Puzzelen met SQL

Het pretpark

Het wordt tijd voor beter weer. Alleen maar sneeuw en ijzel begint na een tijdje ook maar te vervelen. En als het dan lekker weer is, dan gaan we naar een pretpark. Maar wat is dan de beste dag om zo'n pretpark te bezoeken?

Er zijn pretparken die bijhouden hoeveel mensen er komen per dag. En deze data goed bewaren, bijvoorbeeld in een Oracle-database, en hier handig gebruik van maken. Niet alleen maar om te zorgen dat er op drukke dagen voldoende personeel is om iedereen van te dure broodjes en souveniers te voorzien, maar ook om de bezoekers informatie te geven over de verwachte drukte. En om voor het nodige entertainment te zorgen als je twee uur moet wachten voor een ritje in de achtbaan van 45 seconden. Nu gaan we niet voor entertainment zorgen, maar willen we wel een voorspelling maken op basis van bezoekersaantallen op één dag - anders wordt het voorbeeld zo groot. Laten we beginnen met wat voorbeelddata van bezoekersaantallen op één dag sinds het jaar 2000.

```
SQL> with bezoekers as
  2  (select date '2000-07-15' datum, 4534 aantal from dual union all
  3   select date '2001-07-15' datum, 4590 aantal from dual union all
  4   select date '2002-07-15' datum, 4630 aantal from dual union all
  5   select date '2003-07-15' datum, 4675 aantal from dual union all
  6   select date '2004-07-15' datum, 4656 aantal from dual union all
  7   select date '2005-07-15' datum, 4689 aantal from dual union all
  8   select date '2006-07-15' datum, 4730 aantal from dual union all
  9   select date '2007-07-15' datum, 4745 aantal from dual union all
 10   select date '2008-07-15' datum, 4730 aantal from dual union all
 11   select date '2009-07-15' datum, 4745 aantal from dual
 12  )
 13  select *
 14    from bezoekers
 15  /
DATUM          AANTAL
--------- ----------
15-JUL-00        4534
15-JUL-01        4590
15-JUL-02        4630
15-JUL-03        4675
15-JUL-04        4656
15-JUL-05        4689
15-JUL-06        4730
15-JUL-07        4745
15-JUL-08        4730
15-JUL-09        4745
10 rows selected.
```

Het gebruik van de WITH-clause (in de documentatie Subquery Factoring genoemd) definieert de testdata die we gaan gebruiken. Subquery Factoring is een eenvoudige en hele krachtige manier om een resultaatset stukje bij beetje op te bouwen. Wellicht had je in bovenstaande query een TO_DATE verwacht, maar die staat er niet. Tegenwoordig is het mogelijk om een DATE te definieren door 'DATE' gevolgd door een string te gebruiken. Het formaat van de string moet dan wel YYYY-MM-DD zijn, dan kan het omgezet worden naar een echte DATE. Het formaat masker kan niet worden aangepast. Deze manier van een DATE definiëren is de ISO-methode (ISO 8601, als je het echt wil weten).
Om de kracht van Subquery Factoring aan te tonen gaan we op basis van de view 'bezoekers' het verschil in procenten bepalen ten opzichtte van het bezoekersaantal van het vorige jaar. Het is niet nodig om nog een keer de WITH clause op te nemen, een komma volstaat. De volgende view 'bezoekers_vorig_jaar' begint op regel 13 en maakt gebruik van de eerder gedefinieerde view 'bezoekers'.

```
SQL> with bezoekers as
  2  (select date '2000-07-15' datum, 4534 aantal from dual union all
  3   select date '2001-07-15' datum, 4590 aantal from dual union all
  4   select date '2002-07-15' datum, 4630 aantal from dual union all
  5   select date '2003-07-15' datum, 4675 aantal from dual union all
  6   select date '2004-07-15' datum, 4656 aantal from dual union all
  7   select date '2005-07-15' datum, 4689 aantal from dual union all
  8   select date '2006-07-15' datum, 4730 aantal from dual union all
  9   select date '2007-07-15' datum, 4745 aantal from dual union all
 10   select date '2008-07-15' datum, 4730 aantal from dual union all
 11   select date '2009-07-15' datum, 4745 aantal from dual
 12  )
 13  ,bezoekers_vorig_jaar as
 14  (select datum
 15        , aantal
 16        , lag (aantal) over (order by datum) vorig_jaar
 17    from bezoekers
 18  )
 19  select datum
 20        , aantal
 21        , vorig_jaar
 22        , round (
 23           (aantal - vorig_jaar)
 24           / vorig_jaar * 100
 25         , 2) "%"
 26    from bezoekers_vorig_jaar
 27  ;
```

```
DATUM          AANTAL VORIG_JAAR          %
--------- ---------- ---------- ----------
15-JUL-00        4534
15-JUL-01        4590       4534       1.24
15-JUL-02        4630       4590        .87
15-JUL-03        4675       4630        .97
15-JUL-04        4656       4675       -.41
15-JUL-05        4689       4656        .71
15-JUL-06        4730       4689        .87
15-JUL-07        4745       4730        .32
15-JUL-08        4730       4745       -.32
15-JUL-09        4745       4730        .32
10 rows selected.
```

De eigenlijke query, beginnende op regel 19, maakt gebruik van de view 'bezoekers_vorig_jaar'. In versies voor Oracle 11g is het verplicht dat de gedefinieerde views ook gebruikt worden in het SQL-statement. Met behulp van de Analytische Functie LAG kunnen we waarden uit andere rijen tonen. De LAG-functie haalt waarden uit rijen die al in eerdere rijen zijn geweest. Zo is in de tweede regel in de resultaatset in de kolom 'VORIG_JAAR' het bezoekersaantal te zien dat in de voorgaande rij in de kolom 'AANTAL' te vinden is.
Op basis van het huidige bezoekersaantal en het bezoekersaantal van het voorgaande jaar is het verschil in procenten eenvoudig af te leiden.
Nog even een kleinigheidje tussendoor: de naam van de laatste kolom is "%". Om dit voor elkaar te krijgen staat de alias voor deze kolom tussen dubbele aanhalingstekens. Zonder de dubbele aanhalingstekens is het niet toegestaan om een procentteken te gebruiken als alias. Wil je verderop in de query aan deze kolom refereren, en dat willen we - we zijn tenslotte nog niet klaar - dan dien je de kolomnaam tussen dubbele aanhalingsteken te blijven plaatsen.

```
SQL> select 'tekst' %
  2    from dual
  3  /
select 'tekst' %
               *
ERROR at line 1:
ORA-00911: invalid character
```

Tot nu toe hebben we een overzicht van het aantal bezoekers vanaf het jaar 2000, met het percentageverschil per jaar. Nu zouden we op basis van deze gegevens een voorspelling willen doen met betrekking tot het aantal bezoekers voor het komende jaar. Het mooiste zou natuurlijk zijn als deze in de resultaatset als extra rij komt te staan, maar alleen maar op het scherm en niet in de database - het is tenslotte nog geen 15 juli en het daadwerkelijke bezoekersaantal staat nog niet vast. Het is natuurlijk altijd mogelijk om alvast een rij in de tabel op te nemen met een dummywaarde, maar echt een mooie oplossing is het niet.
Juist voor dit soort dingen heeft Oracle de MODEL clause bedacht. Met de MODEL clause kun je Excel-achtige functionalitieit in SQL brengen. Bijna alles wat je met data in een Excelsheet kunt doen, kun je ook met de MODEL clause. Heel krachtig, maar de syntax is nu niet bepaald een makkie.
Laten we de MODEL clause eens introduceren in onze query:

```
SQL> with bezoekers as
  2  (select date '2000-07-15' datum, 4534 aantal from dual union all
  3   select date '2001-07-15' datum, 4590 aantal from dual union all
  4   select date '2002-07-15' datum, 4630 aantal from dual union all
  5   select date '2003-07-15' datum, 4675 aantal from dual union all
  6   select date '2004-07-15' datum, 4656 aantal from dual union all
  7   select date '2005-07-15' datum, 4689 aantal from dual union all
  8   select date '2006-07-15' datum, 4730 aantal from dual union all
  9   select date '2007-07-15' datum, 4745 aantal from dual union all
 10   select date '2008-07-15' datum, 4730 aantal from dual union all
 11   select date '2009-07-15' datum, 4745 aantal from dual
 12  )
 13  ,bezoekers_vorig_jaar as
 14  (select datum
 15        , aantal
 16        , lag (aantal) over (order by datum) vorig_jaar
 17    from bezoekers
 18  )
 19  , percentage as
 20  (
 21  select datum
 22        , aantal
 23        , vorig_jaar
 24        , round (
 25           (aantal - vorig_jaar)
 26           / vorig_jaar * 100
 27         , 2) "%"
 28    from bezoekers_vorig_jaar
 29  )
 30  select datum
 31        , aantal
 32        , vorig_jaar
 33        , "%"
 34    from percentage
 35  model
 36  dimension by (datum)
 37  measures (aantal, vorig_jaar, "%")
 38  rules ()
 39  ;
DATUM          AANTAL VORIG_JAAR          %
--------- ---------- ---------- ----------
15-JUL-00        4534
15-JUL-01        4590       4534       1.24
15-JUL-02        4630       4590        .87
15-JUL-03        4675       4630        .97
15-JUL-04        4656       4675       -.41
15-JUL-05        4689       4656        .71
15-JUL-06        4730       4689        .87
15-JUL-07        4745       4730        .32
15-JUL-08        4730       4745       -.32
15-JUL-09        4745       4730        .32
10 rows selected.
```

Op regel 35 begint de MODEL clause, gevolgd door DIMENSION BY, MEASURES en RULES. Bij de DIMENSION geef je aan hoe de individuele cellen te benaderen zijn, een beetje te vergelijken met een primary key op een tabel. De MEASURES zijn de cellen waar de waarden in staan die we kunnen manipuleren of kunnen gebruiken in onze manipulaties. Deze velden (cellen) hoeven niet in de select voor te komen. Bij de RULES geven we aan op welke manier we de MEASURES gaan manipuleren.

De resultaatset is hetzelfde als we eerder zagen, ook al staat de MODEL clause nu in de query. Toch ziet het er hetzelfde uit,

	A	B	C	D
1	DATUM	AANTAL	VORIG JAAR	%
2	15-7-2000	4534		
3	15-7-2001	4590	4534	1,24
4	15-7-2002	4630	4590	0,87
5	15-7-2003	4675	4630	0,97
6	15-7-2004	4656	4675	-0,41
7	15-7-2005	4689	4689	0,71
8	15-7-2006	4730	4630	0,87
9	15-7-2007	4745	4730	0,32
10	15-7-2008	4730	4745	-0,32
11	15-7-2009	4745	4730	0,32
12	15-7-2010	4760.184		

maar eigenlijk is dit niet zo. Normaliter zie je data die op enig moment in de tijd de waarheid bevatte zoals deze in de database opgeslagen was. Bij gebruik van de MODEL clause kun je gegevens zien die helemaal niet in de database zijn opgeslagen of opgeslagen zijn geweest. De data die je ziet bestaat alleen op het scherm. Nu hebben we de data nog niet gemanipuleerd, dus is de resultaatset gelijk aan die in de database - als we deze hadden opgeslagen. Dit is wel een belangrijk punt, de data bestaat alleen op het scherm. De voorspelling die we dadelijk gaan doen wordt niet in de database opgeslagen.
Laten we met behulp van de MODEL clause eens een extra rij aanmaken voor de dag in 2010, deze willen we tenslotte gaan voorspellen. Om dit te kunnen doen moeten we een Rule definieren.

```
aantal [date '2010-07-15'] = 0
```

Deze regel zegt: de MEASURE aantal krijgt voor de DIMENSION date '2010-07-15' de waarde nul.
Om het effect hiervan in een query te zien:

```
SQL> with bezoekers as
  2  (select date '2000-07-15' datum, 4534 aantal from dual union all
  3   select date '2001-07-15' datum, 4590 aantal from dual union all
  4   select date '2002-07-15' datum, 4630 aantal from dual union all
  5   select date '2003-07-15' datum, 4675 aantal from dual union all
  6   select date '2004-07-15' datum, 4656 aantal from dual union all
  7   select date '2005-07-15' datum, 4689 aantal from dual union all
  8   select date '2006-07-15' datum, 4730 aantal from dual union all
  9   select date '2007-07-15' datum, 4745 aantal from dual union all
 10   select date '2008-07-15' datum, 4730 aantal from dual union all
 11   select date '2009-07-15' datum, 4745 aantal from dual
 12  )
 13  ,bezoekers_vorig_jaar as
 14  (select datum
 15        , aantal
 16        , lag (aantal) over (order by datum) vorig_jaar
 17    from bezoekers
 18  )
 19  , percentage as
 20  (
 21  select datum
 22        , aantal
 23        , vorig_jaar
 24        , round (
 25           (aantal - vorig_jaar)
 26           / vorig_jaar * 100
 27          , 2) "%"
 28    from bezoekers_vorig_jaar
 29  )
 30  select datum
 31        , aantal
 32        , vorig_jaar
 33        , "%"
 34    from percentage
 35  model
 36  dimension by (datum)
 37  measures (aantal, vorig_jaar, "%")
 38  rules (aantal [date '2010-07-15'] = 0)
 39  ;
DATUM          AANTAL VORIG_JAAR          %
--------- ---------- ---------- ----------
15-JUL-00        4534
15-JUL-01        4590       4534       1.24
15-JUL-02        4630       4590        .87
15-JUL-03        4675       4630        .97
15-JUL-04        4656       4675       -.41
15-JUL-05        4689       4656        .71
15-JUL-06        4730       4689        .87
15-JUL-07        4745       4730        .32
15-JUL-08        4730       4745       -.32
15-JUL-09        4745       4730        .32
15-JUL-10           0
11 rows selected.
```

Nu is er een regel aan de resultaatset toegevoegd met de datum 15-JUL-2010 en de waarde in de kolom 'Aantal' is gevuld met nul. Het zou natuurlijk wel fijn zijn als er verder geen bezoekers waren in het pretpark, maar alleen is ook maar alleen. We willen gaan bepalen wat het bezoekersaantal zou zijn indien de toename in bezoekersaantal van vorig jaar ook dit jaar zou plaatsvinden. We willen dus eigenlijk het aantal van de vorige regel vermeerderen met het percentage op diezelfde regel.
Maar hoe gaan we dit doen? Eerst maar het aantal van vorig jaar ophalen. We willen dus de MEASURE aantal van 15-JUL-2009 ophalen. Om het nu niet hard te coderen maken we gebruik van de huidige waarde van de DIMENSION. Dit doen we door de CV() functie te gebruiken. CV staat dan ook voor Current Value.

```
aantal [cv() - interval '1' year]
```

In woorden: de Measure aantal die wordt geïdentificeerd door de Dimension huidige waarde van de dimesion min één jaar. Op deze wijze kunnen we ook het percentage van vorig jaar ophalen. Als we dit doen en een klein rekensommetje erop loslaten, dan hebben we onze voorspelling:

```
SQL> with bezoekers as
  2  (select date '2000-07-15' datum, 4534 aantal from dual union all
  3   select date '2001-07-15' datum, 4590 aantal from dual union all
  4   select date '2002-07-15' datum, 4630 aantal from dual union all
  5   select date '2003-07-15' datum, 4675 aantal from dual union all
  6   select date '2004-07-15' datum, 4656 aantal from dual union all
  7   select date '2005-07-15' datum, 4689 aantal from dual union all
  8   select date '2006-07-15' datum, 4730 aantal from dual union all
  9   select date '2007-07-15' datum, 4745 aantal from dual union all
 10   select date '2008-07-15' datum, 4730 aantal from dual union all
 11   select date '2009-07-15' datum, 4745 aantal from dual
 12  )
 13  ,bezoekers_vorig_jaar as
 14  (select datum
 15        , aantal
 16        , lag (aantal) over (order by datum) vorig_jaar
 17    from bezoekers
 18  )
 19  , percentage as
 20  (
 21  select datum
 22        , aantal
 23        , vorig_jaar
 24        , round (
 25           (aantal - vorig_jaar)
 26           / vorig_jaar * 100
 27         , 2) "%"
 28    from bezoekers_vorig_jaar
 29  )
 30  select datum
 31        , aantal
 32        , vorig_jaar
 33        , "%"
 34    from percentage
 35  model
 36  dimension by (datum)
 37  measures (aantal, vorig_jaar, "%")
 38  rules (aantal [date '2010-07-15'] =
 39          aantal [cv() - interval '1' year] +
 40          ((aantal [cv() - interval '1' year] * "%" [cv() - interval
'1' year])/100)
 41  )
 42  ;
DATUM          AANTAL VORIG_JAAR          %
--------- ---------- ---------- ----------
15-JUL-00        4534
15-JUL-01        4590        4534       1.24
15-JUL-02        4630        4590        .87
15-JUL-03        4675        4630        .97
15-JUL-04        4656        4675       -.41
15-JUL-05        4689        4656        .71
15-JUL-06        4730        4689        .87
15-JUL-07        4745        4730        .32
15-JUL-08        4730        4745       -.32
15-JUL-09        4745        4730        .32
15-JUL-10   4760.184
11 rows selected.
```

We vermoeden dat er ongeveer 4760 bezoekers zijn op 15 juli 2010. Als je vindt dat we beter het gemiddelde percentage kunnen gebruiken, dan kan dat ook. Excel biedt de mogelijkheid om te refereren aan een aantal cellen, dit kan ook met de MODEL clause.

```
avg("%")[any]
```

Met ANY geven we aan dat we van alle Measures (cellen) het gemiddelde percentage willen bepalen. Het resultaat van de query wordt dan:

```
DATUM          AANTAL VORIG_JAAR          %
--------- ---------- ---------- ----------
15-JUL-00        4534
15-JUL-01        4590        4534       1.24
15-JUL-02        4630        4590        .87
15-JUL-03        4675        4630        .97
15-JUL-04        4656        4675       -.41
15-JUL-05        4689        4656        .71
15-JUL-06        4730        4689        .87
15-JUL-07        4745        4730        .32
15-JUL-08        4730        4745       -.32
15-JUL-09        4745        4730        .32
15-JUL-10 4769.09406
```

De rule ziet er dan als volgt uit:

```
aantal [date '2010-07-15'] =
       aantal [cv() - interval '1' year] +
       ((aantal[cv() - interval '1' year] * avg("%")[any])/100)
```

Toch weer tien mensen meer en dat betekent langer wachten. En als het dan toch niet te druk is, dan kunnen we wel een paar keer in de achtbaan. Misselijk maar voldaan weer naar huis.

Patrick Barel is consultant bij AMIS Services. Hij schrijft op het blog van AMIS (http://technology.amis.nl/blog) en op zijn eigen blog (http://blog.bar-solutions.com)

Alex Nuijten is Oracle-consultant bij AMIS Services. Hij schrijft op het blog van AMIS (http://technology.amis.nl/blog) en op zijn eigen blog (http://nuijten.blogspot.com).