Hidden Secrets: die SQL Model Clause

Carsten Czarski, ORACLE Deutschland B.V. & Co. KG

Bereits mit Oracle10*g* wurde die SQL Model Clause eingeführt – sie macht es möglich, im Ergebnis einer SQL-Abfrage so mit Formeln zu rechnen, wie man es von einem Tabellenkalkulations-Programm gewohnt ist. Dieser Artikel beschreibt die Funktionalität der SQL Model Clause und gibt einen Einblick, was mit dieser mächtigen Abfrageform alles möglich ist.

Die SQL Model Clause betrachtet die Ergebnismenge einer Abfrage wie ein Arbeitsblatt einer Tabellenkalkulation – jeder Wert kann, wie eine Zelle, angesprochen und verändert werden. Möchte man die Funktion nutzen, so wird das mit dem Schlüsselwort „model“ eingeleitet. Danach wird festgelegt, wie die Zellen adressiert werden – möchte man das Gehalt („SAL“) von KING („ENAME“) adressieren, muss „ENAME“ zur Dimension („dimensions“) und „SAL“ zum Wert („measures“) werden. Die dann nachfolgenden Formeln („rules“) verändern schließlich die Ergebnismenge, die abschließend an den Aufrufer zurückgegeben wird. Listing 1 zeigt ein erstes Beispiel.

Innerhalb der „RULES“-Klausel werden also die Formeln eingegeben; die als „measure“ deklarierte Tabellenspalte wird mit Namen angesprochen – und in den eckigen Klammern anhand der als „dimensions“ festgelegten Tabellenspalten navigiert. „RULES UPSERT“ bedeutet, dass die Formeln sowohl bestehende Zeilen der Ergebnismenge ändern als auch neue erzeugen können. Ein „RULES UPDATE“ würde nur die Formeln ausführen, die bereits vorhandene Ergebniszeilen betreffen. Wie an den Formeln sehr schön erkennbar ist, macht die SQL Model Clause etwas möglich, das man aus der Welt von SQL eigentlich nicht kennt: Man kann – aus jeder Ergebniszeile heraus – auf jede andere Ergebniszeile zugreifen. Neben einfachen Zuweisungen sind auch Funktionen möglich. In Listing 1 wird das Gehalt des „EMPNO 9999“ mit dem Durchschnitt aller Gehälter gleichgesetzt.

In den Formeln in Listing 1 werden die „Zellen“ absolut adressiert („sal[7566]“). Aus Tabellenkalkulationen kennt man aber auch die „relative

```
select empno, ename, hiredate, sal, comm from emp
model
 dimension by (empno)
 measures     (name, hiredate, sal, comm)
 rules upsert (
   -- Hier werden “Formeln” als “Rules” eingebeben!
   ename[7499] = ‘SCHMIDT’,
   sal[7566]   = sal[7499] + 100 + (sal[7839] / 2),
   sal[9999]   = avg(sal)[ANY]
 )
order by empno
```

Listing 1: Ein einfaches Beispiel für die SQL Model Clause

Adressierung“ – und auch das ist mit der SQL Model Clause mit der Funktion „CV()“ möglich. Listing 2 schlägt für jede Zeile in der Tabelle „EMP“ das jeweils nächsthöhere Gehalt vor. Hierfür muss die Dimension „EMPNO“ jedoch durch eine fortlaufende Nummer („ROWNUM“) ersetzt werden, damit man über „CV() + 1“ auf die jeweils nächste Zeile navigieren kann.

```
select zeile, empno, ename, sal, sal_new from (
  select * from emp order by sal
)
model
 dimension by (rownum zeile)
 measures (empno, ename, sal, 0 as sal_new)
 rules upsert (
   -- Hier werden “Formeln” als “Rules” eingebeben!
   sal_new[zeile] = sal[cv(zeile) + 1]
 )
order by sal asc;
```

Listing 2: Mit „CV()“ kann man in der SQL Model Clause auch relativ adressieren

```
select add_months('2013-01-01', zeile - 1) m, a, b, c, d, e, f
from dual
model
dimension by (rownum zeile)
measures (200000 a, 0 b, 0 c, 1500 d, 0 e, 0 f)
rules upsert iterate (300) until (a[ITERATION_NUMBER]<=0)(
  -- Formeln hier
)
order by zeile;
```

Listing 3: Die Grundstruktur der Abfrage

```
a[0] = 200000
b[ITERATION_NUMBER]   = a[cv(zeile)] * 0.06 / 12,
d[ITERATION_NUMBER]   = least(1500, a[cv(zeile)] +
b[cv(zeile)]),
c[ITERATION_NUMBER]   = d[cv(zeile)] - b[cv(zeile)],
a[ITERATION_NUMBER+1] = a[cv(zeile)-1] - c[cv(zeile)-1]
```

Listing 4: Die Formeln ...

```
M                    A          B          C          D
---------- ---------- ---------- ---------- ----------
2012-12-01  200000,00    1000,00     500,00    1500,00
2013-01-01  199500,00     997,50     502,50    1500,00
2013-02-01  198997,50     994,99     505,01    1500,00
2013-03-01  198492,49     992,46     507,54    1500,00
2013-04-01  197984,95     989,92     510,08    1500,00
         :          :          :          :          :
```

Listing 5: … und das Ergebnis

Dass man mit der SQL Model Clause tatsächlich die Möglichkeiten einer Tabellenkalkulation hat, wird auch am nächsten Beispiel deutlich. Wir setzen das „Lehrbuchbeispiel“ für Tabellenkalkulationen um – der Tilgungsplan eines Darlehens soll generiert werden. Listing 3 zeigt zunächst die grundlegende Struktur der SQL-Abfrage.

Die Spalten „A“ und „D“ sind mit 200.000 („Betrag“) und 1.500 („Rate“) initialisiert. Die Formeln sollen 300-mal angewendet werden, maximal jedoch solange, bis die Spalte „A“ auf 0 reduziert ist. Nun kommen allerdings die Formeln – diese werden einfach in die „RULES UPSERT“-Klausel eingesetzt (siehe Listing 4). Die SQL-Abfrage liefert dann einen kompletten Tilgungsplan zurück (siehe Listing 5).

Weitere Informationen

Blog-Posting des Autors, SQL Model Clause: http://sql-plsql-de.blogspot.co.uk/2008/06/rechnen-wie-in-excel-in-einer-sql.html
SQL Model Clause in der deutschsprachigen Apex-Community: http://apex.oracle.com/url/apxmodel
Oracle Data Warehousing Guide, SQL for modeling: http://docs.oracle.com/cd/E11882_01/server.112/e25554/sqlmodel.htm#i1011770

Carsten Czarski
carsten.czarski@oracle.com
http://twitter.com/cczarski
http://sql-plsql-de.blogspot.com