DuckPGQ: Efficient Property Graph Queries in an analytical RDBMS

Daniël ten Wolde
Centrum Wiskunde & Informatica

Abstract

In the past decade, property graph databases have emerged as a growing niche in data management. Many native graph systems and query languages have been created such as Neo4j with Cypher and Tigergraph with GSQL. However, each query language has a different syntax, semantics, and capabilities. Furthermore, the functionality and performance often leave much room for improvement. The combination of these problems makes working with graph-based data cumbersome for users.

For querying graph data, two functionalities are deemed most important: *graph pattern matching* and *path-finding*. In SQL, it is possible to write queries containing graph pattern matching and path-finding, but these queries are often hard to write and understand, and inefficient to evaluate. The upcoming SQL:2023 will introduce the Property Graph Queries (SQL/PGQ) [2] sub-language, giving relational systems the opportunity to standardize graph queries and provide mature graph query functionality. The syntax of SQL/PGQ will make writing graph pattern matching or path-finding queries more intuitive to write by introducing a *visual graph syntax*.

We argue that (i) competent graph data systems must build on all technology that makes up a state-of-the-art relational system, (ii) the graph use case requires the addition to that of a many-source/destination path-finding algorithm and compact graph representation. We outline our design of DuckPGQ that follows this recipe, by adding efficient SQL/PGQ support to the popular open-source "embeddable analytics" relational database system DuckDB [4], also originally developed at CWI. Our design aims at minimizing technical debt using an approach that relies on efficient vectorized UDFs. By relying on UDFs the impact on the mainline DuckDB code base is minimized, making DuckPGQ maintainable going forward. DuckPGQ introduces bulk path-finding by adopting SIMD-friendly multi-source algorithms, as well as on-the-fly creation of CSR (Compressed Sparse Row) in-memory graph representations.

We evaluated the performance and scalability of DuckPGQ using queries from the Linked Data Benchmark Council's Social Network Benchmark (LDBC SNB) [1] for path-finding and the Labeled Subgraph Query Benchmark (LSQB) [3] for pattern matching. The benchmarks show encouraging performance and scalability on large graph data sets. Results show that DuckPGQ is able to outperform Neo4j on both pattern matching and path-finding; and is comparable to Umbra on the former.

References

[1] R. Angles, J. B. Antal, A. Averbuch, P. A. Boncz, O. Erling, A. Gubichev, V. Haprian, M. Kaufmann, J. L. Larriba-Pey, N. Martínez-Bazan, J. Marton, M. Paradies, M. Pham, A. Prat-Pérez, M. Spasic, B. A. Steer, G. Szárnyas, and J. Waudby. The LDBC social network benchmark. *CoRR*, abs/2001.02299, 2020.

[2] A. Deutsch et al. Graph pattern matching in GQL and SQL/PGQ. In *SIGMOD*, 2022.

[3] A. Mhedhbi, M. Lissandrini, L. Kuiper, J. Waudby, and G. Szárnyas. LSQB: a large-scale subgraph query benchmark. In V. Kalavri and N. Yakovets, editors, *GRADES-NDA '21: Proceedings of the 4th ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), Virtual Event, China, 20 June 2021*, pages 8:1–8:11. ACM, 2021.

[4] M. Raasveldt and H. Mühleisen. Duckdb: an embeddable analytical database. In P. A. Boncz, S. Manegold, A. Ailamaki, A. Deshpande, and T. Kraska, editors, *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 1981–1984. ACM, 2019.