

Hierarchical Retrieval

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this appendix, you should be able to:

- Interpret the concept of a hierarchical query
- Create a tree-structured report
- Format hierarchical data
- Exclude branches from the tree structure

ORACLE

In this appendix, you learn how to use hierarchical queries to create tree-structured reports.

Sample Data from the EMPLOYEES Table

	EMPLOYEE_ID	LAST_NAME	JOB_ID	MANAGER_ID
1	100	King	AD_PRES	(null)
2	101	Kochhar	AD_VP	100
3	102	De Haan	AD_VP	100
4	103	Hunold	IT_PROG	102
5	104	Ernst	IT_PROG	103
6	107	Lorentz	IT_PROG	103

...

16	200	VWhalen	AD_ASST	101
17	201	Hartstein	MK_MAN	100
18	202	Fay	MK_REP	201
19	205	Higgins	AC_MGR	101
20	206	Gietz	AC_ACCOUNT	205

ORACLE

F - 3

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

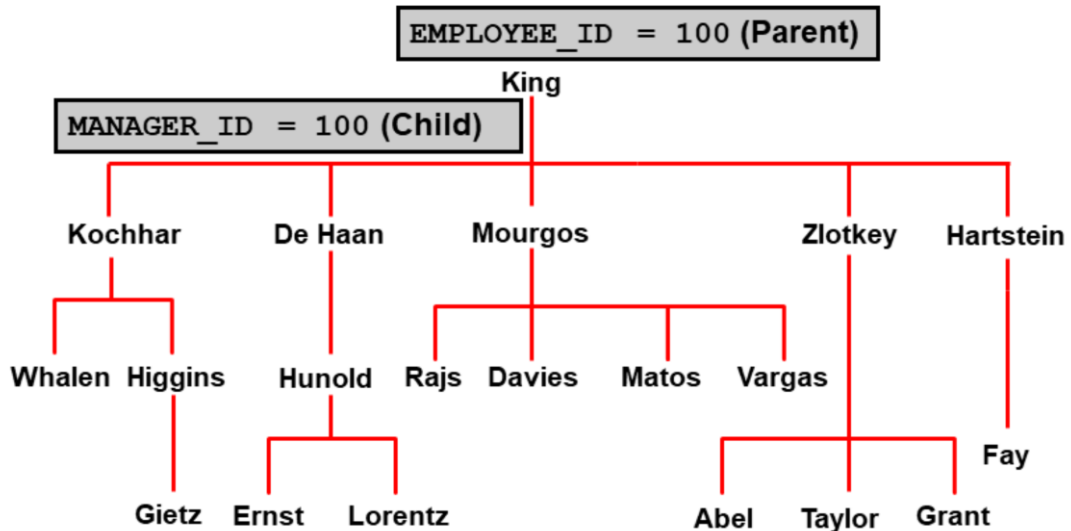
Using hierarchical queries, you can retrieve data based on a natural hierarchical relationship between the rows in a table. A relational database does not store records in a hierarchical way. However, where a hierarchical relationship exists between the rows of a single table, a process called *tree walking* enables the hierarchy to be constructed. A hierarchical query is a method of reporting, with the branches of a tree in a specific order.

Imagine a family tree with the eldest members of the family found close to the base or trunk of the tree and the youngest members representing branches of the tree. Branches can have their own branches, and so on.

A hierarchical query is possible when a relationship exists between rows in a table. For example, in the slide, you see that Kochhar, De Haan, and Hartstein report to `MANAGER_ID` 100, which is King's `EMPLOYEE_ID`.

Note: Hierarchical trees are used in various fields such as human genealogy (family trees), livestock (breeding purposes), corporate management (management hierarchies), manufacturing (product assembly), evolutionary research (species development), and scientific research.

Natural Tree Structure



ORACLE

F - 4

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The `EMPLOYEES` table has a tree structure representing the management reporting line. The hierarchy can be created by looking at the relationship between equivalent values in the `EMPLOYEE_ID` and `MANAGER_ID` columns. This relationship can be exploited by joining the table to itself. The `MANAGER_ID` column contains the employee number of the employee's manager.

The parent/child relationship of a tree structure enables you to control:

- The direction in which the hierarchy is walked
- The starting point inside the hierarchy

Note: The slide displays an inverted tree structure of the management hierarchy of the employees in the `EMPLOYEES` table.

Hierarchical Queries

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

condition:

```
expr comparison_operator expr
```

ORACLE

F - 5

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Keywords and Clauses

Hierarchical queries can be identified by the presence of the `CONNECT BY` and `START WITH` clauses.

In the syntax:

<code>SELECT</code>	Is the standard <code>SELECT</code> clause
<code>LEVEL</code>	For each row returned by a hierarchical query, the <code>LEVEL</code> pseudocolumn returns 1 for a root row, 2 for a child of a root, and so on.
<code>FROM table</code>	Specifies the table, view, or snapshot containing the columns. You can select from only one table.
<code>WHERE</code>	Restricts the rows returned by the query without affecting other rows of the hierarchy
<i>condition</i>	Is a comparison with expressions
<code>START WITH</code>	Specifies the root rows of the hierarchy (where to start). This clause is required for a true hierarchical query.
<code>CONNECT BY</code> <code>PRIOR</code>	Specifies the columns in which the relationship between parent and child <code>PRIOR</code> rows exist. This clause is required for a hierarchical query.

Walking the Tree

Starting Point

- Specifies the condition that must be met
- Accepts any valid condition

```
START WITH column1 = value
```

Using the `EMPLOYEES` table, start with the employee whose last name is Kochhar.

```
...START WITH last_name = 'Kochhar'
```

ORACLE

F - 6

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The row or rows to be used as the root of the tree are determined by the `START WITH` clause. The `START WITH` clause can contain any valid condition.

Examples

Using the `EMPLOYEES` table, start with King, the president of the company.

```
... START WITH manager_id IS NULL
```

Using the `EMPLOYEES` table, start with employee Kochhar. A `START WITH` condition can contain a subquery.

```
... START WITH employee_id = (SELECT employee_id
                               FROM   employees
                               WHERE  last_name = 'Kochhar')
```

If the `START WITH` clause is omitted, the tree walk is started with all the rows in the table as root rows.

Note: The `CONNECT BY` and `START WITH` clauses are not American National Standards Institute (ANSI) SQL standard.

Walking the Tree

```
CONNECT BY PRIOR column1 = column2
```

Walk from the top down, using the `EMPLOYEES` table.

```
... CONNECT BY PRIOR employee_id = manager_id
```

Direction

Top down	→	Column1 = Parent Key Column2 = Child Key
Bottom up	→	Column1 = Child Key Column2 = Parent Key

ORACLE

F - 7

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The direction of the query is determined by the `CONNECT BY PRIOR` column placement. For top-down, the `PRIOR` operator refers to the parent row. For bottom-up, the `PRIOR` operator refers to the child row. To find the child rows of a parent row, the Oracle server evaluates the `PRIOR` expression for the parent row and the other expressions for each row in the table. Rows for which the condition is true are the child rows of the parent. The Oracle server always selects child rows by evaluating the `CONNECT BY` condition with respect to a current parent row.

Examples

Walk from the top down using the `EMPLOYEES` table. Define a hierarchical relationship in which the `EMPLOYEE_ID` value of the parent row is equal to the `MANAGER_ID` value of the child row:

```
... CONNECT BY PRIOR employee_id = manager_id
```

Walk from the bottom up using the `EMPLOYEES` table:

```
... CONNECT BY PRIOR manager_id = employee_id
```

The `PRIOR` operator does not necessarily need to be coded immediately following `CONNECT BY`. Thus, the following `CONNECT BY PRIOR` clause gives the same result as the one in the preceding example:

```
... CONNECT BY employee_id = PRIOR manager_id
```

Note: The `CONNECT BY` clause cannot contain a subquery.

Walking the Tree: From the Bottom Up

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

	EMPLOYEE_ID	LAST_NAME	JOB_ID	MANAGER_ID
1	101	Kochhar	AD_VP	100
2	100	King	AD_PRES	(null)

ORACLE

The example in the slide displays a list of managers starting with the employee whose employee ID is 101.

Walking the Tree: From the Top Down

```
SELECT last_name||' reports to '||  
PRIOR last_name "Walk Top Down"  
FROM employees  
START WITH last_name = 'King'  
CONNECT BY PRIOR employee_id = manager_id ;
```

Walk Top Down
1 King reports to
2 King reports to
3 Kochhar reports to King
4 Greenberg reports to Kochhar
5 Faviet reports to Greenberg

...

105 Grant reports to Zlotkey
106 Johnson reports to Zlotkey
107 Hartstein reports to King
108 Fay reports to Hartstein

ORACLE

F - 9

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Walking from the top down, display the names of the employees and their manager. Use employee King as the starting point. Print only one column.

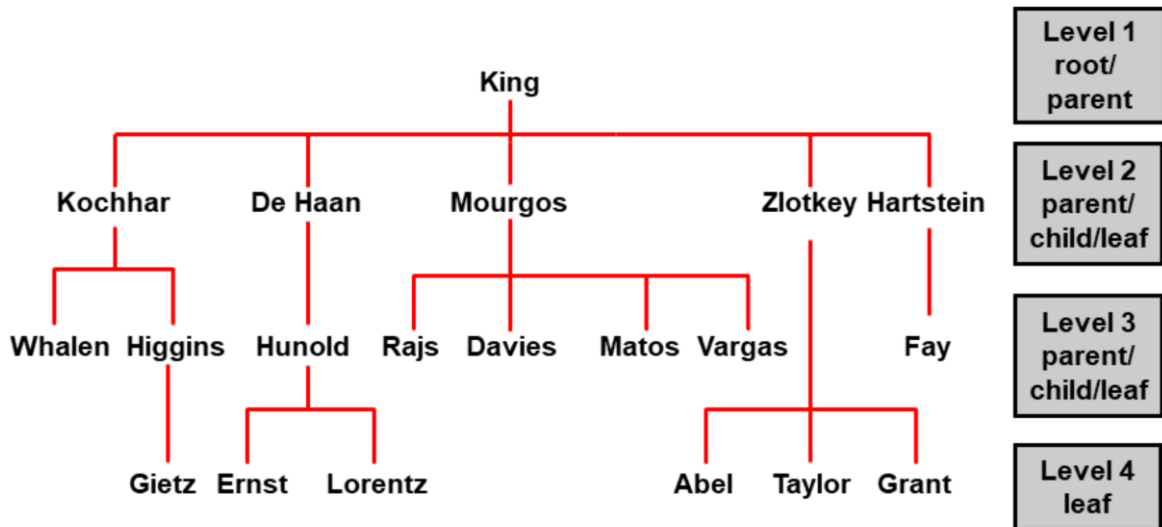
Example

In the following example, `EMPLOYEE_ID` values are evaluated for the parent row and `MANAGER_ID` and `SALARY` values are evaluated for the child rows. The `PRIOR` operator applies only to the `EMPLOYEE_ID` value.

```
... CONNECT BY PRIOR employee_id = manager_id  
AND salary > 15000;
```

To qualify as a child row, a row must have a `MANAGER_ID` value equal to the `EMPLOYEE_ID` value of the parent row and must have a `SALARY` value greater than \$15,000.

Ranking Rows with the LEVEL Pseudocolumn



ORACLE

F - 10

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can explicitly show the rank or level of a row in the hierarchy by using the `LEVEL` pseudocolumn. This will make your report more readable. The forks where one or more branches split away from a larger branch are called nodes, and the very end of a branch is called a leaf or leaf node. The graphic in the slide shows the nodes of the inverted tree with their `LEVEL` values. For example, employee Higgins is a parent and a child, whereas employee Davies is a child and a leaf.

LEVEL Pseudocolumn

Value	Level for Top Down	Level for Bottom up
1	A root node	A root node
2	A child of a root node	The parent of a root node
3	A child of a child, and so on	A parent of a parent, and so on

In the slide, King is the root or parent (`LEVEL = 1`). Kochhar, De Haan, Mourgos, Zlotkey, Hartstein, Higgins, and Hunold are children and also parents (`LEVEL = 2`). Whalen, Rajs, Davies, Matos, Vargas, Gietz, Ernst, Lorentz, Abel, Taylor, Grant, and Fay are children and leaves (`LEVEL = 3` and `LEVEL = 4`).

Note: A *root node* is the highest node within an inverted tree. A *child node* is any nonroot node. A *parent node* is any node that has children. A *leaf node* is any node without children.

The number of levels returned by a hierarchical query may be limited by available user memory.

Formatting Hierarchical Reports Using LEVEL and LPAD

Create a report displaying company management levels, beginning with the highest level and indenting each of the following levels.

```
COLUMN org_chart FORMAT A12
SELECT LPAD(last_name, LENGTH(last_name)+(LEVEL*2)-2, '_')
       AS org_chart
FROM   employees
START WITH first_name='Steven' AND last_name='King'
CONNECT BY PRIOR employee_id=manager_id
```

ORACLE

F - 11

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The nodes in a tree are assigned level numbers from the root. Use the `LPAD` function in conjunction with the `LEVEL` pseudocolumn to display a hierarchical report as an indented tree.

In the example in the slide:

- `LPAD(char1, n [, char2])` returns *char1*, left-padded to length *n* with the sequence of characters in *char2*. The argument *n* is the total length of the return value as it is displayed on your terminal screen.
- `LPAD(last_name, LENGTH(last_name)+(LEVEL*2)-2, '_')` defines the display format
- *char1* is the `LAST_NAME`, *n* the total length of the return value, is length of the `LAST_NAME + (LEVEL*2) - 2`, and *char2* is `'_'`

That is, this tells SQL to take the `LAST_NAME` and left-pad it with the `'_'` character until the length of the resultant string is equal to the value determined by `LENGTH(last_name) + (LEVEL*2) - 2`.

For King, `LEVEL = 1`. Therefore, $(2 * 1) - 2 = 2 - 2 = 0$. So King does not get padded with any `'_'` character and is displayed in column 1.

For Kochhar, `LEVEL = 2`. Therefore, $(2 * 2) - 2 = 4 - 2 = 2$. So Kochhar gets padded with 2 ' ' characters and is displayed indented.

The rest of the records in the `EMPLOYEES` table are displayed similarly.

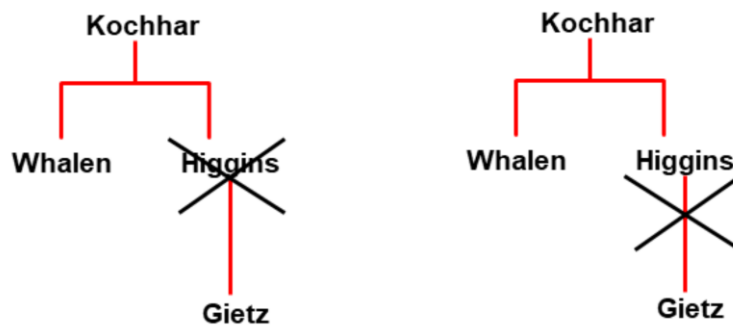
1	2	ORG_CHART
1	King	
2	__Kochhar	
3	___Greenberg	
4	____Faviet	
5	____Chen	
6	____Sciarra	
7	____Urman	
8	____Popp	
9	___Whalen	
10	___Mavris	
11	___Baer	
12	___Higgins	
13	____Gietz	
14	__De Haan	
15	___Hunold	
16	____Ernst	
17	____Austin	

Pruning Branches

Use the `WHERE` clause
to eliminate a node.

Use the `CONNECT BY` clause
to eliminate a branch.

```
WHERE last_name != 'Higgins' CONNECT BY PRIOR  
employee_id = manager_id  
AND last_name != 'Higgins'
```



ORACLE

F - 13

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can use the `WHERE` and `CONNECT BY` clauses to prune the tree (that is, to control which nodes or rows are displayed). The predicate you use acts as a Boolean condition.

Examples

Starting at the root, walk from the top down, and eliminate employee Higgins in the result, but process the child rows.

```
SELECT department_id, employee_id, last_name, job_id, salary
FROM employees
WHERE last_name != 'Higgins'
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id;
```

Starting at the root, walk from the top down, and eliminate employee Higgins and all child rows.

```
SELECT department_id, employee_id, last_name, job_id, salary
FROM employees
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id
```

```
AND      last_name != 'Higgins';
```


Summary

In this appendix, you should have learned how to:

- Use hierarchical queries to view a hierarchical relationship between rows in a table
- Specify the direction and starting point of the query
- Eliminate nodes or branches by pruning

ORACLE

You can use hierarchical queries to retrieve data based on a natural hierarchical relationship between rows in a table. The `LEVEL` pseudocolumn counts how far down a hierarchical tree you have traveled. You can specify the direction of the query using the `CONNECT BY PRIOR` clause. You can specify the starting point using the `START WITH` clause. You can use the `WHERE` and `CONNECT BY` clauses to prune the tree branches.