Vol.2019-DBS-169 No.11
Vol.2019-IFAT-136 No.11
2019/9/10

ネイティブグラフデータベースにおける再帰的走査の効率化に向けた経路管理方法の提案

楠 和馬[1,2,a)]　波多野 賢治[3,b)]

概要：グラフは節点と関係辺で構成されるデータ構造であり，ソーシャルネットワークや知識ベース，電子商取引履歴などのデータ内に存在する関係性を取り扱うデータの表現に多く利用されている．しかしそれらグラフデータの問合せや分析には，データベースに格納されている関係性を辿る（以下，グラフ走査）ことで NP 完全問題である部分グラフ抽出を行う必要がある．これにより，データベースのデータ量に依存せずグラフ走査を一定の処理コストで行うことができるネイティブグラフデータベース (以下，GDB) においても，関係辺を辿る回数が多くなるほどグラフ走査性能がスケールしない問題が生じる．この原因の一つとして，グラフ走査において複数の関係辺を一問合せにおいて発生する際に，ネイティブ GDB では不要な節点でもグラフを展開するために一度読み込む必要がある，といった冗長さにある．したがって，複数の関係辺を辿らずに直接的に目的の節点への経路を辿ることが可能になれば，節点読込みやグラフ走査回数の少ない効率的なグラフ走査を行うことができる．そこで本稿では，データベース内に含まれている同一関係性の連なりを記録しネイティブ GDB で管理することで，効率的なグラフ走査の補助方法を提案する．

1. はじめに

近年の電子商取引（以下，EC）サイトやソーシャルネットワーキングサービス（以下，SNS）の普及により，それらサービスで取り扱うデータは膨大で複雑な関係データとして扱われている．また，関係データはグラフと呼ばれるデータ構造で表現され，そのデータ構造はデータ内で実体のある事物を表す節点と，エンティティ間に存在する関係を表す辺で構成されてる．さらにグラフは，他のデータモデルでは直接的に表現できないデータ内のリレーションを扱えるため，EC サイトにおけるトランザクションデータや SNS 上の人間関係をデータ化したソーシャルグラフなどのように多様な分野で扱われている．このような巨大で複雑なグラフデータに対して問合せや分析を行うことで，新たなビジネス機会に繋げることが可能になるが，関係性を辿る（以下，グラフ走査）ことで NP 完全問題である部分グラフ抽出を行う必要がある．そのため，グラフの問合せや分析の高速化への需要が高まっているため，多くの GDB が開発されている．

汎用的にグラフデータを扱うことが可能な GDB はプロパティグラフ（以下，PG）モデルと呼ばれるデータモデルのグラフ構造をストレージに管理している．PG モデルは節点および辺に属性を持つようなグラフデータを柔軟に表現することができるため，多様な分野におけるグラフデータに適用可能なグラフモデルである [1, 2]．また，PGQL や Cypher, GCORE [3–5] などの PG モデルをサポートする GDB 専用の問合せ言語も盛んに提案されており，GDB において PG モデルがデファクトスタンダードになっている．

グラフデータの問合せや分析には，データベースに格納されている関係性を辿ることで部分グラフ抽出を行う必要があるが，この処理は NP 完全問題と知られている．したがって，グラフの問合せおよび分析の際には，部分グラフ抽出の際に指定したグラフパタンによっては膨大な処理コストになる．これにより，データベースのデータ量に依存せずグラフ走査を一定の処理コストで行うことができるネイティブ GDB においても，関係辺を辿る回数が多くなるほどグラフ走査性能がスケールしない問題が生じる．この原因の一つに，グラフ走査において複数の関係辺を一問合せにおいて発生する際に，ネイティブ GDB では不要な節点でもグラフを展開するために一度読み込む必要がある，

[1] 同志社大学大学院文化情報学研究科
Doshisha University, Kyotanabe, Kyoto 610–0394, Japan
[2] 日本学術振興会特別研究員 DC
[3] 同志社大学文化情報学部
Doshisha University, Kyotanabe, Kyoto 610–0394, Japan
[a)] kusu@mil.doshisha.ac.jp
[b)] khatano@mail.doshisha.ac.jp

といった冗長さがある．したがって，複数の関係辺を辿らずに直接的に目的の節点への経路を辿ることが可能になれば，節点読込みやグラフ走査回数の少ない効率的なグラフ走査を行うことができる．

そこで本研究では，データベース内に含まれている同一関係性の連なりを記録しネイティブ GDB で管理することで，効率的なグラフ走査の補助方法を提案することでグラフ問合せの効率化を図る．また，本研究で提案するグラフ管理方法により，問合せ性能やスケーラビリティの向上について評価するために，ベンチマークを利用した評価実験を行う．

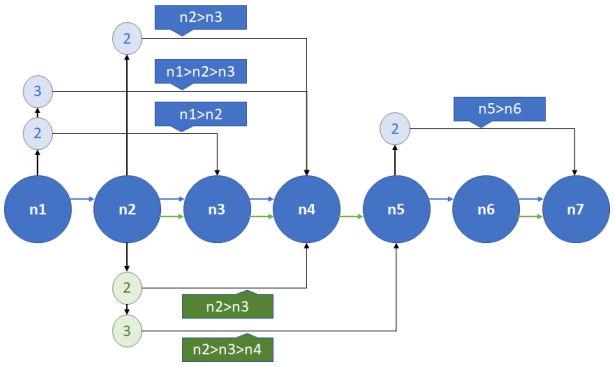

図 1　提案グラフの形状

2. 関連研究

GDB は多様な種類が存在しているが，基本的にネイティブ GDB と非ネイティブ GDB が存在している [2]．簡単にその違いと述べると，ネイティブグラフストレージ（以下，GS）に対応しているか否かであり，ストレージへのグラフデータの管理方法に違いがある．ネイティブ GS では，ストレージに隣接節点へのポインタを各節点が保持することによって，一定コストでグラフ走査を行うことができる．この性質はインデックスフリー隣接性と呼ばれており [2]，グラフ走査のコストはデータベースの量に依存しない特徴がある．また，GDB で管理可能なグラフのデータモデルには PG モデルが採用されており，PG モデルは Multi-graph，Vertex/Edge labeled-graph，Vertex/Edge attributed-graph，Directed-graph の六つの特徴を組み合わせたグラフモデルである [1]．したがって，PG モデルはデータの表現能力が高く，多くの GDB においてサポートされている．

多様な GDB の中でもより一定コストでグラフ走査可能である GDB には Neo4j がある [6, 7]．Neo4j は節点や辺のデータ長を固定にしてストレージコストを高める代わりに，各 ID が特定できればそのデータの格納位置を導出することを可能にし，グラフ要素の読込み速度も一定に行うことが可能になる [2]．しかし，さまざまな面で効率化が図られている Neo4j においても，再帰的なグラフ走査を行う際にはスケーラビリティの悪化が見られているため，グラフ走査の効率化の余地がある．

グラフ問合せを高速化させる補助は，グラフ要素の属性を利用した索引構築によるグラフ要素読込みの高速化や，グラフ走査の効率化を行うことによって実現することができる．Neo4j や他の GDB において，基本的にインデックスフリー隣接性の特性でのみグラフ走査の効率化を対応しており，他の方法については考慮されていない．したがって，単一の辺を辿る際に効果的であるが，グラフ走査の回数が多くなればコストも大きくなる．グラフ問合せにおいて，異なる関係性を組み合わせた問合せのほかに，同一の関係性を再帰的に辿る問合せも多く存在する [8]．再帰的に辿る関係性が 1 対多関係であれば，グラフ走査を行う必要のある節点数は爆発的に増加するため，そのようなグラフ走査のスケーラビリティは低くなる．

3. 提案手法

既存の GDB では再帰的に同一辺を辿る際に，辿る必要の無い節点が存在することを考慮すると，グラフ走査前に辿る必要の無い節点が把握できていれば直接的に目的節点へ到達することができる．

そこで本研究では，データベース内に含まれている同一関係性の連なりを記録しネイティブ GDB で管理することで，効率的なグラフ走査の補助方法を提案することでグラフ問合せの効率化を図る．

3.1 再帰的経路の管理方法

再帰的なグラフ走査において辿る必要の無い節点をグラフ走査の候補にならないよう省略可能にするために，各節点から再帰的に同一の関係性を辿った際に到達する節点を事前に記録し，その節点への経路をデータベースに格納する．上記の説明する内容のイメージは図 1 に示す．大きい円が表すのはデータベースに格納されているグラフデータであり，矢印が関係辺を表している．同一関係性が連続している場合は，図中の小さい円が表す節点に指定された回数の再帰的な経路を保存している．図中 `n1` は `n4` まで同一関係性の辺が連続しているため，再帰回数 2 の経路と再帰回数 3 の経路が記録されている．図中の大きい円で表現しているグラフデータから直接経路を記録しない理由としては，再帰的に辿ることのできる経路の数だけ辺を作成する必要があるため，再帰的なグラフ走査ではない場合に非効率になるためである．

本稿では，提案するグラフデータの形状を構築するために Cypher により，格納後に再帰的にグラフ走査可能な同一の関係性を問合せて，起点となる節点から図のように節点と辺を追加する．本研究ではグラフ走査を一定コストで処理可能な Neo4j を利用するため，グラフ問合せ言語には

Cypher[*1]を用いる．

3.2 再帰的経路を利用したグラフ走査

本研究で提案した同一関係性の再帰的な経路情報を利用した問合せについて説明する．Cypher において，同一関係性を再帰的に辿る際には次のように記述する．

```
()-[:RELTYPE*1..5]->()
```

この場合，関係性 RELTYPE を 1〜5 回の再帰的にグラフ走査を行う，という意味になる．本研究でグラフデータの他に格納した経路を辿ることにより，グラフ走査が可能か確認が必要なくなり効率的にグラフ走査を進めることが可能になる，と考えられる．一方，下記の場合は 3〜5 回の間で再帰的にグラフ走査を行う．

```
()-[:RELTYPE*3..5]->()
```

この場合，既存の GDB では RELTYPE の関係性を 2 回グラフ走査した後，さらにグラフ走査可能な節点が問合せの対象であるため，1〜2 回ほどグラフ走査を無駄に行う必要があるため非効率的なグラフ走査が発生する．

4. 評価実験

本節では，本研究で提案したグラフ管理方法により，問合せ性能やスケーラビリティの向上について評価するために，ベンチマークを利用した評価実験を行う．本実験環境は，CentOS 7.6.1801 (x86_64), Intel Xeon Silver 4114 CPU @ 2.20GHz, RAM 256 GB という構成で行う．また，GDB のバージョンは実験実施時点で最新版である Neo4j ver. 3.4.7 を利用する．

4.1 データセット

本研究では，非営利団体の Linked Data Benchmark Council が提供している Social Network Benchmark（以下，LDBC SNB）を用いる．LDBC は，近年におけるデータの膨大化・分散化・複雑化に伴い，要求が高まりつつあるグラフ処理および分析の性能を計測可能にすることを目的としている．LDBC SNB は，グラフを取り扱うサービスとして代表的な SNS サイトで管理されるようなデータをモデル化したものである．LDBC SNB のデータセットの形状を図 2 に示す[*2]．また，さまざまなデータ規模での問合せ性能の計測を可能にするため，Scale Factor（以下，SF）の値によってデータ量を制御することが可能になる．さらに LDBC SNB のグラフ問合せには，Interactive Workload [8] と Business Intelligence Workload [10] の 2 種類が用意されている．Interactive Workload は，サービス上でユーザが扱う情報の検索および更新に関するトランザクションのテストセットであり，その中に簡単なグラフ走査を行う Interactive Short（以下，IS）Workload が 7 種類，複雑なグラフ走査や分析を行う Interactive Complex（以下，IC）Workload が 14 種類，グラフ要素の追加・削除・更新を行う命令が 8 種類用意されている．Business Intelligence Workload は，サービス運営者がビジネスに役立つ情報を得る目的で行うようなデータ分析に関するトランザクションのテストセットであり，分析的な問合せ事例が 25 種用意されている．このように，LDBC SNB には計 54 種類のグラフ問合せが用意されている．

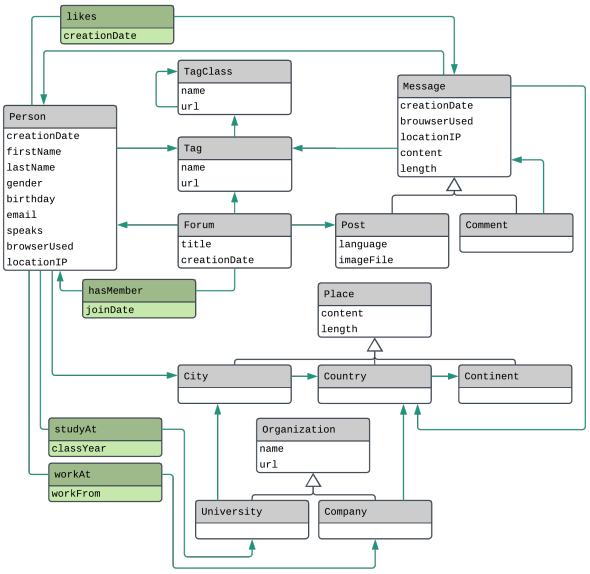

図 2 LDBC SNB データセット

表 1 グラフ問合せの特徴に関する変数

問合せ	試行回数	再帰パス最大長
IC3	15	1 - 2
IC5	15	1 - 2
IC6	15	1 - 2
IC10	15	2
IC11	15	1 - 2

本研究では，LDBC SNB のデータ生成プログラム[*3]を用いて，SF 値が 0.1 のデータセットを生成する．次に，LDBC SNB のデータ格納および問合せパラメタ生成用プログラム[*4]を用いて，データセットを Neo4j へ格納し，有効な問合せパラメタの生成を行う．本実験では，複雑なグラフパタンを問い合わせる IC Workload を利用する．また，本稿では簡易的に Neo4j へ格納したため，実行が可能な表 1 を利用する．

[*1] Neo4j: The Neo4j Cypher Manual, https://neo4j.com/docs/cypher-manual/current/ （2019 年 8 月 7 日閲覧.）

[*2] これは LDBC SNB のマニュアル [9] に掲載されている．

[*3] LDBC: ldbc_snb_datagen.git, https://github.com/ldbc/ldbc_snb_datagen （2019 年 8 月 7 日閲覧.）

[*4] LDBC: ldbc_snb_implementation.git, https://github.com/ldbc/ldbc_snb_implementation （2019 年 8 月 7 日閲覧.）

図 3　IC Workload 実行時間

4.2　実験結果

表 1 に示した LDBC SNB の IC Workload をデータの形状を変更せずに Neo4j に格納した場合と提案手法のグラフ管理方法を適用した場合のそれぞれで実行した．その結果を図 3 に示す．縦軸には IC Workload の種類と，各 Workload で 15 回クエリを実行した累積時間（秒）を表示している．横軸は適用した手法であり，左側に Neo4j に単純に格納した場合，右側には提案手法を適用した場合を表示している．IC5, 6 に対しては提案手法が効果的であったが，IC3, 10, 11 に対してはグラフ問合せの性能が非効率になる結果となった．これは，グラフ走査時に辺が存在するか確認が必要なくなった分だけ効率化ができている事例もあったが，他の非効率さが生じる事例がありその原因について調査を行う必要がある．

5.　おわりに

本研究では，データベース内に含まれている同一関係性の連なりを記録し GDB に管理することで，効率的なグラフ走査の補助方法を提案した．

評価実験では，提案手法により効率化できた事例とできなかった事例が存在したため，できなかった事例に対しても対応できるように提案手法を改善する必要がある．また，今回は格納後に提案部分のグラフを構築したが，この場合データ量が大きくなればなるほど，提案部分のグラフ構築に膨大な時間を要する．そのため，大量データ読込みを高速的に行うバルクロード時に，並行して提案部分のグラフを構築する方法について考える必要がある．

参考文献

[1] Rodriguez, M. A. and Neubauer, P.: Constructions from Dots and Lines, *Bulletin of the American Society for Information Science and Technology*, Vol. 36, No. 6, pp. 35–41 (online), DOI: 10.1002/bult.2010.1720360610 (2010).
[2] Robinson, I., Webber, J. and Eifrem, E.: *Graph Databases*, O'Reilly Media, Inc. (2015).
[3] van Rest, O., Hong, S., Kim, J., Meng, X. and Chafi, H.: PGQL: A Property Graph Query Language, *Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems*, GRADES '16, ACM, pp. 7:1–7:6 (online), DOI: 10.1145/2960414.2960421 (2016).
[4] Francis, N., Green, A., Guagliardo, P., Libkin, L., Lindaaker, T., Marsault, V., Plantikow, S., Rydberg, M., Selmer, P. and Taylor, A.: Cypher: An Evolving Query Language for Property Graphs, *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, ACM, pp. 1433–1445 (online), DOI: 10.1145/3183713.3190657 (2018).
[5] Angles, R., Arenas, M., Barcelo, P., Boncz, P., Fletcher, G., Gutierrez, C., Lindaaker, T., Paradies, M., Plantikow, S., Sequeda, J., van Rest, O. and Voigt, H.: G-CORE: A Core for Future Graph Query Languages, *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, ACM, pp. 1421–1432 (online), DOI: 10.1145/3183713.3190654 (2018).
[6] Kolomičenko, V., Svoboda, M. and Mlýnková, I. H.: Experimental Comparison of Graph Databases, *Proceedings of International Conference on Information Integration and Web-based Applications & Services*, IIWAS'13, ACM, pp. 115:115–115:124 (2013).
[7] Jouili, S. and Vansteenberghe, V.: An Empirical Comparison of Graph Databases, *Proceedings of the 2013 International Conference on Social Computing*, SOCIALCOM '13, IEEE Computer Society, pp. 708–715 (2013).
[8] Erling, O., Averbuch, A., Larriba-Pey, J., Chafi, H., Gubichev, A., Prat, A., Pham, M.-D. and Boncz, P.: The LDBC Social Network Benchmark: Interactive Workload, *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, New York, NY, USA, ACM, pp. 619–630 (online), DOI: 10.1145/2723372.2742786 (2015).
[9] Linked Data Benchmark Council: *LDBC The graph & RDF benchmark reference The LDBC Social Network Benchmark (version 0.3.1)*.
[10] Szárnyas, G., Prat-Pérez, A., Averbuch, A., Marton, J., Paradies, M., Kaufmann, M., Erling, O., Boncz, P., Haprian, V. and Benjamin, J. A.: An Early Look at the LDBC Social Network Benchmark's Business Intelligence Workload, *Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, GRADES-NDA '18, New York, NY, USA, ACM, pp. 9:1–9:11 (online), DOI: 10.1145/3210259.3210268 (2018).