Uma Proposta de Mapeamento do Modelo Conceitual Entidade-Relacionamento Estendido para o Modelo de Dados NoSQL de Grafos

Luiz Sergio Velasques Urquiza Junior[1], **Luis Mariano del Val Cura**[1]

[1] Programa de Mestrado em Ciência da Computação
UNIFACCAMP – Centro Universitário Campo Limpo Paulista
Campo Limpo Paulista – SP – Brasil

lsurquiza@gmail.com, delval@cc.faccamp.br

Abstract. *This article proposes a series of high-level algorithms for mapping elements of Extended Entity-Relationship (EER) conceptual model to a propertie graph logical model. In this approach, traditional database design is considered, starting with conceptual modeling and later mapping, via well-defined algorithms, for a logical graph model. It is intended that the generated logical model is used as a tool for validation and verification of integrity constraints in graph databases (GDB).*

Resumo. *Este artigo propõe uma série de algoritmos em alto nível para o mapeamento dos elementos do modelo conceitual Entidade-Relacionamento Estendido (EER) para um modelo lógico de multigafo de propriedades. Nesta abordagem, considera-se um projeto de banco de dados tradicional, iniciando com a modelagem conceitual e posterior mapeamento, via algoritmos bem definidos, para um modelo lógico de grafos. Pretende-se que o modelo lógico gerado seja utilizado como ferramenta de validação e verificação de restrições de integridade em bancos de dados de grafos (BDG).*

1. Introdução

Já há muito tempo os bancos de dados relacionais dominam o mercado de sistemas gerenciadores de bancos de dados (SGBD) baseados nas ideias de Edgar Frank Codd em seu artigo "A relational model of data for large shared databanks" [Codd 1970]. Entretanto, para empresas como a Amazon, o Google e o Facebook a web evoluiu para um grande emaranhado de informações distribuídas em que bancos de dados relacionais passaram a apresentar algumas dificuldades em termos de desempenho e disponibilidade dos dados [Pokorny 2013].

Diante disso, abordagens alternativas têm surgido para lidar com sistemas que lidam com um volume e uma variedade de dados cada vez maior, exigindo altíssima velocidade em seu processamento. Estas alternativas são denominadas *modelos de dados NoSQL*[Sadalage and Fowler 2013] e são divididos em quatro categorias: Chave-valor, orientados a documento, colunares e orientados a grafo [Atzeni et al. 2020].

Este artigo propõe uma metodologia de projeto de banco de dados *NoSQL* orientado a grafos partindo do mapeamento direto do modelo conceitual Entidade-Relacionamento Estendido (EER), considerando elementos complexos como relacionamentos *n-ários* e especializações, para o modelo lógico de multigrafo de propriedades. O

objetivo principal é que este modelo lógico possa reunir metadados suficientes para permitir a validação das instâncias do Banco de Dados de Grafo (BDG), a fim de verificar a consistência e a integridade dos dados armazenados.

A próxima seção apresenta alguns trabalhos relacionados. A seção 3 apresenta uma ideia de modelo lógico de multigrafo de propriedades, enquanto a seção 4 mostra os algoritmos de mapeamento do modelo conceitual para modelo lógico proposto. Por fim, um estudo de caso é exemplificado na seção 5 e as conclusões são mostradas na seção 6.

2. Trabalhos Relacionados

Apesar dos bancos de dados *NoSQL* estarem se tornando relevantes nos últimos anos, poucos trabalhos tem sido publicados quanto ao projeto e modelagem de tais bancos de dados. Um destes trabalhos propõe um processo de engenharia reversa em BDGs a fim de gerar os esquemas conceitual e lógico, destacando a relevância de um projeto de banco de dados mesmo em modelos *NoSQL* [Comyn-Wattiau and Akoka 2017].

Outra abordagem, defendida por [Roy-Hubara et al. 2017], propõe um projeto de banco de dados a partir da conversão, em duas etapas, de um modelo conceitual EER para um modelo de grafos de propriedades. A primeira etapa trata de um refinamento do modelo conceitual, transformando o esquema conceitual em um diagrama composto apenas de relacionamentos binário. A segunda etapa trata da conversão do modelo conceitual refinado para o modelo lógico de grafos. Além disso, propõe a adição de restrições de cardinalidade ao grafos.

A nossa abordagem reconhece a relevância dos esquemas conceitual e lógico em um projeto de BDG, a exemplo de [Comyn-Wattiau and Akoka 2017] e, assim como [Roy-Hubara et al. 2017], propõe um projeto de banco de dados de grafo a partir do mapeamento do modelo conceitual EER, porém, vai além ao propor uma metodologia que trata diretamente os relacionamentos n-ários e processos de especialização do modelo EER.

3. Modelo Lógico de Grafos

Em um modelo lógico de grafos as entidades e os relacionamentos são representadas respectivamente por nós e arestas e a grande maioria dos Sistemas Gerenciadores de Bancos de Dados de Grafos (SGBDGs) implementa um tipo específico de grafo, chamado *multigrafo de propriedades* [Robinson et al. 2013].

Este tipo de grafo permite que seus nós e arestas admitam rótulos e contenham conjuntos de propriedades. Suas arestas são direcionadas, conectando sempre um *nó de origem* e um *nó de destino*. Em SGBDGs como o *Neo4j*, os nós podem admitir mais de um rótulo e também é possível que as propriedades admitam *arrays* de tipos primitivos, para dar suporte a atributos definidos como multivalorados no modelo conceitual.

A proposta de modelagem lógica de BDG deste artigo consiste na geração de um modelo lógico representado por um multigrafo de propriedades, denominado *grafo de esquema*, que representa um esquema lógico para as instâncias do BDG.

4. Algoritmos de Mapeamento do Modelo Conceitual para Modelo Lógico

Nesta seção são apresentados formalmente os algoritmos em pseudocódigo utilizados para geração de um modelo lógico de BDG a partir do modelo conceitual EER.

Algorithm 1 Mapeia_Entidade (Entidade: e)

```
1: alpha ← Nó (e.nome)
2: for Cada atributo em e.atributos_monovalorados do
3:     alpha.insere_propriedade(atributo)
4: end for
5: for Cada atributo em e.atributos_multivalorados do
6:     beta ← Nó (atributo.nome)
7:     beta.insere_propriedade (atributo)
8:     beta.cria_aresta ("Pertence a", alpha)
9: end for
```

O Algoritmo 1 mapeia uma entidade e do modelo EER para um nó $alpha$ no grafo de esquema. Considera-se que a entidade e é um objeto que contém um nome e listas de atributos monovalorados e multivalorados. O método "`insere_propriedade`" trata do mapeamento dos atributos de e para propriedades de $alpha$, lidando com atributos simples e compostos. Em caso de atributo composto, são geradas várias propriedades, sendo uma para cada um de seus componentes.

Para mapear atributos multivalorados de e é criado o nó $beta$, que se conecta a $alpha$ executando o método "`cria_aresta`", que cria uma aresta dirigida, rotulada como "*Pertence a*", originando em $beta$ (o nó que a invocou) e chegando em $alpha$ (o nó referenciado no método). Considera-se que o as instâncias do banco de dados não dão suporte a propriedades de tipos complexos de dados, como *arrays* e *listas*, portanto, atributos multivalorados são transformados em nós.

O mapeamento de relacionamentos cria arestas entre dois nós participantes. Os atributos do relacionamento são mapeados como propriedades para a aresta. O Algoritmo 2 trata estes casos. As exceções a este tratamento são os relacionamentos *n-ários* e relacionamentos que possuem atributos multivalorados. Os nós de origem e de destino, utilizados como parâmetros no Algoritmo 2 são definidos de acordo com a semântica do relacionamento no modelo EER.

Algorithm 2 Mapeia_Relacionamento_Aresta (Relacionamento: r; No: no_origem, no_destino)

```
1: epson ← no_origem.cria_aresta (r.nome, no_destino)
2: for Cada atributo in r.atributos_monovalorados do
3:     epson.insere_propriedade (atributo)
4: end for
```

Relacionamentos *n-ários* ou com atributos multivalorados são traduzidos em nós e conectados a cada uma das entidades participantes com arestas rotuladas como "*Participa de*". Atributos multivalorados são transformados em nós e ligados ao nó do relacionamento com arestas rotuladas como "*Pertence a*". O Algoritmo 3 é executado para tratar estes casos.

Quanto às especializações, nos casos mais simples, onde se configura uma hierarquia do tipo "*Is a*" [Pokorny et al. 2017], basta criar uma aresta rotulada como "*É um*" originando na subclasse e *subindo* até a superclasse. Outros casos envolvem uma superclasse que se especializa em duas ou mais subclasses. A superclasse pode assumir *participação total* na especialização e/ou pode ser definida uma *disjunção* entre as sub-

Algorithm 3 Mapeia_Relacionamento_No (Relacionamento: r; Nó: participantes[ ])

```
1: alpha ← Nó (r.Nome)
2: for Cada participante em participantes do
3:     participante.cria_aresta ("Participa de", alpha)
4: end for
5: for Cada atributo em r.atributos_monovalorados do
6:     alpha.insere_propriedade (atributo)
7: end for
8: for Cada atributo em r.atributos_multivalorados do
9:     beta ← Nó (atributo.Nome)
10:    beta.insere_propriedade (atributo)
11:    beta.cria_aresta ("Pertence a", alpha)
12: end for
```

classes. Para tratar estas situações é criada uma *hiperaresta* [Gallo et al. 1993], que se origina na superclasse e se ramifica para as subclasses, indicando a participação (total ou parcial) da superclasse e se a especialização admite ou não a disjunção entre as subclasses. O mapeamento de especializações é tratado no Algoritmo 4.

Algorithm 4 Mapeia_Especialização (Especialização: p; Nó: superclasse, subclasses[ ])

```
1: if subclasses.quantidade = 1 then
2:     subclasses[0].cria_aresta("É um", alpha)
3: else
4:     if p.participação = "Total" then
5:         if p.disjunção = true then
6:             superclasse.cria_aresta("Disjunção Total", subclasses)
7:         else
8:             superclasse.cria_aresta("Sobreposição Total", subclasses)
9:         end if
10:    else
11:        if p.disjunção = true then
12:            superclasse.cria_aresta("Disjunção Parcial", subclasses)
13:        else
14:            superclasse.cria_aresta("Sobreposição Parcial", subclasses)
15:        end if
16:    end if
17: end if
```

O objeto *p* representa uma especialização, o qual mantém atributos para informar a participação e se há ou não disjunção. Este algoritmo passa um conjunto de nós das subclasses como parâmetro no método "`cria_aresta`", indicando que cada subclasse referencia a mesma aresta criada pela superclasse. Na prática, a aresta criada possui uma origem na superclasse e vários destinos nas subclasses.

5. Estudo de Caso

Como estudo de caso, considera-se o modelo EER mostrado pela Figura 1. Este diagrama apresenta um modelo conceitual de banco de dados para armazenamento de dados de trabalhos científicos em exposição em algum evento. Neste exemplo, as pessoas assumem os papéis de orientadores, alunos ou avaliadores que se relacionam com os trabalhos de maneiras distintas. Os orientadores tem o papel de orientar ou coorientar (tipo) os trabalhos. Alunos desenvolvem trabalhos enquanto que os avaliadores atribuem notas a cada

trabalho. Cada trabalho tem um nível de ensino (fundamental ou médio) e concorre a uma ou mais categorias (exatas, humanas, biológicas ou multidisciplinar).

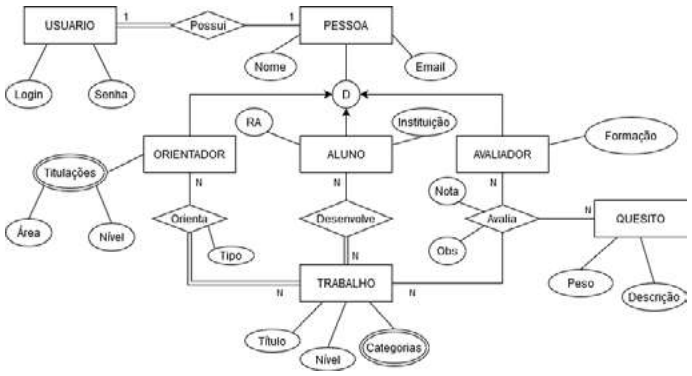

Figura 1. Diagrama EER "Exposição de Trabalhos Científicos"

Aplicando os algoritmos de mapeamento para geração do grafo de esquema, obtém-se o diagrama da Figura 2. Nesta figura, as entidades foram todas mapeadas para nós rotulados, contendo conjuntos de propriedades mapeados a partir dos atributos monovalorados. Os atributos multivalorados "Titulações" (Orientador) e "Categorias"(Trabalho) foram mapeados como nós e ligados aos respectivos nós proprietários com arestas rotuladas como "Pertence a". A grande maioria dos relacionamentos foi mapeada como arestas no grafo de esquema, exceto o relacionamento ternário "Avalia", que foi mapeado como um nó e ligado aos outros nós participantes do relacionamento com arestas rotuladas como "Participa de". A especialização de "Pessoa" em "Orientador/Aluno/Avaliador" foi mapeada como uma hiperaresta rotulada como "Disjunção Parcial" com a superclasse "Pessoa" na cauda e as subclasses na cabeça da hiperaresta.

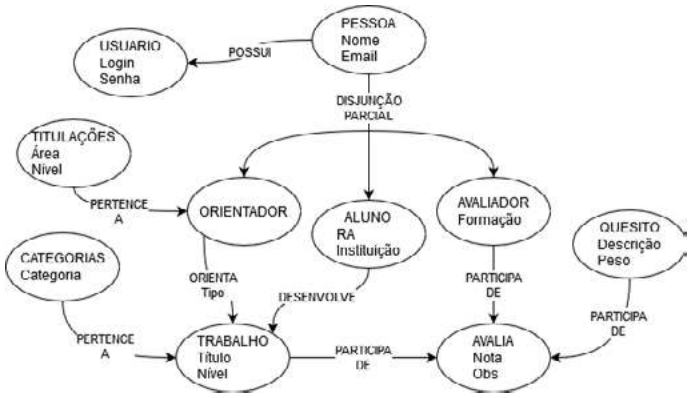

Figura 2. Grafo de esquema "Exposição de Trabalhos Científicos"

Apesar do projeto lógico do BDG apresentar uma modelagem de especializações utilizando hiperarestas, estas não ocorrem em uma eventual instância do BDG, bem como as subclasses. Nas instâncias do BDG ocorrem apenas instâncias de "Pessoa" que agregam os rótulos, as propriedades e as arestas das subclasses modeladas. Na Figura 3 as pessoas "Ana" (aluna), "Jorge" (orientador) e "Mirna" (avaliadora) desempenham papéis diferentes no BDG.

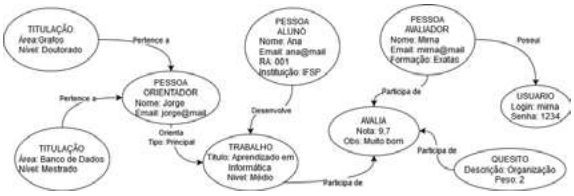

Figura 3. Instância do BDG "Exposição de Trabalhos Científicos"

6. Conclusões

Os algoritmos propostos neste artigo mapeiam o modelo conceitual EER para um grafo de esquema, porém não se consideram características específicas de nenhum SGBDG. No entanto, para fins de estudo, foram observadas algumas características básicas do SGBDG *Neo4j*. Como complementação aos algoritmos, está em fase de desenvolvimento um aplicativo que lê um diagrama EER de um arquivo XML e que gera o grafo de esquema em um arquivo XML distinto. A estrutura do grafo de esquema gerado deverá conter informações sobre restrições de integridade para serem utilizados como parâmetros de validação das eventuais instâncias do BDG. Espera-se que os grafos de esquema gerados possam ser genéricos o suficiente para serem utilizados independentemente do SGBDG e ao mesmo tempo possam agregar metadados suficientes para garantir uma validação eficiente das instâncias do BDG.

Referências

Atzeni, P., Bugiotti, F., Cabibbo, L., and Torlone, R. (2020). Data Modeling in the NoSQL World. *Computer Standards and Interfaces*, 67:103149.

Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Commun. ACM*, 13(6):377–387.

Comyn-Wattiau, I. and Akoka, J. (2017). Model Driven Reverse Engineering of NoSQL Property Graph Databases: The case of Neo4j. *2017 IEEE International Conference on Big Data (Big Data)*, pages 453–458.

Gallo, G., Longo, G., Pallotino, S., and Nguyen, S. (1993). Directed hypergraphs and applications. *Discrete applied mathematics*, 42(2-3):177–201.

Pokorny, J. (2013). NoSQL databases: a step to database scalability in web environment. *International Journal of Web Information Systems*, 9(1):69–82.

Pokorny, J., Valenta, M., and Kovacic, J. (2017). Integrity constraints in graph databases. *The 7th International Symposium on Frontiers in Ambient and Mobile Systems (FAMS 2017)*, pages 975–981.

Robinson, I., Webber, J., and Eifrem, E. (2013). *Graph databases*. "O'Reilly Media, Inc.".

Roy-Hubara, N., Rokach, L., Shapira, B., and Shoval, P. (2017). Modeling Graph Database Schema. *IT Professional*, 19(6):34–43.

Sadalage, P. J. and Fowler, M. (2013). *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Estados Unidos: Addison-Wesley Educational Publishers.