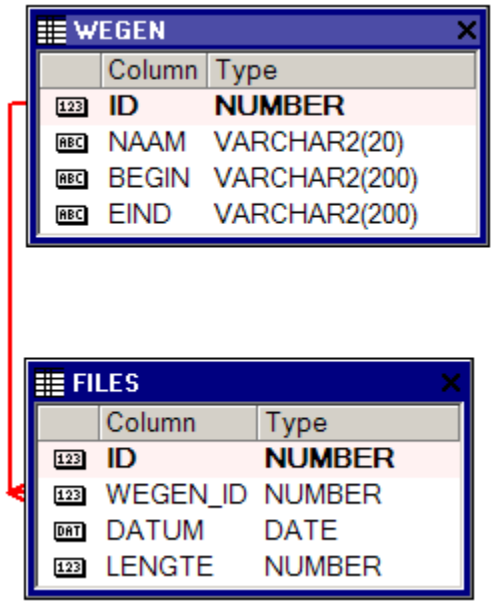
Puzzelen met SQL: Fileleed

Patrick Barel, Alex Nuijten - AMIS Services BV

Na “begin de dag met een dansje” en de NOS Headlines worden op Radio 3 de files voorgelezen. Heleen de Geest of John Bakker, van de ANWB, geeft dan een klein overzicht van de files. Giel Beelen, ‘s ochtends vroeg op de radio, vraagt altijd aan de “ANWB’er van de dag” een voorspelling van de totale lengte van de files om half negen. Is het natte vinger-werk, of gebruiken ze een betere methode om tot een voorspelling te komen? In deze “Puzzelen met SQL” gaan we eens kijken of we een soortgelijke voorspelling kunnen doen.
“Over een uurtje 180 kilometer?”.

Voor deze puzzel gebruiken we twee tabellen. Eentje om de wegen vast te leggen - de “naam” van de weg en het traject op de weg. En een tabel om de daadwerkelijke file vast te leggen, de lengte van de file en op welke datum dit was.



Laten we eerst maar eens simpel beginnen met de gegevens die we hebben, eerst maar de gemiddelde lengte per weg en dag.

```
SQL> select w.naam
  2       , avg (f.lengte) gemiddelde_lengte

3       , to_char (f.datum, 'Day', 'nls_date_language = dutch') DagVanDeWeek
  4    from files f
  5    join wegen w
  6      on w.id = f.wegen_id
  7   group by w.naam
```

```
  8          , to_char (f.datum, 'Day', 'nls_date_language = dutch')
  9  ;

NAAM                 GEMIDDELDE_LENGTE DAGVANDEW
-------------------- ----------------- ---------
A1                                   9 Zaterdag
A1                                  12 Zondag
A1                           5.66666667 Maandag
A1                                   9 Vrijdag
A2                                   0 Dinsdag
A1                           6.33333333 Dinsdag

6 rows selected.
```

Wat wellicht opvalt in het bovenstaande SQL statement is de derde parameter van de TO_CHAR functie. Het is reeds geruime tijd - zeker sinds Oracle 8.1.7. - mogelijk om een NLS parameter mee te geven om te specificeren in welke taal je de dag van de week wilt zien. Aangezien we in Nederland zijn willen we de dag van de week dan ook in het Nederlands zien: NLS_DATE_LANGUAGE = DUTCH. Deze zelfde parameter kun je ook opgeven bij de TO_DATE functie.
Let echter wel op indien je de TO_CHAR in een WHERE clause gebruikt:

```
SQL> select f.*
  2    from files f
  3   where to_char (f.datum, 'Day', 'nls_date_language = dutch') = 'Maandag'
  4  /

no rows selected
```

“No rows selected”? Is dat niet vreemd? In het vorige SQL statement is toch duidelijk te zien dat we wel gegevens hebben voor een Maandag, maar toch word er voor dit statement geen rijen getoond. Dit heeft alles te maken met de TO_CHAR functie.
TO_CHAR maakt van een NUMBER of DATE een CHAR, geen VARCHAR2. Een CHAR is per definitie een string met een vaste lengte. In het geval van de dagen van de week moet er voor gezorgd worden dat er voor dag met de langste naam voldoende ruimte is.
Zoals je in onderstaande voorbeeld kan zien, wordt iedere dag van de week aangevuld met spaties:

```
SQL> with rijen as
  2  (select rownum rn
  3     from dual
  4  connect by level <= 7
  5  )
  6  select '['||
  7         to_char (sysdate + rn, 'Day', 'nls_date_language = dutch')
  8         ||']'       dagen
  9    from rijen
 10  /
```

```
DAGEN
-----------
[Woensdag ]
[Donderdag]
[Vrijdag  ]
[Zaterdag ]
[Zondag   ]
[Maandag  ]
[Dinsdag  ]

7 rows selected.
```

In het Nederlands is Donderdag de dag met de langste naam, in het Engels is dat Wednesday:

```
SQL> with rijen as
  2  (select rownum rn
  3     from dual
  4  connect by level <= 7
  5  )
  6  select '['||
  7         to_char (sysdate + rn, 'Day', 'nls_date_language = english')
  8         ||']'       dagen
  9    from rijen
 10 /

DAGEN
-----------
[Wednesday]
[Thursday ]
[Friday   ]
[Saturday ]
[Sunday   ]
[Monday   ]
[Tuesday  ]

7 rows selected.
```

Dit is natuurlijk maar een kleinigheidje, maar soms zijn het deze kleinigheidjes waar veel tijd in gaat zitten. De gegevens die we van de files hebben zijn maar van een beperkt aantal dagen. Om een overzicht te krijgen van de gemiddelde file lengte per dag zullen we de dagen waar we geen gegevens van hebben erbij moeten “verzinnen”. Gebruik makend van Subquery Factoring (de WITH clause in onderstaand SQL statement) kunnen we de dagen van de week aan de Files tabel koppelen middels een OUTER JOIN.

```
SQL> with dagen
  2  as
  3  (select to_char (
  4             trunc (sysdate, 'iw') - 1 + rownum
```

```
  5          , 'Day', 'nls_date_language = dutch') dag
  6    from dual
  7   connect by level <= 7)
  8  select avg (f.lengte)
  9       , d.dag
 10    from dagen d
 11    left outer
 12    join files f
 13      on to_char (f.datum, 'Day', 'nls_date_language = dutch') = d.dag
 14   group by d.dag
 15  /

AVG(F.LENGTE) DAG
------------- ---------
   5.66666667 Maandag
            9 Zaterdag
              Woensdag
            9 Vrijdag
         4.75 Dinsdag
              Donderdag
           12 Zondag

7 rows selected.
```

Op regel 4 van bovenstaande code word het formaat masker “IW” gebruikt, wat voor ISO week nummer staat. Een ISO week begint altijd op maandag en eindigt altijd op zondag. In dit voorbeeld hadden we ook kunnen kiezen om het formaat masker “WW” te gebruiken, we zijn slechts geinteresseerd in de dagen van de week.

Laten we eens gaan kijken of we een voorspelling kunnen gaan doen. Om het voorbeeld eenvoudig te houden beperken we ons tot de dinsdag. Waarom de dinsdag? Omdat we op deze dag gegevens hebben van twee verschillende wegen. Let op de spaties in regel 5 bij de voluitgeschreven dag van de week.

```
SQL> select wegen_id
  2       , datum
  3       , lengte
  4    from files
  5   where to_char (datum, 'Day', 'nls_date_language = dutch') = 'Dinsdag  '
  6  /

  WEGEN_ID DATUM         LENGTE
---------- --------- ----------
         1 14-SEP-10         10
         1 21-SEP-10          7
         1 28-SEP-10          2
         2 28-SEP-10          0
```

De voorspelling gaan we doen met behulp van de MODEL clause.De MODEL clause is

geintroduceerd in Oracle 10*g* en is erg krachtig. Niet alleen krachtig maar ook lastig in gebruik. Met de MODEL clause kun je vrijwel alles doen wat met Microsoft Excel ook kan, maar dan is het ingebakken in de SQL engine. Op basis van de resultaat set word in het geheugen een array opgebouwd die vervolgens gemanipuleerd kan worden.
Even snel wat begrippen: Dimensions zijn vergelijkbaar met de aanduidingen die in Excel vergelijkbaar zijn met “A3” of “D5” ofwel de cellen. Hiermee kun je dus uniek een cel aanwijzen. De Dimension moet uniek zijn.
Measures zijn vergelijkbaar met de cellen in Excel en deze waarden mogen dan ook gemanipuleerd worden. Alle manipulaties die je doet op de Measures worden niet in de database opgeslagen, maar worden alleen uitgevoerd op de resultaat set.
De Rules bepalen de uiteindelijke manipulatie van de Measures.

```
SQL> select wegen_id
  2       , datum
  3       , lengte
  4    from files
  5   where to_char (datum, 'Day', 'nls_date_language = dutch') = 'Dinsdag  '
  6    model
  7  dimension by (wegen_id, datum)
  8   measures (lengte)
  9      rules ()
 10  /

  WEGEN_ID DATUM         LENGTE
---------- --------- ----------
         1 14-SEP-10         10
         1 28-SEP-10          2
         2 28-SEP-10          0
         1 21-SEP-10          7
```

De MODEL clause begint met het keyword MODEL, gevolgd door DIMENSION BY, MEASURES en RULES. Al deze keywords moeten aanwezig zijn, anders krijg je compilatie fouten. Het echte werk doe je in de RULES, maar deze zijn optioneel. Als je ze weg laat krijg je dan ook niets bijzonders te zien, maar gewoon de resultaten die we al eerder zagen. Alleen de kolommen die genoemd worden in de DIMENSION BY en de MEASURES clause kunnen worden gebruikt in het SELECT deel van de query.
Omdat de manipulatie van de resultaat set alleen maar “op het scherm” en niet in de database bestaat ook de mogelijkheid om extra rijen te laten genereren. In ons voorbeeld willen een extra rij genereren met de datum van volgende week dinsdag.
Deze bewerking van de resultaat set definieer je dan ook in de Rules clause. En dat is heel eenvoudig, het is een kwestie van een dimension toevoegen van een datum die nog niet bestaat. Zou je echter Measures willen bewerken die wel in de resultaat set voorkomen, dan is het een kwestie van het opgeven van de bestaande dimension en daarvan de measure een nieuwe waarde toekennen.

```
SQL> select wegen_id
  2       , datum
  3       , lengte
```

```
  4    from files
  5   where to_char (datum, 'Day', 'nls_date_language = dutch') = 'Dinsdag  '
  6    model
  7  dimension by (wegen_id, datum)
  8   measures (lengte)
  9      rules (
 10      lengte [1, to_date ('05-10-2010 07:00', 'dd-mm-yyyy hh24:mi')] = -10
 11      )
 12  /

  WEGEN_ID DATUM         LENGTE
---------- --------- ----------
         1 14-SEP-10         10
         1 21-SEP-10          7
         1 28-SEP-10          2
         2 28-SEP-10          0
         1 05-OCT-10        -10
```

Op regel 10 staat de rule die ervoor zorgt dat een extra rij getoond wordt. De syntax is wennen, eerlijk is eerlijk. Nu de uitleg. De lengte kunnen we zetten, want dat is de MEASURE. Welke lengte je dan wilt zetten wordt bepaald door de opgegeven DIMENSION. De rule die er staat kun je vrij vertalen naar: *de lengte die wordt geïdentificeerd door "wegen_id 1" en "5 oktober" krijgt de waarde - 10.*
Het opgeven van de DIMENSION staat altijd tussen rechte haken. Nu is het toekennen van de waarde -10 natuurlijk volstrekt onzinnig, het gaat om het idee. Eigenlijk willen we de gemiddelde waarde van de overige rijen op deze plaats hebben.
Aangezien zoiets in Excel mogelijk is, dan is dat met de MODEL clause ook mogelijk.

In plaats van "-10" zetten we dit in de rule:

```
avg (lengte) [1, any]
```

Dit vertaald naar: *de gemiddelde waarde van de lengtes met de dimension wegen_id 1 voor alle data in de resultaat set.*

Het totaal plaatje ziet er dan zoals onderstaande query uit, als we voor beide wegen de lengte willen voorspellen voor volgende week dinsdag.

```
SQL> select wegen_id
  2       , datum
  3       , lengte
  4    from files
  5   where to_char (datum, 'Day', 'nls_date_language = dutch') = 'Dinsdag  '
  6    model
  7  dimension by (wegen_id, datum)
  8   measures (lengte)
  9      rules (
 10         lengte [1
 11                ,to_date ('05-10-2010 07:00', 'dd-mm-yyyy hh24:mi')
 12                ] = avg (lengte) [1, any]
```

```
 13        ,lengte [2
 14                ,to_date ('05-10-2010 07:00', 'dd-mm-yyyy hh24:mi')
 15                ] = avg (lengte) [2, any]
 16      )
 17  ;

  WEGEN_ID DATUM         LENGTE
---------- --------- ----------
         1 14-SEP-10         10
         1 21-SEP-10          7
         1 28-SEP-10          2
         2 28-SEP-10          0
         2 05-OCT-10          0
         1 05-OCT-10 6.33333333
```

Krachtig, niet waar? Wel jammer dat we de rule twee keer moeten opnemen. Ook jammer als er op een gegeven moment een weg bij komt en we de query moeten uitbreiden. Is daar nu niets voor? Uiteraard is daar iets voor.
Net als bij analytische functies kun je ook bij de MODEL clause een PARTITION BY clause opnemen. Met de PARTITION BY clause kun je de resultaat set in logische stukken verdelen. Het meest logische in dit voorbeeld zou dan ook het wegen_id zijn. Per wegen_id wil je een bepaalde berekening laten uitvoeren.

```
SQL> select wegen_id
  2       , datum
  3       , lengte
  4    from files
  5   where to_char (datum, 'Day', 'nls_date_language = dutch') = 'Dinsdag  '
  6    model
  7  partition by (wegen_id)
  8  dimension by (datum)
  9   measures (lengte)
 10      rules (
 11        lengte [to_date ('05-10-2010 07:00', 'dd-mm-yyyy hh24:mi')]
 12           = avg (lengte) [any]
 13      )
 14  ;

  WEGEN_ID DATUM         LENGTE
---------- --------- ----------
         1 14-SEP-10         10
         1 21-SEP-10          7
         1 28-SEP-10          2
         1 05-OCT-10 6.33333333
         2 28-SEP-10          0
         2 05-OCT-10          0

6 rows selected.
```

De partition clause is toegevoegd op regel 7. Omdat we deze clausule toevoegen kunnen we de wegen_id uit de DIMENSION clause halen (regel 8). Het gevolg hiervan is dat de dimension die we gebruiken om de lengte te manipuleren slechts één waarde hoeft te bevatten, namelijk de datum (regel 11).

Dat het nogal complex kan worden met de MODEL clause zal je niet verwonderen. Onderstaand is een voorbeeld om een overzicht te krijgen van de files op de wegen voor volgende week. Hierin worden nog meer mogelijkheden van de MODEL clause getoond, zoals het reference model (regels 8 tot en met 16), het gebruik van het reference model (regels 22 tot en met 25) en de iteratie van de rules (regel 21).

```
SQL> select wegen_id
  2       , datum
  3       , to_char (datum, 'Day'
  4                 ,'nls_date_language=dutch') dag
  5       , lengte
  6    from files
  7    model return updated rows
  8  reference dagen
  9         on (select rownum rn
 10                  , trunc (sysdate + 7, 'iw') + rownum - 1
 11                    + interval '7' hour volgende_week
 12               from dual
 13            connect by level <= 7
 14            )
 15     dimension by (rn)
 16     measures (volgende_week)
 17  main result
 18  partition by (wegen_id)
 19  dimension by (datum, to_char (datum, 'Day') dd)
 20   measures (lengte)
 21      rules iterate (7)(
 22      lengte [dagen.volgende_week [iteration_number + 1]
 23             ,to_char (dagen.volgende_week [iteration_number + 1], 'Day')
 24             ] = avg (lengte) [any

25                              ,to_char (dagen.volgende_week
[iteration_number + 1]
 26                                       ,'Day')
 27                               ]
 28      )
 29  order by wegen_id, datum
 30  ;

  WEGEN_ID DATUM     DAG           LENGTE
---------- --------- --------- ----------
         1 04-OCT-10 Maandag   5.66666667
         1 05-OCT-10 Dinsdag   6.33333333
```

```
         1 06-OCT-10 Woensdag
         1 07-OCT-10 Donderdag
         1 08-OCT-10 Vrijdag             9
         1 09-OCT-10 Zaterdag            9
         1 10-OCT-10 Zondag             12
         2 04-OCT-10 Maandag
         2 05-OCT-10 Dinsdag             0
         2 06-OCT-10 Woensdag
         2 07-OCT-10 Donderdag
         2 08-OCT-10 Vrijdag
         2 09-OCT-10 Zaterdag
         2 10-OCT-10 Zondag

14 rows selected.
```

Voor een volledige beschrijving van de MODEL clause leent dit artikel zich niet, maar hopelijk heeft het wel je interesse gewekt om eens naar deze krachtige syntax nader te kijken.
Ook al hebben we de ANWB nu geholpen met het voorspellen van de lengte van de files, het wordt er niet leuker op om in de file te staan.