SQL/PGQ Support in DataFusion

Semyon Grigorev

St. Petersburg State University

!!! 2025

Semyon Grigorev

- Research interests:
  - **Graph analysis** in the context of **graph databases**
  - **Formal language theory** in the context of **graph querying**
  - **Applied linear algebra** in the context of **graph analysis**
- Associate professor at St. Petersburg State University
- Email: rsdpisuy@gmail.com
- GitHub: gsvgit
- Google Scholar: Semyon Grigorev
- DBLP: Semyon V. Grigorev

Proposal

- To provide support of PGQ in DataFusion[1]
  - PGQ is an SQL extension to query **Property Graphs**
  - ISO standard: SQL:2023 Part 16: SQL/PGQ – Property Graph Queries
- PGQ adopters
  - Oracle
  - Google Spanner Graph
  - DuckDB
  - ...

[1] Respective issue on PGQ in DataFusion

Steps

1. Support PGQ in SQL parser[2]
2. Improve recursive queries performance[3]
    - Ideas from "On the Optimization of Recursive Relational Queries: Application to Graph Queries" by Louis Jachiet et al.
3. Translate PGQ to existing building blocks
    - It should be possible: "GQL and SQL/PGQ: Theoretical Models and Expressive Power" by Amélie Gheerbrant et al.
    - May be not the most performant solution, but the most straightforward way to the baseline
4. Investigate graph-specific techniques
    - Indexes
    - Data structures for data representation
    - Optimization techniques
5. Implement graph-specific techniques

[2] Respective issue on PGQ in sqlparser-rs

[3] Related issue on recursive queries in DataFusion with performance issues discussion

Steps

1. Support PGQ in SQL parser[2]
2. Improve recursive queries performance[3]
   - Ideas from "On the Optimization of Recursive Relational Queries: Application to Graph Queries" by Louis Jachiet et al.
3. Translate PGQ to existing building blocks
   - It should be possible: "GQL and SQL/PGQ: Theoretical Models and Expressive Power" by Amélie Gheerbrant et al.
   - May be not the most performant solution, but the most straightforward way to the baseline
4. Investigate graph-specific techniques
   - Indexes
   - Data structures for data representation
   - Optimization techniques
5. Implement graph-specific techniques

First four steps are (almost) independent

- 1 and 3 share new AST nodes types and related staff
- 3 uses results of 2
- At least, 1–4 can be stated in parallel

[2] Respective issue on PGQ in sqlparser-rs

[3] Related issue on recursive queries in DataFusion with performance issues discussion

A Bit More on PGQ to Existing Building Blocks Translation

- "Thus, LCRA[a] proposed as the relational processing engine of graph languages like Cypher and GQL is the good old RA in a slight disguise."[b]
- Moreover: "There are queries that are expressible in positive recursive SQL, and in linear Datalog, and yet are not expressible in Core GQL nor Core PGQ."[c]
- Seems that Oracle translates PGQ to SQL and use generic SQL engine[d]

[a] Linear Composition Relational Algebra

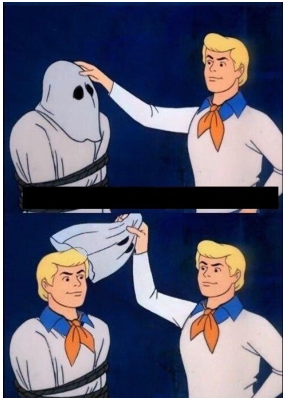
[b] "GQL and SQL/PGQ: Theoretical Models and Expressive Power" by Amélie Gheerbrant et al.

[c] Theorem 6.1 from "GQL and SQL/PGQ: Theoretical Models and Expressive Power"

[d] PGQ to SQL query transformation from "SQL Property Graphs and SQL/PGQ in Oracle Database 23ai"

A Bit More on PGQ to Existing Building Blocks Translation

- "Thus, LCRA[a] proposed as the relational processing engine of graph languages like Cypher and GQL is the good old RA in a slight disguise."[b]
- Moreover: "There are queries that are expressible in positive recursive SQL, and in linear Datalog, and yet are not expressible in Core GQL nor Core PGQ."[c]
- Seems that Oracle translates PGQ to SQL and use generic SQL engine[d]



[a] Linear Composition Relational Algebra

[b] "GQL and SQL/PGQ: Theoretical Models and Expressive Power" by Amélie Gheerbrant et al.

[c] Theorem 6.1 from "GQL and SQL/PGQ: Theoretical Models and Expressive Power"

[d] PGQ to SQL query transformation from "SQL Property Graphs and SQL/PGQ in Oracle Database 23ai"

A Bit More on Linear Algebra

- Sparse linear algebra is a promising way to high-performance graph analysis
  - GrpahBLAS — linear-algebra-based building blocks for graph analysis algorithms
    - GraphBLAS-Pointers — collection of GraphBLAS-related materials
    - SuiteSparse:GraphBLAS — reference implementation in pure C
  - FalkorDB — linear-algebra-based graph database
  - DuckDB[4] — column-oriented BD that uses matrix-based representation of graphs for PGQ
- Not only graphs: mixing of Relational and Linear Algebras is a way to analytical queries
  - TenSQL[5] — RDBMS that uses sparse linear algebra to execute SQL queries
  - "TensorTable: Extending PyTorch for mixed relational and linear algebra pipelines" by Xu Wen
  - TCUDB: Accelerating Database with Tensor Processors by Yu-Ching Hu
  - A Relational Matrix Algebra and its Implementation in a Column Store by Oksana Dolmatova

[4] "DuckPGQ:Efficient Property Graph Queries in an analytical RDBMS" by Daniel ten Wolde et al.
[5] TenSQL: An SQL Database Built on GraphBLAS by Jon Roose et al.

Qustions to Discuss

- PGQ integration ways
  - Oracle-like way: $\overbrace{PGQ \to SQL}^{\text{new AST-level transformation}} \underbrace{\to \ldots}_{\text{existing pipeline}}$
  - Alternative way: $\overbrace{SQL + PGQ}^{\text{extended existing AST}} \xrightarrow[\text{existing translator}]{\text{extended}} \underbrace{\text{logical plan} \to \ldots}_{\text{existing pipeline}}$
  - Other ideas?

Qustions to Discuss

- PGQ integration ways
  - Oracle-like way: $\overbrace{PGQ \to SQL}^{\text{new AST-level transformation}} \underbrace{\to \ldots}_{\text{existing pipeline}}$
  - Alternative way: $\overbrace{SQL + PGQ}^{\text{extended existing AST}} \xrightarrow[\text{existing translator}]{\text{extended}} \underbrace{\text{logical plan} \to \ldots}_{\text{existing pipeline}}$
  - Other ideas?

What one should we choose?

- First one is easier to implement
- Second one allows for fine-grained optimizations

Qustions to Discuss

- PGQ integration ways
  - Oracle-like way: $\overbrace{PGQ \rightarrow SQL}^{\text{new AST-level transformation}} \underbrace{\rightarrow \dots}_{\text{existing pipeline}}$
  - Alternative way: $\overbrace{SQL + PGQ}^{\text{extended existing AST}} \xrightarrow[\text{existing translator}]{\text{extended}} \underbrace{\text{logical plan} \rightarrow \dots}_{\text{existing pipeline}}$
  - Other ideas?

What one should we choose?

- First one is easier to implement
- Second one allows for fine-grained optimizations

- Linear algebra
  - Is it in the scope of te community?
  - If yes, what direction is preferable?
- Other advanced techniques for PGQ/graphs
  - Is it in the scope of te community?
  - If yes, what direction is preferable?