

Lateral Joins

wim.bertels@ucll.be

Naamsvermelding-NietCommercieel-GelijkDelen 4.0 Unported
Licentie

Inleiding

```
CREATE TEMPORARY TABLE nummers AS  
SELECT generate_series(1,3) AS max_num;
```

```
SELECT  *  
FROM    nummers;
```

-- Tijdelijke tabellen worden periodiek opgeruimd

max_num
1
2
3
(3 rows)

Problem?

```
SELECT    *  
FROM      nummers,  
          (SELECT generate_series(1,max_num)) AS max_lijs;
```

Problem?

```
SELECT  *  
FROM    nummers,  
        (SELECT generate_series(1,max_num)) AS max_lijs;
```

```
/*  
psql:04_2_LATERAL.sql:11: ERROR:  column "max_num" does not exist  
LINE 3:  (SELECT generate_series(1,max_num)) AS max_lijs;  
  
HINT:  There is a column named "max_num" in table "nummers",  
but it cannot be referenced from this part of the query.  
*/
```

LATERAL

Subqueries die verschijnen in FROM kunnen voorafgegaan worden door het sleutelwoord LATERAL.

Hierdoor kunnen ze verwijzen naar kolommen uit voorafgaande FROM-items.

(Zonder LATERAL, wordt elke subquery onafhankelijk geëvalueerd en kan deze dus niet verwijzen naar een ander FROM item).

Oftewel: gecorreleerde subqueries in de FROM-komponent

LATERAL Detail

Via LATERAL kunnen we verwijzen naar een eerdere referentie uit de FROM-komponent:

- naar een eerdere tabelreferentie
- naar een eerdere subquery
- naar een eerdere functie die een verzameling teruggeeft (SRF)
 - (In dit geval wordt een LATERAL gedrag door de standaard bepaalt, je kan dus LATERAL weglaten in dit geval)

Oplossing

```
SELECT *  
FROM   nummers,  
       (SELECT generate_series(1,max_num)) AS max_lijs;
```

```
/*  
psql:04_2_LATERAL.sql:11: ERROR: column "max_num" does not exist  
LINE 3: (SELECT generate_series(1,max_num)) AS max_lijs;  
  
HINT: There is a column named "max_num" in table "nummers",  
but it cannot be referenced from this part of the query.  
*/
```

```
SELECT *  
FROM   nummers, LATERAL  
       (SELECT generate_series(1,max_num)) AS max_lijs;
```

Uitvoer

max_num
1
2
3

(3 rows)

max_num	generate_series
1	1
2	1
2	2
3	1
3	2
3	3

(6 rows)

```
SELECT *  
FROM   nummers, LATERAL  
       (SELECT generate_series(1,max_num)) AS max_lijs;
```


PostgreSQL uitbreiding

```
SELECT *
```

```
FROM    nummers, LATERAL generate_series(1,max_num);
```

```
SELECT *
```

```
FROM    nummers, generate_series(1,max_num);
```

-- SELECT voor functie is optioneel in dit geval

Voorbeeld

```
SELECT      *  
FROM        klanten k  
            LEFT JOIN LATERAL  
            (SELECT  age(k.geboortedatum) as leeftijd) AS l  
            ON (leeftijd=age(k.geboortedatum));
```

Voorbeeld

```
SELECT      *
FROM        klanten k
LEFT JOIN LATERAL
  (SELECT    age(k.geboortedatum) as leeftijd) AS l
ON (leeftijd=age(k.geboortedatum));
```

-- toon per klant de leeftijd

klantnr	naam	vnaam	geboortedatum	leeftijd
121	Hassoui	Sjeik	1975-06-12	43 years 8 mons 27 days
122	Martens	Hedwich	1978-06-30	40 years 8 mons 9 days
123	Ellison	Larry	1975-10-10	43 years 4 mons 30 days
124	Van Rossem	Jean-Pierre	1975-01-12	44 years 1 mon 28 days
125	Frimout	Dirk	1980-11-29	38 years 3 mons 10 days
126	Gates	Bill	1982-12-25	36 years 2 mons 15 days
(6 rows)				

JOIN Conditie

- Is de join conditie nodig?
 - De subquery is gecorreleerd.
- Wat is het effect van het cartesisch produkt in dit geval?

Voorbeeld

```
SELECT      *  
FROM        klanten k  
            LEFT JOIN LATERAL  
            (SELECT  age(k.geboortedatum) as leeftijd) AS l  
            ON (leeftijd=age(k.geboortedatum));
```

```
SELECT      *  
FROM        klanten k  
            LEFT JOIN LATERAL  
            (SELECT  age(k.geboortedatum) as leeftijd) AS l  
            ON true;
```

JOIN Conditie

```
When a FROM item contains LATERAL cross-references,  
evaluation proceeds as follows:  
for each row of the FROM item providing the cross-referenced column(s),  
or set of rows of multiple FROM items providing the columns,  
the LATERAL item is evaluated using that row or row set's values of the columns.  
The resulting row(s) are joined as usual with the rows they were computed from.  
This is repeated for each row or set of rows from the column source table(s).
```

> Cartesisch produkt gedraagt zich hier als een foreach lus

JOIN Conditie

Welk soort JOIN?

- CROSS, INNER, LEFT
- Niet: RIGHT of FULL!
- De subquery is gecorreleerd.

Gedrag LEFT blijft tov INNER

Een geval

Geef voor elke reis
de twee kleinste objecten die bezocht worden

- Ter vergelijking:

```
SELECT *
```

```
FROM bezoeken NATURAL INNER JOIN hemelobjecten
```

```
WHERE reisnr = 32
```

```
ORDER BY diameter;
```

objectnaam	reisnr	volgnr	verblijfsduur	satellietvan	afstand	diameter
Deimos	32	3	0	Mars	23.400	7
Phobos	32	2	1	Mars	9.270	14
Maan	32	5	2	Aarde	384.400	3476
Maan	32	1	0	Aarde	384.400	3476
Mars	32	4	3	Zon	227900.000	6794
(5 rows)						

Een geval

Geef voor elke reis
de twee kleinste objecten die bezocht worden

```
SELECT *  
FROM reizen r LEFT JOIN LATERAL  
  (SELECT *  
   FROM bezoeken b NATURAL INNER JOIN hemelobjecten h  
   WHERE b.reisnr=r.reisnr  
   ORDER BY h.diameter  
   FETCH FIRST 2 ROWS ONLY) AS l  
ON (true);
```

Wim Bertels (CC)BY-SA-NC

Referenties:

- <https://www.postgresql.org/docs/current/sql-select.html>
- <https://www.postgresql.org/docs/current/static/queries-table-expressions.html>
- <https://blog.2ndquadrant.com/join-lateral/>
- <https://modern-sql.com/slides>
- Becoming A SQL Guru, Stella Nisenbaum