Migração de Bancos de Dados Relacionais para Grafos: Proposta e Implementação de uma Ferramenta Automatizada*

Gabriel Lisboa Conegero[1], **Carmem S. Hara**[1]

[1]Departamento de Informática, Curitiba-PR
Universidade Federal do Paraná

{gabriellisboa, carmemhara}@ufpr.br

Abstract. *The growth of unstructured and highly connected data poses challenges for data analysis in relational databases due to the high cost of finding patterns and cross-referencing data. Graph databases have emerged as a solution to these problems, as they allow these interconnections to be represented directly as edges in a graph. In order to benefit from these advantages, there is a need to migrate relational databases to graphs. There are various works and alternatives, such as the native Neo4j ETL tool. However, most of the existing methods present an indirect mapping, which includes an inference step based on concepts from the entity-relationship model. The proposal in this article aims to simplify and speed up the migration process, as well as present a mapping without changing the structure of the data. In addition, we propose the R2G2 tool, which implements the method, guaranteeing total automation of the process.*

Resumo. *O crescimento de dados não estruturados e altamente conectados impõe desafios na análise de dados em bancos relacionais. Neste modelo, encontrar padrões fundamentados nas interconexões dos dados é desafiador devido ao alto custo no cruzamento de dados. Os bancos de dados de grafos surgiram como uma solução para tais problemas, pois permitem a representação destas interconexões diretamente como arestas de um grafo. A fim de aproveitar essas vantagens, aparece a necessidade de migrar bases de dados relacionais para grafos. Existem diversos trabalhos e alternativas, como a ferramenta nativa Neo4j ETL. Entretanto, a maioria das abordagens existentes apresenta um mapeamento indireto, com etapas que incluem inferência ou uso de conceitos do modelo entidade-relacionamento. A proposta desse artigo visa simplificar e acelerar o processo de migração, além de apresentar um mapeamento sem alterar a estrutura dos dados. Além disso, propomos a ferramenta R2G2, que implementa o método, garantindo a automação total do processo.*

1. Introdução

Dentre os sistemas gerenciadores de bancos de dados, o modelo relacional é o mais comumente adotado. Existem diversos sistemas gerenciadores de bancos de dados relacionais (SGBDR), cada um com suas particularidades. Alguns são mais otimizados para desempenho com grande número de dados, como o PostgreSQL, já outros são focados na

*Este trabalho foi parcialmente financiado pelo CNPq (Processo 407644/2021-0) e pela CAPES (Programa de Excelência Acadêmica - PROEX).

simplicidade e facilidade de uso, como o SQLite. Entretanto, todos os bancos relacionais apresentam o mesmo paradigma: são bem estruturados e organizados em relações que se interligam por restrições de chave primária e chave estrangeira. Esse tipo de modelo de banco de dados é muito útil para a manipulação, agregação e ordenação de dados alfanuméricos [Codd 1970].

Com a crescente demanda de dados semi-estruturados, provinda da popularidade da Internet, nasce a necessidade de explorar relações mais complexas, como o descobrimento de padrões nas relações entre os dados. Contudo, os SGBDRs demandam muito processamento para revelar e manipular dados que possuem muitos relacionamentos. Como alternativa, o modelo de banco de dados em grafos se mostra eficiente para lidar com tal formato de dados [Vicknair et al. 2010]. Assim, muitos sistemas e aplicações que utilizam bancos relacionais poderiam se beneficiar de bancos de grafos para a exploração destes tipos de relações. Nesse contexto, surge a necessidade de métodos e ferramentas para a transformação de bases relacionais para grafos. Para realizar explorações de dados de forma eficiente, os dados podem ser carregados e organizados em sistemas gerenciadores de bancos de dados de grafos (SGBDG). Alguns SGBDGs que têm ganhado popularidade são o Neo4j e o Amazon Neptune.

Dos métodos e ferramentas de transformação existentes, destacam-se os propostos por [Đukić et al. 2024], [Virgilio et al. 2013] e a ferramenta Neo4j ETL[1]. Em todas as propostas, tuplas das relações são mapeadas em vértices dos grafos, e chaves estrangeiras, em arestas. Um destes métodos foi proposto por [Boudaoud et al. 2022]. Porém, ele é apenas uma proposta conceitual, sem uma ferramenta associada. Em outras propostas, conceitos do modelo de Entidade-Relacionamento (ER) são considerados no mapeamento. Em [Đukić et al. 2024], é utilizado o ER estendido. O método recebe como entrada quais relações correspondem a entidades especializadas, genéricas e relacionamentos muitos para muitos. Já em [Virgilio et al. 2013] e Neo4j ETL, o modelo ER é usado indiretamente. Ambas as propostas utilizam um algoritmo para inferir o esquema ER, principalmente para identificar relacionamentos muitos para muitos.

O método proposto neste artigo visa apresentar uma abordagem sistemática da migração de uma base de dados relacional para uma base de grafos, utilizando o modelo proposto por [Boudaoud et al. 2022]. A abordagem utiliza apenas informações do SGBDR, sem inferência de relacionamentos que não são representados diretamente no esquema relacional, gerando assim uma base de grafos de estrutura semelhante ao esquema da base relacional. Embora o trabalho de [Boudaoud et al. 2022] apresente uma proposta de mapeamento, nenhuma implementação e nem possíveis otimizações são exploradas pelos autores.

A abordagem sistemática apresentada neste artigo garante a simplicidade, rapidez e incrementabilidade da migração. Otimizações foram feitas para acelerar o processo de migração, como a criação de índices para a importação de vértices e criação de arestas.

Além do método, a ferramenta que o implementa, chamada de R2G2, é apresentada, juntamente com uma análise do desempenho na transformação dos dados. A ferramenta exporta as relações do SGBDR para arquivos do formato CSV e executa *scripts* que fazem o mapeamento proposto pelo método para um banco Neo4j.

[1] `https://neo4j.com/labs/etl-tool/`

O restante do artigo está organizado da seguinte forma. A Seção 2 apresenta definições preliminares necessárias para o entendimento do artigo. A Seção 3 apresenta o método de migração de uma base relacional para uma base de grafos. A Seção 4 descreve o funcionamento da ferramenta R2G2. A Seção 5 apresenta resultados experimentais usando a ferramenta R2G2, incluindo os tempos de execução para diferentes bases de dados. A Seção 6 apresenta uma análise das diferenças da proposta deste artigo com alguns trabalhos relacionados. Na Seção 7 são apresentados os trabalhos futuros e conclusões finais sobre o método e a ferramenta.

2. Definições

Esta seção apresenta definições que serão utilizadas no decorrer deste artigo.

Base de dados relacional
Uma base de dados relacional é composta por um conjunto de relações. O esquema de cada relação R é definida por uma tupla (N, A, C_p, C_e), onde: N é o nome da relação, A é um conjunto de atributos, $C_p = (N_p, A_p)$ é a chave primária de R, onde N_p é o nome da chave primária e A_p é uma tupla $(A_1, \ldots, A_n)$, onde $A_i \in A$. C_e é um conjunto de chaves estrangeiras $e = (N_e, R_e, A_e)$, onde N_e é o nome da chave estrangeira, R_e é a relação referenciada e A_e é uma tupla $(E_1, \ldots, E_n)$ que forma uma chave estrangeira, onde E_i é um elemento de A e para todo $i, 1 <= i <= n$, E_i corresponde ao atributo A_i que forma a chave primária de R_e.

Base de dados de grafos
Uma base de dados de grafos é composta por um conjunto de grafos de propriedade, nos quais vértices e arestas contém rótulos e atributos, chamados de propriedades. Dentre as funções de manipulação de grafos, podem ser destacadas duas: *MERGE* e *MATCH*. A função *MATCH* recebe um padrão de busca composto por vértices, arestas, rótulos e propriedades. Ela faz uma busca de subgrafos dentro da base de grafos, e retorna um conjunto de subgrafos que casam com o padrão. A função *MERGE* tem funcionamento semelhante à função *MATCH*; entretanto, se nenhum subgrafo for encontrado, ela estende o grafo, criando o subgrafo passado como padrão.

Preservação de Informação
Com relação ao mapeamento entre modelos, a preservação de informação é uma propriedade importante. Um mapeamento de uma base relacional para grafos preserva informação se houver um mapeamento inverso que reconstrua a base relacional original.

3. Migração relacional para grafos

Nesta seção, um método é proposto para fazer a migração de uma base de dados relacional para uma base de dados de grafos. Considera-se que a base relacional vai estar carregada em um SGBDR e a base de grafos em um SGBDG. Assume-se também, que o banco de origem seja consistente e livre de registros duplicados. O método abrange três etapas: (3.1) preparação e captura de metadados, (3.2) criação de índices e rótulos, carregamento dos vértices e geração das arestas e (3.3) limpeza do SGBDG. A Figura 1 apresenta um fluxograma do funcionamento geral do método de migração.

O método foca na migração de relações, chaves primárias e chaves estrangeiras, tendo como objetivo obter uma transformação que seja reversível, direta e incremental.

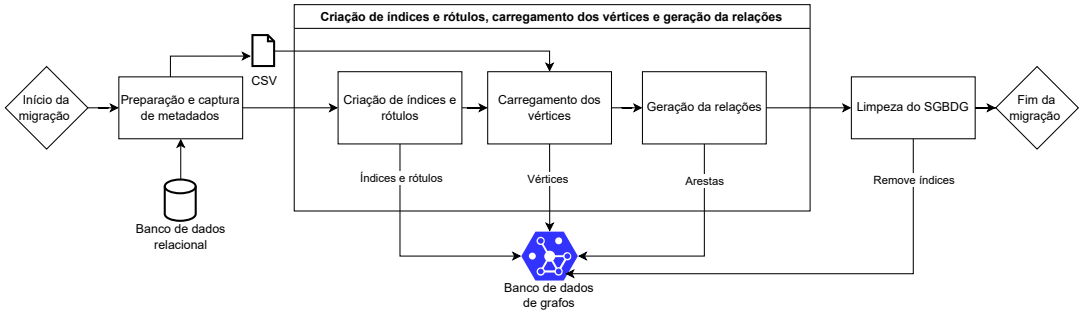


Figura 1. Fluxograma do funcionamento do método de migração de uma base de dados relacional para uma base de dados de grafos

A reversibilidade é importante para manter a propriedade de preservação de informação. Uma transformação direta garante que ela seja simples e eficiente de ser executada.

A importação incremental impede modificações ou adições de componentes ao SGBDG em execuções posteriores à migração inicial. Esse comportamento é alcançado por meio da função *MERGE*, que suprime a criação de vértices e arestas duplicados. Assim, o método atua de forma complementar, integrando novos dados ao SGBDG sem redundância.

3.1. Preparação e captura de metadados

A primeira etapa da transformação consiste em coletar os metadados do SGBDR, que compreendem: chaves primárias, chaves estrangeiras, relações, e os atributos de cada relação.

Após coletar as informações, as instâncias das relações são exportadas para arquivos CSV. Cada relação é exportada para um arquivo CSV com o mesmo nome da relação. Um atributo extra, chamado $technical\_id$, é adicionando ao esquema. O valor deste novo atributo deve ser único entre as linhas do CSV gerado. A Figura 2 apresenta o esquema relacional da base de dados retirada do *Videogames Dataset*[2], enquanto a tabela da Figura 3 apresenta um exemplo da extração dos dados para arquivos CSV da relação *game*.

3.2. Criação de índices e rótulos, carregamento dos vértices e geração das arestas

Essa etapa utiliza as informações coletadas na etapa anterior para criar os vértices do grafo e conectá-los corretamente através de arestas. A etapa 2 pode ser dividida em 4 subetapas. São elas: criar índices e rótulos, carregar vértices e gerar arestas.

Criar índices e rótulos

A primeira subetapa consiste em criar os índices e rótulos que vão melhorar o desempenho do carregamento de vértices, geração de arestas e incrementabilidade do método.

Para cada relação r é criado um rótulo com nome da relação $r.N$. Para cada rótulo, é criado um índice no SGBDG sobre os atributos que fazem parte da chave primária da relação $r.C_p.A_p$. Para relações sem chave primária deve ser criado um índice usando o atributo extra $technical\_id$. Para cada chave estrangeira é criado um índice. O índice vai ser sobre o rótulo da relação referenciada, usando os atributos referenciados $r.C_e.A_e$.

[2] `https://github.com/bbrumm/databasestar`

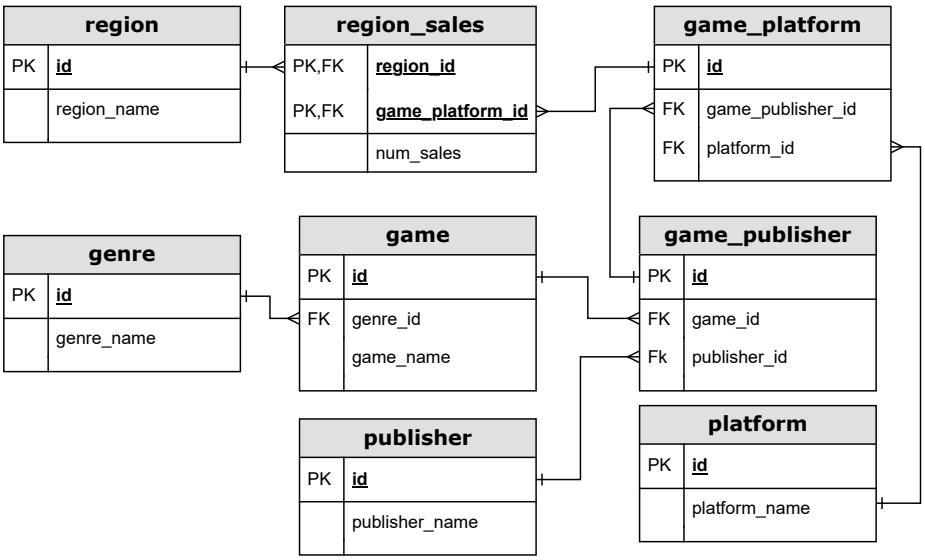


Figura 2. Esquema relacional da base de dados *Videogames Dataset***.**

technical_id	id	genre_id	game_name
1	1187	9	Call of Duty 2
2	1188	9	Call Of Duty 2: Big Red One
3	1189	9	Call of Duty 3
4	1190	9	Call of Duty 4: Modern Warfare

Figura 3. Subconjunto de vértices gerados pela relação *game*.

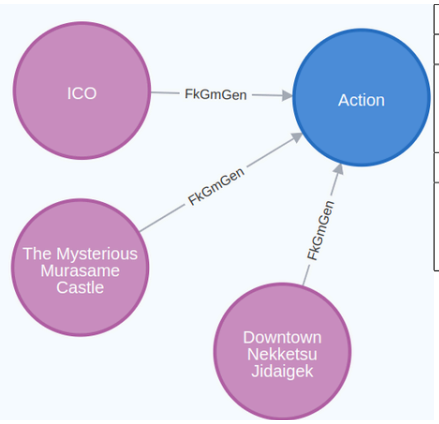
Carregar vértices

A segunda subetapa é responsável pelo carregamento dos dados para o SGBDG, utilizando os arquivos CSV gerados pela etapa anterior. Para cada tupla t da relação r deve ser criado um vértice v_t com o rótulo $r.N$ utilizando a função *MERGE*. Para cada atributo da tupla, uma propriedade de mesmo nome e valor é adicionada ao vértice v_t. A Figura 3 apresenta uma amostra dos vértices criados pela relação *game*.

Gerar arestas

A última subetapa cria as arestas utilizando as informações sobre as chaves estrangeiras coletadas na primeira etapa. Para cada chave estrangeira e da relação r, é utilizada a função *MATCH* para encontrar um vértice x com o rótulo $r.N$. Em seguida, aplica-se novamente a função *MATCH* para filtrar os vértices y de rótulo $e.R_e$ que contém as propriedades $e.A_e$ com mesmo valor da chave primária da relação referenciada $e.R_e$. Após obter os dois vértices x e y, é criada uma aresta entre eles, onde o rótulo da aresta é nome da chave estrangeira $e.N_e$.

Considere a relação *game* com a chave estrangeira e para a relação *genre*. Vamos nominar a chave de *FkGmGen*, ou seja $e.N_e = FkGmGen$. Além disso, tem-se que



game		
id	genre_id	game_name
2351	1	Downtown Nekketsu Jidaigek
4221	1	ICO
9891	1	The Mysterious Murasame Castle

genre	
id	genre_name
1	Action

Figura 4. Arestas geradas usando a chave estrangeira entre as relações *game* e *genre*.

$e.R_e = genre$ e $e.A_e = (id)$. Então para todo vértice x com rótulo $game$ vai ser criada uma aresta para um vértice y de rótulo $genre$ no qual $x.genre\_id = y.id$. A Figura 4 apresenta uma amostra dos dados da relação *game* e *genre* e como são as arestas geradas entre eles.

3.3. Limpeza do SGBDG

A última etapa consiste em remover os índices que foram usados na geração das arestas. São removidos os índices apresentados na Seção 3.1 que foram originados de uma chave estrangeira. Os índices associados a chaves primárias não serão removidos, devido a alta probabilidade de serem usados no processamento de consultas.

4. R2G2: Uma ferramenta de migração de um SGBDR para o SGDBG Neo4j

Para implementar o método descrito na Seção 3, foi criada a ferramenta R2G2[3], que automatiza o processo de migração de bancos de dados relacionais (SQLite e PostgreSQL) para o Neo4J. A ferramenta foi desenvolvida em Python, utilizando as bibliotecas Neo4j, sqlite3, psycopg2 e Jinja2. Ela tem como objetivo automatizar totalmente o processo de migração e proporcionar uma visão detalhada do processo por meio de um resumo do tempo total de execução e o número de vértices e arestas criados.

A ferramenta é composta por dois módulos principais: geração de *scripts* e arquivos CSV (4.1) e execução dos *scripts* (4.2). O funcionamento da ferramenta é simples e direta. O usuário só precisa fornecer os detalhes de acesso ao SGBDR de origem e do SGBDG de destino.

4.1. Geração de *scripts* e arquivos CSV

O primeiro módulo é responsável por gerar os *scripts* que vão realizar o mapeamento proposto na seção 3. A ferramenta em si não executa o método, mas gera código Cypher que vai realizar o mapeamento. Ele coleta as informações sobre a base de dados relacional, utilizando um *script* SQL[4]. Com essas informações, são gerados arquivos de criação

[3]Disponível em `https://github.com/gabriellisboaconegero/neo4j-adega`
[4]Os scripts utilizados foram retirados da ferramenta `https://chartdb.io/`

de vértices, arestas e índices, e para remoção de índices de chave estrangeira. Depois, usando ferramentas internas, é feita a exportação das relações para arquivos CSV. O atributo $technical\_id$ do CSV é gerado usando um número sequencial.

Exemplo de scripts gerados pela ferramenta são apresentados em Código 1. Os *scripts* de vértice utilizam o comando *LOAD CSV* para importar o arquivo CSV e criar os vértices com o comando *MERGE*. Os *scripts* de arestas, utilizam as informações sobre chaves estrangeiras para criar arestas entre vértices. Usando o comando *MATCH* duas vezes, ele itera sobre todos os vértices com um rótulo específico e cria uma aresta para outro vértice que tenha a chave primária igual à chave estrangeira. Existem dois tipos de *scripts* de índice, os de criação e os de remoção. Os de criação utilizam o comando *CREATE INDEX* e os de remoção utilizam o comando *DROP INDEX*.

Código 1. *Scripts* gerados usando a base *Videogames Dataset*

```
// ============== Script de importação de vértices ==============
LOAD CSV WITH HEADERS FROM 'file:///videogames/csv/game.csv' AS row
MERGE (x:Game {id: row.id})
ON CREATE SET
    x.game_name = row.game_name,
    x.genre_id = row.genre_id
ON MATCH SET
    x.game_name = row.game_name,
    x.genre_id = row.genre_id;

// ============== Script de criação de arestas ==============
MATCH (x:Game)
MATCH (y:Genre { id: x.genre_id })
MERGE (x)-[:FkGmGen]->(y);

// ============== Script de criação de índices ==============
CREATE INDEX fk_gpu_pub_fk_index IF NOT EXISTS
FOR (x:Publisher) ON (x.id);
```

4.2. Execução dos scripts

Esse módulo abre uma conexão com o banco Neo4j e utiliza a função fornecida pelo *driver* do Neo4j para python para executar os *scripts* criados no módulo anterior. O primeiro grupo de *scripts* a ser executados devem ser os de criar índices e depois os de vértices e arestas, assim respeitando a ordem imposta pelo método proposto. Por último, são executados os *scripts* de remoção de índices.

5. Estudo de Caso

Experimentos foram realizados utilizando a ferramenta R2G2, para migrar bases de dados do PostgreSQL para bancos Neo4j, todos rodando em Docker. O *host* tem um processador 12th Gen Intel(R) Core(TM) i5-12500H com 8GB de memória e 128 GB de disco. Foram utilizadas quatro bases de dados diferentes, com variados números de registros e chaves estrangeiras. A Tabela 1 apresenta a quantidade de tabelas, registros e chaves estrangeiras para cada base de dados utilizada.

5.1. Validação dos resultados

Para validar o processo de migração, algumas métricas foram observadas. Considere a execução da ferramenta para a base de dados *Videogames Dataset*, com o esquema da Figura 2. A Tabela 1 mostra que o número de vértices criados é igual ao número de registros da base.

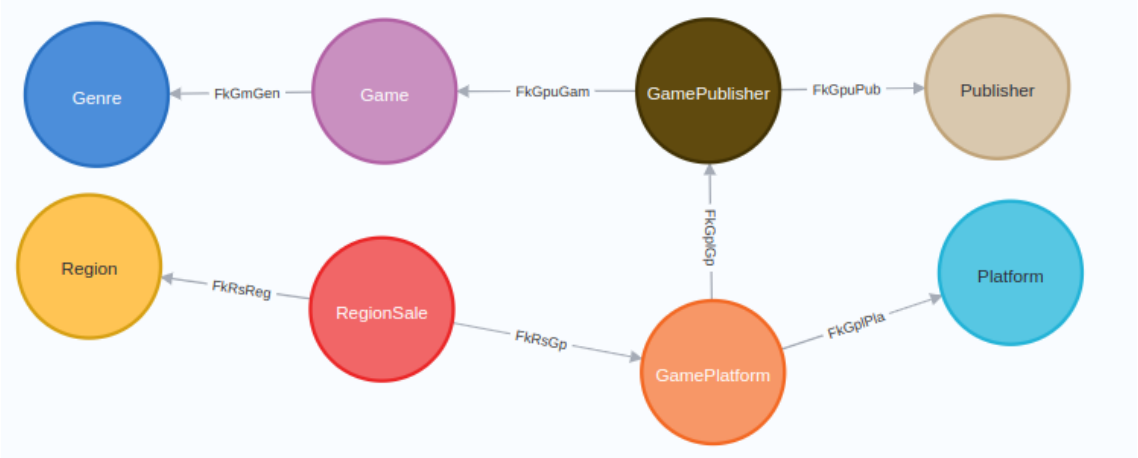


Figura 5. Resultado da consulta `CALL db.schema.visualization()` no Neo4j após a migração da base *Videogames Dataset*

Outra validação feita foi a comparação do esquema do SGBDR com o grafo gerado utilizando a consulta Cypher `CALL db.schema.visualization()`, chamada de consulta de esquema. A consulta de esquema faz parte de biblioteca interna do Neo4j. Ela gera um grafo baseado nas informações sobres arestas e vértices, como rótulos, índices e restrições. Segundo a documentação do Neo4j[5], para cada rótulo existente no banco é retornado um vértice que tem o mesmo rótulo, e como a migração mantém o número de rótulos igual ao número de relações da base de dados, então existe um mapeamento entre relações da base relacional e os vértices retornados pela consulta de esquema. As arestas retornadas são baseadas em toda combinação possível de vértice de saída e vértice de chegada de uma aresta no banco. Ou seja, se no banco Neo4j existir pelo menos uma aresta c de rótulo $L(C)$ saindo de um vértice x para um vértice y que contenham os rótulos $L(A)$ e $L(B)$, respectivamente, então há uma outra aresta no grafo do esquema com rótulo $L(C)$ ligando os vértices associados aos rótulos $L(A)$ e $L(B)$ no grafo de esquema. A Figura 5 apresenta que o único tipo de aresta entre vértices com rótulos `Game` e `Genre` são as de rótulo `FkGmGen`. Também mostra que não existem arestas entre vértices com rótulos `Platform` e `Region`. Transformando o esquema da Figura 2 em um grafo, onde os vértices são as relações e as arestas as chaves estrangeiras, vemos que o grafo gerado pela consulta de esquema e este grafo são homomorficamente equivalentes.

5.2. Análise dos resultados

Os tempos de execução dos módulos 1 e 2 da ferramenta R2G2, utilizados na migração das bases de dados, estão detalhados na Tabela 1. Observa-se que, embora a diferença no tempo de geração de *scripts* entre as bases *Movies Dataset* e *Olympics Dataset* seja de apenas 0.1226 segundos, a importação de dados para o Neo4j é 7.695 segundos mais lenta para *Olympics Dataset* (aproximadamente 48%). Essa disparidade decorre do fato de que o Módulo 1 é predominantemente influenciado pelo número de registros (diferença de 261.089 registros), enquanto o Módulo 2 é fortemente impactado pelo número de arestas (diferença de 525.716 arestas).

[5] `https://neo4j.com/docs/operations-manual/current/procedures/#procedure_db_schema_visualization`

Base de dados	Nº de tabelas	Nº de registros	Nº de chaves estrangeiras	Módulo 1	Módulo 2	Vértices criados	Arestas criadas
Gravity Dataset[6]	15	92.186	14	0.1707s	3.028s	92.188	163.386
Movies Dataset[7]	17	440.712	17	0.5019s	15.742s	440.712	879.604
Northwind Dataset[8]	14	3.362	13	0.0490s	0.595s	3.362	7.113
Olympics Dataset[9]	11	701.801	10	0.6245s	23.437s	701.801	1.405.320

Tabela 1. Resultados da análise experimental

6. Trabalhos Relacionados

Os trabalhos relacionados são classificados de acordo com a abordagem utilizada para migração de um sistema de banco de dados relacional para um sistema de banco de dados de grafos. [Silva and Mello 2021] apresentam uma abordagem focada na transformação do modelo EER (*Enhanced Entity-Relationship*) para restrições no SGBDG, onde a transformação é feita no modelo entidade-relacionamento e não no SGDBR. Ele foca em criar restrições com índices e restrições de integridade no Neo4j para recriar as restrições impostas pelo modelo EER e não em apresentar o processo de migração dos dados. [Virgilio et al. 2013], [Dimple and Bhardwaj 2015], [Đukić et al. 2024], [Bhandari and Chitrakar 2024] tem uma abordagem que utiliza informações sobre o esquema do banco de dados e o modelo ER da base de dados relacional.

[Virgilio et al. 2013] não utiliza explicitamente o modelo ER, porém apresenta uma forma de identificar relações que poderiam ser classificadas no modelo EER como relacionamentos n:m e entidades especializadas. Os outros trabalhos apresentam explicitamente a necessidade de identificar os relacionamentos n:m para continuar a transformação. [Virgilio et al. 2013] utiliza um esquema de grafos gerado do esquema da base relacional e percorre esse grafo para fazer a transformação. Em contraste, a proposta apresentada neste artigo itera pelas relações e suas chaves primárias e estrangeiras para criar os vértices e arestas.

[Đukić et al. 2024] apresenta, além de uma transformação semelhante a de [Virgilio et al. 2013] para relacionamentos n:m, uma abordagem para relacionamentos especializados, onde vértices gerados por relações especializadas contêm os rótulos da relação especializada e da relação genérica.

O trabalho de [Boudaoud et al. 2022] propõe um método conceitual para a transformação genérica de bases de dados relacionais em grafos de propriedade, com demonstrações e definições formais do mapeamento. Adicionalmente, o trabalho elabora a propriedade de preservação da informação, fornecendo provas de que o mapeamento proposto a respeita. Este artigo, aderindo a esse princípio, adota o trabalho de [Boudaoud et al. 2022] como base. No entanto, o método e a ferramenta aqui apresentados derivam de [Boudaoud et al. 2022] com uma abordagem sistemática e otimizações. Enquanto [Boudaoud et al. 2022] mapeia tuplas de relações para vértices e chaves estrangeiras para arestas em um modelo conceitual, este artigo prioriza a implementação e otimização das etapas de exportação e importação. Especificamente, a criação de índices

[6] `https://github.com/bbrumm/databasestar`
[7] `https://github.com/bbrumm/databasestar`
[8] `https://github.com/pthom/northwind_psql`
[9] `https://github.com/bbrumm/databasestar`

e a coleta de metadados são explicitamente incluídas no método proposto, diferentemente de [Boudaoud et al. 2022], que não especifica tais operações.

7. Conclusão

Este trabalho apresentou um método sistemático para a migração de bases de dados relacionais para base de dados de grafos, garantindo um mapeamento direto, reversível e incremental. A ferramenta R2G2 foi desenvolvida para automatizar essa migração, facilitando a exploração de dados altamente conectados sem a necessidade de inferência do modelo ER. Os experimentos realizados demonstraram que a abordagem proposta é viável, gerando uma representação em um grafo equivalente ao esquema do banco relacional e apresentando um desempenho otimizado por meio da criação de índices.

Como direções futuras, pretende-se realizar uma validação formal para demonstrar que o método proposto é equivalente ao método apresentado em [Boudaoud et al. 2022]. Outro aspecto a ser estudado é a exploração do caráter incremental do método, avaliando como ele pode ser utilizado para recriar transformações feitas por outros métodos a partir de sua estrutura base. Uma possível extensão deste trabalho seria explorar outros métodos de exportação e importação de dados, como importação nativa usando conectores Java para o Neo4j. Por fim, análises de desempenho e escalabilidade mais aprofundadas serão realizadas, considerando diferentes cenários e bases de dados para otimizar ainda mais a migração e garantir sua escalabilidade. Com essas melhorias, espera-se que a abordagem proposta contribua para a simplificação da transição de bancos relacionais para grafos, ampliando sua aplicabilidade.

Referências

Bhandari, H. and Chitrakar, R. (2024). Enhancement of a transformation algorithm to migrate sql database into nosql graph database. *Data Science Journal*, 23:35. Abordagem Mista.

Boudaoud, A., Mahfoud, H., and Chikh, A. (2022). Towards a complete direct mapping from relational databases to property graphs. Abordagem Conceitual.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387.

Dimple, D. and Bhardwaj, H. (2015). A tool to convert e-r diagram to property graph database. 10:23207–23221. Abordagem Mista.

Silva, T. and Mello, R. (2021). A rule-based conversion of an eer schema to neo4j schema constraints. pages 181–192. Abordagem EER.

Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y., and Wilkins, D. (2010). A comparison of a graph database and a relational database: a data provenance perspective. In *Proceedings of the 48th Annual ACM Southeast Conference*, ACMSE '10, New York, NY, USA. Association for Computing Machinery.

Virgilio, R., Maccioni, A., and Torlone, R. (2013). Converting relational to graph databases. Abordagem Mista.

Đukić, M., Pantelic, O., Pajic Simovic, A., Krstović, S., and Jejić, O. (2024). A systematic approach for converting relational to graph databases. *IPSI Transactions on Internet Research*, 20:17–28. Abordagem Mista.