Advanced SQL
Forum: `https://forum-db.informatik.uni-tuebingen.de/c/ss20-asql`

Assignment 8

Relevant videos: up to #51
`https://tinyurl.com/AdvSQL-2020`
Submission: Tuesday, 30.06.2020, 10:00 AM

Please note that all tasks below have to be solved using *recursive queries* only.

1. [10 Points] **Tree Labels**

Consider the following definition of table `trees`. The table holds trees and is similarly defined in exercise 3 of assignment #5.

```
CREATE TABLE trees (
  tree     int PRIMARY KEY,
  parents  int[],
  labels   text[]
);
```

The following recursive SQL query traces a path to the root of every tree in `trees` starting from label `'f'`, if it exists:

```
WITH RECURSIVE
paths(tree, pos, node) AS (
  SELECT t.tree, 0 AS pos, array_position(t.labels, 'f') AS node
  FROM   trees AS t
    UNION
  SELECT t.tree, p.pos + 1 AS pos, t.parents[p.node] AS node
  FROM   paths AS p, trees AS t
  WHERE  p.tree = t.tree AND p.node IS NOT NULL
  -- avoid infinite recursion once we reach the root
)
SELECT p.tree, p.pos, p.node
FROM   paths AS p
WHERE  p.node IS NOT NULL
ORDER BY p.tree, p.pos;
```

Your task is to adapt the query above such that label `'f'` may occur more than once in a tree. You will have to extend the result with a new column `path_id` which uniquely identifies each path produced in this way.

Hint: Consider using `array_positions(...)`[1] to generate the mentioned path identifiers.

[1] `https://www.postgresql.org/docs/12/functions-array.html#ARRAY-FUNCTIONS-TABLE`

2. [10 Points] **Fibonacci**

Complete the following set-returning user-defined function `fib(n numeric)`:

```
CREATE FUNCTION fib(n numeric)
RETURNS TABLE(i numeric, "fib(i)" numeric) AS $$
  -- Your (recursive) CTE here
$$ LANGUAGE SQL;
```

Assume that `n` is always a natural number and that `n` ≥ 0. Calling `fib(n)` produces the Fibonacci sequence[2] up to `n` starting with 0. For example, `fib(7)` produces the following result:

i	fib(i)
0	0
1	1
2	1
3	2
4	3
5	5
6	8
7	13

3. [10 Points] **Travel by Car**

You decide to travel by car to various cities in Germany starting from *Saarbrücken*. You know your car can hold **up to 100 units of fuel** and **each unit of distance traveled costs you one unit of fuel**. Therefore, should your fuel run out of fuel before you can reach a city with a refueling station, you cannot travel any further. Luckily, some cities allow for refueling.

Based on the roadmap provided in the SQL file `travel.sql`, formulate a SQL query which lists every city reachable with your car.

Example: Consider this simplified roadmap seen here (in Figure 1):

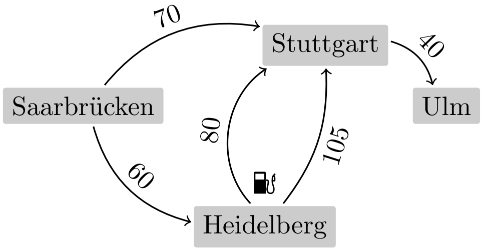

Figure 1: Road network with travel distances between cities, locations of fueling stations (⛽).

reachable
Saarbrücken
Stuttgart
Heidelberg

[2] https://oeis.org/A000045