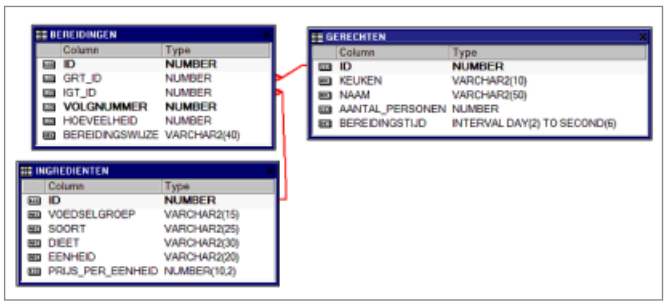
Puzzelen met SQL

Het Kookboek

Mijn moeder is Indisch. Zij heeft een kookboek dat al vele generaties gebruikt en uitgebreid wordt. Intussen kent mijn moeder dit kookboek vrijwel geheel uit haar hoofd, en heeft dus geen enkele moeite om de gerechten te bereiden. Een rijsttafel staat dan ook zo op tafel, schijnbaar zonder enige inspanning.

Zelden ben ik zelf in de keuken te vinden en wanneer dit wel het geval is dan kost het bereiden van een maaltijd veel meer moeite dan gemiddeld. Vooral het bepalen van de hoeveelheden voor een gerecht - voor meer of minder personen dan aangegeven bij een gerecht - kost mij nogal wat moeite. Een mooi plaatje erbij en goede stap-voor-stap instructies zijn voor mij vereisten om een gerecht te kunnen bereiden. Improviseren met ingrediënten die we toevallig in huis hebben? Nee, dat is aan mij niet besteed. "Dan moet je het vaker doen," zegt mijn vrouw dan. In deze puzzel gaan we koken met SQL.

Om te beginnen de tabellen met welke we gaan werken.



Afbeelding 1. Tabellen

Wat zijn de tabellen die we gebruiken in het kookboek? Allereerst hebben we een lijst nodig van ingrediënten. Voor verschillende gerechten zijn dezelfde ingrediënten te gebruiken. Uiteraard hebben we ook de Gerechten nodig, waar het immers allemaal om draait. En als laatste de stap-voor-stap Bereidingen. Om het model eenvoudig te houden, zijn deze tabellen niet genormaliseerd.

De tabel met Gerechten bevat een kolom met het datatype INTERVAL DAY(2) TO SECOND (6). Het INTERVAL-datatype is geïntroduceerd in Oracle 9i en behoort tot de DATE-familie. De INTERVAL wordt gebruikt om een tijdsduur aan te duiden. Van dit datatype zijn twee varianten, de INTERVAL DAY TO SECOND en de INTERVAL YEAR TO MONTH. Om te laten zien hoe dit datatype eruit zit, zullen we de Gerechten-tabel eens queriën.

```
SQL> select naam
  2       , bereidingstijd
  3    from gerechten
  4  /

NAAM                               BEREIDINGSTIJD
---------------------------------- --------------------
Kippenpootjes met Zaanse mosterd   +00 00:25:00.000000
```

Query 1.

Het eerste deel van de Bereidingstijd-kolom - de +00 - geeft aan hoeveel dagen er nodig zijn om dit gerecht te bereiden. Het volgende deel, 00:25:00.000000, geeft aan de hoeveelheid in uren, minuten, seconden en fractie van seconden. Dit gerecht, Kippenpootjes met Zaanse Mosterd – niet uit mijn moeders kookboek, maar uit de Margriet, heeft een bereidingstijd van 0 dagen, 0 uur, 25 minuten en 0 seconden.

Een voordeel van het gebruik van een INTERVAL om de bereidingstijd op te slaan, is dat deze zeer eenvoudig te gebruiken is zonder ingewikkelde conversies te hoeven doen. Stel dat je wilt weten wanneer je zou kunnen eten als je nu zou beginnen met koken.

```
SQL> select to_char (sysdate
  2                 ,'dd-mm-yyyy hh24:mi:ss'
  3                 ) start_met_koken
  4       , to_char (sysdate + bereidingstijd
  5                 ,'dd-mm-yyyy hh24:mi:ss'
```

```
  6                ) wanneer_is_het_klaar
  7    from gerechten
  8  /


START_MET_KOKEN     WANNEER_IS_HET_KLAA
------------------- -------------------
23-05-2008 16:15:49 23-05-2008 16:40:49
23-05-2008 16:15:49 23-05-2008 16:30:49
```

Query 2.

Met INTERVAL is het wel mogelijk om deze bij elkaar op te tellen, maar is het niet mogelijk om aggregaten te gebruiken. Het sommeren van een INTERVAL werkt dus niet.

De Puzzels

1. Stel een boodschappenlijstje op voor gerecht x voor y personen.

Om een boodschappenlijstje te kunnen maken hebben we gegevens uit alle drie de tabellen nodig. Laten we eerst een selectie maken op basis van de drie tabellen voor het eerste gerecht.

```
SQL> select g.aantal_personen
  2       , b.hoeveelheid
  3       , i.eenheid
  4       , i.soort
  5    from gerechten g
  6    join bereidingen b
  7      on (g.id = b.grt_id)
  8    join ingredienten i
  9      on (i.id = b.igt_id)
 10   where g.id = 1
 11  /

AANTAL_PERSONEN HOEVEELHEID EENHEID              SOORT
--------------- ----------- -------------------
-------------------------
              4         0,5 kilogram             kippenpoot
              4           1 snufje               zout
              4           1 snufje               peper
              4           2 eetlepel             olijfolie
              4          50 gram                 boter
              4           1 scheutje             water
              4           4 eetlepel             mosterd
              4         0,5 beker                creme fraiche

8 rows selected
```

Query 3.

Aangezien de hoeveelheid, opgegeven in de BEREIDINGEN-tabel, de hoeveelheid toont voor het aantal personen waar het gerecht voor is, moeten we een deling maken om tot de hoeveelheid per persoon te komen. Dit vermenigvuldigen met het aantal mee-eters en je weet hoeveel boodschappen je moet doen. De totale boodschappenlijst voor "Kippenpootjes in Zaanse Mosterd" wordt dan:

```
SQL> with mee_eters
  2  as
  3  (select 8 genodigden
  4        , 1 gerecht
  5     from dual
  6  )
  7  select (b.hoeveelheid/ g.aantal_personen)
  8           * m.genodigden aan_te_schaffen_hoeveelheid
  9       , i.eenheid
 10       , i.soort
 11    from gerechten g
 12    join bereidingen b
 13      on (g.id = b.grt_id)
 14    join ingredienten i
 15      on (i.id = b.igt_id)
 16    cross
 17    join mee_eters m
 18   where g.id = m.gerecht
 19  /

AAN_TE_SCHAFFEN_HOEVEELHEID EENHEID              SOORT
--------------------------- -------------------
-------------------------
                          1 kilogram             kippenpoot
                          2 snufje               zout
                          2 snufje               peper
                          4 eetlepel             olijfolie
                        100 gram                 boter
                          8 eetlepel             mosterd
                          1 beker                creme fraiche
                          2 scheutje             water

8 rows selected
```

Query 4.

Het gebruik van de WITH-clause aan het begin van de query, volgens de Oracle-documentatie Subquery Factoring genoemd, maakt het eenvoudig om het aantal personen en het gerecht te wijzigen. Hoewel het hier niet echt tot uitdrukking komt, biedt Subquery Factoring de mogelijkheid om stukje voor stukje jouw query op te bouwen.

1a. Hoeveel geld krijgt de boodschapper mee voor het boodschappenlijstje uit vraag 1?

Aan de hand van de query die we gebruikt hebben bij de eerste vraag, kunnen we een uitbreiding maken die het totaalbedrag voor ons berekend. Echt spannend is dit niet.

```
SQL> with mee_eters
  2  as
  3  (select 8 genodigden
  4        , 1 gerecht
  5     from dual
  6  )
```

```
 7  select sum (prijs_per_eenheid * aan_te_schaffen_hoeveelheid) bedrag
 8    from (select i.soort artikel
 9               , (b.hoeveelheid/ g.aantal_personen)
10                    * m.genodigden aan_te_schaffen_hoeveelheid
11               , i.eenheid
12               , i.prijs_per_eenheid
13            from gerechten g
14            join bereidingen b
15              on (g.id = b.grt_id)
16            join ingredienten i
17              on (i.id = b.igt_id)
18            cross
19            join
20            mee_eters m
21           where g.id = m.gerecht
22         )
23  /

    BEDRAG
----------
      9,45
```

Query 5.

Laten we een overzicht maken van het artikel dat we moeten kopen, met de prijs per artikel en de totaalprijs erbij. Misschien klinkt het ingewikkelder dan het is. Als we uit bovenstaande query het artikel uit de inline-view halen en naast het uitgerekende bedrag plaatsen, dan moeten we hierop ook groeperen. Zo werken aggregaten, zoals SUM, nu eenmaal. Dan hebben we in ieder geval het bedrag per artikel. Maar hoe bepalen we nu het totaalbedrag? Sinds Oracle 8i bestaat de ROLLUP-functie als uitbereiding op de GROUP BY-clause. Met deze functie kun je subtotalen en totalen berekenen. De query komt er dan als volgt uit te zien:

```
SQL> with mee_eters
  2  as
  3  (select 8 genodigden
  4        , 1 gerecht
  5     from dual
  6  )
  7  select artikel
  8       , sum (prijs_per_eenheid * aan_te_schaffen_hoeveelheid) bedrag
  9    from (select i.soort artikel
 10               , (b.hoeveelheid/ g.aantal_personen)
 11                    * m.genodigden aan_te_schaffen_hoeveelheid
 12               , i.eenheid
 13               , i.prijs_per_eenheid
 14            from gerechten g
 15            join bereidingen b
 16              on (g.id = b.grt_id)
 17            join ingredienten i
 18              on (i.id = b.igt_id)
 19            cross
 20            join
 21            mee_eters m
 22           where g.id = m.gerecht
 23         )
 24   group by rollup (artikel)
 25  /

ARTIKEL                       BEDRAG
------------------------- ----------
boter                              1
creme fraiche                    1,8
kippenpoot                      6,25
mosterd                         0,16
olijfolie                        0,2
peper                           0,02
water                              0
zout                            0,02
                                9,45
```

Query 6.

Het overzicht zoals we het graag wilden. Op de laatste regel staat nu het totaalbedrag. In dit overzichtje is dat gemakkelijk te zien, maar hoe kun je nu het verschil zien tussen de regel met het totaalbedrag ten opzichte van een regel met een NULL-artikel en wel een bedrag?
Speciaal hiervoor is de GROUPING-functie. Indien het een subtotaal of totaalregel betreft, aangemaakt door de ROLLUP-functie, dan geeft GROUPING een 1 en anders een 0. Hiervan kunnen we gebruik maken in de bovenstaande query:

```
...
case grouping (artikel)
   when 1
   then 'Totaal Bedrag'
   else artikel
end
...

CASEGROUPING(ARTIKEL)WHEN1THEN     BEDRAG
------------------------------ ----------
boter                                   1
creme fraiche                         1,8
kippenpoot                           6,25
mosterd                              0,16
olijfolie                             0,2
peper                                0,02
water                                   0
zout                                 0,02
Totaal Bedrag                        9,45
```

Query 7.

2. Wat is het duurste gerecht in onze receptendatabase?
Om het duurste gerecht in onze receptendatabase te kunnen bepalen, moeten we eerst de prijs per gerecht bepalen. Omdat verschillende gerechten voor verschillende aantallen personen bedoeld zijn, wordt het noodzakelijk om de prijs per persoon eerst te bepalen.

```
SQL> select g.naam
  2       , sum ((b.hoeveelheid/ g.aantal_personen) * prijs_per_eenheid)
prijs_pp
  3    from gerechten g
  4    join bereidingen b
  5      on (g.id = b.grt_id)
  6    join ingredienten i
  7      on (i.id = b.igt_id)
  8   group by g.naam
  9  order by prijs_pp desc
 10  ;

NAAM                                               PRIJS_PP
-------------------------------------------------- ----------
Pizza                                                  2,19
Spekkoek                                            1,66875
Kippenpootjes met Zaanse mosterd                    1,18125
```

Query 8.

Nu we de prijs per persoon weten, word het een kwestie van sorteren en alleen de eerste laten zien.

```
SQL> select gerecht
  2       , prijs_pp
  3    from (select g.naam gerecht
  4               , sum ((b.hoeveelheid/ g.aantal_personen) * prijs_per_
eenheid) prijs_pp
  5            from gerechten g
  6            join bereidingen b
  7              on (g.id = b.grt_id)
  8            join ingredienten i
  9              on (i.id = b.igt_id)
 10          group by g.naam
 11         order by prijs_pp desc
 12        )
 13   where rownum = 1;

GERECHT                                            PRIJS_PP
-------------------------------------------------- ----------
Pizza                                                  2,19
```

Query 9.

De filtering die we toe hebben gepast om alleen de eerste te laten zien (ROWNUM = 1) moet buiten de inline staan. Op deze manier wordt eerst de sortering uitgevoerd, en daarna pas de eerste rij (die met de hoogste prijs per persoon) geselecteerd. Laat je niet verleiden om dit zonder inline-view op te lossen, want dan zal er eerst een - willekeurige -rij worden geselecteerd die vervolgens wordt geordend.

3. Welke gerechten zijn volledig vegetarisch?

Voor het uitzoeken van een vegetarisch recept moeten we gebruik maken van de BEREIDINGEN- en de INGREDIENTEN-tabellen. Om te bepalen welke gerechten alleen vegetarische ingredienten bevatten, gaan we kijken naar het verschil tussen het aantal benodigde ingredienten en het aantal vegetarische ingredienten. Als deze aantallen gelijk zijn, is het dus een volledig vegetarisch gerecht.

```
SQL> select count (b.igt_id)
  2       , sum (case i.dieet
  3               when 'vegetarisch'
  4               then 1
  5               else 0
  6             end
  7            )
  8    from bereidingen b
  9    join ingredienten i
 10      on (i.id = b.igt_id)
 11   group by b.grt_id
 12  ;

COUNT(B.IGT_ID) SUM(CASEI.DIEETWHEN'VEGETARISC
--------------- ------------------------------
              8                              7
              1                              0
             11                             10
```

Query 10.

De hierboven getoonde query toont een overzicht van het aantal ingredienten en het aantal vegetarische ingredienten. Voor het bepalen van de laatste groep maken we gebruik van een CASE-statement. Zou je deze query zonder aggregaten uitvoeren dan levert het CASE-statement een lijstje van 1 en 0 op afhankelijk van of het wel of niet een vegetarisch ingrediënt betreft. Door deze lijst van enen en nullen te sommeren krijgen wel het totaal van vegetarische gerechten.

```
SQL> select g.naam
  2    from gerechten g
  3       , (select b.grt_id
  4                 , count (b.igt_id) aant_ingredienten
  5                 , sum (case i.dieet
  6                           when 'vegetarisch'
  7                           then 1
  8                           else 0
  9                         end
 10                        ) aant_vegetarische_ingredienten
 11            from bereidingen b
 12            join ingredienten i
 13              on (i.id = b.igt_id)
 14           group by b.grt_id
 15         ) vegetarisch
 16   where g.id = vegetarisch.grt_id
 17     and vegetarisch.aant_vegetarische_ingredienten = vegetarisch.
aant_ingredienten
 18  /

no rows selected
```

Query 11.

Nu kunnen we de totalen gaan vergelijken. Wat we willen zien is de naam van het gerecht dat vegetarisch is. De eerder geschreven query wordt nu als inline-view gebruikt en gecombineerd met de GERECHTEN-tabel. Zoals je kunt zien, zijn er geen volledig vegetarische gerechten in onze gerechtendatabase. Als we besluiten om eieren wel vegetarisch te maken, op dit moment zijn ze nog aangemerkt als niet vegetarisch, dan levert de query wel een resultaat op.

```
SQL> update ingredienten
  2     set dieet = 'vegetarisch'
  3   where soort = 'eieren'
  4  /

1 row updated.

SQL> select g.naam
  2    from gerechten g
  3       , (select b.grt_id
  4                 , count (b.igt_id) aant_ingredienten
  5                 , sum (case i.dieet
  6                           when 'vegetarisch'
  7                           then 1
  8                           else 0
  9                         end
 10                       ) aant_vegetarische_ingredienten
 11            from bereidingen b
 12            join ingredienten i
 13              on (i.id = b.igt_id)
 14           group by b.grt_id
 15         ) vegetarisch
 16   where g.id = vegetarisch.grt_id
 17     and vegetarisch.aant_vegetarische_ingredienten = vegetarisch.
aant_ingredienten
 18  /

NAAM
--------------------------------------------------
Spekkoek
```

Query 12.

4. Welk gerecht is het meest complex (met de meeste bereidingsstappen)

Het meest complexe gerecht in onze receptendatabase meten we door het aantal benodigde bereidingsstappen te tellen. We kunnen dit eenvoudig oplossen door de BEREIDINGEN-tabel te gebruiken, te groeperen op het GRT_ID (de foreign key naar de GERECHTEN tabel) en een COUNT(*) te doen. Als we dit resultaat sorteren op aantal van hoog naar laag, dan staat het gerecht met de meeste bereidingsstappen bovenaan. Selecteren we van dit resultaat nu alleen de eerste rij, dan hebben we al het juiste GERECHT ID te pakken.

```
SQL> select (select g.naam
  2            from gerechten g
  3           where g.id = grt_id
  4         )
  5       , aantal_stappen
  6    from (select b.grt_id
  7                , count(*) aantal_stappen
  8             from bereidingen b
  9           group by b.grt_id
 10           order by count(*) desc
 11         )
 12   where rownum = 1
 13  /

(SELECTG.NAAMFROMGERECHTENGWHEREG.ID=GRT_ID)       AANTAL_STAPPEN
-------------------------------------------------- --------------
Spekkoek                                                       19
```

Query 13.

Maar met een ID weet je nog niet welk gerecht het betreft. In bovenstaande query 13 maken we gebruik van een Scalar Subquery om de naam uit de GERECHTEN-tabel te halen. Het resultaat, misschien niet helemaal verwonderlijk gezien de inhoud van onze receptendatabase, is spekkoek. Een heerlijk Indisch gerecht, maar nogal bewerkelijk. Gemakkelijker is het om het gewoon bij een toko te kopen. Of je laat het over aan je moeder, als deze tenminste ook Indisch is.

5. Voor een gegeven set ingredienten in de voorraadkast, welk gerecht kunnen we het best op het menu zetten?

Laten we voor deze vraag een eenvoudige voorraadkast aanleggen, wat zout, water en een diepvriespizza. Door gebruik te maken van Subquery Factoring kunnen we een lijstje van aanwezige ingrediënten samenstellen. Simpelweg selecteren van de DUAL-tabel en met een UNION ALL alles aan elkaar knopen. Als we de aanwezige voorraad nu combineren met de GERECHTEN-tabel in een cartesisch product, krijgen we per aanwezig ingrediënt een ID van de verschillende GERECHTEN.

```
SQL> with voorraad as
  2  (select 'zout' ingredient from dual
  3   union all
  4   select 'water' from dual
  5   union all
  6   select 'diepvriespizza' from dual
  7  )
  8  , voorraad_gerecht as
  9  (select v.ingredient
 10        , g.id
 11    from gerechten g
 12   cross join
 13         voorraad v
 14  )
 15  select ingredient, id
 16    from voorraad_gerecht
 17  /

INGREDIENT             ID
-------------- ----------
zout                    1
zout                    2
zout                    3
```

```
water                  1
water                  2
water                  3
diepvriespizza         1
diepvriespizza         2
diepvriespizza         3

9 rows selected.
```

Query 14.

Hiermee zijn we er nog niet. De volgende stap is een lijst maken van de verschillende ingredienten die nodig zijn per gerecht, dit is een kwestie van de BEREIDINGEN-tabel te combineren met de INGREDIENTEN-tabel. Als we nu de lijst van aanwezige ingredienten van de lijst met benodigde ingredienten afhalen blijven alleen die GERECHTEN IDs achter waar we niets voor in huis hebben.

```
SQL> with voorraad as
  2  (select 'zout' ingredient from dual
  3   union all
  4   select 'water' from dual
  5   union all
  6   select 'diepvriespizza' from dual
  7  )
  8  , voorraad_gerecht as
  9  (select v.ingredient
 10        , g.id
 11    from gerechten g
 12   cross join
 13         voorraad v
 14  )
 15  select i.soort, b.grt_id
 16    from bereidingen b
 17    join ingredienten i
 18      on (i.id = b.igt_id)
 19   minus
 20  select ingredient, id
 21   from voorraad_gerecht
 22  /

SOORT                       GRT_ID
------------------------- ----------
anijspoeder                      3
basterdsuiker                    3
boter                            1
boter                            3
creme fraiche                    1
eieren                           3
gem. kruidnagel                  3
kaneel                           3
kardemom                         3
kippenpoot                       1
mosterd                          1
nootmuskaat                      3
olijfolie                        1
peper                            1
vanillestokje                    3
vanillesuiker                    3

16 rows selected.
```

Query 15.

Zoals je kunt zien in het resultaat van bovenstaande query is het resultaat een lijst met ingredienten die voor de GERECHTEN met ID 3 en 1 nodig zijn, maar die we niet in huis hebben. De laatste stap is nu een vergelijking te maken met de GERECHTEN-tabel via een NOT IN-constructie om erachter te komen wat we zouden kunnen eten met de ingredienten die we hebben.

```
SQL> with voorraad as
  2  (select 'zout' ingredient from dual
  3   union all
  4   select 'water' from dual
  5   union all
  6   select 'diepvriespizza' from dual
  7  )
  8  , voorraad_gerecht as
  9  (select v.ingredient
 10        , g.id
 11    from gerechten g
 12   cross join
 13         voorraad v
 14  )
 15  select naam, id
 16    from gerechten
 17   where id not in
 18        (select distinct
 19                grt_id
 20           from (select i.soort, b.grt_id
 21                   from bereidingen b
 22                   join ingredienten i
 23                     on (i.id = b.igt_id)
 24                 minus
 25                 select ingredient, id
 26                   from voorraad_gerecht
 27                )
 28        )
 29  /

NAAM                            ID
------------------------ ----------
Pizza                             2
```

Query 16.

De huidige stand van de voorraadkast biedt nu dus maar een gerecht die we zouden kunnen bereiden: pizza. Niet echt een Indisch gerecht, maar wel lekker.

Ook deze keer is dit artikel, net als de source-code, te vinden op het weblog van AMIS: http://technology.amis.nl/blog/?p=3196

Slamat Makan. Eet Smakelijk.

Alex Nuijten en **Patrick Barel**, AMIS Services BV

NIEUWS

Artikelen met praktische informatie, geschreven door en bestemd voor Oracle-professionals vindt u in het Online Archief van Array Publications. Vaktijdschriften als Database Magazine, Software Release en Java Magazine hebben hun artikelenarchief online gezet. Met een heldere zoekstructuur vindt u snel wat u zoekt op www.optimize.nl.

Marco Gralike van AMIS Oracle ACE

Oracle heeft Marco Gralike, senior databaseconsultant bij AMIS benoemd tot Oracle ACE. Het Oracle ACE-programma is in het leven geroepen als aanmoediging voor personen die waarde toevoegen aan het Oracle-netwerk en hun Oracle-ervaring delen door middel van artikelen, boeken, weblogs of presentaties. Gralike staat vooral bekend als Oracle XMLDB-enthousiast. Binnen het vakgebied Oracle XMLDB is Marco Gralike de eerste Oracle ACE. Gralike is wereldwijd de 172ste ACE en de negende in Nederland. AMIS heeft vanaf 2005 al twee Oracle ACE en twee Oracle ACE Director-benoemingen in de wacht gesleept. Marco Gralike (42) is ruim vier jaar in dienst bij AMIS. Voor zijn komst naar AMIS werkte hij elf jaar als senior databaseconsultant, coach, databasebeheerder en trouble-shooter. Bij AMIS is Gralike technisch verantwoordelijk voor het remote databasebeheer van klanten en trekker van het AMIS kenniscentrum databasebeheer. Binnen het Oracle ACE-programma mogen alleen Oracle-productmanagers en andere ACE's een nieuwe ACE voordragen. Gralike werd voorgedragen door de senior productmanager voor Oracle XML Technology, onder andere naar aanleiding van Gralike's bijdrage aan het XMLDB Forum en het bètaprogramma van Oracle 11gR1. Hij kreeg het alleenrecht van Oracle om, twee weken voor de officiële release datum van Oracle 11gR1 tijdens Oracle Open World 2007, te schrijven over het nieuwe databaseproduct op de AMIS-blogsite