Lateral Joins

wim.bertels@ucll.be

Naamsvermelding-NietCommercieel-GelijkDelen 4.0 Unported Licentie

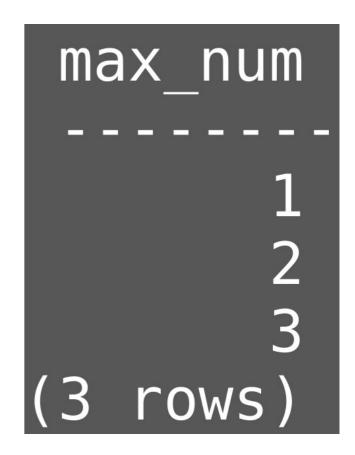
Inleiding

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

SELECT *

FROM nummers;

-- Tijdelijke tabellen worden periodiek opgeruimd



Probleem?

Probleem?

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

```
/*
psql:04_2_LATERAL.sql:11: ERROR: column "max_num" does not exist
LINE 3: (SELECT generate_series(1,max_num)) AS max_lijst;
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_lijst;
               psql:04 2 LATERAL.sql:11: ERROR: column "max num" does not exist
               LINE 3: (SELECT generate series(1, max num)) AS max lijst;
               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_lijst;

Uitvoer

```
max_num
1
2
3
(3 rows)
```

```
generate series
max num
  rows)
```

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

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

FROM klanten k
LEFT JOIN LATERAL
(SELECT age(k.geboortedatum) as leeftijd) AS I
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 122 Martens 123 Ellison 124 Van Rosse 125 Frimout 126 Gates (6 rows)	Sjeik Hedwich Larry m Jean-Pierre Dirk Bill	1975-06-12 1978-06-30 1975-10-10 1975-01-12 1980-11-29 1982-12-25	43 years 8 mons 27 days 40 years 8 mons 9 days 43 years 4 mons 30 days 44 years 1 mon 28 days 38 years 3 mons 10 days 36 years 2 mons 15 days

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
                   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 Phobos Maan Maan Mars (5 rows)	32 32 32 32 32	3 2 5 1 4	0 1 2 0 3	Mars Mars Aarde Aarde Zon	23.400 9.270 384.400 384.400 227900.000	7 14 3476 3476 6794

Een geval

Geef voor elke reis

de twee kleinste objecten die bezocht worden SELECT * FROM reizen r LEFT JOIN LATERAL (SELECT bezoeken b NATURAL INNER JOIN hemelobjecten h FROM b.reisnr=r.reisnr WHERE ORDER BY h.diameter FETCH FIRST 2 ROWS ONLY) AS I 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