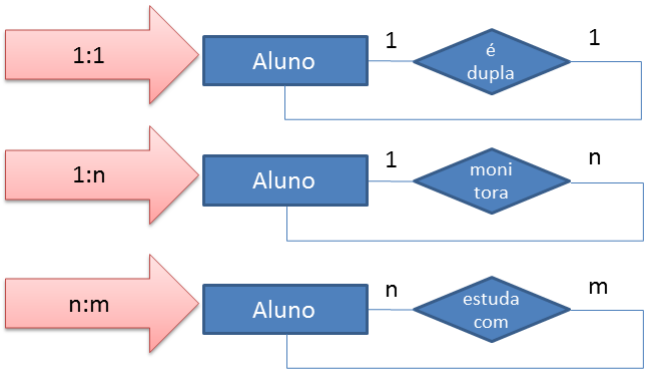


Figura 2.8: MER - Relacionamento Unário

O relacionamento `terciário` ou `ternário` envolve três entidades numa mesma relação, conforme pode-se ver na figura 2.9, onde a relação `Cursa` indica que um `Aluno` pode cursar diferentes `Disciplinas` em diferentes `Câmpus`.

Podem existir outros tipos de relações com quatro ou mais entidades envolvidas, mas são bastante semelhantes à terciária, apenas mudando a quantidade de entidades conectadas ao relacionamento.

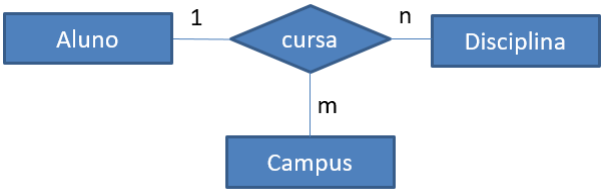


Figura 2.9: MER - Relacionamento Ternário

O próximo elemento do MER é o `atributo`. É definido como o elemento que contém uma propriedade de uma entidade. Por exemplo, a entidade Aluno tem o atributo `nome` que o identifica. É representado, no MER, por uma elipse ligada por uma linha à entidade a qual pertence.

A exemplo dos relacionamentos, os atributos também podem ter cardinalidade, expressa na forma de cardinalidade mínima e máxima.

A cardinalidade mínima de um atributo pode ser 0, indicando que o atributo é opcional, ou 1, indicando que o atributo é obrigatório.

A cardinalidade máxima pode ser 1 dizendo que há apenas um valor para o atributo por ocorrência da entidade (monovalorado) ou n indicando que o atributo é multivalorado, ou seja, pode haver mais de um valor para o atributo numa mesma ocorrência da entidade.

Na prática, a cardinalidade $(1, 1)$, obrigatório e monovalorado, é omitida no diagrama.

Normalmente a descrição dos atributos é feita de forma textual, fora do diagrama, pois podem existir entidades com muitos atributos e então fica difícil colocá-los todos de forma clara no MER gráfico.

A figura 2.10 traz o exemplo da entidade `Aluno` e dois atributos monovalorados, `nome` e `sexo` e um opcional e multivalorado, `fone`.

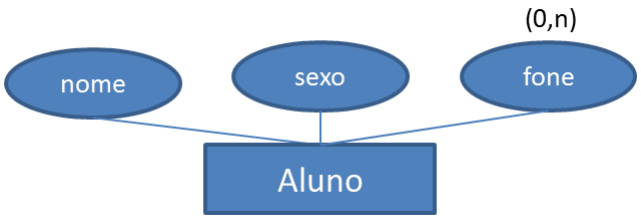


Figura 2.10: MER - Atributo

Outras característica do MER, como os tipos de atributos, por não serem relevantes ao discutido nesta dissertação, não são aqui apresentadas.

2.1.2 Modelo Relacional

Para Codd (apud HEUSER, 2009) o Modelo Relacional é um modelo formal, preciso e não ambíguo, de tal maneira que qualquer pessoa que o ler terá sempre o mesmo entendimento.

Esse modelo revolucionou a forma como os dados são estruturados, pois propõe que dados de mesma natureza sejam organizados em forma de tabelas bi-dimensionais, do tipo linhas X colunas. Na proposta, ainda, a interação entre esses diferentes dados (tabelas) é expresso na forma de relações cartesianas.

Ele pode ser criado a partir do modelo entidade relacionamento quando o banco de dados a ser implantado é o relacional.

Nesta dissertação, interessa para o estudo o passo a passo no tratamento das entidades, atributos e relacionamentos do MER, não sendo relevante a tipagem dos dados.

O primeiro elemento do MER a ser tratado é a `Entidade`. Para cada entidade forte do MER será criada uma tabela onde:

- o nome da Entidade dará o nome para a Tabela. Se não for possível, ou, seguindo o padrão de nomenclatura do local, um nome que remeta ao nome da entidade;
- os atributos `monovalorados` da entidade serão colunas da tabela;
- caso no MER já tenha sido estabelecido um atributo como atributo identificador dos registros da tabela, este atributo será a chave do registro(chave primária, doravante chamada PK). Caso não tenha sido estabelecido, deve-se verificar entre os atributos se algum deles pode servir como identificador e daí defini-lo como PK, do contrário deve-se criar um novo atributo para servir a esse fim. Nada impede que a PK seja composta por dois ou mais atributos;
- outras colunas que também podem individualizar os registros da tabela mas que não foram escolhidas como PK, podem ser marcados no modelo como Chaves Alternativas ou AK, indicando que poderá ser gerado um índice de acesso a tabela por essa coluna;
- os nomes das colunas devem seguir as normas de nomenclatura do local, mas de forma a facilitar a identificação de que o atributo Y pertence à tabela T.

Antes de prosseguir com a conversão dos relacionamentos, é preciso definir o conceito

de `chave estrangeira` (FK). FK denota a origem estrangeira da coluna, que é migrada a partir de outra tabela, onde é chave primária.

A conversão dos `relacionamentos unários` segue os seguintes procedimentos:

- nos relacionamentos $1:1$ e $1:n$ - cria-se uma coluna do mesmo tipo da PK da tabela, repetindo seu nome mais um sufixo, indicando ser FK, definindo-a como um atributo FK para a própria tabela;
- nos relacionamentos $m:n$ - cria-se uma nova tabela (*joined table*) para representar este relacionamento, incluindo como colunas duas FKs da chave primária da tabela. A PK desta nova tabela será uma chave composta pelas duas FKs.

Ja na conversão dos `relacionamentos binários` deve-se:

- nos relacionamentos $1:1$ - identificar, entre as duas tabelas participantes da relação, aquela que é a principal (ou forte) e qual é a secundária (ou fraca) entre as duas. Criar, então, na tabela secundária uma coluna FK com a PK da tabela principal fazendo desta nova coluna a PK da tabela secundária. Em resumo, as duas tabelas passarão a ter a mesma PK;
- nos relacionamentos $1:n$ - criar na tabela do lado n uma coluna com a PK da tabela do lado 1, e defini-la como um atributo FK para a tabela do lado 1;
- nos relacionamentos $m:n$ - criar uma nova tabela (*joined table*) para representar este relacionamento e incluir como colunas duas FKs com a PK da tabela do lado m e com a PK da tabela do lado n, fazendo da composição destas duas chaves a chave primária da tabela.

O processo para conversão dos relacionamentos ternários, e de todos aqueles com mais de duas entidades envolvidas, é semelhante ao processo das relações binárias $m:n$, ou seja, deve-se criar uma tabela (*joined table*) para representar o relacionamento e uma FK com a PK de cada uma das entidades participantes. A PK da nova tabela será composta pelas FKs criadas.

A conversão dos atributos é mais simples e segue os seguintes passos:

- os atributos monovalorados serão colunas da tabela gerada a partir da entidade ou relacionamento a que pertencem;

- para os atributos multivalorados deve ser criada uma nova tabela com uma coluna capaz de guardar uma ocorrência do atributo, tornado-o monovalorado nesta tabela e outra coluna com a PK da tabela origem. Entretanto, a PK da nova tabela precisa ser composta, ou pela composição das duas colunas criadas, ou pela criação de uma terceira coluna, que irá armazenar um número sequencial e esta junto com a FK comporá a PK da nova tabela.

A figura 2.11 traz um exemplo de modelo relacional gerado a partir dos exemplos utilizados no tópico sobre o MER.

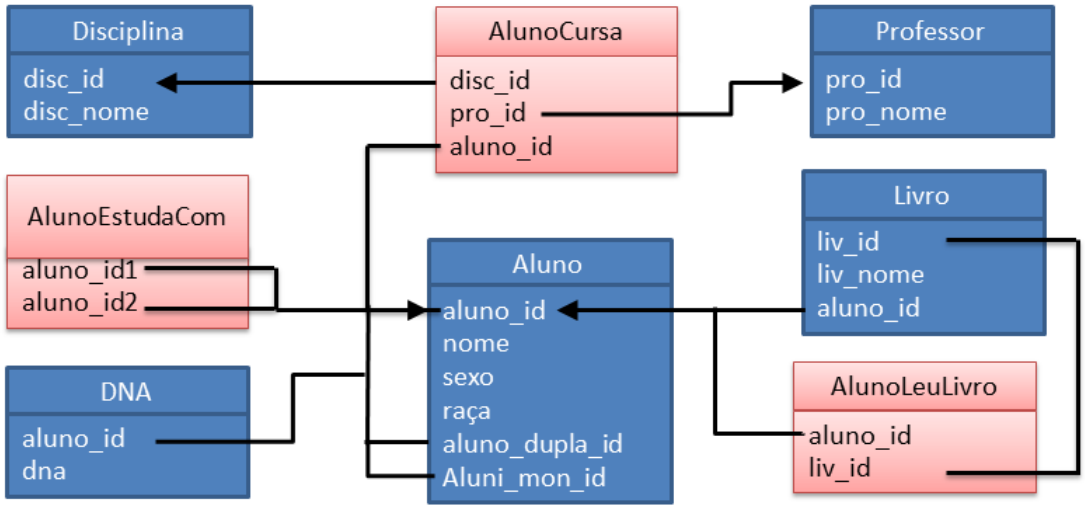


Figura 2.11: Modelo Relacional - Exemplo

2.2 Retropesctiva dos Sistemas Gerenciadores de Bancos de Dados

Revisando a história dos bancos de dados desde os anos 50 do século XX, Fowler e Sadalage (2013) e Lake e Crowther (2013) mostram que desde o início a persistência dos dados processados nos computadores foi uma preocupação constante.

No final da década de 60, com o surgimento dos primeiros discos para armazenamento dos dados e consequente aumento na capacidade de armazenamento, passou a ser possível gravar e recuperar dados. Surgiram os primeiros modelos de armazenamento como o modelo de banco de dados hierárquico e o modelo em rede. O modelo hierárquico, figura 2.12, tem uma estrutura em árvores, onde nos ramos finais ficam os registros, o que obriga acessá-los sequencialmente, tornando a operação de acesso aos dados extremamente

demorada. Já a estrutura em redes, figura 2.13, é uma extensão do modelo hierárquico, onde os dados são representados por coleções de registros que podem se relacionar com um ou mais registros da base. A grande desvantagem desses dois modelos é que os registros tinham que ser gravados na base conforme a ordem desejada para recuperação dos dados.

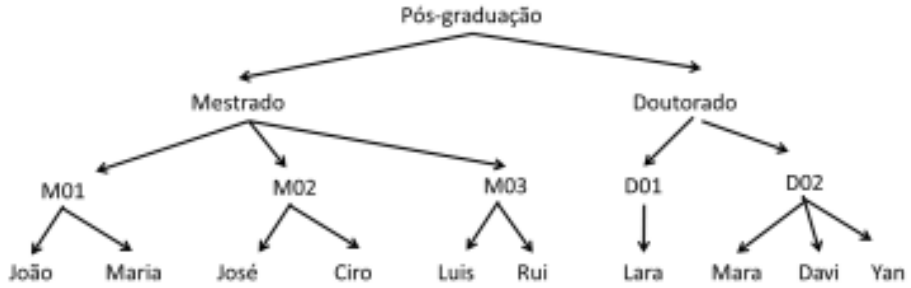


Figura 2.12: Modelo Hierárquico

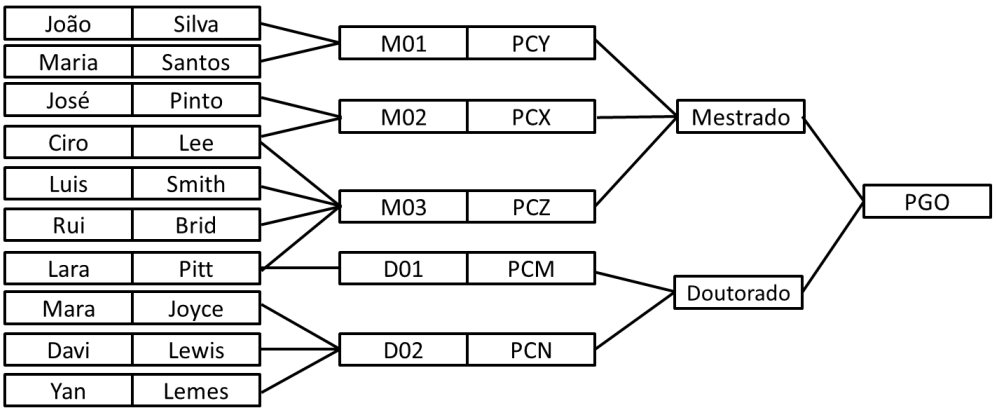


Figura 2.13: Modelo em Redes

Em 1970 Codd (1970) apresentou em um seminário a base do que ficou conhecido como modelo relacional, visto no tópico 2.1.2, o qual revolucionou o armazenamento de dados, por tornar o acesso a eles independente da forma como eram gravados.

IBM DB2, Oracle, MySQL e PostgreSQL são SGBDs que foram implementados conforme o modelo de Codd. Eles, como outros bancos relacionais, apresentam como característica comum a implementação das propriedades ACID[1] e uma estrutura de dados fixa, onde todos as linhas (registros) de uma tabela têm os mesmos atributos, todos com a mesma tipagem, garantindo que qualquer registro terá sempre todos os atributos, mesmo que, eventualmente, sem valor ou valor nulo.

[1]Uma transação em banco de dados deve ter Atomicidade (ou ela executa tudo ou não executa nada), Consistência (o banco deve estar consistente antes e depois dela), Isolamento (uma vez iniciada a transação outras transações não devem interferir nela) e Disponibilidade (o resultado dela deve persistir até que outra nova transação altere os dados).

A figura 2.14 traz um exemplo do modelo relacional onde, como se pode ver, o relacionamento entre tabelas é expresso pela relação (‘chave primária’,‘chave estrangeira’). Na figura, por exemplo, o atributo ‘are’, da tabela Subarea, funciona como chave estrangeira, apontando para o registro de ‘chave primária’ da tabela Area.

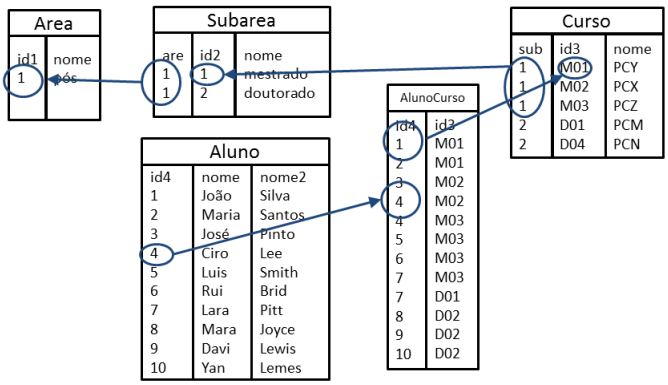


Figura 2.14: Modelo Relacional

Por permitir um grande grau de independência dos dados e das aplicações, flexibilidade na ordenação e tipagem dos dados das colunas de cada tabela, prever técnicas de normalização que ajudavam a diminuir a redundância dos dados, integridade dos dados e fácil manipulação e acesso aos dados rapidamente esse modelo tornou-se um padrão, dominando a área de bancos de dados até hoje.

Entretanto, com a expansão no uso dos computadores para praticamente todas as atividades humanas, na opinião de muitos, o modelo relacional vem se revelando um modelo engessado, com problemas devido a algumas limitações como a dificuldade em, por exemplo, tratar de objetos mais complexos ou a falta de escalabilidade horizontal, isto é, a dificuldade em dividir o banco de dados em diferentes computadores, limitando o tamanho das bases de dados à capacidade de armazenamento do servidor. Já outros vêm sustentando que isso não é verdade uma vez que novas versões dos sistemas gerenciadores de bancos de dados relacionais têm trazido novas funcionalidades que atendem essas novas necessidades. Esses novos bancos estão sendo conhecidos como *NewSQL*. Em contraponto ao modelo relacional, outras concepções de modelos de dados têm surgido.

Brewer (2000) e Pritchett (2008) lançaram, em contraponto à necessidade das propriedades ACID para bancos de dados, o teorema de CAP e as propriedades BASE, onde, em resumo, é colocado que dentre as propriedades Consistência forte, Alta disponibilidade e Particionamento dos dados na rede somente duas delas podem ser garantidas ao mesmo tempo, devendo a aplicação ser capaz de corrigir eventual falha de uma delas após o fim da transação no banco. Do teorema CAP vem as propriedades BASE para os bancos

de dados, que consistem no fato do banco de dados estar Basicamente (*Basically*) todo o tempo disponível, não precisando ser consistente o tempo todo (*Soft State*) e que ele ficará consistente no momento devido (*Eventually consistent*) .

Baseado nesses princípios surgiram os chamados bancos NoSQL (*Not Only SQL*), que mantendo a ideia de relação entre elementos do bancos de dados, daí SQL, incorporam outras características que diminuíram o engessamento do banco, como não obrigatoriedade de atributos, ou possibilidade de particionamento dos dados em *clusters*, trazendo escalabilidade horizontal, daí o *Not Only*.

A principal característica dos bancos NoSQL é a ausência de esquema, *schema-free*, onde os registros são basicamente entradas no formato chave-valor, ou seja, uma chave, formada por um valor qualquer único no banco que identifica um valor de qualquer tipo.

De acordo com as propriedades das transações, os bancos NoSQL podem ser classificados em:

- sistemas CP - são sistemas que priorizam alta consistência e possibilidade de particionamento dos dados em diversos servidores, mesmo que eventualmente haja atraso na disponibilidade de um dado;
- sistemas CA - são sistemas que priorizam alta consistência e alta disponibilidade de dados, mesmo que isso restrinja a divisão das bases em mais de um servidor;
- sistemas AP - são sistemas que priorizam alta disponibilidade de dados e possibilidade de particionamento dos dados em diversos servidores, mesmo que momentaneamente possa haver uma falta de consistência dos dados.

Conforme noSql.org (2017), quanto à forma de armazenamento de dados, os bancos NoSQL podem ser classificados em:

- chave-valor (*key-value* ou *tuple store*) - implementado através de uma tabela *hash* na qual há uma chave única identificando um determinado valor conforme mostra o exemplo da figura 2.15. São exemplos: DynamoDB[2], Ryak[3], Tokyo Cabinet[4], Voldemort[5];

[2]http://aws.amazon.com/dynamodb/
[3]http://basho.com/products/#riak
[4]http://fallabs.com/tokyocabinet/
[5]http://project-voldemort.com/

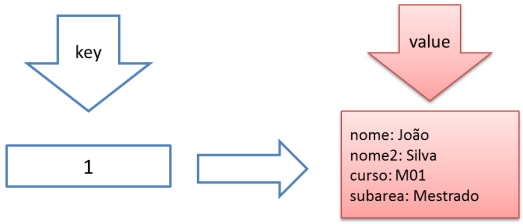


Figura 2.15: Exemplo de Banco NoSQL Key-Value

- orientado a coluna (*wide column*) - uma evolução da forma chave-valor onde as chaves podem apontar para atributos com múltiplas colunas, sendo os dados identificados por uma chave tripla, formada por linha, coluna e *timestamp*, onde este último serve para identificar as múltiplas versões de um mesmo dado. Cassandra[6], HyperTable[7] e Haddop/HBase[8] são exemplos de bancos *wide column*, cujo modelo pode ser visto na figura 2.16;

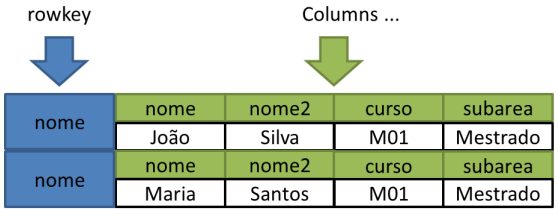


Figura 2.16: Exemplo de Banco NoSQL WideColumn

- orientado a documentos - Conforme se vê na figura 2.17 este modelo, como o nome diz, armazena documentos que são objetos compostos por um identificador e um conjunto de campos que podem ser do tipo *string*, listas ou mesmo outros documentos. Aqui tem-se a aplicação prática do sem esquema, pois mesmo documentos armazenados de mesmo tipo dentro do banco podem ter atributos diferentes. Mesmo na ausência de um valor para um atributo não é necessário incluir o atributo com valor nulo, basta não incluí-lo no documento. São exemplos: MongoDB[9], CouchDB[10], TerraStore[11], SimpleDB[12];

- orientado a grafos (*graph databases*) - este modelo parte da ideia de representar os dados e seus relacionamentos como grafos dirigidos, valorizando mais o relaciona-

[6]https://cassandra.apache.org/
[7]http://hypertable.org/
[8]http://hadoop.apache.org/
[9]http://www.mongodb.org/
[10]http://couchdb.apache.org/
[11]http://code.google.com/p/terrastore/
[12]http://aws.amazon.com/simpledb/

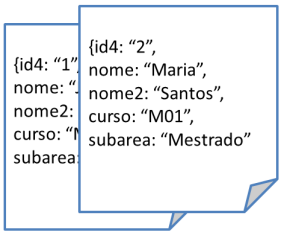


Figura 2.17: Exemplo de Banco NoSQL Orientado a Documentos

mento entre os dados do banco do que o dado em si, dentro da premissa de permitir descobrir a informação escondida entre os dados do que simplesmente retornar esses dados. Como mostrado na figura 2.18, ele possui basicamente três tipos de elementos: os vértices (ou nós), as arestas (relacionamentos entre os nós) e atributos dos nós ou arestas, sendo que cada vértice ou aresta é sem esquema, ou seja, não precisam ter os mesmos atributos. São exemplos: Neo4J[13], Infinite Graph[14], TITAN[15], AlegroGraph[16]. Uma vez que a modelagem de bancos de dados voltados a grafos é o objetivo deste trabalho, este modelo será melhor detalhado mais adiante;

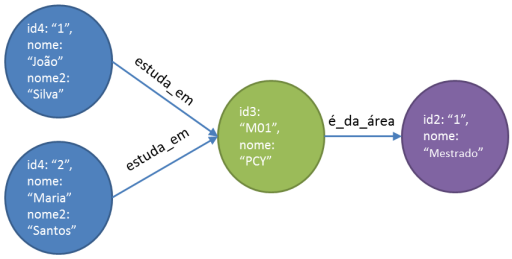


Figura 2.18: Exemplo de Banco NoSQL Orientado a Grafos

- multimodelos (*multimodel*) - conforme representado na figura 2.19 são bancos que têm características de dois ou mais outros modelos, por exemplo, o *OrientDB* é orientado a grafos e seus vértices ou arestas são documentos. São exemplos: OrientDB[17], ArangoDB[18].

A figura 2.20 traz um gráfico com os principais bancos NoSQL agrupados conforme as classificações apresentadas.

[13]http://www.neo4j.org/
[14]http://www.infinitegraph.com/
[15]https://github.com/thinkaurelius/titan/wiki
[16]http://www.franz.com/agraph/
[17]http://orientdb.com/
[18]https://www.arangodb.com/

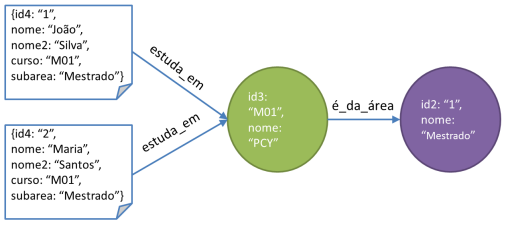


Figura 2.19: Exemplo de Banco NoSQL Multimodelos

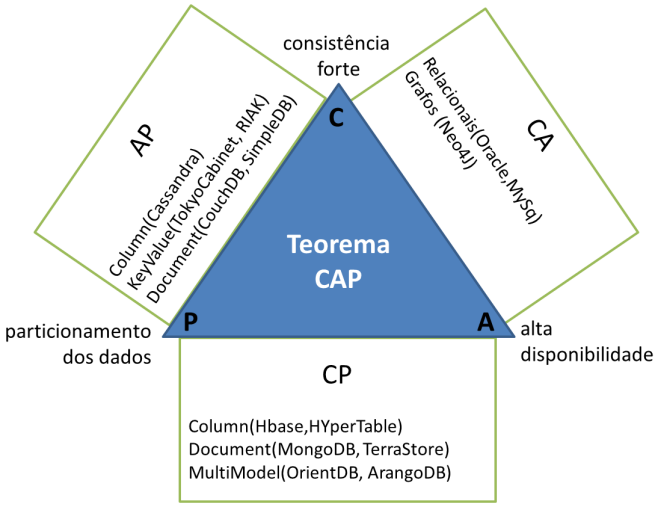


Figura 2.20: SGBDs X Teorema de CAP (adaptado de Vale (2015))

Em contraponto ao movimento NoSQL surgiram os chamados bancos NewSQL. Conforme Moniruzzaman (2014), NewSQL são sistemas gerenciadores de bancos de dados que possuem as caracteríticas dos chamados bancos relacionais (RDBMS), acrescidos de características dos bancos NoSQL, como tipos de dados voltados ao *Big Data*, como campos JSON, escalabilidade horizontal e maior velocidade nas operações. NuoDB[19] e VoltDB[20] são exemplos de bancos NewSQL. Além disso, bancos de dados relacionais mais tradicionais como o MySQL[21] e o PostgreSQL[22] também têm disponibilizado novos tipos de dados voltados ao *Big Data*, como campos JSON.

2.3 Bancos de Dados Voltados a Grafos

A retrospectiva dos diferentes modelos de bancos de dados se faz necessária para poder situar o objetivo deste trabalho, que é apresentar uma proposta de modelagem de dados para bancos de dados voltados a grafos, *graph databases*.

[19]http://www.nuodb.com/
[20]https://www.voltdb.com
[21]https://www.mySQL.com/, a partir da versão 5.7.8
[22]https://www.postgreSQL.org/,a partir da versão 9.3

Nesse intuito, a primeira questão é detalhar melhor os bancos de dados voltados a grafos, ou simplesmente, bancos de dados de grafos (`BDGs`).

Para Fowler e Sadalage (2013), bancos de dados de grafos são bancos que permitem que sejam armazenadas entidades e os relacionamentos entre essas entidades na forma de, respectivamente, vértices (ou nós) e arestas.

Os nós são entidades que podem ter propriedades. Eles devem possuir pelo menos uma propriedade que será o seu identificador. Normalmente esse identificador é atribuído pelo SGBDG. O nó Aluno, por exemplo, pode ter as propriedades Nome e Data de Nascimento.

As arestas são entidades especiais que têm, automaticamente três propriedades pelo menos: um identificador único, atribuído pelo SGBDG; e dois ponteiros, um ponteiro apontando para uma entidade do tipo nó que é a origem (ou ponto de partida) da aresta; e outro ponteiro apontando para a entidade destino da aresta, também do tipo nó. Uma entidade do tipo aresta pode possuir outras propriedades, da mesma maneira que as entidades nós. Ainda, não há limite, nem mínimo nem máximo, para o número de arestas partindo de uma entidade nó. E, também, podem existir mais de uma arestas unindo os mesmos nós.

Num primeiro momento, quando comparados os BDGs com bancos relacionais, quanto à estrutura de seus elementos, poderia-se dizer que os vértices são tabelas sem chaves estrangeiras e as arestas são os relacionamentos, entretanto, isso não é verdade. Na realidade, BDGs, nesse contexto, tem apenas duas grandes tabelas, que pode-se chamar de tabela de vértices e de tabela de arestas. Quando, por exemplo, as tabelas relacionais X, com 500 registros, e Y, com 200 registros, são migradas para um BDG, não vão existir o vértice X no BDG com 500 elementos e o Y com 200 elementos, e sim 700 elementos na tabela de vértices, 500 elementos com as propriedades de X e 200 com as propriedades de Y.

Ainda, conforme Angles (2012), os BDGs podem ser classificados quanto ao tipo de grafos por eles suportados:

- grafos simples: os vértices têm apenas uma propriedade que o identifica e as arestas têm, basicamente, apenas os ponteiros para os vértices origem e destino Deve-se observar que não se aplica aqui a definição de grafo simples da Teoria dos Grafos que diz que um grafo é simples se ele não tem laços nem mais de uma aresta ligando dois vértices;
- hipergrafos: grafos onde uma mesma aresta pode conectar mais de dois vértices;

- grafos aninhados: grafos onde um vértice pode ele mesmo ser outro grafo;
- grafos de propriedades: grafos onde vértices e arestas podem conter atributos para descrever suas propriedades;

As modelagens para grafos analisadas neste dissertação tratam apenas dos grafos simples ou de propriedades. Para estes dois tipos de grafos, com relação as propriedades das arestas, pode-se dizer que:

- uma aresta não pode ter uma propriedade apontando para outra entidade além dos ponteiros origem e destino;
- os ponteiros devem obrigatoriamente apontar para uma entidade do tipo nó. A entidade apontada deve existir. Caso uma das entidades apontadas seja excluída, antes dessa operação a aresta deverá ser, também, excluída;
- os ponteiros origem e destino podem, conforme a aplicação, apontar para a mesma entidade.

Um conceito adicional é o uso dos BDGs como um serviço do sistema de bancos de dados. Park et al. (2014) diferenciam `gerenciamento de dados` de `serviço de dados`. Para eles, `gerenciamento de dados` trata do armazenamento de dados, onde a preocupação maior está na persistência e não redundância dos dados, numa estrutura tolerante a erros, onde os dados são modelados para armazenamento seguindo as regras de normalização de Codd (1970) e então, no dia a dia da aplicação, dados são incluídos, alterados e/ou excluídos. Já `serviço de dados` trata de minerar os dados armazenados no banco, através de consultas analíticas, que envolvem selecionar, unificar e extrair dados. Quando da mineração dos dados, muitas vezes é necessário desnormalizar esses dados de forma a facilitar as pesquisas. Assim, quando da execução de um serviço de dados, os dados necessários a sua execução são importados pelo serviço, minerados e após a apresentação do resultado, descartados.

Nesse sentido, os BDGs vem se firmando como bancos de dados propícios ao serviço de dados, pois eles podem ser implementados como estruturas paralelas a sistemas de gerenciamento de dados, de forma que, dinamicamente, os dados armazenados em bases relacionais, por exemplo, são modelados e exportados para BDGs de forma a complementar o serviço de dados.

2.4 Neo4J

Conforme Neo4j (2017), o banco de dados Neo4j é um robusto modelo que apresenta características marcantes de escalabilidade e desempenho para mapeamento, armazenamento e caminhamento de grafos, aplicável em situações em que se tem entidades e relacionamentos entre elas.

Ele implementa a idéia do grafo de propriedades, ou seja, os seus elementos nós e arestas podem conter outros atributos para descrever suas propriedades.

Ele apresenta:

- alta disponibilidade;
- dimensionamento para bilhões de nós e relacionamentos;
- alta velocidade de consulta através de caminhamentos;
- linguagem de consulta declarativa de grafos.

Conforme Neubauer (2013), Neo4j implementa o controle ACID de transações, estando no lado CA do triângulo CAP (figura 2.20), ao contrário dos outros bancos NoSQL que ficam do lado CP (VALE, 2015).

O Neo4j se apresenta como um banco de dados em que todas as operações que modificam dados ocorrem dentro de uma transação, garantindo dados consistentes, ao mesmo tempo que se propõe ter escalabilidade horizontal, isto é, poder ser distribuído em diversos *clusters*.

Um grafo pode variar em tamanho e complexidade com a evolução da aplicação, e, em Neo4j, isso deve gerar um baixo impacto no desempenho da base de dados e na aplicação como um todo. Normalmente Neo4j só é limitado pelo *hardware* físico, explorando ao máximo principalmente os recursos de memória para garantir a agilidade em consultas e caminhamentos.

Um único servidor configurado com a base de dados Neo4j pode lidar com um grafo de bilhões de nós e relacionamentos, mas quando o tratamento de dados for insuficiente, esta base de dados pode ser distribuída entre vários servidores em uma configuração de alta disponibilidade, garantindo o desempenho.

O armazenamento de grafos apresenta melhor desempenho ao armazenar dados ricamente conectados. A consulta é realizada através de caminhamentos e indexações, que podem realizar milhões de passos por segundo.

São características do Neo4J:

- índices - permite a criação de índices sobre uma propriedade para todos os nós que tenham um determinado atributo. Uma vez criado um índice, ele será automaticamente gerenciado, atualizado e utilizado pelo banco de dados sempre que o grafo for alterado;
- linguagem Cypher - é a linguagem oficial de consultas do Neo4j e permite que se crie, modifique e procure dados em uma estrutura baseada em um grafo de informações e relacionamentos;
- integridade - o sistema de controle do Neo4j provê mecanismos para garantir a integridade tanto para os elementos do grafo (nós, relacionamentos e propriedades) como para dados que não são do grafo, como os índices. Sua arquitetura transacional garante a proteção dos dados e sua recuperação rápida no caso de uma falha inesperada, sem a necessidade de reconstruir índices internos ou outras operações dispendiosas;
- integração - Neo4j pode ser combinado com uma base relacional, sendo usado como uma extensão para complementar buscas para a tomada de decisão mais rápida. Essa sincronização pode ser de duas maneiras:
  - sincronização *on-line* - uma mesma transação na aplicação faz a atualização no banco relacional e no banco Neo4j, onde esta última base funciona como uma replicação da primeira;
  - sincronização posterior - efetuada de maneira assíncrona, normalmente trabalha com a exportação dos dados da base relacional via uma consulta *SQL* que gera um arquivo que será importado pela base Neo4j;
- disponibilidade e confiabilidade - prevê três formas diferentes de abordagem:
  - na abordagem *Cold Spare*, ou 'Backup Frio', é executado um *backup on-line* de uma instância do banco. Em caso de falha, uma nova instância é criada a partir dos dados salvos e daí as operações feitas no banco entre o momento do *backup* e o momento da falha deverão ser refeitas;

- na abordagem *Hot Spare*, ou 'Backup Quente', é feito um *backup* com as transações efetuadas a partir de uma instância do banco. Em caso de falha, essa instância é restaurada e as transações refeitas;
- Na abordagem *Cluster* de alta disponibilidade, o sistema prevê a distribuição dos dados em um conjunto de servidores, sendo um deles considerado a instância mestre e os demais instâncias escravas. A atualização *on-line* é feita apenas na instância mestre e nas demais são efetuadas apenas leituras, com posterior atualização. Caso uma instância escrava falhe, esta é simplesmente abandonada. Caso a instância mestre falhe, uma nova instância mestre é eleita entre as escravas.

No tocante à segurança dos dados cabe ressaltar que o Neo4j não faz criptografia de dados explicitamente, então cabe à aplicação trabalhar para proteger esses dados, seja gravando-os previamente criptografados, ou provendo um meio de permissão ou restrição de acesso.

O Neo4j foi o SGBDG utilizado nos experimentos com os métodos de modelagem abordados nesta dissertação.

2.5 Modelagem de Bancos de Dados de Grafos

Nesta seção são analisados quatro métodos de modelagem para bancos de dados voltados a grafos.

Como todos os métodos analisados partem, ou do modelo entidade relacionamento ou do modelo relacional de uma base para fazer a modelagem e para melhor entendimento do passo a passo de cada método, será apresentada aqui uma base de dados que será migrada para o modelo de grafos conforme cada abordagem.

2.5.1 Modelo Base (BaseTeste)

Para esse trabalho foi idealizada uma base que em suas características expusesse nas modelagens as diferenças entre cada método a fim de permitir uma melhor análise dos mesmos.

O modelo prevê oito entidades, denominadas A, B, C, D, E, F, G e H.

As relações entre essas entidades serão nomeadas da seguinte maneira:

- relações binárias: nome da primeira entidade em maiúsculo, seguido da expressão inglesa *go* em minúsculo e do nome da segunda entidade em maiúsculo. Por exemplo, o relacionamento entre as entidades D e A será nomeado $DgoA$;

- relações terciárias: nome da primeira entidade em maiúsculo, seguido da expressão inglesa *go*, seguido do nome da segunda entidade em maiúsculo, novamente *go* e o nome da terceira entidade em maiúsculo. Por exemplo, o relacionamento entre as entidades D, E e G será nomeado $DgoEgoG$;

- no caso de existirem dois relacionamentos diferentes entre duas entidades, será acrescido 1 ou 2 para diferenciá-las. Por exemplo, existindo dois relacionamentos entre as entidades H e C, estes serão nomeados como $HgoC\_1$ e $HgoC\_2$;

- eventuais *joined tables* decorrentes dessas relações serão nomeadas com a junção do nome de duas entidades envolvidas no relacionamento. Os relacionamentos $DgoA$ e $DgoEgoG$ darão origem as *joined tables* DA e DE.

Na definição do nome dos atributos de cada entidade, foi adotada a seguinte convenção:

- o nome dos atributos de uma entidade inicia pela letra que dá nome a ela em minúsculo seguida do caracter _. Assim, todos os atributos da entidade A começam com $a\_$;

- os atributos que são a chave principal da entidade tem nome pk. Assim a chave principal da entidade A é a_pk. Todas as entidades possuem este atributo;

- a sigla st, no nome do atributo, identifica que o atributo em questão é um *string*;

- a sigla int é usada para atributos do tipo inteiro;

- a sigla fl é usada para atributos do tipo real, ou seja, um número de ponto flutuante;

- a sigla dt é usada para atributos do tipo data;

- a sigla en é usada para atributos do tipo enumerado, isto é, atributos com valores constantes, por exemplo, o atributo pode ter apenas os valores 1 ou 2.

Os atributos $\_st1$ são, para efeitos deste trabalho, o atributo que, junto com a chave primária, identifica o registro. Quando o atributo não for um atributo com característica de único, ele receberá um número que não é sequencial dentro da Entidade e sim dentro

do modelo. Essa convenção é necessária em função da Modelagem Grafo Simples (método $M03$, que será depois analisado), onde os atributos não identificadores serão modelados como vértices.

Como foi dito, todas as entidades possuem o atributo *_pk*, do tipo inteiro e sequencial; então, na descrição das entidades ele somente será mencionado se possuir alguma característica específica para aquela entidade.

A entidade principal é a entidade D (figura 2.21). Ela tem quatro atributos além da chave principal:

- $st1$ - um campo *string*, com característica de único, que aceita qualquer conteúdo diferente de nulo e único;
- $int1$ - um campo inteiro com valor no intervalo entre 180 e 267;
- $st2$ - um campo *string* que aceita qualquer conteúdo, inclusive nulo, não único;
- $en2$ - um campo que aceita apenas os valores 1, 2 ou 3 e não pode ser nulo.

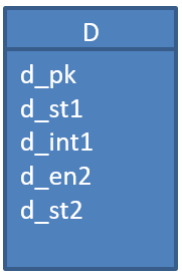


Figura 2.21: Modelo Base - Entidade D

A entidade A possui a chave principal e mais um único atributo, $st1$, do tipo *string*, não nulo.

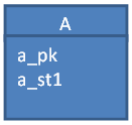


Figura 2.22: Modelo Base - Entidade A

A entidade B (figura 2.23) possui também um único atributo, $st1$, com valor não nulo.

Um registro da entidade D pode se relacionar com outros n registros da mesma entidade D, sendo cada relacionamento obrigatoriamente ligado também a um elemento da

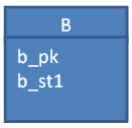


Figura 2.23: Modelo Base - Entidade B

entidade B. Um elemento de D pode ter mais de um relacionamento com outro elemento de D, desde que o elemento de B seja diferente. É o relacionamento $DgoBgoD$.

As entidades E e F (figura 2.24), à semelhança de A e B, têm, também, um único atributo cada, $st1$, com valor não nulo;

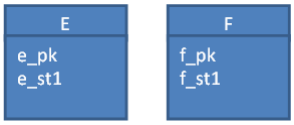


Figura 2.24: Modelo Base - Entidades E e F

A entidade G (figura 2.25) possui três atributos:

- $st1$ - um campo *string* que aceita qualquer conteúdo diferente de nulo;
- $fl1$ - um campo do tipo real, com valores positivos ou negativos, não nulos;
- $fl2$ - um campo do tipo real, com valores positivos ou negativos, não nulos.

Figura 2.25: Modelo Base - Entidade G

Cada elemento de G está ligado a um elemento de F.

A entidade C (figura 2.26) possui três atributos:

- pk - com valores variando entre 1 e 12, 101 e 112 e 201 e 212;
- $st1$ - um campo *string* que aceita qualquer conteúdo diferente de nulo;
- $en1$ - um campo constante que pode ser 1, 2 ou 3.

A entidade H (figura 2.27) possui 2 atributos:

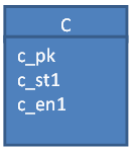


Figura 2.26: Modelo Base - Entidade C

- $dt1$ - um campo data, que não é único na tabela. Os relacionamentos com a tabela C é que individualizam o registro;
- $int2$ - um campo inteiro, dividido em duas partes, $xxyy$, onde xx pode variar de 0 (zero) à 29 e yy de 0 (zero) à 59, totalizando 1800 valores diferentes possíveis.

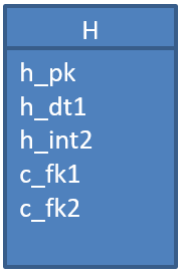


Figura 2.27: Modelo Base - Entidade H

Cada registro de H tem duas ligações com registros de C.

Existem, também, três relacionamentos do tipo $1:n$, com atributos:

- o primeiro é o relacionamento entre as entidades D e A, $DgoA$, onde cada relacionamento tem um atributo ($int4$), que pode ser um número entre 0 (zero) e 99.
- o segundo é o relacionamento terciário entre as entidades D e C, $DgoC$, com as seguintes características:
  - cada registro de D tem 12 relacionamentos com C, onde C deve ser ter o atributo $c\_en1 = 1$;
  - cada um desses relacionamento também deve estar ligado a outros elementos de C, onde C deve ser ter o atributo $c\_en1 = 2$;
  - cada relacionamento tem três atributos:
    - $int3$ - um valor inteiro, que irá variar entre 1 e 12;
    - $int2$ - um valor inteiro com as mesmas características do atributo de mesmo nome da tabela H;

    - $en4$ - um campo constante que pode ser + ou −;

- o terceiro é o relacionamento terciário entre D, E e G, $DgoEgoG$. Este relacionamento tem as seguintes características:
  - cada registro de D tem no mínimo 4 relacionamentos destes;
  - para cada D, E não pode ser repetido;
  - não há restrições quanto à G, que pode ser repetido ou não;
  - cada relacionamento tem três atributos:
    - $dt1$ - um valor do tipo data, não nulo;
    - $st3$ - um valor no formato de hora ($hh : mm$), não nulo;
    - $en3$ - uma constante que pode ter quatro diferentes valores.

Ainda com relação às entidades, pode-se dizer que as entidades B, C e E têm um perfil de conteúdo constante, isto é, depois de implantada a base e recebidos seus registros iniciais dificilmente receberão novos valores. Já as entidades A, D, G e H serão atualizadas constantemente ao longo da vida do sistema e a entidade F poderá ter atualizações periódicas.

Além da convenção utilizada para nomear entidades, relacionamentos e atributos, a figura 2.28 traz a convenção utilizada na representação dos modelos da base e do passo a passo dos métodos de modelagem analisados neste capítulo e no seguinte.

A figura 2.29 traz o modelo entidade relacionamento da base proposta, a partir do qual foi elaborado o modelo relacional da base, como apresentado na figura 2.30.

Neste modelo, além das convenções já detalhadas para a nomenclatura dos atributos, foi seguida a seguinte convenção para os relacionamentos entre as tabelas:

- as chaves estrangeiras, originadas de relacionamentos 1 : 1, foram nomeadas com o nome da tabela apontada em minúsculo, seguida de $\_fk$ mais um número sequencial de acordo com o quantidade de chaves estrangeiras na tabela;
- as *joined tables*, originadas de relacionamentos 1 : n ou m : n, ou relacionamentos 1 : 1 com atributos, foram nomeadas com a junção do nome das tabelas envolvidas. Foram inseridas as siglas $R2$ e $R3$ para indicar se a relação que deu origem a *joined table* é uma relação binária ou terciária.

A seguir serão apresentados os quatro métodos de modelagem analisados nesse trabalho.

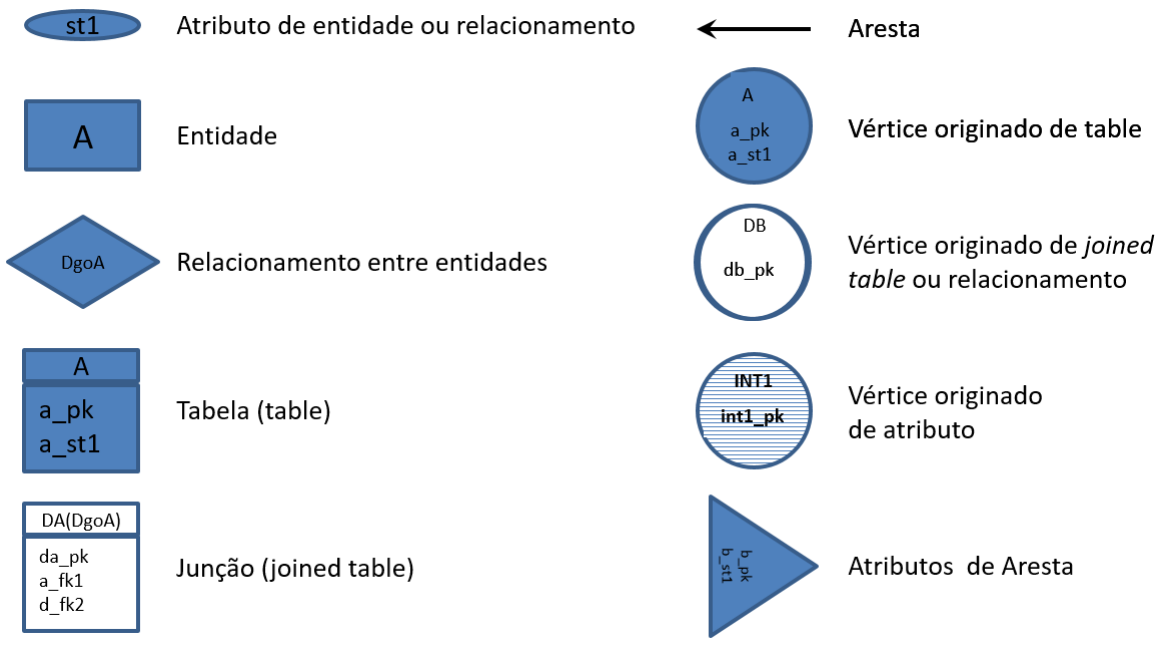


Figura 2.28: Gráficos dos Modelos - Convenções Utilizadas

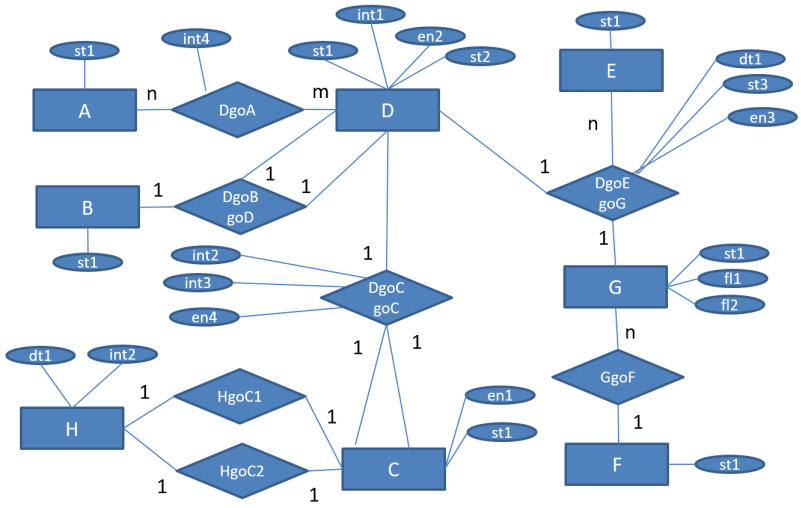


Figura 2.29: Modelo Entidade Relacionamento da BaseTeste

2.5.2 Método $M01$ - Modelagem 3NF - Equivalent Graph

Park et al. (2014) deram forma às diretrizes deste método, uma vez que seu passo a passo segue a solução normalmente proposta pelos desenvolvedores dos SGBDGs, como o Neo4j, Hunger (2015). As duas propostas, em sua essência, seguem os mesmos passos, então, nesta seção serão descritos os passos da proposta de Hunger Por exemplo, Park trata da conversão das tuplas e Hunger das *tables* e *joined tables*, mas em essência são o mesmo passo.

Este método, basicamente, transforma elementos (tuplas) de entidades em elementos

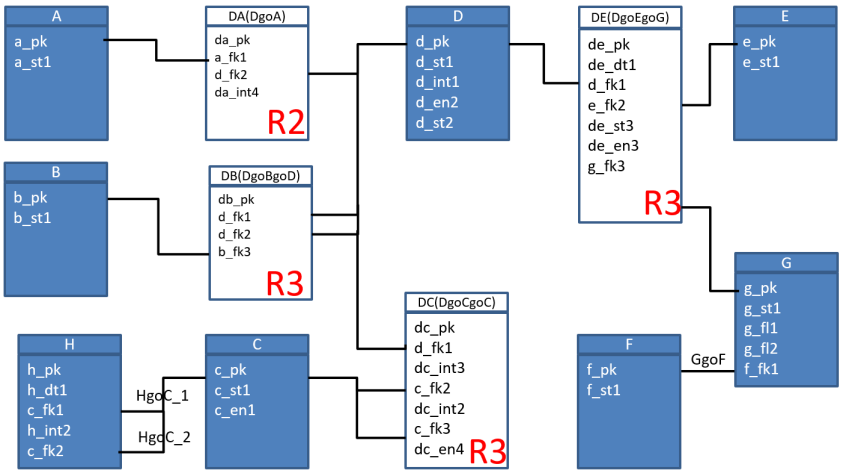


Figura 2.30: Modelo Relacional da BaseTeste

do tipo vértice e elementos do tipo relacionamento em elementos do tipo arestas.

A modelagem parte do modelo relacional. A primeira etapa deste método passa pela identificação do tipo das tabelas, diferenciando-as entre *tables* e *joined tables*:

- *tables* são as tabelas originadas das entidades do modelo. Na figura são as tabelas com fundo escuro: A, B, C, D, E, F, G e H;
- *joined tables* são as tabelas originadas de relacionamentos $1:n$ ou $m:n$ entre as entidades. Na figura são as tabelas com fundo claro mais a sigla $R2$ ou $R3$. No exemplo, são *joined tables* as tabelas DA, DB, DC e DE.

Cada *table* será um vértice do grafo e sua chave primária será o identificador. Não existindo a chave primária, deverá ser incluído um atributo para este fim.

Todos os atributos das tabelas, com exceção dos atributos chave estrangeira (originados de relacionamentos $1:1$), serão atributos do vértice criado.

As chave estrangeiras dão origem a arestas que partem do vértice que as continham com destino ao vértice onde a chave estrangeira é chave primária. Essas arestas serão nomeadas com o nome do respectivo relacionamento no modelo Entidade Relacionamento (figura 2.29).

Nesse passo foram criadas três arestas:

- $GgoF$, originada da chave estrangeira f_fk1 da tabela G, partindo do vértice G para o vértice F;

- $HgoC\_1$, originada da chave estrangeira c_fk1 da tabela H, partindo do vértice H para o vértice C;
- $HgoC\_2$, originada da chave estrangeira c_fk2 da tabela H, partindo do vértice H para o vértice C.

O resultado parcial da modelagem até o final desta etapa pode ser visto na figura 2.31.

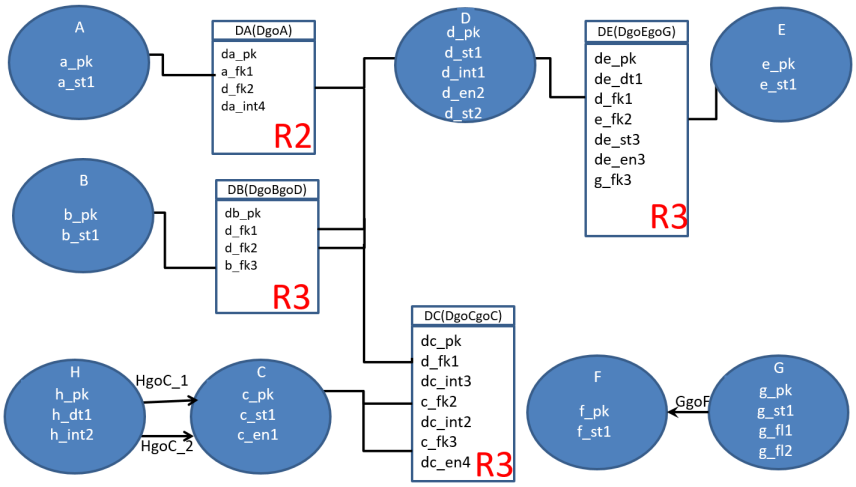


Figura 2.31: Método $M01$ - 3NF - Passo 01

Falta agora fazer a migração das *Joined Tables*.

As *Joined Tables* originadas de relações binárias, no modelo apenas a *table* DA, irão gerar arestas bidirecionais entre os dois vértices originados das tabelas associadas por ela. Essas arestas também serão nomeadas, por convenção, como verbos. Assim, a *joined table* DA dará origem à aresta $DgoA$.

O mesmo critério não pode ser utilizado nas *joined tables* das relações terciárias, pois são três tabelas envolvidas e, por definição, embora uma aresta possa ter atributos, nenhum dos atributos pode apontar para outro elemento do grafo, apenas os ponteiros de origem e destino dela. Assim fica a questão: o que fazer com a terceira tabela envolvida? Como Park não trata desse questão em seu artigo, foi utilizado o mesmo critério adotado por Bordoloi e Kalita (2013a) e tratado a seguir. Assim, é preciso quebrar a *joined table* em um novo vértice e este vértice terá tantas arestas quantas forem as chaves estrangeiras nela embutidas. Dessa forma:

- a *joined table* DB irá gerar um vértice de mesmo nome e três arestas:

  - a aresta $DgoDB$, partindo do vértice D para o novo vértice DB, originada da chave estrangeira d_fk1;
  - a aresta $DBgoD$, partindo do novo vértice DB para o vértice D, originada da chave estrangeira d_fk2;
  - a aresta $DBgoB$, partindo do novo vértice DB para o vértice B, originada da chave estrangeira b_fk3.

- a *joined table* DC irá gerar um vértice de mesmo nome, com os mesmos atributos não chave, e três arestas:
  - a aresta $DgoDC$, partindo do vértice D para o novo vértice DC, originada da chave estrangeira d_fk1;
  - a aresta $DCgoC\_1$, partindo do novo vértice DC para o vértice C, originada da chave estrangeira c_fk2;
  - a aresta $DCgoC\_2$, partindo do novo vértice DC para o vértice C, originada da chave estrangeira c_fk3.

- a *joined table* DE irá gerar um vértice de mesmo nome com os mesmos atributos não chave e três arestas:
  - a aresta $DgoDE$, partindo do vértice D para o novo vértice DE, originada da chave estrangeira d_fk1;
  - a aresta $DEgoE$, partindo do novo vértice DE para o vértice E, originada da chave estrangeira e_fk2;
  - a aresta $DEgoG$, partindo do novo vértice DE para o vértice G, originada da chave estrangeira g_fk3.

Os atributos das entidades que não são nem chave primária, nem chave estrangeira, serão atributos do vértice ou aresta gerados.

O modelo final, já com todas as conversões feitas, pode ser visto na figura 2.32.

A figura 2.33 traz um resumo do passo a passo realizado para a construção deste modelo.

2.5.3 Método $M02$ - Modelagem Reference Graph

Bordoloi e Kalita (2013a, 2013b) trazem uma alternativa à modelagem anterior, mas com resultado final semelhante ao resultado obtido pelo método $M01$. As diferenças entre

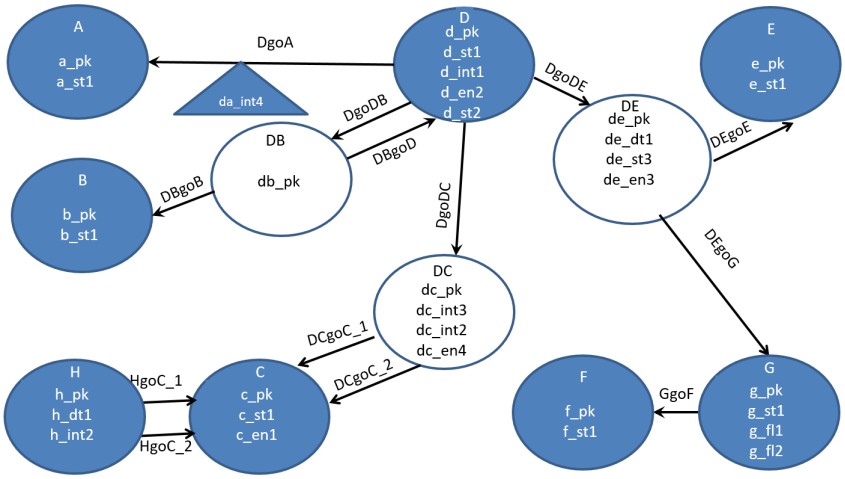


Figura 2.32: Método $M01$ - 3NF - Modelo Final

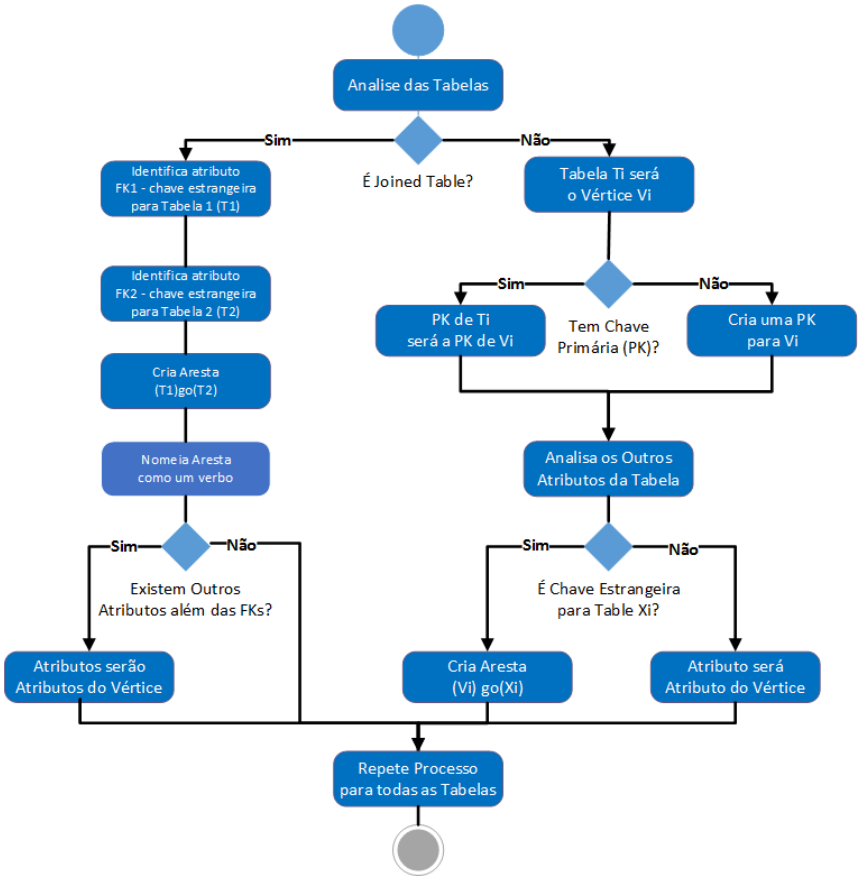


Figura 2.33: Método $M01$ - 3NF - Diagrama de Atividades

os dois métodos estão no fato do primeiro partir do modelo relacional e o segundo partir do modelo entidade relacionamento, além disso as etapas intermediárias de cada um têm caminhos diferentes.

Os autores definem o processo com os seguintes passos:

1. identificar as entidades e seus atributos e determinar um deles para ser a chave primária PK da entidade. Na ausência de um atributo que possa ser a PK, deve-se incluir um atributo para esse fim;

2. mapear os relacionamentos entre as entidades, determinando suas cardinalidades. As cardinalidades podem ser obtidas no modelo E-R, conforme a figura 2.34;

3. as entidades serão os vértices do grafo, semelhante ao método anterior que converte *tables* em vértices;

4. identificar as relações unárias (entidade referencia a própria entidade) e substituí-las por arestas auto-referentes;

5. identificar as relações binárias (entre duas entidades) e:

   - substituir os relacionamentos $1 : n$ por arestas direcionadas da entidade de cardinalidade n para 1;
   - nos relacionamentos $m : n$, com $n > 1$ e $m > 1$, criar um vértice intermediário, e substituir o relacionamento $m : n$ por arestas dirigidas do novo vértice para os vértices que participam do relacionamento;

6. identificar as relações terciárias (entre três entidades) e, a semelhança das relações binárias, gerar um novo vértice com arestas direcionadas para as três entidades envolvidas;

7. caso os relacionamentos binários ou terciários possuam atributos, além dos atributos chave estrangeira, esses deverão fazer parte do vértice criado;

8. criar chaves primárias para os novos vértices. No modelo exemplo, DA(PK *da_pk*), DB(PK *db_pk*), DC(PK *dc_pk*) e DE(PK *de_pk*);

9. nomear as arestas criadas.

Na apresentação do método, antes deste último passo, os autores ainda prevêem uma etapa onde as arestas são nomeadas, provisoriamente, com o nome da *primary key* dos respectivos vértices destino e, depois é feita a montagem de uma matriz de adjacência e do modelo matemático da base, mas que não serão abordados aqui por não serem relevantes para a comparação entre este e os demais métodos.

O resultado final pode ser visto na figura 2.35 e um resumo do passo a passo do método no diagrama da figura 2.36.

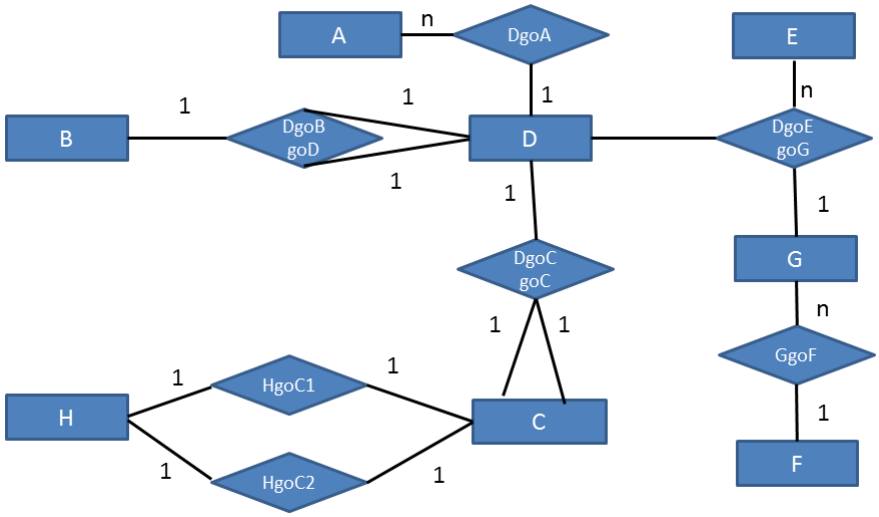


Figura 2.34: Método $M02$ - Reference Graph - Passo 01 - Cardinalidade dos Relacionamentos

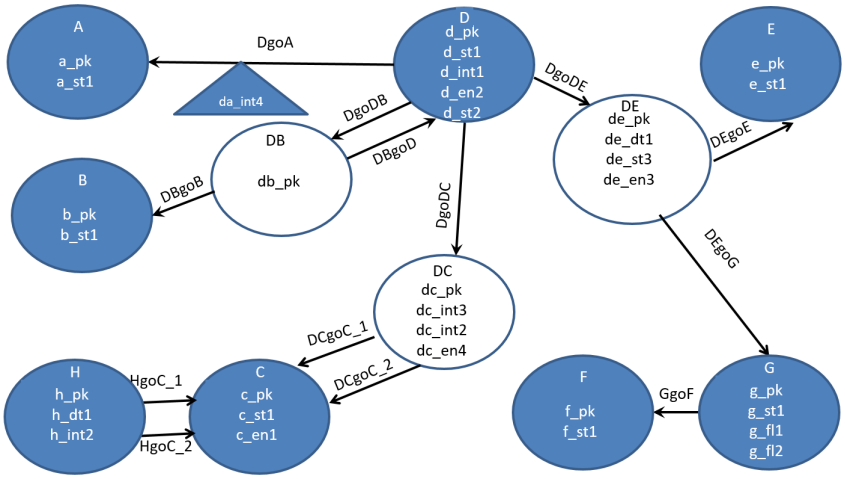


Figura 2.35: Método $M02$ - Reference Graph - Modelo Final

Analisando este método e o anterior, verifica-se que embora os passos do processo sejam teoricamente diferentes, na prática são semelhantes e o resultado final é o mesmo.

2.5.4 Método $M03$ - Modelagem para Grafos Simples (RDF)

Diversos autores, como Erven (2015), Lysenko et al. (2016), Lampoltshammer e Wiegand (2015), em seus trabalhos tem abordado a modelagem dos dados baseada no conceito de grafos simples, isto é, com vértices com uma número reduzido de atributos, aquele considerado chave primária e o atributo que identifica o dado. Mais uma vez, não se aplica aqui o conceito de grafo simples da Teoria dos Grafos, pois nesta modelagem o vértice pode se relacionar com ele mesmo e ter mais de uma aresta ligando-o a outros vértices.

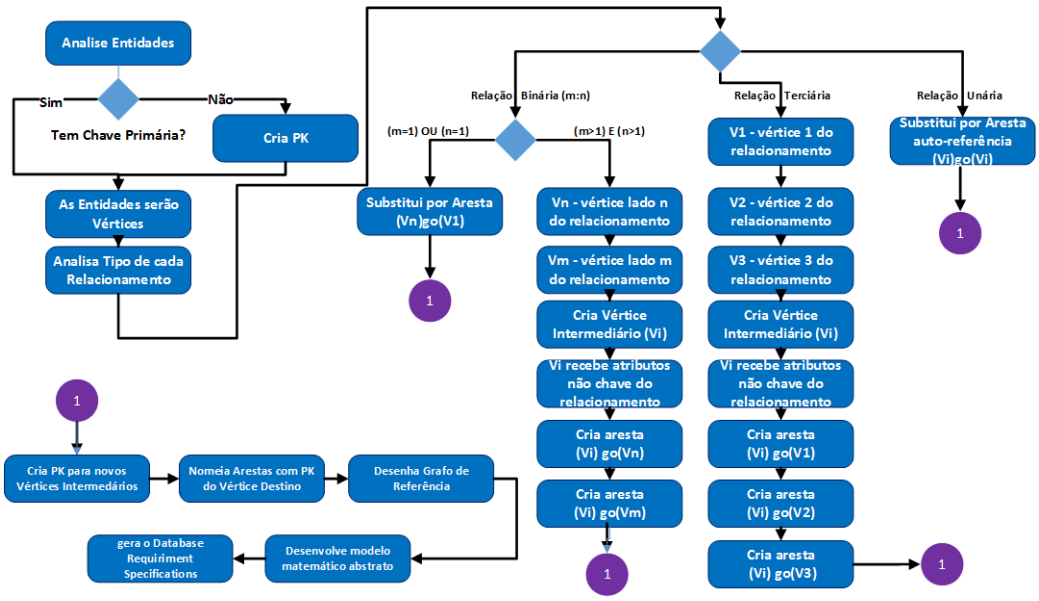


Figura 2.36: Método $M02$ - Reference Graph - Diagrama de Atividades

Seu objetivo parte da premissa de aproximar a representação grafo dos dados da representação da informação pelo método `RDF` (*Resource Description Framework*), que propõe decompor a informação em triplas sujeito, predicado, valor (Sequeda, Arenas e Miranker (2012)), base da Web Semântica[23].

Por exemplo, duas entidades, uma entidade `Pessoa` com os atributos *id*, *nome*, *dataNascimento* e *pesoNascimento*, e outra entidade `Postagem` com os atributos *id*, *dataPostagem* e *comentario* irão gerar:

- um vértice do tipo `Pessoa`, com os atributos *id* e *nome*, pois eles identificam o dado;
- um vértice do tipo `Data`, com o atributo *data*, que será preenchido com os valores *dataNascimento* de cada pessoa e uma aresta que apontará do elemento no vértice `Pessoa` para a data correspondente no vértice `Data`. Duas datas iguais gerarão apenas um registro em `Data`;
- o mesmo critério será adotado ao criar o vértice do tipo `Peso`, com o atributo *valor*, para o atributo *pesoNascimento*;
- um vértice do tipo `Postagem`, com os atributos *id* e *comentario*;
- o atributo *dataPostagem* é do mesmo tipo do atributo *Pessoa.dataNascimento*. Assim, em vez de dar origem a um outro vértice com uma propriedade do tipo *data*,

[23]É a Web de Dados, isto é, padrões, conceitos e técnicas definidos pela W3C para permitir que mecanismos de busquem extraiam informação das páginas WEB.

ele irá atualizar o vértice do tipo `Data` já criado.

Dentro da premissa do grafo simples mais RDF, a partir do vértice `Data`, se necessário é possível chegar tanto a uma `Pessoa` como a uma *Postagem*, cabendo ao mecanismo que pesquisa os dados fazer a distinção.

Numa etapa inicial, este método segue os mesmos passos descritos por Park et al. (2014) e Hunger (2015), com entidades se tornando vértices e relacionamentos se tornando arestas.

O passo seguinte parte do grafo exibido na figura 2.32 e propõe que todos os vértices do modelo atendam a estrutura de grafos simples, ou seja:

- um vértice terá basicamente o seu *id* e um único atributo que o identifica;
- cada um dos demais atributos do vértice gerará outro vértice que será relacionado com a entidade do modelo E-R via arestas. Na implementação do banco deverá ser considerado que estes novos vértices não deverão ter valores duplicados, de forma que na inclusão de um novo elemento, se o valor do atributo já existir, deve-se gerar apenas uma aresta para o valor já existente.

Assim, o passo a passo para modelar o exemplo desta dissertação será:

- os vértices A, B, E, F e DB não sofrem nenhuma alteração pois contém apenas a chave primária e mais um campo que os identificam;
- no vértice C:
  - o atributo c_st1 permanece no vértice por identificá-lo;
  - o atributo c_en1 irá gerar o vértice $EN1$, com 2 atributos, $en3\_pk$ e $en3\_valor$;
- no vértice D:
  - o atributo d_st1 permanece no vértice por identificá-lo;
  - o atributo d_int1 irá gerar o vértice $INT1$, com 2 atributos, $int1\_pk$ e $int1\_valor$;
  - o atributo d_en2 irá gerar o vértice $EN2$, com 2 atributos, $en2\_pk$ e $en2\_valor$;
  - o atributo d_st2 irá gerar o vértice $ST2$, com 2 atributos, $st2\_pk$ e $st2\_valor$;
- no vértice G:

  - o atributo g_st1 permanece no vértice por identificá-lo;
  - uma vez que os atributos g_fl1 e g_fl2 têm o mesmo tipo de dado, eles irão gerar o vértice FL, com 2 atributos, fl_pk e fl_valor;

- no vértice H:
  - o atributo h_dt1 não permanece no vértice pois ele não é único, então ele gera o vértice dt, com 2 atributos, dt_pk e dt_valor;
  - o atributo h_int2 irá gerar o vértice $INT2$, com 2 atributos, $int2\_pk$ e $int2\_valor$;

- no vértice DC:
  - o atributo dc_int3 irá gerar o vértice $INT3$, com 2 atributos, $int3\_pk$ e $int3\_valor$;
  - uma vez que o atributo dc_int2 tem as mesmas características e restrições do atributo h_int2, seus valores também estarão ligados ao vértice $INT2$;
  - o atributo dc_en4 irá gerar o vértice $EN4$, com 2 atributos, $en4\_pk$ e $en4\_valor$;

- no vértice DE:
  - o atributo de_st3 irá gerar o vértice $ST3$, com 2 atributos, $st3\_pk$ e $st3\_valor$;
  - o atributo de_en3 irá gerar o vértice $EN3$, com 2 atributos, $en3\_pk$ e $en3\_valor$;
  - uma vez que o atributo de_dt1 tem as mesmas características e restrições do atributo h_dt1, seus valores também estarão ligados ao vértice DT.

O modelo final resultante da aplicação deste método é o apresentado na figura 2.37, onde os vértices com fundo escuro são originados de entidades, os vértices de fundo claro vêm de relacionamentos (*joined tables*) e os vértices hachurados são os decorrentes deste último passo.

O passo a passo completo deste método é apresentado na figura 2.38.

Comparando o resultado desse método com os dois anteriores, verifica-se que, a sua primeira fase segue os mesmos passos que o método $M01$ na modelagem das entidades e seus relacionamentos. Na fase 2 é que esse modelo é expandido, com a criação de vértices para todos os atributos sem característica de valor único na base e as respectivas arestas ligando vértice original e novo vértice, resultando num grafo muito mais fragmentado.

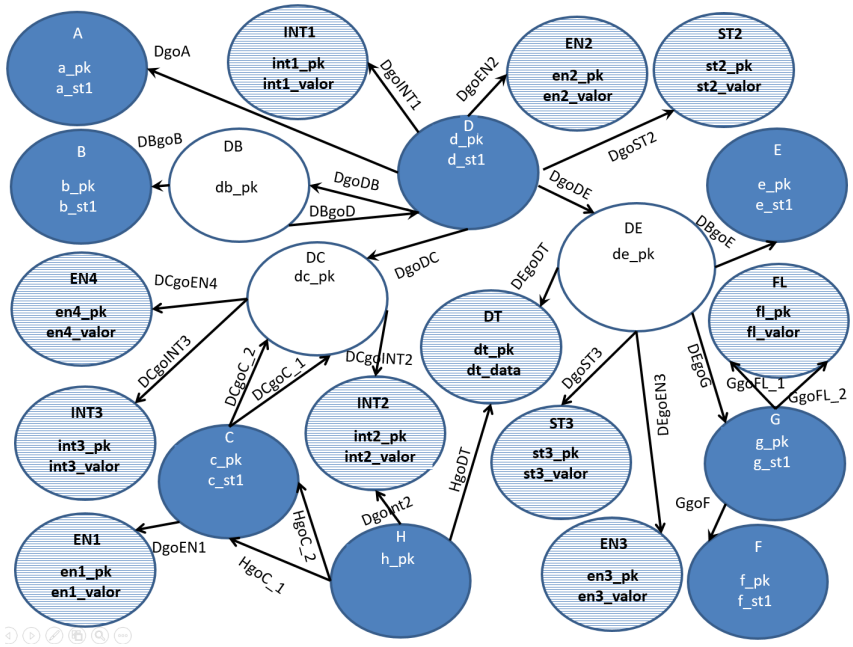


Figura 2.37: Método $M03$ - Grafo Simples - Modelo Final

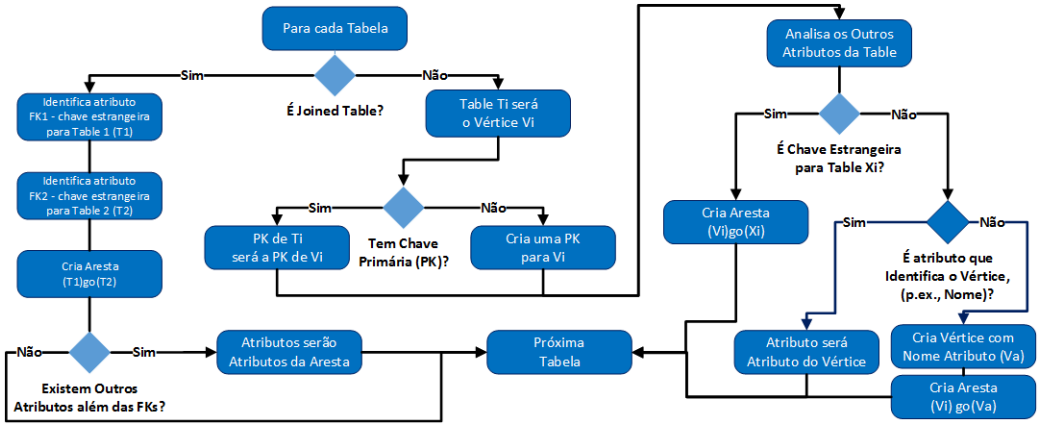


Figura 2.38: Método $M03$ - Grafo Simples - Diagrama de Atividades

2.5.5 Método $M04$ - Modelagem Dirigida

Proposto por Virgilio, Maccioni e Torlone (2014), este método parte do modelo entidade relacionamento e propõe funções de análise a partir dos relacionamentos entre as entidades, que poderão indicar a junção de duas ou mais delas numa só, mesmo que isso implique em redundância de dados, quebrando as regras de normalização de Codd (1970). Para tal, os autores propõem a atribuição de pesos para as arestas, a fim de determinar quais entidades podem ser unificadas numa única a fim de ter um melhor desempenho nas pesquisas à base.

No artigo, os autores não tratam da modelagem de relações de grau > 2, antes de começar a aplicar o método $M04$ as relações ternárias da *BaseTeste* foram convertidas em

relações binárias, com a criação de uma entidade intermediária e de relacionamentos de cardinalidade 1 : 1 entre esta nova entidade e as entidades originais do relacionamento, a semelhança da modelagem para o Modelo Relacional. A tabela 2.1 traz os relacionamentos convertidos e a figura 2.39 traz um modelo entidade relacionamento simplificado da base após essa conversão, com a cardinalidade dos relacionamentos em destaque, daqui prá frente, chamado, nesta seção, de $MER2$.

Relacionamento	Entidade Criada	Relacionamentos Criados
DgoBgoD	DB	DgoDB, DBgoD, DBgoB
DgoCgoC	DC	DgoDC, DCgoC1, DCgoC2
DgoEgoG	DE	DgoDE, DEgoE, DEgoG

Tabela 2.1: Método $M04$ - Modelagem Dirigida - Convertendo os Relacionamentos Ternários em Binários

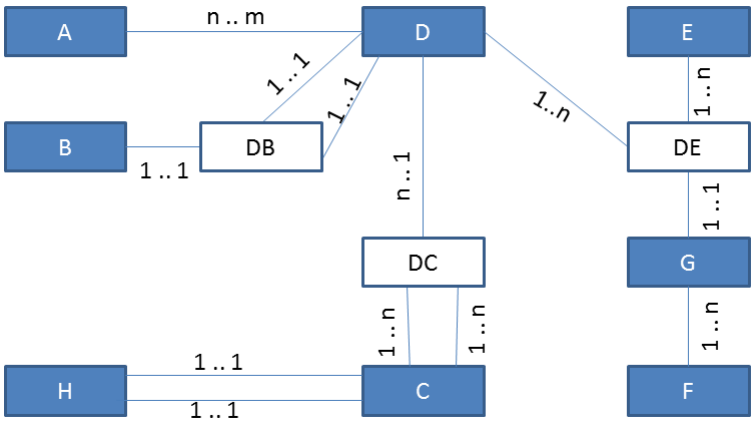


Figura 2.39: Método $M04$ - Modelagem Dirigida - Modelo E-R com cardinalidade em evidência

Como já dito, o ponto de partida é o modelo entidade relacionamento da base a ser modelada. Nessa análise, de acordo com a sua cardinalidade, os relacionamentos são substituídos por arestas com pesos e direção, onde:

- partindo de zero, o peso da aresta é calculado somando-se 1 para cada lado do relacionamento com cardinalidade maior que 1;
- relacionamentos $1 : n$ geram arestas direcionadas do lado 1 para o lado n, os demais geram arestas não dirigidas, isto é, pode-se caminhar em qualquer sentido do relacionamento.

Assim:

- os relacionamentos 1 para 1 são substituídos por arestas não dirigidas com $peso = 0$;

- os relacionamentos $1 : n$, com $n > 1$, geram arestas dirigidas, da entidade de cardinalidade 1 para a de cardinalidade n, com $peso = 1$;
- por fim, os relacionamentos n para m, com $n > 1$ e $m > 1$, originam arestas não dirigidas, com $peso = 2$.

O resultado da aplicação desse passo para o $MER2$ pode ser visto na figura 2.40, onde cada aresta foi nomeada com o nome da entidade origem, seguido da expressão *go* mais o nome da entidade destino para facilitar sua identificação nos cálculos a seguir. Embora não dirigidas, as arestas de peso 2 foram representadas com setas nas duas pontas a fim de facilitar a visualização das arestas que entram e saem de uma entidade.

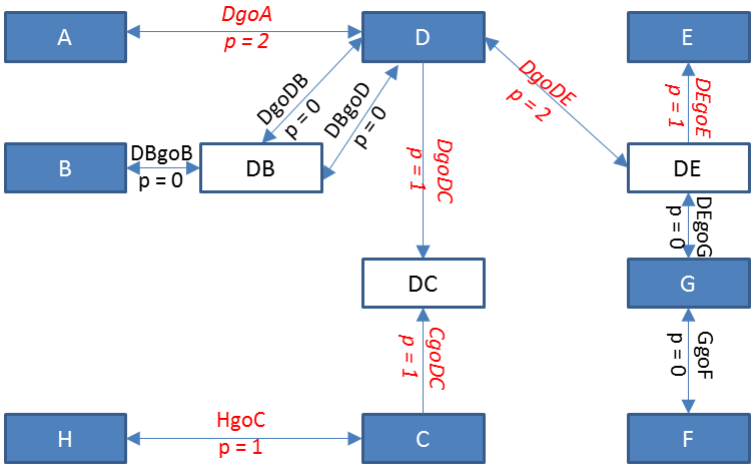


Figura 2.40: Método $M04$ - Modelagem Dirigida - Modelo Entidade Relacionamento Orientado após Fase 1

O passo 2, particionamento do modelo, começa com o cálculo dos pesos de saída w^+ e de entrada w^- de cada entidade, onde:

- $w^+(n)$ é a soma do peso n de todas as arestas que saem da entidade;
- $w^-(n)$ é a soma do peso n de todas as arestas que apontam para ela.

Para efeito do cálculo, duas arestas com mesma origem e mesmo destino foram consideradas uma só.

Este cálculo e as arestas envolvidas é apresentado na tabela 2.2. As arestas com peso igual a zero, por não serem relevantes ao cálculo, não figuram na tabela.

O resultado desse passo pode ser visto na figura 2.41.

Entidade	$w^+(n)$	$w^-(n)$
A	DgoA(2) = 2	DgoA(2) = 2
B	DBgoB(1) = 1	0
C	CgoDC(1) = 1	
D	DgoA(2) + DgoDB(1) + DBgoD(1) + DgoDE(1) + DgoDC(1) = 6	DgoA(2) = 2
E	DgoDE(1) = 1	0
F	0	GgoF(1) = 1
G	GgoDE(1) + GgoF(1) = 2	0
H	0	0
DB	0	DgoDB(1) + DBgoD(1) + DBgoE
DC	0	DgoDc(1) + CgoDC(1) = 2
DE	0	DgoDE(1) + DEgoE(1) + DEgoG

Tabela 2.2: Método $M04$ - Modelagem Dirigida - Fase 2 - Aplicação das Funções $w^+(n)$ e $w^-(p)$

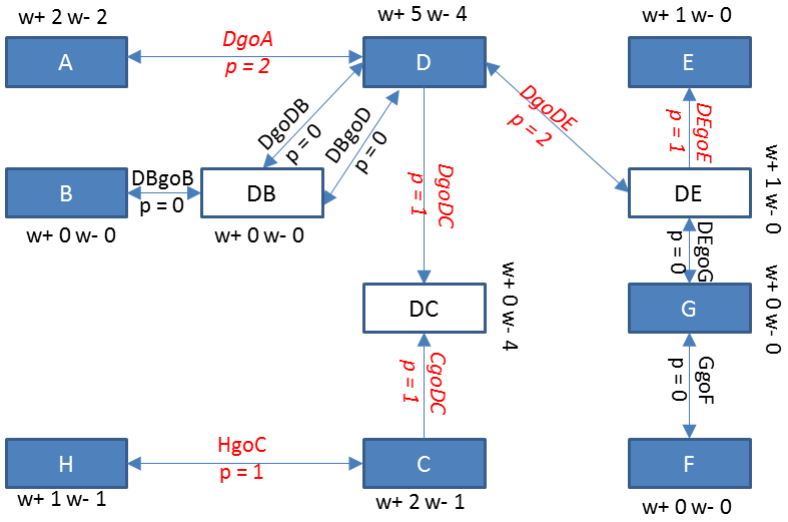


Figura 2.41: Método $M04$ - Modelagem Dirigida - Fase 2 - Resultado do cálculo de $w^+(n)$ e $w^-(n)$

Ainda no passo 2, é realizado o particionamento do modelo em grupos, conforme as regras a seguir:

- Regra 1 - cada entidade não conectada com nenhuma outra entidade será um grupo separado;
- Regra 2 - Se $(w^-(n) > 1) \wedge (w^+(n) \geq 1)$ então a entidade será um grupo isolado;
- Regra 3 - Se $(w^-(n) \leq 1) \wedge (w^+(n) \leq 1)$ então a entidade deve ser adicionada ao grupo de uma entidade com o qual ela possua um relacionamento $m : n$.

Nesta fase, a regra 1 não se aplica ao modelo em questão, pois todas as entidades estão relacionadas com pelo menos uma outra entidade. A tabela 2.3 traz um resumo da

aplicação das funções w^- e w^+ às entidades do modelo e a regra resultante a ser aplicada no particionamento.

Vértice	$w^+(n)$	$w^-(n)$	Regra	Ação
A	2	2	2	grupo A
B	1	0	3	grupo B+DB
C	1	0	3	grupos H+C e DC+C
D	6	2	2	grupo D
E	1	0	3	grupo E+DE
F	0	1	3	grupo F+G
G	2	0	3	grupo F+G
H	0	0	2	grupo H+C
DB	0	3	3	grupo B+DB
DC	0	2	2	grupo DC+C
DE	1	3	3	grupo E+DE

Tabela 2.3: Método $M04$ - Modelagem Dirigida - Determinando a Regra a Ser Aplicada

A figura 2.42 apresenta o particionamento da base após a execução da fase 2, quando as entidades foram reunidas em sete grupos.

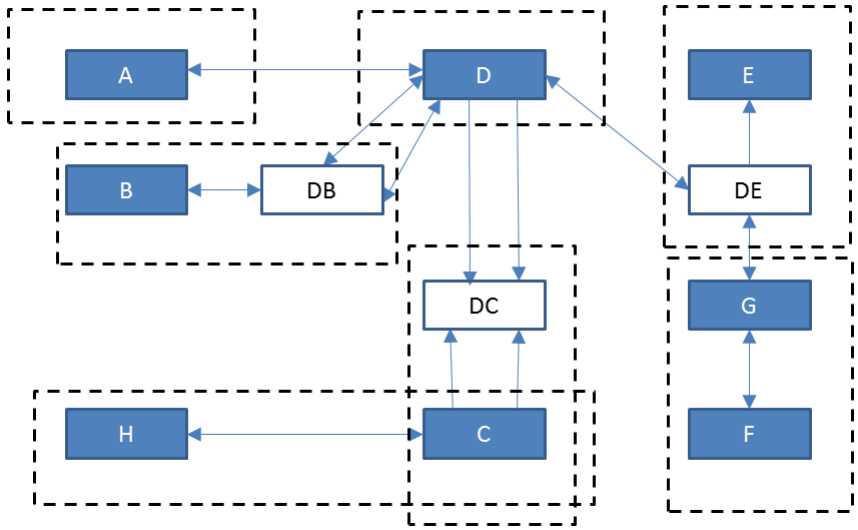


Figura 2.42: Método $M04$ - Modelagem Dirigida - Fase 2 - Agrupamento

Entretanto, ao observar o modelo resultante, figura 2.44, esse ainda pode ser otimizado, aplicando a fase 2 ao modelo resultante. As figuras 2.44 e 2.45 mostram esse processo, no qual o grupo $G(+F)$ é incorporado ao grupo $(DE(+E))$.

Isso feito, cada grupo será um vértice do grafo, unindo os atributos de cada entidade nele reunidas.

As figuras 2.46 e 2.47 trazem o modelo final do banco de dados e um resumo das etapas realizadas pelo método.

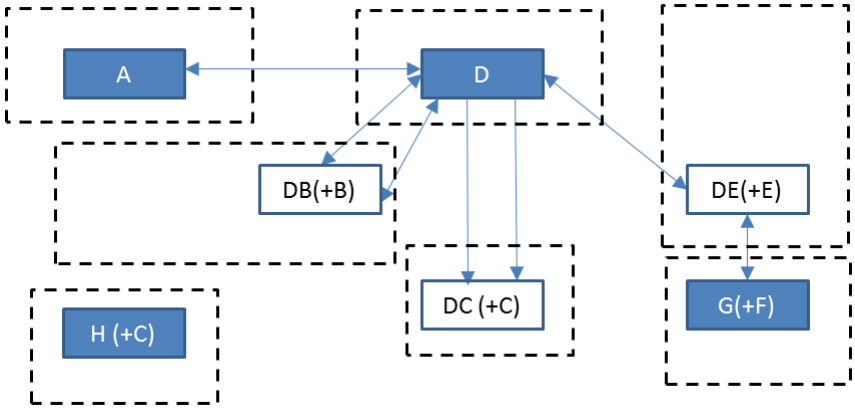


Figura 2.43: Método $M04$ - Modelagem Dirigida - Fase 2 - Grupos

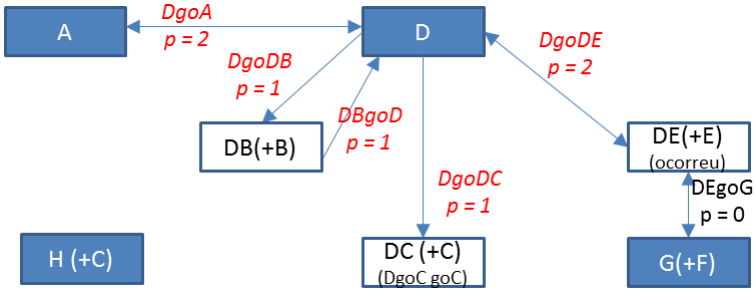


Figura 2.44: Método $M04$ - Modelagem Dirigida - Fase 2B - Peso das Arestas

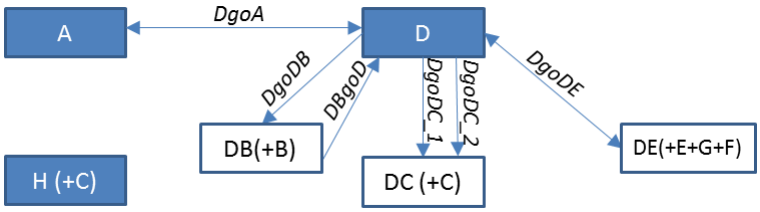


Figura 2.45: Método $M04$ - Modelagem Dirigida - Fase 2B - Novos Grupos

Comparando o modelo final deste método com o modelo final dos três métodos anteriores observa-se que este é o menor em número de vértices, 6 (seis), e no número de arestas, 5 (cinco). Tal resultado advém da junção de várias entidades em uma só, gerando, assim, duplicação de dados. O objetivo aqui é que ao acessar um registro todos os seus dados estejam disponíveis, sem necessidade de acessos a outros vértices. Essa característica vem da quebra das regras de Codd (1970), onde, naquela época, havia uma grande preocupação com o tamanho do banco de dados por uma limitação de *hardware* e hoje já não é tão crítico, podendo haver uma repetição de dados na tentativa de um melhor desempenho no acesso ao banco.

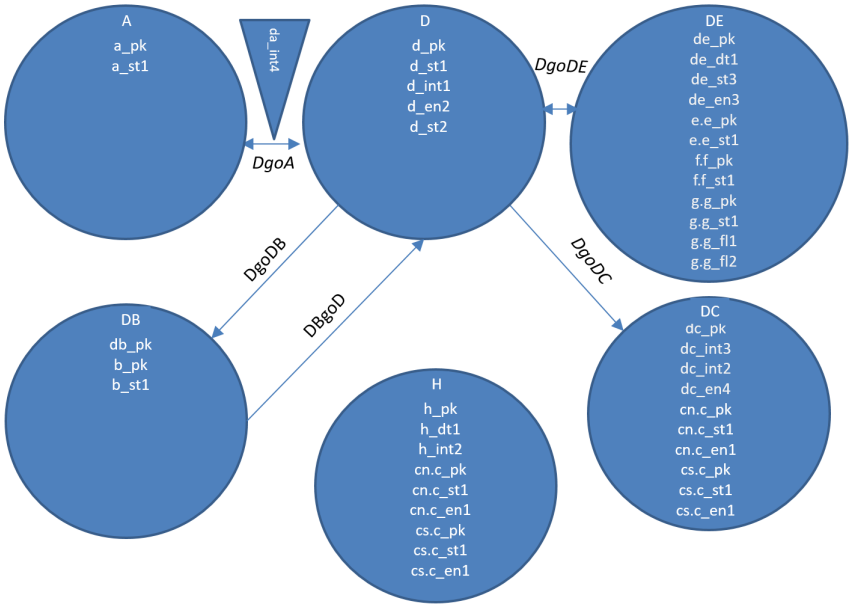


Figura 2.46: Método $M04$ - Modelagem Dirigida - Modelo Final

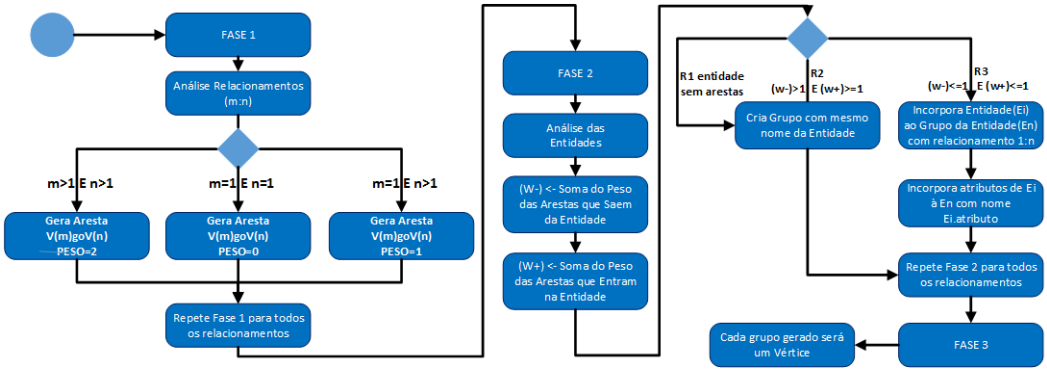


Figura 2.47: Método $M04$ - Modelagem Dirigida - Diagrama de Atividades

2.5.6 Compararativo entre os Métodos

A tabela 2.4 traz um comparativo entre os quatro métodos analisados, quanto ao número de vértices e arestas.

Método	Vértices	Arestas
$M01$ - 3NF - Equivalent Graph	11	13
$M02$ - Reference Graph	11	13
$M03$ - Grafos Simples (RDF)	22	27
$M04$ - Modelagem Dirigida	6	5

Tabela 2.4: Revisão - Comparativo entre os Métodos

Observando o resultado final das modelagens conforme os quatro métodos analisados pode-se dizer que um ponto comum é que *tables* irão sempre gerar vértices, embora estes vértices possam ser incorporados a outros vértices;

As modelagens conforme os métodos $M01$, $M03$ e $M04$ iniciam o processo pelo modelo relacional da base, ao contrário do método $M02$, que inicia pelo modelo entidade relacionamento.

Isso pode levar à conclusão, num primeiro momento, que essas técnicas de modelagem servem para fazer a migração de uma base de dados no modelo relacional para uma base de dados de grafos. De fato, isso ocorre. Mas não apenas para isso.

Conforme Chen (1976) e Teorey, Yang e Fry (1986), o modelo relacional é elaborado a partir do modelo entidade relacionamento da base. Com isso, pode-se afirmar que todos os métodos vêm do modelo entidade relacionamento.

Na definição de Chen (1976), o modelo entidade relacionamento é um modelo conceitual que descreve os objetos, as entidades, envolvidos em um domínio de negócios, com suas propriedades (atributos) e como esses objetos se relacionam entre si.

Costuma-se dizer que o modelo relacional é o próximo passo após elaborado o modelo entidade relacionamento. Isso é parcialmente verdade, já que, ainda conforme Chen (1976), a partir deste modelo podiam ser elaborados também modelos para outros tipos de bancos de dados existentes na época, como o modelo em rede, e, atualmente vem servindo de base para modelagens em bancos orientados a objetos Markowitz e Makowsky (1990), Karp (1995).

Isso posto, então esse modelo entidade relacionamento pode ser usado também, para, a partir dele, modelar bancos de dados de grafos.

O capítulo 4 traz uma série de experimentos que foram realizados com a implantação de bases de dados conforme cada uma das modelagens. O que se pode antecipar das conclusões destes experimentos é que nenhuma das bases teve um desempenho superior às demais, conforme as características das consultas efetuadas, ora uma, ora outra base teve melhor desempenho.

Em cima dessas conclusões, foi elaborada uma nova proposta de modelagem de dados, partindo do modelo entidade relacionamento, como o método $M02$, onde na proposta busca-se contemplar os pontos fortes detectados em cada um dos experimentos com os quatro métodos.

O próximo capítulo traz a proposta deste novo método.

3 Modelagem Participativa

Neste capítulo é apresentada a `Modelagem Participativa`, uma nova abordagem para modelar bancos de dados voltados a grafos.

Este método foi elaborado a partir da análise dos pontos fortes e fragilidades dos métodos analisados no capítulo anterior, os quais são uma sequência de passos que simplesmente convertem entidades ou tabelas e relacionamentos em vértices e arestas seguindo regras matemáticas, sem uma preocupação com o conteúdo dos elementos convertidos. O método proposto, pelo contrário, pede que o projetista do banco tenha um amplo conhecimento das regras do negócio que será atendido pelo `SGBDG`, daí receber o nome de Modelagem Participativa.

Todos os autores dos métodos analisados no capítulo anterior e outros autores, como Wardani e Kiing (2014), Batra e Tyagi (2012), Jaiswal e Agrawal (2013) e Fong, Wong e Cheng (2003), partem sempre do modelo entidade relacionamento ou do modelo relacional para modelar bancos de dados voltados a grafos. Siriwaradhana (2014) coloca que entre os modelos concentuais utilizados para descrever os dados do domínio de um sistema estão a UML[1] e o modelo entidade relacionamento, mas que considera o segundo modelo mais apropriado para descrever dados e o primeiro para descrever especificações de código. Nenhum dos autores listados traz um modelo conceitual do domínio de um sistema já voltado para grafos. Uma das explicações Uma das possíveis explicações, mas certamente não a única, é que uma vez que durante décadas os projetistas de bancos de dados acostumaram-se a moldar as regras de negócio e os requisitos de sistemas utilizando o modelo entidade relacionamento e, a partir deste, o modelo relacional, que tem uma característica de grafos, usar estes modelos como ponto de partida foi natural. Escolheu-se partir de um ponto conhecido para chegar à nova modelagem. Esta situação tem levado muitos desenvolvedores a acreditarem que bancos de dados voltados a grafos (`BDGs`) são propícios a serem utilizados como apoio a bancos relacionais, onde os primeiros seriam

[1] U*nified Modeling Language* - linguagem para visualização, especificação, construção e documentação de artefatos de um software em desenvolvimento.

gerados para uma ferramenta específica e depois do uso removidos, sendo persistente apenas o banco relacional. Essa é uma forma de uso, mas não a única. Novos sistemas estão sendo desenvolvidos especificamente para uso com `BDGs` e, em outros, a base relacional está sendo inteiramente migrada para grafos.

3.1 Método Proposto - Modelagem Participativa

3.1.1 Objetivo

Com o incremento no uso dos bancos de dados voltados a grafos, tem aumentado a necessidade de se definir como modelar o banco de forma a obter um melhor desempenho sem perder o foco na garantia de manter a integridade dos dados e possibilidade de expansão do mesmo.

Antes de prosseguir, é importante definir o que é modelar os dados. Para este trabalho, modelar os dados é, partindo do modelo conceitual do sistema, definir o modelo lógico dos seus dados utilizando o conceito de grafos, ou seja, organizando-os em vértices e arestas.

Nesse sentido, esta dissertação apresenta um novo método de modelagem para bancos de dados voltados a grafos a partir do modelo entidade relacionamento (MER) e que incorpora as melhores características dos métodos avaliados no capítulo anterior.

Uma vez que os métodos anteriores foram numerados de $M01$ à $M04$, o método proposto será nominado nesta dissertação como $M05$.

3.1.2 O Método

Conforme visto na seção 2.3, o modelo entidade relacionamento pode ser representado por um grafo de referência, que neste capítulo será denotado por $G = E, R$, onde:

- E - é o conjunto das entidades originais do modelo entidade relacionamento mais os relacionamentos que na modelagem relacional geram uma nova entidade. Esses elementos são:
  - as entidades originais do modelo, que neste capítulo serão representadas por (E_E);
  - os relacionamentos $m : n$, com $m > 1$ e $n > 1$, doravante representados por E_{Rmn};

– e os relacionamentos com grau maior que dois, aqui representados por E_{R3};

- R - são os relacionamentos que no modelo relacional geram chaves estrangeiras, ou seja, os unários, binários do tipo 1 : 1 e binários do tipo 1 : n, respectivamente representados neste capítulo por R_1, R_{R11} e R_{R1n}.

O $M05$, partindo do modelo entidade relacionamento (MER) da base, tem três fases, onde na primeira o MER é analisado elemento a elemento e um novo MER simplificado é gerado (MER2), o qual, na segunda fase, será moldado como grafo e na última fase, os vértices serão analisados e eventualmente divididos em novos vértices.

Na fase 1 do método, modelagem do MER2, é realizada uma análise das entidades e relacionamentos do modelo. Essas entidades poderão ser, durante o processo, absorvidas por outras entidades ou mesmo relacionamentos. Estes serão convertidos, também, até que existam apenas relações binárias ou unárias no MER2.

Esta fase está dividida em três passos, descritos a seguir.

3.1.2.1 Método $M05$ - Fase 1 - Criando MER2 - Passo 1 - Classificando as Entidades

Este primeiro passo calcula, para cada entidade original (E_E) do `MER`, o que se denomina seu grau de incorporação ($G_I(E_E)$). O grau de incorporação tem o propósito de indicar se uma entidade pode ser absorvida por outra entidade ou relacionamento do modelo.

Para poder ser incorporada a outro elemento, uma entidade deve:

- participar de um e apenas um relacionamento, não importando se o relacionamento envolve duas ou mais entidades. Deve ser um único relacionamento, porque incorporá-la a duas ou mais entidades dificulta a manutenção quando houver necessidade de alterar algum de seus dados, se incorporados em diferentes entidades. E, ainda, este único relacionamento não deve ser unário, isto é, não pode ser um auto-relacionamento, porque não tem sentido incorporar uma entidade a ela mesma;
- ter um perfil de conteúdo constante, isto é, existir uma pequena possibilidade de seus itens sofrerem manutenção, como, por exemplo, entidades geradas a partir de classes do tipo *enum*[2], já que a alteração de um dado da entidade será sempre uma operação onerosa por exigir alterar todos os registros com o novo valor do dado e

[2]Enum ou enumerates em Java são estruturas de dados que armazenam uma coleção de valores fixos predefinidos e imutáveis

não apenas um registro e a inclusão de novos dados poder implicar na revisão das consultas de acesso à base.

Ao final deste passo todas as entidades do modelo estarão classificadas com grau de incorporação igual a um (1) indicando que pode ser incorporada ou zero (0), não incorporável. Este processo está detalhado no algoritmo 3.1, que é aplicado para todas as entidades do modelo. O algoritmo realiza 3 testes:

- se a entidade E_E tem perfil de expansão 'pouco provável´, então $G_E = 1$ senão $G_E = 0$ (linhas 3 à 7);
- se a entidade E_E participa de apenas 1 relacionamento, então $G_R = 1$, senão $G_R = 0$ (linha 8 à 12);
- se a entidade atendeu às duas exigências ($G_E + G_R = 2$), então tem grau de incorporação 1, senão 0 (linhas 13 à 17).

O grau de incorporação das entidades será utilizado no cálculo do Grau de Incorporação dos Relacionamentos, que será a soma do grau de incorporação das entidades por ele conectadas.

Algoritmo 3.1: Cálculo do Grau de Incorporação das Entidades($G_I(E_E)$)

```
   Input: Entidades Originais (E_E) do MER
   Output: Graus de Incorporação das Entidades (G_I(E_E)) Calculados
 1 Function CalculaG_I(E_E) is
 2     for ∀E_E do
 3         if Expansão no Número de Registros de E_E = 'pouco provável´ then
 4             G_E ← 1
 5         else
 6             G_E ← 0
 7         end
 8         if Número de Relacionamentos em que E_E participa = 1 then
 9             G_R ← 1
10         else
11             G_R ← 0
12         end
13         if (G_E + G_R) = 2 then
14             G_I(E_E) ← 1
15         else
16             G_I(E_E) ← 0
17         end
18     end
19 end
```

3.1.2.2 Método $M05$ - Fase 1 - Criando MER2 - Passo 2 - Convertendo Relacionamentos de $Grau > 2$ para Entidades e Relacionamentos

Os experimentos realizados com as bases geradas a partir dos métodos apresentados no capítulo anterior mostraram que uma atenção especial deve ser dada à modelagem de relacionamentos de grau > 2. Por exemplo, ao analisar a modelagem do relacionamento de grau 3, $DgoEgoG$. Na modelagem pelos métodos $M01$ e $M03$ foi criada uma nova entidade, DE, e três arestas $DgoDE$, $DEgoE$ e $DEgoG$. Na modelagem $M04$, as entidades E e G foram incorporadas à entidade DE e restou uma única aresta, $DgoDE$. Agora, imagine-se que se quer acessar para um elemento de D todos os elementos de G que estão ligados a ele, mas restringindo-se a pesquisa aos relacionamentos cuja entidade E tenha $e\_pk = 1$. No primeiro caso, é necessário acessar, partindo de D acessar DE, a partir dele chegar a E e, caso o elemento de E tenha $e_pk = 1$, então, a partir de DE chegar a G. Já no segundo caso, com um único acesso chega-se aos elementos desejados. Isso, naturalmente, diminui o tempo de travessia do grafo.

Como foi visto, incorporar uma entidade a uma aresta ou vértice pode melhorar o desempenho dos acessos à base `BDG`, mas nem todo relacionamento pode ter entidades a ele conectadas convertidas em propriedades de uma aresta. Determinar quais relacionamentos podem ou não ter entidades convertidas em propriedades do relacionamento ou em propriedade de outra entidade é o objetivo deste passo.

Encerrado o passo 1, este passo começa pelo cálculo do grau de incorporação de cada relacionamento (G_{IR}) que é a soma do grau de incorporação ($G_I(E_E)$) das entidades conectadas por ele (algoritmo 3.2). Assim, quando um relacionamento tem seu grau de incorporação maior que zero (> 0) isso indica que pelo menos uma das entidades a ele conectadas pode ser incorporada por ele ou por outra entidade também conectada ao relacionamento. O algoritmo 3.2 é executado para cada relacionamento que se quer analisar e nele é calculado o grau de incorporação das entidades conectadas pelo relacionamento, G_I (algoritmo 3.1), e no final retorna a soma destes graus.

Então, quando uma entidade com grau de incorporação igual à 1 participar de um relacionamento de grau n, com $n > 2$, seus dados, em vez de serem absorvidos por uma das outras entidades do relacionamento, serão incorporados ao relacionamento, que, a partir de então, com uma entidade a menos, passará a ser de grau $n - 1$. De maneira recursiva, o novo grau de incorporação do relacionamento é calculado e, enquanto ele for de grau > 2 e grau de incorporação > 0, uma nova entidade será por ele absorvida.

Caso duas entidades participantes do relacionamento tenham grau de incorporação

Algoritmo 3.2: Cálculo do Grau de Incorporação de um Relacionamento de grau $n(G_{IR}(E_E))$

Input: Relacionamento de Grau n (R_n)
Output: Grau de Incorporação ($GI_{IR}(R_n)$) calculado

```
1 Function CalculaG_IR(R_n) is
2 |   g_ir ← 0
3 |   for ∀E_E|E_E é conectada à R_n do
4 |   |   g_ir ← g_ir+ CalculaG_I(E_E); /* algoritmo 3.1 */
5 |   end
6 |   return g_ir
7 end
```

igual à 1, uma delas deve ser eleita para ser incorporada ao relacionamento. Os critérios a serem adotados para essa seleção não são fixos, pois variam de caso a caso e dependem da análise e escolha do projetista da base. Por exemplo:

- se escolher incorporar a entidade da qual se espera que tenha menor número de registros, a tendência é que existam mais registros com o mesmo dado repetido. Nesse sentido, uma operação de eventual alteração do dado será mais onerosa, entretanto uma seleção de dados irá filtrar mais registros numa única operação que o contrário;
- se escolher incorporar a entidade que tenha menor número de atributos, menor será o número de atributos com o mesmo valor na base e, portanto, a área ocupada pelo banco será menor.

No caso dos relacionamentos de grau igual a 2, dois critérios devem ser analisados. O primeiro é verificar se o relacionamento tem atributos, além das duas entidades por ele conectadas. Se o relacionamento possuir atributos, ele permanece como está, isto é, nenhuma das entidades pode ser incorporada pela outra.

O próximo critério a ser analisado é a cardinalidade do relacionamento. Se o relacionamento for do tipo $m : n, m > 1 \wedge n > 1$, a incorporação não é possível por ser impraticável incluir nos m elementos da entidade do lado m, as n ocorrências da entidade n, ou vice-versa. Se a cardinalidade for $1 : n$, a incorporação somente será possível se a entidade do lado n tiver grau de incorporação 1, ou seja, os atributos da entidade do lado 1 serão incluídos como atributos da entidade do lado n.

O processo de escolha da entidade incorporável (E_I) pode ser visto no algoritmo 3.3, que:

- recebe o Relacionamento a ser analisado e devolve a entidade que deve ser incor-

porada a ele ou a outra entidade. Se retornar *null* indica que não há entidade incorporável;

- calcula o Grau de Incorporação do Relacionamento (G_IR), chamando a função descrita no algoritmo 3.2 (*linha* 2);
- se o grau for 0 (zero), não há entidade a ser incorporada, então retorna *null* (linhas 3 à 5);
- cria *vetor Einc* para salvar as entidades incorporáveis do relacionamento (linha 7);
- para cada entidade (E_E) do relacionamento (R_n) com grau > 2, calcula o grau de incorporação da entidade, G_I (algoritmo 3.1). Ao achar a primeira entidade com $G_I = 1$, se o grau de incorporação do relacionamento (G_IR) for 1, então esta entidade é a incorporável e encerra a função retornando esta entidade. Se $G_IR > 1$, salva esta e as demais entidades com $G_I = 1$ no *vetor* (linhas 8 à 19) e ao final do processo o projetista escolhe a entidade a ser incorporada (linha 22);
- trata os relacionamentos de grau 2 (linhas 23 à 42):
  - se o relacionamento tem atributos nenhuma entidade poder ser incorporada. (linha 25);
  - se a cardinalidade do relacionamento for do tipo $m : n, m > 1 \wedge n > 1$, a incorporação não é possível (linha 28);
  - se a cardinalidade for $1 : n$, (linha 31)
    - a incorporação somente é possível se a entidade do lado n tiver grau de incorporação 1 (linhas 34 à 36);
    - senão, deve ser devolvido *null*, indicando que não há nenhuma entidade incorporável (linhas 37 à 40).

Se o grau de incorporação de um relacionamento de grau maior que 2 (> 2) for zero (0), isto é, todas as entidades a ele conectadas não podem ser incorporadas a outros elementos, então, este relacionamento será removido do modelo e em seu lugar será criada uma entidade intermediária (E_R) com relacionamentos de cardinalidade $1 : 1$ e grau 2, entre as entidades conectadas ao relacionamento original e a entidade E_R criada.

Convertidos os relacionamentos de grau maior que 2, o processo segue com a análise dos relacionamentos binários quanto à possibilidade de conversão, com a incorporação de uma das entidades do relacionamento pela outra.

Algoritmo 3.3: Elege Entidade a Ser Incorporada

Input: Relacionamento de Grau n (R_n)
Output: Entidade a Ser Incorporada ($E_{Incorpora}$)

```
1  Function BuscaEntidadeIncorpora(Rn) is
2      grauIR ← CalculaG_IR(R_n); /* algoritmo 3.2 */
3      if grauIR = 0 then
4          /* não há entidade incorporável */
5          return null
6      qtdI1 ← 0
7      cria vetor E_inc com n elementos
8      if n > 2 then
9          /* R_n é de grau > 2 */
10         for ∀E_E | E_E conectada à R_n do
11             grauIE ← CalculaG_I(E_E); /* algoritmo 3.1 */
12             if grauIE = 1 then
13                 /* é entidade incorporável */
14                 if grauIR = 1 then
15                     /* é a única incorporável */
16                     return E_E
17                 else
18                     qtdI1 ← qtdI1 + 1
19                     E_inc[qtdI1] ← E_E
20                 end
21         end
22         analisa entidades em E_inc e faz return E_escolhida
23     else
24         /* R_n é de grau 2 */
25         if Rn ⊂ atributos then
26             /* Se R_n tem atributos Entidades não podem ser fundidas */
27             return null
28         if ( (Cardinalidade(R_n) = m : n) ∧ (m > 1) ∧ (n > 1) ) then
29             /* Se R_n é muitos para muitos Entidades não podem ser fundidas */
30             return null
31         if Cardinalidade()R_n) = 1 : n then
32             /* R_n é 1 : n */
33             grauIn ← CalculaG_I(E_ladoN); /* grau de incorporação da entidade do lando n
                   da relação */
34             if grauIn = 1 then
35                 /* entidade do lado n é incorporável */
36                 return E_ladoN
37             else
38                 /* entidade do lado 1 é a incorporável, mas não pode ser incorporada */
39                 return null
40             end
41     end
42 end
```

Como descrito anteriormente, se houver uma entidade incorporável, seus atributos serão absorvidos pela outra e depois ela e o relacionamento serão removidos do modelo.

O processo de análise e conversão dos relacionamentos está descrito no algoritmo 3.4.

Este algoritmo é executado para todos os relacionamentos do modelo e tem duas etapas distintas. Na primeira, linhas 4 à 24, são tratados os relacionamentos de grau maior que 2. Esta etapa começa com o cálculo do grau de incorporação do relacionamento (G_{IR}) (linha 5) e se este for zero (0) é criado uma nova entidade (E_R), com uma chave primária e com os atributos de R_n (se existirem). Depois cria-se um relacionamento entre cada uma das entidades conectadas à R_n e a nova E_R e exclui-se o R_n (linhas 8 à 15). Ainda na primeira etapa, há o tratamento do relacionamento quando existe pelo menos uma entidade incorporável (linhas 19 à 22). Aqui busca-se a entidade a ser incorporada e, então, seus atributos são absorvidos pelo relacionamento, seguidos da sua remoção do modelo e consequente remoção da sua conexão com o relacionamento, que, com uma entidade a menos conectada a ele, passa a ser de grau $n-1$ e, então, volta a ser analisado até que seu grau não seja mais maior que 2.

No início da etapa 2, linhas 29 à 36, busca-se, se houver, para R_n, que agora é de grau 2, a entidade a ele conectada que será absorvida pela outra entidade, com a consequente remoção da entidade incorporada e do relacionamento do modelo.

Executado esse passo, o novo modelo entidade relacionamento (`MER2`) pode ser representado pelo grafo $G = E, R$, sem relacionamentos $m : n$ e sem relacionamentos de grau maior que 2, que passa a ter o seguinte conteúdo:

- E - o conjunto das entidades (E_E) do modelo, mais as entidades criadas a partir de relacionamentos (E_R);
- R - os relacionamentos unários (R_1) e binários do tipo $1:1$ (R_{R11}) e $1:n$ (R_{R1n}), mais os relacionamentos criados nesse passo (R_{EE}), também binários.

A figura 3.1 traz o esquema de atividades da fase 1 deste método.

3.1.2.3 Método $M05$ - Fase 2 - Gerando MBDG - Passo 1 - Convertendo Entidades do MER2 para Vértices

Como foi explicado, o objetivo da fase 2 é mapear o modelo E-R resultado da fase 1, $MER2$, para o modelo banco de dados de grafo (`MBDG`).

Algoritmo 3.4: Conversão dos Relacionamentos R_n para R_2

Input: Relacionamento de Grau n (R_n)
Output: Relacionamento (R_n) convertido para Grau 2

```
1  Function ConverteR_n(R_n) is
2      for ∀R_n|R_n ⊂ MER do
3          grau ← n
4          while grau > 2 do
5              gir ← CalculaG_IR(R_n); /* algoritmo 3.2 */
6              if gir = 0 then
7                  /* R_n não tem entidade incorporável */
8                  cria entidade E_R com nome R_n
9                  cria atributo E_R_pk em E_R como sua chave primária
10                 E_R(atributos) ← R_n(atributos)
11                 for ∀E_E|E_E é conectada à R_n do
12                     cria relacionamento E_E ↔ E_R com cardinalidade 1 : 1
13                     remove E_E do relacionamento R_n
14                 end
15                 remove R_n do modelo
16                 grau ← 0; /* R_n não existe mais */
17             else
18                 /* R_n tem pelo menos 1 entidade incorporável */
19                 E_Converte ← BuscaEntidadeIncorpora(Rn); /* algoritmo 3.3 */
20                 R_n(atributos) ← R_n(atributos) + E_Converte(atributos)
21                 Remove E_Converte do relacionamento R_n
22                 Remove entidade E_Converte
23                 grau ← grau − 1
24             end
25         end
26         if grau = 0 then
27             /* R_n não existe mais */
28             Continue
29         /* Agora R_n tem grau 2 */
30         E_Converte ← BuscaEntidadeIncorpora(Rn); /* algoritmo 3.3 */
31         if E_Converte ≠ null then
32             /* R_n existe entidade incorporável */
33             E_Outra ← E_E conectada por R_n à E_Converte
34             E_Outra(atributos) ← E_Outra(atributos) + E_Converte(atributos)
35             remove E_Converte
36             remove R_n
37     end
38 end
```

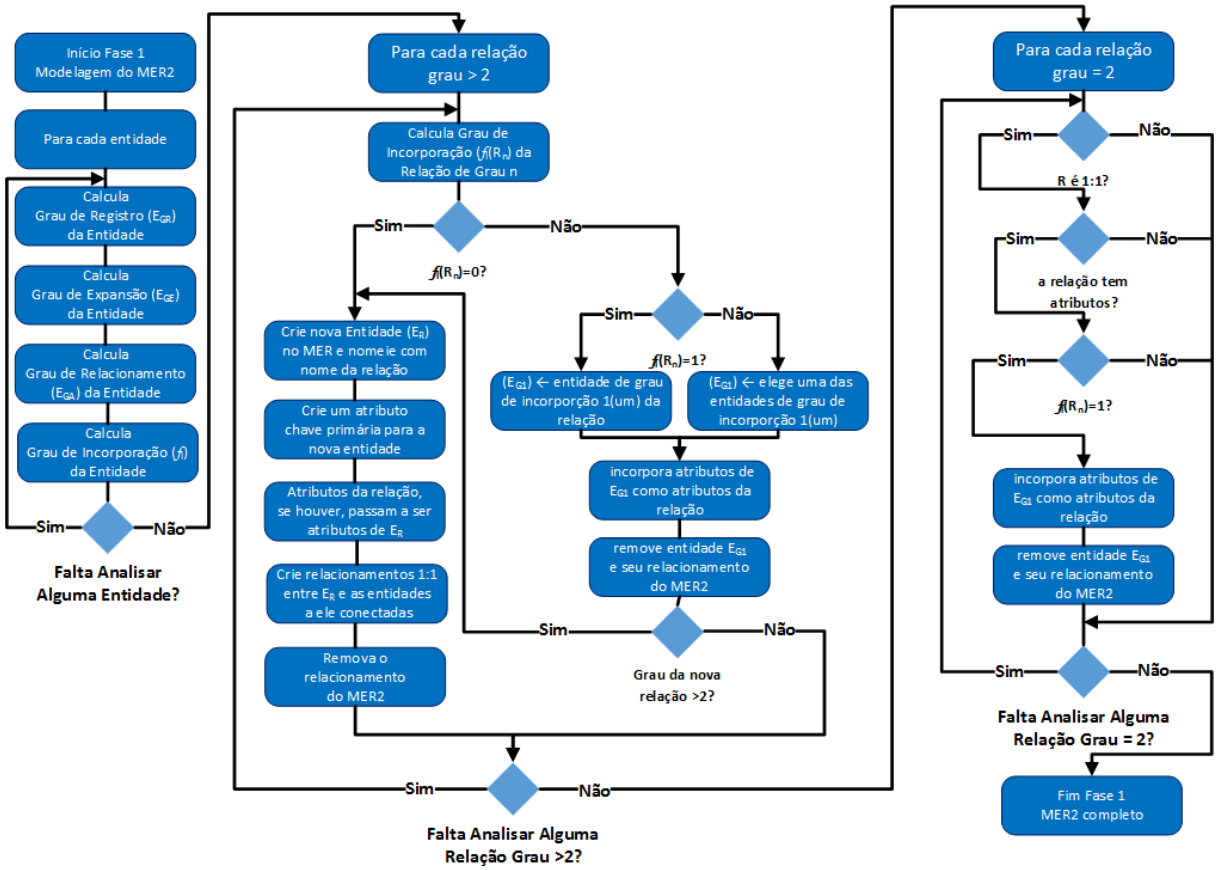


Figura 3.1: Modelo - Método $M05$ - Diagrama de Atividades da Fase 1

Antes de iniciar a conversão do modelo, fica convencionado que os vértices serão intitulados como um nome (substantivo, sigla) e as arestas como algo que denote ação, como um verbo.

No primeiro passo da fase 2, para cada Entidade, $(E_E \in E \wedge E_R \in E)$, será criado um vértice (V) no `MBDG` com o mesmo nome e cujas propriedades serão os atributos da entidade. Caso o vértice criado ainda não possua uma propriedade do tipo chave primária, deverá ser criada uma propriedade para tal fim no vértice. Por chave primária do vértice, entenda-se como aquela propriedade que será usada para identificá-lo nas arestas a ele ligadas.

Este processo está detalhado no algoritmo 3.5. Os passos deste algoritmo são bastante simples. Todas as entidades do `MER2`, as originais de `MER`, E_E, e aquelas originadas de relacionamentos, E_R, serão convertidas para vértices, que receberão seu nome e todos os seus atributos. O único passo adicional será criar uma propriedade para servir como chave primária, se esta ainda não existir na entidade.

Algoritmo 3.5: Converte Entidades em Vértices

```
   Input: Entidades de MER2 (E_E ∪ E_R)
   Output: Vértices do Grafo (V_E)
 1 Function ConverteEntidadeVertice(E_E ∪ E_R) is
 2 |   E_converte ← E_E ∪ E_R
 3 |   for ∀E_converte do
 4 |   |   V_E(nome) ← E_converte(nome)
 5 |   |   V_E(propriedades) ← E_converte(atributos)
 6 |   |   if V_E ⊂ V_E-pk then
 7 |   |   |   continue;
 8 |   |   else
 9 |   |   |   cria V_E-pk em V_E
10 |   |   end
11 |   end
12 end
```

3.1.2.4 Método $M05$ - Fase 2 - Gerando MBDG - Passo 2 - Convertendo Relacionamentos

Após a definição dos vértices no primeiro passo, neste passo será realizada a identificação e conversão dos relacionamentos do modelo, ou seja, os relacionamentos unários (R_{R11}), binários de cardinalidade $1:1$ ou $1:n$ (R_{R11} e R_{R1n}) não convertidos na fase anterior e os binários de cardinalidade $m:n$ (R_{mn}) do `MER` original mais os relacionamentos, todos binários $1:1$, criados na fase anterior (R_{EE}). Assim:

- os relacionamentos unários, R_1, geram arestas bidirecionais do vértice para ele mesmo;
- os relacionamentos binários, R_{R11}, R_{REE}, R_{mn} e R_{EE}, geram arestas bidirecionadas entre os vértices correspondentes às entidades por ele conectadas;
- os relacionamentos R_{R1n} geram arestas direcionadas do vértice correspondente à entidade do lado n para o vértice correspondente à entidade do lado 1 do relacionamento.

Todas as arestas criadas devem ser nomeadas como um verbo, que remeta ao nome ou objetivo do relacionamento.

O algoritmo 3.6 traz o detalhamento deste passo. Este algoritmo percorre todos os relacionamentos unários de `MER2` (linhas 2 à 4) e os substitui por arestas bidirecionais do vértice para ele mesmo. Depois percorre todos os relacionamentos binários de cardinalidade diferente de $1:n$ (linhas 6 à 10) e os transforma em aresta bidirecionais entre os dois

vértices conectados ao relacionamento. E, por fim, percorre os relacionamentos binários $1:n$ (linhas 11 à 14) e cria uma aresta que sai do vértice do lado n do relacionamento para o vértice do lado 1.

Algoritmo 3.6: Converte Relacionamentos em Arestas

Input: Relacionamentos de MER2 ($R_{R11}, R_{EE}, R_{mn}, R_{R1n}$)
Output: Arestas do Grafo (A_{bi}, A_{di})

```
1  Function ConverteRelacionamentoAresta(R_1, R_R11, R_EE, R_mn, R_R1n) is
2      for ∀R_1 do
3          cria aresta bidirecional de V_E para ele mesmo (A_1 = V_E ↔ V_E)
4      end
5      R_bi ← R_R11 ∪ R_EE ∪ R_mn
6      for ∀R_bi do
7          V_E1 ← V_E do lado 1 de R_bi
8          V_E2 ← V_E do lado 2 do R_bi
9          cria aresta bidirecional entre V_E1 e V_E2 (A_bi = V_E1 ↔ V_E2)
10     end
11     for ∀R_R1n do
12         V_E1 ← V_E do lado 1 de R_R1n
13         V_En ← V_E do lado n do R_R1n
14         cria aresta direcionada de V_En para V_E1 (A_di = V_En → V_E1)
15     end
16 end
```

A figura 3.2 traz o esquema de atividades da fase 2 deste método.

3.1.2.5 Método $M05$ - Fase 3 - Gerando MBDG - Passo 1 - Análise das Propriedades dos Vértices

Como visto anteriormente, os `BDGs` permitem a *clusterização* da base entre diversos servidores, mas isso traz um custo no acesso às entidades do banco. Uma das soluções, aqui apontada, é dividir vértices que possuam mais de duas propriedades e possam ter uma grande quantidade de registros, em dois ou mais vértices. Assim, no vértice original (principal) ficariam suas propriedades mais usadas, como aquela que além da chave primária identifica o registro; e as demais propriedades, acessadas eventualmente, ficariam em outros vértices de forma que este poderia ficar num segundo servidor, sem afetar o desempenho do banco nas ações mais comuns.

Além disso, nos experimentos com as bases do capítulo anterior, foi visto que consultas que acessam um único vértice têm melhor desempenho quanto menor for o número de propriedades do vértice.

Esta etapa não envolve as arestas, pois estas não podem apontar para outros elementos

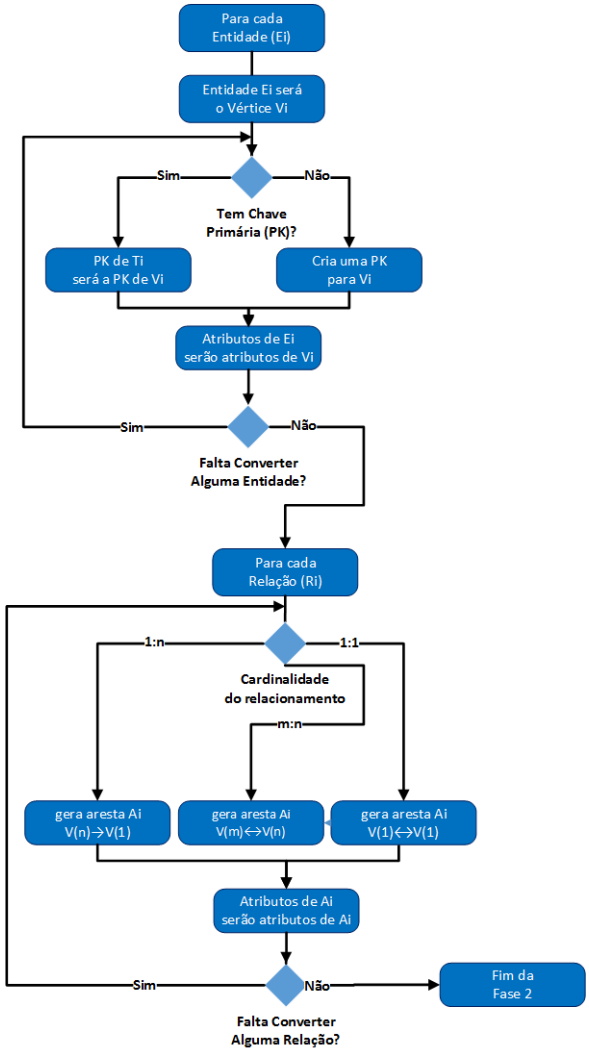


Figura 3.2: Modelo - Método $M05$ - Diagrama de Atividades da Fase 2

que não sejam seus vértices de origem e destino.

Os critérios para determinar se um vértice será dividido ou não são:

- relacionamentos do vértice - a entidade que deu origem ao vértice deve ser amplamente ligada a outras entidades, participando em muitas das consultas ao banco. Num sistema de vendas, por exemplo, as entidades `clientes`, `vendas` e `produtos` são entidades que atendem a este quesito;
- número de propriedades - para ser dividido, o vértice deve ter mais de duas propriedades, pelo menos a chave primária e mais duas;
- uso das propriedades - as propriedades que serão migradas para outro vértice devem ser propriedades que não são de uso frequente. Por exemplo, nesse mesmo sistema de vendas, na entidade `cliente`, seu `nome` e `cpf` são de uso frequente, já `endereço`

e `ocupação profissional` são propriedades de uso eventual e que podem ir para outro vértice.

Para cada vértice a ser convertido, pode ser escolhida a migração sintética ou a analítica. Na migração sintética, todos as propriedades que serão migradas irão para um único novo vértice. Já na migração analítica, cada propriedade do vértice pode gerar um outro vértice, a semelhança do passo final do método $M03$. A migração pode, também, a critério do projetista, ser híbrida, onde algumas propriedades são agrupadas num único vértice e outras geram um vértice por propriedade. No sistema de vendas, por exemplo, na entidade `cliente`, os campos `endereço`, `número do RG`, `nome de pai` e `nome da mãe`, que variam de cliente a cliente, podem ser migrados para um único vértice, já a `data de nascimento` do `cliente`, que pode se repetir em vários clientes, pode ser migrada para um vértice de nome *Data*.

Na migração sintética será gerado um novo vértice com as propriedades selecionadas e mais a chave primária do vértice original, que será comum aos dois vértices. O nome do novo vértice será o nome do vértice original mais um adendo que o identifique como complemento do original, por exemplo, a letra O de Outros. Deve ser criada, ainda, uma aresta direcionada do vértice original para o novo vértice, com as chaves primárias como identificadores de entrada e saída da aresta.

Na migração analítica, para cada propriedade selecionada, será criado um novo vértice com o nome da propriedade, e esta será seu único atributo e também sua chave primária. Deve ser criada, também, uma aresta direcionada do vértice original para o novo vértice, sendo a chave primária do vértice a entrada da aresta e a chave primária do novo vértice sua saída. Ao popular o banco, antes de gravar um vértice, deve ser verificado se já não existe o vértice. Se existir, será gerada apenas uma aresta apontando para o dado que já existe. Caso contrário, serão gravados o vértice e a aresta;

Este processo pode ser acompanhado no algoritmo 3.7. Este algoritmo é executado para todos os vértices do modelo e são descartados do processo aqueles com menos do que três propriedades e os pouco relacionados (linhas 3 à 6). Para os vértices selecionados, suas propriedades são separadas entre as fixas no vértice (linhas 9 à 14), as de perfil sintético e as de perfil analítico. As de perfil sintético geram um vértice próprio, com ela de único atributo (linhas 20 à 22) e as analíticas vão ser agrupadas num único novo vértice (linhas 25 à 31).

Encerrado o quarto passo, o modelo está convertido.

Algoritmo 3.7: Analisa e Particiona Vértices

Input: Vértices do Grafo (V_E)
Output: Vértices Convertidos ($V_O \wedge V_{pr}$))

```
1  AnalisaParticionaVertices(V_E)
2  for ∀V_E do
3      if V_E(quantidade_propriedades) < 3 then
4          continue;
5      if V_E não é amplamente relacionado then
6          continue;
7      cria vetor propriedadesAnalitica; ind ← 0
8      for ∀p_r|p_r é propriedade de V_E do
9          if p_r é pk then
10             continue;
11         if p_r é propriedade identificadora then
12             continue;
13         if p_r é propriedade de uso frequente then
14             continue;
15         if p_r tem perfil analitico then
16             ind ← ind + 1
17             propriedadesAnalitica[ind] ← p_r
18             continue;
19         /* p_r tem perfil sintético */
20         cria vértice V_pr
21         p_r será propriedade e pk de V_pr
22         cria aresta direcionada de V_E para V_pr (A_di = V_E → V_pr)
23     end
24     if ind > 0 then
25         /* cria V_O com propriedades de perfil analítico */
26         cria vértice V_O
27         V_Opk ← V_Epk
28         foreach p_r in propriedadesAnalitica do
29             V_O ← p_r
30         end
31         cria aresta direcionada de V_E para V_O (A_di = V_O → V_O)
32 end
```

A figura 3.3 traz o esquema de atividades da fase 3 deste método.

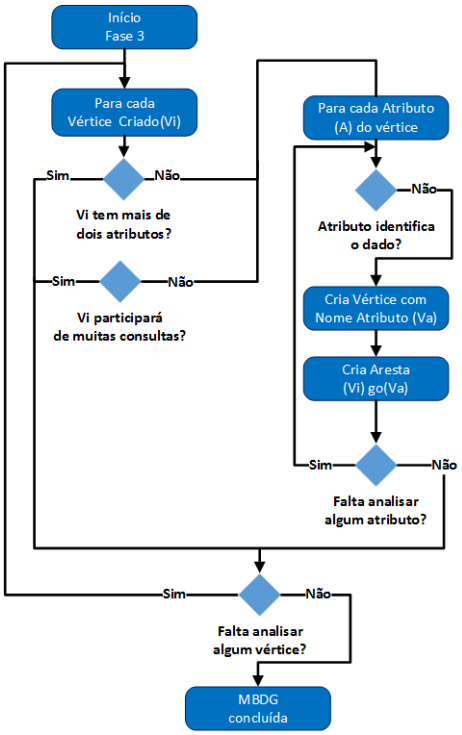


Figura 3.3: Modelo - Método $M05$ - Diagrama de Atividades da Fase 3

3.1.3 Modelando a $BaseTeste$ com o Novo Método

A exemplo do passo a passo executado na modelagem para as bases voltadas a grafos segundo os quatro métodos analisados no capítulo anterior, agora será demonstrado o procedimento para a modelagem da $BaseTeste$ seguindo os passos do método proposto.

Como foi explicado, o ponto de partida será o modelo entidade relacionamento da base (2.29.

Montando o universo do grafo tem-se:

- os elementos de E:
  - o conjunto E_E das entidades, representado por $E_E = \{A, B, C, D, E, F, G, H\}$;
  - o conjunto vazio E_{Rmn}, pois não existem relações $m : n$ na $BaseTeste$;
  - o conjunto E_{R3} dos relacionamentos de grau maior que 2, representado por $E_{R3} = \{DgoBgoD, DgoCgoC, DgoEgoG\}$;
- os elementos de R:

- o conjunto dos relacionamentos unários, R_1, que é vazio, pois não existem relacionamentos deste tipo na $BaseTeste$;
- o conjunto dos relacionamentos do tipo 1 : 1, representado por: $R_{R11} = \{HgoC1, HgoC2\}$;
- o conjunto dos relacionamentos do tipo 1 : n, representado por: $R_{R1n} = \{DgoA, GgoF\}$.

Assim a representação do grafo é expressa como:

G= {{A, B, C, D, E, F, G, H,DgoBgoD, DgoCgoC, DgoEgoG},{HgoC1, HgoC2, DgoA, GgoF}} (3.1)

As subseções seguintes descrevem os passos de cada uma das fases deste método.

3.1.3.1 Modelando a $BaseTeste$ com o Novo Método - Fase 1 - Passo 1

O primeiro passo é calcular o grau de incorporação das entidades E_E do modelo, que é formado pela combinação do grau de expansão (G_E) e do grau de relacionamento (G_R) de cada uma delas.

Conforme foi visto na subseção 2.5.1, as entidades A, D, H e G têm perfil de atualização constante e F tem perfil de atualização provável, então o grau de expansão delas é 0 (zero). As demais entidades, B, C e E, têm perfil de atualização pouco provável, isso implica que seu grau de expansão é 1.

Analisando o modelo entidade relacionamento da base, figura 2.29, pode-se calcular o grau de relacionamento das entidades. Assim:

- A participa apenas de 1 relacionamento, $DgoA$, então seu grau de relacionamento (G_R) é 1;
- B participa apenas de 1 relacionamento, $DgoBgoD$, então tem $G_R = 1$;
- C participa de 3 relacionamentos, $DgoCgoC$, $HgoC1$ e $HgoC2$, então tem $G_R = 0$. Mesmo que C participe duas vezes do relacionamento $DgoCgoC$, deve ser somado apenas 1;
- D participa de 4 relacionamentos, $DgoA$, $DgoBgoD$, $DgoCgoC$ e $DgoEgoG$, então $G_R = 0$. Mesmo que D participe duas vezes do relacionamento $DgoBgoD$, deve ser somado apenas 1;

- E participa apenas de 1 relacionamento, $DgoEgoG$, então $G_R = 1$;
- F participa apenas de 1 relacionamento, $GgoF$, então $G_R = 1$;
- G participa de 2 relacionamentos, $DgoEgoG$ e $GgoF$, então $G_R = 0$;
- H participa de 2 relacionamentos $HgoC1$ e $HgoC2$, então $G_R = 0$.

Calculados os graus de Expansão (G_E) e de Relacionamentos (G_R) das entidades, agora podem ser calculados seus respectivos graus de incorporação (G_I), onde, se a soma de G_E e G_R for igual à dois (2) então G_I será um (1), senão será zero (0). Assim, as entidades B e E têm grau de incorporação 1 e as demais 0, isto é, apenas essas duas entidades é que podem ser incorporadas por um relacionamento ou por outra entidade do modelo.

O resumo deste cálculo está na tabela 3.1.

Tabela 3.1: Modelo - Método $M05$ - Exemplo - Fase 1 - Passo 1 - Cálculo do Grau de Incorporação de E_E

E_E	Grau de Expansão		Grau de Relacionamento		$G_I(E_E)$= se$(G_E + G_R) = 2$ então 1 senão 0
	Expansão	G_E	Relações	G_R	
A	constante	0	1	1	0
B	pouco provável	1	1	1	1
C	pouco provável	1	3	0	0
D	constante	0	4	0	0
E	pouco provável	1	1	1	1
F	provável	0	1	1	0
G	constante	0	2	0	0
H	constante	0	2	0	0

3.1.3.2 Modelando a $BaseTeste$ com o Novo Método - Fase 1 - Passo 2

Calculados os graus de incorporação das entidades, agora as relações de grau maior que 2 podem ser analisadas e modeladas. Assim, neste passo vão ser analisadas e modeladas as relações de $E_{R3} = \{DgoBgoD, DgoCgoC, DgoEgoG\}$.

O passo inicia calculando o grau de incorporação (G_{IR}) de cada um dos relacionamentos, que nada mais é do que a soma do grau de incorporação das entidades que participam deles.

A tabela 3.2 traz o resumo deste cálculo, que depois vem detalhado na modelagem de cada relacionamento.

Tabela 3.2: Modelo - Método $M05$ - Exemplo - Fase 1 - Passo 2 - Cálculo do Grau de Incorporação de E_{R3}

		Entidades Participantes						GIR(R3)
Relacionamento	Grau	EE1	GI(EE1)	EE2	GI(EE2)	EE3	GI(EE3)	GI(EE1)+GI(EE2)+GI(EE3)
DgoBgoD	3	D	0	B	1	D	0	1
DgoCgoC	3	D	0	C	0	C	0	0
DgoEgoG	3	D	0	E	1	G	0	1

O relacionamento $DgoCgoC$ tem conectadas a ele as entidades D e C duas vezes. Conforme pode ser visto na tabela 3.1, as duas entidades tem $G_R = 0$, então o grau de incorporação do relacionamento será zero ($G_{IR}(DgoCgoC) = 0$), ou seja, nenhuma delas pode ser incorporada a outra entidade ou relacionamento.

Neste caso, $DgoCgoC$ será assim convertido:

- cria-se uma nova entidade no modelo (E_R) e atribui-se a ela o nome de DC;
- DC recebe um atributo de nome *dc_pk* para servir como sua chave primária;
- os atributos de $DgoCgoC$ passam a ser atributos de DC;
- cria-se um relacionamento do tipo $1:1$ (R_{11}) entre a entidade D e a nova entidade DC. Este relacionamento recebe o nome $DgoDC$;
- finalmente são criados dois relacionamentos do tipo $1:1$ (R_{11}) entre a entidade C e a nova entidade DC com, respectivamente, o nome $DgoDC1$ e $DgoDC2$.

A figura 3.4 mostra a conversão efetuada.

O próximo relacionamento a ser modelado é o $DgoBgoD$. O cálculo de seu grau de incorporação resulta em $G_{IR}(DgoBgoD) = 1$, já que D tem grau de incorporação zero (0) e B um (1) (vide tabela 3.1). Isso indica que B pode ser removida do modelo e incorporada ao relacionamento. Então é executada a seguinte sequência de passos:

- a entidade B é removida do modelo e seus atributos são incluídos como atributos do relacionamento $DgoBgoD$;
- o nome do relacionamento é alterado para $DgoD$. Essa ação em princípio é opcional, pode ser mantido o nome original;

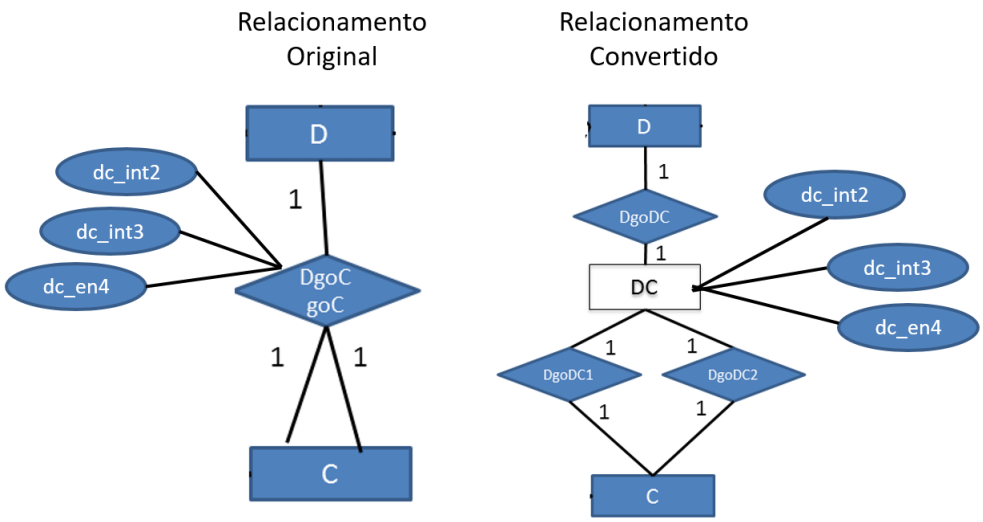


Figura 3.4: Modelo - Método $M05$ - Exemplo - Fase 1 - Passo 2 - Convertendo DgoCgoC

- como agora $DgoD$ é um auto-relacionamento da entidade D, passou a ser uma relação unária e nenhuma modelagem adicional será feita.

A figura 3.5 mostra a conversão efetuada. Os atributos de B foram colocados no gráfico, mesmo que os demais não tenham sido colocados, para facilitar a visualização da conversão ocorrida.

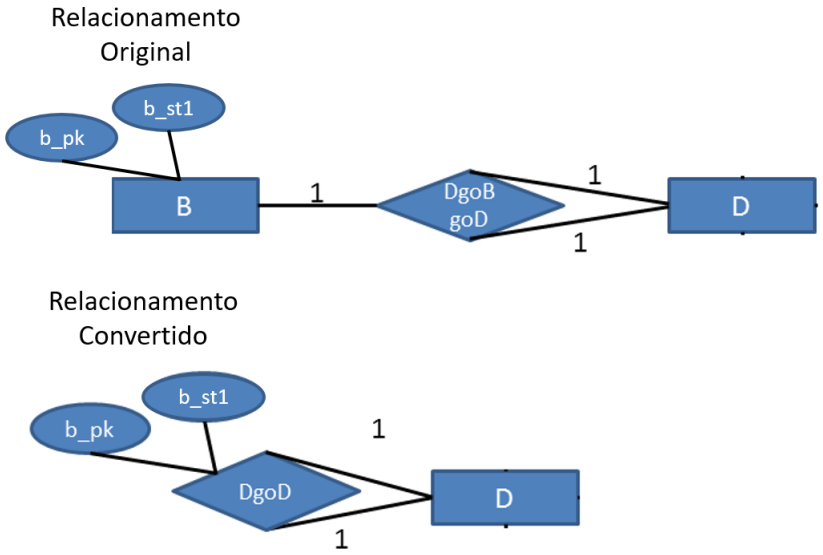


Figura 3.5: Modelo - Método $M05$ - Exemplo - Fase 1 - Passo 2 - Convertendo DgoBgoD

O último relacionamento de E_{R3} a ser convertido é o $DgoEgoG$. As entidades envolvidas no relacionamento, D, E e G têm, respectivamente, conforme pode ser visto na tabela 3.1, grau de incorporação zero (0), um (1) e zero (0), resultando num grau de incorporação do relacionamento um ($G_{IR} = 1$). Assim, a entidade de $G_I = 1$, E, pode

ser incorporada ao relacionamento. Desta forma, segue-se que:

- a entidade E é removida do modelo e seus atributos são incluídos como atributos do relacionamento $DgoEgoG$, que passa a se chamar $DgoG$;
- $DgoG$ passa a ser uma relação de grau 2 e cardinalidade $1:1$, conectando D com G.

A figura 3.6 mostra a conversão efetuada. Os atributos de E foram colocados no gráfico, mesmo que os demais não tenham sido, para facilitar a visualização da conversão ocorrida.

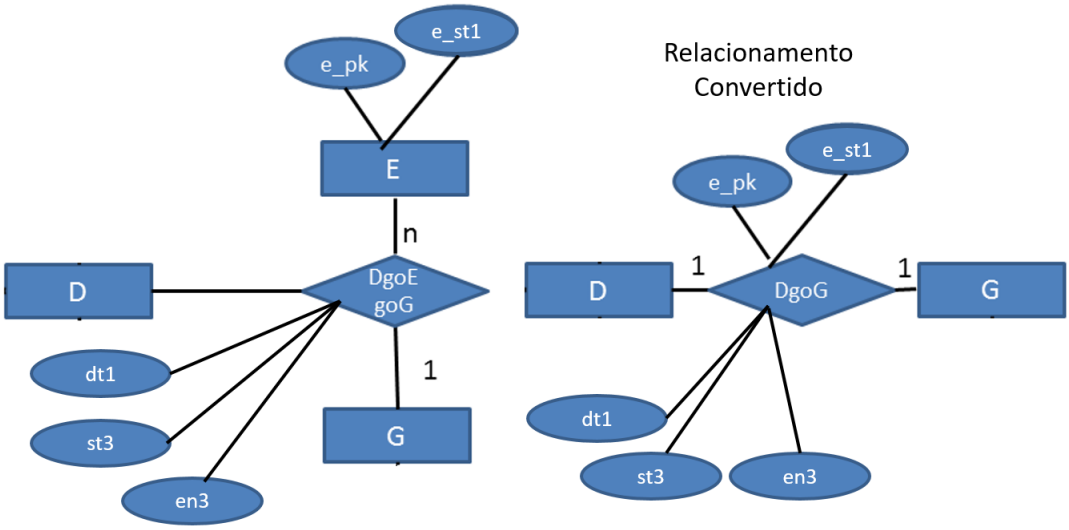


Figura 3.6: Modelo - Método $M05$ - Exemplo - Fase 1 - Passo 2 - Convertendo DgoEgoG

3.1.3.3 Modelando a $BaseTeste$ com o Novo Método - Fase 1 - Passo 3

No próximo passo, devem ser analisadas as relações binárias do tipo $1:1$ ($R_{R11} = \{HgoC1, HgoC2\}$).

As duas relações envolvem as mesmas entidades, C e H, que têm grau de incorporação zero ($G_I = 0$). Assim, o grau de incorporação das duas relações será, também, zero ($G_{IR} = 0$), o que indica que nenhuma das duas entidades pode ser incorporada à outra. Então, essas duas relações permanecem como estão.

A fase 1 está concluída e o novo modelo entidade relacionamento da base está pronto (`MER2`).

Os elementos do grafo agora passam a ser:

- os elementos de E:
  - as entidades originais do MER (E_E) não removidas na fase 1, representado por $E_E = \{A, C, D, F, G, H\}$;
  - a entidade criada na fase 1 (E_R), representada por $E_R = \{DC\}$;
- os elementos de R:
  - os relacionamentos do tipo 1 : 1 originais do modelo e não removidos na fase 1, representado por: $R_{R11} = \{HgoC1, HgoC2\}$;
  - os relacionamentos do tipo 1 : n não tratados na fase 1, representado por: $R_{R1n} = \{DgoA, GgoF\}$;
  - os relacionamentos do tipo 1 : 1 criados na fase 1, representado por $R_{EE} = \{DgoD, DgoDC, DCgoC1, DCgoC2, DgoG\}$.

Este novo modelo, o qual pode ser visto na figura 3.7, será analisado e modelado como grafo na próxima fase.

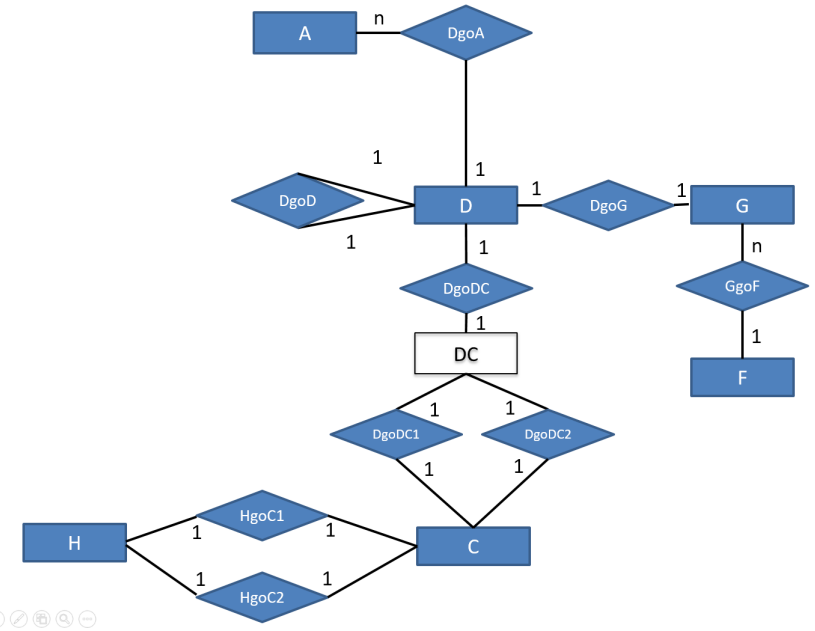


Figura 3.7: Modelo - Método $M05$ - Exemplo - Fase 1 - Passo 2 - MER2

3.1.3.4 Modelando a *BaseTeste* com o Novo Método - Fase 2 - Passo 1

Nesta fase o novo modelo entidade relacionamento (`MER2`) será convertido para MBDG.

O primeiro passo é converter as Entidades do modelo (E_E e E_R) para vértices.

O nome das entidades é uma sigla, então o nome de cada vértice será o mesmo nome da respectiva entidade. Todas as entidades já têm uma chave primária, então não é necessário criar essa propriedade para nenhum vértice. Todos os atributos das entidades passam a ser propriedades dos vértices delas originados.

Por consequência, foram criados os vértices A, C, D, F, G, H e DC. A figura 3.8 traz as entidades convertidas para vértices e suas propriedades.

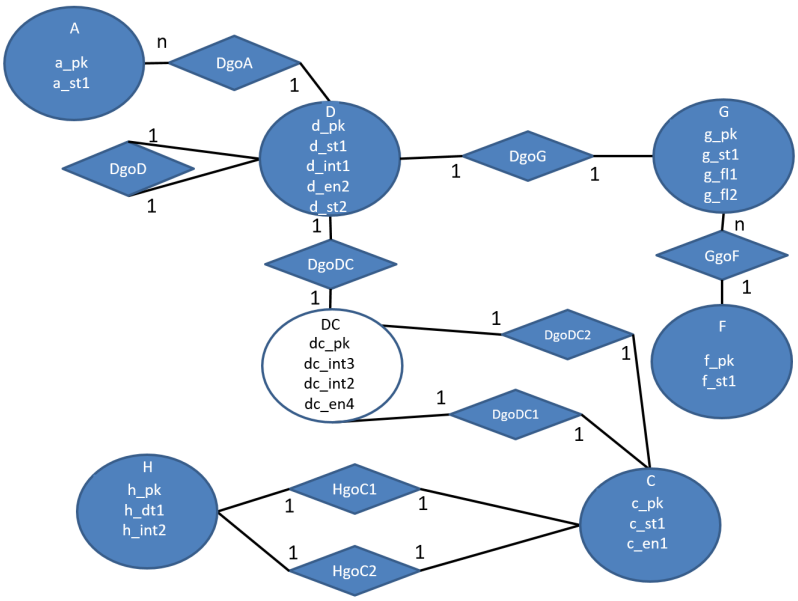


Figura 3.8: Modelo - Método $M05$ - Exemplo - Fase 2 - Passo 1 - Vértices

3.1.3.5 Modelando a $BaseTeste$ com o Novo Método - Fase 2 - Passo 2

Convertidas as entidades, o próximo passo é realizar a conversão dos relacionamentos.

Na $BaseTeste$ não existem relacionamentos $m : n$, apenas relações $1 : 1$ e $1 : n$. Então, a conversão destes relacionamentos em arestas será feita da seguinte forma:

- os relacionamentos do tipo $1 : 1$, $R_{EE} = \{DgoD, DgoDC, DCgoC1, DcgoC2, DgoG\}$ e $R_{R11} = \{HgoC1, HgoC2\}$ vão gerar arestas bidirecionais:
  - $HgoC1$ gera a aresta $HgoC\_1$ entre os vértices C e H, sem propriedades adicionais;
  - $HgoC2$ gera a aresta $HgoC\_2$ entre os vértices C e H, sem propriedades adicionais;
  - $DgoD$ gera a auto-aresta $DgoD$ com origem e destino no vértice D e as propriedades b_pk e b_st1, originadas da entidade B, que foi incorporada ao rela-

cionamento na fase 1;

- $DgoDC$ gera a aresta $DgoDC$ entre os vértices D e DC, sem propriedades adicionais;
- $DCgoC1$ gera a aresta $DCgoC\_1$ entre os vértices DC e C, sem propriedades adicionais;
- $DCgoC2$ gera a aresta $DCgoC\_2$ entre os vértices DC e C, sem propriedades adicionais;
- $DgoG$ gera a aresta $DgoG$ entre os vértices D e G. As propriedades desta aresta serão os atributos originais do relacionamento (de_dt2, de_st3, de_en3), mais os atributos da entidade E que foi incorporada a ele (e_pk, e_st1);

- os relacionamentos do tipo $1:n$, $R_{R1n} = \{DgoA, GgoF\}$ vão gerar arestas dirigidas do lado n para o lado 1:
  - $DgoA$ gera a aresta $DgoA$, dirigida de D para A, com a propriedade da_int4;
  - $GgoF$ gera a aresta $GgoF$, dirigida de G para F, sem propriedades adicionais.

A figura 3.9 traz as arestas geradas a partir dos relacionamentos do MER2.

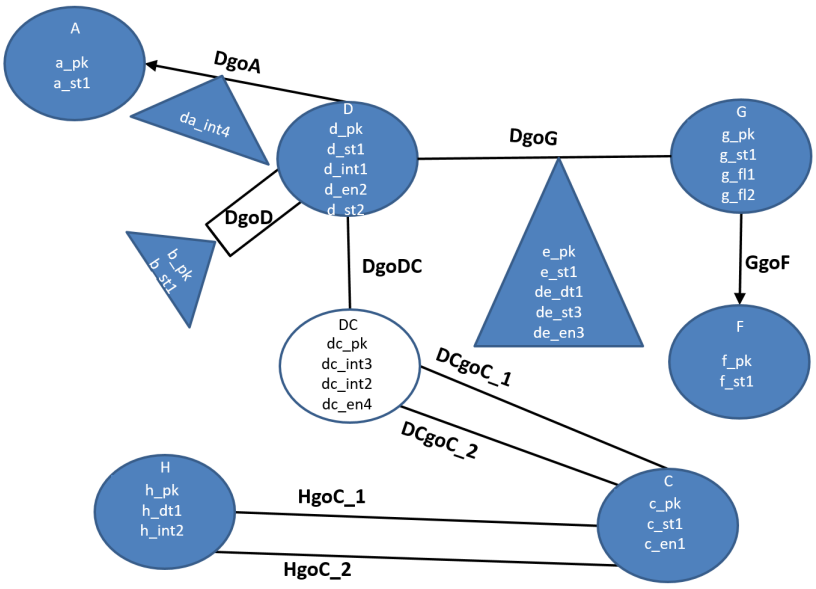


Figura 3.9: Modelo - Método $M05$ - Exemplo - Fase 2 - Passo 2 - Arestas

3.1.3.6 Modelando a $BaseTeste$ com o Novo Método - Fase 2 - Passo 3

Convertidos os vértices e arestas, resta agora apenas verificar se algum dos vértices pode ser dividido em dois ou mais vértices.

Pelo critério de possuir mais de duas propriedades, são eliminados os vértices A e F.

Pelo segundo critério, vértice deve ser amplamente ligado a outros vértices e ser peça chave nas consultas ao banco, são descartados os vértices C, G e H, restando apenas o vértice D.

Analisando as propriedades de D tem-se:

- d_pk - é a chave primária, então deve permanecer no vértice;
- d_st1 - junto com a chave primária individualiza os dados do vértice, então também permanece;
- as propriedades d_int1, d_en2 e d_st2 não são propriedades que individualizam os dados, então podem ser migrados para o novo vértice.

Na definição de D (subseção 2.5.1), viu-se que d_st2 pode receber qualquer tipo de valor, inclusive nulo e não único e d_int1 um valor qualquer entre 180 e 267, então, optou-se por fazer a migração sintética, isto é, gerar um único novo vértice com as três propriedades. Portanto será criado um novo vértice de nome DO com as propriedades d_int1, d_en2 e d_st2, que serão removidas de D, mais a propriedade d_pk, que será comum aos dois vértices, sendo a chave primária de ambos.

E, finalmente, será gerada uma aresta dirigida de D para DO de nome $DgoDO$.

Encerrada essa fase, o processo de modelagem da $BaseTeste$ para uma base voltada a grafos está concluída e o resultado final pode ser visto na figura 3.10.

3.2 Comparando o Modelo Final com os Métodos Anteriores

Nesta seção será apresentada a comparação entre o processo de modelagem dos métodos anteriores com o método proposto e, também, uma análise dos modelos finais dos cinco métodos aqui analisados.

Comparando o processo de modelagem do método proposto e aqueles apresentados no capítulo anterior vê-se duas diferenças fundamentais entre eles, onde o novo método:

- prevê a possibilidade de uma entidade do modelo ser incorporada por um relacionamento;

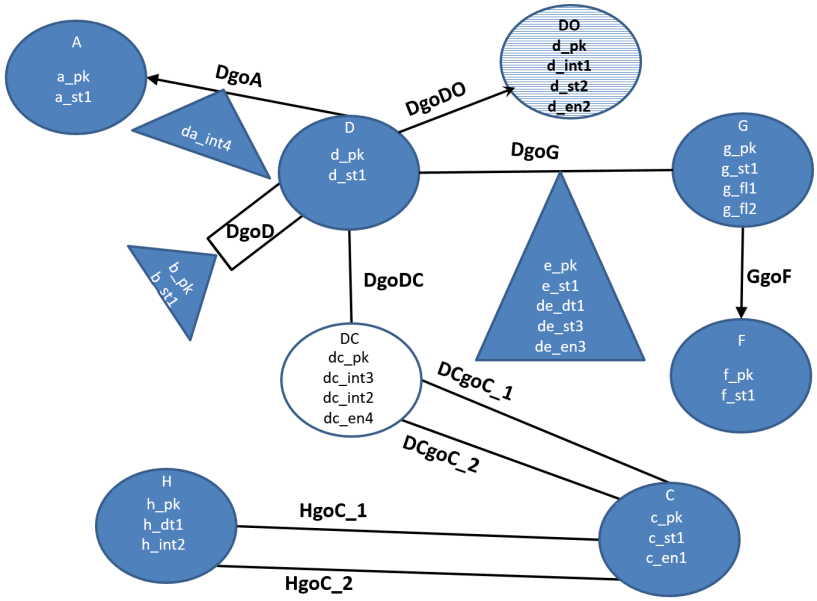


Figura 3.10: Modelo - Método $M05$ - Exemplo - Base Modelada para Grafos

- exige um conhecimento maior das regras de negócio do sistema para o qual a base de dados voltada a grafos está sendo modelada, enquanto os métodos anteriores têm apenas a aplicação de algoritmos de conversão, não exigindo um conhecimento maior do conteúdo da base. Este ponto pode ser visto como uma desvantagem do método em relação aos demais uma vez as demais bases, ao seguirem apenas métodos algorítmicos, pode ser modeladas sem iteração com o usuário, resultando, sempre, em based iguais. Entretanto, em vez de ser uma desvantagem, a aplicação das regras do negócio à modelagem da base pode gerar bases mais adequadas a essas regras.

O método proposto, à semelhança do método $M02$, começa pelo modelo entidade relacionamento da base, enquanto os métodos $M01$, $M03$ e $M04$ começam do modelo relacional. Outra diferença, em relação aos demais métodos é a primeira fase, onde um novo modelo entidade relacionamento, sem relacionamentos de grau > 2 é modelado.

Os passos 1 e 2 da fase 2 do novo método são, em essência, a aplicação dos métodos $M01$ ou $M02$ e o passo final (passo 3 da fase 2) tem correspondência com o último passo do método $M03$, a diferença está em que este último gera um vértice para cada atributo não identificador e o método proposto permite, a critério do projetista, que seja gerado apenas um novo vértice independente da quantidade de atributos.

O passo 1 da fase 1, cálculo do grau de incorporação das entidades, traz alguma semelhança com o método $M04$, ao prever a possibilidade de incorporar entidades a

outras entidades, mas ao contrário deste último, o método proposto não permite que uma mesma entidade seja incorporada a duas entidades diferentes e ainda prevê a possibilidade de incorporação de entidades por relacionamentos.

Quanto ao número de elementos, vértices e arestas, gerados ao final da modelagem, o método proposto ficou com um menor número de vértices e arestas que os modelos finais dos métodos $M01$, $M02$ e $M03$, e maior que o método $M04$.

A tabela 3.3 traz um comparativo entre o método proposto e os quatro métodos analisados no capítulo anterior quanto ao número de vértices e arestas.

Método	Vértices	Arestas
$M01$ - 3NF - Equivalent Graph	11	13
$M02$ - Reference Graph	11	13
$M03$ - Grafos Simples (RDF)	22	27
$M04$ - Modelagem Dirigida	6	5
$M05$ - Modelagem Participativa	8	10

Tabela 3.3: Modelo - Comparativo entre o Método Proposto e os Anteriores

A figura 3.11 traz o resultado final da modelagem dos 05 métodos.

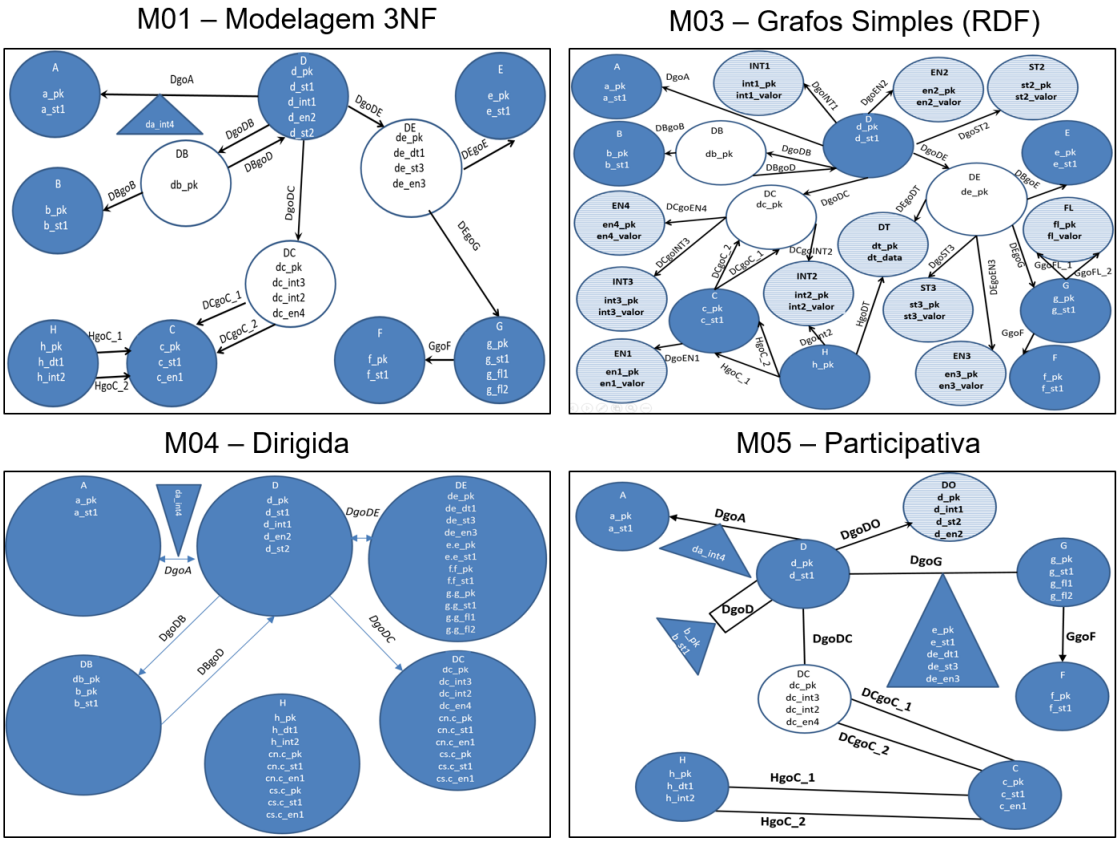


Figura 3.11: Modelo - Comparativo do Modelo Final de Cada Método

No próximo capítulo são apresentados os resultados da implementação da base mode-

lada em Neo4j e a comparação desta base com as bases dos métodos anteriores em termos de tamanho, desempenho e manutenção.

4 Experimentos e Resultados

Como apresentado na seção 2.3, os BDGs vêm sendo bastante utilizados como um serviço de dados, em paralelo a uma base de dados relacional. Assim, para comparar os métodos de modelagem para bancos de dados apresentados no capítulo 2, optou-se por realizar um experimento dividido em quatro fases:

1. criação das bases de dados - onde foram criadas uma base *PostgreSQL* e quatro (4) bases de dados Neo4j, conforme as modelagens analisadas no capítulo 2 e a modelagem proposta nesta dissertação;

2. migração dos dados - os dados da base relacional foram migrados para as bases do banco Neo4j;

3. realização de consultas - foram realizadas 18 diferentes consultas sobre as bases de dados do banco Neo4j;

4. análise dos resultados - os tempos médios de execução de cada consulta foram analisados para determinar os melhores resultados de desempenho.

A fase 1, criação das bases de dados, prevê os seguintes passos:

- construir, em *PostgreSQL*[1], a base $BaseTeste$, definida na subseção 2.5.1;

- implementar esta $BaseTeste$ com duas quantidades diferentes de registros para a entidade D. A primeira com cem mil registros em D ($Base100Mil$) e a segunda com um milhão de registros para esta entidade ($Base1Milhao$);

- a partir do modelo relacional da base criada, construir, em Neo4j, uma base conforme as modelagens resultantes dos métodos `M01 - Modelagem 3NF` e `M02 - Modelagem Reference Graph`, daqui em diante tratada como $BaseM01$;

[1] https://www.postgreSQL.org

- da mesma forma, criar as bases $BaseM03$ e $BaseM04$, conforme os métodos `M03 - Modelagem para Grafos Simples (RDF)` e `M04 - Modelagem Dirigida`;
- seguindo os passos do método $M05$, a partir do modelo entidade relacionamento da $BaseTeste$, criar, em Neo4j, a $BaseM05$.

A próxima fase foi a migração dos dados, através dos seguintes passos:

- exportar as bases *Base100Mil* e *Base1Milhao* para as bases *basecemil.m01*, *basecemmil.m03*, *basecemmil.m04*, *basecemmil.m05*, *base1milhao.m01*, *base1milhao.m03*, *base1milhao.m04* e *base1milhao.m05* na forma de arquivos `CSV`[2] com os vértices e arestas de cada modelo;
- importar para as bases do banco Neo4j os respectivos arquivos `CSV`.

Na fase seguinte, realização das consultas, foram realizadas 18 diferentes consultas em *Cypher*, para cada uma das oito bases e foram anotados os resultados, em milissegundos, do tempo de execução de cada consulta/base.

Finalmente, na última fase, os resultados foram analisados e assim apontados os pontos fortes e fracos de cada método.

Este capítulo está dividido em duas seções distintas. A primeira traz a descrição do processo de geração dos dados utilizados nos experimentos, seguida da seção onde são apresentadas as consultas realizadas com os cinco métodos e a análise dos resultados. Embora os experimentos tenham sido inicialmente executados para as bases $BaseM01$, $BaseM03$ e $BaseM04$ e, somente depois de analisados esses resultados, executados para a base $BaseM05$, dados como o tamanho das bases em disco e número de vértices e arestas apresentados juntos para evitar a repetição de informações, já as tabelas com os tempos de execução de cada consulta vêm primeiro apenas com os métodos $M01$, $M03$ e $M04$, e depois são repetidas com o acréscimo dos tempos para o método $M05$.

4.1 Geração da Base de Dados (BaseTeste)

Conforme o modelo relacional da $BaseTeste$ (figura 2.30), foram implementadas as tabelas A, B, C, D, E, F, G, H, DA, DB, DC e DE.

[2] `CSV` é o acrônimo para *Comma Separated Value*. Diz-se dos arquivos texto utilizados na importação e exportação de registros entre bancos de dados, onde os valores para cada coluna dos registros exportados ou importados vêm separados por vírgulas, daí o nome. Entretanto, outro caractere pode ser definido como o separador das colunas, se a vírgula for um valor possível para algumas colunas.

A *BaseTeste* implementada simula uma base para acompanhamento de um grupo de pessoas (entidade D) desde o nascimento até a morte. Estas pessoas relacionam-se com outras pessoas e essa relação pode ser como 'amigo', 'familiar', etc (entidade B). Esta pessoa tem características que a descrevem, como 'simpática', 'inteligente', 'agitada', etc. Esses adjetivos compõem a entidade A.

O acompanhamento das pessoas é feito a partir da análise de diferentes eventos (entidade E), como 'nascimento', 'entrada na escola', 'morte do pai', 'casamento', etc. Cada evento ocorreu numa data e hora específicas e por ocasião do evento a pessoa estava numa cidade (entidade G), localizado num país específico (entidade F).

A cada evento é registrado para a pessoa a posição planetária de determinados corpos celestes registrados na entidade C ('Sol em Áries', 'Lua em Touro', 'Terra em Virgem', etc.). A posição destes corpos celestes está registrada por data nos elementos gravados na entidade H e com base nesta informação é calculada a posição do corpo celeste na hora do evento (dados registrados em DC).

As tabelas A, B, C, E, F, G e H foram atualizadas com dados de uma base própria do autor:

- A recebeu um total de 6.186 registros, com adjetivos;
- B recebeu um total de 2 registros, com tipos de relacionamento entre as pessoas;
- C recebeu um total de 36 registros, 3 grupos diferentes de 12 registros com diferentes elementos astronômicos;
- E recebeu um total de 6 registros, uma relação de diferentes tipos de eventos;
- F recebeu um total de 247 países;
- G recebeu um total de 136.829 localidades (cidades), com seu nome, latitude e longitude;
- H recebeu um total de 554.788 registros, uma relação de datas e posição de corpos celestes à zero hora da data indicada.

As demais tabelas receberam um número variável de registros. Como já foi dito, na primeira bateria de testes, a tabela D recebeu cem mil registros (*Base100Mil*) e na segunda bateria um milhão (*Base1Milhao*). As tabelas DA, DB, DC e DE receberam uma quantidade variável de registros em função dos registros de D.

O programa de geração dos dados seguiu a seguinte lógica:

- tabela D - como já dito, na *Base100Mil* foram gerados cem mil registros e na *Base1Milhao* um milhão de registros:
  - d_st1 - foi gerado com nomes próprios fictícios, cada nome composto de um *firstname* e um *lastname*, obtidos de uma base com 5.494 *firstnames* e 88.799 *lastnames* disponível em Quiet Affiliate Marketing (2009), com o cuidado de evitar nomes repetidos;
  - d_en2 - os registros com a parte *firstname* do nome próprio gerado terminado com as letras a ou y receberam o valor 2, os terminados com a letra l, o valor 3 e os demais registros receberam o valor 1;
  - d_int2 - de maneira aleatória receberam um número inteiro entre 180 e 267 dias que simulam o número de dias para o primeiro evento a ser registrado para a pessoa;
  - d_st2 - foi criado um *e-mail* com a primeira letra do *firstname* seguido do *lastname* e da expressão $@ficticio.com$;
  - $grupo1$ - não migrado para nenhuma modelagem, sendo apenas um atributo que foi usado na geração dos elementos de DB. Aleatoriamente, a cada registro foi atribuído um número entre 1 e 90% da quantidade de registros de D;
  - $grupo2$ - não migrado para nenhuma modelagem, sendo apenas um atributo que foi usado na geração dos elementos de DB. Aleatoriamente, a cada registro foi atribuído um número entre 1 e 70% da quantidade de registros de D;
- tabela DA - aleatoriamente foram associados entre 1 e 19 elementos de A para cada elemento de D, gerando registros com as chaves d_pk e a_pk e mais o atributo da_int4, com um valor aleatório que poderia variar de 0 (zero) a 99;
- tabela DB
  - os registros de D com o mesmo $grupo1$ foram associados entre si gerando um registro na tabela com $d\_pk1 = d\_pk$ do primeiro registro de D, $d\_pk2 = d\_pk$ do segundo registro de D e $b\_pk = 1$;
  - os registros de D com o mesmo $grupo2$ foram associados entre si gerando um registro na tabela com $d\_pk1 = d\_pk$ do primeiro registro de D, $d\_pk2 = d\_pk$ do segundo registro de D e $b\_pk = 2$;

- tabela DE - para cada registro de D foram gerados 4 registros em DE:
  - o primeiro registro recebeu os seguintes dados:
    - d_fk1 - d_pk do registro de D;
    - e_fk2 - o valor 4, relacionando-o com o registro de $e\_pk = 4$ de E;
    - g_fk3 - foi escolhido aleatoriamente um registro dentro de G;
    - de_dt2 - foi preenchido com uma data entre 01/01/1928 e 31/12/2008;
    - de_en3 - aleatoriamente *che* ou *min* ou *nov* ou *cre*;
    - de_st3 - aleatoriamente um valor *string* representando uma hora;
  - o segundo registro recebeu os seguintes dados:
    - d_fk1 - d_pk do registro de D;
    - e_fk2 - o valor 1, relacionando-o com o registro de $e\_pk = 1$ de E;
    - g_fk3 - repetido o g_fk3 (G) do primeiro registro;
    - de_dt2 - a data do registro anterior ($e\_pk = 4$) menos o número de dias registrado em d_int1 (267, 237 ou 207);
    - de_en3 - repetido o de_en3 do registro anterior;
    - de_st3 - aleatoriamente um valor *string* representando uma hora;
  - o terceiro registro recebeu os seguintes dados:
    - d_fk1 - d_pk do registro de D;
    - e_fk2 - o valor 2, relacionando-o com o registro de $e\_pk = 2$ de E;
    - g_fk3 - repetido o g_fk3 (G) do primeiro registro;
    - de_dt2 - a data do registro anterior ($e\_pk = 1$) mais 89 dias;
    - de_en3 - repetido o de_en3 do registro anterior;
    - de_st3 - aleatoriamente um valor *string* representando uma hora;
  - o quarto registro recebeu os seguintes dados:
    - d_fk1 - d_pk do registro de D;
    - e_fk2 - o valor 3, relacionando-o com o registro de $e\_pk = 3$ de E;
    - g_fk3 - repetido o g_fk3 (G) do primeiro registro;
    - de_dt2 - a data do registro anterior ($e\_pk = 2$) mais 89 dias;
    - de_en3 - repetido o de_en3 do registro anterior;
    - de_st3 - aleatoriamente um valor *string* representando uma hora;

- tabela DC - para cada registro de DE foram gerados 3 registros em DC, ou seja, 12 por D, combinando a data de DE com a data de H e com base nisso gravando DC:
  - d_fk1 - d_pk do registro de D;
  - dc_int3 - repetição do valor de h_int3 correspondente;
  - dc_en4 - aleatoriamente + ou −;
  - c_fk2 - repetição do valor de c_fk1 correspondente em H;
  - c_fk3 - repetição do valor de c_fk2 correspondente em H.

Uma vez gerados os dados para as bases *PostgreSQL*, *Base100Mil* e *Base1Milhao*, o próximo passo foi gerar as oito bases Neo4j dos experimentos, isto é, as bases correspondentes aos métodos $M01$, $M03$, $M04$ e $M05$ para cada base Neo4j.

Para a geração das bases Neo4j, fez-se a opção de utilizar o processo com arquivos `CSV`. Este processo tem dois passos:

- exportação dos dados das bases *PostgreSQL* para arquivos `CSV` - para cada um dos métodos analisados nesta dissertação foram gerados *scripts* que gravam arquivos `CSV` com os dados dispostos em vértices e arestas conforme cada modelagem;
- importação dos Arquivos `CSV` para as bases Neo4j - foram geradas, inicialmente, seis diferentes bases Neo4j, uma para cada um dos métodos $M01$, $M03$, $M04$ e $M05$ de *Base100Mil* e *Base1Milhao*. Para efeito dos experimentos realizados, é importante destacar:
  - criados os vértices, foram criados índices do tipo *UNIQUE* para a propriedade considerada chave primária de cada um dos vértices;
  - também foram criados índices para as propriedades identificadoras dos vértices (a_st1, b_st1, c_st1, d_st1, e_st1, f_st1 e g_st1);
  - como já dito, os atributos do tipo `data` foram importados como campos *string* no formato $AAAA/MM/DD$.

A tabela 4.1 e a figura 4.1 trazem a área total ocupada por cada uma das bases após a importação e as tabelas 4.2 e 4.3 apresentam os totais de elementos que foram importados em cada base.

Fazendo uma proporção entre as bases $M03$ e $M04$, respectivamente àquelas com maior e menor número de elementos e maior e menor tamanho, tem-se:

Tabela 4.1: Experimentos - Bases X Métodos - Tamanho das Bases

Metódo	Área da Base (GB)	
	Base100Mil	Base1Milhao
M01	1,3	47,1
M03	2,1	73,7
M04	1,3	34,0
M05	1,1	35,5

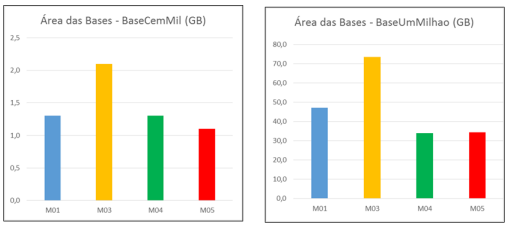


Figura 4.1: Experimentos - Área das Bases por Método - em GB

Tabela 4.2: Experimentos - Bases X Métodos - Total de Vértices por Base

Vértice	Base100Mil				Base1Milhao			
	M01/M02	M03	M04	M05	M01/M02	M03	M04	M05
A	6.186	6.186	6.186	6.186	6.186	6.186	6.186	6.186
B	2	2	-	-	2	2	-	-
C	36	36	-	36	36	36	-	36
D	100.000	100.000	100.000	100.000	1.000.000	1.000.000	1.000.000	1.000.000
DB	1.843.836	1.843.836	1.843.836	-	13.686.104	13.686.104	13.686.104	-
DC	1.200.000	1.200.000	1.200.000	1.200.000	12.000.000	12.000.000	12.000.000	12.000.000
DE	400.000	400.000	400.000	-	4.000.000	4.000.000	4.000.000	-
DO	-	-	-	100.000	-	-	-	1.000.000
DT	-	47.480	-	-	-	47.480	-	-
E	6	6	-	-	6	6	-	-
EN1	-	3	-	-	-	3	-	-
EN2	-	3	-	-	-	3	-	-
EN3	-	4	-	-	-	4	-	-
EN4	-	2	-	-	-	2	-	-
F	247	247	-	247	247	247	-	247
FL	-	237.982	-	-	-	237.982	-	-
G	136.829	136.829	-	136.829	136.829	136.829	-	136.829
H	554.788	554.788	554.788	554.788	554.788	554.788	554.788	554.788
INT1	-	3	-	-	-	3	-	-
INT2	-	1.800	-	-	-	1.800	-	-
INT3	-	12	-	-	-	12	-	-
ST5	-	100.000	-	-	-	1.000.000	-	-
ST6	-	1.440	-	-	-	1.440	-	-
	4.241.930	4.630.659	4.104.810	2.098.086	31.384.198	32.672.927	31.247.078	14.698.086

- no experimento com `Cem Mil Elementos em` D, o total de vértices e arestas de $M03$ é 236% maior que a de $M04$ e o tamanho da base 65% maior;
- no experimento com `Um Milhão de Elementos em` D, o total de vértices e arestas de $M03$ é 217% maior que a de $M04$ e o tamanho da base 117% maior.

Isso mostra que não há uma relação direta entre o total de vértices e arestas e o tamanho da base, mas também deve ser levado em conta as propriedades com valores duplicados por vértice.

Tabela 4.3: Experimentos - Bases X Métodos - Total de Arestas por Base

Aresta	Base100Mil				Base1Milhao			
	M01/M02	M03	M04	M05	M01/M02	M03	M04	M05
CgoEN1	-	36	-	-	-	36	-	-
DBgoB	1.843.836	1.843.836	1.843.836	-	13.686.104	13.686.104	13.686.104	-
DBgoD	1.843.836	1.843.836	1.843.836	-	13.686.104	13.686.104	13.686.104	-
DCgoC_1	1.200.000	1.200.000	-	1.200.000	12.000.000	12.000.000	-	12.000.000
DCgoC_2	1.200.000	1.200.000	-	1.200.000	12.000.000	12.000.000	-	12.000.000
DCgoEN4	-	1.200.000	-	-	-	12.000.000	-	-
DCgoINT2	-	1.200.000	-	-	-	12.000.000	-	-
DCgoINT3	-	1.200.000	-	-	-	12.000.000	-	-
DEgoE	400.000	400.000	-	-	4.000.000	4.000.000	-	-
DEgoEN3	-	400.000	-	-	-	4.000.000	-	-
DEgoG	400.000	400.000	-	-	4.000.000	4.000.000	-	-
DEgoST6	-	400.000	-	-	-	4.000.000	-	-
DgoA	950.313	950.313	950.313	950.313	9.496.864	9.496.864	9.496.864	9.496.864
DgoD	-	-	-	1.843.836	-	-	-	13.686.104
DgoDB	1.843.836	1.843.836	-	-	13.686.104	13.686.104	-	-
DgoDC	1.200.000	1.200.000	1.200.000	1.200.000	12.000.000	12.000.000	12.000.000	12.000.000
DgoDE	400.000	400.000	400.000	-	4.000.000	4.000.000	4.000.000	-
DgoDO	-	-	-	100.000	-	-	-	1.000.000
DgoEN2	-	100.000	-	-	-	1.000.000	-	-
DgoG	-	-	-	400.000	-	-	-	4.000.000
DgoINT1	-	100.000	-	-	-	1.000.000	-	-
DgoST5	-	100.000	-	-	-	1.000.000	-	-
GgoF	136.829	136.829	-	136.829	136.829	136.829	-	136.829
HgoC_1	554.788	554.788	-	554.788	554.788	554.788	-	554.788
HgoC_2	554.788	554.788	-	554.788	554.788	554.788	-	554.788
HgoINT2	-	554.788	-	554.788	-	554.788	-	554.788
	12.528.226	17.783.050	6.237.985	8.695.342	99.801.581	147.356.405	52.869.072	65.984.161

4.2 Consultas Efetuadas

Como os métodos acabaram por gerar bases de características bastante diferentes entre si, optou-se por realizar uma série de consultas que, em suas estruturas, explorassem essas diferenças, para ver qual método teria o melhor desempenho.

Os experimentos com as bases seguiram o seguinte roteiro, base por base:

- antes da execução para cada método, o servidor onde foi instalado o Neo4j foi reiniciado;
- foi executado o *script* apresentado na listagem 4.1, que percorre todos os vértices e arestas da base e retorna o total de vértices e arestas percorridos. Esse *script* foi executado para carregar o banco de memória (*warm*) conforme sugerido por Gordon (2015) e Gundy (2015) que em seus artigos colocaram que uma vez que o Neo4j tenha os elementos da base na memória, o desempenho dos acessos à base melhora, o que ficou evidenciado nos experimentos realizados. Os autores sugerem, ainda, que este *script* seja executado 3 vezes consecutivas a cada vez que o servidor do Neo4j seja reiniciado. Assim, antes do início das consultas com cada base, esse *script* foi executado três vezes;
- ainda a título de aquecimento, as 18 consultas foram executadas uma vez com

descarte do resultado;

- as consultas foram executadas via o *Neo4j Browser*, acessado via `Firefox` no endereço *http* : *//localhost* : 7474;
- as consultas foram executadas em sequência, da 01 à 18, oito execuções por consulta (a de aquecimento mais sete execuções), e o tempo das sete últimas execuções registrado para depois calcular a média. O tempo de cada consulta foi aquele exibido após o resultado da consulta na tela do *Neo4j Browser*, conforme mostra a figura 4.2;
- o resultado das consultas está em milissegundos (ms) e, em nenhuma das execuções, para todas as consultas, o resultado foi 0 (zero).

Listagem 4.1: Aquecendo a Base

```
START n=node(*)
OPTIONAL MATCH (n) -[r]-> ()
RETURN count(n) AS vertices, count(r) as arestas;
```

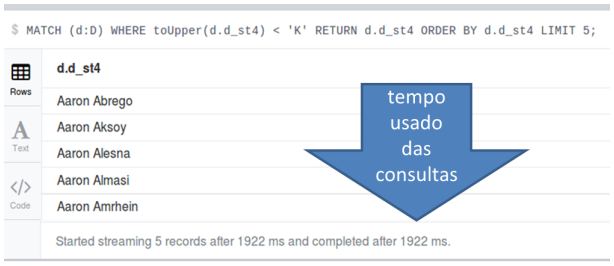


Figura 4.2: Experimentos - Tempo Considerado para as Consultas

As 18 diferentes consultas foram elaboradas com o objetivo de explorar as diferenças entre as diversas modelagens, explorando modelos com maior ou menor quantidade de vértices e arestas, vértices com maior ou menor quantidade de propriedades, maior ou menor número de arestas, etc. Nesse sentido as consultas buscaram responder a questões como:

- o desempenho de uma consulta que envolve um único vértice e retorna dados de uma única propriedade tem variação conforme o número de propriedades do vértice envolvido?
- há diferença de desempenho se as propriedades retornadas numa consulta estão num único vértice ou divididas em dois ou mais vértices?

- uma consulta tem melhor ou pior desempenho se a propriedade usada como filtro na seleção está no vértice ou na aresta?
- o caminho mínimo entre dois vértices muda conforme a modelagem?

Os experimentos foram executados em um equipamento AMD FX(tm)-8350 *Eight-Core Processor*, 8 GB de memória, sistema operacional Ubuntu 14.04.5 LTS e com o Neo4j *Community Edition*, versão 3.1.4 e Java versão 1.8.0_131, 64 *bits*. As configurações de *heap size* e *cache* do Java seguiram o *default* do Neo4j, ou seja, o tamanho foi calculado de acordo com a memória disponível no momento do início da execução do serviço Neo4j.

A tabela 4.4 traz a relação das consultas utilizadas nos experimentos e, em seguida, a tabela 4.5 traz essas consultas classificadas de acordo com a característica de cada uma.

4.3 Análise do Resultado das Consulta com as Bases dos Métodos M01/02, M03 e M04

Após a execução das dezoito consultas para os métodos $M01/M02$, $M03$ e $M04$, os dados foram tabulados, sendo, como já foi colocado, descartado o tempo da primeira execução de cada consulta, considerados como de aquecimento da consulta. A partir do tempo das sete últimas execuções de cada consulta foi calculado o tempo médio de cada consulta, seu desvio padrão e o intervalo de confiança da amostra.

No caso da consulta 13 para a $Base1Milhao$, o segundo tempo de execução foi, em todos os métodos, bastante discrepante em relação ao tempo de execução para as demais execuções. Então, para esta consulta, o segundo valor também foi descartado no cálculo da média, desvio padrão e intervalor de confiança.

Os resultados foram tabulados nas tabelas 4.6 e 4.7, uma para cada uma das bases, cem mil e um milhão de registros. Cada tabela contém as seguintes colunas:

- *Consulta* - número da consulta;
- $M01$ $M03$ $M04$- para cada umas das modelagens:
  - MED - tempo médio em milissegundos (ms) das 7 execuções das consultas nas bases dos métodos. Quando a média exibida na célula corresponder a menor média entre os métodos, o fundo da célula estará cinza escuro e os algarismos em negrito e itálico. Caso outras das médias estejam dentro do intervalo de

Tabela 4.4: Experimentos - Relação das Consultas

Consulta	Entidades Envolvidas	Objetivo
01	D	Retorna d_st4 de todos os D com d_st4 entre as letras 'A' e 'J'. Utiliza apenas D em todos os métodos.
02	D×DC×C	Lista elementos de D com maior ocorrência de um mesmo elemento de C. Testa desempenho com seleção por uma aresta e agrupamento.
03	D	Retorna d_st4, d_st5, d_int1 de todos os D com d_st4 entre as letras 'K' E 'Z' e com d_en2 = 3 em ordem decrescente de d_pk. d_st5, d_int1 e d_en2 podem estar em vértices diferentes.
04	H×C	Retorna elementos de H combinados com C em duas datas. Testa desempenho quanto a busca num único vértice ou mais de um vértices e arestas.
05	A×D	Retorna total de elementos de D para cada A. Grafo típico - Dois vértices e a aresta que os relaciona.
06	F com mais D	Retorna o total de elementos de D para um mesmo F. Testa desempenho com seleção e agrupamento variando o caminho percorrido (vértices e arestas).
07	A×DgoA×A	Retorna total de elementos de D com d_int4 = 1 para cada A. Dois vértices e um filtro na propriedade do vértice que os relaciona.
08	D×DB×B	Lista D e a lista dos D ligados a ele para d_pk < 10000 e b_pk = 2. Testa desempenho conforme seleção na aresta ou no vértice destino da aresta.
09	G com mais D	Retorna o total de elementos de D para um par (F,G). Testa desempenho com seleção e agrupamento variando o caminho percorrido (vértices e arestas) e o local das propriedades usadas como filtro.
10	D×D com b_pk 2	Retorna um grafo com os elementos de D e seus amigos. Testa o desempenho na seleção de nós com maior ou menor caminho de arestas e vértices.
11	Maior A×B em D	Pares (A, B) com maior ocorrência entre elementos de D. Desempenho da consulta envolvendo diferentes vértices e arestas.
12	D×DC×C×A	Lista elementos de D que tenham uma combinação específica de elementos de C. Testa desempenho com seleção por duas arestas.
13	A×D×DE×E×B	Pares (A e B) comuns a elementos de D com seu evento de e_pk=4 ocorrido no ano de 1985 Desempenho da consulta envolvendo diversos vértices e arestas e seleção por propriedades de vértices ou arestas.
14	A1×D×B×A2	Seleciona todos os elementos de D com determinada ocorrência de A que tenham amigos com uma outra ocorrência específica de A. Testa desempenho e seleção percorrendo duas vezes as mesmas arestas.
15	A×B×C×D×E×F×G (relação)	Lista elementos de D combinados com DC, B, D novamente e A. Testa desempenho percorrendo todo o grafo, exceto o vértice H. Retorna o resultado na forma de relação.
16	A×B×C×D×E×F×G (vértice)	Retorna um grafo com os nós D, DC, B, D novamente e A. Testa desempenho percorrendo todo o grafo, exceto o vértice H.
17	Caminho Mínimo entre D=1 e D=2	Calcula o caminho mínimo entre dois elementos de D. Testa o desempenho e o caminho retornado conforme o número de vértices e arestas envolvidos.
18	Todos os Caminhos Mínimos entre D=1 e D=2.	Calcula todos os caminhos mínimos entre dois elementos de D. Testa o desempenho e os caminhos retornados conforme o número de vértices e arestas envolvidos.

confiança, para mais ou para menos, então a célula estará com fundo cinza claro e os algarismos em negrito e itálico;

– DP - desvio padrão do tempo das 7 execuções;

– IC - intervalo de confiança da distribuição *t de student* para as 7 execuções, com 95% de confiança. Na consulta 6 para a $Base1Milhao$

A análise dos tempos médios expostos nas tabelas para os métodos $M01$, $M03$ e $M04$ permite afirmar que nenhum dos métodos se mostrou melhor ou pior que os demais, sendo cada solução mais ou menos adequada conforme o tipo de consulta efetuada.

O apêndice B traz em detalhes todas as consultas efetuadas sobre as bases, desde o

Tabela 4.5: Experimentos - Consultas por Tipo

Caracterísitica das Consultas	Consultas
Quantidade de elementos do vértice	01/02
Vértice único ou Mais de Um vértice	03/04
Vértice –> Aresta –> Vértice SEM Filtro	05/06
Vértice –> Aresta –> Vértice COM Filtro na Aresta	07
Vértice –> Aresta –> Vértice COM Filtro na Aresta OU Vértice	08/09/10
Quantidade de Arestas Envolvidas	11/12
Quantidade de Arestas Envolvidas COM Filtro na Aresta OU Vértice	13
Caminho Repetido	14
Percorrer todas as entidades	15/16
Caminho Mínimo com Mais ou Menos Arestas	17/18

Tabela 4.6: Experimentos - Base100Mil - 7 Execuções da Consulta (M01/02, M03 e M04) - tempo em ms

Cons	M01/M02			M03			M04		
	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
01	***396***	57,6	53,3	***413***	108,2	100,1	***386***	60,8	56,3
02	2.500	143,1	132,3	2.695	32,7	30,2	***2.179***	139,0	128,6
03	176	19,9	18,4	85	28,6	26,5	***141***	13,9	12,9
04	***15***	5,1	4,8	8.408	33,8	31,3	***9***	2,3	2,1
05	***2.477***	44,6	41,2	2.580	14,6	13,5	***2.454***	17,7	16,4
06	868	79,1	73,2	1.965	33,6	31,1	***352***	42,6	39,4
07	***1.681***	48,8	45,1	1.960	94,9	87,7	***1.750***	68,5	63,4
08	1.613	17,3	16,0	1.712	57,6	53,2	***1.290***	67,3	62,2
09	***494***	39,5	36,5	579	25,1	23,2	***564***	69,0	63,9
10	***56***	4,6	4,3	***49***	5,4	5,0	***52***	3,9	3,6
11	***21***	3,0	2,7	***20***	3,3	3,1	***15***	6,0	5,5
12	380	29,4	27,2	566	8,9	8,2	***254***	6,9	6,4
13	1.485	47,1	43,5	1.929	37,6	34,8	***1.282***	63,3	58,6
14	74	6,9	6,4	***67***	10,7	9,9	***59***	2,3	2,1
15	***38***	3,4	3,1	130	4,0	3,7	1.332	42,8	39,6
16	***39***	4,4	4,1	129	5,9	5,5	1.352	127,4	117,8
17	***15***	3,0	2,8	15	1,4	1,3	***11***	3,0	2,8
18	16	1,9	1,8	17	2,4	2,2	***13***	2,7	2,5

esquema, consulta a consulta, dos vértices e arestas envolvidos, até a análise dos resultados da consulta, o *script* em *Cypher* de todas as consultas e os tempos de cada execução das consultas, método por método e base a base.

Da análise do resultado, pode-se verificar que:

- a consulta 01, que envolveu apenas um tipo de vértice, D, tendo como variante o número de propriedades do vértice, mostrou um melhor desempenho para o método

Tabela 4.7: Experimentos - Base1Milhao - 7 Execuções da Consulta (M01/02, M03 e M04) - tempo em ms

Cons	M01/M02			M03			M04		
	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
01	4.434	136,2	126,0	***3.670***	148,6	137,4	4.236	146,2	135,2
02	***263.445***	34.504,3	31.911,1	468.496	14.869,6	13.752,1	469.688	31.194,8	28.850,4
03	1.473	36,9	34,1	***547***	9,4	8,7	1.378	40,4	37,4
04	15	4,0	3,7	6.147	34,9	32,3	***10***	1,8	1,6
05	26.017	20,4	18,9	27.172	21,2	19,6	***25.797***	97,6	90,3
06(*)	20.706	190,4	176,0	21.921	107,5	99,4	***2.763***	93,8	86,8
07	21.475	153,7	142,2	***16.628***	144,1	133,3	19.727	191,7	177,3
08	1.328	15,3	14,1	1.275	28,5	26,4	***1.094***	25,1	23,2
09	***440***	7,5	7,0	796	40,3	37,3	3.203	79,7	73,7
10	76	6,1	5,6	47	5,0	4,7	***46***	4,3	3,9
11	110	2,0	1,9	126	2,6	2,4	***100***	6,2	5,7
12	3.117	92,9	85,9	29.460	3.502,9	3.239,6	***2.145***	34,1	31,6
13	11.650	299,3	314,1	75.012	1.251,3	1.313,1	***10.944***	291,9	306,3
14	***454***	26,3	24,3	565	43,3	40,0	491	8,0	7,4
15	***340***	11,2	10,3	697	37,6	34,8	9.147	74,4	68,8
16	***334***	15,8	14,6	642	21,2	19,6	9.342	77,7	71,8
17	59	10,6	9,8	60	11,0	10,2	***42***	3,4	3,2
18	51	6,4	5,9	55	8,8	8,2	***48***	2,9	2,7

(*) foram descartados o tempo de execução das duas primeiras consultas de cada método

$M03$, onde D possui apenas 2 propriedades, ao contrário das demais onde D tem 5 propriedades;

- a consulta 04 mostrou que sintetizar dados num único vértice torna a operação mais rápida ($M04$) quando o intervalo pesquisado for pequeno, entretanto quando o intervalo for grande, como na consulta 02, o tempo diminui se o vértice filtrado for aquele com menor número de atributos;

- as consultas 05 e 07 reforçam a constatação que selecionar vértices com menor número de propriedades produz um melhor desempenho nas consultas;

- as consultas 06, 11, 13, 17 e 18 demonstraram que diminuir a travessia realmente diminui o tempo da consulta, então quanto menos vértices e arestas estiverem envolvidos nas consultas melhor o desempenho.

Disso tudo, pode-se entender que:

- compactar a base unificando entidades num único vértice se revelou uma boa estratégia quando a aplicação não tem muitos filtros com os dados unificados;

- consultas executam melhor quando os vértices envolvidos têm um menor número de atributos;

- quanto maior o caminho, maior o tempo se não houver filtragem;
- quando um atributo de um vértice for constantemente utilizado como filtro de consultas e não for um atributo que funcione como um identificador do elemento, nesse caso cabe sim criar um vértice para ele.

Com base na análise dos resultados dessas consultas, foi desenvolvido o método objeto desta dissertação, conforme exposto no capítulo 3.

Em seguida, uma nova base foi gerada conforme o novo método de modelagem e a mesma foi submetida às mesmas consultas aqui expostas. A análise dessas consultas vem colocado na próxima seção.

4.4 Experimentos com a Base pelo Modelo Proposto

Da mesma forma que na análise para os métodos $M01$, $M03$ e $M04$, finda a execução das consultas para o método $M05$, seus resultados foram tabulados nas tabelas 4.8 e 4.9, com as mesmas características das tabelas 4.6 e 4.7 acrescidas das colunas referentes ao método $M05$.

Tabela 4.8: Experimentos - Base100Mil - 7 Execuções da Consulta (M01/02, M03, M04 e M05) - tempo em ms

Cons	M01/M02			M03			M04			M05		
	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
01	396	57,6	53,3	413	108,2	100,1	386	60,8	56,3	***299***	39,1	36,1
02	2.500	143,1	132,3	2.695	32,7	30,2	***2.179***	139,0	128,6	***2.187***	132,2	122,2
03	176	19,9	18,4	85	28,6	26,5	141	13,9	12,9	***25***	8,0	7,4
04	***15***	5,1	4,8	8.408	33,8	31,3	***9***	2,3	2,1	1.535	88,8	82,1
05	2.477	44,6	41,2	2.580	14,6	13,5	2.454	17,7	16,4	***2.218***	18,7	17,3
06	868	79,1	73,2	1.965	33,6	31,1	***352***	42,6	39,4	780	64,4	59,5
07	1.681	48,8	45,1	1.960	94,9	87,7	1.750	68,5	63,4	***1.582***	36,3	33,6
08	1.613	17,3	16,0	1.712	57,6	53,2	1.290	67,3	62,2	***777***	12,2	11,3
09	494	39,5	36,5	579	25,1	23,2	564	69,0	63,9	***314***	6,9	6,4
10	56	4,6	4,3	49	5,4	5,0	52	3,9	3,6	***29***	5,2	4,8
11	21	3,0	2,7	20	3,3	3,1	***15***	6,0	5,5	***10***	2,0	1,8
12	380	29,4	27,2	566	8,9	8,2	***254***	6,9	6,4	360	41,6	38,5
13	1.485	47,1	43,5	1.929	37,6	34,8	1.282	63,3	58,6	***1.157***	47,7	44,1
14	74	6,9	6,4	67	10,7	9,9	59	2,3	2,1	***1***	0,4	0,3
15	***38***	3,4	3,1	130	4,0	3,7	1.332	42,8	39,6	45	3,6	3,3
16	***39***	4,4	4,1	129	5,9	5,5	1.352	127,4	117,8	***42***	2,9	2,6
17	15	3,0	2,8	15	1,4	1,3	11	3,0	2,8	***7***	2,9	2,7
18	16	1,9	1,8	17	2,4	2,2	13	2,7	2,5	***7***	2,5	2,4

Comparando agora os resultados das consultas com o modelo proposto ($M05$) tem-se que:

Tabela 4.9: Experimentos - Base1Milhao - 7 Execuções da Consulta (M01/02, M03, M04 e M05) - tempo em ms

Cons	M01/M02			M03			M04			M05		
	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
01	4.434	136,2	126,0	***3.670***	148,6	137,4	4.236	146,2	135,2	***3.502***	77,1	71,3
02	263.445	34.504,3	31.911,1	468.496	14.869,6	13.752,1	469.688	31.194,8	28.850,4	***204.804***	8.265,0	7.643,8
03	1.473	36,9	34,1	547	9,4	8,7	1.378	40,4	37,4	***165***	22,0	20,3
04	***15***	4,0	3,7	6.147	34,9	32,3	***10***	1,8	1,6	1.348	33,2	30,7
05	26.017	20,4	18,9	27.172	21,2	19,6	25.797	97,6	90,3	***24.068***	97,2	89,9
06(*)	20.706	190,4	176,0	21.921	107,5	99,4	***2.763***	93,8	86,8	6.417	37,9	35,0
07	21.475	153,7	142,2	***16.628***	144,1	133,3	19.727	191,7	177,3	17.759	135,9	125,7
08	1.328	15,3	14,1	1.275	28,5	26,4	1.094	25,1	23,2	***761***	17,8	16,5
09	440	7,5	7,0	796	40,3	37,3	3.203	79,7	73,7	***420***	7,3	6,7
10	76	6,1	5,6	47	5,0	4,7	46	4,3	3,9	***31***	3,4	3,1
11	110	2,0	1,9	126	2,6	2,4	100	6,2	5,7	***62***	4,9	4,5
12	3.117	92,9	85,9	29.460	3.502,9	3.239,6	***2.145***	34,1	31,6	3.402	53,4	49,4
13	11.650	299,3	314,1	75.012	1.251,3	1.313,1	10.944	291,9	306,3	***10.362***	218,5	229,3
14	454	26,3	24,3	565	43,3	40,0	491	8,0	7,4	***401***	2,0	1,8
15	340	11,2	10,3	697	37,6	34,8	9.147	74,4	68,8	***247***	11,7	10,8
16	334	15,8	14,6	642	21,2	19,6	9.342	77,7	71,8	***225***	31,3	29,0
17	59	10,6	9,8	60	11,0	10,2	42	3,4	3,2	***14***	1,5	1,4
18	51	6,4	5,9	55	8,8	8,2	48	2,9	2,7	***28***	4,2	3,9

(*) foram descartados o tempo de execução das duas primeiras consultas de cada método

- o tamanho da base referente ao método $M05$ ficou dentro do esperado, dado a quantidade de vértices e arestas do modelo, ou seja, maior que a base $M04$ e menor que as demais;
- o método $M05$ trouxe melhora no desempenho da maioria das consultas efetuadas sobre a base em comparação com as demais bases;
- mesmo nas consultas onde o $M05$ não foi aquele de melhor desempenho, em apenas uma delas teve o pior desempenho, ou seja, os critérios propostos no novo método trouxeram melhorias em relação aos demais;
- na consulta 04 já era esperado que o método $M04$ tivesse um melhor resultado, pois os dados selecionados na consulta estão todos num único vértice, enquanto no método $M05$ os dados são obtidos acessando dois vértices e duas arestas;
- nas consultas de 15 à 18, que em suas estruturas exploraram a utilização de filtros nas arestas, o método proposto teve um desempenho bem superior aos demais, o que comprova a validade de se incorporar atributos de vértices a arestas.

Os experimentos realizados mostram que o método proposto nesta dissertação é válido e atinge o objetivo de unir as melhores características dos quatro métodos antes analisados.

Um segundo experimento foi realizado com outra base, esta de Filmes, Pessoas que atuaram neles e suas Premiações, a partir de dados reais obtidos na *Internet*. Este experimento está descrito no Apêndice C. Neste experimento pode ser verificado que o padrão

de resultados se repetiu, isto é, onde se reduziu caminhos com a incorporação de entidades por relacionamentos ou outras entidades, houve melhora no desempenho em relação as demais modelagens.

5 Conclusão

A análise de quatro métodos de modelagem de dados para `BDGs` mostrou que moldar os dados é muito mais do que transformar entidades e relacionamentos em vértices e arestas. Os diferentes autores apresentaram métodos cuja base resultante, no tocante a vértices e arestas, tiveram resultados bem diferentes entre si. Para descobrir qual era de fato o melhor método, consultas que exploravam, em suas estruturas, as diferenças das bases geradas por cada método, foram executadas e seus resultados comparados entre si. Com relação ao desempenho destas operações, os métodos tiveram resultados ora melhores, ora piores, conforme os elementos envolvidos em cada consulta, sem se sobressair um método como o melhor dentre os quatro.

Este trabalho propôs um novo método de modelagem que foi construído em cima da análise das melhores características dos quatro métodos anteriores no tocante ao desempenho na recuperação de informações do banco de dados. A grande inovação do método proposto está na conversão de relacionamentos de grau maior que 2, onde todos os outros métodos, à semelhança do modelo relacional, criam um vértice intermediário com arestas entre as entidades envolvidas e na proposição uma ou mais entidades podem ser removidas do modelo e seus atributos incorporados como atributos do relacionamento, sem a criação de um vértice intermediário, diminuindo o caminho a ser percorrido por ocasião das consultas ao grafo.

Outro diferencial está no fato do método proposto exigir mais do que a aplicação de fórmulas matemáticas na conversão, pois o projetista do banco deverá conhecer os dados do negócio e com base neles influir, com as informações prestadas, no processo de modelagem.

Os mesmos testes aplicados às bases dos métodos anteriores foram, então, aplicados a base do novo método e, embora, ele tenha tido um melhor desempenho na maioria das consultas, em algumas isso não ocorreu. O fato de não ter melhor desempenho em todas as consultas não é uma falha do método, mas sim uma decorrência de escolhas

na modelagem. Unificar vértices com a consequente eliminação das arestas entre eles traz um melhor desempenho em consultas que retornam propriedades que antes estariam distribuídas em diferentes vértices, mas piora o desempenho se o retorno da consulta pertence a propriedades de um só dos vértices.

Assim, respondendo às perguntas norteadoras dos experimentos, apresentadas no capítulo 4, temos que:

- o número de propriedades dos vértices envolvidos numa consulta será relevante para o desempenho da mesma, mas essa melhora será mais perceptível a medida que aumentar a quantidade de elementos desses vértices;
- o número de vértices envolvidos numa consulta é relevante para o desempenho da consulta, entretanto não é o único fator a ser considerado, uma vez que as consultas demonstraram que a quantidade de propriedades do vértice e sua quantidade de elementos também influem no desempenho;
- a quantidade de vértices e arestas envolvidos numa consulta, como a de caminho mínimo, muda conforme a modelagem e, neste caso, quanto menor a quantidade envolvida e, portanto, o caminho percorrido, melhor será o desempenho da consulta.

Apesar da resposta positiva a estas três questões, a constatação de que em algumas consultas as modelagens dos métodos $M01/M02$, $M03$ ou $M04$ tiveram desempenho superior à modelagem proposta leva a conclusão, para essa base teste, de que nenhum dos métodos será sempre mais ou menos adequado que outros métodos e sim, que um ou outro método deverá ser escolhido conforme as características da aplicação que será atendida pela base que está sendo modelada.

5.1 Trabalhos Futuros

O método proposto foi implementado e comparado com os demais métodos analisados nesta dissertação em três bases diferentes, a $BaseTeste$ com cem mil e um milhão de elementos no vértice D e a $BaseFilmes$ apresentada no apêndice C. Essa análise demonstrou a validade da Modelagem Participativa, entretanto novos experimentos devem ser feitos considerando:

- outros Sistemas Gerenciadores de Bancos de Dados voltados a Grafos, como o ArangoDB e o OrientDB, cujo uso vêm crescendo entre os desenvolvedores;

- outros equipamentos, com arquiteturas diferentes, com maior memória, outros processadores, etc;
- outras configurações do Java, variando o *Page Cache Sizing*, o *Heap Sizing* e o *Number of Open Files*;
- outras bases, aplicando a modelagem a bases reais, com dados não fictícios;
- outras modelagens da $BaseTeste$ conforme a Modelagem Participativa, onde diferentes respostas do projetista aos parâmetros do Grau de Incorporação das Entidades geram bases com estrutura diferentes, determinando o impacto das respostas no desempenho de acesso ao banco.

Outro trabalho futuro é o desenvolvimento de uma aplicação para a migração de uma base relacional para um base voltada a grafos, conforme os critérios do método $M05$.

A constatação de que nenhum dos métodos é mais eficiente em todos os cenários dessa base teste que os demais traz a luz a ideia de um outro método de modelagem que pode ser estudado e depois, se viável, proposto. Neste novo método deve-se fazer um estudo dos vértices e relacionamentos para identificar um particionamento do grafo em diferentes grupos, que, conforme suas características, poderão ser ora migrados de acordo com o método $M01$, outro grupo pelo método $M03$ ou $M04$ ou $M05$.

Referências

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

BATRA, S.; TYAGI, C. Comparative analysis of relational and graph databases. *International Journal of Soft Computing and Engineering (IJSCE)*, Citeseer, v. 2, n. 2, p. 509–512, 2012.

BORDOLOI, S.; KALITA, B. Designing Graph Database Models from existing relational databases. *International Journal of Computer Applications*, v. 74, n. 1, 2013.

BORDOLOI, S.; KALITA, B. ER Model to an Abstract Mathematical Model for Database Schema using Reference Graph. *International Journal of Engineering Research And Development, e-ISSN*, p. 51–60, 2013.

BREWER, E. A. Towards robust distributed systems. In: *PODC*. [S.l.: s.n.], 2000. v. 7.

CHEN, P. P.-S. The entity-relationship model?toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, ACM, v. 1, n. 1, p. 9–36, 1976.

CODD, E. F. A relational model of data for large shared data banks. *Communications of the ACM*, v. 13, n. 6, p. 377–387, 1970.

DB-ENGINES. *DBMS popularity broken down by database model*. 2017. Disponível em: <http://db-engines.com/en/rankingcategories>.

ERVEN, G. C. G. V. *MDG-NoSQL: Modelo de Dados para Bancos NoSQL Baseados em Grafos*. Dissertação (Mestrado) — Universidade de Brasília, 2015.

FONG, J.; WONG, H. K.; CHENG, Z. Converting relational database into xml documents with dom. *Information and Software Technology*, Elsevier, v. 45, n. 6, p. 335–355, 2003.

FOWLER, M.; SADALAGE, P. J. NoSQL Essencial: Um Guia Conciso Para O Mundo Emergente Da Persistência Poliglota. *Novatec, 1a edição*, v. 1, n. 1.1, 2013.

GORDON, D. *Warm the cache to improve performance from cold start*. 2015. Disponível em: <https://neo4j.com/developer/kb/warm-the-cache-to-improve-performance-from-cold-start/>.

GUNDY, K. V. *Hot Ca$h: A Note on Performance Testing Neo4j*. 2015. Disponível em: <http://kvangundy.com/wp/hot-cache-testing-neo4j>.

HEUSER, C. A. *Projeto de banco de dados: Volume 4 da Série Livros didáticos informática UFRGS*. [S.l.]: Bookman Editora, 2009.

HUNGER, M. *RDBMS to Graph*. 2015. Disponível em: <https://neo4j.com/blog/webinar-follow-relational-graph/>.

JAISWAL, G.; AGRAWAL, A. P. Comparative analysis of relational and graph databases. *IOSR Journal of Engineering (IOSRJEN)*, 2013.

KAGGLE. *Oscar Nominations from 1927 to 2015*. 2017. Disponível em: <https://dl.dropboxusercontent.com/u/14493611/blog/data/oscars-1926-2015.csv>.

KARP, P. D. A strategy for database interoperation. *Journal of Computational Biology*, v. 2, n. 4, p. 573–586, 1995.

LAKE, P.; CROWTHER, P. *Concise guide to databases*. [S.l.]: Springer, 2013.

LAMPOLTSHAMMER, T. J.; WIEGAND, S. Improving the computational performance of ontology-based classification using graph databases. *Remote Sensing*, Multidisciplinary Digital Publishing Institute, v. 7, n. 7, p. 9473–9491, 2015.

LYSENKO, A. et al. Representing and querying disease networks using graph databases. *BioData mining*, BioMed Central, v. 9, n. 1, p. 23, 2016.

MARKOWITZ, V. M.; MAKOWSKY, J. A. Identifying extended entity-relationship object structures in relational schemas. *IEEE Transactions on Software Engineering*, IEEE, v. 16, n. 8, p. 777–790, 1990.

MONIRUZZAMAN, A. Newsql: towards next-generation scalable rdbms for online transaction processing (oltp) for big data management. *arXiv preprint arXiv:1411.7343*, 2014.

NEO4J, T. *Neo4j: The World Leading Graph Database*. 2017. Disponível em: <https://neo4j.com/product/>.

NEUBAUER, P. *Neo4J JUG Karlsruhe*. 2013. Disponível em: <https://blog.synyx.de/2013/09/neo4j-jug-karslruhe/>.

NOSQL.ORG. *NoSQL - Your Ultimate Guide to the Non-Relational Universe!* 2017. Disponível em: <http://nosql-database.org/>.

PARK, Y. et al. Graph databases for large-scale healthcare systems: A framework for efficient data management and data services. In: *Data Engineering Workshops (ICDEW), 2014 IEEE 30th International Conference on*. [S.l.]: IEEE, 2014. p. 12–19.

PRITCHETT, D. Base: An acid alternative. *Queue*, ACM, v. 6, n. 3, p. 48–55, 2008.

QUIET AFFILIATE MARKETING. *Free First-name and Last-name Databases (CSV and SQL)*. 2009. Disponível em: <http://www.quietaffiliate.com/free-first-name-and-last-name-databases-csv-and-sql/>.

SEQUEDA, J. F.; ARENAS, M.; MIRANKER, D. P. On directly mapping relational databases to rdf and owl. In: ACM. *Proceedings of the 21st international conference on World Wide Web*. [S.l.], 2012. p. 649–658.

SIRIWARADHANA, S. *From the entity-relationship to the property-graph model*. 2014. Disponível em: <http://lambdazen.blogspot.com.br/2014/01/from-entity-relationship-to-property.html>.

TEMPLE, O. *popculture/IMDB 5000 Movie Dataset*. 2016. Disponível em: <https://data.world/popculture/imdb-5000-movie-dataset>.

TEOREY, T. J.; YANG, D.; FRY, J. P. A logical design methodology for relational databases using the extended entity-relationship model. *ACM Computing Surveys (CSUR)*, ACM, v. 18, n. 2, p. 197–222, 1986.

VALE, V. *Introdução ao NoSQL*. 2015. Disponível em: <http://www.viniciusvale.com/nosql/>.

VIRGILIO, R. D.; MACCIONI, A.; TORLONE, R. Model-driven design of graph databases. In: *Conceptual Modeling*. [S.l.]: Springer, 2014. p. 172–185.

WARDANI, D. W.; KIING, J. Semantic mapping relational to graph model. In: IEEE. *Computer, Control, Informatics and Its Applications (IC3INA), 2014 International Conference on*. [S.l.], 2014. p. 160–165.

APÊNDICE A – Base Teste - Importação das Bases

Como já foi dito, as bases utilizadas para os experimentos foram populadas a partir da importação de arquivos CSV. Estes arquivos foram gerados a partir de consultas SQL nas bases Postgres e posteriormente foram importados manualmente através da execução de comandos via *browser* usando o `Firefox`, acessando o endereço $http://localhost:7474$.

As figuras A.1 e A.2 trazem exemplos dos comandos para importação de vértices e arestas gravados em arquivos CSV.

```
// Create Vértice A - atributos
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///a.csv" AS row
CREATE (:A { a_pk: toInt(row.a_pk),
             a_st1: row.a_st1
           });
```

Figura A.1: Apêndice A - Exemplo de Importação de Vértices em CSV

```
// Create Aresta DgoA
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///d_go_a.csv" AS row
MATCH (dFrom:D {d_pk: toInt(row.d_pk_from)})
MATCH (aTo:A {a_pk: toInt(row.a_pk_to)})
MERGE (dFrom)-[:DgoA{da_int4: toInt(row.da_int4)}]->(aTo);
```

Figura A.2: Apêndice A - Exemplo de Importação de Arestas em CSV

O tempo de importação foi alto, principalmente nas bases com um milhão de registros, decorrente do equipamento, com pouca memória, mas serviu ao propósito de comparar o tempo para cada método. Como esperado, o menor tempo ocorreu para as bases dos métodos $M04$ e $M05$, ficando o tempo para o método $M01$ na faixa intermediária e um tempo bem maior para o método $M03$. Isso reflete exatamente a quantidade de diferentes vértices e arestas em cada modelagem.

A tabela A.1 traz o tempo total, em horas, minutos e segundos decorridos na importação de cada uma das bases e métodos.

Tabela A.1: Tempo (HH:MM:SS) de importação das bases

Método	Base100Mil	Base1Milhao
M01/M02	00:29:27	03:18:51
M03	00:30:53	04:12:17
M04	00:11:52	02:20:18
M05	00:16:37	02:10:39

As tabelas A.2 e A.3 trazem o detalhamento da importação de cada base por elemento importado, vértice ou aresta. Para cada vértice ou aresta são colocados o total de elementos importados e o tempo, em milissegundos, da importação. Ao final é totalizado o total de elementos importados, o tempo total em milissegundos e este tempo convertido para horas, minutos e segundos.

Tabela A.2: Apêndice B - BaseCemMil - Total de Elementos e Tempo de Importação

Vértice	Base100Mil							
	M01/M02		M03		M04		M05	
	Registros	Tempo(ms)	Registros	Tempo(ms)	Registros	Tempo(ms)	Registros	Tempo(ms)
A	6.186	1.201	6.186	246	6.186	343	6.186	327
B	2	140	2	109	-	-	-	-
C	36	125	36	110	-	-	36	125
D	100.000	3.744	100.000	1.716	100.000	1.513	100.000	1.701
DB	1.843.836	24.585	1.843.836	23.198	1.843.836	20.764	-	-
DC	1.200.000	22.620	1.200.000	15.272	1.200.000	14.742	1.200.000	22.900
DE	400.000	8.361	400.000	5.226	400.000	18.174	-	-
DO	-	-	-	-	-	-	100.000	2.325
DT	-	-	47.480	968	-	-	-	-
E	6	125	6	125	-	-	-	-
EN1	-	-	3	125	-	-	-	-
EN2	-	-	3	125	-	-	-	-
EN3	-	-	4	125	-	-	-	-
EN4	-	-	2	125	-	-	-	-
F	247	140	247	125	-	-	247	140
FL	-	-	237.982	3.354	-	-	-	-
G	136.829	3.385	136.829	2.152	-	-	136.829	3.182
H	554.788	9.937	554.788	7.083	554.788	1.906	554.788	10.172
INT1	-	-	3	125	-	-	-	-
INT2	-	-	1.800	187	-	-	-	-
INT3	-	-	12	124	-	-	-	-
ST5	-	-	100.000	1.982	-	-	-	-
ST6	-	-	1.440	156	-	-	-	-
CgoEN1	-		36	124	-	-	-	-
DBgoB	1.843.836	176.557	1.843.836	187.918	-	-	-	-
DBgoD	1.843.836	212.836	1.843.836	222.833	1.843.836	192.161	-	-
DCgoC_1	1.200.000	111.948	1.200.000	113.613	-	-	1.200.000	112.868
DCgoC_2	1.200.000	113.446	1.200.000	112.168	-	-	1.200.000	112.945
DCgoEN4	-	-	1.200.000	111.182	-	-	-	-
DCgoINT2	-	-	1.200.000	125.898	-	-	-	-
DCgoINT3	-	-	1.200.000	123.106	-	-	-	-
DEgoE	400.000	37.336	400.000	37.374	-	-	-	-
DEgoEN3	-	-	400.000	37.966	-	-	-	-
DEgoG	400.000	43.501	400.000	42.221	-	-	-	-
DEgoST6	-	-	400.000	41.683	-	-	-	-
DgoA	950.313	551.922	950.313	101.992	950.313	104.052	950.313	280.271
DgoD	-	-	-		-		1.843.836	161.289
DgoDB	1.843.836	164.408	1.843.836	173.166	1.843.836	193.644	-	
DgoDC	1.200.000	126.026	1.200.000	118.667	1.200.000	123.458	1.200.000	127.624
DgoDE	400.000	42.267	400.000	39.446	400.000	41.340	-	
DgoDO	-	-	-		-	-	100.000	11.170
DgoEN2	-	-	100.000	8.626	-	-	-	
DgoG	-	-	-		-	-	400.000	36.442
DgoINT1	-	-	100.000	9.111	-	-	-	-
DgoST5	-	-	100.000	9.937	-	-	-	-
GgoF	136.829	13.969	136.829	13.355	-	-	136.829	13.340
HgoC_1	554.788	48.476	554.788	51.480	-	-	554.788	49.871
HgoC_2	554.788	50.325	554.788	51.620	-	-	554.788	50.532
HgoINT2	-	-	554.788	57.160	-	-	-	-
Total	16.770.156	1.767.380	22.413.709	1.853.404	10.342.795	712.097	10.238.640	997.224
Tempo		00:29:27		00:30:53		00:11:52		00:16:37

Tabela A.3: Apêndice B - Base1Milhao - Total de Elementos e Tempo de Importação

Vértice	Base1Milhão							
	M01/M02		M03		M04		M05	
	Registros	Tempo(ms)	Registros	Tempo(ms)	Registros	Tempo(ms)	Registros	Tempo(ms)
A	6.186	997	6.186	264	6.186	278	6.186	298
B	2	141	2	125	-	-	-	-
C	36	140	36	125	-	-	36	125
D	1.000.000	36.270	1.000.000	19.784	1.000.000	35.247	1.000.000	20564
DB	13.686.104	287.699	13.686.104	275.486	13.686.104	264.478	-	-
DC	12.000.000	330.785	12.000.000	247.856	12.000.000	284.756	12.000.000	254123
DE	4.000.000	171.716	4.000.000	108.451	4.000.000	278.451	-	
DO	-	-	-		-	-	1.000.000	25896
DT	-	-	47.480	987	-	-	-	-
E	6	140	6	125	-	-	-	-
EN1	-	-	3	125	-	-	-	-
EN2	-	-	3	125	-	-	-	-
EN3	-	-	4	125	-	-	-	-
EN4	-	-	2	125	-	-	-	-
F	247	171	247	125	-	-	247	-
FL	-	-	237.982	3.547	-	-	-	-
G	136.829	5.522	136.829	2.145	-	-	136.829	2987
H	554.788	15.117	554.788	15.125	554.788	14.562	554.788	12471
INT1	-	-	3	125	-	-	-	-
INT2	-	-	1.800	190	-	-	-	-
INT3	-	-	12	124	-	-	-	-
ST5	-	-	1.000.000	35.471	-	-	-	-
ST6	-	-	1.440	154	-	-	-	-
CgoEN1	-	-	36	125	-	-	-	-
DBgoB	13.686.104	955.516	13.686.104	972.345	13.686.104	965.432	-	-
DBgoD	13.686.104	1.089.410	13.686.104	1.101.435	13.686.104	976.543	-	-
DCgoC_1	12.000.000	949.532	12.000.000	967.890	-	-	12.000.000	958.678
DCgoC_2	12.000.000	845.460	12.000.000	839.123	-	-	12.000.000	111.645
DCgoEN4	-	-	12.000.000	834.562	-	-	-	-
DCgoINT2	-	-	12.000.000	857.654	-	-	-	-
DCgoINT3	-	-	12.000.000	823.453	-	-	-	-
DEgoE	4.000.000	341.152	4.000.000	340.976	-	-	-	-
DEgoEN3	-	-	4.000.000	341.256	-	-	-	-
DEgoG	4.000.000	335.010	4.000.000	339.863	-	-	-	-
DEgoST6	-	-	4.000.000	340.786	-	-	-	-
DgoA	9.496.864	4.145.231	9.496.864	4.150.987	9.496.864	4.231.453	9.496.864	4.356.112
DgoD	-	-	-	-	-	-	13.686.104	897.670
DgoDB	13.686.104	939.220	13.686.104	943.234	-	-	-	
DgoDC	12.000.000	960.260	12.000.000	958.789	12.000.000	961.342	12.000.000	985.461
DgoDE	4.000.000	416.127	4.000.000	400.234	4.000.000	405.678	-	-
DgoDO	-	-	-	-	-	-	1.000.000	20.652
DgoEN2	-	-	1.000.000	20.345	-	-	-	-
DgoG	-	-	-	-	-	-	4.000.000	54.389
DgoINT1	-	-	1.000.000	19.984	-	-	-	-
DgoST5	-	-	1.000.000	20.347	-	-	-	-
GgoF	136.829	13.547	136.829	13.457	-	-	136.829	12.765
HgoC_1	554.788	41.476	554.788	41.238	-	-	554.788	42.301
HgoC_2	554.788	50.523	554.788	49.876	-	-	554.788	41.678
HgoINT2	-	-	554.788	48.793	-	-	554.788	42.005
Total	131.185.779	11.931.162	180.029.332	15.137.461	84.116.150	8.418.220	80.682.247	7.839.820
Tempo		03:18:51		04:12:17		02:20:18		02:10:39

APÊNDICE B – Base Teste - Consultas Efetuadas

Conforme a discussão no capítulo 4 todas as consultas foram efetuadas via *browser* usando o `Firefox`, acessando o endereço *http* : *//localhost* : 7474. A figura B.1 traz a tela padrão do aplicativo, *Neo4j Browser*[1], acessado nesta página.

A esquerda são exibidos os elementos da base: vértices (*nodes*), arestas (*relationships*) e as propriedades dos vértices e arestas (*property keys*). Na parte central superior fica a caixa de entrada do comando a ser executado.

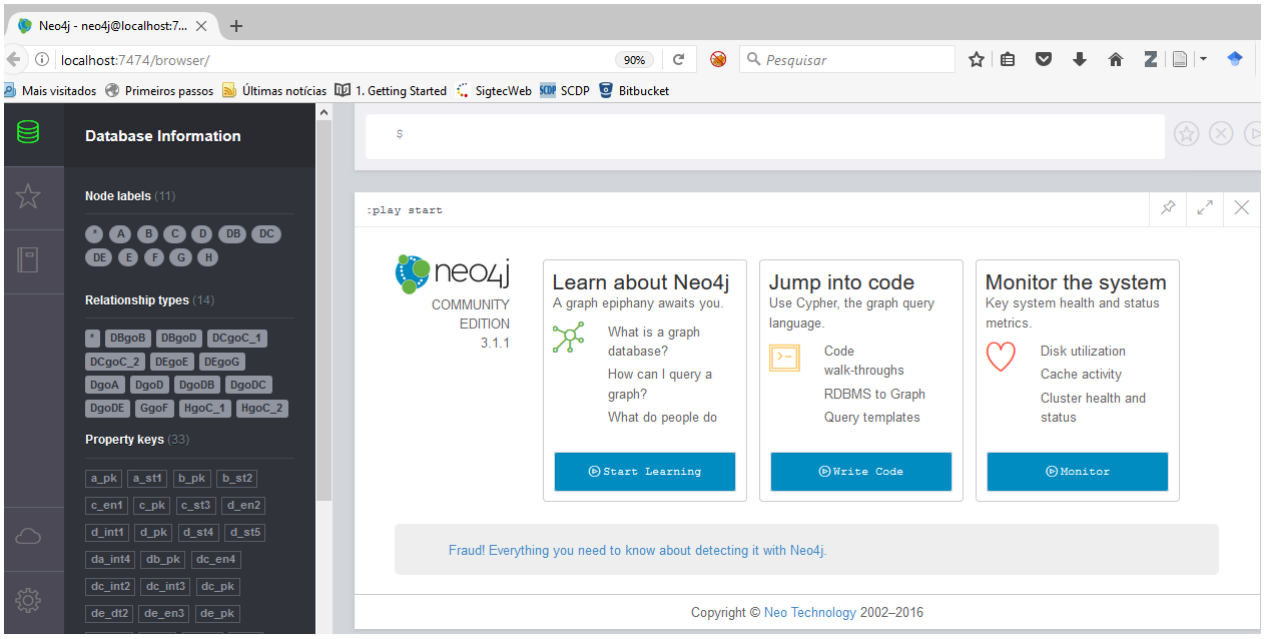


Figura B.1: Tela Inicial do Neo4j:Browser

As figuras B.2 e B.3 trazem exemplos do resultado de duas diferentes consultas, a primeira com uma saída em forma de relação e a segunda em forma de grafo.

O *Neo4j Browser* foi escolhido para os experimentos por permitir a execução das consultas independente da utilização de alguma linguagem de programação e por já apre-

[1]https://neo4j.com/developer/guide-neo4j-browser/

```
MATCH (d:D)
WHERE toUpper(d.d_st4) > 'K' AND d.d_en2 = 0 AND d.d_int1 <> 267
RETURN d.d_st4, d.d_st5, d.d_int1
ORDER BY d.d_pk DESC;
```

$ MATCH (d:D) WHERE toUpper(d.d_st4) > 'K' AND d.d_en2 = 0 AND d.d_int1 <> 267 RETURN d.d_st4,...

Rows | Text | Code

d.d_st4	d.d_st5	d.d_int1
Neal Bieschke	nbieschke99689@ficticio.com	207
Val Petticrew	vpetticrew98874@ficticio.com	237
Rachell Gederman	rgederman98524@ficticio.com	207
Micheal Janrhett	mjanrhett98017@ficticio.com	207
Laurel Infantolino	linfantolino97913@ficticio.com	207
Mardell Mee	mmee97614@ficticio.com	207
Marivel Maslakowski	mmaslakowski97490@ficticio.com	237
Randell Remo	rremo97268@ficticio.com	207
Nigel Benabides	nbenabides97076@ficticio.com	237
Michal Elgart	melgart96634@ficticio.com	207
Virgil Broadaway	vbroadaway95695@ficticio.com	237
Rachael Voisine	rvoisine94985@ficticio.com	237
Kendal Cottone	kcottone94723@ficticio.com	207
Michel Pennell	mpennell94469@ficticio.com	237
Shantel Marander	smarander94324@ficticio.com	207

Figura B.2: Consulta no Neo4j:Browser - Saída em formato de relação

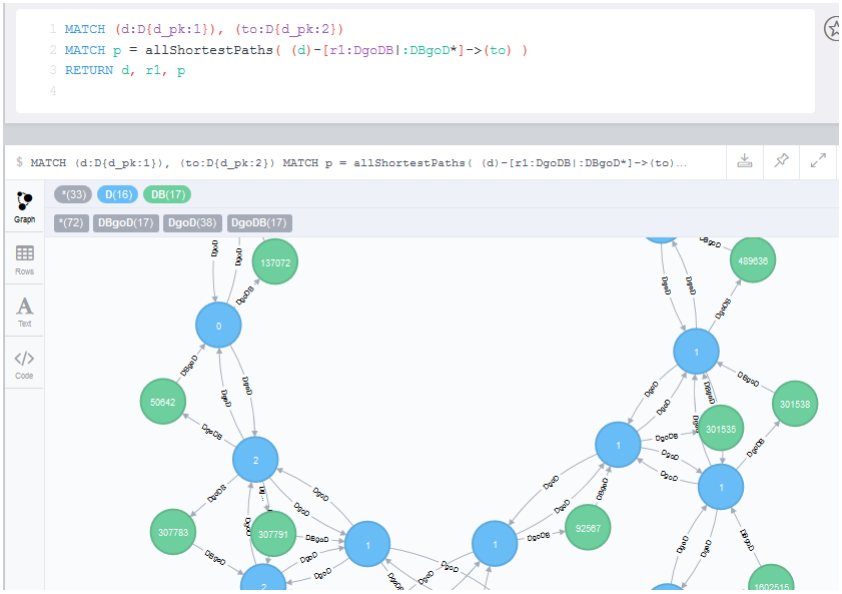


Figura B.3: Consulta no Neo4j:Browser - Saída em formato de Grafo

sentar, ao final de cada operação, a quantidade de registros retornados e seu tempo efetivo de execução, tempo este que foi o utilizado como tempo dos experimentos realizados.

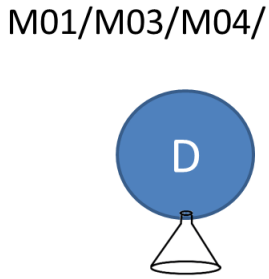
As seções seguintes detalham cada uma das consultas efetuadas na base, com suas estruturas e tempos de execução. Em seguida são apresentados o tempo de execução de cada base e respectivo método. Ao final são listados o *script* de cada uma das consultas efetuadas.

B.1 Detalhando as Consultas

Nesta seção são apresentadas as estruturas de cada uma das consultas mencionadoas no capítulo 4.

B.1.1 Consulta 01 - Listar a propriedade d_st4 de todos os vértices D com a primeira letra do nome entre as letras A e J

Esta consulta envolve apenas o vértice D para todos os métodos, sem envolver arestas. Seu objetivo é verificar se há diferença significativa ao buscar os elementos de um vértice, variando apenas a quantidade de propriedades por vértice.



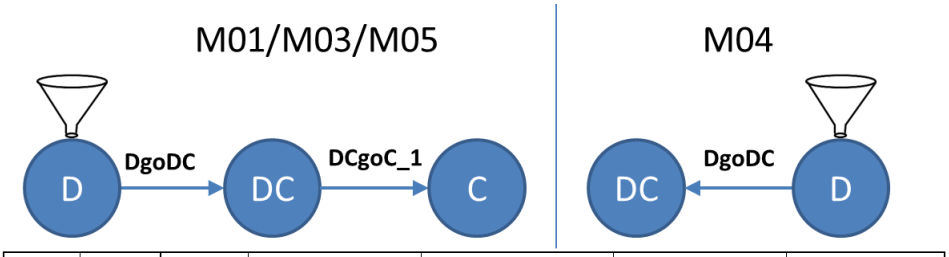
Consulta 01	Base	Registros Retornados	M01/M02			M03			M04			M05		
			MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
	Cem Mil	48.081	396	57,6	53,3	413	108,2	100,1	386	60,8	56,3	299	39,1	36,1
	Um Milhão	475.934	4.434	136,2	126,0	3.670	148,6	137,4	4.236	146,2	135,2	3.502	77,1	71,3

Figura B.4: Consulta 01 - Listar a propriedade d_st4 de todos os vértices D com $d\_st4 <$ ‘K

Pelos resultados com a $Base1Milhao$ vê-se que o melhor desempenho foi com os métodos $M03$ e $M05$, que possuem o vértice D com apenas duas propriedades. Na $Base100Mil$ embora o método $M03$ tenha tido pior desempenho, ele está, em comparação com os métodos $M01$ e $M04$, dentro do intervalo de confiança.

B.1.2 Consulta 02 - D×DC×C

Lista elementos de D com maior ocorrência de um mesmo elemento de C, objetiva testar desempenho com seleção por uma aresta e agrupamento.



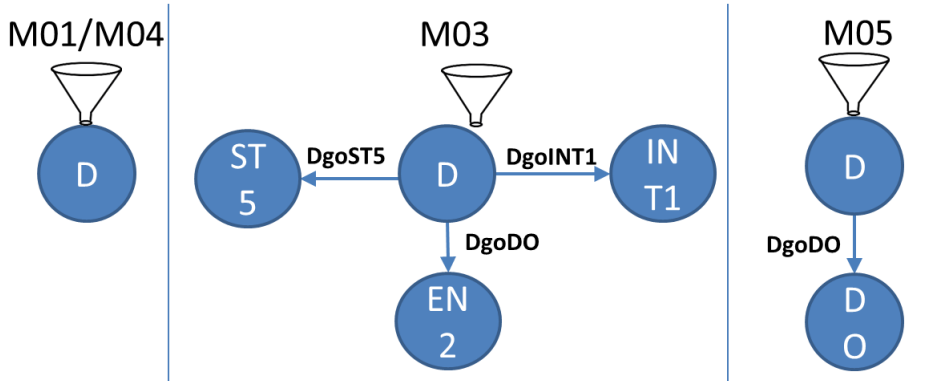
Consulta	Base	Registros Retornados	M01/M06			M03			M04			M05		
			MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
02	Cem Mil	16.041	2.500	143,1	132,3	2.695	32,7	30,2	2.179	139,0	128,6	2.187	132,2	122,2
	Um Milhão	159.978	263.445	34.504,3	31.911,1	468.496	14.869,6	13.752,1	469.688	31.194,8	28.850,4	204.804	8.265,0	7.643,8

Figura B.5: Consulta 02 - D×DC×C

Nesta consulta, o método $M04$ teve o pior desempenho na $Base1Milhao$, decorrente, acredita-se, do número de propriedades do vértice DC, dez, enquanto os demais têm apenas quatro.

B.1.3 Consulta 03 - Listar as propriedade d_st4, d_st5 e d_int1 de todos os vértices D com d_st4 com a primeira letra do nome entre as letras entre 'K´ e 'Z´ e d_en3 = 3

Esta consulta objetiva determinar se há perda de tempo em acessar dados a partir de arestas. Os métodos $M01/M03$ e $M04$ envolvem, como na consulta anterior, apenas o vértice D, já para o método $M03$ a consulta envolve o vértice D e o vértice $EN2$, acessado pela aresta $DgoEN2$, vértice $INT1$, acessado pela aresta $DgoINT1$ e o vértice $ST5$, acessado pela aresta $DgoST5$. O método $M05$ envolve os vértices D e DO e a aresta $DgoDO$.



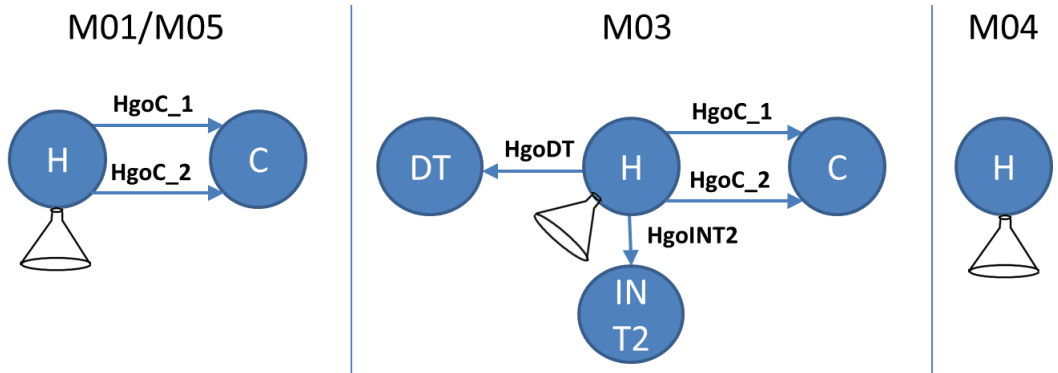
Consulta	Base	Registros Retornados	M01/M02			M03			M04			M05		
			MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
03	Cem Mil	335	176	19,9	18,4	85	28,6	26,5	141	13,9	12,9	25	8,0	7,4
	Um Milhão	3.260	1.473	36,9	34,1	547	9,4	8,7	1.378	40,4	37,4	165	22,0	20,3

Figura B.6: Consulta 03 - Listar as propriedade d_st4, d_st5 e d_int1 de todos os vértices D com $d\_st4 >$ 'K´ e $d\_en3 = 3$

Apesar de ter que buscar os dados do resultado em dois tipos de vértices diferentes, o método $M05$ teve desempenho melhor que o método $M03$, que buscou os dados em quatro tipos diferentes de vértices. Os dois tiveram, ainda, desempenho superior aos métodos $M01$ e $M04$, apesar destes terem todos os dados num único vértice. Acredita-se que isso se deva à aplicação do filtro em D pelo Neo4j, pois primeiro ele coloca na memória todos os elementos de D com suas cinco propriedades e então seleciona os registros no caso dos métodos $M01$ e $M04$. Já no método $M05$ ele coloca na memória apenas duas propriedades por elemento de D, filtra D e só então busca os dados complementares em DO, deixando a consulta mais rápida. No caso de $M03$, os dados complementares são buscados em outros três vértices, daí o aumento do tempo em relação a $M05$.

B.1.4 Consulta 04 - Listar as propriedade de H

Retorna elementos do vértice H combinados com o vértice C em duas datas. Testa desempenho quanto a busca num único vértice ou mais de um vértices e arestas.



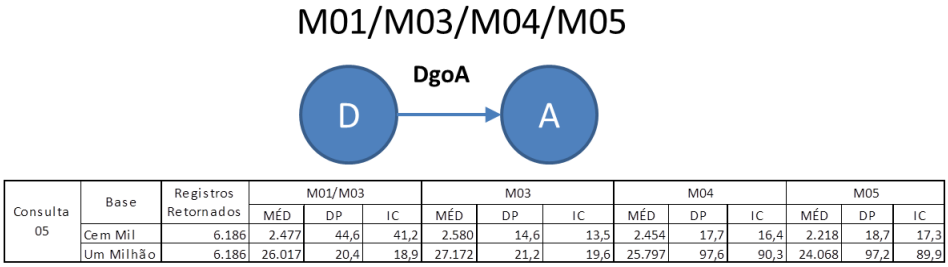
Consulta	Base	Registros Retornados	M01/M03			M03			M04			M05		
			MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
04	Cem Mil	2	15	5,1	4,8	8.408	33,8	31,3	9	2,3	2,1	1.535	88,8	82,1
	Um Milhão	2	15	4,0	3,7	6.147	34,9	32,3	10	1,8	1,6	1.348	33,2	30,7

Figura B.7: Consulta 04 - Listar as propriedade de H

Embora $M04$ tenha tido melhor desempenho nas consultas para as duas bases, este ficou dentro do intervalo de confiança em relação a $M01$. Acredita-se que aqui poderia repetir o resultado anterior, entretanto a baixa quantidade de registros retornados, apenas dois, faz com que o tempo de acesso aos demais vértices seja mais significativo aqui.

B.1.5 Consulta 05 - D×A

Retorna total de elementos de D para cada A. É um grafo típico - dois vértices e a aresta que os relaciona.



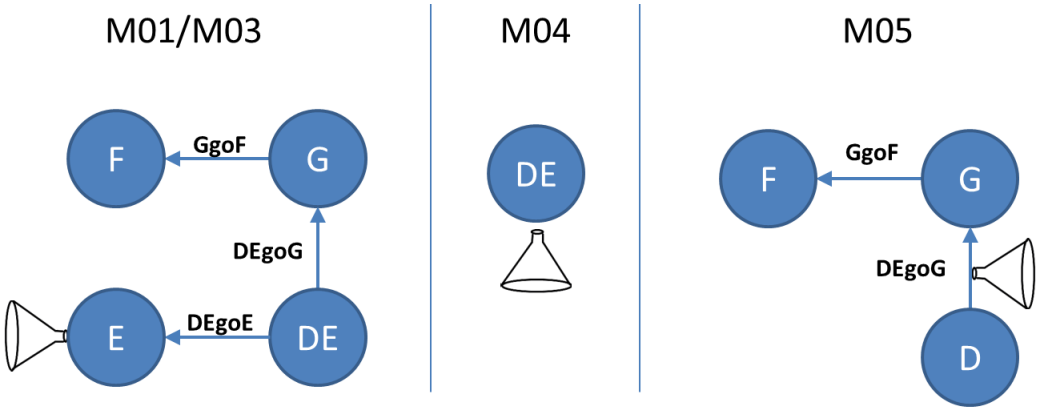
Consulta 05	Base	Registros Retornados	M01/M03			M03			M04			M05		
			MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
	Cem Mil	6.186	2.477	44,6	41,2	2.580	14,6	13,5	2.454	17,7	16,4	2.218	18,7	17,3
	Um Milhão	6.186	26.017	20,4	18,9	27.172	21,2	19,6	25.797	97,6	90,3	24.068	97,2	89,9

Figura B.8: Consulta 05 - D×A

Embora envolva os mesmos vértices e arestas, com a sintaxe do comando de seleção igual para todos os métodos, o método $M05$ teve melhor desempenho que os demais, entretanto, com os outros métodos o resultado foi diferente, o $M03$ teve pior desempenho na base menor e melhor desempenho na base maior. Acredita-se que isso se deve, mais uma vez, ao fato de o Neo4j colocar o vértice D e suas propriedades na memória e depois tabular os dados, assim quando aumenta a quantidade de registros o fato de colocar menos propriedades em memória dominui o tempo de execução de maneira mais significativa.

B.1.6 Consulta 06 - F com mais D

Retorna o total de elementos de D para um mesmo F. Testa desempenho com seleção e agrupamento variando o caminho percorrido (vértices e arestas).



Consulta 06	Base	Registros Retornados	M01/M07			M03			M04			M05		
			MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
	Cem Mil	110	868	79,1	73,2	1.965	33,6	31,1	352	42,6	39,4	780	64,4	59,5
	Um Milhão	206	20.706	190,4	176,0	21.921	107,5	99,4	2.763	93,8	86,8	6.417	37,9	35,0

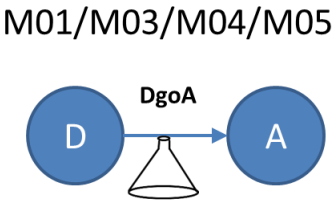
Figura B.9: Consulta 06 - F com mais D

Os resultados dessa consulta demonstram que, na ausência de filtro, quanto menor o número de arestas na consulta, melhor será o desempenho: $M04$ nenhuma, $M05$ duas e

$M01$ e $M03$ três.

B.1.7 Consulta 07 - A×DgoA×A

Retorna total de elementos de D conectados com A onde o atributo com a propriedade d_int4 da aresta $DgoA$ igual à 1.

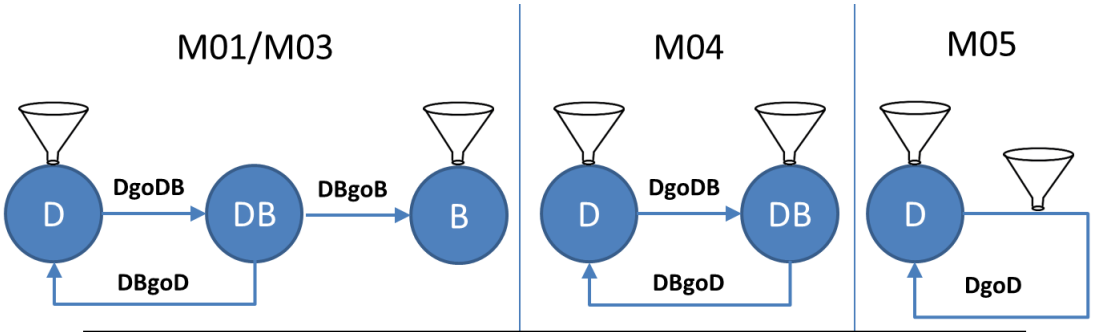


Consulta 07	Base	Registros Retornados	M01/M04			M03			M04			M05		
			MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
	Cem Mil	4.939	1.681	48,8	45,1	1.960	94,9	87,7	1.750	68,5	63,4	1.582	36,3	33,6
	Um Milhão	6.186	21.475	153,7	142,2	16.628	144,1	133,3	19.727	191,7	177,3	17.759	135,9	125,7

Figura B.10: Consulta 07 - A×DgoA×A

B.1.8 Consulta 08 - D×DB×B

Lista D e a lista dos D ligados ao vértice para $d\_pk < 10000$ e $b\_pk = 2$ e testa desempenho conforme seleção na aresta ou no vértice destino da aresta.



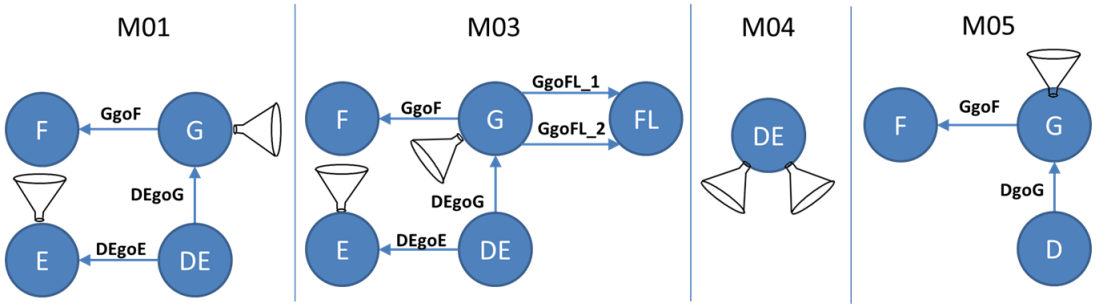
Consulta 08	Base	Registros Retornados	M01/M04			M03			M04			M05		
			MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
	Cem Mil	9.999	1.613	17,3	16,0	1.712	57,6	53,2	1.290	67,3	62,2	777	12,2	11,3
	Um Milhão	9.999	1.328	15,3	14,1	1.275	28,5	26,4	1.094	25,1	23,2	761	17,8	16,5

Figura B.11: Consulta 08 - D×DB×B

Esta consulta mostra a importância do aquecimento, isto é, o descarte da primeira consulta. O tempo médio da execução da consulta para todos os métodos descartando a primeira execução foi menor, para todos os métodos, na $Base1Milhao$ do que na $BaseCemMil$. As consultas devolvem a mesma quantidade de registros, 9.999, então, executada a consulta de aquecimento, os dados na memória são simplesmente devolvidos novamente. O tempo de aquecimento pode ser verificado nas tabelas da seção B.2.

B.1.9 Consulta 09 - G com mais D

Retorna o total de elementos de D para um par (F, G). Objetiva testar o desempenho com seleção e agrupamento variando o caminho percorrido (vértices e arestas) e o local das propriedades usadas como filtro.



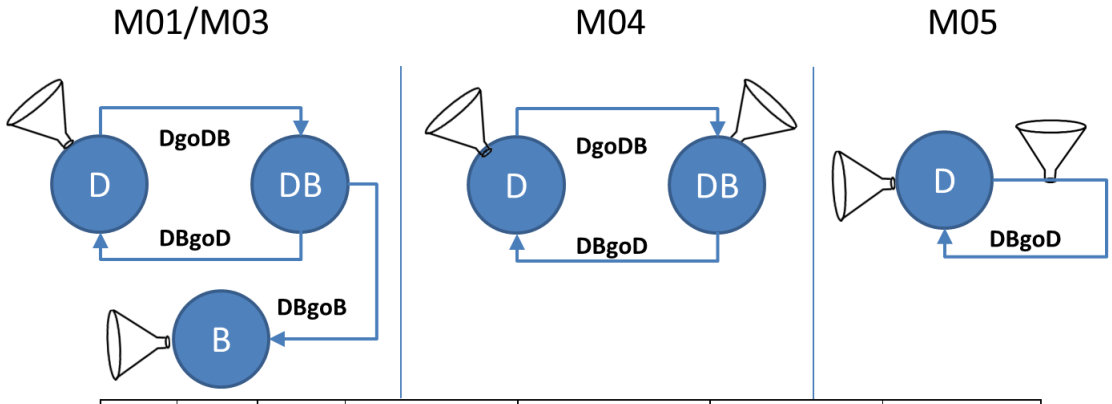
Consulta	Base	Registros Retornados	M01/M07			M03			M04			M05		
			MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
09	Cem Mil	50	494	39,5	36,5	579	25,1	23,2	564	69,0	63,9	314	6,9	6,4
	Um Milhão	50	440	7,5	7,0	796	40,3	37,3	3.203	79,7	73,7	420	7,3	6,7

Figura B.12: Consulta 09 - G com mais D

O melhor desempenho para o método $M05$ mostra que filtrar na aresta é mais eficiente do que filtrar no vértice.

B.1.10 Consulta 10 - D×D com b_pk 2 (grafo)

Retorna um grafo com os elementos de D e seus amigos. Testa o desempenho na seleção de nós com maior ou menor caminho de arestas e vértices.



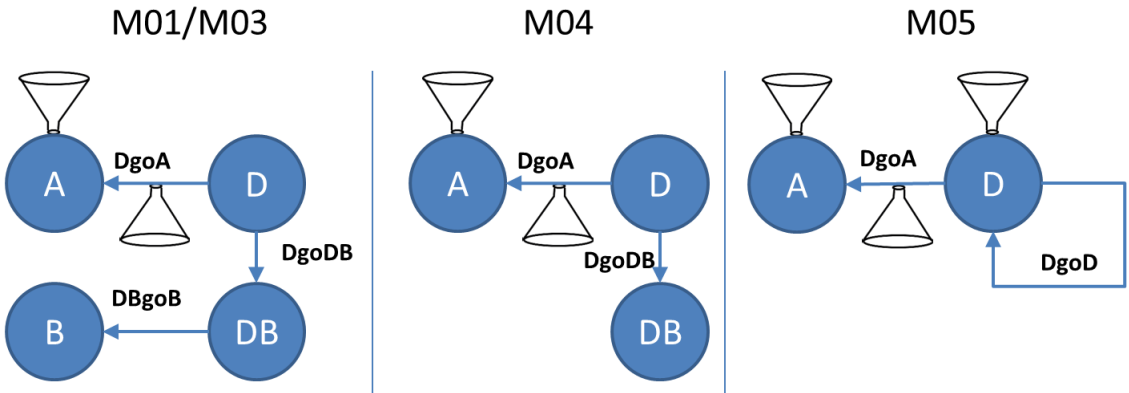
Consulta	Base	Registros Retornados	M01/M09			M03			M04			M05		
			MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
10	Cem Mil	300	56	4,6	4,3	49	5,4	5,0	52	3,9	3,6	29	5,2	4,8
	Um Milhão	2.531	76	6,1	5,6	47	5,0	4,7	46	4,3	3,9	31	3,4	3,1

Figura B.13: Consulta 10 - D×D com b_pk 2 (grafo)

A semelhança das duas consultas anteriores, o menor caminho em $M05$ determinou o melhor desempenho.

B.1.11 Consulta 11 - Maior A×B em D

Esta consulta busca os pares (A, B) com maior ocorrência entre elementos de D. Ela objetiva aferir se o desempenho muda conforme o número de vértices e arestas envolvidos na consulta.



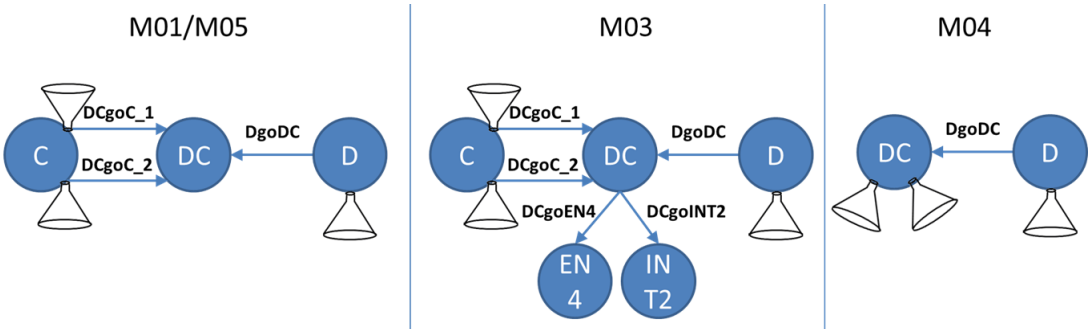
Consulta 11	Base	Registros Retornados	M01/M05			M03			M04			M05		
			MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
	Cem Mil	25	21	3,0	2,7	20	3,3	3,1	15	6,0	5,5	10	2,0	1,8
	Um Milhão	40	110	2,0	1,9	126	2,6	2,4	100	6,2	5,7	62	4,9	4,5

Figura B.14: Consulta 11 - Maior A×B em D

Embora o desempenho das consultas para a $Base100Mil$ tenha sido semelhante, considerando o intervalo de confiança, quando aumenta-se o número de registros, pode-se observar que ao eliminar arestas, diminuindo o caminho, o desempenho melhora. Isso foi, ainda, reforçado, com o método $M05$, onde mais um caminhamento foi eliminado na consulta. Diferente da consulta 02, onde o filtro atuou sobre o vértice D que conforme a modelagem tinha maior ou menor número de propriedades e essa característica interferiu no desempenho da consulta, aqui o filtro está na aresta antes de D, então a quantidade de propriedades de D não foi relevante aqui.

B.1.12 Consulta 12 - D×DC×C×A

Lista elementos de D que tenham uma combinação específica de elementos de C. Objetiva testar o desempenho quando se acessa os dados usando 2 diferentes arestas entre os mesmos dois vértices.



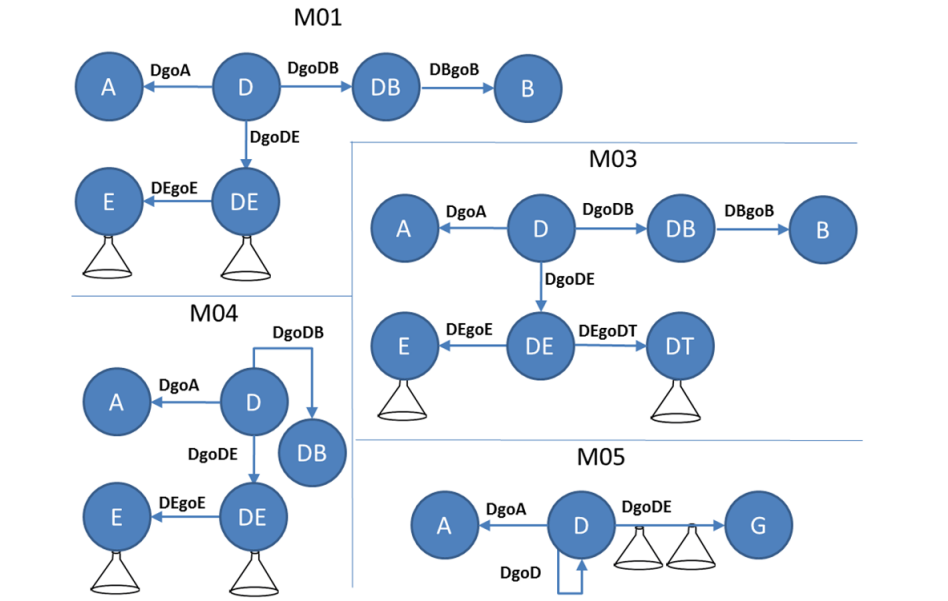
Consulta	Base	Registros Retornados	M01/M06			M03			M04			M05		
			MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
12	Cem Mil	2.264	380	29,4	27,2	566	8,9	8,2	254	6,9	6,4	360	41,6	38,5
	Um Milhão	22.891	3.117	92,9	85,9	29.460	3.502,9	3.239,6	2.145	34,1	31,6	3.402	53,4	49,4

Figura B.15: Consulta 12 - D×DC×C×A

Aqui a base com menor caminho na consulta, $M04$, teve melhor desempenho que as demais, em função de utilizar apenas dois vértices e uma aresta, enquanto os métodos $M01$ e $M05$ usaram três vértices e três arestas e o método $M03$, de pior desempenho, utilizou cinco vértices e cinco arestas.

B.1.13 Consulta 13 - A×D×DE×E×B

Pares (A, B) comuns a elementos de D com seu evento de $e\_pk = 4$ ocorrido no ano de 1985, são o foco destra consulta.



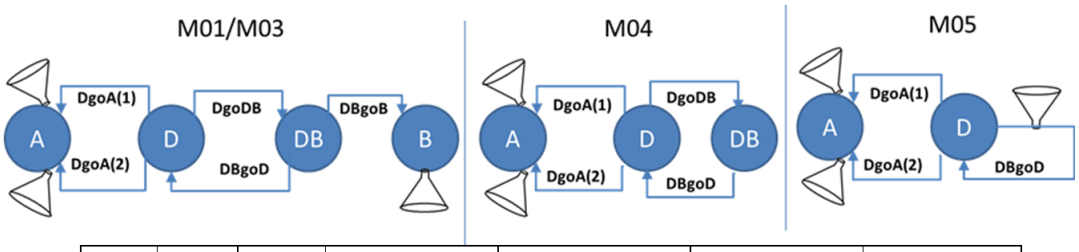
Consulta	Base	Registros Retornados	M01/M05			M03			M04			M05		
			MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
13	Cem Mil	9.794	1.485	47,1	43,5	1.929	37,6	34,8	1.282	63,3	58,6	1.157	47,7	44,1
	Um Milhão	12.372	11.650	299,3	314,1	75.012	1.251,3	1.313,1	10.944	291,9	306,3	10.362	218,5	229,3

Figura B.16: Consulta 13 - A×D×DE×E×B

A consulta para a base $M05$ teve um desempenho melhor que as demais, entretanto, o intervalo de confiança ficou bastante alto em função de, na $Base1Milhao$, a primeira consulta após o aquecimento ter tido um tempo praticamente o dobro das seguintes.

B.1.14 Consulta 14 - A1×D×B×A2

Seleciona todos os elementos de D com determinada ocorrência de A e todos seus amigos com uma outra ocorrência específica de A. Testa desempenho e seleção percorrendo duas vezes as mesmas arestas.



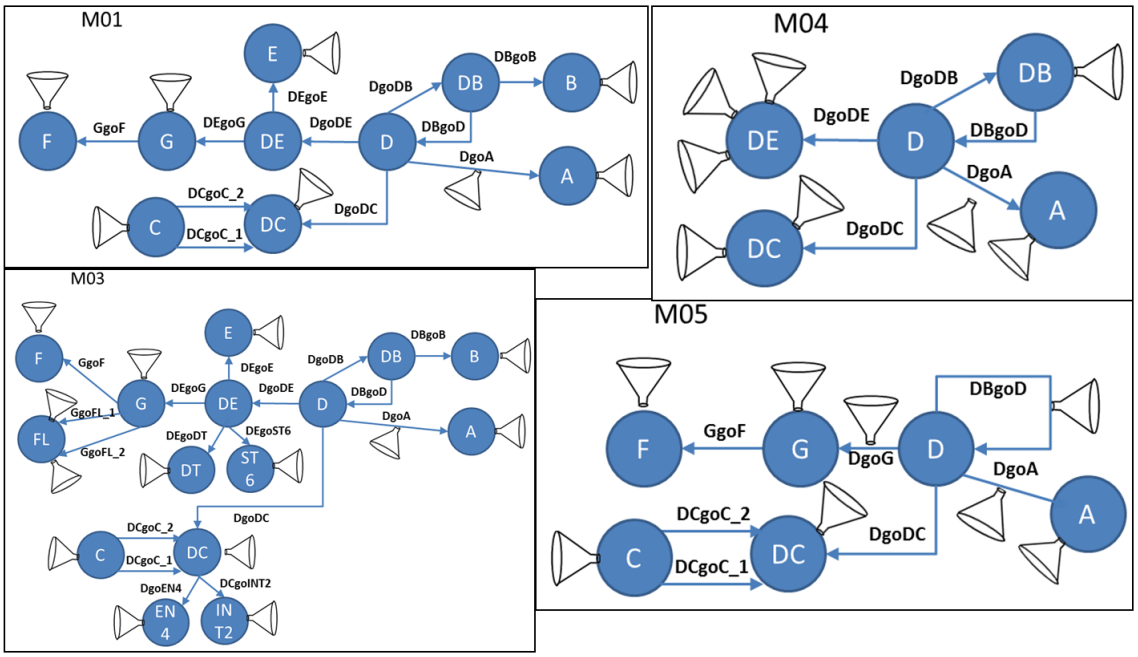
Consulta 14	Base	Registros Retornados	M01/M08			M03			M04			M05		
			MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
	Cem Mil	6	74	6,9	6,4	67	10,7	9,9	59	2,3	2,1	1	0,4	0,3
	Um Milhão	31	454	26,3	24,3	565	43,3	40,0	491	8,0	7,4	401	2,0	1,8

Figura B.17: Consulta 14 -A1×D×B×A2

O menor caminho percorrido para selecionar os elementos com $b\_pk = 2$, na aresta, foi fundamental para o melhor desempenho do método $M05$.

B.1.15 Consulta 15 - A×B×C×D XE×F×G (relação)

Lista elementos de D combinados com DC, B, D novamente e A. Testa desempenho percorrendo todo o grafo, exceto o vértice H e retorna o resultado na forma de relação.



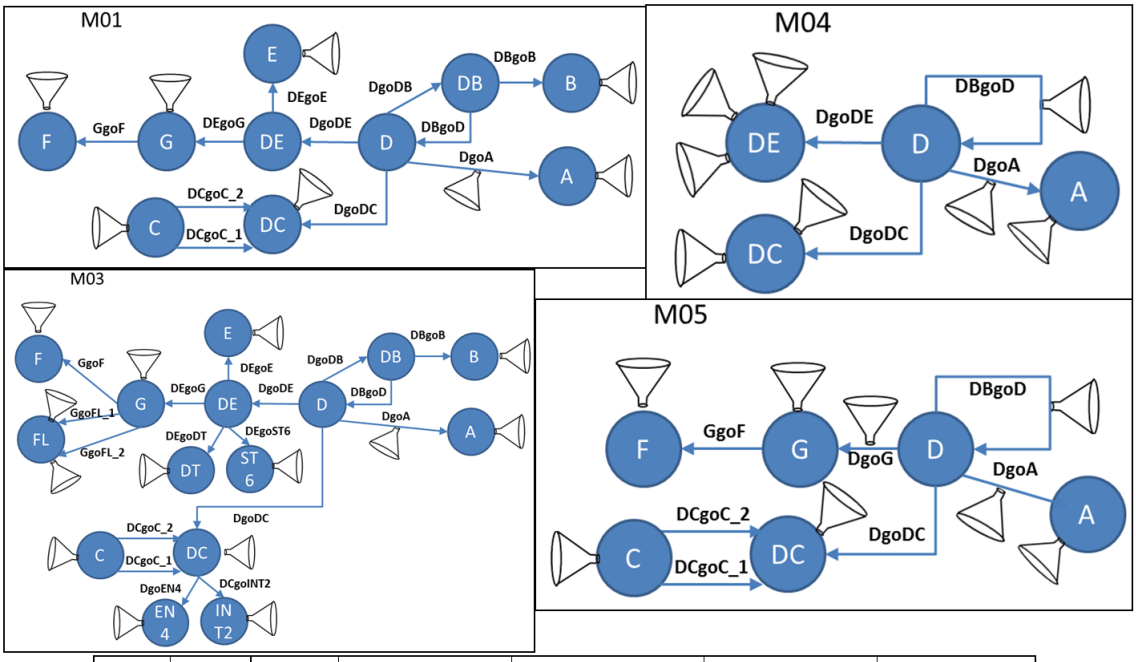
Consulta 15	Base	Registros Retornados	M01/M08			M03			M04			M05		
			MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
	Cem Mil	4	38	3,4	3,1	130	4,0	3,7	1.332	42,8	39,6	45	3,6	3,3
	Um Milhão	221	340	11,2	10,3	697	37,6	34,8	9.147	74,4	68,8	247	11,7	10,8

Figura B.18: Consulta 15 - A×B×C×D XE×F×G (relação)

O menor caminho percorrido para selecionar os elementos de B e E (substituídos por propriedades de arestas) foi fundamental para o melhor desempenho do método $M05$, mesmo que na $BaseCemMil$ ele tenha tido pior desempenho que o método $M01$, mas dentro do intervalo de confiança. Ao aumentar a quantidade de registros envolvidos, $Base1Milhao$, o desempenho do $M05$ melhorou.

B.1.16 Consulta 16 - A×B×C×D XE×F×G (grafo)

Semelhante à consulta anterior, apenas a saída é na forma de grafo, isto é, os vértices e arestas selecionados e seus relacionamentos.



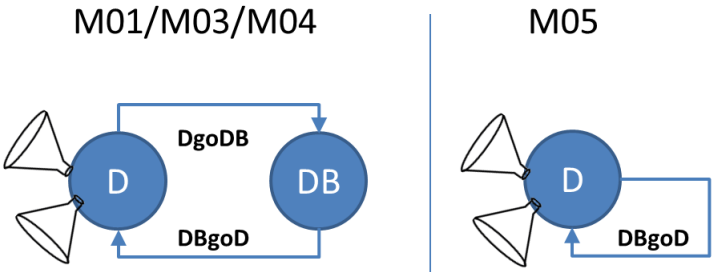
Consulta 16	Base	Registros Retornados	M01/M09			M03			M04			M05		
			MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
	Cem Mil	19	39	4,4	4,1	129	5,9	5,5	1.352	127,4	117,8	42	2,9	2,6
	Um Milhão	221	334	15,8	14,6	642	21,2	19,6	9.342	77,7	71,8	225	31,3	29,0

Figura B.19: Consulta 16 - A×B×C×D XE×F×G (relação)

Se repetiu o resultado da consulta anterior.

B.1.17 Consulta 17 - Caminho Mínimo entre D=1 e D=2

Calcula o caminho mínimo entre dois elementos de D. Testa o desempenho e o caminho retornado conforme o número de vértices e arestas envolvidos.



Consulta 17	Base	Registros Retornados	M01/M10			M03			M04			M05		
			MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
	Cem Mil	7	15	3,0	2,8	15	1,4	1,3	11	3,0	2,8	7	2,9	2,7
	Um Milhão	28	59	10,6	9,8	60	11,0	10,2	42	3,4	3,2	14	1,5	1,4

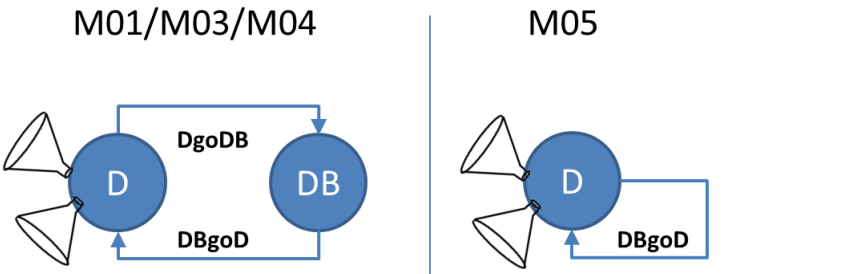
Figura B.20: Consulta 17 - Caminho Mínimo entre $D = 1$ e $D = 2$

Aqui tem-se um reflexo perfeito da diferença das modelagens. Esta consulta e a seguinte são as duas únicas em que o total de registros retornado muda conforme os

métodos. Nos métodos $M01$ e $M03$ o total de vértices percorridos é maior do que nos métodos $M04$ e $M05$, onde a eliminação de um vértice (B), no caso do método $M04$, e dois vértices (B e DB), no caso do método $M05$, torna o caminho percorrido bem menor.

B.1.18 Consulta 18 - Todos os Caminhos Mínimos entre D=1 e D=2

Calcula o caminho mínimo entre dois elementos de D. Testa o desempenho e o caminho retornado conforme o número de vértices e arestas envolvidos.



Consulta 18	Base	Registros Retornados	M01/M10			M03			M04			M05		
			MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC	MÉD	DP	IC
	Cem Mil	54	16	1,9	1,8	17	2,4	2,2	13	2,7	2,5	7	2,5	2,4
	Um Milhão	131	51	6,4	5,9	55	8,8	8,2	48	2,9	2,7	28	4,2	3,9

Figura B.21: Consulta 18 - Todos os Caminhos Mínimos entre $D = 1$ e $D = 2$

A exemplo da consulta anterior, aqui também o número de registros retornados varia conforme a modelagem e isso se reflete no desempenho das consultas.

B.2 Tempo de Execução de Todas as Consultas

Nesta seção são apresentadas oito tabelas, de B.1 até B.8 com o tempo de execução de todas as consultas, para os métodos $M01/M02$, $M03$, $M04$ e $M05$, para as bases $Base100Mil$ e $Base1Milhao$. As tabelas tem as seguintes colunas:

- consulta - número da consulta;
- AQ - tempo, em milissegundos, da primeira execução da consulta, considerada de aquecimento e descartada do cálculo do tempo médio;
- $E01$ à $E07$ - tempo, em milissegundos, das execuções 1, 2, 3, 4, 5, 6 e 7 da consulta após aquela de aquecimento. Cada consulta foi sempre executada em sequência;
- média - tempo médio das sete execuções da consulta;

•DP - desvio padrão sobre o tempo das 7 execuções consideradas para a média.

•IC - intervalo de confiança da amostra, considerando um fator de 95% *student t*.

Tabela B.1: Apêndice B - Base100Mil - Métodos M01/M02 - Execução das Consultas

Consulta	AQ	E01	E02	E03	E04	E05	E06	E07	MÉD	DP	IC
01	904	468	339	420	355	357	356	474	396	57,6	53,3
02	392	22	16	19	18	11	13	7	15	5,1	4,8
03	2.732	2.461	2.521	2.428	2.548	2.486	2.458	2.434	2.477	44,6	41,2
04	10.571	1.671	1.645	1.685	1.630	1.666	1.688	1.781	1.681	48,8	45,1
05	27	22	16	21	22	18	25	22	21	3,0	2,7
06	3.504	1.454	1.514	1.456	1.466	1.579	1.476	1.448	1.485	47,1	43,5
07	1.026	915	846	818	849	809	814	1.028	868	79,1	73,2
08	104	70	68	88	73	68	73	76	74	6,9	6,4
09	595	479	478	582	487	488	467	476	494	39,5	36,5
10	227	215	161	166	154	179	183	177	176	19,9	18,4
11	1.858	1.614	1.636	1.601	1.638	1.596	1.609	1.599	1.613	17,3	16,0
12	2.352	400	409	402	397	365	337	347	380	29,4	27,2
13	107	37	38	34	39	34	44	38	38	3,4	3,1
14	61	45	39	32	38	44	38	37	39	4,4	4,1
15	165	54	53	53	56	66	55	54	56	4,6	4,3
16	2.362	2.479	2.679	2.341	2.692	2.435	2.347	2.524	2.500	143,1	132,3
17	197	19	17	17	13	14	10	15	15	3,0	2,8
18	86	14	16	13	15	16	16	19	16	1,9	1,8

Tabela B.2: Apêndice B - Base100Mil - Método M03 - Execução das Consultas

Consulta	AQ	E01	E02	E03	E04	E05	E06	E07	MÉD	DP	IC
01	909	610	467	318	316	443	317	418	413	108,2	100,1
02	35.563	8.421	8.353	8.389	8.441	8.452	8.409	8.391	8.408	33,8	31,3
03	2.630	2.612	2.576	2.582	2.570	2.573	2.572	2.576	2.580	14,6	13,5
04	9.944	1.909	2.068	1.973	2.105	1.899	1.844	1.923	1.960	94,9	87,7
05	23	16	17	23	22	23	22	16	20	3,3	3,1
06	2.055	1.972	1.912	1.905	1.924	1.877	1.926	1.984	1.929	37,6	34,8
07	2.112	1.974	1.957	1.966	2.016	1.949	1.907	1.985	1.965	33,6	31,1
08	91	59	64	65	62	91	66	64	67	10,7	9,9
09	1.572	587	562	565	586	552	628	571	579	25,1	23,2
10	497	77	147	70	73	68	68	93	85	28,6	26,5
11	1.769	1.774	1.674	1.729	1.802	1.682	1.661	1.660	1.712	57,6	53,2
12	585	577	569	564	577	559	564	553	566	8,9	8,2
13	222	130	130	136	131	131	127	123	130	4,0	3,7
14	202	136	138	124	125	125	132	125	129	5,9	5,5
15	136	51	48	50	59	41	47	47	49	5,4	5,0
16	2.535	2.665	2.725	2.684	2.672	2.754	2.674	2.690	2.695	32,7	30,2
17	178	15	17	14	14	16	13	16	15	1,4	1,3
18	82	17	20	13	15	18	15	18	17	2,4	2,2

Tabela B.3: Apêndice B - Base100Mil - Método M04 - Execução das Consultas

Consulta	AQ	E01	E02	E03	E04	E05	E06	E07	MÉD	DP	IC
01	859	405	368	333	460	337	326	471	386	60,8	56,3
02	308	11	9	13	6	9	8	8	9	2,3	2,1
03	2.402	2.480	2.433	2.430	2.454	2.463	2.452	2.464	2.454	17,7	16,4
04	10.640	1.706	1.722	1.868	1.726	1.696	1.827	1.704	1.750	68,5	63,4
05	166	27	13	10	11	15	18	11	15	6,0	5,5
06	4.456	1.269	1.245	1.261	1.272	1.246	1.424	1.259	1.282	63,3	58,6
07	347	332	448	343	338	333	341	329	352	42,6	39,4
08	572	61	62	58	58	55	59	60	59	2,3	2,1
09	609	531	525	607	528	527	705	524	564	69,0	63,9
10	236	142	151	154	157	122	134	126	141	13,9	12,9
11	2.265	1.254	1.269	1.440	1.261	1.269	1.250	1.285	1.290	67,3	62,2
12	2.927	259	266	254	253	246	249	249	254	6,9	6,4
13	1.576	1.317	1.403	1.306	1.300	1.371	1.343	1.283	1.332	42,8	39,6
14	1.507	1.264	1.495	1.266	1.259	1.342	1.272	1.568	1.352	127,4	117,8
15	153	59	52	48	50	56	51	50	52	3,9	3,6
16	2.006	2.132	2.108	2.441	2.150	2.004	2.267	2.149	2.179	139,0	128,6
17	148	17	13	9	10	10	9	9	11	3,0	2,8
18	47	14	16	11	14	11	8	14	13	2,7	2,5

Tabela B.4: Apêndice B - Base100Mil - Método M05 - Execução das Consultas

Consulta	AQ	E01	E02	E03	E04	E05	E06	E07	MÉD	DP	IC
01	822	385	300	286	285	278	284	272	299	39,1	36,1
02	3.319	1.508	1.409	1.503	1.707	1.535	1.539	1.541	1.535	88,8	82,1
03	2.396	2.222	2.225	2.247	2.209	2.216	2.222	2.185	2.218	18,7	17,3
04	10.666	1.569	1.623	1.583	1.638	1.546	1.572	1.543	1.582	36,3	33,6
05	195	13	7	9	9	10	11	8	10	2,0	1,8
06	3.652	1.130	1.150	1.102	1.212	1.224	1.110	1.168	1.157	47,7	44,1
07	931	726	855	728	878	746	730	799	780	64,4	59,5
08	19	1	2	1	1	1	1	1	1	0,4	0,3
09	488	320	311	312	311	316	324	303	314	6,9	6,4
10	356	38	30	22	27	23	18	14	25	8,0	7,4
11	871	776	776	801	770	768	783	765	777	12,2	11,3
12	3.195	347	343	341	344	453	354	335	360	41,6	38,5
13	154	47	40	46	49	47	43	40	45	3,6	3,3
14	45	44	38	39	46	42	42	44	42	2,9	2,6
15	87	18	30	33	28	27	31	33	29	5,2	4,8
16	2.124	2.003	2.268	2.102	2.369	2.315	2.116	2.136	2.187	132,2	122,2
17	28	6	11	9	8	4	9	3	7	2,9	2,7
18	20	5	5	6	11	10	6	5	7	2,5	2,4

Tabela B.5: Apêndice B - Base1Milhao - Métodos M01/M02 - Execução das Consultas

Consulta	AQ	E01	E02	E03	E04	E05	E06	E07	MÉD	DP	IC
01	6.006	4.245	4.456	4.679	4.356	4.478	4.465	4.356	4.434	136,2	126,0
02	954	23	12	14	11	13	15	14	15	4,0	3,7
03	126.435	25.999	26.014	26.032	25.999	26.004	26.017	26.055	26.017	20,4	18,9
04	152.004	21.342	21.321	21.678	21.555	21.657	21.342	21.431	21.475	153,7	142,2
05	20.689	112	107	109	110	109	111	113	110	2,0	1,9
06	354.256	75.614	11.533	11.822	11.876	11.456	12.001	11.211	11.650	299,3	314,1
07	22.346	20.533	20.456	20.654	21.001	20.876	20.768	20.656	20.706	190,4	176,0
08	305.654	461	447	466	478	425	485	415	454	26,3	24,3
09	998	454	431	433	444	440	441	440	440	7,5	7,0
10	1.620	1.532	1.499	1.456	1.491	1.432	1.467	1.432	1.473	36,9	34,1
11	233.187	1.342	1.329	1.332	1.333	1.345	1.314	1.302	1.328	15,3	14,1
12	37.765	3.231	3.212	2.965	3.094	3.045	3.121	3.148	3.117	92,9	85,9
13	123.478	345	345	357	321	333	339	341	340	11,2	10,3
14	357	315	341	309	333	344	348	347	334	15,8	14,6
15	17.145	83	78	65	82	73	77	75	76	6,1	5,6
16	354.121	214.123	232.432	321.123	256.790	265.437	276.543	277.666	263.445	34.504,3	31.911,1
17	52.951	77	55	68	45	53	54	59	59	10,6	9,8
18	3.003	61	43	47	48	47	53	57	51	6,4	5,9

Tabela B.6: Apêndice B - Base1Milhao - Método M03 - Execução das Consultas

Consulta	AQ	E01	E02	E03	E04	E05	E06	E07	MÉD	DP	IC
01	4.963	3.678	3.876	3.567	3.777	3.770	3.568	3.452	3.670	148,6	137,4
02	40.013	6.100	6.199	6.175	6.166	6.122	6.145	6.122	6.147	34,9	32,3
03	107.542	27.145	27.158	27.187	27.192	27.201	27.155	27.167	27.172	21,2	19,6
04	136.014	16.672	16.304	16.645	16.701	16.699	16.678	16.695	16.628	144,1	133,3
05	33.871	124	130	125	125	126	129	123	126	2,6	2,4
06	1.425.128	432.001	77.014	75.258	74.652	75.412	73.215	74.521	75.012	1.251,3	1.313,1
07	28.001	22.014	22.077	21.750	21.863	21.935	21.866	21.944	21.921	107,5	99,4
08	491.256	654	522	545	537	578	555	561	565	43,3	40,0
09	1.878	821	874	777	754	787	794	767	796	40,3	37,3
10	15.003	547	564	538	537	545	545	554	547	9,4	8,7
11	165.125	1.263	1.255	1.338	1.265	1.264	1.263	1.278	1.275	28,5	26,4
12	84.562	33.012	35.012	29.145	28.015	27.451	29.001	24.587	29.460	3.502,9	3.239,6
13	154.231	701	770	699	688	678	645	699	697	37,6	34,8
14	2.541	688	643	632	645	631	630	628	642	21,2	19,6
15	27.013	39	44	45	48	47	54	52	47	5,0	4,7
16	432.125	444.512	454.784	466.124	478.125	487.562	469.888	478.475	468.496	14.869,6	13.752,1
17	97.854	83	58	63	49	54	56	57	60	11,0	10,2
18	13.685	52	49	45	63	71	53	54	55	8,8	8,2

Tabela B.7: Apêndice B - Base1Milhao - Método M04 - Execução das Consultas

Consulta	AQ	E01	E02	E03	E04	E05	E06	E07	MÉD	DP	IC
01	4.901	4.278	4.054	4.023	4.423	4.337	4.289	4.245	4.236	146,2	135,2
02	688	11	12	10	7	8	10	11	10	1,8	1,6
03	43.567	26.001	25.748	25.741	25.700	25.814	25.779	25.799	25.797	97,6	90,3
04	131.056	19.431	20.001	19.784	19.564	19.662	19.874	19.774	19.727	191,7	177,3
05	22.410	93	105	89	103	102	104	102	100	6,2	5,7
06	255.632	35.632	11.478	10.632	10.755	10.863	10.962	10.974	10.944	291,9	306,3
07	14.013	2.917	2.874	2.756	2.684	2.698	2.701	2.714	2.763	93,8	86,8
08	312.784	503	497	487	478	488	489	493	491	8,0	7,4
09	3.332	3.199	3.212	3.223	3.101	3.145	3.356	3.183	3.203	79,7	73,7
10	1.601	1.432	1.401	1.324	1.325	1.388	1.401	1.378	1.378	40,4	37,4
11	184.562	1.145	1.087	1.064	1.089	1.101	1.084	1.087	1.094	25,1	23,2
12	45.625	2.178	2.145	2.128	2.099	2.121	2.145	2.199	2.145	34,1	31,6
13	118.745	9.145	8.999	9.125	9.222	9.214	9.148	9.173	9.147	74,4	68,8
14	9.555	9.232	9.315	9.245	9.397	9.401	9.415	9.388	9.342	77,7	71,8
15	18.001	48	39	49	49	44	43	51	46	4,3	3,9
16	512.124	412.125	457.635	499.874	487.521	451.214	487.899	491.546	469.688	31.194,8	28.850,4
17	44.523	47	46	39	40	38	43	42	42	3,4	3,2
18	15.985	47	48	48	46	47	45	54	48	2,9	2,7

Tabela B.8: Apêndice B - Base1Milhao - Método M05 - Execução das Consultas

Consulta	AQ	E01	E02	E03	E04	E05	E06	E07	MÉD	DP	IC
01	4.198	3.654	3.398	3.465	3.487	3.498	3.501	3.513	3.502	77,1	71,3
02	3.109	1.309	1.299	1.356	1.345	1.362	1.378	1.387	1.348	33,2	30,7
03	45.870	23.978	23.987	24.014	24.214	24.125	24.163	23.994	24.068	97,2	89,9
04	132.145	18.013	17.788	17.845	17.654	17.624	17.687	17.699	17.759	135,9	125,7
05	11.458	73	61	63	62	59	60	59	62	4,9	4,5
06	134.231	31.238	10.784	10.234	10.345	10.368	10.260	10.179	10.362	218,5	229,3
07	133.987	6.401	6.368	6.457	6.478	6.398	6.401	6.413	6.417	37,9	35,0
08	121.452	402	398	401	400	403	404	401	401	2,0	1,8
09	1.287	409	425	414	418	421	423	431	420	7,3	6,7
10	22.722	161	129	145	187	166	188	178	165	22,0	20,3
11	161.478	787	787	751	749	748	753	752	761	17,8	16,5
12	7.456	3.509	3.400	3.330	3.387	3.399	3.387	3.401	3.402	53,4	49,4
13	48.123	271	253	241	236	241	244	245	247	11,7	10,8
14	264	156	251	234	238	236	230	233	225	31,3	29,0
15	8.386	31	38	32	29	28	29	30	31	3,4	3,1
16	50.652	187.823	215.212	207.456	206.852	205.142	206.145	205.001	204.804	8.265,0	7.643,8
17	7.897	16	14	16	13	12	14	15	14	1,5	1,4
18	1.111	33	29	27	28	21	33	26	28	4,2	3,9

A seguir vem as listagens dos *scripts* das consultas efetuadas nas bases para os cinco métodos analisados nesta dissertação.

B.3 Scripts Cypher das Consultas Efetuadas nas Bases

B.3.1 Consultas Cypher - Métodos M01 e M02

Listagem B.1: Scripts das Consultas nas Bases Neo4j - Métodos M01 e M02

```
// 01 -- Listar st4 de todos os D com st4 entre 'A' e 'J'
MATCH (d:D)
WHERE toUpper(d.d_st1) < 'K'
RETURN d.d_st1
ORDER BY d.d_st1
LIMIT 1000000;

// 02 -- D X DC X C - contador
MATCH (d:D)-[x:DgoDC]->(dc:DC),
      (dc)-[y:DCgoC_1]->(c1:C)
```

```
WHERE toUpper(SUBSTRING(d.d_st1, 0, 1)) = 'A' OR toUpper(
   SUBSTRING(d.d_st1, 0, 1)) = 'L'
WITH d.d_st1 AS d, c1.c_st1 AS c, COUNT(1) AS qtd
WHERE qtd > 1
RETURN d, c, qtd
ORDER BY qtd DESC, d;

// 03 -- Listar st4, st5, int1 de todos os D com st4
//       entre 'K' E 'Z' e com en2 = 3
//       em ordem descrescente de pk
MATCH (d:D)
WHERE toUpper(d.d_st1) > 'K' AND d.d_en2 = 0 AND d.d_int1 <>
   267
RETURN d.d_st1, d.d_st2, d.d_int1
ORDER BY d.d_pk DESC;

// 04 -- H x C
MATCH (hSol:H{h_dt1:'1988/11/22'})-[r01n:HgoC_1]->(cSol:C{
   c_pk:105}),(hSol)-[r01s:HgoC_2]->(cSol2:C),
      (hLua:H{h_dt1:'1988/11/22'})-[r02n:HgoC_1]->(cLua:C{
         c_pk:112}),(hLua)-[r02s:HgoC_2]->(cLua2:C),
      (hMer:H{h_dt1:'1988/11/22'})-[r03n:HgoC_1]->(cMer:C{
         c_pk:103}),(hMer)-[r03s:HgoC_2]->(cMer2:C),
      (hVen:H{h_dt1:'1988/11/22'})-[r04n:HgoC_1]->(cVen:C{
         c_pk:108}),(hVen)-[r04s:HgoC_2]->(cVen2:C),
      (hTer:H{h_dt1:'1988/11/22'})-[r05n:HgoC_1]->(cTer:C{
         c_pk:111}),(hTer)-[r05s:HgoC_2]->(cTer2:C),
      (hMar:H{h_dt1:'1988/11/22'})-[r06n:HgoC_1]->(cMar:C{
         c_pk:109}),(hMar)-[r06s:HgoC_2]->(cMar2:C),
      (hJup:H{h_dt1:'1988/11/22'})-[r07n:HgoC_1]->(cJup:C{
         c_pk:107}),(hJup)-[r07s:HgoC_2]->(cJup2:C),
      (hSat:H{h_dt1:'1988/11/22'})-[r08n:HgoC_1]->(cSat:C{
         c_pk:106}),(hSat)-[r08s:HgoC_2]->(cSat2:C),
      (hUra:H{h_dt1:'1988/11/22'})-[r09n:HgoC_1]->(cUra:C{
         c_pk:102}),(hUra)-[r09s:HgoC_2]->(cUra2:C),
```

```
        (hNet:H{h_dt1:'1988/11/22'})-[r10n:HgoC_1]->(cNet:C{
            c_pk:104}),(hNet)-[r10s:HgoC_2]->(cNet2:C),
        (hPlu:H{h_dt1:'1988/11/22'})-[r11n:HgoC_1]->(cPlu:C{
            c_pk:101}),(hPlu)-[r11s:HgoC_2]->(cPlu2:C),
        (hEln:H{h_dt1:'1988/11/22'})-[r12n:HgoC_1]->(cEln:C{
            c_pk:110}),(hEln)-[r12s:HgoC_2]->(cEln2:C)
    RETURN hSol.h_dt1 AS data,
           hSol.h_int2 + '/' + cSol2.c_st1 AS sol,
           hLua.h_int2 + '/' + cLua2.c_st1 AS lua,
           hMer.h_int2 + '/' + cMer2.c_st1 AS mer,
           hVen.h_int2 + '/' + cVen2.c_st1 AS ven,
           hTer.h_int2 + '/' + cTer2.c_st1 AS ter,
           hMar.h_int2 + '/' + cMar2.c_st1 AS mar,
           hJup.h_int2 + '/' + cJup2.c_st1 AS jup,
           hSat.h_int2 + '/' + cSat2.c_st1 AS sat,
           hUra.h_int2 + '/' + cUra2.c_st1 AS ura,
           hNet.h_int2 + '/' + cNet2.c_st1 AS net,
           hPlu.h_int2 + '/' + cPlu2.c_st1 AS plu,
           hEln.h_int2 + '/' + cEln2.c_st1 AS eln
    UNION MATCH (hSol:H{h_dt1:'1988/11/23'})-[r01n:HgoC_1]->(cSol
        :C{c_pk:105}),(hSol)-[r01s:HgoC_2]->(cSol2:C),
        (hLua:H{h_dt1:'1988/11/23'})-[r02n:HgoC_1]->(cLua:C{
            c_pk:112}),(hLua)-[r02s:HgoC_2]->(cLua2:C),
        (hMer:H{h_dt1:'1988/11/23'})-[r03n:HgoC_1]->(cMer:C{
            c_pk:103}),(hMer)-[r03s:HgoC_2]->(cMer2:C),
        (hVen:H{h_dt1:'1988/11/23'})-[r04n:HgoC_1]->(cVen:C{
            c_pk:108}),(hVen)-[r04s:HgoC_2]->(cVen2:C),
        (hTer:H{h_dt1:'1988/11/23'})-[r05n:HgoC_1]->(cTer:C{
            c_pk:111}),(hTer)-[r05s:HgoC_2]->(cTer2:C),
        (hMar:H{h_dt1:'1988/11/23'})-[r06n:HgoC_1]->(cMar:C{
            c_pk:109}),(hMar)-[r06s:HgoC_2]->(cMar2:C),
        (hJup:H{h_dt1:'1988/11/23'})-[r07n:HgoC_1]->(cJup:C{
            c_pk:107}),(hJup)-[r07s:HgoC_2]->(cJup2:C),
        (hSat:H{h_dt1:'1988/11/23'})-[r08n:HgoC_1]->(cSat:C{
            c_pk:106}),(hSat)-[r08s:HgoC_2]->(cSat2:C),
```

```
        (hUra:H{h_dt1:'1988/11/23'})-[r09n:HgoC_1]->(cUra:C{
           c_pk:102}),(hUra)-[r09s:HgoC_2]->(cUra2:C),
        (hNet:H{h_dt1:'1988/11/23'})-[r10n:HgoC_1]->(cNet:C{
           c_pk:104}),(hNet)-[r10s:HgoC_2]->(cNet2:C),
        (hPlu:H{h_dt1:'1988/11/23'})-[r11n:HgoC_1]->(cPlu:C{
           c_pk:101}),(hPlu)-[r11s:HgoC_2]->(cPlu2:C),
        (hEln:H{h_dt1:'1988/11/23'})-[r12n:HgoC_1]->(cEln:C{
           c_pk:110}),(hEln)-[r12s:HgoC_2]->(cEln2:C)
RETURN hSol.h_dt1 AS data,
       hSol.h_int2 + '/' + cSol2.c_st1 AS sol,
       hLua.h_int2 + '/' + cLua2.c_st1 AS lua,
       hMer.h_int2 + '/' + cMer2.c_st1 AS mer,
       hVen.h_int2 + '/' + cVen2.c_st1 AS ven,
       hTer.h_int2 + '/' + cTer2.c_st1 AS ter,
       hMar.h_int2 + '/' + cMar2.c_st1 AS mar,
       hJup.h_int2 + '/' + cJup2.c_st1 AS jup,
       hSat.h_int2 + '/' + cSat2.c_st1 AS sat,
       hUra.h_int2 + '/' + cUra2.c_st1 AS ura,
       hNet.h_int2 + '/' + cNet2.c_st1 AS net,
       hPlu.h_int2 + '/' + cPlu2.c_st1 AS plu,
       hEln.h_int2 + '/' + cEln2.c_st1 AS eln;

// 05 -- Maior A em D
MATCH (a:A)<-[r:DgoA]-(d:D)
WITH a.a_st1 AS a,  COUNT(d) AS qtd_d
RETURN a, qtd_d
ORDER BY qtd_d DESC, a;

// 06 -- F com mais D
MATCH (f:F)<-[GgoF]-()<-[DEgoG]-()-[DEgoE]->(e:E{e_pk:4})
RETURN f.f_st1 AS f, COUNT(1) AS qtd
ORDER BY qtd DESC, f;

// 07 -- Maior de grau 1 A em D
MATCH (a:A)<-[r:DgoA{da_int4:1}]-(d:D)
```

```
WITH a.a_st1 AS a,  COUNT(d) AS qtd_d
RETURN a, qtd_d
ORDER BY qtd_d DESC, a;

// 08 -- D X B para d_pk < 10000 e b_pk = 2
MATCH (d1:D)-[r01:DgoDB]->(db:DB)-[r02:DBgoB]-({b_pk:2}),(db)
   -[r03:DBgoD]->(d2:D)
WHERE d1.d_pk < 10000
RETURN d1.d_st1 as d, COLLECT(d2.d_st1) AS bs
ORDER BY SIZE(bs) DESC, d;

// 09 -- G X D
MATCH (f:F)<-[GgoF]-(g:G)<-[DEgoG]-()-[DEgoE]->(e:E{e_pk:4})
WHERE g.g_fl1 < 0 AND g.g_fl2 < 0 AND SUBSTRING(g.g_st1, 0,
   1) = 'B'
RETURN f.f_st1 AS f, g.g_st1 AS g, COUNT(1) AS qtd
ORDER BY qtd DESC, f, g
LIMIT 50;

// 10 -- D X D com b_pk 2 - saída em vértices
MATCH (d1:D)-[r01:DgoDB]->(db:DB)-[r02:DBgoB]-({b_pk:2}),(db)
   -[r03:DBgoD]->(d2:D)
WHERE d1.d_pk > 90000 AND d1.d_pk < 90201
RETURN d1, d2;

// 11 -- Maior A de a_pk até 20 de grau 2 X B em  D
MATCH (a:A)<-[r:DgoA{da_int4:2}]-(d:D)-[DgoDB]->()-[DBgoB]->(
   b:B)
WHERE a.a_pk < 21
WITH a.a_st1 AS a,  b.b_st1 AS b, COUNT(d) AS qtd_d
RETURN a, b, qtd_d
ORDER BY qtd_d DESC, a, b;

// 12 -- DC X C X D
```

```
MATCH (dc:DC)-[x:DCgoC_1]->(c1:C{c_pk:203}),(dc)-[y:DCgoC_2
   ]->(c2:C{c_pk:206}),(dc)<-[DgoDC]-(d:D)
WHERE toUpper(SUBSTRING(d.d_st1, 0, 1)) >= 'J' AND toUpper(
   SUBSTRING(d.d_st1, 0, 1)) <= 'R'
RETURN d.d_st1 AS d, dc.dc_int3,  dc.dc_en4
ORDER BY d, dc.dc_int3;

// 13 -- A X D X DE X E X B
MATCH (e:E{e_pk:4})<-[DEgoE]-(de:DE)<-[DgoDE]-(d:D)
WHERE SUBSTRING(de.de_dt2, 0, 4) = '1985'
WITH SUBSTRING(de.de_dt2, 0, 4) AS ano, d
MATCH (d)-[r:DgoA]->(a:A), (d)-[DgoDB]->()-[DBgoB]->(b:B)
WITH ano, a.a_st1 AS a,  b.b_st1 AS b, COUNT(d) AS qtd_d
RETURN ano, a, b, qtd_d
ORDER BY qtd_d DESC, ano, a, b;

// 14 -- A1 X D X B X A2
MATCH (a:A{a_pk:1871})<-[r1:DgoA]-(d1:D)-[r2:DgoDB]->(db:DB)
   -[r3:DBgoB]->(b:B{b_pk:2}),
      (db)-[r4:DBgoD]->(d2:D)-[r5:DgoA]->({a_pk:1879})
RETURN d1.d_st1, d2.d_st1;

// 15 -- A X B X C X D XE X F X G - saída em relação
MATCH (f1:F)<-[r01:GgoF]-(g1:G)
WHERE toLower(SUBSTRING(f1.f_st1, 0, 1)) = 'c' AND g1.g_fl1 <
    0 AND g1.g_fl2 < 0
WITH g1
MATCH (g1)<-[r02:DEgoG]-(de1:DE)-[r03:DEgoE]->({e_pk:4}),(de1
   )<-[r04:DgoDE]-(d1:D)
WHERE de1.de_dt2 < '2000/01/01'
WITH d1
MATCH (d1)-[r05:DgoDB]->(db:DB)-[r06:DBgoB]-({b_pk:2}), (db)
   -[r07:DBgoD]-(d2:D),
      (d2)-[r08:DgoDE]->(de:DE)-[r09:DEgoE]->({e_pk:4})
WHERE de.de_st1 < '12:00'
```

```
WITH d1, COUNT(d2) AS qtd_d2
WHERE qtd_d2 > 2
OPTIONAL MATCH (d1)-[r10:DgoA{da_int4:10}]->(a:A)
WHERE toLower(SUBSTRING(a.a_st1, 0, 1)) = 'm'
WITH d1, qtd_d2, COUNT(a) AS qtd_a
WITH d1, qtd_d2, qtd_a
MATCH (d1)-[r11:DgoDC]->(dc:DC)-[r12:DCgoC_1]->({c_pk:203}),(
   dc)-[r13:DCgoC_2]->(c:C)
WHERE dc.dc_en4 = '+'
  AND dc.dc_int2 > '1500'
RETURN d1.d_st1, c.c_st1, dc.dc_int2, dc.dc_en4, qtd_d2,
   qtd_a
ORDER BY dc.dc_int2;

// 16 -- A X B X C X D XE X F X G - saída em vértices
MATCH (f1:F)<-[r01:GgoF]-(g1:G)
WHERE toLower(SUBSTRING(f1.f_st1, 0, 1)) = 'c' AND g1.g_fl1 <
    0 AND g1.g_fl2 < 0
WITH g1
MATCH (g1)<-[r02:DEgoG]-(de1:DE)-[r03:DEgoE]->({e_pk:4}),(de1
   )<-[r04:DgoDE]-(d1:D)
WHERE de1.de_dt2 < '2000/01/01'
WITH d1
MATCH (d1)-[r05:DgoDB]->(db:DB)-[r06:DBgoB]-({b_pk:2}), (db)
   -[r07:DBgoD]-(d2:D),
      (d2)-[r08:DgoDE]->(de:DE)-[r09:DEgoE]->({e_pk:4})
WHERE de.de_st1 < '12:00'
WITH d1, COUNT(d2) AS qtd_d2
WHERE qtd_d2 > 2
OPTIONAL MATCH (d1)-[r10:DgoA{da_int4:10}]->(a:A)
WHERE toLower(SUBSTRING(a.a_st1, 0, 1)) = 'm'
WITH d1, qtd_d2, COUNT(a) AS qtd_a
WITH d1, qtd_d2, qtd_a
MATCH (d1)-[r11:DgoDC]->(dc:DC)-[r12:DCgoC_1]->({c_pk:203}),(
   dc)-[r13:DCgoC_2]->(c:C)
```

```
WHERE dc.dc_en4 = '+'
  AND dc.dc_int2 > '1500'
RETURN d1, c, dc;

// 17 -- Caminho Mínimo entre D=1 e D=2
MATCH (d:D{d_pk:1}), (to:D{d_pk:2})
MATCH p = shortestPath( (d)-[r1:DgoDB|:DBgoD*]->(to) )
RETURN d, r1, p;

// 18 -- Todos Caminhos Mínimo entre D=1 e D=2
MATCH (d:D{d_pk:1}), (to:D{d_pk:2})
MATCH p = allShortestPaths( (d)-[r1:DgoDB|:DBgoD*]->(to) )
RETURN d, r1, p;
```

B.3.2 Consultas Cypher - Método M03

Listagem B.2: Scripts das Consultas nas Bases Neo4j - Método M03

```
// 01 -- Listar st4 de todos os D com st4 entre 'A' e 'J'
MATCH (d:D)
WHERE toUpper(d.d_st1) < 'K'
RETURN d.d_st1
ORDER BY d.d_st1
LIMIT 1000000;

// 02 -- D X DC X C
MATCH (d:D)-[x:DgoDC]->(dc:DC),
      (dc)-[y:DCgoC_1]->(c1:C)
WHERE toUpper(SUBSTRING(d.d_st1, 0, 1)) = 'A' OR toUpper(
   SUBSTRING(d.d_st1, 0, 1)) = 'L'
WITH d.d_st1 AS d, c1.c_st1 AS c, COUNT(1) AS qtd
WHERE qtd > 1
RETURN d, c, qtd
ORDER BY qtd DESC, d;

// 03 -- Listar st4, st5, int1 de todos os D com st4
```

```
//        entre 'K' E 'Z' e com en2 = 3
//        em ordem descrescente de pk
MATCH (d:D)-[DgoEN2]->(en2:EN2{en2_pk:0}),(d)-[DgoST5]->(st5:
   ST5),(d)-[DgoINT1]->(int1:INT1)
WHERE toUpper(d.d_st1) > 'K' AND int1.int1_valor <> '267'
RETURN d.d_st1, st5.st5_valor, int1.int1_valor
ORDER BY d.d_pk DESC;

// 04 -- H x C
MATCH (dt:DT{dt_valor:'1988/11/22'})<-[r01r:HgoDT]-(hSol:H)-[
   r01n:HgoC_1]->(cSol:C{c_pk:105}),(hSol)-[r01s:HgoC_2]->(
   cSol2:C), (hSol)-[r01h:HgoINT2]->(int2Sol:INT2),
      (dt)<-[r02r:HgoDT]-(hLua:H)-[r02n:HgoC_1]->(cLua:C{c_pk
         :112}),(hLua)-[r02s:HgoC_2]->(cLua2:C), (hLua)-[r02h
         :HgoINT2]->(int2Lua:INT2),
      (dt)<-[r03r:HgoDT]-(hMer:H)-[r03n:HgoC_1]->(cMer:C{c_pk
         :103}),(hMer)-[r03s:HgoC_2]->(cMer2:C), (hMer)-[r03h
         :HgoINT2]->(int2Mer:INT2),
      (dt)<-[r04r:HgoDT]-(hVen:H)-[r04n:HgoC_1]->(cVen:C{c_pk
         :108}),(hVen)-[r04s:HgoC_2]->(cVen2:C), (hVen)-[r04h
         :HgoINT2]->(int2Ven:INT2),
      (dt)<-[r05r:HgoDT]-(hTer:H)-[r05n:HgoC_1]->(cTer:C{c_pk
         :111}),(hTer)-[r05s:HgoC_2]->(cTer2:C), (hTer)-[r05h
         :HgoINT2]->(int2Ter:INT2),
      (dt)<-[r06r:HgoDT]-(hMar:H)-[r06n:HgoC_1]->(cMar:C{c_pk
         :109}),(hMar)-[r06s:HgoC_2]->(cMar2:C), (hMar)-[r06h
         :HgoINT2]->(int2Mar:INT2),
      (dt)<-[r07r:HgoDT]-(hJup:H)-[r07n:HgoC_1]->(cJup:C{c_pk
         :107}),(hJup)-[r07s:HgoC_2]->(cJup2:C), (hJup)-[r07h
         :HgoINT2]->(int2Jup:INT2),
      (dt)<-[r08r:HgoDT]-(hSat:H)-[r08n:HgoC_1]->(cSat:C{c_pk
         :106}),(hSat)-[r08s:HgoC_2]->(cSat2:C), (hSat)-[r08h
         :HgoINT2]->(int2Sat:INT2),
      (dt)<-[r09r:HgoDT]-(hUra:H)-[r09n:HgoC_1]->(cUra:C{c_pk
         :102}),(hUra)-[r09s:HgoC_2]->(cUra2:C), (hUra)-[r09h
```

```
        :HgoINT2]->(int2Ura:INT2),
    (dt)<-[r10r:HgoDT]-(hNet:H)-[r10n:HgoC_1]->(cNet:C{c_pk
        :104}),(hNet)-[r10s:HgoC_2]->(cNet2:C), (hNet)-[r10h
        :HgoINT2]->(int2Net:INT2),
    (dt)<-[r11r:HgoDT]-(hPlu:H)-[r11n:HgoC_1]->(cPlu:C{c_pk
        :101}),(hPlu)-[r11s:HgoC_2]->(cPlu2:C), (hPlu)-[r11h
        :HgoINT2]->(int2Plu:INT2),
    (dt)<-[r12r:HgoDT]-(hEln:H)-[r12n:HgoC_1]->(cEln:C{c_pk
        :110}),(hEln)-[r12s:HgoC_2]->(cEln2:C), (hEln)-[r12h
        :HgoINT2]->(int2Eln:INT2)
RETURN dt.dt_valor AS data,
       int2Sol.int2_valor + '/' + cSol2.c_st1 AS sol,
           int2Lua.int2_valor + '/' + cLua2.c_st1 AS lua,
       int2Mer.int2_valor + '/' + cMer2.c_st1 AS mer,
       int2Ven.int2_valor + '/' + cVen2.c_st1 AS ven,
       int2Ter.int2_valor + '/' + cTer2.c_st1 AS ter,
       int2Mar.int2_valor + '/' + cMar2.c_st1 AS mar,
       int2Jup.int2_valor + '/' + cJup2.c_st1 AS jup,
       int2Sat.int2_valor + '/' + cSat2.c_st1 AS sat,
       int2Ura.int2_valor + '/' + cUra2.c_st1 AS ura,
       int2Net.int2_valor + '/' + cNet2.c_st1 AS net,
       int2Plu.int2_valor + '/' + cPlu2.c_st1 AS plu,
       int2Eln.int2_valor + '/' + cEln2.c_st1 AS eln
UNION MATCH (dt:DT{dt_valor:'1988/11/23'})<-[r01r:HgoDT]-(
   hSol:H)-[r01n:HgoC_1]->(cSol:C{c_pk:105}),(hSol)-[r01s:
   HgoC_2]->(cSol2:C), (hSol)-[r01h:HgoINT2]->(int2Sol:INT2),
    (dt)<-[r02r:HgoDT]-(hLua:H)-[r02n:HgoC_1]->(cLua:C{c_pk
        :112}),(hLua)-[r02s:HgoC_2]->(cLua2:C), (hLua)-[r02h
        :HgoINT2]->(int2Lua:INT2),
    (dt)<-[r03r:HgoDT]-(hMer:H)-[r03n:HgoC_1]->(cMer:C{c_pk
        :103}),(hMer)-[r03s:HgoC_2]->(cMer2:C), (hMer)-[r03h
        :HgoINT2]->(int2Mer:INT2),
    (dt)<-[r04r:HgoDT]-(hVen:H)-[r04n:HgoC_1]->(cVen:C{c_pk
        :108}),(hVen)-[r04s:HgoC_2]->(cVen2:C), (hVen)-[r04h
        :HgoINT2]->(int2Ven:INT2),
```

```
55        (dt)<-[r05r:HgoDT]-(hTer:H)-[r05n:HgoC_1]->(cTer:C{c_pk
             :111}),(hTer)-[r05s:HgoC_2]->(cTer2:C), (hTer)-[r05h
             :HgoINT2]->(int2Ter:INT2),
          (dt)<-[r06r:HgoDT]-(hMar:H)-[r06n:HgoC_1]->(cMar:C{c_pk
             :109}),(hMar)-[r06s:HgoC_2]->(cMar2:C), (hMar)-[r06h
             :HgoINT2]->(int2Mar:INT2),
          (dt)<-[r07r:HgoDT]-(hJup:H)-[r07n:HgoC_1]->(cJup:C{c_pk
             :107}),(hJup)-[r07s:HgoC_2]->(cJup2:C), (hJup)-[r07h
             :HgoINT2]->(int2Jup:INT2),
          (dt)<-[r08r:HgoDT]-(hSat:H)-[r08n:HgoC_1]->(cSat:C{c_pk
             :106}),(hSat)-[r08s:HgoC_2]->(cSat2:C), (hSat)-[r08h
             :HgoINT2]->(int2Sat:INT2),
          (dt)<-[r09r:HgoDT]-(hUra:H)-[r09n:HgoC_1]->(cUra:C{c_pk
             :102}),(hUra)-[r09s:HgoC_2]->(cUra2:C), (hUra)-[r09h
             :HgoINT2]->(int2Ura:INT2),
60        (dt)<-[r10r:HgoDT]-(hNet:H)-[r10n:HgoC_1]->(cNet:C{c_pk
             :104}),(hNet)-[r10s:HgoC_2]->(cNet2:C), (hNet)-[r10h
             :HgoINT2]->(int2Net:INT2),
          (dt)<-[r11r:HgoDT]-(hPlu:H)-[r11n:HgoC_1]->(cPlu:C{c_pk
             :101}),(hPlu)-[r11s:HgoC_2]->(cPlu2:C), (hPlu)-[r11h
             :HgoINT2]->(int2Plu:INT2),
          (dt)<-[r12r:HgoDT]-(hEln:H)-[r12n:HgoC_1]->(cEln:C{c_pk
             :110}),(hEln)-[r12s:HgoC_2]->(cEln2:C), (hEln)-[r12h
             :HgoINT2]->(int2Eln:INT2)
   RETURN dt.dt_valor AS data,
          int2Sol.int2_valor + '/' + cSol2.c_st1 AS sol,
65        int2Lua.int2_valor + '/' + cLua2.c_st1 AS lua,
          int2Mer.int2_valor + '/' + cMer2.c_st1 AS mer,
          int2Ven.int2_valor + '/' + cVen2.c_st1 AS ven,
          int2Ter.int2_valor + '/' + cTer2.c_st1 AS ter,
          int2Mar.int2_valor + '/' + cMar2.c_st1 AS mar,
70        int2Jup.int2_valor + '/' + cJup2.c_st1 AS jup,
          int2Sat.int2_valor + '/' + cSat2.c_st1 AS sat,
          int2Ura.int2_valor + '/' + cUra2.c_st1 AS ura,
            int2Net.int2_valor + '/' + cNet2.c_st1 AS net,
```

```
        int2Plu.int2_valor + '/' + cPlu2.c_st1 AS plu,
        int2Eln.int2_valor + '/' + cEln2.c_st1 AS eln;

// 05 -- Maior A em D
MATCH (a:A)<-[r:DgoA]-(d:D)
WITH a.a_st1 AS a,  COUNT(d) AS qtd_d
RETURN a, qtd_d
ORDER BY qtd_d DESC, a;

// 06 -- F com mais D
MATCH (f:F)<-[GgoF]-()<-[DEgoG]-()-[DEgoE]->(e:E{e_pk:4})
RETURN f.f_st1 AS f, COUNT(1) AS qtd
ORDER BY qtd DESC, f;

// 07 -- Maior de grau 1 A em D
MATCH (a:A)<-[r:DgoA{da_int4:1}]-(d:D)
WITH a.a_st1 AS a,  COUNT(d) AS qtd_d
RETURN a, qtd_d
ORDER BY qtd_d DESC, a;

// 08 -- Maior A X B em  D
MATCH (d1:D)-[r01:DgoDB]->(db:DB)-[r02:DBgoB]-({b_pk:2}),(db)
   -[r03:DBgoD]->(d2:D)
WHERE d1.d_pk < 10000
RETURN d1.d_st1 as d, COLLECT(d2.d_st1) AS bs
ORDER BY SIZE(bs) DESC, d;

// 09 -- G X D
MATCH (f:F)<-[GgoF]-(g:G), (g)-[GgoFL1]->(fl1:FL), (g)-[
   GgoFL2]->(fl2:FL)
WHERE toFloat(fl1.fl_valor) < 0 AND toFloat(fl2.fl_valor) < 0
    AND SUBSTRING(g.g_st1, 0, 1) = 'B'
WITH DISTINCT f, g
MATCH (g)<-[DEgoG]-()-[DEgoE]->(e:E{e_pk:4})
RETURN f.f_st1 AS f, g.g_st1 AS g, COUNT(1) AS qtd
```

```
ORDER BY qtd DESC, f, g
LIMIT 50;

// 10 -- D X D com b_pk 2 - saída em vértices
MATCH (d1:D)-[r01:DgoDB]->(db:DB)-[r02:DBgoB]-({b_pk:2}),(db)
   -[r03:DBgoD]->(d2:D)
WHERE d1.d_pk > 90000 AND d1.d_pk < 90201
RETURN d1, d2;
// 11 -- Maior A de a_pk até 20 de grau 2 X B em  D
MATCH (a:A)<-[r:DgoA{da_int4:2}]-(d:D)-[DgoDB]->()-[DBgoB]->(
   b:B)
WHERE a.a_pk < 21
WITH a.a_st1 AS a,  b.b_st1 AS b, COUNT(d) AS qtd_d
RETURN a, b, qtd_d
ORDER BY qtd_d DESC, a, b;

// 12 -- DC X C X A
MATCH (dc:DC)-[x:DCgoC_1]->(c1:C{c_pk:203}),(dc)-[y:DCgoC_2
   ]->(c2:C{c_pk:206}),
       (dc)<-[DgoDC]-(d:D), (dc)-[DCgoint2]->(int2:INT2), (dc)
          -[DCgoEN4]->(en4:EN4)
WHERE toUpper(SUBSTRING(d.d_st1, 0, 1)) >= 'J' AND toUpper(
   SUBSTRING(d.d_st1, 0, 1)) <= 'R'
RETURN d.d_st1 AS d, int2.int2_valor,  en4.en4_valor
ORDER BY d, en4.en4_valor;

// 13 -- A X D X DE X B
MATCH (dt:DT)
WHERE SUBSTRING(dt.dt_valor, 0, 4) = '1985'
WITH dt
MATCH (dt)<-[DEgoDT]-(de:DE)-[DEgoE]->(e:E{e_pk:4})
WITH SUBSTRING(dt.dt_valor, 0, 4) AS ano, de
MATCH (de)<-[DgoDE]-(d:D)
WITH ano, d
MATCH (d)-[r:DgoA]->(a:A), (d)-[DgoDB]->()-[DBgoB]->(b:B)
```

```
WITH ano, a.a_st1 AS a,  b.b_st1 AS b, COUNT(d) AS qtd_d
RETURN ano, a, b, qtd_d
ORDER BY qtd_d DESC, ano, a, b;

// 14 -- A1 X D X B X A2
MATCH (a:A{a_pk:1871})<-[r1:DgoA]-(d1:D)-[r2:DgoDB]->(db:DB)
   -[r3:DBgoB]->(b:B{b_pk:2}),
      (db)-[r4:DBgoD]->(d2:D)-[r5:DgoA]->({a_pk:1879})
RETURN d1.d_st1, d2.d_st1;

// 15 -- A X B X C X D XE X F X G - saída em relação
MATCH (f1:F)<-[r01:GgoF]-(g1:G),(g1)-[r01a:GgoFL1]->(fl1:FL),
    (g1)-[r01b:GgoFL2]->(fl2:FL)
WHERE toLower(SUBSTRING(f1.f_st1, 0, 1)) = 'c' AND toFloat(
   fl1.fl_valor) < 0 AND toFloat(fl2.fl_valor) < 0
WITH DISTINCT g1
MATCH (g1)<-[r02:DEgoG]-(de1:DE)-[r03:DEgoE]->({e_pk:4}),(de1
   )<-[r04:DgoDE]-(d1:D),
      (de1)-[r04a:DEgoDT]->(dt1:DT)
WHERE dt1.dt_valor < '2000/01/01'
WITH d1
MATCH (d1)-[r05:DgoDB]->(db:DB)-[r06:DBgoB]-({b_pk:2}), (db)
   -[r07:DBgoD]-(d2:D),
      (d2)-[r08:DgoDE]->(de:DE)-[r09:DEgoE]->({e_pk:4}),
      (de)-[r09b:DEgost1]->(st1:st1)
WHERE st1.st1_valor < '12:00'
WITH d1, COUNT(d2) AS qtd_d2
WHERE qtd_d2 > 2
OPTIONAL MATCH (d1)-[r10:DgoA]-(a:A)
WHERE toLower(SUBSTRING(a.a_st1, 0, 1)) = 'm'
WITH d1, qtd_d2, COUNT(a) AS qtd_a
WITH d1, qtd_d2, qtd_a
MATCH (d1)-[r11:DgoDC]->(dc:DC)-[r12:DCgoC_1]->({c_pk:203}),(
   dc)-[r13:DCgoC_2]->(c:C),
```

```
        (dc)-[r13b:DCgoEN4]->(en4:EN4),(dc)-[r13c:DCgoINT2]->(
           int2:INT2)
WHERE en4.en4_valor = '+'
  AND int2.int2_valor > '1500'
RETURN d1.d_st1, c.c_st1, int2.int2_valor, en4.en4_valor,
   qtd_d2, qtd_a
ORDER BY int2.int2_valor;

// 16 -- A X B X C X D XE X F X G - saída em vértices
MATCH (f1:F)<-[r01:GgoF]-(g1:G),(g1)-[r01a:GgoFL1]->(fl1:FL),
    (g1)-[r01b:GgoFL2]->(fl2:FL)
WHERE toLower(SUBSTRING(f1.f_st1, 0, 1)) = 'c' AND toFloat(
   fl1.fl_valor) < 0 AND toFloat(fl2.fl_valor) < 0
WITH DISTINCT g1
MATCH (g1)<-[r02:DEgoG]-(de1:DE)-[r03:DEgoE]->({e_pk:4}),(de1
   )<-[r04:DgoDE]-(d1:D),
        (de1)-[r04a:DEgoDT]->(dt1:DT)
WHERE dt1.dt_valor < '2000/01/01'
WITH d1
MATCH (d1)-[r05:DgoDB]->(db:DB)-[r06:DBgoB]-({b_pk:2}), (db)
   -[r07:DBgoD]-(d2:D),
        (d2)-[r08:DgoDE]->(de:DE)-[r09:DEgoE]->({e_pk:4}),
        (de)-[r09b:DEgost1]->(st1:st1)
WHERE st1.st1_valor < '12:00'
WITH d1, COUNT(d2) AS qtd_d2
WHERE qtd_d2 > 2
OPTIONAL MATCH (d1)-[r10:DgoA]-(a:A)
WHERE toLower(SUBSTRING(a.a_st1, 0, 1)) = 'm'
WITH d1, qtd_d2, COUNT(a) AS qtd_a
WITH d1, qtd_d2, qtd_a
MATCH (d1)-[r11:DgoDC]->(dc:DC)-[r12:DCgoC_1]->({c_pk:203}),(
   dc)-[r13:DCgoC_2]->(c:C),
        (dc)-[r13b:DCgoEN4]->(en4:EN4),(dc)-[r13c:DCgoINT2]->(
           int2:INT2)
WHERE en4.en4_valor = '+'
```

```
  AND int2.int2_valor > '1500'
RETURN d1, c, dc;

// 17 -- Caminho Mínimo entre D=1 e D=2
MATCH (d:D{d_pk:1}), (to:D{d_pk:2})
MATCH p = shortestPath( (d)-[r1:DgoDB|:DBgoD*]->(to) )
RETURN d, r1, p;

// 18 -- Todos Caminhos Mínimo entre D=1 e D=2
MATCH (d:D{d_pk:1}), (to:D{d_pk:2})
MATCH p = allShortestPaths( (d)-[r1:DgoDB|:DBgoD*]->(to) )
RETURN d, r1, p;
```

B.3.3 Consultas Cypher - Método M04

Listagem B.3: Scripts das Consultas nas Bases Neo4j - Método M04

```
// 01 -- Listar st4 de todos os D com st4 entre 'A' e 'J'
MATCH (d:D)
WHERE toUpper(d.d_st1) < 'K'
RETURN d.d_st1
ORDER BY d.d_st1
LIMIT 1000000;

// 02 -- D X DC X C
MATCH (d:D)-[x:DgoDC]->(dc:DC)
WHERE toUpper(SUBSTRING(d.d_st1, 0, 1)) = 'A' OR toUpper(
   SUBSTRING(d.d_st1, 0, 1)) = 'L'
WITH d.d_st1 AS d, dc.cn_st1 AS c, COUNT(1) AS qtd
WHERE qtd > 1
RETURN d, c, qtd
ORDER BY qtd DESC, d;

// 03 -- Listar st4, st5, int1 de todos os D com st4
//       entre 'K' E 'Z' e com en2 = 3
//       em ordem descrescente de pk
```

```
MATCH (d:D)
WHERE toUpper(d.d_st1) > 'K' AND d.d_en2 = 0 AND d.d_int1 <>
   267
RETURN d.d_st1, d.d_st2, d.d_int1
ORDER BY d.d_pk DESC;

// 04 -- H x C
MATCH (hSol:H{h_dt1:'1988/11/22',cn_pk:105}),
      (hLua:H{h_dt1:'1988/11/22',cn_pk:112}),
      (hMer:H{h_dt1:'1988/11/22',cn_pk:103}),
      (hVen:H{h_dt1:'1988/11/22',cn_pk:108}),
      (hTer:H{h_dt1:'1988/11/22',cn_pk:111}),
      (hMar:H{h_dt1:'1988/11/22',cn_pk:109}),
      (hJup:H{h_dt1:'1988/11/22',cn_pk:107}),
      (hSat:H{h_dt1:'1988/11/22',cn_pk:106}),
      (hUra:H{h_dt1:'1988/11/22',cn_pk:102}),
      (hNet:H{h_dt1:'1988/11/22',cn_pk:104}),
      (hPlu:H{h_dt1:'1988/11/22',cn_pk:101}),
      (hEln:H{h_dt1:'1988/11/22',cn_pk:110})
RETURN hSol.h_dt1 AS data,
       hSol.h_int2 + '/' + hSol.cs_st1 AS sol,
       hLua.h_int2 + '/' + hLua.cs_st1 AS lua,
       hMer.h_int2 + '/' + hMer.cs_st1 AS mer,
       hVen.h_int2 + '/' + hVen.cs_st1 AS ven,
       hTer.h_int2 + '/' + hTer.cs_st1 AS ter,
       hMar.h_int2 + '/' + hMar.cs_st1 AS mar,
       hJup.h_int2 + '/' + hJup.cs_st1 AS jup,
       hSat.h_int2 + '/' + hSat.cs_st1 AS sat,
       hUra.h_int2 + '/' + hUra.cs_st1 AS ura,
       hNet.h_int2 + '/' + hNet.cs_st1 AS net,
       hPlu.h_int2 + '/' + hPlu.cs_st1 AS plu,
       hEln.h_int2 + '/' + hEln.cs_st1 AS eln
UNION MATCH (hSol:H{h_dt1:'1988/11/23',cn_pk:105}),
      (hLua:H{h_dt1:'1988/11/23',cn_pk:112}),
      (hMer:H{h_dt1:'1988/11/23',cn_pk:103}),
```

```
        (hVen:H{h_dt1:'1988/11/23',cn_pk:108}),
        (hTer:H{h_dt1:'1988/11/23',cn_pk:111}),
        (hMar:H{h_dt1:'1988/11/23',cn_pk:109}),
        (hJup:H{h_dt1:'1988/11/23',cn_pk:107}),
        (hSat:H{h_dt1:'1988/11/23',cn_pk:106}),
        (hUra:H{h_dt1:'1988/11/23',cn_pk:102}),
        (hNet:H{h_dt1:'1988/11/23',cn_pk:104}),
        (hPlu:H{h_dt1:'1988/11/23',cn_pk:101}),
        (hEln:H{h_dt1:'1988/11/23',cn_pk:110})
RETURN hSol.h_dt1 AS data,
       hSol.h_int2 + '/' + hSol.cs_st1 AS sol,
       hLua.h_int2 + '/' + hLua.cs_st1 AS lua,
       hMer.h_int2 + '/' + hMer.cs_st1 AS mer,
       hVen.h_int2 + '/' + hVen.cs_st1 AS ven,
       hTer.h_int2 + '/' + hTer.cs_st1 AS ter,
       hMar.h_int2 + '/' + hMar.cs_st1 AS mar,
       hJup.h_int2 + '/' + hJup.cs_st1 AS jup,
       hSat.h_int2 + '/' + hSat.cs_st1 AS sat,
       hUra.h_int2 + '/' + hUra.cs_st1 AS ura,
       hNet.h_int2 + '/' + hNet.cs_st1 AS net,
       hPlu.h_int2 + '/' + hPlu.cs_st1 AS plu,
       hEln.h_int2 + '/' + hEln.cs_st1 AS eln;

// 05 -- Maior A em D
MATCH (a:A)<-[r:DgoA]-(d:D)
WITH a.a_st1 AS a,  COUNT(d) AS qtd_d
RETURN a, qtd_d
ORDER BY qtd_d DESC, a;

// 06 -- F com mais D
MATCH (de:DE{e_pk:4})
RETURN de.f_st1 AS f, COUNT(1) AS qtd
ORDER BY qtd DESC, f;

// 07 -- Maior de grau 1 A em D
```

```
MATCH (a:A)<-[r:DgoA{da_int4:1}]-(d:D)
WITH a.a_st1 AS a,  COUNT(d) AS qtd_d
RETURN a, qtd_d
ORDER BY qtd_d DESC, a;

// 08 -- D X B para d_pk < 10000 e b_pk = 2
MATCH (d1:D)-[r01:DgoDB]->(db:DB{b_pk:2})-[r02:DBgoD]->(d2:D)
WHERE d1.d_pk < 10000
RETURN d1.d_st1 as d, COLLECT(d2.d_st1) AS bs
ORDER BY SIZE(bs) DESC, d;

// 09 -- G X D
MATCH (de:DE{e_pk:4})
WHERE de.g_fl1 < 0 AND de.g_fl2 < 0 AND SUBSTRING(de.g_st1,
   0, 1) = 'B'
RETURN de.f_st1 AS f, de.g_st1 AS g, COUNT(1) AS qtd
ORDER BY qtd DESC, f, g
LIMIT 50;

// 10 -- D X D com b_pk 2 - saída em vértices
MATCH (d1:D)-[r01:DgoDB]->(db:DB{b_pk:2})-[r02:DBgoD]->(d2:D)
WHERE d1.d_pk > 90000 AND d1.d_pk < 90201
RETURN d1, d2;

// 11 -- Maior A X B em  D
MATCH (a:A)<-[r:DgoA{da_int4:2}]-(d:D)-[DgoDB]->(db:DB)
WHERE a.a_pk < 21
WITH a.a_st1 AS a,  db.b_st1 AS b, COUNT(d) AS qtd_d
RETURN a, b, qtd_d
ORDER BY qtd_d DESC, a, b;

// 12 - DC X C X A
MATCH (dc:DC{cn_pk:203, cs_pk:206})<-[DgoDC]-(d:D)
WHERE toUpper(SUBSTRING(d.d_st1, 0, 1)) >= 'J' AND toUpper(
   SUBSTRING(d.d_st1, 0, 1)) <= 'R'
```

```
RETURN d.d_st1 AS d, dc.dc_int2,  dc.dc_en4
ORDER BY d, dc.dc_int3;

// 13 -- A X D X E X DE X B
MATCH (de:DE{e_pk:4})<-[DgoDE]-(d:D)
WHERE SUBSTRING(de.de_dt2, 0, 4) = '1985'
WITH SUBSTRING(de.de_dt2, 0, 4) AS ano, d
MATCH (d)-[r:DgoA]->(a:A), (d)-[DgoDB]->(db:DB)
WITH ano, a.a_st1 AS a,  db.b_st1 AS b, COUNT(d) AS qtd_d
RETURN ano, a, b, qtd_d
ORDER BY qtd_d DESC, ano, a, b;

// 14 -- A1 X D X B X A2
MATCH (a:A{a_pk:1871})<-[r1:DgoA]-(d1:D)-[r2:DgoDB]->(db:DB{
   b_pk:2}),
      (db)-[r4:DBgoD]->(d2:D)-[r5:DgoA]->({a_pk:1879})
RETURN d1.d_st1, d2.d_st1;

// 15 -- A X B X C X D XE X F X G - saída em relação
MATCH (de1:DE{e_pk:4})<-[r04:DgoDE]-(d1:D)
WHERE de1.de_dt2 < '2000/01/01'
  AND toLower(SUBSTRING(de1.f_st1, 0, 1)) = 'c' AND de1.g_fl1
      < 0 AND de1.g_fl2 < 0
WITH d1
MATCH (d1)-[r05:DgoDB]->(db:DB{b_pk:2})-[r07:DBgoD]-(d2:D)-[
   r08:DgoDE]->(de:DE{e_pk:4}),
WHERE de.de_st1 < '12:00'
WITH d1, COUNT(d2) AS qtd_d2
WHERE qtd_d2 > 2
OPTIONAL MATCH (d1)-[r10:DgoA]-(a:A)
WHERE toLower(SUBSTRING(a.a_st1, 0, 1)) = 'm'
WITH d1, qtd_d2, COUNT(a) AS qtd_a
WITH d1, qtd_d2, qtd_a
MATCH (d1)-[r11:DgoDC]->(dc:DC{cn_pk:203})
WHERE dc.dc_en4 = '+'
```

```
  AND dc.dc_int2 > '1500'
RETURN d1.d_st1, dc.cs_st1, dc.dc_int2, dc.dc_en4, qtd_d2,
   qtd_a
ORDER BY dc.dc_int2;

// 16 -- A X B X C X D XE X F X G - saída em vértices
MATCH (de1:DE{e_pk:4})<-[r04:DgoDE]-(d1:D)
WHERE de1.de_dt2 < '2000/01/01'
  AND toLower(SUBSTRING(de1.f_st1, 0, 1)) = 'c' AND de1.g_fl1
      < 0 AND de1.g_fl2 < 0
WITH d1
MATCH (d1)-[r05:DgoDB]->(db:DB{b_pk:2})-[r07:DBgoD]-(d2:D)-[
   r08:DgoDE]->(de:DE{e_pk:4}),
WHERE de.de_st1 < '12:00'
WITH d1, COUNT(d2) AS qtd_d2
WHERE qtd_d2 > 2
OPTIONAL MATCH (d1)-[r10:DgoA]-(a:A)
WHERE toLower(SUBSTRING(a.a_st1, 0, 1)) = 'm'
WITH d1, qtd_d2, COUNT(a) AS qtd_a
WITH d1, qtd_d2, qtd_a
MATCH (d1)-[r11:DgoDC]->(dc:DC{cn_pk:203})
WHERE dc.dc_en4 = '+'
  AND dc.dc_int2 > '1500'
RETURN d1,  dc;

// 17 -- Caminho Mínimo entre D=1 e D=2
MATCH p = shortestPath( (from:D) - [r1:DgoDB|:DBgoD*] -> (to:
   D) )
WHERE (from.d_pk = 1)
  AND (to.d_pk = 2)
RETURN p;

// 18 -- Todos Caminhos Mínimo entre D=1 e D=2
MATCH p = allShortestPaths( (from:D) - [r1:DgoDB|:DBgoD*] ->
   (to:D) )
```

```
WHERE (from.d_pk = 1)
  AND (to.d_pk = 2)
RETURN p;
```

B.3.4 Consultas Cypher - Método M05

Listagem B.4: Scripts das Consultas nas Bases Neo4j - Método M05

```
// 01 -- Listar st4 de todos os D com st4 entre 'A' e 'J'
MATCH (d:D)
WHERE toUpper(d.d_st1) < 'K'
RETURN d.d_st1
ORDER BY d.d_st1
LIMIT 1000000;

// 02 -- D X DC X C-contador
MATCH (d:D)-[x:DgoDC]->(dc:DC),
      (dc)-[y:DCgoC_1]->(c1:C)
WHERE toUpper(SUBSTRING(d.d_st1, 0, 1)) = 'A' OR toUpper(
   SUBSTRING(d.d_st1, 0, 1)) = 'L'
WITH d.d_st1 AS d, c1.c_st1 AS c, COUNT(1) AS qtd
WHERE qtd > 1
RETURN d, c, qtd
ORDER BY qtd DESC, d;

// 03 -- Listar st4, st5, int1 de todos os D com st4
//       entre 'K' E 'Z' e com en2 = 3
//       em ordem descrescente de pk
MATCH (d:D)-[DgoDO]->(do:DO)
WHERE toUpper(d.d_st1) > 'K' AND do.d_en2 = 0 AND do.d_int1
   <> 267
RETURN d.d_st1, do.d_st2, do.d_int1
ORDER BY d.d_pk DESC;

// 04 -- H x C
```

```
MATCH (hSol:H{h_dt1:'1988/11/22'})-[r01n:HgoC_1]->(cSol:C{
   c_pk:105}),(hSol)-[r01s:HgoC_2]->(cSol2:C),
      (hLua:H{h_dt1:'1988/11/22'})-[r02n:HgoC_1]->(cLua:C{
         c_pk:112}),(hLua)-[r02s:HgoC_2]->(cLua2:C),
      (hMer:H{h_dt1:'1988/11/22'})-[r03n:HgoC_1]->(cMer:C{
         c_pk:103}),(hMer)-[r03s:HgoC_2]->(cMer2:C),
      (hVen:H{h_dt1:'1988/11/22'})-[r04n:HgoC_1]->(cVen:C{
         c_pk:108}),(hVen)-[r04s:HgoC_2]->(cVen2:C),
      (hTer:H{h_dt1:'1988/11/22'})-[r05n:HgoC_1]->(cTer:C{
         c_pk:111}),(hTer)-[r05s:HgoC_2]->(cTer2:C),
      (hMar:H{h_dt1:'1988/11/22'})-[r06n:HgoC_1]->(cMar:C{
         c_pk:109}),(hMar)-[r06s:HgoC_2]->(cMar2:C),
      (hJup:H{h_dt1:'1988/11/22'})-[r07n:HgoC_1]->(cJup:C{
         c_pk:107}),(hJup)-[r07s:HgoC_2]->(cJup2:C),
      (hSat:H{h_dt1:'1988/11/22'})-[r08n:HgoC_1]->(cSat:C{
         c_pk:106}),(hSat)-[r08s:HgoC_2]->(cSat2:C),
      (hUra:H{h_dt1:'1988/11/22'})-[r09n:HgoC_1]->(cUra:C{
         c_pk:102}),(hUra)-[r09s:HgoC_2]->(cUra2:C),
      (hNet:H{h_dt1:'1988/11/22'})-[r10n:HgoC_1]->(cNet:C{
         c_pk:104}),(hNet)-[r10s:HgoC_2]->(cNet2:C),
      (hPlu:H{h_dt1:'1988/11/22'})-[r11n:HgoC_1]->(cPlu:C{
         c_pk:101}),(hPlu)-[r11s:HgoC_2]->(cPlu2:C),
      (hEln:H{h_dt1:'1988/11/22'})-[r12n:HgoC_1]->(cEln:C{
         c_pk:110}),(hEln)-[r12s:HgoC_2]->(cEln2:C)
RETURN hSol.h_dt1 AS data,
       hSol.h_int2 + '/' + cSol2.c_st1 AS sol,
       hLua.h_int2 + '/' + cLua2.c_st1 AS lua,
       hMer.h_int2 + '/' + cMer2.c_st1 AS mer,
       hVen.h_int2 + '/' + cVen2.c_st1 AS ven,
       hTer.h_int2 + '/' + cTer2.c_st1 AS ter,
       hMar.h_int2 + '/' + cMar2.c_st1 AS mar,
       hJup.h_int2 + '/' + cJup2.c_st1 AS jup,
       hSat.h_int2 + '/' + cSat2.c_st1 AS sat,
       hUra.h_int2 + '/' + cUra2.c_st1 AS ura,
       hNet.h_int2 + '/' + cNet2.c_st1 AS net,
```

```
        hPlu.h_int2 + '/' + cPlu2.c_st1 AS plu,
        hEln.h_int2 + '/' + cEln2.c_st1 AS eln
UNION MATCH (hSol:H{h_dt1:'1988/11/23'})-[r01n:HgoC_1]->(cSol
   :C{c_pk:105}),(hSol)-[r01s:HgoC_2]->(cSol2:C),
      (hLua:H{h_dt1:'1988/11/23'})-[r02n:HgoC_1]->(cLua:C{
         c_pk:112}),(hLua)-[r02s:HgoC_2]->(cLua2:C),
      (hMer:H{h_dt1:'1988/11/23'})-[r03n:HgoC_1]->(cMer:C{
         c_pk:103}),(hMer)-[r03s:HgoC_2]->(cMer2:C),
      (hVen:H{h_dt1:'1988/11/23'})-[r04n:HgoC_1]->(cVen:C{
         c_pk:108}),(hVen)-[r04s:HgoC_2]->(cVen2:C),
      (hTer:H{h_dt1:'1988/11/23'})-[r05n:HgoC_1]->(cTer:C{
         c_pk:111}),(hTer)-[r05s:HgoC_2]->(cTer2:C),
      (hMar:H{h_dt1:'1988/11/23'})-[r06n:HgoC_1]->(cMar:C{
         c_pk:109}),(hMar)-[r06s:HgoC_2]->(cMar2:C),
      (hJup:H{h_dt1:'1988/11/23'})-[r07n:HgoC_1]->(cJup:C{
         c_pk:107}),(hJup)-[r07s:HgoC_2]->(cJup2:C),
      (hSat:H{h_dt1:'1988/11/23'})-[r08n:HgoC_1]->(cSat:C{
         c_pk:106}),(hSat)-[r08s:HgoC_2]->(cSat2:C),
      (hUra:H{h_dt1:'1988/11/23'})-[r09n:HgoC_1]->(cUra:C{
         c_pk:102}),(hUra)-[r09s:HgoC_2]->(cUra2:C),
      (hNet:H{h_dt1:'1988/11/23'})-[r10n:HgoC_1]->(cNet:C{
         c_pk:104}),(hNet)-[r10s:HgoC_2]->(cNet2:C),
      (hPlu:H{h_dt1:'1988/11/23'})-[r11n:HgoC_1]->(cPlu:C{
         c_pk:101}),(hPlu)-[r11s:HgoC_2]->(cPlu2:C),
      (hEln:H{h_dt1:'1988/11/23'})-[r12n:HgoC_1]->(cEln:C{
         c_pk:110}),(hEln)-[r12s:HgoC_2]->(cEln2:C)
RETURN hSol.h_dt1 AS data,
        hSol.h_int2 + '/' + cSol2.c_st1 AS sol,
        hLua.h_int2 + '/' + cLua2.c_st1 AS lua,
        hMer.h_int2 + '/' + cMer2.c_st1 AS mer,
        hVen.h_int2 + '/' + cVen2.c_st1 AS ven,
        hTer.h_int2 + '/' + cTer2.c_st1 AS ter,
        hMar.h_int2 + '/' + cMar2.c_st1 AS mar,
        hJup.h_int2 + '/' + cJup2.c_st1 AS jup,
        hSat.h_int2 + '/' + cSat2.c_st1 AS sat,
```

```
        hUra.h_int2 + '/' + cUra2.c_st1 AS ura,
        hNet.h_int2 + '/' + cNet2.c_st1 AS net,
        hPlu.h_int2 + '/' + cPlu2.c_st1 AS plu,
        hEln.h_int2 + '/' + cEln2.c_st1 AS eln;

// 05 -- Maior A em D
MATCH (a:A)<-[r:DgoA]-(d:D)
WITH a.a_st1 AS a,  COUNT(d) AS qtd_d
RETURN a, qtd_d
ORDER BY qtd_d DESC, a;

// 06 -- F com mais D
MATCH ()-[r02:DgoG{e_pk:4}]->()-[r01:GgoF]->(f:F)
RETURN f.f_st1 AS f, COUNT(1) AS qtd
ORDER BY qtd DESC, f;

// 07 -- Maior de grau 1 A em D
MATCH (a:A)<-[r:DgoA{da_int4:1}]-(d:D)
WITH a.a_st1 AS a,  COUNT(d) AS qtd_d
RETURN a, qtd_d
ORDER BY qtd_d DESC, a;

// 08 -- D X B para d_pk < 10000 e b_pk = 2
MATCH (d1:D)-[r01:DgoD{b_pk:2}]->(d2:D)
WHERE d1.d_pk < 10000
RETURN d1.d_st1 as d, COLLECT(d2.d_st1) AS bs
ORDER BY SIZE(bs) DESC, d;

// 09 -- G X D
MATCH ()-[r01:DgoG{e_pk:4}]->(g:G)-[r02:GgoF]->(f:F)
WHERE g.g_fl1 < 0 AND g.g_fl2 < 0 AND SUBSTRING(g.g_st1, 0,
   1) = 'B'
RETURN f.f_st1 AS f, g.g_st1 AS g, COUNT(1) AS qtd
ORDER BY qtd DESC, f, g
LIMIT 50;
```

```
// 10 -- D X D com b_pk 2-saída em vértices
MATCH (d1:D)-[r01:DgoD{b_pk:2}]->(d2:D)
WHERE d1.d_pk > 90000 AND d1.d_pk < 90201
RETURN d1, d2;

// 11 -- Maior A X B em  D
MATCH ()<-[r2:DgoD]-(d:D)-[r1:DgoA{da_int4:2}]->(a:A)
WHERE a.a_pk < 21
WITH a.a_st1 AS a,  r2.b_st1 AS b, COUNT(d) AS qtd_d
RETURN a, b, qtd_d
ORDER BY qtd_d DESC, a, b;

// 12 -- DC X C X A
MATCH (dc:DC)-[x:DCgoC_1]->(c1:C{c_pk:203}),(dc)-[y:DCgoC_2
   ]->(c2:C{c_pk:206}),(dc)<-[DgoDC]-(d:D)
WHERE toUpper(SUBSTRING(d.d_st1, 0, 1)) >= 'J' AND toUpper(
   SUBSTRING(d.d_st1, 0, 1)) <= 'R'
RETURN d.d_st1 AS d, dc.dc_int3,  dc.dc_en4
ORDER BY d, dc.dc_int3;

// 13 -- A X D X DE X E X B
MATCH (d:D)-[r01:DgoG{e_pk:4}]-()
WHERE SUBSTRING(r01.de_dt2, 0, 4) = '1985'
WITH SUBSTRING(r01.de_dt2, 0, 4) AS ano, d
MATCH (d)-[r:DgoA]->(a:A), (d)-[r02:DgoD]->()
WITH ano, a.a_st1 AS a,  r02.b_st1 AS b, COUNT(d) AS qtd_d
RETURN ano, a, b, qtd_d
ORDER BY qtd_d DESC, ano, a, b;

// 14 -- A1 X D X B X A2
MATCH (a:A{a_pk:1871})<-[r01:DgoA]-(d1:D)-[r2:DgoD{b_pk
   :2}]->(d2:D)-[r5:DgoA]->({a_pk:1879})
RETURN d1.d_st1, d2.d_st1;
```

```
// 15 -- A X B X C X D XE X F X G-saída em relação
MATCH (f1:F)<-[r01:GgoF]-(g1:G)
WHERE toLower(SUBSTRING(f1.f_st1, 0, 1)) = 'c' AND g1.g_fl1 <
    0 AND g1.g_fl2 < 0
WITH g1
MATCH (g1)<-[r02:DgoG{e_pk:4}]-(d1:D)
WHERE r02.de_dt2 < '2000/01/01'
WITH d1
MATCH (d1)-[r03:DgoD{b_pk:2}]->(d2:D),
      (d2)-[r04:DgoG{e_pk:4}]->()
WHERE r04.de_st1 < '12:00'
WITH d1, COUNT(d2) AS qtd_d2
WHERE qtd_d2 > 2
OPTIONAL MATCH (d1)-[r10:DgoA]->(a:A)
WHERE toLower(SUBSTRING(a.a_st1, 0, 1)) = 'm'
WITH d1, qtd_d2, COUNT(a) AS qtd_a
WITH d1, qtd_d2, qtd_a
MATCH (d1)-[r11:DgoDC]->(dc:DC)-[r12:DCgoC_1]->({c_pk:203}),(
   dc)-[r13:DCgoC_2]->(c:C)
WHERE dc.dc_en4 = '+'
  AND dc.dc_int2 > '1500'
RETURN d1.d_st1, c.c_st1, dc.dc_int2, dc.dc_en4, qtd_d2,
   qtd_a
ORDER BY dc.dc_int2;

// 16 -- A X B X C X D XE X F X G-saída em vértices
MATCH (f1:F)<-[r01:GgoF]-(g1:G)
WHERE toLower(SUBSTRING(f1.f_st1, 0, 1)) = 'c' AND g1.g_fl1 <
    0 AND g1.g_fl2 < 0
WITH g1
MATCH (g1)<-[r02:DgoG{e_pk:4}]-(d1:D)
WHERE r02.de_dt2 < '2000/01/01'
WITH d1
MATCH (d1)-[r03:DgoD{b_pk:2}]->(d2:D),
      (d2)-[r04:DgoG{e_pk:4}]->()
```

```
WHERE r04.de_st1 < '12:00'
WITH d1, COUNT(d2) AS qtd_d2
WHERE qtd_d2 > 2
OPTIONAL MATCH (d1)-[r10:DgoA]->(a:A)
WHERE toLower(SUBSTRING(a.a_st1, 0, 1)) = 'm'
WITH d1, qtd_d2, COUNT(a) AS qtd_a
WITH d1, qtd_d2, qtd_a
MATCH (d1)-[r11:DgoDC]->(dc:DC)-[r12:DCgoC_1]->({c_pk:203}),(
   dc)-[r13:DCgoC_2]->(c:C)
WHERE dc.dc_en4 = '+'
  AND dc.dc_int2 > '1500'
RETURN d1, c, dc;

// 17 -- Caminho Mínimo entre D=1 e D=2
MATCH p = shortestPath( (from:D)-[:DgoD*] -> (to:D) )
WHERE (from.d_pk = 1)
  AND (to.d_pk = 2)
RETURN p;

// 18 -- Todos Caminhos Mínimo entre D=1 e D=2
MATCH p = allShortestPaths( (from:D)-[:DgoD*] -> (to:D) )
WHERE (from.d_pk = 1)
  AND (to.d_pk = 2)
RETURN p;
```

APÊNDICE C – Uma Outra Base - Filmes

Uma segunda base de testes foi modelada e implementada com os métodos analisados e proposto nesta dissertação.

Para tal, foi definida uma base de filmes, pessoas que atuaram nestes filmes e eventuais indicações a prêmios ligados ao cinema, como o Oscar, prêmio do cinema norte-americano, César, premiação do cinema francês, etc.

Neste apêndice são apresentados os modelos entidade relacionamento e relacional desta base, depois esta base é modelada conforme os métodos $M01$ - Modelagem 3NF, $M03$ - Grafos Simples, $M04$ - Dirigida e o método proposto nesta dissertação ($M05$ - Participativa). Num terceiro momento, experimentos foram realizados nestas diferentes modelagens e o seu resultado analisado. Como foi visto que a base resultante dos métodos $M01$ (Modelagem 3NF) e $M02$ (*Reference Graph*) é sempre idêntico, a modelagem pelo método $M02$ não foi colocada neste apêndice.

C.1 A Base Filmes

A base de dados objeto deste apêndice é a $BaseFilmes$. Para eventuais comparações com esta nova base, nesta seção a base tratada nos capítulos 2, 3 e 4 será chamada de $BaseTeste$.

A primeira entidade da $BaseFilmes$ é a entidade `Pessoa`. Esta `pessoa` tem um *nome*, uma *data* de nascimento e um *sexo*, que pode não ser informado, mas, se informado, será feminino ou masculino.

Esta pessoa tem `Atuação` em `Filmes`. Ela pode ter atuado como Ator, Atriz, Diretor, Roteirista, Produtor ou Músico num `Filme`, inclusive com atuações diferentes em um mesmo `Filme`.

Um `Filme` tem um `nome`, um `ano` de lançamento e uma `duração` em minutos. O

filme pode ser classificado em diferentes `Categorias`, com *nome* como Drama, Comédia, Documentário, etc. O `Filme` é produzido em um `País`, que tem um *nome*.

Anualmente, os filmes e as pessoas podem ser indicados para diferentes `Prêmios` ligados ao cinema. Cada `Prêmio` tem um `nome` e diferentes categorias de premiação. As diferentes premiações podem ser para o `Filme` ou para uma `Pessoa` pela atuação no `Filme`, mas uma mesma premiação não pode ser atribuída a um filme e a uma pessoa. Por exemplo, o prêmio de $MelhorFilme$ é atribuído apenas a filmes, e o de $MelhorDiretor$ apenas a pessoas pela atuação em um filme.

A figura C.1 traz o modelo entidade relacionamento da $BaseFilmes$ com as entidades $Pessoa$, $Filme$, $Funcao$, $Categoria$, $Pais$, $Premio$ e $TipoPremio$ e os seguintes relacionamentos entre estas entidades:

- de grau 2:
  - $FilmeCategoria$, do tipo $m : n$, entre as entidades $Filme$ e $Categoria$;
  - $FilmePais$, do tipo $1 : 1$, entre as entidades $Filme$ e $Pais$;
- de grau 3:
  - $Atuou$, entre as entidades $Pessoa$, $Filme$ e $Funcao$;
  - $Indicado$, entre as entidades $Filme$, $Premio$ e $TipoPremio$;
- de grau 4:
  - $Indicada$, entre as entidades $Pessoa$, $Filme$, $Premio$ e $TipoPremio$;

A partir do modelo entidade relacionamento foi gerado o modelo relacional da base, conforme mostra a figura C.2. Nele, as entidades do modelo deram origem às *tables* $Pessoa$, $Filme$, $Funcao$, $Categoria$, $Pais$, $Premio$ e $TipoPremio$. O relacionamento FilmePais, de cardinalidade $1 : 1$, deu origem ao atributo chave estrangeira *pai_id* na tabela $Filme$. Os demais relacionamentos deram origem às *joined tables* $FilmeCategoria$, $Atuacao$, $Indicado$ e $Indicada$.

C.2 Modelando a $BaseFilmes$ para Grafos

Nesta seção são detalhados os passos de modelagem da $BaseFilmes$ para um modelo de banco de dados voltado a grafos conforme os métodos de modelagem analisados nesta dissertação.

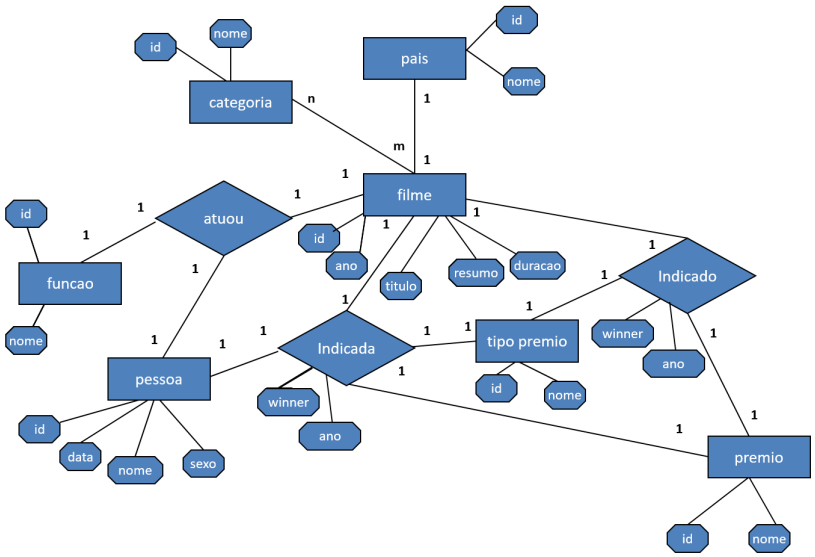


Figura C.1: BaseFilmes - Modelo Entidade Relacionamento

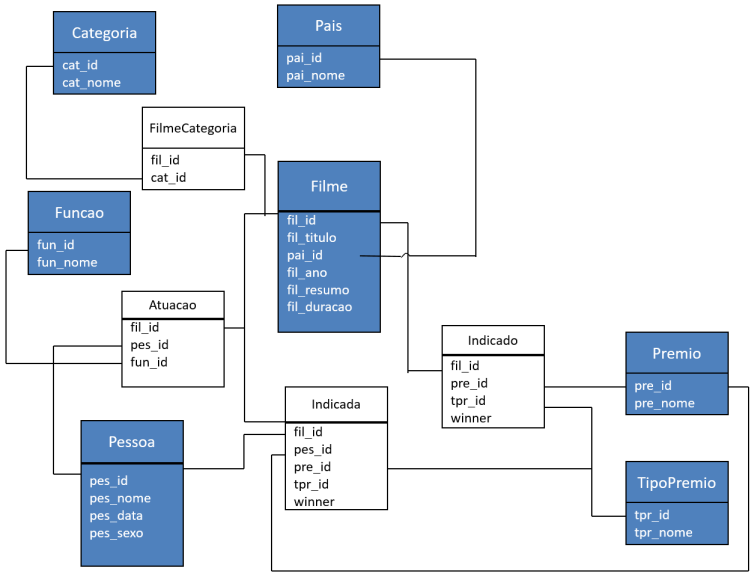


Figura C.2: BaseFilmes - Modelo Entidade Relacionamento Sem Atributos

C.2.1 Modelando a *BaseFilmes* conforme o método *M*01 - Modelagem 3NF

Como visto, este método parte do modelo relacional (figura C.2). No primeiro passo, as *tables* são convertidas nos vértices *Pessoa*, *Filme*, *Funcao*, *Categoria*, *Pais*, *Premio* e *TipoPremio* e o atributo chave estrangeira *pai_id*, do vértice *Filme*, dá origem à aresta *filmeTOpais*. A figura C.3 traz o modelo após o passo 1.

No passo seguinte, a *joined table* binária *FilmeCategoria* é substituída pela aresta *filmeTOcategoria* (figura C.4).

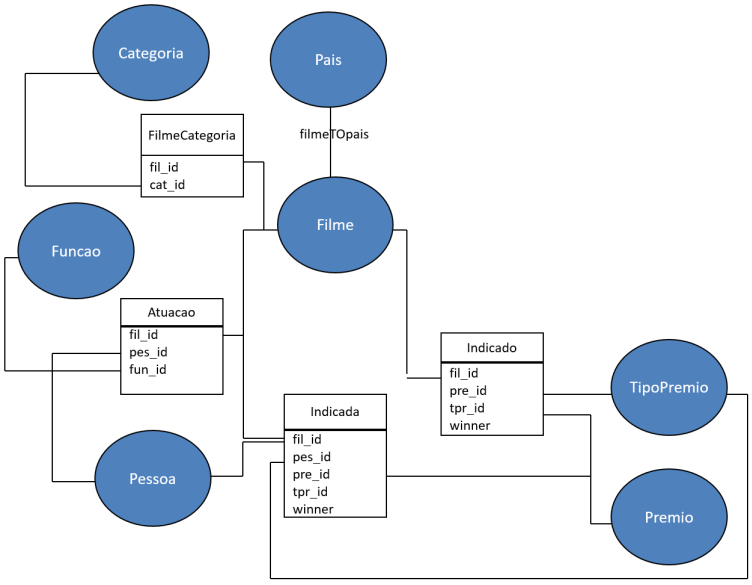


Figura C.3: BaseFilmes - Método $M01$ - Passo 1

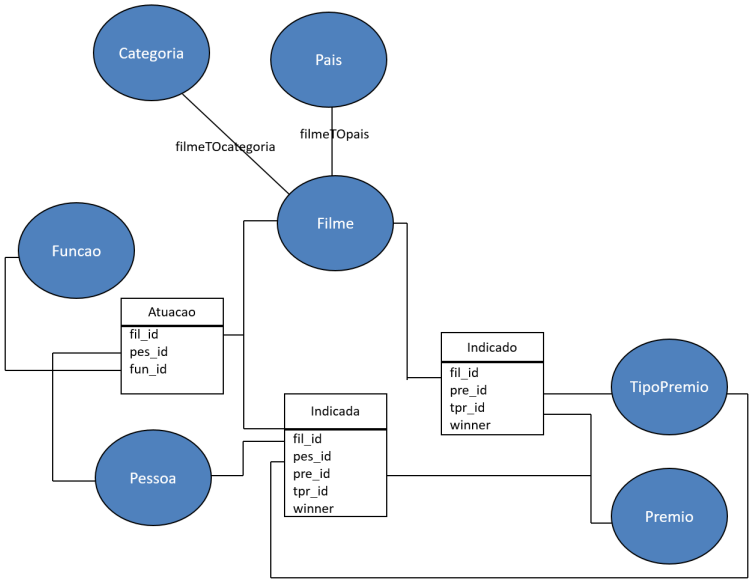


Figura C.4: BaseFilmes - Método $M01$ - Passo 2

No último passo, as demais *joined tables* dão origem aos vértices *Atuacao*, *Indicado* e *Indicada*, e arestas substituindo os relacionamentos entre elas e as *tables* por ela ligadas.

O modelo final do grafo pode ser visto na figura C.5.

C.2.2 Modelando a $BaseFilmes$ conforme o método $M03$ - Grafos Simples (RDF)

Como visto, este método parte do modelo relacional (figura C.2) e tem duas fases, onde os passos do método $M01$ correspondem à primeira fase (figura C.5).

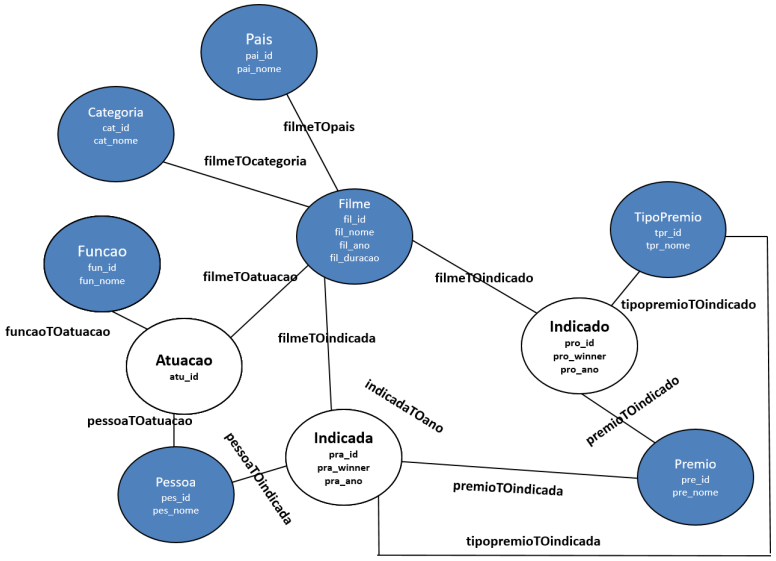


Figura C.5: BaseFilmes - Método $M01$ - Passo 3 (Modelo Final)

Na segunda fase os atributos não identificadores do vértice geram vértices com o mesmo nome e uma aresta entre o vértice original e o novo vértice.

Assim, os atributos *Data* e *Sexo* do vértice *Pessoa* dão origem aos vértices de mesmo nome e são criadas as arestas *pessoaTOdata* e *pessoaTOsexo*. Isso pode ser visto na figura C.6.

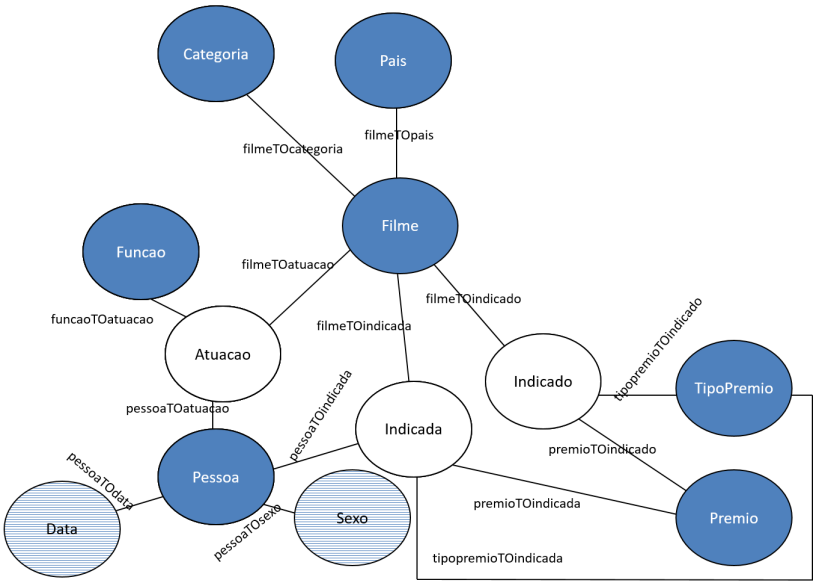


Figura C.6: BaseFilmes - Método $M03$ - Criando vértices Data e Sexo

Seguindo, os atributos *Ano* e *Duracao* são removidos do vértice *Filme* dando origem aos vértices de mesmo nome e as arestas $filmeTOano$ e $filmeTOduracao$. Além disso, o atributo *Ano* dos vértices *Indicado* e *Indicada* também são removidos e substituídos

pelas arestas *indicadoTOano* e *indicadaTOano*.

Executado esse passo, o grafo está modelado e o resultado pode ser visto na figura C.7.

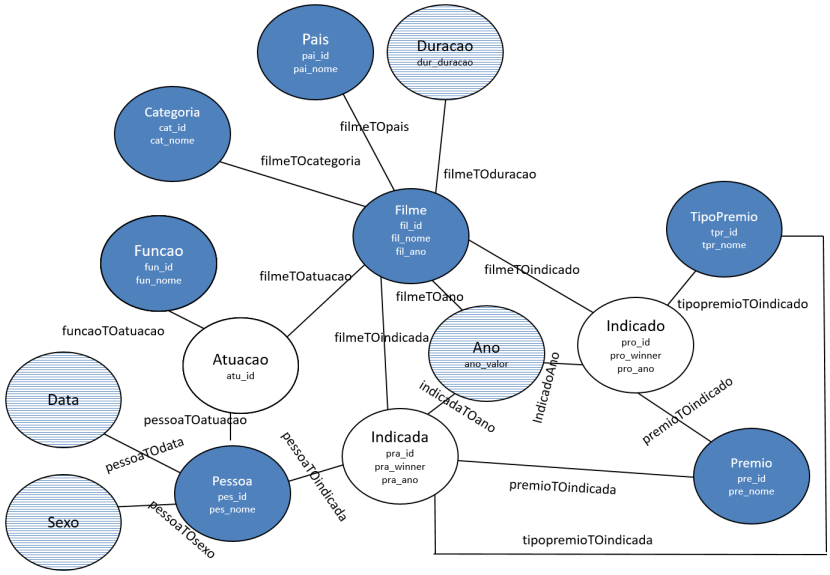


Figura C.7: BaseFilmes - Método $M03$ - Criando vértices Ano e Duracao (Modelo Final)

C.2.3 Modelando a $BaseFilmes$ conforme o método $M04$ - Dirigida

Com base no modelo entidade relacionamento, os relacionamentos são analisados e convertidos em arestas ponderadas, onde, partindo de zero, soma-se um para cada lado do relacionamento com cardinalidade m ou n, considerando-se que $n > 1$ e $m > 1$. Então:

- o relacionamento $m : n$, entre as entidades $Filme$ e $Categoria$, gera uma aresta não direcionada de peso 2 entre as duas entidades;
- o relacionamento $1 : 1$, entre as entidades $Filme$ e $Pais$, gera uma aresta não direcionada de peso 1, entre as duas entidades;
- o relacionamento de grau 3, *Atuou*, gera a entidade *Atuacao*, e três arestas de peso 1 (cardinalidade $1 : n$) dirigidas das entidades $Pessoa$, $Filme$ e $Funcao$ para a nova entidade;
- o relacionamento de grau 3, *Indicado*, gera a entidade *Indicado* e três arestas de peso 1 dirigidas das entidades $Filme$, $Premio$, e $TipoPremio$ para a nova entidade;

- o relacionamento de grau 4, *Indicada*, gera a entidade *Indicada* e quatro arestas de peso 1 dirigidas das entidades *Pessoa*, *Filme*, *Premio* e *TipoPremio* para a nova entidade.

O resultado desse passo pode ser visto na figura C.8.

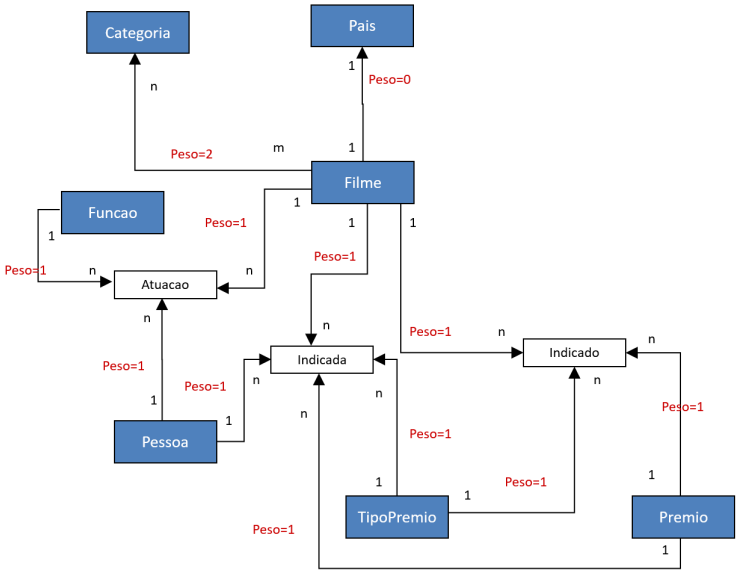


Figura C.8: BaseFilmes - Método $M04$ - Passo 1

No próximo passo são calculados os pesos de entrada ($w^+(n)$) e saída ($w^-(n)$) de cada entidade, isto é, a soma, respectivamente, do peso das arestas que entram e que saem da entidade. Com base nesses passos, foi definido que a entidade *Pais* deve ser incorporada à entidade *Filme* e a entidade *Funcao* à entidade *Atuacao*.

A tabela C.1 e a figura C.9 trazem a aplicação desse passo.

Encerrado o passo 2, no último passo, as entidades são transformadas em vértices, com os atributos originais de cada vértice e os atributos dos vértices incorporados a eles. Assim, os atributos da entidade *Pais* passam a ser atributos do vértice *Filme* e os atributos da entidade *Funcao* passam a ser atributos do vértice *Atuacao*.

A figura C.10 traz o modelo final do grafo pelo método $M04$.

C.2.4 Modelando a *BaseFilmes* conforme o método proposto M05 - Participativa

Como visto, este método parte do modelo entidade relacionamento e seu primeiro passo é o cálculo do grau de incorporação das entidades do modelo.

Tabela C.1: BaseFilmes - Método $M04$ - Passo 2 - Cálculo dos Pesos de Entrada e Saída

Entidade	Aresta que Sai	w+	Aresta que Entra	w-	Ação	Onde
Atuacao		0	pessoaTOatuacao(1) funcaoTOatuacao(1) filmeTOatuacao(1)	3	Isolado	
Categoria	categoriaTOfilme (2)	2	filmeTOcategoria(2)	2	Isolado	
Filme	filmeTOcategoria(2) filmeTOPais(0) filmeTOatuacao(1) filmeTOindicado(1) filmeTOindicada(1)	5		0	Isolado	
Pessoa	pessoaTOatuacao(1) pessoaTOindicada(1)	2		0	Isolado	
Pais		0	filmeTOpais(0)	0	Incorpora	Filme
TipoPremio	tipopremioTOindicado(1) tipopremioTOindicada(1)	2		0	Isolado	
Premio	premioTOindicado(1) premioTOindicada(1)	2		0	Isolado	
Funcao	funcaoTOatuacao(1)	1		0	Incorpora	Atuacao
Indicada		0	pessoaTOindicada(1) filmeTOindicada(1) premioTOindicada(1) tipopremioTOindicada(1)	4	Isolado	
Indicado		0	filmeTOindicado(1) premioTOindicado(1) tipopremioTOindicado(1)	3	Isolado	

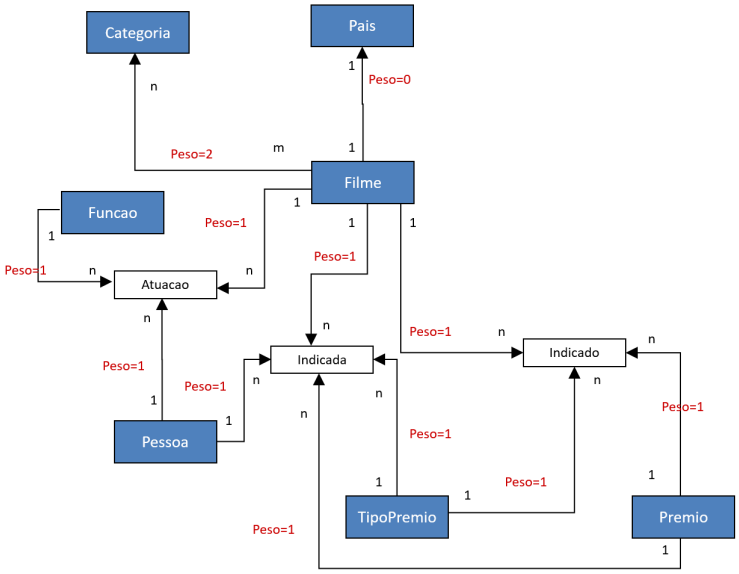


Figura C.9: BaseFilmes - Método $M04$ - Passo 2 - Grupos

A tabela C.2 traz o cálculo do grau de incorporação das entidades. Deve ser observado que a entidade *TipoPremio*, apesar de estar relacionada com duas entidades diferentes, *Indicada* e *Indicado* pode ser considerada como relacionada com apenas 1 entidade, já que os prêmios ligados a pessoas são diferentes dos prêmios ligados a filmes, ou seja, os que participam de um relacionamento não participam do outro.

Com base no grau de incorporação das entidades, é calculado, então, o grau de incorporação dos relacionamentos de grau maior que 1, que é a soma do grau de incorporação

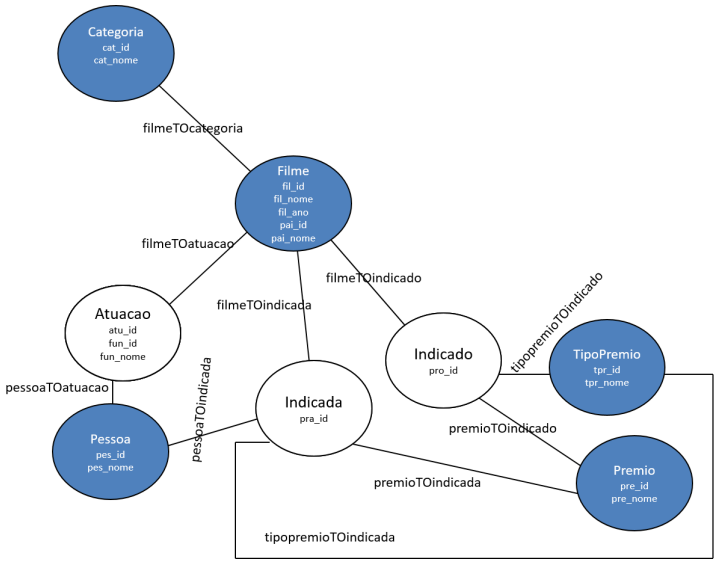


Figura C.10: BaseFilmes - Método $M04$ - Modelo Final

Tabela C.2: BaseFilmes - Método $M05$ - Fase 1 - Grau de Incorporação das Entidades

EE	Grau de Expansão		EGE	Relações			GI(EE)	Obs.
	Expansão	GE	EGE	Relações	EGA	GR	=se(GE+GR)=2 então 1 senão 0	
Categoria	pouco provável	0	1	1	1	1	1	
Filme	constante	2	0	6	6	0	0	
Funcao	pouco provável	0	1	1	1	1	1	
Premio	provável	1	0	2	2	0	0	
TipoPremio	pouco provável	0	1	1	1	1	1	(1)
Pessoa	constante	2	0	3	3	0	0	
Pais	pouco provável	0	1	1	1	1	1	

(1) - embora tenha duas relações, registros que participam de uma relação não participam de outra

das entidades que dele participam. A tabela C.3 traz esse cálculo.

Tabela C.3: BaseFilmes - Método $M05$ - Fase 1 - Grau de Incorporação dos Relacionamentos

Relacionamento	Relacionamento	Entidades Participantes								GIR(R3)
		EE1	GI(EE1)	EE2	GI(EE2)	EE3	GI(EE3)	EE4	GI(EE4)	GI(EE1)+GI(EE2)+GI(EE3)
FilmePais	2	Filme	0	Pais	1					1
Atuou	3	Filme	0	Pessoa	0	Funcao	1			1
Indicado	3	Filme	0	Premio	0	TipoPremio	1			1
Indicada	4	Filme	0	Premio	0	TipoPremio	1	Pessoa	0	1

Atuou, relacionamento de grau 3, tem grau de incorporação igual à 1, decorrente da entidade *Funcao*. Então, esta entidade deve ser eliminada do modelo e seus atributos incorporados pelo relacionamento, que passa a ser de grau 2 e ter grau de incorporação 0 (zero). Esse passo pode ser visto na figura C.11.

Indicado, também relacionamento de grau 3, temgrau de incorporação igual à 1, decorrente da entidade *TipoPremio*. Então, esta entidade deve ser eliminada do modelo e seus atributos incorporados pelo relacionamento, que passa a ser de grau 2 e ter grau de incorporação 0 (zero). Esse passo pode ser visto na figura C.12.

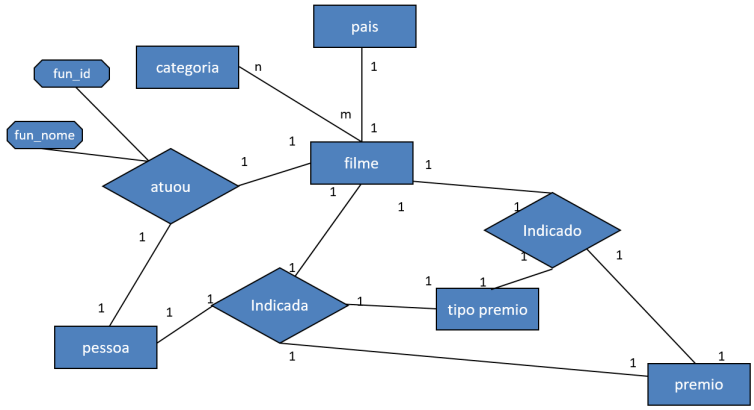


Figura C.11: BaseFilmes - Método $M05$ - Incorporação da Entidade Funcao ao Relacionamento Atuou

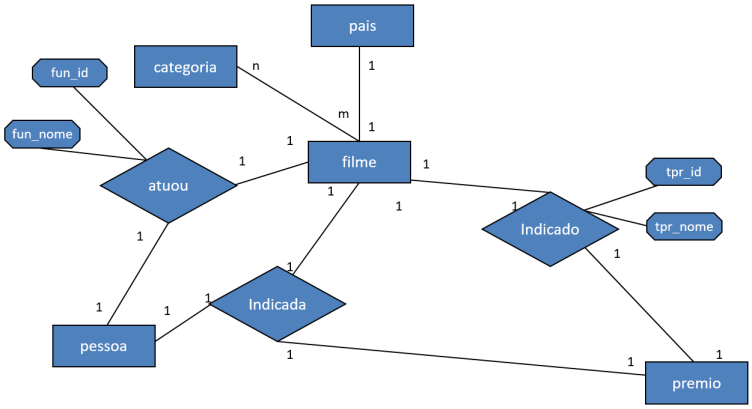


Figura C.12: BaseFilmes - Método $M05$ - Incorporação da Entidade TipoPremio ao Relacionamento Indicado

Indicada, relacionamento de grau 4, tem grau de incorporação igual à 1, decorrente da entidade *TipoPremio*. Então, esta entidade deve ser eliminada do modelo e seus atributos incorporados pelo relacionamento, que passa a ser de grau 3 e ter grau de incorporação 0 (zero). Como o novo relacionamento tem grau 3 e grau de incorporação 0 não há mais entidades a serem incorporadas por ele. Assim, o relacionamento deve ser substituído por uma entidade de nome *Indicada*, que receberá seus atributos e os atributos de *TipoPremio* e pelos relacionamentos entre a nova entidade e as entidades *Premio*, *Pessoa* e *Filme*. Esse passo pode ser visto nas figura C.13.

Finalmente, o relacionamento *FilmePais*, de grau 2, tem grau de incorporação 1, decorrente da entidade *Pais*, que deve ser, então, incorporada pela entidade *Filme*.

Com esse passo, agora só existem relacionamentos de grau 2 e grau de incorporação 0 no modelo entidade relacionamento. Assim, a fase 1 está completa. Este modelo pode ser visto na figura C.14

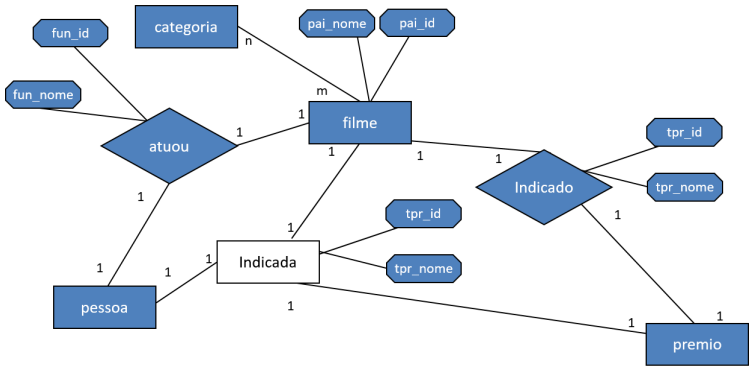


Figura C.13: BaseFilmes - Método $M05$ - Conversão do Relacionamento Indicada em Vértice

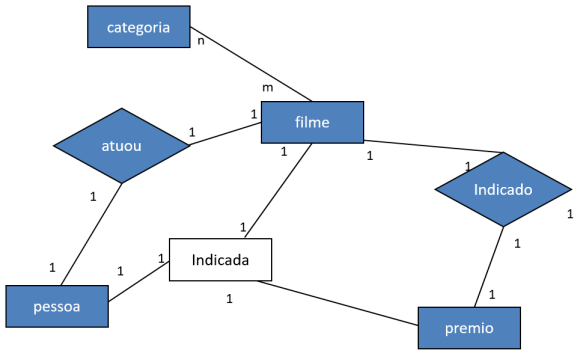


Figura C.14: BaseFilmes - Método $M05$ - Fim da Fase 1 - MER2

Agora, na fase 2, o primeiro passo é converter as entidades em vértices e os relacionamentos em arestas, como pode ser visto na figura C.15.

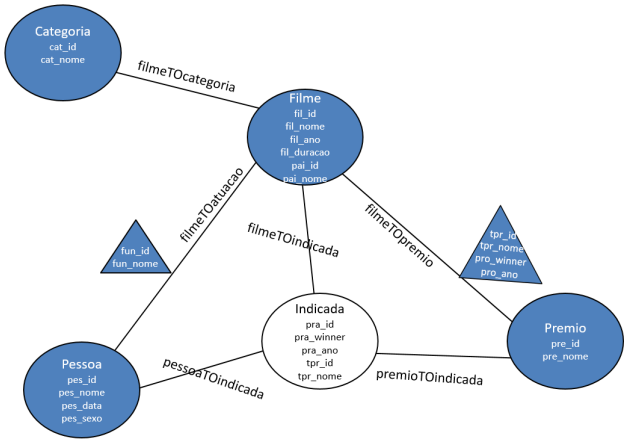


Figura C.15: BaseFilmes - Método $M05$ - Fase 2 - Converte Entidades em Vértices e Relacionamentos em Arestas

Feito isso, o processo está finalizado e o grafo gerado. O passo 3 do método prevê a possibilidade da divisão de vértices com mais de duas propriedades em dois vértices, um com as propriedades identificadoras do vértice e o segundo com as demais. Entretanto, o

volume previsto de dados para esta base, de cerca de 10.000 (dez mil) registros de filmes e pessoas torna esse passo desnecessário.

A figura C.16 traz o modelo final dos quatro métodos a fim de facilitar a comparação do resultado de cada método.

Pode ser observado que o resultado final tem as mesmas características da base modelo da dissertação.

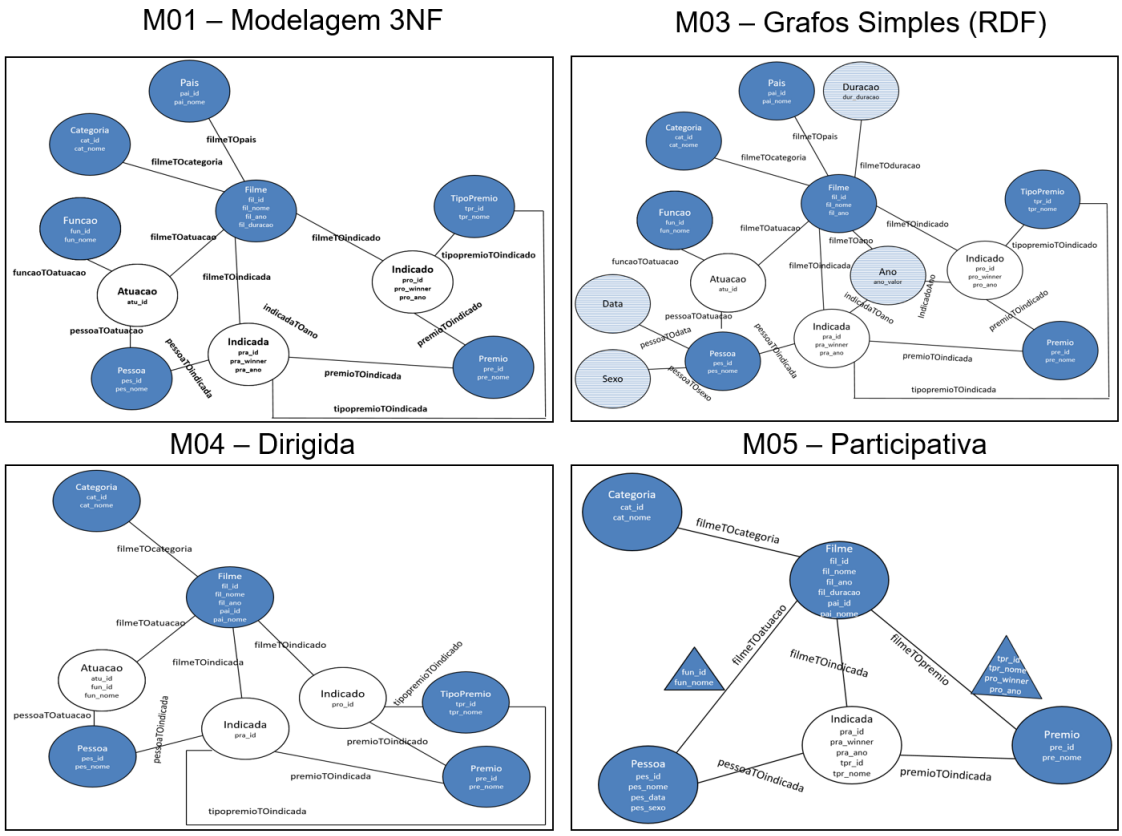


Figura C.16: BaseFilmes - Resultado Final dos Quatro Métodos

C.3 Implementando a *BaseFilmes*

A *BaseFilmes* foi implementada a partir de duas bases disponíveis na Internet com dados reais de filmes e pessoas que atuaram neles:

- base 1 - (KAGGLE, 2017) - arquivo CSV com todos os filmes, atores, diretores, roteiristas indicados e ganhadores do Oscar, prêmio do cinema americano, desde 1927 até 2015;

•base 2 - (TEMPLE, 2016) - arquivo CSV com 5.043 filmes como nome do filme, diretor, até três atores e atrizes que atuaram no filme e categoria do filme.

As diversas entidades e relacionamentos da base foram populadas conforme segue:

•*pessoa* - o nome da pessoa nos registros de indicação para Ator, Atriz, Diretor, etc da base 1 mais os nomes dos três atores em cada registro de filmes na base 2. O campo sexo foi preenchido com 1, masculino, para aqueles indicados à melhor ator, 2, feminino, para as indicadas a melhor atriz, e 3, não informado, para os demais. Como não existiam datas, foi gerada uma data aleatória para cada registro;

•$filme$ - o nome dos filmes nas duas bases. O atributo ano foi preenchido com o ano do Oscar para os filmes vindos da base 1, ou o ano do filme naqueles da base 2. O atributo duração foi preenchido com o valor informado na base 2. Não existindo essa informação foi atribuído o valor -1;

•*categoria* - as diversas categorias informadas nas duas bases foram combinadas e montada uma base sem registros duplicados;

•$funcao$ - as funções foram definidas como 'Ator', 'Atriz, 'Diretor, 'Roteirista' e 'Músico´;

•*premio* - foram colocados os registros 'Oscar', 'Festival de Cannes','César','Framboesa de Outro' (piores filmes do ano nos Estados Unidos) e 'Gramado';

•*tipoPremio* - os diversos tipos de prêmios constantes na base 1;

•relacionamento $FilmeCategoria$ - conforme as diferentes categorias indicadas para o filme nas bases 1 e 2, foram criados estes relacionamentos;

•relacionamento *Atuou* - de acordo com a base 1, foram combinados o $filme$, a *pessoa* indicada ao prêmio e, de acordo com o tipo do prêmio, foi definida a função. Se Melhor Atriz, atuou como Atriz, se Melhor Diretor, atuou como Diretor, etc;

•relacionamento *indicada* - de acordo com a base 1, onde os registros traziam a *pessoa*, o $filme$, o *tipoPremio* a que ela foi indicada (melhor atriz, melhor ator, etc.) e o *premio* (Oscar). Além disso, o registro trazia o atributo *winner*, com o valor 1 se a pessoa foi a premiada ou 0 se apenas indicada;

•relacionamento *indicado* - da mesma maneira que o relacionamento *indicada*, sendo relacionados apenas aqueles registros da base 1 sem pessoa informada, ou seja, indicação do filme, como, por exemplo, 'Melhor filme'.

As tabelas C.4, C.5 e C.6 trazem o total de vértices por método, total de arestas por método e tamanho das bases.

Analisando as tabelas pode-se observar que, ao contrário da *BaseTeste* onde a base com menor número de vértices e arestas foi a do método $M04$, aqui isso ocorreu com a base do método $M05$, mas que apesar disso, ainda teve área total maior que a do método $M04$. De resto, as mesmas conclusões podem ser obtidas aqui.

Tabela C.4: BaseFilmes - Tamanho das Bases por Método

Base	Tamanho (MB)
M01	335
M03	488
M04	129
M05	132

Tabela C.5: BaseFilmes X Métodos - Total de Vértices por Método

Vértice	M01	M03	M04	M05
Ano	-	89	-	-
Atuacao	21.850	21.850	21.850	-
Categoria	26	26	26	26
Data	-	7.034	-	-
Duracao	-	192	-	-
Filme	12.509	12.509	12.509	12.509
Funcao	6	6		-
Indicada	2.681	2.681	2.681	2.681
Indicado	5.902	5.902	5.902	5.902
Pais	86	85	-	-
Pessoa	9.595	9.595	9.595	9.595
Premio	5	5	5	5
Sexo	-	3	-	-
TipoPremio	114	114	114	-
	52.774	60.091	52.682	30.718

C.4 Consultas Efetuadas

Seguindo todos os passos descritos na seção 4.2 para as consultas na *BaseTeste* da dissertação, para a *BaseFilmes* foram executadas 14 diferentes consultas com o objetivo de explorar as diferentes características dos diversos modelos gerados. Da mesma maneira que nos experimentos com a *BaseTeste*, aqui também cada consulta foi executada oito vezes, sendo a primeira considerada como aquecimento e descartada e as sete últimas consideradas para a média.

A tabela C.7 traz a relação das consultas utilizadas nos experimentos.

Tabela C.6: BaseFilmes X Métodos - Total de Arestas por Método

Aresta	M01	M03	M04	M05
atuacaoTOfilme	21.850	21.850	21.850	-
filmeTOano	-	12.509	-	-
filmeTOatuacao	21.850	21.850	21.850	-
filmeTOcategoria	14.241	14.241	14.241	14.241
filmeTOduracao	-	12.509	-	-
filmeTOindicada	2.681	2.681	2.681	2.681
filmeTOindicado	5.902	5.902	5.902	-
filmeTOpais	12.509	12.509	-	-
filmeTOpremio	-	-	-	5.902
funcaoTOatuacao	21.850	21.850	-	-
indicadaTOano	-	2.681	-	-
indicadaTOpremio	2.681	2.681	2.681	-
indicadaTOtipoPremio	2.681	2.681	2.681	2.681
indicadoTOano	-	5.902	-	-
indicadoTOpremio	5.902	5.902	5.902	-
indicadoTOtipoPremio	5.902	5.902	5.902	-
pessoaTOatuacao	43.700	43.700	43.700	-
pessoaTOdata	-	9.595	-	-
pessoaTOindicada	2.681	2.681	2.681	2.681
pessoaTOsexo	-	9.595	-	-
	164.430	217.221	130.071	28.186

As consultas foram executadas no mesmo equipamento das consultas com a $BaseTeste$ e seguindo os mesmos critérios. A tabela C.8 traz o tempo médio das consultas. À semelhança das tabelas do capítulo 4, elas contém as colunas:

- $Consulta$ - número da consulta;
- $M01$ $M03$ $M04$ $M05$, com as seguintes colunas para cada modelagem:
  - MED - tempo médio em milissegundos (ms) das 7 execuções das consultas nas bases dos métodos;
  - DP - desvio padrão do tempo das 7 execuções;
  - IC - intervalo de confiança *student t* para as 7 execuções.

A análise dos resultados permite afirmar que:

- o mesmo ganho de desempenho obtido com a aplicação do método $M05$ na $BaseTeste$ se repetiu na $BaseFilmes$;
- da mesma maneira que na base anterior, aqui também o método proposto não foi o de melhor desempenho em todas as consultas, mas, também, não foi o de pior performance, reafirmando, assim, sua validade;

Tabela C.7: BaseFilmes - Consultas Efetuadas

Consulta	Entidades Envolvidas	Objetivo
01	Pessoa, Filme, Funcao	Pessoas Com Maior Número de Atuações
02	Pessoa, Filme, Funcao, Atuação	Pessoas Com Maior Número de Atuações como Diretor
03	Pessoa, Filme, Funcao, Atuação, TipoPremio, Indicada	Pessoas com Maior Número de Indicações
04	Pessoa, Filme, Funcao, Atuação, TipoPremio, Indicada	Pessoas com Maior Número de Premiações
05	Pessoa, Filme, Funcao, Atuação, TipoPremio, Indicada	Pessoas com Maior Número de Indicações Sem Premiação
06	Pessoa, Filme, TipoPremio, Indicada	Pessoas que Receberam Uma Única Indicação e Foram Premiadas
07	Filme, Categoria	Categorias com Maior Número de Filmes
08	Pessoa, Filme, Funcao, Atuação, TipoPremio, Indicada, Indicado, Premio, Categoria	Filmes por Pessoas
09	Pessoa, Atuacao, Funcao, Indicada	Pessoas Com Maior Número de Indicações
10	Filme, Indicada, Indicado, Premio	Filmes Mais Indicados que Foram Premiados em Todas as Indicações
11	Filme, Pais, Indicada, Indicado, Premio	Países com Mais Indicações e Premiações para Melhor Filme Estrangeiro
12	Filme, Pais, Indicada, Indicado, Premio	Paises com Maior Numero de Filmes na Base exceto EUA
13	Pessoa	Caminho Mínimo Entre Fernanda Torres e Charles Chaplin
14	Pessoa	Todos os Caminhos Mínimos entre Fernanda Torres e Keanu Reeves

Tabela C.8: BaseFilmes - 7 Execuções da Consulta ($M01/M02$, $M03$, $M04$, $M05$)

Cons	M01/M02			M03			M04			M05		
	MED	DP	IC	MED	DP	IC	MED	DP	IC	MED	DP	IC
01	465	22,4	±20,7	617	39,5	±36,5	333	35,0	±32,4	185	78,8	±72,9
02	101	15,0	±13,9	365	555,5	±513,8	92	15,2	±14,0	82	3,3	±3,1
03	50	4,0	±3,7	65	4,3	±4,0	52	3,7	±3,4	39	4,2	±3,9
04	17	3,6	±3,4	20	3,8	±3,5	19	4,5	±4,1	17	2,6	±2,4
05	51	4,8	±4,5	56	8,1	±7,5	52	3,1	±2,8	38	3,3	±3,1
06	24	6,9	±6,4	23	3,8	±3,5	21	2,9	±2,7	20	3,7	±3,4
07	32	4,9	±4,5	39	6,5	±6,0	30	1,1	±1,0	30	2,7	±2,5
08	22	4,2	±3,8	42	3,8	±3,6	19	4,9	±4,6	44	5,8	±5,4
09	954	68,1	±63,0	865	21,6	±20,0	906	41,2	±38,1	781	24,1	±22,3
10	965	17,8	±16,5	918	8,8	±8,1	1.030	28,0	±25,9	1.010	5,1	±4,7
11	186	14,9	±13,8	185	6,3	±5,8	182	17,5	±16,2	195	4,2	±3,9
12	14	4,2	±3,9	17	3,7	±3,4	34	4,1	±3,8	31	3,9	±3,6
13	9	2,5	±2,3	9	2,3	±2,1	8	3,5	±3,2	1	0,5	±0,5
14	16	2,1	±1,9	16	3,0	±2,8	17	4,9	±4,5	2	0,5	±0,5

•as consultas demonstraram que onde houve redução de caminho com a eliminação de vértices e incorporação de seus atributos por relacionamentos, houve melhora no desempenho.

C.5 Tempo de Execução das Consultas para a $BaseFilmes$

Nesta seção são apresentadas quatro tabelas, de C.9 até C.12, com o tempo de execução de todas as consultas, para os métodos $M01/M02$, $M03$, $M04$ e $M05$, para a base $BaseFilmes$. As tabelas têm as seguintes colunas:

•consulta - número da consulta;

•AQ - tempo, em milissegundos, da primeira execução da consulta, considerada de aquecimento e descartada do cálculo do tempo médio;

•$E01$ à $E07$ - tempo, em milissegundos, das execuções 1, 2, 3, 4, 5, 6 e 7 da consulta após aquela de aquecimento. Cada consulta foi sempre executada em sequência;

•MED - tempo médio das sete execuções da consulta;

•DP - desvio padrão sobre o tempo das 7 execuções consideradas para a média;

•IC - intervalo de confiança da amostra, considerando um fator de 95% *student t*.

Tabela C.9: Apêndice B - $BaseFilmes$ - Métodos M01/M02 - Execução das Consultas

Consulta	AQ	E01	E02	E03	E04	E05	E06	E07	MED	DP	IC
01	1.006	456	443	488	495	453	441	482	465	22,4	±20,7
02	175	127	100	115	100	96	87	85	101	15,0	±13,9
03	120	45	54	48	47	51	48	56	50	4,0	±3,7
04	70	14	25	16	17	15	18	16	17	3,6	±3,4
05	56	51	51	50	56	46	59	46	51	4,8	±4,5
06	60	29	28	20	16	20	22	36	24	6,9	±6,4
07	58	34	36	27	27	33	39	27	32	4,9	±4,5
08	88	31	24	19	21	21	22	19	22	4,2	±3,8
09	1.566	1.082	918	924	922	904	1.017	912	954	68,1	±63,0
10	1.004	932	978	971	950	982	969	974	965	17,8	±16,5
11	230	205	176	170	183	173	207	186	186	14,9	±13,8
12	20	20	10	16	11	19	10	14	14	4,2	±3,9
13	110	13	12	7	7	9	10	7	9	2,5	±2,3
14	49	20	16	17	16	13	16	16	16	2,1	±1,9

Tabela C.10: Apêndice B - *BaseFilmes* - Método M03 - Execução das Consultas

Consulta	AQ	E01	E02	E03	E04	E05	E06	E07	MED	DP	IC
01	1.455	663	600	596	645	571	665	581	617	39,5	±36,5
02	262	169	161	145	150	1.625	152	155	365	555,5	±513,8
03	108	68	72	68	61	65	63	60	65	4,3	±4,0
04	121	24	24	17	20	14	20	23	20	3,8	±3,5
05	75	67	68	56	48	49	54	52	56	8,1	±7,5
06	53	30	24	21	20	18	23	23	23	3,8	±3,5
07	51	31	39	41	46	47	32	34	39	6,5	±6,0
08	139	48	38	40	46	38	42	41	42	3,8	±3,6
09	1.516	903	871	876	853	843	866	841	865	21,6	±20,0
10	1.205	924	907	912	913	931	913	925	918	8,8	±8,1
11	196	189	184	190	180	174	192	183	185	6,3	±5,8
12	21	19	12	19	14	23	17	15	17	3,7	±3,4
13	123	12	10	6	11	9	6	9	9	2,3	±2,1
14	70	17	17	18	16	12	11	19	16	3,0	±2,8

Tabela C.11: Apêndice B - *BaseFilmes* - Método M04 - Execução das Consultas

Consulta	AQ	E01	E02	E03	E04	E05	E06	E07	MED	DP	IC
01	984	354	386	307	288	312	320	361	333	35,0	±32,4
02	186	102	120	74	91	87	80	91	92	15,2	±14,0
03	77	55	51	45	56	51	54	50	52	3,7	±3,4
04	58	27	17	13	16	20	21	20	19	4,5	±4,1
05	73	51	57	49	53	48	52	54	52	3,1	±2,8
06	38	23	16	24	22	21	19	24	21	2,9	±2,7
07	33	31	32	31	29	30	29	31	30	1,1	±1,0
08	135	29	19	20	21	15	18	14	19	4,9	±4,6
09	1.384	929	913	880	882	987	878	874	906	41,2	±38,1
10	1.091	1.070	1.035	1.029	1.018	1.048	979	1.029	1.030	28,0	±25,9
11	247	185	220	176	172	171	172	176	182	17,5	±16,2
12	41	40	34	33	35	28	36	29	34	4,1	±3,8
13	81	13	11	6	5	11	4	7	8	3,5	±3,2
14	55	22	23	20	12	14	11	15	17	4,9	±4,5

Tabela C.12: Apêndice B - *BaseFilmes* - Método M05 - Execução das Consultas

Consulta	AQ	E01	E02	E03	E04	E05	E06	E07	MED	DP	IC
01	813	269	208	210	180	18	217	194	185	78,8	±72,9
02	170	82	75	83	82	83	82	86	82	3,3	±3,1
03	92	47	35	41	41	37	35	39	39	4,2	±3,9
04	93	20	20	15	17	18	16	13	17	2,6	±2,4
05	46	39	32	43	37	39	37	39	38	3,3	±3,1
06	49	22	24	17	22	18	23	14	20	3,7	±3,4
07	33	33	32	33	28	27	29	27	30	2,7	±2,5
08	85	56	43	45	43	41	38	41	44	5,8	±5,4
09	1.367	828	778	763	760	788	789	761	781	24,1	±22,3
10	1.162	1.016	1.012	1.008	1.014	1.012	1.007	1.001	1.010	5,1	±4,7
11	230	199	187	193	199	194	193	197	195	4,2	±3,9
12	64	27	35	35	33	27	32	26	31	3,9	±3,6
13	108	2	1	2	1	1	2	1	1	0,5	±0,5
14	38	2	2	2	1	2	2	1	2	0,5	±0,5

C.6 Scripts Cypher das Consultas Efetuadas nas BasesFilmes

Nesta seção estão as listagens das 14 consultas Cypher efetuadas sobre as bases de filmes.

C.6.1 Consultas Cypher - *BaseFilmes* - Métodos M01 e M02

Listagem C.1: Scripts das Consultas na *BaseFilmes* - Métodos M01 e M02

```
//Consulta 01 - Pessoas Com Maior Número de Atuações
MATCH (p:Pessoa)-[PessoaAtuou]->(a:Atuacao)<-[funcaoTOatuacao
   ]-(fun:Funcao),(a)<-[FilmeComnAtuacao]-(f:Filme)
RETURN p.pes_nome AS pessoa, fun.fun_nome, COLLECT(f.fil_nome
    + '(' + f.fil_ano + ')') AS filmes, COUNT(*) AS qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000;

//Consulta 02 - Pessoas Com Maior Número de Atuações como
   Diretor
MATCH (p:Pessoa)-[PessoaAtuou]->(a:Atuacao)<-[funcaoTOatuacao
   ]-(fun:Funcao{fun_nome:'Diretor'}),
```

```
      (a)<-[FilmeComnAtuacao]-(f:Filme)
RETURN p.pes_nome AS pessoa, fun.fun_nome, COLLECT(f.fil_nome
    + '(' + f.fil_ano + ')') AS filmes, COUNT(*) AS qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000;

//Consulta 03 - Pessoas com Maior Número de Indicações
MATCH (p:Pessoa)-[r1:pessoaTOindicada]->(pra:Indicada)<-[
   filmeTOindicada]-(f:Filme),
      (pra)-[indicadaTOtipoPremio]-(tpr:TipoPremio)
RETURN p.pes_nome AS pessoa, COLLECT(f.fil_nome + '(' + tpr.
   tpr_nome + ')') AS atuacao, COUNT(*) AS qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000;

//Consulta 04 - Pessoas com Maior Número de Premiações
MATCH (p:Pessoa)-[r1:pessoaTOindicada]->(pra:Indicada{
   pra_winner:1})<-[filmeTOindicada]-(f:Filme),
      (pra)-[indicadaTOtipoPremio]-(tpr:TipoPremio)
RETURN p.pes_nome AS pessoa, COLLECT(f.fil_nome + '(' + tpr.
   tpr_nome + ')') AS atuacao, COUNT(*) AS qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000;

//Consulta 05 - Pessoas com Maior Número de Indicações Sem
   Premiação
MATCH (p:Pessoa)-[r1:pessoaTOindicada]->(pra:Indicada)<-[
   filmeTOindicada]-(f:Filme),
         (pra{pra_winner:0})-[indicadaTOtipoPremio]-(tpr:
            TipoPremio)
RETURN p.pes_nome AS pessoa, COLLECT(f.fil_nome + '(' + tpr.
   tpr_nome + ')') AS atuacao, COUNT(*) AS qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000;
```

```
35 //Consultas 06 - Pessoas que Receberam Uma Única Indicação e
     Foram Premiadas
   MATCH (p:Pessoa)-[r1:pessoaTOindicada]->(pra1:Indicada)
   WITH p, COUNT(*) AS qtd_indicacao
   WHERE qtd_indicacao = 1
   MATCH (p)-[r2:pessoaTOindicada]->(pra2:Indicada{pra_winner
     :1})<-[filmeTOindicada]-(f:Filme),
40       (pra2)-[indicadaTOtipoPremio]->(tpr:TipoPremio)
   RETURN p.pes_nome AS pessoa, f.fil_nome, tpr.tpr_nome AS
     premio, pra2.pra_ano AS anoPremio
   ORDER BY pessoa
   LIMIT 1000;

45 //Consulta 07 - Categorias com Maior Número de Filmes
   MATCH (f:Filme)-[filmeTOcategoria]->(cat:Categoria)
   RETURN cat.cat_nome AS categoria, COUNT(*) AS qtd_filmes
   ORDER BY qtd_filmes DESC, categoria;

50 //Consulta 08 - Filmes por Pessoas
   MATCH (p:Pessoa)
   WHERE (p.pes_sexo = 2) AND (SUBSTRING(p.pes_data,0,4) = '1962
     ')
   WITH p
   MATCH (p)-[r2:pessoaTOatuacao]->(a:Atuacao)<-[filmeTOatuacao
     ]-(f:Filme),(a)-[funcaoTOatuacao]-(fun:Funcao)
55 WITH p, f, fun.fun_nome as funcao
   OPTIONAL MATCH (p)-[r3:pessoaTOindicada]->(pra:Indicada)<-[
     filmeTOindicada]-(f),
       (pra)-[indicadaTOtipoPremio]->(tpr:TipoPremio)
   WITH  p, f, funcao, pra, tpr,
           SIZE((f)-[:filmeTOpremio]->()) +
60         SIZE((f)-[:filmeTOindicada]->()) AS qtd_indic_film,
           SIZE((f)-[:filmeTOpremio]->({pro_winner:1})) +
           SIZE((f)-[:filmeTOindicada]->({pra_winner:1})) AS
              qtd_prem_film,
```

```
        SIZE((f)-[:filmeTOcategoria]->()) AS qtd_cat_film
RETURN p.pes_nome AS nome, f.fil_nome AS filme, f.fil_ano AS
   ano, funcao, tpr.tpr_nome AS premio,
        CASE pra.pra_winner
        WHEN 0 THEN 'Indicado'
        WHEN 1 THEN 'Premiado'
        ELSE ''
        END AS resultado,
        qtd_indic_film,
        qtd_prem_film
ORDER BY qtd_prem_film DESC;

// Consulta 09 - Pessoas Com Maior Número de Indicações
MATCH (p:Pessoa)-[r1:pessoaTOatuacao]->(a)<-[funcaoTOatuacao
   ]-(fun:Funcao)
WHERE fun.fun_id < 3
WITH DISTINCT p,
SIZE((p)-[:pessoaTOindicada]->()) AS qtd_indic,
SIZE((p)-[:pessoaTOindicada]->({pra_winner:1})) AS qtd_prem
RETURN p.pes_nome AS nome, qtd_indic, qtd_prem
ORDER BY qtd_indic DESC;

// Consulta 10 - Filmes Mais Indicados que Foram Premiados em
    Todas as Indicações
MATCH (p:Filme)
WITH DISTINCT p,
SIZE((p)-[:filmeTOpremio]->()) +
SIZE((p)-[:filmeTOindicada]->()) AS qtd_indic,
SIZE((p)-[:filmeTOpremio]->({pro_winner:1})) + SIZE((p)-[:
   filmeTOindicada]->({pra_winner:1})) AS qtd_prem
WHERE qtd_indic = qtd_prem
RETURN p.fil_nome AS nome, qtd_indic, qtd_prem
ORDER BY qtd_prem DESC;
```

```
//Consulta 11 - Países com Mais Indicações e Premiações para
   Melhor Filme Estrangeiro
MATCH (f:Filme)-[filmeTOpais]->(pai:Pais)
WITH f.fil_nome AS filme, pai.pai_nome AS pais,
           SIZE((f)-[:filmeTOpremio]-()-[:
              indicadoTOtipoPremio]-({tpr_id:30})) AS qtd_i,
           SIZE((f)-[:filmeTOpremio]-({pro_winner:1})-[:
              indicadoTOtipoPremio]-({tpr_id:30})) AS qtd_p
RETURN pais, SUM(qtd_i) AS qtd_indic, SUM(qtd_p) AS qtd_prem
ORDER BY qtd_indic DESC;

// Consulta 12 - Paises com Maior Numero de Filmes na Base
   exceto EUA
MATCH (f:Filme)-[filmeTOpais]-(p:Pais)
WHERE p.pai_id > 0 AND p.pai_id <> 65
RETURN p.pai_nome AS pais, COUNT(*) AS qtd
ORDER BY qtd DESC, pais
LIMIT 25;

//Consultas 13 - Caminho Mínimo Entre Fernanda Torres e
   Charles Chaplin
MATCH p = shortestPath( (from:Pessoa) - [r1:pessoaTOatuacao|:
   atuacaoTOfilme|:filmeTOatuacao|:pessoaTOatuacao*] -> (to:
   Pessoa) )
WHERE (from.pes_nome = 'Fernanda Torres')
  AND (to.pes_nome = 'Charles Chaplin')
RETURN p;

//Consulta 14 - Todos os Caminhos Mínimos entre Fernanda
   Torres e Keanu Reeves
MATCH p = allShortestPaths( (from:Pessoa) - [r1:
   pessoaTOatuacao|:atuacaoTOfilme|:filmeTOatuacao|:
   pessoaTOatuacao*] -> (to:Pessoa) )
WHERE (from.pes_nome = 'Fernanda Torres')
  AND (to.pes_nome = 'Keanu Reeves')
```

```
RETURN p;
```

C.6.2 Consultas Cypher - *BaseFilmes* - Método M03

Listagem C.2: Scripts das Consultas na *BaseFilmes* - Método M03

```
//Consulta 01 - Pessoas Com Maior Número de Atuações
MATCH (p:Pessoa)-[PessoaAtuou]->(a:Atuacao)<-[funcaoTOatuacao
   ]-(fun:Funcao),
      (a)<-[FilmeComnAtuacao]-(f:Filme)-[filmeTOano]->(ano:
         Ano)
RETURN p.pes_nome AS pessoa, fun.fun_nome, COLLECT(f.fil_nome
    + '(' + ano.ano_valor + ')') AS filmes, COUNT(*) AS qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000;

//Consulta 02 - Pessoas Com Maior Número de Atuações como
   Diretor
MATCH (p:Pessoa)-[PessoaAtuou]->(a:Atuacao)<-[funcaoTOatuacao
   ]-(fun:Funcao{fun_nome:'Diretor'}),
      (a)<-[FilmeComnAtuacao]-(f:Filme)-[filmeTOano]->(ano:
         Ano)
RETURN p.pes_nome AS pessoa, fun.fun_nome, COLLECT(f.fil_nome
    + '(' + ano.ano_valor + ')') AS filmes, COUNT(*) AS qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000;

//Consulta 03 - Pessoas com Maior Número de Indicações
MATCH (p:Pessoa)-[r1:pessoaTOindicada]->(pra:Indicada)<-[
   filmeTOindicada]-(f:Filme),
      (pra)-[indicadaTOtipoPremio]-(tpr:TipoPremio)
RETURN p.pes_nome AS pessoa, COLLECT(f.fil_nome + '(' + tpr.
   tpr_nome + ')') AS atuacao, COUNT(*) AS qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000;
```

```
//Consulta 04 - Pessoas com Maior Número de Premiações
MATCH (p:Pessoa)-[r1:pessoaTOindicada]->(pra:Indicada{
   pra_winner:1})<-[filmeTOindicada]-(f:Filme),
      (pra)-[indicadaTOtipoPremio]-(tpr:TipoPremio)
RETURN p.pes_nome AS pessoa, COLLECT(f.fil_nome + '(' + tpr.
   tpr_nome + ')') AS atuacao, COUNT(*) AS qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000;

//Consulta 05 - Pessoas com Maior Número de Indicações Sem
   Premiação
MATCH (p:Pessoa)-[r1:pessoaTOindicada]->(pra:Indicada)<-[
   filmeTOindicada]-(f:Filme),
         (pra{pra_winner:0})-[indicadaTOtipoPremio]-(tpr:
            TipoPremio)
RETURN p.pes_nome AS pessoa, COLLECT(f.fil_nome + '(' + tpr.
   tpr_nome + ')') AS atuacao, COUNT(*) AS qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000;

//Consultas 06 - Pessoas que Receberam Uma Única Indicação e
   Foram Premiadas
MATCH (p:Pessoa)-[r1:pessoaTOindicada]->(pra1:Indicada)
WITH p, COUNT(*) AS qtd_indicacao
WHERE qtd_indicacao = 1
MATCH (p)-[r2:pessoaTOindicada]->(pra2:Indicada{pra_winner
   :1})<-[filmeTOindicada]-(f:Filme),
      (pra2)-[indicadaTOtipoPremio]->(tpr:TipoPremio),
      (pra2)-[indicadoTOano]->(ano:Ano)
RETURN p.pes_nome AS pessoa, f.fil_nome, tpr.tpr_nome AS
   premio, ano.ano_valor AS anoPremio
ORDER BY pessoa
LIMIT 1000;

//Consulta 07 - Categorias com Maior Número de Filmes
```

```
MATCH (f:Filme)-[filmeTOcategoria]->(cat:Categoria)
RETURN cat.cat_nome AS categoria, COUNT(*) AS qtd_filmes
ORDER BY qtd_filmes DESC, categoria;

//Consulta 08 - Filmes por Pessoas
MATCH (p:Pessoa)-[pessoaTOsexo]->(sex:Sexo),(p)-[pessoaTOdata
   ]->(dat:Data)
WHERE (sex.sex_sexo = 2) AND (SUBSTRING(dat.dat_data,0,4) = '
   1962')
WITH p
MATCH (p)-[r2:pessoaTOatuacao]->(a:Atuacao)<-[filmeTOatuacao
   ]-(f:Filme),(a)-[funcaoTOatuacao]-(fun:Funcao)
WITH p, f, fun.fun_nome as funcao
OPTIONAL MATCH (p)-[r3:pessoaTOindicada]->(pra:Indicada)<-[
   filmeTOindicada]-(f)-[filmeTOano]->(ano:Ano),
     (pra)-[indicadaTOtipoPremio]->(tpr:TipoPremio)
WITH  p, f, funcao, pra, tpr, ano,
        SIZE((f)-[:filmeTOpremio]->()) +
        SIZE((f)-[:filmeTOindicada]->()) AS qtd_indic_film,
        SIZE((f)-[:filmeTOpremio]->({pro_winner:1})) +
        SIZE((f)-[:filmeTOindicada]->({pra_winner:1})) AS
           qtd_prem_film,
        SIZE((f)-[:filmeTOcategoria]->()) AS qtd_cat_film
RETURN p.pes_nome AS nome, f.fil_nome AS filme, ano.ano_valor
    AS ano, funcao, tpr.tpr_nome AS premio,
        CASE pra.pra_winner
        WHEN 0 THEN 'Indicado'
        WHEN 1 THEN 'Premiado'
        ELSE ''
        END AS resultado,
        qtd_indic_film,
        qtd_prem_film
ORDER BY qtd_prem_film DESC;

// Consulta 09 - Pessoas Com Maior Número de Indicações
```

```
MATCH (p:Pessoa)-[r1:pessoaTOatuacao]->(a)<-[funcaoTOatuacao
   ]-(fun:Funcao)
WHERE fun.fun_id < 3
WITH DISTINCT p,
SIZE((p)-[:pessoaTOindicada]->()) AS qtd_indic,
SIZE((p)-[:pessoaTOindicada]->({pra_winner:1})) AS qtd_prem
RETURN p.pes_nome AS nome, qtd_indic, qtd_prem
ORDER BY qtd_indic DESC;

// Consulta 10 - Filmes Mais Indicados que Foram Premiados em
    Todas as Indicações
MATCH (p:Filme)
WITH DISTINCT p,
SIZE((p)-[:filmeTOpremio]->()) +
SIZE((p)-[:filmeTOindicada]->()) AS qtd_indic,
SIZE((p)-[:filmeTOpremio]->({pro_winner:1})) + SIZE((p)-[:
   filmeTOindicada]->({pra_winner:1})) AS qtd_prem
WHERE qtd_indic = qtd_prem
RETURN p.fil_nome AS nome, qtd_indic, qtd_prem
ORDER BY qtd_prem DESC;

//Consulta 11 - Países com Mais Indicações e Premiações para
   Melhor Filme Estrangeiro
MATCH (f:Filme)-[filmeTOpais]->(pai:Pais)
WITH f.fil_nome AS filme, pai.pai_nome AS pais,
           SIZE((f)-[:filmeTOpremio]-()-[:
              indicadoTOtipoPremio]-({tpr_id:30})) AS qtd_i,
           SIZE((f)-[:filmeTOpremio]-({pro_winner:1})-[:
              indicadoTOtipoPremio]-({tpr_id:30})) AS qtd_p
RETURN pais, SUM(qtd_i) AS qtd_indic, SUM(qtd_p) AS qtd_prem
ORDER BY qtd_indic DESC;

// Consulta 12 - Paises com Maior Numero de Filmes na Base
   exceto EUA
MATCH (f:Filme)-[filmeTOpais]-(p:Pais)
```

```
WHERE p.pai_id > 0 AND p.pai_id <> 65
RETURN p.pai_nome AS pais, COUNT(*) AS qtd
ORDER BY qtd DESC, pais
LIMIT 25;

//Consultas 13 - Caminho Mínimo Entre Fernanda Torres e
   Charles Chaplin
MATCH p = shortestPath( (from:Pessoa) - [r1:pessoaTOatuacao|:
   atuacaoTOfilme|:filmeTOatuacao|:pessoaTOatuacao*] -> (to:
   Pessoa) )
WHERE (from.pes_nome = 'Fernanda Torres')
  AND (to.pes_nome = 'Charles Chaplin')
RETURN p;

//Consulta 14 - Todos os Caminhos Mínimos entre Fernanda
   Torres e Keanu Reeves
MATCH p = allShortestPaths( (from:Pessoa) - [r1:
   pessoaTOatuacao|:atuacaoTOfilme|:filmeTOatuacao|:
   pessoaTOatuacao*] -> (to:Pessoa) )
WHERE (from.pes_nome = 'Fernanda Torres')
  AND (to.pes_nome = 'Keanu Reeves')
RETURN p;
```

C.6.3 Consultas Cypher - *BaseFilmes* - Método M04

Listagem C.3: Scripts das Consultas na *BaseFilmes* - Método M04

```
//Consulta 01 - Pessoas Com Maior Número de Atuações
MATCH (p:Pessoa)-[PessoaAtuou]->(a:Atuacao),(a)<-[
   filmeTOatuacao]-(f:Filme)
RETURN p.pes_nome AS pessoa, a.fun_nome, COLLECT(f.fil_nome +
    '(' + f.fil_ano + ')') AS filmes, COUNT(*) AS qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000;
```

```
//Consulta 02 - Pessoas Com Maior Número de Atuações como
   Diretor
MATCH (p:Pessoa)-[PessoaAtuou]->(a:Atuacao{fun_nome:'Diretor'
   }),(a)<-[filmeTOatuacao]-(f:Filme)
RETURN p.pes_nome AS pessoa, a.fun_nome, COLLECT(f.fil_nome +
    '(' + f.fil_ano + ')') AS filmes, COUNT(*) AS qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000;

//Consulta 03 - Pessoas com Maior Número de Indicações
MATCH (p:Pessoa)-[r1:pessoaTOindicada]->(pra:Indicada)<-[
   filmeTOindicada]-(f:Filme),
       (pra)-[indicadaTOtipoPremio]-(tpr:TipoPremio)
RETURN p.pes_nome AS pessoa, COLLECT(f.fil_nome + '(' + tpr.
   tpr_nome + ')') AS atuacao, COUNT(*) AS qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000;

//Consulta 04 - Pessoas com Maior Número de Premiações
MATCH (p:Pessoa)-[r1:pessoaTOindicada]->(pra:Indicada{
   pra_winner:1})<-[filmeTOindicada]-(f:Filme),
       (pra)-[indicadaTOtipoPremio]-(tpr:TipoPremio)
RETURN p.pes_nome AS pessoa, COLLECT(f.fil_nome + '(' + tpr.
   tpr_nome + ')') AS atuacao, COUNT(*) AS qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000;

//Consulta 05 - Pessoas com Maior Número de Indicações Sem
   Premiação
MATCH (p:Pessoa)-[r1:pessoaTOindicada]->(pra:Indicada)<-[
   filmeTOindicada]-(f:Filme),
           (pra{pra_winner:0})-[indicadaTOtipoPremio]-(tpr:
              TipoPremio)
RETURN p.pes_nome AS pessoa, COLLECT(f.fil_nome + '(' + tpr.
   tpr_nome + ')') AS atuacao, COUNT(*) AS qtd
```

```
ORDER BY qtd DESC, pessoa
LIMIT 1000;

//Consultas 06 - Pessoas que Receberam Uma Única Indicação e
   Foram Premiadas
MATCH (p:Pessoa)-[r1:pessoaTOindicada]->(pra1:Indicada)
WITH p, COUNT(*) AS qtd_indicacao
WHERE qtd_indicacao = 1
MATCH (p)-[r2:pessoaTOindicada]->(pra2:Indicada{pra_winner
   :1})<-[filmeTOindicada]-(f:Filme),
      (pra2)-[indicadaTOtipoPremio]->(tpr:TipoPremio)
RETURN p.pes_nome AS pessoa, f.fil_nome, tpr.tpr_nome AS
   premio, pra2.pra_ano AS anoPremio
ORDER BY pessoa
LIMIT 1000;

//Consulta 07 - Categorias com Maior Número de Filmes
MATCH (f:Filme)-[filmeTOcategoria]->(cat:Categoria)
RETURN cat.cat_nome AS categoria, COUNT(*) AS qtd_filmes
ORDER BY qtd_filmes DESC, categoria;

//Consulta 08 - Filmes por Pessoas
MATCH (p:Pessoa)
WHERE (p.pes_sexo = 2) AND (SUBSTRING(p.pes_data,0,4) = '1962
   ')
WITH p
MATCH (p)-[r2:pessoaTOatuacao]->(a:Atuacao)<-[filmeTOatuacao
   ]-(f:Filme)
WITH p, f, a.fun_nome as funcao
OPTIONAL MATCH (p)-[r3:pessoaTOindicada]->(pra:Indicada)<-[
   filmeTOindicada]-(f),
     (pra)-[indicadaTOtipoPremio]->(tpr:TipoPremio)
WITH  p, f, funcao, pra, tpr,
        SIZE((f)-[:filmeTOpremio]->()) +
        SIZE((f)-[:filmeTOindicada]->()) AS qtd_indic_film,
```

```
60          SIZE((f)-[:filmeTOpremio]->({pro_winner:1})) +
            SIZE((f)-[:filmeTOindicada]->({pra_winner:1})) AS
               qtd_prem_film,
            SIZE((f)-[:filmeTOcategoria]->()) AS qtd_cat_film
    RETURN p.pes_nome AS nome, f.fil_nome AS filme, f.fil_ano AS
       ano, funcao, tpr.tpr_nome AS premio,
            CASE pra.pra_winner
65          WHEN 0 THEN 'Indicado'
            WHEN 1 THEN 'Premiado'
            ELSE ''
            END AS resultado,
            qtd_indic_film,
70          qtd_prem_film
    ORDER BY qtd_prem_film DESC;

    // Consulta 09 - Pessoas Com Maior Número de Indicações
    MATCH (p:Pessoa)-[r1:pessoaTOatuacao]->(a)
75  WHERE a.fun_id < 3
    WITH DISTINCT p,
    SIZE((p)-[:pessoaTOindicada]->()) AS qtd_indic,
    SIZE((p)-[:pessoaTOindicada]->({pra_winner:1})) AS qtd_prem
    RETURN p.pes_nome AS nome, qtd_indic, qtd_prem
80  ORDER BY qtd_indic DESC;

    // Consulta 10 - Filmes Mais Indicados que Foram Premiados em
        Todas as Indicações
    MATCH (p:Filme)
    WITH DISTINCT p,
85  SIZE((p)-[:filmeTOpremio]->()) +
    SIZE((p)-[:filmeTOindicada]->()) AS qtd_indic,
    SIZE((p)-[:filmeTOpremio]->({pro_winner:1})) + SIZE((p)-[:
       filmeTOindicada]->({pra_winner:1})) AS qtd_prem
    WHERE qtd_indic = qtd_prem
    RETURN p.fil_nome AS nome, qtd_indic, qtd_prem
90  ORDER BY qtd_prem DESC;
```

```
//Consulta 11 - Países com Mais Indicações e Premiações para
   Melhor Filme Estrangeiro
MATCH (f:Filme)
WITH f.fil_nome AS filme, f.pai_nome AS pais,
            SIZE((f)-[:filmeTOpremio]-()-[:
               indicadoTOtipoPremio]-({tpr_id:30})) AS qtd_i,
            SIZE((f)-[:filmeTOpremio]-({pro_winner:1})-[:
               indicadoTOtipoPremio]-({tpr_id:30})) AS qtd_p
RETURN pais, SUM(qtd_i) AS qtd_indic, SUM(qtd_p) AS qtd_prem
ORDER BY qtd_indic DESC;

// Consulta 12 - Paises com Maior Numero de Filmes na Base
   exceto EUA
MATCH (f:Filme)
WHERE f.pai_id > 0 AND f.pai_id <> 65
RETURN f.pai_nome AS pais, COUNT(*) AS qtd
ORDER BY qtd DESC, pais
LIMIT 25;

//Consultas 13 - Caminho Mínimo Entre Fernanda Torres e
   Charles Chaplin
MATCH p = shortestPath( (from:Pessoa) - [r1:pessoaTOatuacao|:
   atuacaoTOfilme|:filmeTOatuacao|:pessoaTOatuacao*] -> (to:
   Pessoa) )
WHERE (from.pes_nome = 'Fernanda Torres')
  AND (to.pes_nome = 'Charles Chaplin')
RETURN p;

//Consulta 14 - Todos os Caminhos Mínimos entre Fernanda
   Torres e Keanu Reeves
MATCH p = allShortestPaths( (from:Pessoa) - [r1:
   pessoaTOatuacao|:atuacaoTOfilme|:filmeTOatuacao|:
   pessoaTOatuacao*] -> (to:Pessoa) )
WHERE (from.pes_nome = 'Fernanda Torres')
```

```
  AND (to.pes_nome = 'Keanu Reeves')
RETURN p;
```

C.6.4 Consultas Cypher - *BaseFilmes* - Método M05

Listagem C.4: Scripts das Consultas na *BaseFilmes* - Método M05

```
//Consulta 01 - Pessoas Com Maior Número de Atuações
MATCH (p:Pessoa)-[r1:pessoaTOfilme]->(f:Filme)
RETURN p.pes_nome AS pessoa, r1.fun_nome, COLLECT(f.fil_nome
   + '(' + f.fil_ano + ')') AS filmes, COUNT(*) AS qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000.

//Consulta 02 - Pessoas Com Maior Número de Atuações como
   Diretor
MATCH (p:Pessoa)-[r1:pessoaTOfilme{fun_nome:'Diretor'}]->(f:
   Filme)
RETURN p.pes_nome AS pessoa, COLLECT(f.fil_nome) AS filmes,
   COUNT(*) AS qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000;

//Consulta 03 - Pessoas com Maior Número de Indicações
MATCH (p:Pessoa)-[r1:pessoaTOindicada]->(pra:Indicada)<-[
   filmeTOindicada]-(f:Filme),(pra)-[r2:indicadaTOpremio]->()
RETURN p.pes_nome AS pessoa, COLLECT(f.fil_nome + '(' + pra.
   tpr_nome + ')') AS atuacao, COUNT(*) AS qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000;

//Consulta 04 - Pessoas com Maior Número de Premiações
MATCH (p:Pessoa)-[r1:pessoaTOindicada]->(pra:Indicada)<-[
   filmeTOindicada]-(f:Filme),(pra{pra_winner:1})-[r2:
   indicadaTOpremio]->()
```

```
RETURN p.pes_nome AS pessoa, COLLECT(f.fil_nome + '(' + pra.
    tpr_nome + '/'+ pra.pra_ano + ')') AS atuacao, COUNT(*) AS
     qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000;

//Consulta 05 - Pessoas com Maior Número de Indicações Sem
    Premiação
MATCH (p:Pessoa)-[r1:pessoaTOindicada]->(pra:Indicada)<-[
    filmeTOindicada]-(f:Filme),(pra{pra_winner:0})-[r2:
    indicadaTOpremio]->()
RETURN p.pes_nome AS pessoa, COLLECT(f.fil_nome + '(' + pra.
    tpr_nome + ')') AS atuacao, COUNT(*) AS qtd
ORDER BY qtd DESC, pessoa
LIMIT 1000;

//Consultas 06 - Pessoas que Receberam Uma Única Indicação e
    Foram Premiadas
MATCH (p:Pessoa)-[r1:pessoaTOindicada]->(pra1:Indicada)
WITH p, COUNT(*) AS qtd_indicacao
WHERE qtd_indicacao = 1
MATCH (p)-[r2:pessoaTOindicada]->(pra2:Indicada{pra_winner
    :1})<-[filmeTOindicada]-(f:Filme)
RETURN p.pes_nome AS pessoa, f.fil_nome, pra2.tpr_nome AS
    premio, pra2.pra_ano AS anoPremio
ORDER BY pessoa
LIMIT 1000;

//Consulta 07 - Categorias com Maior Número de Filmes
MATCH (f:Filme)-[filmeTOcategoria]->(cat:Categoria)
RETURN cat.cat_nome AS categoria, COUNT(*) AS qtd_filmes
ORDER BY qtd_filmes DESC, categoria;

//Consulta 08 - Filmes por Pessoas
MATCH (po:PessoaOutro)<-[r1:pessoaTOpessoaOutro]-(p:Pessoa)
```

```
WHERE (po.pes_sexo = 2) AND (SUBSTRING(po.pes_data,0,4) = '
   1962')
WITH p
MATCH (p)-[r2:pessoaTOfilme]->(f:Filme)
WITH p, f, r2.fun_nome as funcao
OPTIONAL MATCH (p)-[r3:pessoaTOindicada]->(pra:Indicada)<-[
   filmeTOindicada]-(f)
WITH  p, f, funcao, pra,
        SIZE((f)-[:filmeTOpremio]->()) +
        SIZE((f)-[:filmeTOindicada]->()) AS qtd_indic_film,
        SIZE((f)-[:filmeTOpremio{pro_winner:1}]->()) +
        SIZE((f)-[:filmeTOindicada]->({pra_winner:1})) AS
           qtd_prem_film,
        SIZE((f)-[:filmeTOcategoria]->()) AS qtd_cat_film
RETURN p.pes_nome AS nome, f.fil_nome AS filme, f.fil_ano AS
   ano, funcao, pra.tpr_nome AS premio,
        CASE pra.pra_winner
        WHEN 0 THEN 'Indicado'
        WHEN 1 THEN 'Premiado'
        ELSE ''
        END AS resultado,
        qtd_indic_film,
        qtd_prem_film
ORDER BY qtd_prem_film DESC;

// Consulta 09 - Pessoas Com Maior Número de Indicações
MATCH (p:Pessoa)-[r1:pessoaTOfilme]->()
WHERE r1.fun_id < 3
WITH DISTINCT p,
SIZE((p)-[:pessoaTOindicada]->()) AS qtd_indic,
SIZE((p)-[:pessoaTOindicada]->({pra_winner:1})) AS qtd_prem
RETURN p.pes_nome AS nome, qtd_indic, qtd_prem
ORDER BY qtd_indic DESC;
```

```
// Consulta 10 - Filmes Mais Indicados que Foram Premiados em
    Todas as Indicações
MATCH (p:Filme)
WITH DISTINCT p,
SIZE((p)-[:filmeTOpremio]->()) +
SIZE((p)-[:filmeTOindicada]->()) AS qtd_indic,
SIZE((p)-[:filmeTOpremio{pro_winner:1}]->()) + SIZE((p)-[:
   filmeTOindicada]->({pra_winner:1})) AS qtd_prem
WHERE qtd_indic = qtd_prem
RETURN p.fil_nome AS nome, qtd_indic, qtd_prem
ORDER BY qtd_prem DESC;

//Consulta 11 - Países com Mais Indicações e Premiações para
   Melhor Filme Estrangeiro
MATCH (f:Filme)
WITH f.fil_nome AS filme, f.pai_nome AS pais,
            SIZE((f)-[:filmeTOpremio{tpr_id:30}]-()) AS qtd_i,
            SIZE((f)-[:filmeTOpremio{tpr_id:30,pro_winner
               :1}]-()) AS qtd_p
RETURN pais, SUM(qtd_i) AS qtd_indic, SUM(qtd_p) AS qtd_prem
ORDER BY qtd_indic DESC;

// Consulta 12 - Paises com Maior Numero de Filmes na Base
   exceto EUA
MATCH (f:Filme)
WHERE f.pai_id > 0 AND f.pai_id <> 65
RETURN f.pai_nome AS pais, COUNT(*) AS qtd
ORDER BY qtd DESC, pais
LIMIT 25;

//Consultas 13 - Caminho Mínimo Entre Fernanda Torres e
   Charles Chaplin
MATCH p = shortestPath( (from:Pessoa) - [r1:pessoaTOfilme|:
   filmeTOpessoa*] -> (to:Pessoa) )
WHERE (from.pes_nome = 'Fernanda Torres')
```

```
105   AND (to.pes_nome = 'Charles Chaplin')
RETURN p;

//Consulta 14 - Todos os Caminhos Mínimos entre Fernanda
   Torres e Keanu Reeves
MATCH p = allShortestPaths( (from:Pessoa) - [r1:pessoaTOfilme
   |:filmeTOpessoa*] -> (to:Pessoa) )
110 WHERE (from.pes_nome = 'Fernanda Torres')
  AND (to.pes_nome = 'Keanu Reeves')
RETURN p;
```