**Hierarchical Queries in Oracle**

**Setup**

The following table contains hierarchical data.

CREATE TABLE tab1 (

id NUMBER,

parent\_id NUMBER,

CONSTRAINT tab1\_pk PRIMARY KEY (id),

CONSTRAINT tab1\_tab1\_fk FOREIGN KEY (parent\_id) REFERENCES tab1(id)

);

CREATE INDEX tab1\_parent\_id\_idx ON tab1(parent\_id);

INSERT INTO tab1 VALUES (1, NULL);

INSERT INTO tab1 VALUES (2, 1);

INSERT INTO tab1 VALUES (3, 2);

INSERT INTO tab1 VALUES (4, 2);

INSERT INTO tab1 VALUES (5, 4);

INSERT INTO tab1 VALUES (6, 4);

INSERT INTO tab1 VALUES (7, 1);

INSERT INTO tab1 VALUES (8, 7);

INSERT INTO tab1 VALUES (9, 1);

INSERT INTO tab1 VALUES (10, 9);

INSERT INTO tab1 VALUES (11, 10);

INSERT INTO tab1 VALUES (12, 9);

COMMIT;

## Basic Hierarchical Query

In its simplest form a hierarchical query needs a definition of how each child relates to its parent. This is defined using the **CONNECT BY .. PRIOR** clause, which defines how the current row (child) relates to a prior row (parent). In addition, the **START WITH** clause can be used to define the root node(s) of the hierarchy. Hierarchical queries come with operators, pseudocolumns and functions to help make sense of the hierarchy.

* **LEVEL** : The position in the hierarchy of the current row in relation to the root node.
* **CONNECT\_BY\_ROOT** : Returns the root node(s) associated with the current row.
* **SYS\_CONNECT\_BY\_PATH** : Returns a delimited breadcrumb from root to the current row.
* **CONNECT\_BY\_ISLEAF** : Indicates if the current row is a leaf node.
* **ORDER SIBLINGS BY** : Applies an order to siblings, without altering the basic hierarchical structure of the data returned by the query.

The following query gives an example of these items based on the previously defined test table.

SELECT id,

parent\_id,

RPAD('.', (level-1)\*2, '.') || id AS tree,

level,

CONNECT\_BY\_ROOT id AS root\_id,

LTRIM(SYS\_CONNECT\_BY\_PATH(id, '-'), '-') AS path,

CONNECT\_BY\_ISLEAF AS leaf

FROM tab1

START WITH parent\_id IS NULL

CONNECT BY parent\_id = PRIOR id

ORDER SIBLINGS BY id;

ID PARENT\_ID TREE LEVEL ROOT\_ID PATH LEAF

---------- ---------- -------------------- ---------- ---------- -------------------- ----------

1 1 1 1 1 0

2 1 ..2 2 1 1-2 0

3 2 ....3 3 1 1-2-3 1

4 2 ....4 3 1 1-2-4 0

5 4 ......5 4 1 1-2-4-5 1

6 4 ......6 4 1 1-2-4-6 1

7 1 ..7 2 1 1-7 0

8 7 ....8 3 1 1-7-8 1

9 1 ..9 2 1 1-9 0

10 9 ....10 3 1 1-9-10 0

11 10 ......11 4 1 1-9-10-11 1

12 9 ....12 3 1 1-9-12 1

## Cyclic Hierarchical Query

It is possible for a hierarchy to be cyclical, which can represent a problem when querying the data.

-- Create a cyclic reference

UPDATE tab1 SET parent\_id = 9 WHERE id = 1;

COMMIT;

SELECT id,

parent\_id,

RPAD('.', (level-1)\*2, '.') || id AS tree,

level,

CONNECT\_BY\_ROOT id AS root\_id,

LTRIM(SYS\_CONNECT\_BY\_PATH(id, '-'), '-') AS path,

CONNECT\_BY\_ISLEAF AS leaf

FROM tab1

START WITH parent\_id IS NULL

CONNECT BY parent\_id = PRIOR id

ORDER SIBLINGS BY id;

ERROR:

ORA-01436: CONNECT BY loop in user data

To simplify matters, the **CONNECT BY NOCYCLE** clause tells the database not to traverse cyclical hierarchies. In this case the **CONNECT\_BY\_ISCYCLE** function indicates which record is responsible for the cycle.

We can now use the **NOCYCLE** option and check the results of the **CONNECT\_BY\_ISCYCLE** function.

SELECT id,

parent\_id,

RPAD('.', (level-1)\*2, '.') || id AS tree,

level,

CONNECT\_BY\_ROOT id AS root\_id,

LTRIM(SYS\_CONNECT\_BY\_PATH(id, '-'), '-') AS path,

CONNECT\_BY\_ISLEAF AS leaf,

CONNECT\_BY\_ISCYCLE AS cycle

FROM tab1

START WITH id = 1

CONNECT BY NOCYCLE parent\_id = PRIOR id

ORDER SIBLINGS BY id;

ID PARENT\_ID TREE LEVEL ROOT\_ID PATH LEAF CYCLE

---------- ---------- -------------------- ---------- ---------- -------------------- ---------- ----------

1 9 1 1 1 1 0 0

2 1 ..2 2 1 1-2 0 0

3 2 ....3 3 1 1-2-3 1 0

4 2 ....4 3 1 1-2-4 0 0

5 4 ......5 4 1 1-2-4-5 1 0

6 4 ......6 4 1 1-2-4-6 1 0

7 1 ..7 2 1 1-7 0 0

8 7 ....8 3 1 1-7-8 1 0

9 1 ..9 2 1 1-9 0 1

10 9 ....10 3 1 1-9-10 0 0

11 10 ......11 4 1 1-9-10-11 1 0

12 9 ....12 3 1 1-9-12 1 0