lab03 bonus

Zacharczuk Jakub

October 2021

1 BFS do Panama Papers. Pierwsze podejście:

```
        WITH RECURSIVE generate_path (queue_it, visited, queue) AS (
        SELECT  1 AS queue_it,
                ARRAY[0] AS visited,
ARRAY[start_id] AS queue
        FROM edge
        WHERE start_id = 12126782 AND end_id = 10152535
      UNION ALL (
          (WITH generate_path(queue_it, visited, queue) AS (TABLE generate_path)
        SELECT queue_it + 1, visited || queue[queue_it], queue || (
SELECT ARRAY(
     SELECT e.nd
    FROM generate_path gp, (SELECT start_id AS st, end_id AS nd
 FROM edge
 UNION ALL (
SELECT end_id AS st, start_id AS nd
FROM edge)) e
     WHERE gp.queue[queue_it] = e.st AND NOT e.nd = ANY(visited)
               AND NOT e.nd = ANY(queue)
  )
)
        FROM generate_path gp
        WHERE queue_it < 200
      ))
)
SELECT queue_it, cardinality(visited), cardinality(queue)
FROM generate_path
LIMIT 200;


Time: 4845.826 ms (00:04.846)
```

W powyższym przykładzie bardzo nieefektywnie, pobieramy pierwszy element z kolejki, kopiując ją i po każdym przejściu powiększając ją.

2 Tu jest znacznie lepiej:

```
        WITH RECURSIVE generate_path (queue_it, visited, queue) AS (
        SELECT  1 AS queue_it,
                ARRAY[0] AS visited,
ARRAY[start_id] AS queue
        FROM edge
        WHERE start_id = 12126782 AND end_id = 10152535
      UNION ALL (
          (WITH generate_path(queue_it, visited, queue) AS (TABLE generate_path)
        SELECT queue_it + 1, visited || queue, (
SELECT ARRAY(
     SELECT e.nd
    FROM generate_path gp, (SELECT start_id AS st, end_id AS nd
 FROM edge
 UNION ALL (
SELECT end_id AS st, start_id AS nd
FROM edge)) e
     WHERE e.st = ANY(queue) AND NOT e.nd = ANY(visited)
               AND NOT e.nd = ANY(queue)
  )
)
        FROM generate_path gp
      ))
)
SELECT queue_it, cardinality(visited), cardinality(queue)
FROM generate_path
LIMIT 6;

 queue_it | cardinality | cardinality
----------+-------------+-------------
        1 |           1 |           1
        2 |           2 |           2
        3 |           4 |           3
        4 |           7 |        2539
        5 |        2546 |        2659
        6 |        5205 |        8613
(6 rows)

Time: 22849.114 ms (00:22.849)
```

3 Uniknięcie konieczności łączenia tabeli krawędzi

W celu przyspieszenia utworzyłem nową tabelę w bazie danych:

```
    CREATE TABLE undirected_edges as (
        SELECT start_id, end_id
        FROM edge
        UNION ALL (
            SELECT end_id, start_id
            FROM edge
            )
        );
```

```
    WITH RECURSIVE generate_path (depth, visited, queue) AS (
        SELECT  0 AS depth,
                ARRAY[0] AS visited,
ARRAY[node_id] AS queue
        FROM officer
        WHERE node_id = 12126782
      UNION ALL (
        (WITH generate_path(depth, visited, queue) AS (TABLE generate_path)
        SELECT depth + 1, visited || queue, (
     SELECT ARRAY(
         SELECT e.end_id
        FROM generate_path gp, undirected_edges e
       WHERE e.start_id = ANY(queue) AND NOT e.end_id = ANY(visited) AND NOT e.end_id = ANY(queue)
       GROUP BY e.end_id
       )
     )
        FROM generate_path gp
      ))
)
SELECT depth, cardinality(visited) AS visited_nodes, cardinality(queue) AS queue_size
FROM generate_path
LIMIT 6;
 depth | visited_nodes | queue_size
-------+---------------+------------
     0 |             1 |          1
     1 |             2 |          2
     2 |             4 |          3
     3 |             7 |       2539
     4 |          2546 |       2155
     5 |          4701 |       7098
(6 rows)

Time: 20435.161 ms (00:20.435)
```

4 To jest najlepsze co mi się udało osiągnąć: (znajduje się też w pliku query.txt)

```
    WITH RECURSIVE generate_path (depth, visited, queue) AS (
        SELECT  0 AS depth,
                ARRAY[0] AS visited,
ARRAY[node_id] AS queue
        FROM officer
        WHERE node_id = 12126782
      UNION ALL (
        SELECT depth + 1, visited || queue, (
SELECT ARRAY(
     SELECT e.end_id
    FROM unnest(gp.queue) elem_id
INNER JOIN undirected_edges e
ON elem_id = e.start_id AND NOT e.end_id = ANY(visited)
  GROUP BY e.end_id
  )
)
        FROM generate_path gp
      )
)
SELECT depth, cardinality(visited) AS visited_nodes, cardinality(queue) AS queue_size
FROM generate_path
LIMIT 6;
```

```
 depth | visited_nodes | queue_size
-------+---------------+------------
     0 |             1 |          1
     1 |             2 |          2
     2 |             4 |          3
     3 |             7 |       2539
     4 |          2546 |       2155
     5 |          4701 |       7098
(6 rows)

Time: 13643.054 ms (00:13.643)
```

5 Plan powyższego zapytania znajduje się w pliku plan.txt

6 Wnioski

Najwięcej czasu zajmuje przeglądanie czy element jest już w kolejce oraz czy był już odwiedzony. Nie wymyśliłem sprytnego sposobu jak można byłoby powyższe zapytanie przyspieszyć.