XIV EDIÇÃO JIC JORNADA DE INICIAÇÃO CIENTÍFICA E TECNOLÓGICA 14 a 16 de outubro EVENTO ON-LINE UNIVERSIDADE FEDERAL DA FRONTEIRA SUL

EXTRAÇÃO DE METADADOS DE BANCO DE DADOS SEMIESTRUTURADOS

HALLYSON CRUZ[1], GEOMAR SCHREINER[2], DENIO DUARTE[2,3]

1 Introdução

Um grafo G=(V,E) é uma estrutura de dados em que um conjunto de vértices V (*ou nodos*) são conectados por arestas E (*ou arcos*). V pode representar dados e E o relacionamento entre os dados. Essa característica permite utilizar grafos para representar diversas estruturas, como redes sociais, rotas e sistemas de recomendação. Bancos de dados orientados a grafos, como o Neo4j (Vukotic, A. et al, 2015), utilizam o modelo chamado grafo de propriedades (Angles, R. 2018) para armazenar informações detalhadas e consultar esses dados de forma eficiente através de sua linguagem de consulta (*Cypher*). No entanto, ao contrário dos bancos de dados (BDs) relacionais, os BDs orientados a grafos não possuem um esquema que define regras e restrições. A ausência de um esquema pode levar a problemas de inconsistência de dados, dificuldade na validação de dados e a falta de compreensão dos dados. Sendo assim, a extração de esquemas é uma solução para melhorar a consistência e compreensão dos dados, bem como otimizar o acesso e o armazenamento dos dados.

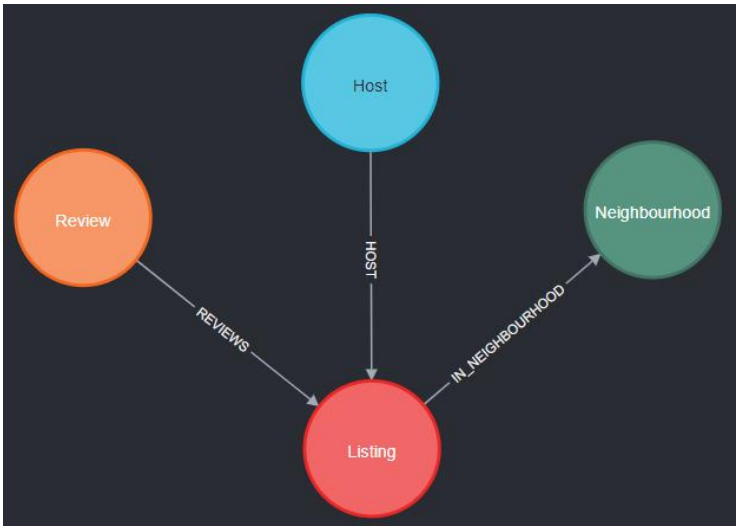

Figura 1. Esquema retornado pelo Neo4j.

[1] Graduação em Ciência da Computação, Universidade Federal da Fronteira Sul, *campus Chapecó,* contato: hallysoncruz4@gmail.com
[2] Titulação acadêmica: Doutor, instituição Universidade Federal da Fronteira Sul.
[3] Titulação acadêmica: Doutor, instituição Universidade Federal da Fronteira Sul, **Orientador**.

Apesar de o Neo4j possuir uma ferramenta capaz de extrair um esquema ("*CALL db.schema.visualization*"), o esquema gerado é simplório. Como é possível observar na Figura 1 apenas rótulos de nodos e relacionamentos são extraídos. Índices e restrições, caso existam, são também retornados. As informações de propriedades não são retornadas e não existe indicação de cardinalidade em suas relações.

2 Objetivos

O objetivo deste trabalho é desenvolver uma ferramenta para extração de esquemas com *tipos de objetos (nodos e relacionamentos), tipos de dados (atômicos e complexos), enums, restrições, cardinalidade* e *hierarquia de tipos (supertipos e subtipos)* em BDs orientados a grafos.

3 Metodologia

Inicialmente, foi realizado um estudo do estado da arte na área de extração de esquemas e para tal, artigos publicados entre 2010 e 2024 foram selecionados a partir da *string* de busca *"graph AND ("schema extraction" OR "schema discovery") AND database AND ("graph database" OR "graph schema")"* na plataforma Google Scholar. A partir disso, 210 trabalhos retornaram e sete foram selecionados. Como critério de seleção foi considerada a qualidade da técnica de extração e diferentes abordagens em comparação com o presente trabalho.

Ademais, a linguagem de programação *Python* foi utilizada para o desenvolvimento da ferramenta. Os dados extraídos foram manipulados através de dicionários, que permitem armazenar pares de chave-valor. No dicionário, cada chave é única, sendo como um identificador para um valor. Além disso, são mutáveis, sendo possível adicionar ou remover dados conforme necessário. São implementados como *tabelas hash*, o que facilita a manipulação posterior dos dados, pois a busca é eficiente.

4 Resultados e Discussão

O resultado dos estudos do estado da arte e planejamento de como seria implementada a abordagem foi implementada uma ferramenta que, dado um banco de dados **D** Neo4j de entrada, gera um esquema **S** que representa a estrutura de **D**. O formato de estrutura para o esquema extraído foi o PG-Schema proposto por (Angles, R. et al, 2023):

(tipoRótulo : nomeRótulo { [OPTIONAL] && nomePropriedade && tipoValor })

Inicialmente, a linguagem *Cypher* foi utilizada para realizar as consultas no Neo4j e então, através da linguagem de programação *Python*, foi realizada a manipulação dos objetos utilizando a estrutura de dicionário. Sendo assim, cada nodo e relacionamento do grafo é uma chave no dicionário. Como observado no código abaixo, o rótulo (tipo do nodo) *Movie* é uma chave e dentro estão armazenando as propriedades desse nodo. As propriedades também são tratadas como chave contendo os seus tipos de dados. A propriedade *tconst* tem o valor *STR (tipo do atributo)* associada a ela. Esses tipos de dados podem ser atômicos ou complexos. Por exemplo, propriedade *genres,* que é opcional (token OPTIONAL), e tem o tipo *ARRAY*, seguido do tipo de dado armazenado na lista (str), finalizando com a cardinalidade indicando o tamanho mínimo e máximo de todas as listas presente no grafo nesta propriedade.

```
(MovieType) : Movie {
        tconst STR,
        OPTIONAL genres ARRAY str (1, 3),
        OPTIONAL startYear INT,
        titleType STR,
        OPTIONAL primaryTitle STR,
        OPTIONAL runtimeMinutes INT}
```

Enums define um conjunto de valores que uma propriedade pode assumir. A abordagem implementada foi inserir todos os valores em uma lista que aceita apenas valores não repetidos, quando o tamanho da lista ultrapassa o valor máximo permitido (valor máximo assumido foi 20), a lista então é removida, e a propriedade deixa de ser *enum*. Um exemplo é dado na Figura 2.

```
Rótulo: Neighbourhood
Quantidade: 472
    neighbourhood_group:
        str, Quantidade: 230
        Is Enum: True
            Values: {'Bronx', 'Queens', 'Manhattan', 'Brooklyn', 'Staten Island'}
```

Figura 2: Exemplo de *enum* extraído do banco de dados do Airbnb.

As restrições são extraídas usando o comando *SHOW CONSTRAINT* no Neo4j. Este comando retorna uma tupla com informações sobre os nodos, relacionamentos ou propriedades que contenham alguma restrição. No PG-Schema existem dois tipos de

restrições: *MANDATORY* e *SINGLETON*. Para definir uma propriedade como única, é feita uma verificação na tupla retornada pelo Neo4j, se a constraint for "UNIQUENESS", a propriedade é *SINGLETON*. Ademais, para ser obrigatória é verificado se a quantidade total de propriedades presentes no nodo é igual a quantidade de nodos com determinado rótulo. Por exemplo, na Figura 2 a quantidade de nodos com rótulo *Neighbourhood* é 472, por outro lado o valor máximo de propriedades de nome *neighbourhood_group* são 230. Dessa forma, a conclusão seria de que a propriedade *neighbourhood_group* não é *MANDATORY*.

É possível extrair a cardinalidade dos relacionamentos através da ferramenta. Como no Neo4j as arestas (relacionamentos) são direcionais, é necessário realizar consultas através do Cypher com o objetivo de facilitar a manipulação dos dados e dessa forma extrair de maneira eficiente as cardinalidades. Consultas de nodo origem para nodo destino e de nodo destino para nodo origem são realizadas no Neo4j para todos os relacionamentos existentes no banco, sempre verificando a quantidade e então verificar se a cardinalidade é N (muitos) ou 1.

```
(:Host)-[HOSTType (1:1);(1:N)]->(:ListingType),
(:Review)-[REVIEWSType (0:N);(1:N)]->(:ListingType),
(:Listing)-[IN_NEIGHBOURHOODType (0:N);(1:1)]->(:NeighbourhoodType)
```

Figura 3: Exemplo de *cardinalidade* extraída do banco de dados do Airbnb em formato PG-Schema.

A identificação de supertipos e subtipos é um diferencial da ferramenta proposta. O Neo4j permite que nodos tenham multi-rótulos. Dessa forma, é possível que um nodo, por exemplo, contenha dois rótulos: (1) Pessoa e (2) Aluno. A ferramenta identifica o nodo supertipo e subtipo através de uma consulta que retorna a quantidade de nodos (1) e (2), sendo o nodo de maior quantidade o supertipo.

Após a extração de todas as informações do banco de dados, o dicionário é percorrido através das chaves com o propósito de marcar a flag *is_shared* em propriedades que são compartilhadas entre nodos diferentes. Exemplo na Figura 4:

```
Rótulo: Pessoa, Aluno
Quantidade: 1
    Supertipo: Pessoa
    Subtipos: Aluno
    Nome:
        str, Quantidade: 1
        is_shared: True
        Total: 1
```

Figura 4: Exemplo de extração de *supertipo e subtipos*.

5 Conclusão

O presente trabalho atingiu o objetivo de desenvolver uma ferramenta para a extração de esquemas em BDs orientado a grafos, utilizando o PG-Schema como modelo estrutural. A ferramenta foi capaz de identificar e extrair *tipos de objetos, tipos de dados, enums, restrições, cardinalidades e hierarquias de tipos*, superando as limitações de abordagens existentes, como o esquema fornecido pelo Neo4j. Testes realizados com dados do IMDB e Airbnb demonstraram a capacidade da ferramenta em lidar com grande volumes de dados, apresentando resultados conforme o esperado. O trabalho também identificou áreas que podem ser aprimoradas, como a otimização do processamento de grandes volumes de dados. Estas questões oferecem oportunidades para futuras pesquisas e desenvolvimento de uma ferramenta ainda mais robusta.

Referências Bibliográficas

Angles, R. et al. (2023). PG-Schema: Schemas for Property Graphs. In: Proceedings of the ACM on Management of Data.

DOI: https://dl.acm.org/doi/10.1145/3589778.

Vukotic, A. et al. Neo4j in action. Vol. 22. Shelter Island: Manning, 2015.

Angles, R. 2018. The property graph database model. In Proceedings of the AMW.

Palavras-chave: Neo4j; extração de esquema; grafo.

Nº de Registro no sistema Prisma: PES-2023-0182

Financiamento: Universidade Federal da Fronteira Sul