Oracle SQL – Hierarchische Abfragen

Stephan Karrer

Hierarchische Abfragen: Oracle-spezifische Syntax

```
SELECT [LEVEL], column, expr ...
FROM   table
[WHERE condition(s)]
[START WITH condition(s)]
CONNECT BY PRIOR condition(s) ;
```

- START WITH:
  wo soll begonnen werden (Wurzel). Für „echte“ hierarchische Abfragen Pflicht

- CONNECT BY PRIOR:
  Angabe der Eltern-Kind-Beziehung, PRIOR steht vor dem Eltern-Attribut

Hierarchische Abfragen: Top Down

```
SELECT employee_id, last_name, manager_id
   FROM employees
   START WITH employee_id = 100
   CONNECT BY PRIOR employee_id = manager_id;
```

EMPLOYEE_ID	LAST_NAME	MANAGER_ID
101	Kochhar	100
108	Greenberg	101
109	Faviet	108
110	Chen	108
111	Sciarra	108
112	Urman	108
113	Popp	108
200	Whalen	101

...

Hierarchische Abfragen: Bottom Up

```
SELECT employee_id, last_name, job_id, manager_id
FROM   employees
CONNECT BY  employee_id = PRIOR manager_id
START  WITH  employee_id =  (SELECT employee_id
                             FROM   employees
                             WHERE  last_name = 'Kochhar') ;
```

- Hier ist die „manager_id“ übergeordnet, somit geht es „nach oben“
- PRIOR kann auch rechts stehen
- CONNECT BY kann auch vor START WITH geschrieben werden
- Bei START WITH und CONNECT BY können auch Unterabfragen verwendet werden

Hierarchische Abfragen: PRIOR-Operator

```
SELECT last_name, 'reports to', PRIOR last_name AS "Chief"
   FROM employees
   START WITH employee_id = 100
   CONNECT BY PRIOR employee_id = manager_id ;

SELECT last_name, 'is Manager of', PRIOR last_name AS "Clerk"
FROM   employees
CONNECT BY  employee_id = PRIOR manager_id
START  WITH  employee_id =  110 ;
```

- PRIOR ist unärer Operator, der in einer hierarchischen Abfrage auch ausserhalb von CONNECT BY verwendet werden kann

Hierarchische Abfragen mit Verwendung der LEVEL-Pseudospalte

```
SELECT employee_id, last_name, manager_id, LEVEL
  FROM employees
  START WITH EMPLOYEE_ID = 101
  CONNECT BY PRIOR employee_id = manager_id;
```

EMPLOYEE_ID	LAST_NAME	MANAGER_ID	LEVEL
101	Kochhar	100	1
108	Greenberg	101	2
109	Faviet	108	3
110	Chen	108	3
111	Sciarra	108	3
112	Urman	108	3
113	Popp	108	3

...

Pfadangabe mittels der Funktion SYS_CONNECT_BY_PATH

```
SELECT employee_id, last_name, manager_id,
       LEVEL, SYS_CONNECT_BY_PATH(last_name, '/') "Path"
   FROM employees
   START WITH employee_id = 100
   CONNECT BY PRIOR employee_id = manager_id;
```

```
EMPLOYEE_ID LAST_NAME MANAGER_ID LEVEL Path
----------- --------- ---------- ----- ------------------------------
        100 King                1     /King
        101 Kochhar   100         2     /King/Kochhar
        108 Greenberg 101         3     /King/Kochhar/Greenberg
        109 Faviet    108         4     /King/Kochhar/Greenberg/Faviet
        110 Chen      108         4     /King/Kochhar/Greenberg/Chen
        111 Sciarra   108         4     /King/Kochhar/Greenberg/Sciarra
```

Anzeige ob Blatt und der jeweiligen Wurzel

```
SELECT employee_id "empid", last_name, manager_id "manid",
       LEVEL, CONNECT_BY_ISLEAF "leaf",
       CONNECT_BY_ROOT last_name "Name",
       SYS_CONNECT_BY_PATH(last_name, '/') "Path"
   FROM employees
   START WITH employee_id = 100
   CONNECT BY PRIOR employee_id = manager_id ;
```

```
empid    LAST_NAME    manid    LEVEL    leaf     Name  Path
100      King                  1        0        King  /King
101      Kochhar      100      2        0        King  /King/Kochhar
108      Greenberg    101      3        0        King  /King/Kochhar/Greenberg
109      Faviet       108      4        1        King  /King/Kochhar/Greenberg/Faviet
110      Chen         108      4        1        King  /King/Kochhar/Greenberg/Chen
...
```

Sortieren: Verwendung des SIBLINGS-Schlüsselworts

```
SELECT last_name, employee_id, manager_id, LEVEL
  FROM employees
  START WITH employee_id = 100
  CONNECT BY PRIOR employee_id = manager_id
  ORDER SIBLINGS BY last_name;
```

LAST_NAME	EMPLOYEE_ID	MANAGER_ID	LEVEL
King	100		1
Cambrault	148	100	2
Bates	172	148	3
Bloom	169	148	3
Fox	170	148	3
Kumar	173	148	3
Ozer	168	148	3
Smith	171	148	3
De Haan	102	100	2

Verwendung der WHERE-Klausel

```
SELECT employee_id, last_name, manager_id,
       LEVEL, SYS_CONNECT_BY_PATH(last_name, '/') "Path"
   FROM employees
   WHERE last_name != 'Kochhar'
   START WITH employee_id = 100
   CONNECT BY PRIOR employee_id = manager_id;
```

```
100     King        null    1   /King
108     Greenberg   101     3   /King/Kochhar/Greenberg
109     Faviet      108     4   /King/Kochhar/Greenberg/Faviet
110     Chen        108     4   /King/Kochhar/Greenberg/Chen
111     Sciarra     108     4   /King/Kochhar/Greenberg/Sciarra
112     Urman       108     4   /King/Kochhar/Greenberg/Urman
113     Popp        108     4   /King/Kochhar/Greenberg/Popp
```

- Die WHERE-Klausel schränkt nur die Knoten ein, nicht den ganzen Zweig, d.h. die Rekursion findet trotzdem statt.

Ausblenden von Zweigen

```
SELECT employee_id, last_name, manager_id,
       LEVEL, SYS_CONNECT_BY_PATH(last_name, '/') "Path"
   FROM employees
   START WITH employee_id = 100
   CONNECT BY PRIOR employee_id = manager_id
               AND last_name != 'Kochhar' ;
```

```
100    King                1   /King
102    De Haan     100     2   /King/De Haan
103    Hunold      102     3   /King/De Haan/Hunold
104    Ernst       103     4   /King/De Haan/Hunold/Ernst
105    Austin      103     4   /King/De Haan/Hunold/Austin
106    Pataballa   103     4   /King/De Haan/Hunold/Pataballa
107    Lorentz     103     4   /King/De Haan/Hunold/Lorentz
114    Raphaely    100     2   /King/Raphaely
```

Hierarchische Abfragen : Zyklus wird entdeckt

```
UPDATE employees SET manager_id = 145
WHERE employee_id = 100;

SELECT employee_id, last_name, manager_id,
       LEVEL, SYS_CONNECT_BY_PATH(last_name, '/') "Path"
   FROM employees
   START WITH employee_id = 100
   CONNECT BY PRIOR employee_id = manager_id;
```

```
EMPLOYEE_ID LAST_NAME MANAGER_ID LEVEL Path
----------- --------- ---------- ----- ------------------------------
        100 King                    1     /King

        101 Kochhar   100           2     /King/Kochhar

 ...
```

SQL-Fehler: ORA-01436: CONNECT BY loop in user data
01436. 00000 - "CONNECT BY loop in user data"

Hierarchische Abfragen : Zyklus nicht folgen

```
UPDATE employees SET manager_id = 145
WHERE employee_id = 100;

SELECT last_name "Employee", CONNECT_BY_ISCYCLE "Cycle",
           LEVEL, SYS_CONNECT_BY_PATH(last_name, '/') "Path"
  FROM employees
  START WITH employee_id = 100
  CONNECT BY NOCYCLE PRIOR employee_id = manager_id
  ORDER BY "Cycle" DESC;
```

```
Employee             Cycle      LEVEL    Path
---------------      ---------- -----    --------------------
Russell              1          2        /King/Russell
King                 0          1        /King
Kochhar              0          2        /King/Kochhar
Greenberg            0          3        /King/Kochhar/Greenberg
...
```