Tipps & Tricks: SQL Erweiterungen

Bereich:	SQL	Erstellung:	02/2006 MP
Versionsinfo:	10.1, 10.2, 11.1, 11.2	Letzte Überarbeitung:	06/2009 MA

SQL Erweiterungen

Unterabfragen:

Unterabfragen sind überall erlaubt, abgesehen von:

- Default-Werten von Spalten,
- Return-Klauseln,
- Hash-Ausdrücken für Cluster,
- funktionsbasierte Indizes,
- Check-Constraints,
- WHEN-Klauseln bei Triggern,
- GROUP BY-Klauseln

Beispiel:

```
SQL> SELECT d.dname,
  2         (SELECT count(*)
  3          FROM emp e WHERE e.deptno = d.deptno) mitarbeiter
  4  FROM dept d;


DNAME          MITARBEITER
-------------- -----------
ACCOUNTING               3
RESEARCH                 5
SALES                    6
OPERATIONS               0
```

Expliziter Default:

Um den angegebenen Default-Wert einer Tabellenspalte explizit zu referenzieren, kann nun das Schlüsselwort DEFAULT im SQL-Statement einer INSERT- oder UPDATE-Anweisung verwendet werden.

Beispiele:

```
SQL> INSERT INTO emp (empno, ename, deptno)
               VALUES (1234,'ROSEN',default);

SQL> UPDATE emp SET deptno = DEFAULT
  2  WHERE job = 'SALESMAN';
```

WITH-Klausel:

In komplexen Abfragen, in denen dieselbe Unterabfrage mehrfach verarbeitet wird, ist man geneigt, die Ergebnisse der Unterabfrage zunächst in einer temporären Tabelle abzulegen und dann weitere Abfragen zu erstellen, die diese temporäre Tabelle referenzieren.
Die WITH-Klausel ermöglicht es, der innerhalb einer Abfrage mehrfach auftretenden Unterabfrage einen Namen zuzuweisen und im weiteren Statement diesen Namen entsprechend zu referenzieren. Somit bleibt es dem Optimizer vorbehalten, die Ergebnisse der Unterabfrage entweder als temporäre Tabelle oder als Inline View zu behandeln. Dadurch lassen sich die "Kosten" der Unterabfrage erheblich reduzieren.
Im folgenden Beispiel werden zwei Tabellen miteinander verknüpft und der Wert SUM(sal) mehrfach berechnet.

Beispiel:

```
SQL> SELECT dname, SUM(sal) AS dept_total
  2    FROM emp, dept
  3    WHERE emp.deptno=dept.deptno
  4    GROUP BY dname
  5    HAVING SUM(sal)>
  6     (
  7       SELECT SUM(sal)*1/3
  8       FROM emp, dept
  9       WHERE emp.deptno=dept.deptno
 10     );

DNAME      DEPT_TOTAL
---------- ----------
RESEARCH        10875
```

Durch Verwendung der WITH-Klausel (subquery_factoring_clause), wird die mehrfache Berechnung von SUM(sal) vermieden und somit die Performance der Abfrage verbessert.

```
SQL> WITH
  2  summary AS
  3   (
  4     SELECT dname, SUM(sal) AS dept_total
  5       FROM emp, dept
  6       WHERE emp.deptno=dept.deptno
  7       GROUP BY dname
  8   )
  9  SELECT dname, dept_total
 10    FROM summary
 11    WHERE dept_total >
 12   (
  9     SELECT sum(dept_total)*1/3
 10      FROM summary
 11   );
```