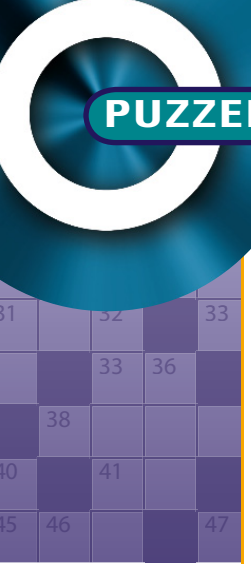

Puzzelen met SQL

Een schone kooi voor Snuffie

Bijna zes jaar oud is onze cavia Snuffie. Best oud voor een cavia, hoewel er op sommige websites staat dat een echt oude cavia tot wel vijfiten jaar kan worden. Als de tante op bezoek komt van wie we de cavia gekregen hebben, als cadeautje voor mijn zoon, dan verbaasd ze zich erover dat Snuffie nog in leven is. "Leeft hij nou nog steeds?" De cavia die ze haar eigen dochter gelijktijdig cadeau heeft gedaan is al een jaar of vier dood.

Als mijn vrouw 's avonds het eten gaat maken, begint Snuffie hard te fluiten. Hij (of misschien is Snuffie wel een zij) weet dan dat het snijafval van de groenten voor hem is. Dagelijks doet hij zich dan ook te goed aan ruime hoeveelheden bloemkool, broccoli en sla. De klokhuizen van de appels die de kinderen eten belanden ook steevast bij Snuffie in de kooi.
Aan eten ontbreekt het niet, het schoonmaken van de kooi is een ander verhaal. Als het even kan 'vergeet' iedereen dat het hoog tijd word dat de kooi weer eens verfrist wordt. Er is altijd wel een excuus te bedenken waarom het 'nu even niet kan'. Zo zit ik nu dit artikel te typen, terwijl vrouw en kinderen met de kooi van Snuffie aan de gang gaan. Tja, deadlines komen soms goed uit...
In deze aflevering van Puzzelen met SQL gaan we een schema maken wie er verantwoordelijk is om de kooi schoon te maken. Met het maken van dit schema kan SQL ons wel helpen, met de daadwerkelijke uitvoering helaas niet.
Laten we beginnen met het genereren van een lijst met weeknummers. Er zijn vele manieren om een lijst rijen te genereren. Een veel geziene manier is de volgende.

```
SQL> with rws as
  2  (select rownum rn
  3     from dual
  4    connect by level <= 53
  5  )
  6  select trunc (add_months (sysdate, 12), 'yy')
  7         + ((rn - 1) * 7) weekdag
  8    from rws
  9  ;

WEEKDAG
---------
01-JAN-11
08-JAN-11
15-JAN-11
22-JAN-11
...
10-DEC-11
17-DEC-11
24-DEC-11
31-DEC-11
```

Het truukje zit in de bovenstaande query in regel 4. Volgens sommigen is het eigenlijk een bug. Met een hierarchische query, CONNECT BY is tenslotte onderdeel van de hierarchische syntax die al sinds de eerste versie van Oracle (Oracle 2) bestaat, zou het volgens de documentatie verplicht zijn om het keyword PRIOR te gebruiken. Aangezien deze niet in de query staat zou het een syntactisch incorrect zijn. Maar velen, inclusief Tom Kyte van AskTom, gebruiken deze truuk.
Sinds Oracle 11g Release 2 is er een nieuwe manier om hierarchische queries te schrijven. Met behulp van Recursive Subquery Factoring. Subquery Factoring bestond al sinds Oracle 10g en dit wordt ook in bovenstaande query gebruikt. De WITH clause die op de eerste regel begint is namelijk Subquery Factoring. Het is altijd goed om de officiele Oracle termen te kennen, dat maakt het doorzoeken van de Oracle documentatie een stuk eenvoudiger. Als je de syntax van Subquery Factoring probeert te zoeken door "WITH" te gebruiken vind je heel veel, maar niet waar je naar op zoek

De officiele Oracle termen kennen maakt het doorzoeken van de Oracle documentatie een stuk eenvoudiger

bent. In SQL Server kennen ze Recursive Subquery Factoring ook maar noemen ze het "Common Table Expression". Als je met deze term in de Oracle documentatie gaat zoeken, dan krijg je helemaal geen resultaat.

Laten we bovenstaande query eens herschrijven door gebruik te maken van Recursive Subquery Factoring.

```
SQL> with rws (rn) as
  2  (select 1
  3     from dual
  4   union all
  5   select rn + 1
  6     from rws
  7    where rn <= 52
  8  )
  9  select trunc (add_months (sysdate, 12), 'yy')
 10         + ((rn - 1) * 7) weekdag
 11    from rws
 12  ;

WEEKDAG
---------
01-JAN-11
08-JAN-11
15-JAN-11
22-JAN-11
...
10-DEC-11
17-DEC-11
24-DEC-11
31-DEC-11
```

De syntax van Recursive Subquery Factoring is, net als iedere nieuwe syntax, even wennen. Het recursieve karakter zit hem in het gebruik van de eigen "tabel" in het tweede deel van de subquery beginende op regel 6.
Het begint met een enkele rij (regels 2 en 3) als begin van de hierarchie. Vervolgens wordt er door de eigen tabel te querien steeds meer rijen aan toegevoegd.
De UNION ALL van regel 4 is een verplichting, indien deze weggelaten word dan krijg je een foutmelding.

```
ERROR at line 6:
ORA-32040: recursive WITH clause must use a UNION ALL operation
```

Op de eerste regel geef je tussen haakjes de alias aan voor de kolommen die uit de query komen. Deze alias word dan ook gebruikt in het tweede deel van de Subquery (regels 5 tot en met 7). In de eerdere query werd de limiet van het aantal rijen op 53 gezet. Bij deze query staat er echter 52. Het eerste deel van de Subquery levert één rij op en het tweede deel zorgt voor de overige 52 rijen.

Misschien dat je je nu afvraagt waarom eerst rijen genereren om vervolgens een lijst met data te genereren. Het lijkt erop dat er een bug in Recursive Subquery Factoring zit.

```
SQL> with rws (rn, dt) as
  2  (select 1, sysdate
  3     from dual
  4   union all
  5   select rn + 1, dt + 1
  6     from rws
  7    where rn <= 2
  8  )
  9  select *
 10    from rws
 11  ;
select rn + 1, dt + 1
                  *
ERROR at line 5:
ORA-01790: expression must have same datatype as corresponding
expression
```

In dit voorbeeld gebruiken we SYSDATE als tweede kolom om hier vervolgens op regel 5 iedere keer 1 dag bij op te tellen. Zoals je ziet wordt er een exception geraised terwijl er ogenschijnlijk niets mis is met de datatypes.
Indien we een expliciete conversie doen naar een DATE, dan werkt het echter wel. Zeer vreemd.

```
SQL> alter session set nls_date_format = 'dd-mm-yyyy hh24:mi:ss'
  2  /

Session altered.

SQL> with rws (rn, dt) as
  2  (select 1, cast (sysdate as date)
  3     from dual
  4   union all
  5   select rn + 1, dt + 1
  6     from rws
  7    where rn <= 2
  8  )
  9  select *
 10    from rws
 11  ;

        RN DT
---------- -------------------
         1 01-11-2010 12:31:51
         2 31-10-2010 12:31:51
         3 30-10-2010 12:31:51
```

Een expliciete conversie met behulp van TO_DATE, werkt echter weer niet. Misschien heeft het hier iets mee te maken:

```
SQL> select dump (sysdate) sdate
  2       , dump (cast (sysdate as date)) castDate
  3    from dual
  4  /

SDATE                                CASTDATE
---------------------------------- ----------------------------------
Typ=13 Len=8: 218,7,11,1,12,34,15,0 Typ=12 Len=7: 120,110,11,1,13,35,16
```

Met de DUMP functie kun je de interne representatie van een expressie bekijken. Bij "Typ" staat wat voor datatype het betreft. Het number geeft dan aan welk datatype het betreft. Deze code kun je dan opzoeken in de Oracle documentatie (zie link onderaan het artikel). Het is ook mogelijk om deze gegevens boven tafel te krijgen door de sourcecode van DBMS_DESCRIBE te querien.

```
SQL> select text
  2    from all_source
  3   where name = 'DBMS_DESCRIBE'
  4     and type = 'PACKAGE'
  5     and line between 150 and 159
  6   order by line
  7  /

TEXT
--------------------------------------------------------------------------
  --       Oracle datatype of the parameter. These are:
  --            0 - This row is a placeholder for a procedure with
  --                no arguments.
  --            1 - VARCHAR2
  --            2 - NUMBER
  --            3 - NATIVE INTEGER (BINARY_INTEGER or PLS_INTEGER)
  --            8 - LONG
  --           11 - ROWID
  --           12 - DATE
  --           23 - RAW
```

Hoewel we uit de DUMP functie de Types 12 en 13 terugkrijgen, is niet terug te vinden wat voor data type 13 nu eigenlijk is. Je zou verwachten, tenminste dat verwachtte ik, dat ook SYSDATE van type 12 zou zijn.
Nog wat meer testen met Recursive Subquery Factoring in combinatie met DATE levert nog meer onverwachte resultaten op, zoals de volgende:

```
SQL> with x (d, i) as
  2  (select cast (sysdate as date)
  3        , 1
  4     from dual
  5    union all
  6   select d + 1, i + 1
  7     from x
  8    where i < 5
  9   )
 10  select *
 11    from x
 12  /

D                           I
------------------- ----------
01-11-2010 13:53:58          1
31-10-2010 13:53:58          2
30-10-2010 13:53:58          3
29-10-2010 13:53:58          4
28-10-2010 13:53:58          5
```

Op regel 6 maken we gebruik van de alias "d" - wat een "gecaste" DATE is. Bij deze DATE (sysdate) tellen we per recursie één erbij. Het resultaat laat echter zien dat er vanaf SYSDATE iedere rij een dag minder wordt. Het wijzigen van de expressie "d + 1" naar "d - 1" lijkt geen effect te hebben. (met dank aan Anton Scheffer voor het opmerken van deze bijzonderheid)

Maar we dwalen een beetje af, zo wordt die kooi natuurlijk nooit schoon.
Het mooie van Subquery Factoring is dat je op basis van de resultaten van een query verder kunt werken naar het uiteindelijke resultaat.
Op basis van de lijst met data bepalen we in welk weeknummer dit valt door gebruik te maken van het "ww" formaat masker.

```
SQL> alter session set nls_date_format = 'dd-mm-yyyy'
  2  /

Session altered.

SQL> with rws (rn) as
  2  (select 1
  3     from dual
  4   union all
  5   select rn + 1
  6     from rws
  7    where rn <= 52
  8  )
  9  , week as
 10  (
 11  select trunc (add_months (sysdate, 12), 'yy')
 12         + ((rn - 1) * 7) weekdag
 13    from rws
 14  )
 15  select weekdag
 16       , to_char (weekdag, 'ww') weekno
 17    from week;

WEEKDAG    WE
---------- --
01-01-2011 01
08-01-2011 02
15-01-2011 03
22-01-2011 04
...
10-12-2011 50
17-12-2011 51
24-12-2011 52
31-12-2011 53
```

Omdat we thuis met z'n vieren zijn, verdelen we de taken met behulp van de MOD functie.

```
mod (to_number (weekno), 4)
```

Met behulp van een CASE expressie bepalen we de uiteindelijke verantwoordelijke die de kooi schoon moet gaan maken.

```
case mod (to_number (weekno), 4)
when 0 then 'Tim'
when 1 then 'Lara'
when 2 then 'Rian'
when 3 then 'Alex'
end wie
```

In dit geval betreft het de zogenoemde SIMPLE Case Expression. De vergelijking tussen de expressie van de CASE en de waarde uit de WHEN clausule is op basis van gelijkheid. Bij een Searched Case komt de vergelijkings expressie volledig in de WHEN clausule te staan.

Nu het schema compleet is, kunnen we ook eens gaan kijken wie er gedurende het gehele jaar het vaakst de kooi schoon moet maken.

```
SQL> with rws (rn) as
  2  (select 1
  3     from dual
  4   union all
  5   select rn + 1
  6     from rws
  7    where rn <= 52
  8  )
  9  , week as
 10  (
 11  select trunc (add_months (sysdate, 12), 'yy')
 12         + ((rn - 1) * 7) weekdag
 13    from rws
 14  )
 15  , wanneer as
 16  (
 17  select weekdag
 18       , to_char (weekdag, 'ww') weekno
 19    from week
 20  )
 21  , balen as
 22  (
 23  select weekdag
 24       , weekno
 25       , case mod (to_number (weekno), 4)
 26         when 0 then 'Tim'
 27         when 1 then 'Lara'
 28         when 2 then 'Rian'
 29         when 3 then 'Alex'
 30         end wie
 31    from wanneer
 32  )
 33  select case grouping (wie)
 34         when 0 then wie
 35         when 1 then 'Totaal:'
 36         end wie
 37       , count(*)
 38    from balen
 39   group by rollup (wie);

WIE       COUNT(*)
------- ----------
Alex            13
Lara            14
Rian            13
Tim             13
Totaal:         53
```

Over het gehele jaar gezien zal Lara de kooi één keer vaker de kooi schoon maken dan de overige familie leden. Ik zal je dan wel helpen, hoor Lara.... tenminste als ik dan niet toevallig nog een deadline te halen heb.

Na veel rekenwerk is de kooi eindelijk schoon...

Links

Oracle Datatypes: http://download.oracle.com/docs/cd/E11882_01/server.112/e17118/sql_elements001.htm#BABCGCHG

Patrick Barel is consultant bij AMIS Services. Hij is te bereiken via email patrick.barel@amis.nl.

Alex Nuijten is Oracle consultant bij AMIS Services. Hij is te bereiken via email alex.nuijten@amis.nl