Genereren van testdata

Een van de grote en vervelende uitdagingen bij het maken van SQL-puzzels, maar belangrijker nog bij het bouwen van de demo-applicaties en vooral bij het testen van nieuw ontwikkelde applicaties, is het samenstellen van een set van voorbeeld-, demo- of test-data. Eindeloos voornamen, achternamen, plaatsnamen en geboortedata invoeren is stomvervelend en als er ook nog eisen aan de kwaliteit worden gesteld - zoals een evenwichtige verdeling over een waardebereik of uniciteit van bepaalde combinaties van velden - wordt het helemaal een nachtmerrie.

Er zijn (open source) tools verkrijgbaar die test-data genereren, maar in deze puzzel gaan we dat zelf doen, met niets anders dan SQL tot onze beschikking. Het einddoel van de opdracht is om in de PERSONEN tabel 250 fictieve personen aan te maken die aan de volgende eisen voldoen:

- de combinatie voornaam, achternaam en straatnaam is uniek
- de huisnummers zijn willekeurig gekozen uit de reeks 1..125
- de geboortedata zijn willekeurig gekozen tussen 1-1-1930 en 31-12-2000
- de initialen bestaan uit een letter en zijn ook willekeurig gekozen

Er zijn drie hulptabellen beschikbaar die al gegevens bevatten: VOORNAMEN, ACHTERNAMEN en STRAATNAMEN.

PERSONEN

Column	Type	Nullable	Default	Comments
INITIALEN	VARCHAR2(25)	Y		
VOORNAAM	VARCHAR2(25)	Y		
ACHTERNAAM	VARCHAR2(50)	Y		
STRAATNAAM	VARCHAR2(50)	Y		
HUISNUMMER	NUMBER(3)	Y		
GEBOORTEDATUM	DATE	Y		

In de tabel PERSONEN worden 250 fictieve personen aangemaakt.

1) Schrijf een query die willekeurige waarden genereert uit de reeks van 1 tot 125.

```
with input as
(select 250 aantal_waarden
       , 1 ondergrens
       , 125 bovengrens
   from dual
)
, random_numbers as
(select round (
            ondergrens + (bovengrens - ondergrens)
            * dbms_random.value
               ) value
   from all_source
       , input
  where rownum <= aantal_waarden
)
select value
  from random_numbers
```

Het eerste waar je waarschijnlijk aan denkt als je 'willekeurig' leest, is DBMS_RANDOM. Dit built-in package is een random number generator. De functie genaamd VALUE genereert een willekeurig nummer tussen de 0 en de 1 met een precisie van 38 getallen achter de komma. Omdat er zoveel getallen achter de komma kunnen staan maken we gebruik van de ROUND functie om het gegenereerde getal af te ronden.
Gebruikmakend van Subquery Factoring (de WITH clause waar het statement mee begint) word stapsgewijs een nummergenerator gebouwd.
Het zou natuurlijk een stuk handiger zijn als Oracle zelf al zo'n soort functionaliteit zou bieden waarbij een willekeurig nummer wordt gegenereerd en je als argument de onder- en bovengrens kan ingeven. Laat dat nu net het geval zijn.
Er is van de VALUE functie binnen het DBMS_RANDOM een overloading beschikbaar die precies doet wat we willen. De specificatie ziet er zo uit:

```
-- get a random Oracle number x, low <= x < high
 FUNCTION value (low IN NUMBER, high IN NUMBER) RETURN NUMBER;
```

Advertentie

De gehele nummergenerator zoals we die zojuist hebben gemaakt, kan dan ook vervangen worden door:

```
select round (dbms_random.value (1, 125))
  from all_source
 where rownum <= 250
```

Let erop dat het bovengrens niet inclusief is. Als je dus wilt dat 125 ook een mogelijk nummer is dan zal het laatste argument 126 moeten zijn, en zal de ROUND moeten wijzigen naar een TRUNC. We gebruiken hier trouwens de tabel all_source als een dummy tabel om 250 records te selecteren, zie http://technology.amis.nl/blog/?p=392 voor een andere manier om records te genereren.

2) Maak een query die geboortedata genereert, tussen 1-1-1930 en 31-12-2000

Helaas heeft het DBMS_RANDOM package geen functie om data te genereren. Hier zullen we dus zelf creatief moeten worden.

```
with input as
(select 250 aantal_waarden
      , 0 ondergrens
      , to_date ('31-12-2000','DD-MM-YYYY')
        -  to_date ('01-01-1930','DD-MM-YYYY') bovengrens
   from dual
)
, random_numbers as
( select round (dbms_random.value (ondergrens, bovengrens)) value
    from all_source
       , input
   where rownum <= aantal_waarden
)
select to_date ('01-01-1930', 'dd-mm-yyyy') + value geboorte_datum
  from random_numbers
;
GEBOORTE_
---------
01-DEC-94
17-MAY-95
17-JAN-56
05-AUG-69
03-OCT-36
22-MAY-88
19-OCT-59
11-SEP-74
...
```

Ook deze query maakt gebruikt van Subquery Factoring. Met behulp van Subquery Factoring kun je een naam geven aan een inline view. Hier hebben we als eerste een inline view gedefinieerd genaamd "Input". Deze inline view bestaat slechts uit een enkele rij met daarin de boven- en ondergrenzen die we nodig hebben in het tweede deel van de query.

Het tweede stukje is waar de willekeurige datum bepaald wordt. Deze inline view, genaamd "Random_Numbers", maakt gebruik van de eerder gedefinieerde Input inline view.
Als laatste stap de uiteindelijke query waar het allemaal om draait. Hier word een selectie gemaakt uit de inline view Random_Numbers.
Het is echter wel een voorwaarde dat de inline view die je definieerd ook daadwerkelijk wordt gebruikt in de query. Ofwel in een andere inline view (zoals Input die gebruikt word in Random_Numbers), ofwel in de hoofdquery (zoals Random_Numbers gebruikt wordt). Je kan alleen maar gebruik maken van een inline view die reeds eerder in het statement is gedeclareerd. Het omdraaien van de inline views is dus niet toegestaan:

```
with random_numbers as
( select ...
    from grenzen
  ...
)
,grenzen as
(select ...
   from dual
)
...
```

Dit zal leiden tot een Oracle-foutmelding:

```
ORA-32031: illegal reference of a query name in WITH clause
```

3) Maak een query die initialen genereert

Het is natuurlijk eenvoudig om voort te borduren op de generatoren zoals we die tot nu toe hebben geschreven, maar het is nog een stuk eenvoudiger om gebruik te maken van iets wat Oracle reeds voor ons gemaakt heeft. In het DBMS_RANDOM package zit ook een functie genaamd STRING.
Deze functie geeft een willekeurige tekenreeks terug. Deze kan bestaan uit hoofdletters, kleine letters, een mix van hoofd- en kleine letters, letters en cijfers door elkaar of alleen karakters die printable zijn. Hier is de functie specificatie:

```
-- get a random string
FUNCTION string (opt char, len NUMBER)
```

Met het eerste argument geef je aan wat voor soort string je terug wilt krijgen. Hieronder een lijstje van mogelijkheden:
- U – Alleen hoofdletters
- L – Alleen kleine letters
- A – Hoofd- en kleine letters door elkaar

• X – Alpha numerieke karakters
• P – Printable karakters

Het maakt trouwens niet uit of dit argument een hoofd- of kleine letter is.

Het tweede argument is de lengte van de string die je terug wilt hebben. Om willekeurige initialen te generen kun je dus volstaan met een query als:

```
select dbms_random.string ('u', 1)
  from all_source
 where rownum <= 250
```

4) Schrijf een stukje SQL dat op basis van de hulptabellen VOORNAMEN, ACHTERNAMEN en STRAATNAMEN unieke combinaties maakt van Voornaam, Achternaam en Straatnaam met liefst een willekeurige verdeling van alle voorkomende waarden in de hulptabellen (dus liefst niet 250 keer Jan met wisselende combinaties van Achternaam en Straatnaam).

```
select voornaam
     , achternaam
     , straatnaam
  from ( select *
           from voornamen
              , achternamen
              , straatnamen
          order by ORA_HASH (rownum)
       )
 where rownum <= 250
;
VOORNAAM                ACHTERNAAM              STRAATNAAM
----------------------- ----------------------- ----------------
Peter                    Bollebob                Kerkplein
Wim                      Postma                  Kerkplein
Peter                    Vink                    Kalverstraat
Anton                    Bollebob                Kennedyboulevard
Margriet                 Geldof                  Kennedyboulevard
Mieke                    van Dijk                Tijgerplein
John                     Bomer                   Kerkplein
Anton                    Jansen                  Eikenplantsoen
Martijn                  Wijers                  Hoeverdam
Maarten                  Postma                  Kennedyboulevard
Jan                      Jansen                  Hoeverdam
```

De basis van deze query is een Cartetisch product van de Voornamen, Achternamen en Straatnamen tabellen. Het meest opvallende uit deze query is de ORA_HASH functie. ORA_HASH bepaald een hash waarde voor een gegeven argument. Waar is dat nu handig voor? Bijvoorbeeld om een willekeurig set gegevens te genereren, zoals in bovenstaande query. Een andere mogelijkheid om een willekeurige set te krijgen, is door de ORDER BY te vervangen door:

```
order by DBMS_RANDOM.VALUE
```

5) Laad de PERSONEN tabel met test-data volgens de voorwaarden die hierboven beschreven zijn.

Met behulp van de queries die we hiervoor hebben gemaakt, is het nu niet meer moeilijk om de PERSONEN tabel te vullen.

```
insert into personen( initialen, voornaam, achternaam, straatnaam,
huisnummer, geboortedatum )
select dbms_random.string ('u', 1)
      , voornaam
      , achternaam
      , straatnaam
      , trunc( dbms_random.value (1, 126) )
      , to_date ('01-01-1930', 'dd-mm-yyyy') +
          trunc( dbms_random.value (0
                                    , to_date ('01-01-2001', 'dd-mm-
yyyy')
-          to_date ('01-01-1930', 'dd-mm-yyyy')
)
               )
from ( select *
         from voornamen
            , achternamen
            , straatnamen
         order by ORA_HASH (rownum)
      )
where rownum <= 250
;
INITI VOORNAAM  ACHTERNAAM    STRAATNAAM        HUISNR  GEBOORTE_DATUM
----- --------  -----------   ---------------   ------  --------------
W     Martijn   Riksen        Julianalaan       55       11-7-1962
J     Hans      Van Rossum    Eikenplantsoen    125      5-9-1941
Z     Tom       Postma        Koedijkerweg      23       21-3-1950
A     Hans      Van Rossum    Kalverstraat      26       25-10-1931
...
```

Zie http://technology.amis.nl/blog/?p=2418 voor aanvullende informatie. Hier zijn ook de gebruikte scripts te verkrijgen om zelf het een en ander uit te proberen.

Anton Scheffer en Alex Nuijten zijn werkzaam bij AMIS.