Integrating SQL/PGQ to DuckDB

Daniël ten Wolde, Tavneet Singh, Gábor Szárnyas, Peter Boncz
CWI Database Architectures group
SQL/PGQ

- Part of the upcoming SQL:2023 standard
- Read-only
- Graph defined in tables
- Queries can contain special syntax
  - Path-finding
  - Pattern matching
- Cheapest path is a language opportunity
Creating a Property Graph

CREATE PROPERTY GRAPH sn

VERTEX TABLES ( 
    person PROPERTIES ( personId, firstName ),
    university PROPERTIES ( universityId, name )
)

EDGE TABLES ( 
    knows SOURCE person DESTINATION person PROPERTIES ( creationDate ),
    studyAt SOURCE person DESTINATION university PROPERTIES ( studyYear )
)
SELECT gt.person1Id, gt.person2Id, gt.studyYear
FROM GRAPH_TABLE ( aml,
    MATCH
    ( p1 IS person ) -[ IS knows ]-> ( p2 IS person )
    -[ s1 IS studyAt ]-> ( u1 IS university )
    WHERE p1.firstName = 'Daniel'
    AND u1.name = 'Universiteit van Amsterdam'
COLUMNS ( p2.name
              , s1.studyYear )
) gt
Goals

- Provide an open-source implementation of SQL/PGQ
- Focus on path-finding algorithms
  - Multi-Source Breadth-First Search for shortest path
  - Batched Bellman-Ford for cheapest path
Challenges

- Graph DBMS should provide a superset of features of an RDBMS
- Efficient shortest path & cheapest path algorithms
- Many-source many-destination queries are common in SQL/PGQ
DuckDB

- Open-source in-process SQL OLAP DBMS
- SQL Parser based on PostgreSQL
  - Changes needed to support SQL/PQG queries
- Vectorized execution engine
- Support for scalar user-defined functions (UDF)
  - Parallelism useful for shortest path & cheapest path
- Allows extension modules
Compressed Sparse Row (CSR) data structure

- On-the-fly creation
- Compact structure with good locality
- Index in the **vertex array** corresponds to the id of the vertex
- Vertex array contains offsets for the edge arrays
Shortest Path

- Expensive to execute shortest path queries one-by-one
- Need for batched solution to amortize this cost
- Ability to share memory access
Multi-Source Breadth-First Search (MS-BFS)

- Batched variant developed by Manuel Then
  - Works like regular BFS, but starts from multiple nodes
- Share the memory access
  - Major bottleneck
  - Can make use of SIMD instructions (AVX-512)
Multi-Source Breadth-First Search (MS-BFS)

- Batched variant developed by Manuel Then
  - Works like regular BFS, but starts from multiple nodes

- Share the memory access
  - Major bottleneck
  - Can make use of SIMD instructions (AVX-512)

The More the Merrier: Efficient Multi-Source Graph Traversal

VLDB'14
Multi-Source Breadth-First Search (MS-BFS)

- Batched variant developed by Manuel Then
  - Works like regular BFS, but starts from multiple nodes
- Share the memory access
  - Major bottleneck
  - Can make use of SIMD instructions (AVX-512)

**Abstract**

Graph analytics on social networks, Web data, and communication networks has been widely used in a plethora of applications. Many graph analytics algorithms are based on breadth-first search (BFS) graph traversal, which is not only time-consuming for large datasets but also involves much redundant computation when executed multiple times from different start vertices. In this paper, we propose Multi-Source BFS (MS-BFS), an algorithm that is designed to have influence on others and, as a consequence, are of great importance to spread information, e.g., for marketing purposes.

In a wide range of graph analytics algorithms, including shortest path computation [1, 10], graph centrality calculation [9, 27], and k-hop neighborhood detection [2], breadth-first search (BFS)-based graph traversal is an elementary building block used to systematically explore a graph, i.e., to visit all reachable vertices and edges of the graph from a given subset of seed vertices. This makes it a common bottleneck in performance.

**The More the Merrier: Efficient Multi-Source Graph Traversal**

Manuel Then†
then@in.tum.de
Ken Pham†
kien.pham@nyu.edu

Manuel Kaufmann‡
kaufmann@in.tum.de
Alfons Kemper†
kemper@in.tum.de

Fernando Chirigati†
chirigati@nyu.edu
Thomas Neumann‡
eunemann@in.tum.de
Tuan-Anh Hoang-Vu†
tuananh@nyu.edu
Huy T. Vo†
huy.vo@nyu.edu

‡ Technische Universität München
† New York University

**BFS 1st level**

<table>
<thead>
<tr>
<th></th>
<th>B1</th>
<th>B2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>6</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>B1</th>
<th>B2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Multi-Source Breadth-First Search (MS-BFS)

- Batched variant developed by Manuel Then
  - Works like regular BFS, but starts from multiple nodes
- Share the memory access
  - Major bottleneck
  - Can make use of SIMD instructions (AVX-512)
Multi-Source Breadth-First Search (MS-BFS)

- Batched variant developed by Manuel Then
  - Works like regular BFS, but starts from multiple nodes
- Share the memory access
  - Major bottleneck
  - Can make use of SIMD instructions (AVX-512)

M S-BFS (MS-BFS), an algorithm that is designed to have influence on others and, as a consequence, are of great importance to spread information, e.g., for marketing purposes. In a wide range of graph analytics algorithms, including shortest path computation [9], graph centrality calculation [9, 27], and k-hop neighborhood detection [12], breadth-first search (BFS)-based graph traversal is an elementary building block used to systematically traverse a graph, i.e., to visit all reachable vertices and edges of the graph from a given start vertex. As the name suggests, BFS visits all vertices in the graph in a breadth-first manner.
Cheapest Path

- Batched Bellman-Ford by Manuel Then
- Can also make use of SIMD instructions
SNB Interactive Q13

- Large search space (all possible knows edges)
- MS-BFS
Results for SNB Interactive Q13

Total runtime of CSR creation + path finding for 400 substitution parameters

DuckDB v0.2.2, Intel(R) Xeon(R) CPU E5-4657L v2 @ 2.40GHz, 96 cores, 768GiB RAM, Fedora 34, NVMe SSD
Relative execution times for Query 13

Query Timings
- Creating CSR
- Other
- Pathfinding
- Precomputing
- Gathering results
- Finding path options
SNB BI Q20

- Pre-compute the edge weights
- Prune the number of nodes and edges to reduce search space
Results for SNB BI Q20

- DuckDB v0.2.2
- Intel(R) Xeon(R) CPU E5-4657L v2 @ 2.40GHz, 96 cores, 1TiB RAM, Fedora 34
Relative execution times for Query 20

Query Timings
- Creating CSR
- Other
- Pathfinding
- Precomputing
- Gathering results
- Finding path options
To conclude

- DuckDB is an ideal candidate for SQL/PGQ
  - Lightweight implementation using scalar UDFs
- Scalability of batched algorithm is promising
- Path finding can be further optimized using SIMD instructions