GTE (CTE)
Gemeenschappelijke Tabel Expressies

wim.bertels@ucll.be

Wat

- Vergelijkbaar met subqueries in the from,
  maar met extras.
- Soms ook with queries genoemd, meestal cte in het engels

- https://www.postgresql.org/docs/current/interactive/queries-with.html

Dummy syntax voorbeeld

- SELECT *
  FROM (**SELECT ***
  FROM spelers) AS spelers_sub;

- /*of*/

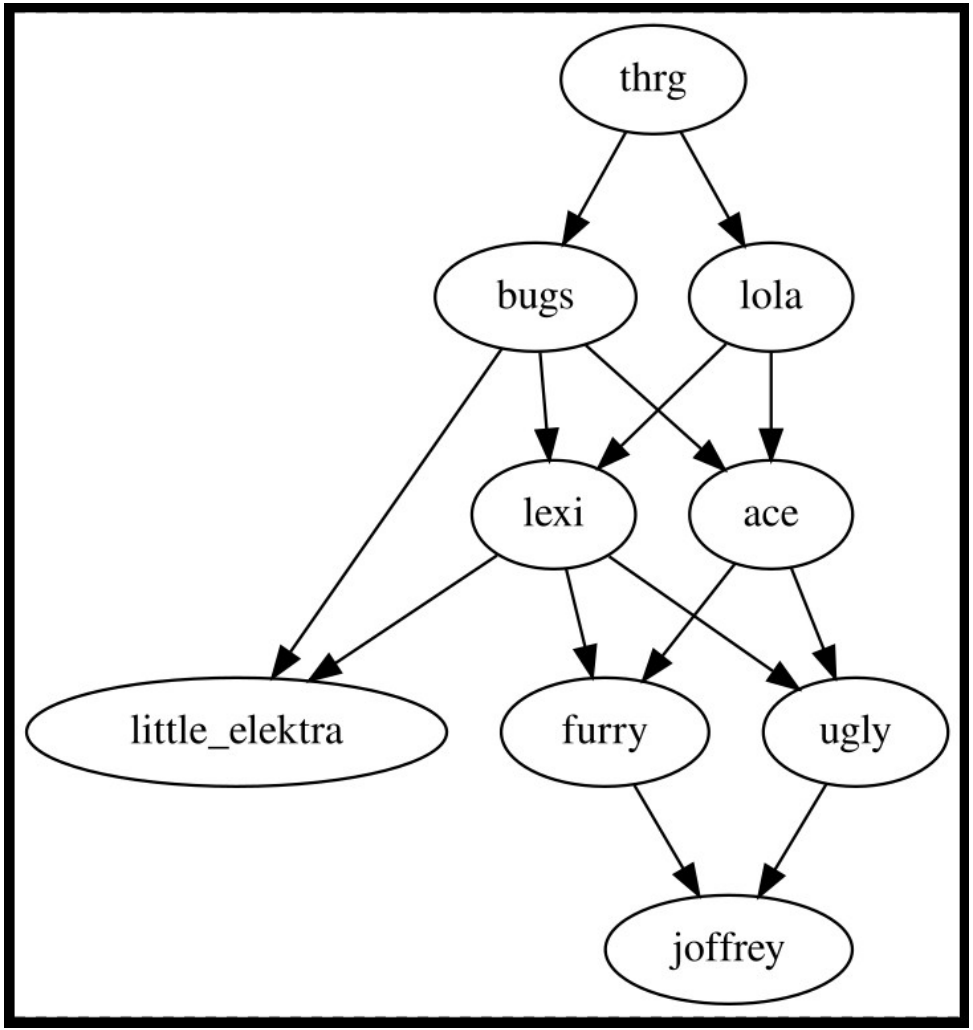
- WITH spelers_sub as (
  SELECT *
  FROM spelers
)
  SELECT *
  FROM spelers_sub;

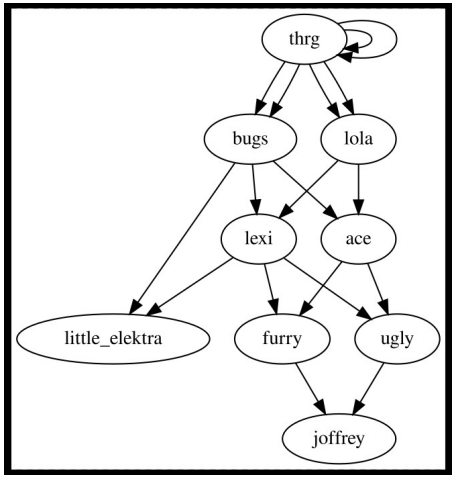
Bugs and Lola

- ```
  CREATE    TABLE familieboom(
      bijnaam      varchar(16) NOT NULL,
      vader        varchar(16),
      moeder       varchar(16),
      CONSTRAINT familieboom_pk
          PRIMARY KEY (bijnaam),
      CONSTRAINT vader_fk
          FOREIGN KEY (vader) REFERENCES familieboom(bijnaam),
      CONSTRAINT moeder_fk
          FOREIGN KEY (moeder) REFERENCES familieboom(bijnaam)
  );
  ```
  ```

Data

- INSERT INTO familieboom VALUES
  ('thrg','thrg','thrg'),
  ('lola','thrg','thrg'),
  ('bugs','thrg','thrg'),
  ('lexi','bugs','lola'),
  ('ace','bugs','lola'),
  ('furry','lexi','ace'),
  ('ugly','lexi','ace'),
  ('little_elektra','bugs','lexi'),
  ('joffrey','furry','ugly');


thrg
bugs
lola
lexi
ace
little_elektra
furry
ugly
joffrey


thrg
bugs
lola
lexi
ace
little_elektra
furry
ugly
joffrey

De nakomelingen van thrg

- SELECT *
  FROM familieboom
  WHERE 'thrg' in (vader,moeder);

```
 bijnaam | vader | moeder
---------+-------+--------
 thrg    | thrg  | thrg
 lola    | thrg  | thrg
 bugs    | thrg  | thrg
(3 rows)
```

De nakomelingen van bugs

- SELECT *
  FROM familieboom
  WHERE vader='bugs';

```
    bijnaam     | vader | moeder
----------------+-------+--------
 lexi           | bugs  | lola
 ace            | bugs  | lola
 little_elektra | bugs  | lexi
(3 rows)
```

De directe nakomelingen van bugs

- SELECT *
  FROM familieboom
  WHERE vader='bugs';

```
    bijnaam     | vader | moeder
----------------+-------+--------
 lexi           | bugs  | lola
 ace            | bugs  | lola
 little_elektra | bugs  | lexi
(3 rows)
```

- -- Dit zijn de lamprelen van bugs
  -- We willen ook de kleinlamprelen van bugs zien
  -- Hoe doe je dit?

De (klein)lamprelen van bugs

- ```
  SELECT   *
  FROM     familieboom
  WHERE    vader = 'bugs'
  OR       vader IN
      (SELECT   bijnaam
      FROM      familieboom
      WHERE     vader = 'bugs');
  ```

```
    bijnaam     | vader | moeder
----------------+-------+--------
 lexi           | bugs  | lola
 ace            | bugs  | lola
 furry          | lexi  | ace
 ugly           | lexi  | ace
 little_elektra | bugs  | lexi
(5 rows)
```
```

De (..)lamprenen van bugs

- ```
  SELECT   *
  FROM     familieboom
  WHERE    vader = 'bugs'
  OR vader IN
      (SELECT   bijnaam
      FROM      familieboom
      WHERE     vader = 'bugs');
  ```

```
    bijnaam     | vader | moeder
----------------+-------+--------
 lexi           | bugs  | lola
 ace            | bugs  | lola
 furry          | lexi  | ace
 ugly           | lexi  | ace
 little_elektra | bugs  | lexi
(5 rows)
```

- -- Worden nu ook de achterkleinlamprenen teruggegeven?

  -- Waar moeten we stoppen?
```

Recursie

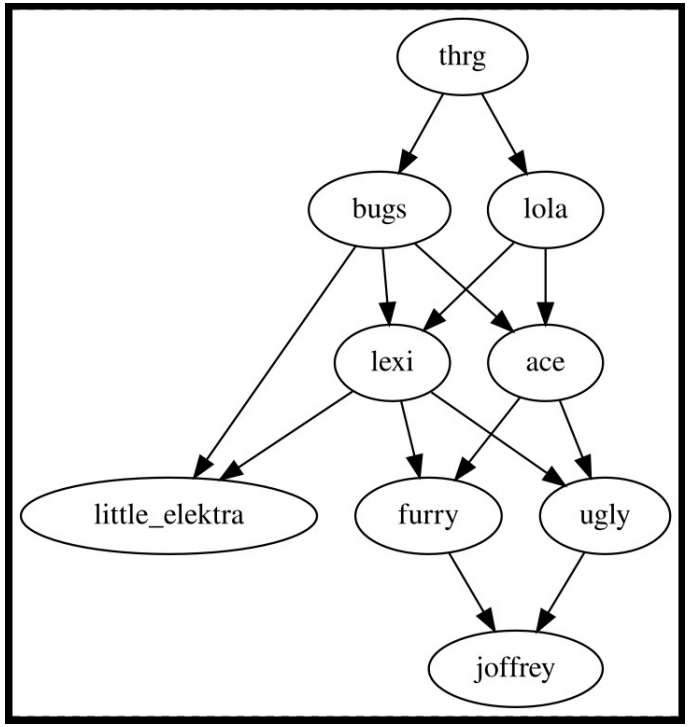
- Cf functionaliteit iteratie, maar men gebruikt het woord recursie
- Alle getallen van 1 tot 100 optellen:

```
cte=# WITH RECURSIVE t(n) AS (
    VALUES (1)
  UNION ALL
    SELECT n+1 FROM t WHERE n < 100
)
SELECT sum(n) FROM t;
 sum
------
 5050
(1 row)
```

Hoe kunnen we dit nu gebruiken?

- WITH RECURSIVE nakomeling(bijnaam, vader, moeder) AS (

```
    SELECT    bijnaam, vader, lola
    FROM      familieboom
    WHERE     vader = 'bugs'
    UNION ALL
    SELECT    f.bijnaam, f.vader, f.moeder
    FROM      familieboom f, nakomeling k
    WHERE     f.vader = k.bijnaam
    )
SELECT  *
FROM    nakomeling;
```



Bugs zijn nakomelingen

- WITH RECURSIVE nakomeling(bijnaam, vader, moeder) AS (
  SELECT bijnaam, vader, lola
  FROM familieboom
  WHERE vader = 'bugs'
  UNION ALL
  SELECT f.bijnaam, f.vader, f.moeder
  FROM familieboom f, nakomeling k
  WHERE f.vader = k.bijnaam
)
  SELECT *
  FROM nakomeling;

```
    bijnaam     | vader | moeder
----------------+-------+--------
 lexi           | bugs  | lola
 ace            | bugs  | lola
 little_elektra | bugs  | lexi
 furry          | lexi  | ace
 ugly           | lexi  | ace
 joffrey        | furry | ugly
(6 rows)
```

Opbouw

- Basis geval:

```
SELECT   bijnaam, vader, moeder
FROM     familieboom
WHERE    vader = 'bugs'
```

Opbouw

- Referentie kader nakomeling

```
WITH RECURSIVE nakomeling(bijnaam, vader,moeder) AS (
    SELECT   bijnaam, vader, moeder
    FROM     familieboom
    WHERE    vader = 'bugs'
```

Opbouw

- Recursie

```
WITH RECURSIVE nakomeling(bijnaam, vader,moeder) AS (
        SELECT   bijnaam, vader, moeder
        FROM     familieboom
        WHERE    vader = 'bugs'
    UNION ALL
        SELECT   f.bijnaam, f.vader, f.moeder
        FROM     familieboom f, nakomeling n
        WHERE    f.vader = n.bijnaam
```

Opbouw

- Volledig:

```
WITH RECURSIVE nakomeling(bijnaam, vader,moeder) AS (
        SELECT   bijnaam, vader, moeder
        FROM     familieboom
        WHERE    vader = 'bugs'
    UNION ALL
        SELECT   f.bijnaam, f.vader, f.moeder
        FROM     familieboom f, nakomeling n
        WHERE    f.vader = n.bijnaam
)
SELECT   bijnaam
FROM     nakomeling;
```

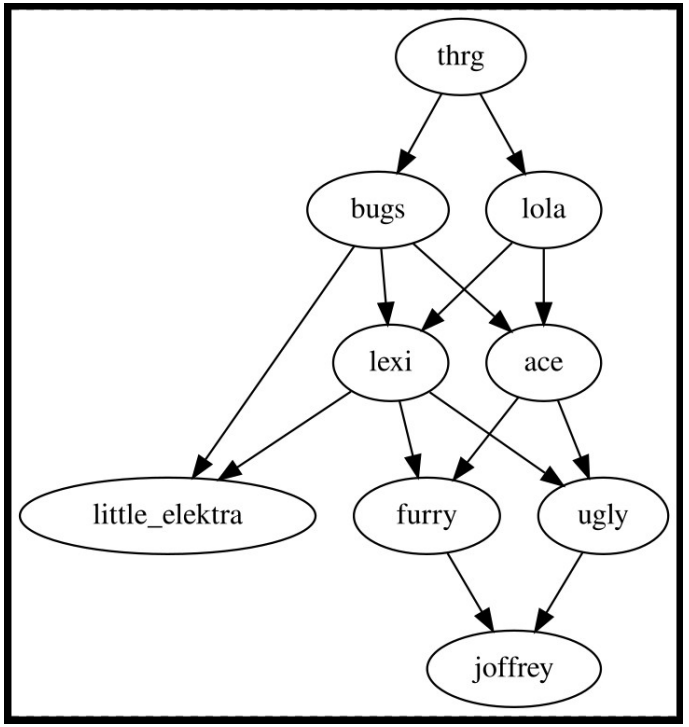
```
    bijnaam
----------------
 lexi
 ace
 little_elektra
 furry
 ugly
 joffrey
(6 rows)
```

- -- niet elke recursieve query is hetzelfde

Genealogie boom

- Van wie zijn nu al die nakomelingen?

```
    bijnaam      |      boom
-----------------+-----------------
 lexi            | bugs
 ace             | bugs
 little_elektra  | bugs
 furry           | bugs-lexi
 ugly            | bugs-lexi
 joffrey         | bugs-lexi-furry
(6 rows)
```



Genealogie boom?

- Probeer de query aan te passen (je mag andere kolommen toevoegen..):

```
WITH RECURSIVE nakomeling(bijnaam, vader,moeder) AS (
        SELECT   bijnaam, vader, moeder
        FROM     familieboom
        WHERE    vader = 'bugs'
    UNION ALL
        SELECT   f.bijnaam, f.vader, f.moeder
        FROM     familieboom f, nakomeling n
        WHERE    f.vader = n.bijnaam
)
SELECT   bijnaam
FROM     nakomeling;
```

```
    bijnaam     |      boom
----------------+-----------------
 lexi           | bugs
 ace            | bugs
 little_elektra | bugs
 furry          | bugs-lexi
 ugly           | bugs-lexi
 joffrey        | bugs-lexi-furry
(6 rows)
```

Tijd voor een micropauze

Genealogie boom?

- Probeer de query aan te passen (je mag andere kolommen toevoegen..):

```
WITH RECURSIVE nakomeling(bijnaam, vader,moeder, boom) AS (
        SELECT   bijnaam, vader, moeder, vader::text
        FROM     familieboom
        WHERE    vader = 'bugs'
    UNION ALL
        SELECT   f.bijnaam, f.vader, f.moeder, boom || '-' || f.vader
        FROM     familieboom f, nakomeling n
        WHERE    f.vader = n.bijnaam
)
SELECT   bijnaam, boom
FROM     nakomeling;
```

```
    bijnaam     |      boom
----------------+-----------------
 lexi           | bugs
 ace            | bugs
 little_elektra | bugs
 furry          | bugs-lexi
 ugly           | bugs-lexi
 joffrey        | bugs-lexi-furry
(6 rows)
```

Oneindige lussen

- Er is **geen** controle door de CTE
- Zelf controles inbouwen

Oneindige lussen

2 strategiën

- teller
- lus dedectie

- combineerbaar

Teller voor de maximum diepte (1)

- WITH RECURSIVE nakomeling(bijnaam, vader, moeder, **diepte**) AS (
  SELECT bijnaam, vader, moeder, **1**
  FROM familieboom
  WHERE vader = 'bugs'
  UNION ALL
  SELECT f.bijnaam, f.vader, f.moeder, **diepte + 1**
  FROM familieboom f, nakomeling k
  WHERE f.vader = k.bijnaam
  AND **k.diepte<7**
)
  SELECT *
  FROM nakomeling;

```
    bijnaam     | vader | moeder | diepte
----------------+-------+--------+--------
 lexi           | bugs  | lola   |      1
 ace            | bugs  | lola   |      1
 little_elektra | bugs  | lexi   |      1
 furry          | lexi  | ace    |      2
 ugly           | lexi  | ace    |      2
 joffrey        | furry | ugly   |      3
(6 rows)
```

lussen dedecteren (2)

- Tabel met lus maken:

- DROP TABLE IF EXISTS familieboom;

```
CREATE  TABLE familieboom(
bijnaam   varchar(16) NOT NULL,
vader     varchar(16),
moeder    varchar(16)
);

INSERT INTO familieboom VALUES ('lexi','bugs','lola');
INSERT INTO familieboom VALUES ('bugs','lexi','lola');
```

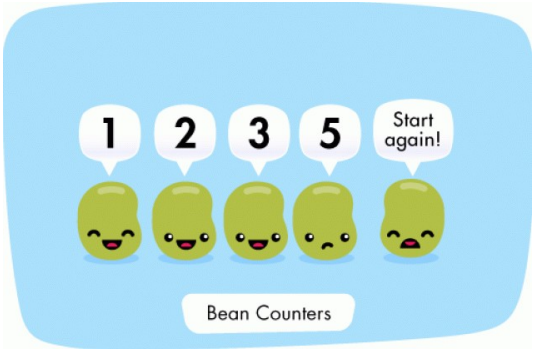
Parameter met pad bijhouden

```
WITH RECURSIVE nakomeling(bijnaam, vader, moeder, pad, lus) AS (
        SELECT  bijnaam, vader, moeder, ARRAY[vader] as pad, false
        FROM    familieboom
        WHERE   vader = 'bugs'
    UNION ALL
        SELECT  f.bijnaam, f.vader, f.moeder, CAST(k.pad || ARRAY[f.vader] as varchar(16)[]) as pad, f.vader = ANY(pad)
        FROM    familieboom f, nakomeling k
        WHERE   f.vader = k.bijnaam
        AND     NOT lus
)
SELECT  *
FROM    nakomeling;
```

```
 bijnaam | vader | moeder |       pad         | lus
---------+-------+--------+-------------------+-----
 lexi    | bugs  | lola   | {bugs}            | f
 bugs    | lexi  | lola   | {bugs,lexi}       | f
 lexi    | bugs  | lola   | {bugs,lexi,bugs}  | t
(3 rows)
```

Teller of dedectie?

Favoriet?



Uitbreiding

- CTEs werken ook met INSERT, UPDATE, DELETE
- Er is geen ISO gedefinieerd, een uitbreiding van PostgreSQL

Temp tabel

- CREATE TEMPORARY TABLE demo (x NUMERIC);

  INSERT INTO demo
  VALUES (random()), (random()), (random())
  RETURNING x;

```
         x
--------------------
 0.831172323863795
 0.867941875721334
 0.925603709126386
(3 rows)
```

Combinatie

- WITH source AS (
    INSERT INTO demo
    VALUES (random()), (random()), (random())
    RETURNING x
)
  SELECT AVG(x) FROM source;

```
          avg
-----------------------
 0.6171700716567213333
(1 row)
```

Genetica?!

- Ah, oei, patat: probeer bugs en al zijn kinderen uit de boom te wissen

```
cte=# \h delete
Command:     DELETE
Description: delete rows of a table
Syntax:
[ WITH [ RECURSIVE ] with_query [, ...] ]
DELETE FROM [ ONLY ] table_name [ * ] [ [ AS ] alias ]
    [ USING from_item [, ...] ]
    [ WHERE condition | WHERE CURRENT OF cursor_name ]
    [ RETURNING * | output_expression [ [ AS ] output_name ] [, ...] ]
```

Genetica?!

- Probeer de query aan te passen (je mag andere kolommen toevoegen..):

```
WITH RECURSIVE nakomeling(bijnaam, vader,moeder) AS (
        SELECT   bijnaam, vader, moeder
        FROM     familieboom
        WHERE    vader = 'bugs'
    UNION ALL
        SELECT   f.bijnaam, f.vader, f.moeder
        FROM     familieboom f, nakomeling n
        WHERE    f.vader = n.bijnaam
)
SELECT   bijnaam, boom
FROM     nakomeling;
```

Recursief verwijderen

- WITH RECURSIVE **nakomeling**(bijnaam) AS (

```
        SELECT  'bugs'::varchar
    UNION ALL
        SELECT  f.bijnaam
        FROM    nakomeling k JOIN familieboom f ON (k.bijnaam=f.vader)
)
DELETE FROM familieboom
USING    nakomeling
WHERE   nakomeling.bijnaam = familieboom.bijnaam;
```

Ontvlooien met CTEs?

- Zoek alle spelers met een evenlange naam, stapsgewijs:
- WITH lengte_namen AS (

```
    SELECT  naam, length(naam) AS n
    FROM    spelers),
evenlange_namen AS (
    SELECT  l1.naam AS naam1, l2.naam AS naam2
    FROM    lengte_namen l1 JOIN lengte_namen l2 ON (l1.n = l2.n)
    WHERE   l1.naam <> l2.naam),
 overzichts_lijst AS (
    SELECT      naam1, array_agg(naam2) AS lijst
    FROM        evenlange_namen
    GROUP BY    naam1)
SELECT  l.naam, o.lijst
FROM    lengte_namen l LEFT JOIN overzichts_lijst o ON (l.naam = o.naam1);
```

- -- Klopt dit?

Ontvlooien met CTEs?

- Tussen resultaten wegschrijven om eventuele fouten te vinden:

- CREATE TEMPORARY TABLE debug_table

  (id serial,

  t text,

  r text);

Ontvlooien met CTEs?

```
WITH lengte_namen AS (
    SELECT  naam, length(naam) AS n FROM   spelers),
debug_lengte_namen AS (
    INSERT  INTO    debug_table(t,r)
        SELECT  'lengte_namen', ROW(l.*)::text
        FROM    lengte_namen l),
evenlange_namen AS (
    SELECT  l1.naam AS naam1, l2.naam AS naam2
    FROM    lengte_namen l1 JOIN lengte_namen l2 ON l1.n = l2.n
    WHERE   l1.naam <> l2.naam),
overzichts_lijst AS (
    SELECT  naam1, array_agg(naam2) AS lijst
    FROM    evenlange_namen
    GROUP BY naam1)
SELECT  l.naam, o.lijst
FROM    lengte_namen l LEFT JOIN overzichts_lijst o ON l.naam = o.naam1;
```

```
      naam       |                                lijst
-----------------+-------------------------------------------------------------------------
 Elfring         | {"Hofland        ","Moerman        "}
 Permentier      | {"Bakker, de     ","Niewenburg     ","Bakker, de     ","Niewenburg     "}
 Wijers          |
 Niewenburg      | {"Bakker, de     ","Permentier     ","Permentier     "}
 Cools           |
 Cools           |
 Bischoff        | {"Meuleman       "}
 Bakker, de      | {"Permentier     ","Niewenburg     ","Permentier     "}
 Bohemen, van    |
 Hofland         | {"Elfring        ","Moerman        "}
 Meuleman        | {"Bischoff       "}
 Permentier      | {"Bakker, de     ","Niewenburg     ","Bakker, de     ","Niewenburg     "}
 Moerman         | {"Elfring        ","Hofland        "}
 Baalen, van     |
(14 rows)
```

Ontvlooien met CTEs?

```
     naam       |                                      lijst
----------------+-------------------------------------------------------------------------
 Elfring        | {"Hofland       ","Moerman       "}
 Permentier     | {"Bakker, de    ","Niewenburg    ","Bakker, de    ","Niewenburg    "}
 Wijers         |
 Niewenburg     | {"Bakker, de    ","Permentier    ","Permentier    "}
 Cools          |
 Cools          |
 Bischoff       | {"Meuleman      "}
 Bakker, de     | {"Permentier    ","Niewenburg    ","Permentier    "}
 Bohemen, van   |
 Hofland        | {"Elfring       ","Moerman       "}
 Meuleman       | {"Bischoff      "}
 Permentier     | {"Bakker, de    ","Niewenburg    ","Bakker, de    ","Niewenburg    "}
 Moerman        | {"Elfring       ","Hofland       "}
 Baalen, van    |
(14 rows)
```

Debug tabel

```
oefeningen=> SELECT * FROM debug_table ORDER BY regexp_replace(r,'[\D]+','','g');
 id |      t       |           r
----+--------------+------------------------
  8 | lengte_namen | ("Bakker, de     ",10)
  2 | lengte_namen | ("Permentier     ",10)
 12 | lengte_namen | ("Permentier     ",10)
  4 | lengte_namen | ("Niewenburg     ",10)
 14 | lengte_namen | ("Baalen, van    ",11)
  9 | lengte_namen | ("Bohemen, van   ",12)
  6 | lengte_namen | ("Cools          ",5)
  5 | lengte_namen | ("Cools          ",5)
  3 | lengte_namen | ("Wijers         ",6)
  1 | lengte_namen | ("Elfring        ",7)
 10 | lengte_namen | ("Hofland        ",7)
 13 | lengte_namen | ("Moerman        ",7)
  7 | lengte_namen | ("Bischoff       ",8)
 11 | lengte_namen | ("Meuleman       ",8)
(14 rows)

oefeningen=> SELECT * FROM debug_table ORDER BY regexp_replace(r,'[\D]+','','g')::numeric;
```

Referenties

- Programming the SQL Way with Common Table Expressions, Bruce Momijan
- Debugging complex SQL queries with writable CTEs, Gianni Ciolli
- Postgis Latest News, Vincent Picavet