Shortest path extensions in MonetDB

Dean De Leo, Peter Boncz
CWI, Amsterdam

9th TUC meeting in Walldorf, Germany
9th and 10th of February 2017

Outline

- Introduction
- Description of the SQL extension
- Implementation details
- Conclusions

Introduction

- Part of the Dutch project COMMIT/ “Graphalyzing4Security”

- Objectives for CWI:
  - Extend MonetDB to perform path traversals and compute weighted shortest path expressions
  - Evaluate the final product over the LDBC SNB IW benchmark

Graphs and reachability

- Assume the tables tbl(*id*, …) and edges(e_{from}, e_{to}, …)
- Let $G(N, E) = G(e_{from} \cup e_{to}, \{< e_{from}, e_{to} >\})$
- Are two entities connected?

```
SELECT v1.*, v2.*
FROM tbl v1, tbl v2
WHERE v1.id REACHES v2.id OVER edges EDGE (e_from, e_to)
```

Example – Reachability

Persons		
ID	**Name**	**Surname**
933	Mahinda	Perera
1063	Gustavo	Arbelaez
1129	Carmen	Lepland
1132	Wei	Chen
…		

Friends		
ID1	**ID2**	**CreationDate**
933	1063	2010-03-13T06:35:35.929+0000
1063	933	2010-03-13T06:35:35.929+0000
1063	1129	2010-04-30T23:42:24.105+0000
1129	1063	2010-04-30T23:42:24.105+0000
…		

>> Is Mahinda Perera connected to Carmen Lepland?

```
SELECT *
FROM persons p1, persons p2
WHERE p1.name = 'Mahinda' and p1.surname = 'Perera'
  AND p2.name = 'Carmen' and p2.surname = 'Lepland'
  AND p1.id REACHES p2.id OVER friends EDGE (id1, id2)
```

Example – Reachability

Persons		
ID	**Name**	**Surname**
933	Mahinda	Perera
1063	Gustavo	Arbelaez
1129	Carmen	Lepland
1132	Wei	Chen
…		

Friends		
ID1	**ID2**	**CreationDate**
933	1063	2010-03-13T06:35:35.929+0000
1063	933	2010-03-13T06:35:35.929+0000
1063	1129	2010-04-30T23:42:24.105+0000
1129	1063	2010-04-30T23:42:24.105+0000
…		

>> Was Mahinda Perera connected to Carmen Lepland before 01/04/2010?

```
WITH f AS ( SELECT * FROM friends WHERE creationDate < '2010-04-01' )
SELECT *
FROM persons p1, persons p2
WHERE p1.name = 'Mahinda' and p1.surname = 'Perera'
  AND p2.name = 'Carmen' and p2.surname = 'Lepland'
  AND p1.id REACHES p2.id OVER f EDGE (id1, id2)
```

Shortest paths

- What is the minimum distance between two entities?

```
SELECT t1.*, t2.*, CHEAPEST SUM (e: expr) AS cost
FROM tbl t1, tbl t2
WHERE t1.id REACHES t2.id OVER edges e EDGE (e_from, e_to)
```

Shortest paths (II)

- What is the shortest path between two entities?

```
SELECT t1.*, t2.*, CHEAPEST SUM (e: expr) AS (cost, path)
FROM tbl t1, tbl t2
WHERE t1.id REACHES t2.id OVER edges e EDGE (e_from, e_to)
```

Shortest paths (II)

- What is the shortest path between two entities?

```
SELECT tmp.*, path.*
FROM (
      SELECT t1.*, t2.*, CHEAPEST SUM (e: expr) AS (cost, path)
      FROM tbl t1, tbl t2
      WHERE t1.id REACHES t2.id OVER edges e EDGE (e_from, e_to)
) AS tmp FLATTEN path
```

Example – Shortest path

Roads			
Segment1	**Segment2**	**Distance**	**MaxSpeed**
1	2	100	80
2	4	200	80
1	3	220	130
3	4	110	90
…			

>> What is the *fastest* path from segment 1 to 4 ?

```
WITH q AS (
SELECT p1.id AS id1, p2.id AS id2,
      CHEAPEST SUM ( distance / ( maxspeed / 3600) )
                              AS (time, path)
FROM   places p1, places p2
WHERE  p1.id = 1 AND p2.id = 4
  AND  p1.id REACHES p2.id
                OVER roads EDGE (segment1, segment2)
)
```

id1	id2	time	path			
			Segment1	Segment2	Distance	MaxSpeed
0	1	9938	1	3	220	130
			3	4	110	90

Example – Shortest path (2)

Roads			
Segment1	**Segment2**	**Distance**	**MaxSpeed**
1	2	100	80
2	4	200	80
1	3	220	130
3	4	110	90
…			

>> What is the *fastest* path from segment 1 to 4 ?

```
WITH q AS (…)
SELECT *
FROM q FLATTEN q.path
```

id1	id2	time	Segment1	Segment2	Distance	MaxSpeed
0	1	9938	1	3	220	130
0	1	9938	3	4	110	90

Notes

- The graph is implicitly created
- All entities are *table expressions* or nested tables (path)
- Many to many weighted shortest paths

Some limitations:

- No possibility to specify path dependent conditions
- In some cases the semantic is nontrivial:

```
SELECT CHEAPEST SUM(e: weight) AS cost
  FROM v1, v2
 WHERE v1.x REACHES v2.y OVER edges e EDGE (e_from, e_to)
    OR v1.m = v2.n
```

Current state

- Working prototype based on MonetDB
- Reachability and minimum distance implemented
- While current work is to support paths (nested tables)

Implementation sketch

- Map the attributes to values in $\{0, \dots, |N| - 1\}$
- Build on-the-fly the graph G(N, E)
- Execute the shortest path operator(s)

Graph representation

1. Sort the edge table according to E0
2. Perform a prefix sum on E0

E0	E1	E_{cost}
0	3	1000
1	2	2000
0	1	6000
0	2	1000
1	0	1000
1	0	2000
1	3	4000
2	1	6000

E0	E1	e_{cost}
0	3	1000
0	1	6000
0	2	1000
1	2	2000
1	0	1000
1	0	2000
1	3	4000
2	1	6000

E0	E1	e_{cost}
3	3	1000
7	1	6000
8	2	1000
8	2	2000
	0	1000
	0	2000
	3	4000
	1	6000

Shortest paths & join

- Use the BFS / Dijkstra algorithm to compute the shortest path(s) while joining the connected tuples
- Project back the results

Future work

- Deterministic paths
- Graph indices
- Being resiliant to updates

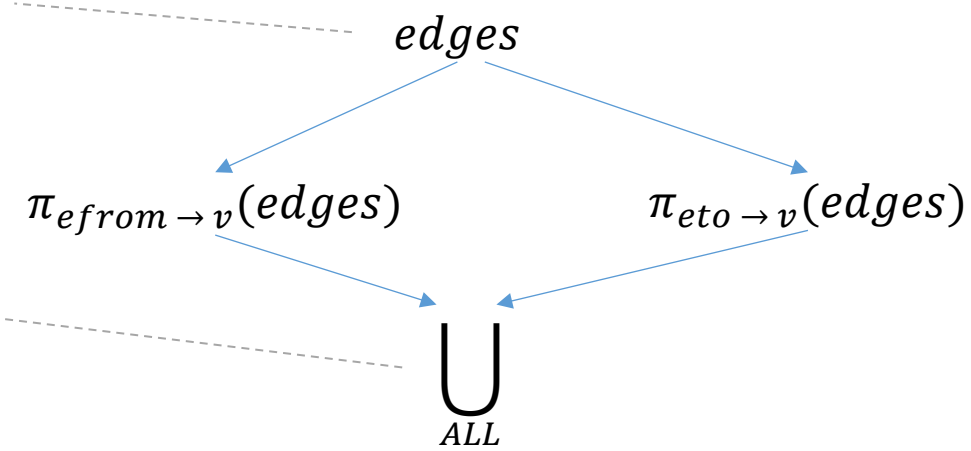
Conclusions

- SQL extension to compute shortest paths over an (implicitly defined) graph
- No evaluation results yet, sorry!
- Implementation to handle paths needs to be completed
- Future work

From attributes to vertices ID…

```
SELECT t1.*, t2.*, CHEAPEST SUM (e: e_cost) AS cost
FROM tbl t1, tbl t2
WHERE t1.id REACHES t2.id OVER edges e EDGE (e_from, e_to)
```

e_{from}	e_{to}	e_{cost}
10	20	1000
20	30	2000



v
10
20
20
30

D

v	group_id
10	0
20	1
30	2

E

E0	E1	e_{cost}
0	1	1000
1	2	2000

v	#group
10	0
20	1
20	1
30	2

From attributes to vertices ID (2)

```
SELECT t1.*, t2.*, CHEAPEST SUM (e: e_cost) AS cost
FROM tbl t1, tbl t2
WHERE t1.id REACHES t2.id OVER edges e EDGE (e_from, e_to)
```