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 )
  )
```

Selecting from a graph table

```
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

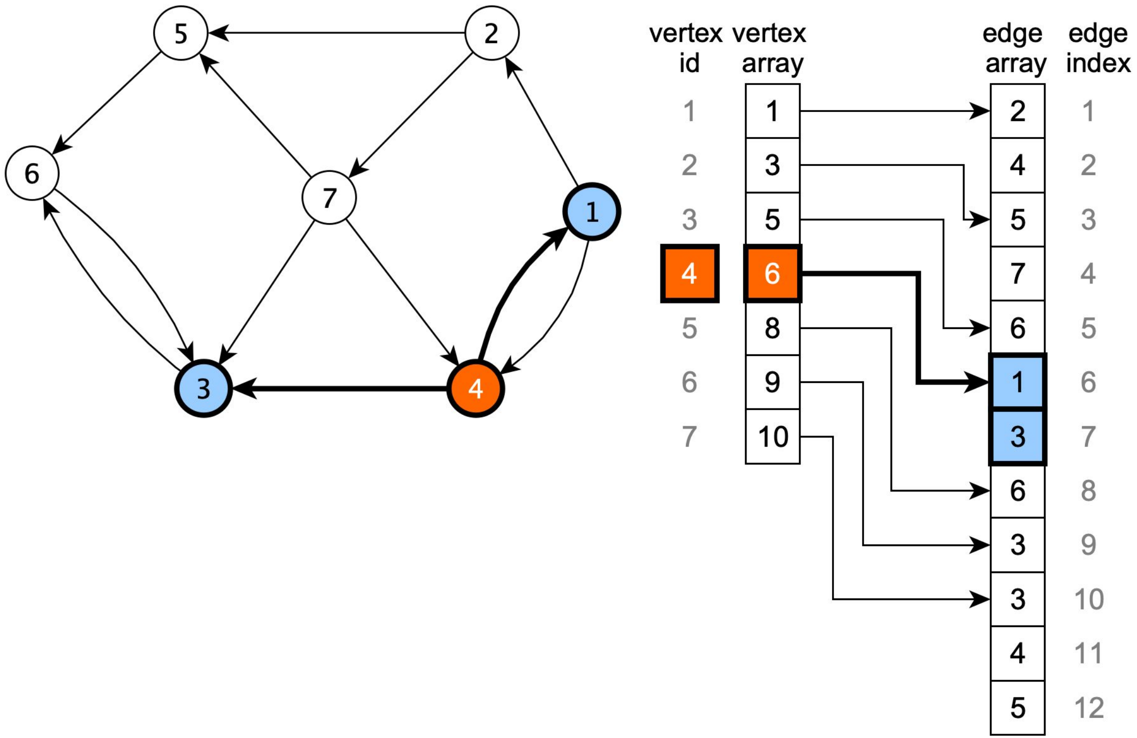
- Graph DBMS should provide a superset of features of an RDBMS
- Efficient shortest path & cheapest path algorithms
- Many-source many-destination queries are common in SQL/PGQ

DuckDB

- Open-source in-process SQL OLAP DBMS
- SQL Parser based on PostgreSQL
    - Changes needed to support SQL/PGQ 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



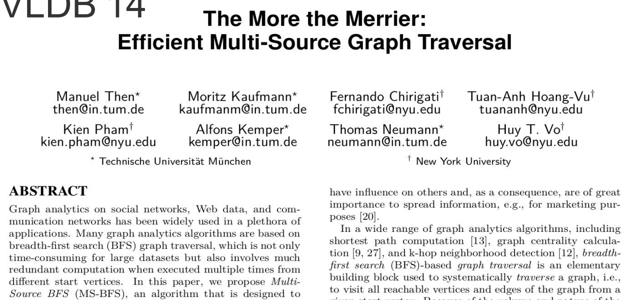
Shortest Path

- 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)

VLDB’14



The More the Merrier: Efficient Multi-Source Graph Traversal

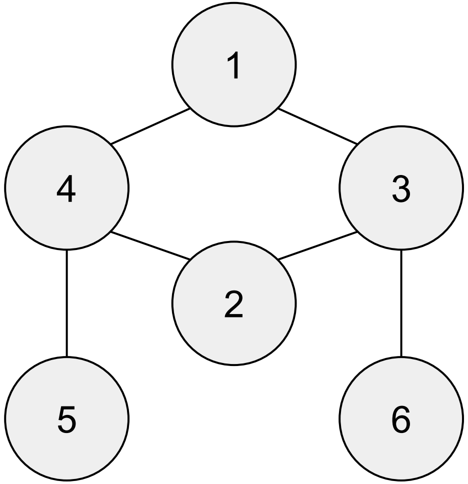
Manuel Then* then@in.tum.de
Moritz Kaufmann* kaufmanm@in.tum.de
Fernando Chirigati† fchirigati@nyu.edu
Tuan-Anh Hoang-Vu† tuananh@nyu.edu
Kien Pham† kien.pham@nyu.edu
Alfons Kemper* kemper@in.tum.de
Thomas Neumann* neumann@in.tum.de
Huy T. Vo† huy.vo@nyu.edu
* Technische Universität München
† New York University

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

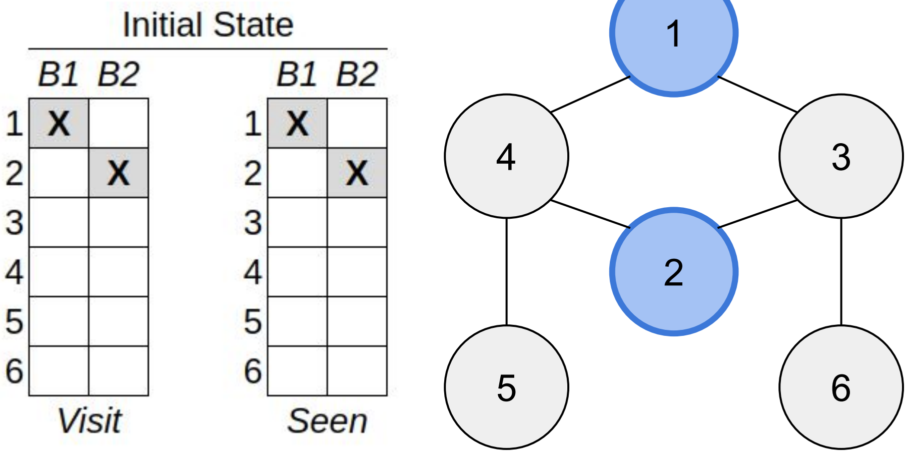
have influence on others and, as a consequence, are of great importance to spread information, e.g., for marketing purposes [20].

In a wide range of graph analytics algorithms, including shortest path computation [13], 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

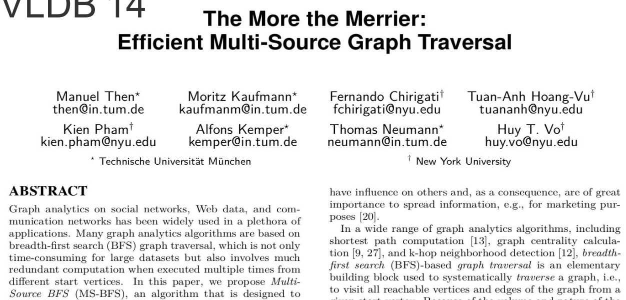


Multi-Source Breadth-First Search (MS-BFS)

- 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)



VLDB'14



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

Manuel Then* then@in.tum.de
Moritz Kaufmann* kaufmanm@in.tum.de
Fernando Chirigati† fchirigati@nyu.edu
Tuan-Anh Hoang-Vu† tuananh@nyu.edu
Kien Pham† kien.pham@nyu.edu
Alfons Kemper* kemper@in.tum.de
Thomas Neumann* neumann@in.tum.de
Huy T. Vo† huy.vo@nyu.edu

* Technische Universität München
† New York University

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

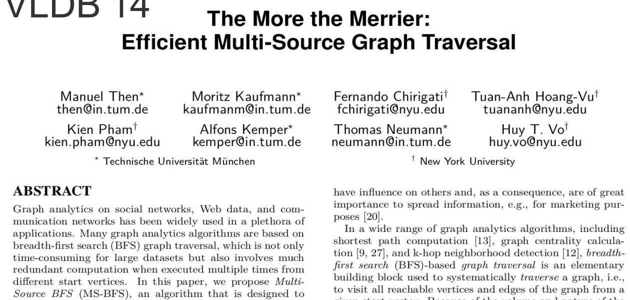
have influence on others and, as a consequence, are of great importance to spread information, e.g., for marketing purposes [20].

In a wide range of graph analytics algorithms, including shortest path computation [13], 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

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)

VLDB'14



The More the Merrier: Efficient Multi-Source Graph Traversal

Manuel Then* then@in.tum.de
Moritz Kaufmann* kaufmanm@in.tum.de
Fernando Chirigati† fchirigati@nyu.edu
Tuan-Anh Hoang-Vu† tuananh@nyu.edu
Kien Pham† kien.pham@nyu.edu
Alfons Kemper* kemper@in.tum.de
Thomas Neumann* neumann@in.tum.de
Huy T. Vo† huy.vo@nyu.edu

* Technische Universität München
† New York University

ABSTRACT

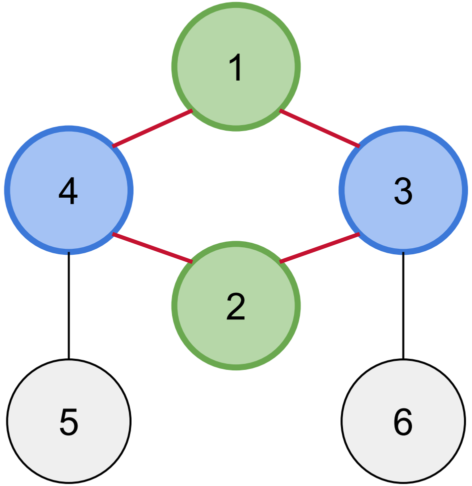
Graph analytics on social networks, Web data, and communication networks has been widely used in a plethora of applications. Many graph analytics algorithms are based on breadth-first search (BFS) graph traversal, which is not only time-consuming for large datasets but also involves much redundant computation 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 [20].

In a wide range of graph analytics algorithms, including shortest path computation [13], 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

BFS 1st level

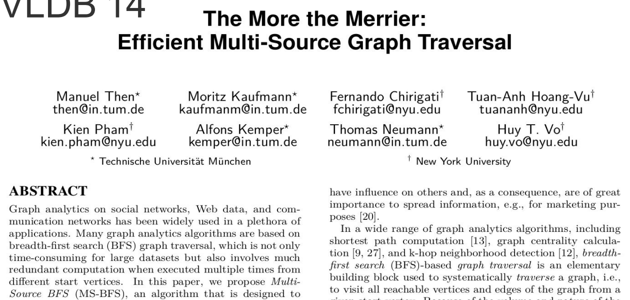
	Visit B1	Visit B2	Seen B1	Seen B2
1			X	
2				X
3	X	X	X	X
4	X	X	X	X
5				
6				



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)

VLDB’14



The More the Merrier: Efficient Multi-Source Graph Traversal

Manuel Then* then@in.tum.de
Moritz Kaufmann* kaufmanm@in.tum.de
Fernando Chirigati† fchirigati@nyu.edu
Tuan-Anh Hoang-Vu† tuananh@nyu.edu
Kien Pham† kien.pham@nyu.edu
Alfons Kemper* kemper@in.tum.de
Thomas Neumann* neumann@in.tum.de
Huy T. Vo† huy.vo@nyu.edu
* Technische Universität München
† New York University

ABSTRACT

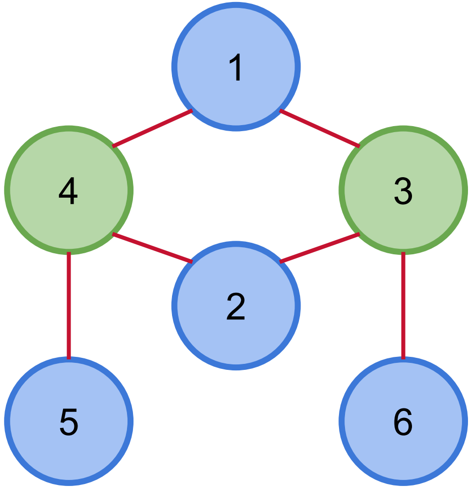
Graph analytics on social networks, Web data, and communication networks has been widely used in a plethora of applications. Many graph analytics algorithms are based on breadth-first search (BFS) graph traversal, which is not only time-consuming for large datasets but also involves much redundant computation 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 [20].

In a wide range of graph analytics algorithms, including shortest path computation [13], 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

BFS 2nd level

	B1	*B2*		*B1*	*B2*
1		X	1	X	X
2	X		2	X	X
3			3	X	X
4			4	X	X
5	X	X	5	X	X
6	X	X	6	X	X
	Visit			*Seen*	

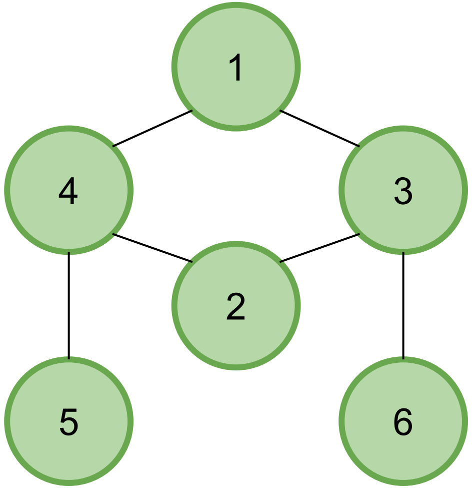


Multi-Source Breadth-First Search (MS-BFS)

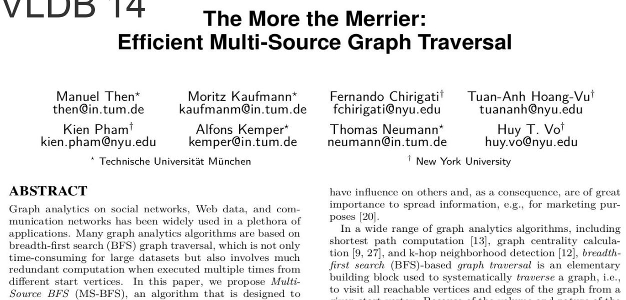
- Batched variant developed by Manuel Then
    - Works like regular BFS, but starts from multiple nodes
- Share the memory access
    - Major bottleneck
    - Can make use of SIMD instructions (AVX-512)

BFS 2nd level

	Visit B1	Visit B2	Seen B1	Seen B2
1		X	X	X
2	X		X	X
3			X	X
4			X	X
5	X	X	X	X
6	X	X	X	X



VLDB'14



The More the Merrier:
Efficient Multi-Source Graph Traversal

Manuel Then* then@in.tum.de
Moritz Kaufmann* kaufmanm@in.tum.de
Fernando Chirigati† fchirigati@nyu.edu
Tuan-Anh Hoang-Vu† tuananh@nyu.edu
Kien Pham† kien.pham@nyu.edu
Alfons Kemper* kemper@in.tum.de
Thomas Neumann* neumann@in.tum.de
Huy T. Vo† huy.vo@nyu.edu
* Technische Universität München
† New York University

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 [20].

In a wide range of graph analytics algorithms, including shortest path computation [13], 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

Cheapest Path

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

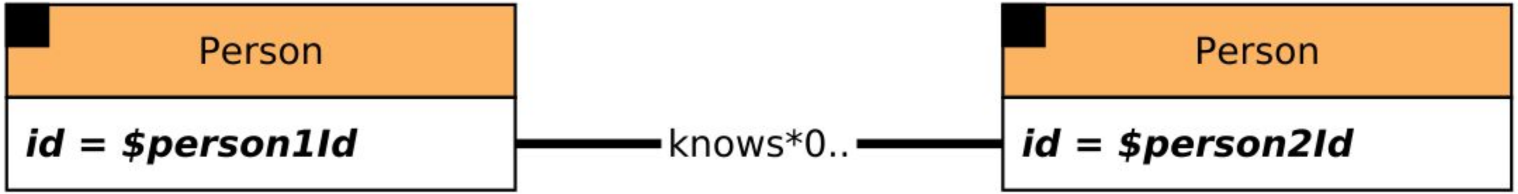
BTW'17

Efficient Batched Distance and Centrality Computation in Unweighted and Weighted Graphs

Manuel Then,[1] Stephan Günnemann,[2] Alfons Kemper,[3] Thomas Neumann[4]

Abstract: Distance and centrality computations are important building blocks for modern graph databases as well as for dedicated graph analytics systems. Two commonly used centrality metrics are the compute-intense closeness and betweenness centralities, which require numerous expensive shortest distance calculations. We propose batched algorithm execution to run multiple distance and centrality computations at the same time and let them share common graph and data accesses. Batched execution amortizes the high cost of random memory accesses and presents new vectorization potential on modern CPUs and compute accelerators. We show how batched algorithm execution can be leveraged to significantly improve the performance of distance, closeness, and betweenness centrality calculations on unweighted and weighted graphs. Our evaluation demonstrates that batched execution can improve the runtime of these common metrics by over an order of magnitude.

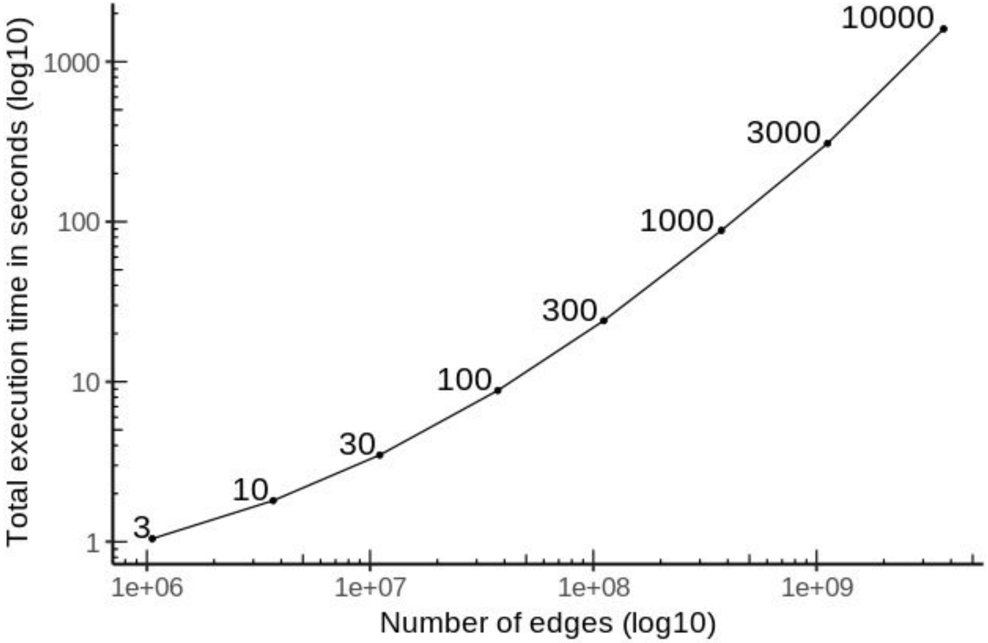
SNB Interactive Q13



- Large search space (all possible knows edges)
- MS-BFS

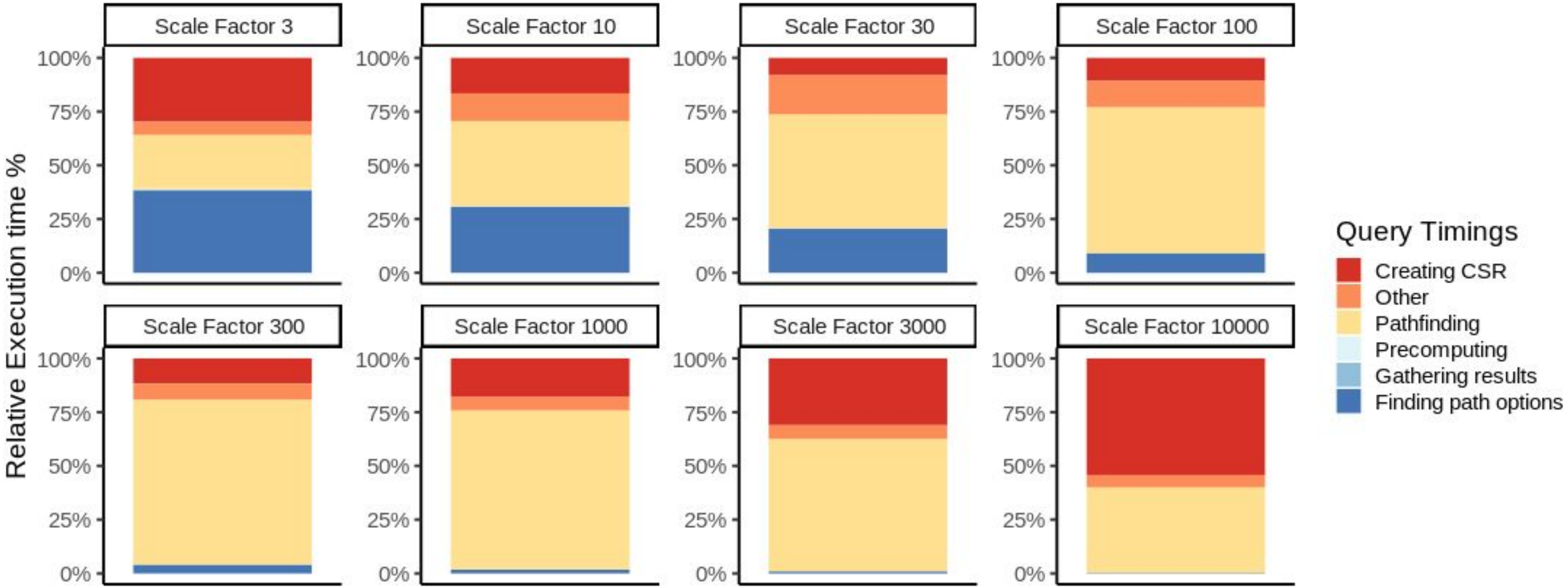
Results for SNB Interactive Q13

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

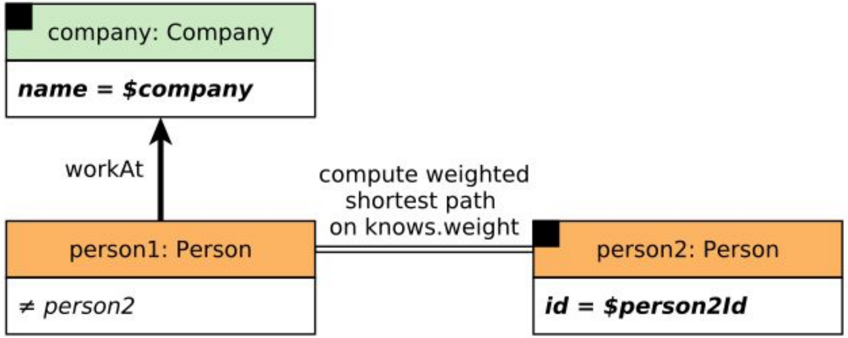


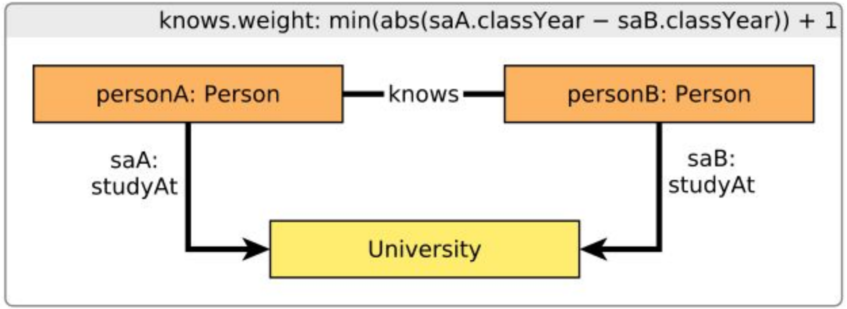
DuckDB v0.2.2, Intel(R) Xeon(R) CPU E5-4657L v2 @ 2.40GHz, 96 cores, 768GiB RAM, Fedora 34, NVMe SSD

Relative execution times for Query 13



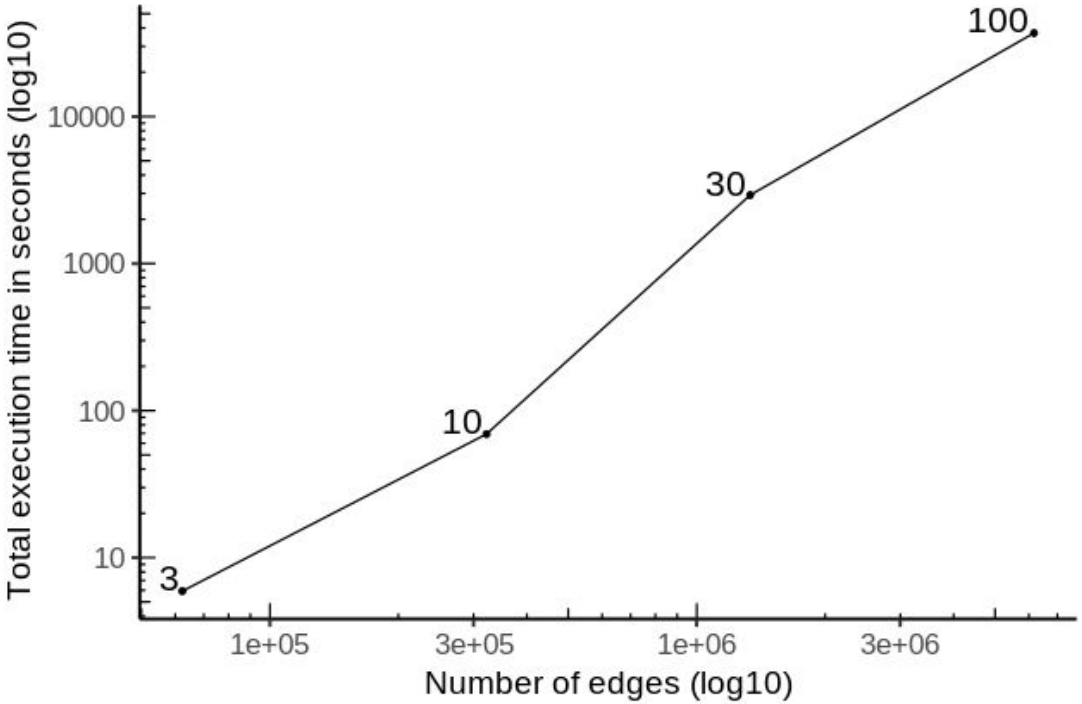
SNB BI Q20





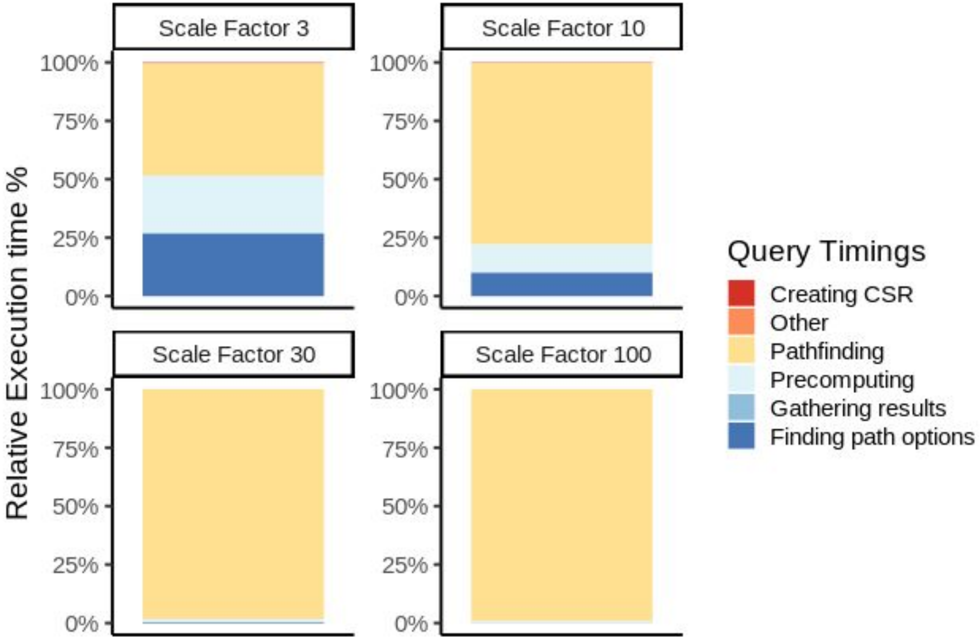
- Pre-compute the edge weights
- Prune the number of nodes and edges to reduce search space

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



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