Hierarchical Queries

If a table contains hierarchical data, then you can select rows in a hierarchical order using the hierarchical query clause:

***hierarchical\_query\_clause*::=**

  
[Description of the illustration hierarchical\_query\_clause.gif](https://docs.oracle.com/cd/B19306_01/server.102/b14200/img_text/hierarchical_query_clause.htm)

START WITH specifies the root row(s) of the hierarchy.

CONNECT BY specifies the relationship between parent rows and child rows of the hierarchy.

* The NOCYCLE parameter instructs Oracle Database to return rows from a query even if a CONNECT BY LOOP exists in the data. Use this parameter along with the CONNECT\_BY\_ISCYCLE pseudocolumn to see which rows contain the loop. Please refer to [CONNECT\_BY\_ISCYCLE Pseudocolumn](https://docs.oracle.com/cd/B19306_01/server.102/b14200/pseudocolumns001.htm#i1009434) for more information.
* In a hierarchical query, one expression in *condition* must be qualified with the PRIOR operator to refer to the parent row. For example,
* ... PRIOR expr = expr
* or
* ... expr = PRIOR expr

If the CONNECT BY *condition* is compound, then only one condition requires the PRIOR operator, although you can have multiple PRIOR conditions. For example:

CONNECT BY last\_name != 'King' AND PRIOR employee\_id = manager\_id ...

CONNECT BY PRIOR employee\_id = manager\_id and

PRIOR account\_mgr\_id = customer\_id ...

PRIOR is a unary operator and has the same precedence as the unary + and - arithmetic operators. It evaluates the immediately following expression for the parent row of the current row in a hierarchical query.

PRIOR is most commonly used when comparing column values with the equality operator. (The PRIOR keyword can be on either side of the operator.) PRIOR causes Oracle to use the value of the parent row in the column. Operators other than the equal sign (=) are theoretically possible in CONNECT BY clauses. However, the conditions created by these other operators can result in an infinite loop through the possible combinations. In this case Oracle detects the loop at run time and returns an error.

Both the CONNECT BY condition and the PRIOR expression can take the form of an uncorrelated subquery. However, the PRIOR expression cannot refer to a sequence. That is, CURRVAL and NEXTVAL are not valid PRIOR expressions.

You can further refine a hierarchical query by using the CONNECT\_BY\_ROOT operator to qualify a column in the select list. This operator extends the functionality of the CONNECT BY [PRIOR] condition of hierarchical queries by returning not only the immediate parent row but all ancestor rows in the hierarchy.

**See Also:**

[CONNECT\_BY\_ROOT](https://docs.oracle.com/cd/B19306_01/server.102/b14200/operators004.htm#i1035022) for more information about this operator and ["Hierarchical Query Examples"](https://docs.oracle.com/cd/B19306_01/server.102/b14200/queries003.htm#i2060615)

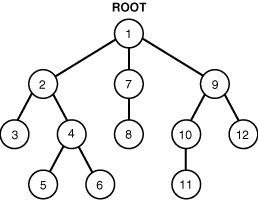
Oracle processes hierarchical queries as follows:

* A join, if present, is evaluated first, whether the join is specified in the FROM clause or with WHERE clause predicates.
* The CONNECT BY condition is evaluated.
* Any remaining WHERE clause predicates are evaluated.

Oracle then uses the information from these evaluations to form the hierarchy using the following steps:

1. Oracle selects the root row(s) of the hierarchy--those rows that satisfy the START WITH condition.
2. Oracle selects the child rows of each root row. Each child row must satisfy the condition of the CONNECT BY condition with respect to one of the root rows.
3. Oracle selects successive generations of child rows. Oracle first selects the children of the rows returned in step [2](https://docs.oracle.com/cd/B19306_01/server.102/b14200/queries003.htm#i2070828), and then the children of those children, and so on. Oracle always selects children by evaluating the CONNECT BY condition with respect to a current parent row.
4. If the query contains a WHERE clause without a join, then Oracle eliminates all rows from the hierarchy that do not satisfy the condition of the WHERE clause. Oracle evaluates this condition for each row individually, rather than removing all the children of a row that does not satisfy the condition.
5. Oracle returns the rows in the order shown in [Figure 9-1](https://docs.oracle.com/cd/B19306_01/server.102/b14200/queries003.htm#i2066595). In the diagram, children appear below their parents. For an explanation of hierarchical trees, see [Figure 3-1, "Hierarchical Tree"](https://docs.oracle.com/cd/B19306_01/server.102/b14200/pseudocolumns001.htm#i1009270).

***Figure 9-1 Hierarchical Queries***

  
[Description of "Figure 9-1 Hierarchical Queries"](https://docs.oracle.com/cd/B19306_01/server.102/b14200/img_text/sqlrf002.htm)

To find the children of a parent row, Oracle evaluates the PRIOR expression of the CONNECT BY condition for the parent row and the other expression for each row in the table. Rows for which the condition is true are the children of the parent. The CONNECT BY condition can contain other conditions to further filter the rows selected by the query. The CONNECT BY condition cannot contain a subquery.

If the CONNECT BY condition results in a loop in the hierarchy, then Oracle returns an error. A loop occurs if one row is both the parent (or grandparent or direct ancestor) and a child (or a grandchild or a direct descendent) of another row.

In a hierarchical query, do not specify either ORDER BY or GROUP BY, as they will destroy the hierarchical order of the CONNECT BY results. If you want to order rows of siblings of the same parent, then use the ORDER SIBLINGS BY clause. See *[order\_by\_clause](https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_10002.htm" \l "i2171079)* .

Hierarchical Query Examples

**CONNECT BY Example**The following hierarchical query uses the CONNECT BY clause to define the relationship between employees and managers:

SELECT employee\_id, last\_name, manager\_id

FROM employees

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

...

**LEVEL Example**The next example is similar to the preceding example, but uses the LEVEL pseudocolumn to show parent and child rows:

SELECT employee\_id, last\_name, manager\_id, LEVEL

FROM employees

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

...

**START WITH Examples**The next example adds a START WITH clause to specify a root row for the hierarchy and an ORDER BY clause using the SIBLINGS keyword to preserve ordering within the hierarchy:

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

Hunold 103 102 3

Austin 105 103 4

Ernst 104 103 4

Lorentz 107 103 4

Pataballa 106 103 4

Errazuriz 147 100 2

Ande 166 147 3

Banda 167 147 3

...

In the hr.employees table, the employee Steven King is the head of the company and has no manager. Among his employees is John Russell, who is the manager of department 80. If we update the employees table to set Russell as King's manager, we will create a loop in the data:

UPDATE employees SET manager\_id = 145

WHERE employee\_id = 100;

SELECT last\_name "Employee",

LEVEL, SYS\_CONNECT\_BY\_PATH(last\_name, '/') "Path"

FROM employees

WHERE level <= 3 AND department\_id = 80

START WITH last\_name = 'King'

CONNECT BY PRIOR employee\_id = manager\_id AND LEVEL <= 4;

2 3 4 5 6 7 ERROR:

ORA-01436: CONNECT BY loop in user data

The NOCYCLE parameter in the CONNECT BY condition causes Oracle to return the rows in spite of the loop. The CONNECT\_BY\_ISCYCLE pseudocolumn shows you which rows contain the cycle:

SELECT last\_name "Employee", CONNECT\_BY\_ISCYCLE "Cycle",

LEVEL, SYS\_CONNECT\_BY\_PATH(last\_name, '/') "Path"

FROM employees

WHERE level <= 3 AND department\_id = 80

START WITH last\_name = 'King'

CONNECT BY NOCYCLE PRIOR employee\_id = manager\_id AND LEVEL <= 4;

Employee Cycle LEVEL Path

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

Russell 1 2 /King/Russell

Tucker 0 3 /King/Russell/Tucker

Bernstein 0 3 /King/Russell/Bernstein

Hall 0 3 /King/Russell/Hall

Olsen 0 3 /King/Russell/Olsen

Cambrault 0 3 /King/Russell/Cambrault

Tuvault 0 3 /King/Russell/Tuvault

Partners 0 2 /King/Partners

King 0 3 /King/Partners/King

Sully 0 3 /King/Partners/Sully

McEwen 0 3 /King/Partners/McEwen

...

**CONNECT\_BY\_ROOT Examples**The following example returns the last name of each employee in department 110, each manager above that employee in the hierarchy, the number of levels between manager and employee, and the path between the two:

SELECT last\_name "Employee", CONNECT\_BY\_ROOT last\_name "Manager",

LEVEL-1 "Pathlen", SYS\_CONNECT\_BY\_PATH(last\_name, '/') "Path"

FROM employees

WHERE LEVEL > 1 and department\_id = 110

CONNECT BY PRIOR employee\_id = manager\_id;

Employee Manager Pathlen Path

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

Higgins Kochhar 1 /Kochhar/Higgins

Gietz Kochhar 2 /Kochhar/Higgins/Gietz

Gietz Higgins 1 /Higgins/Gietz

Higgins King 2 /King/Kochhar/Higgins

Gietz King 3 /King/Kochhar/Higgins/Gietz

The following example uses a GROUP BY clause to return the total salary of each employee in department 110 and all employees below that employee in the hierarchy:

SELECT name, SUM(salary) "Total\_Salary" FROM (

SELECT CONNECT\_BY\_ROOT last\_name as name, Salary

FROM employees

WHERE department\_id = 110

CONNECT BY PRIOR employee\_id = manager\_id)

GROUP BY name;

NAME Total\_Salary

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

Gietz 8300

Higgins 20300

King 20300

Kochhar 20300