

Hierarchical Retrieval

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

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

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

In this lesson, 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	105	Austin	IT_PROG	103
7	106	Pataballa	IT_PROG	103
8	107	Lorentz	IT_PROG	103
9	108	Greenberg	FL_MGR	101
10	109	Faviet	FL_ACCOUNT	108
...				
101	200	Whalen	AD_ASST	101
102	201	Hartstein	MK_MAN	100
103	202	Fay	MK_REP	201
104	203	Mavris	HR_REP	101
105	204	Baer	PR_REP	101
106	205	Higgins	AC_MGR	101
107	206	Gietz	AC_ACCOUNT	205

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Sample Data from the EMPLOYEES Table

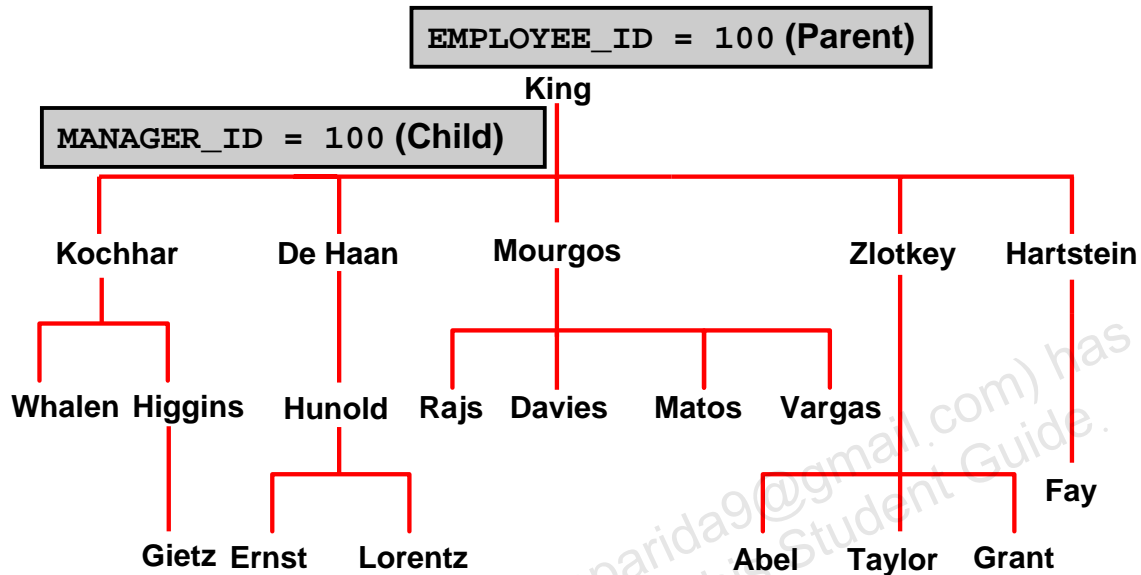
Using hierarchical queries, you can retrieve data based on a natural hierarchical relationship between rows in a table. A relational database does not store records in a hierarchical way. However, where a hierarchical relationship exists between the rows of a single table, a process called *tree walking* 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 employees with the job IDs of AD_VP, ST_MAN, SA_MAN, and MK_MAN report directly to the president of the company. You know this because the MANAGER_ID column of these records contains the employee ID 100, which belongs to the president (AD_PRES).

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

Copyright © 2009, Oracle. All rights reserved.

Natural Tree Structure

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)] ;
```

WHERE *condition*:

```
expr comparison_operator expr
```

ORACLE

Copyright © 2009, Oracle. 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:

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

The SELECT statement cannot contain a join or query from a view that contains a join.

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

Copyright © 2009, Oracle. All rights reserved.

Walking the Tree

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 be used in conjunction with 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. If a WHERE clause is used, the walk is started with all the rows that satisfy the WHERE condition. This no longer reflects a true hierarchy.

Note: The CONNECT BY PRIOR and START WITH clauses are not 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

Copyright © 2009, Oracle. All rights reserved.

Walking the Tree (continued)

The direction of the query, whether it is from parent to child or from child to parent, is determined by the CONNECT BY PRIOR column placement. The PRIOR operator refers to the parent 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

Copyright © 2009, Oracle. All rights reserved.

Walking the Tree: From the Bottom Up

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

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.

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
6	Chen reports to Greenberg
7	Sciarra reports to Greenberg
8	Urman reports to Greenberg

...

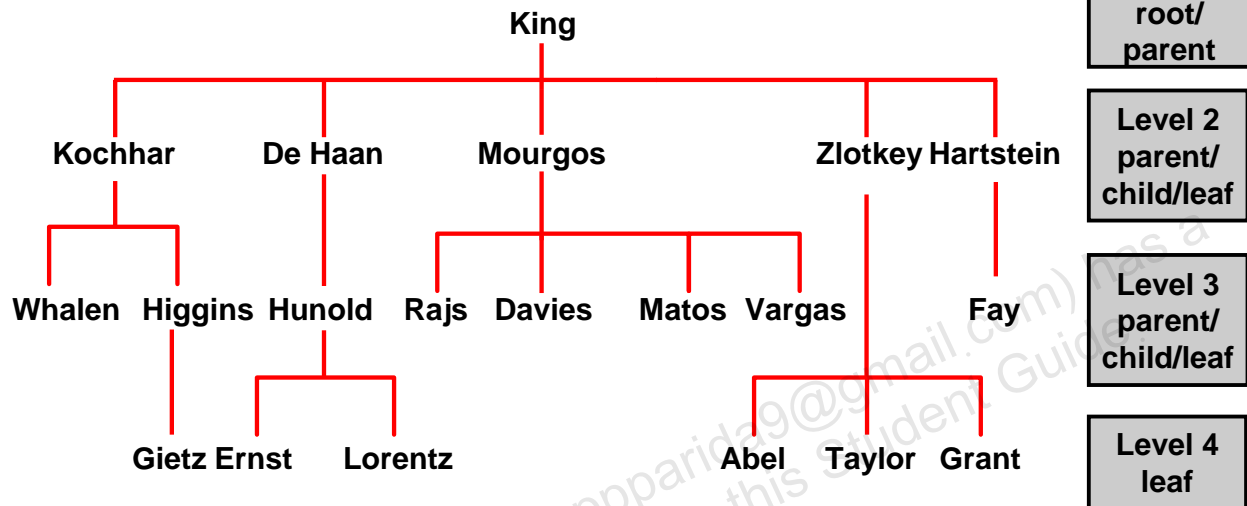
ORACLE

Copyright © 2009, Oracle. All rights reserved.

Walking the Tree: From the Top Down

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.

Ranking Rows with the LEVEL Pseudocolumn



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Ranking Rows with the LEVEL Pseudocolumn

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 diagram 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
1	A root node
2	A child of a root node
3	A child of a child, 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

Copyright © 2009, Oracle. All rights reserved.

Formatting Hierarchical Reports Using LEVEL and LPAD

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.

Formatting Hierarchical Reports Using LEVEL and LPAD (continued)

	AZ	ORG_CHART
1		King
2		__Kochhar
3		___Greenberg
4		____Faviet
5		_____Chen
6		____Sciarr
7		_____Urman
8		_____Popp
9		____Whalen
10		____Mavris

...

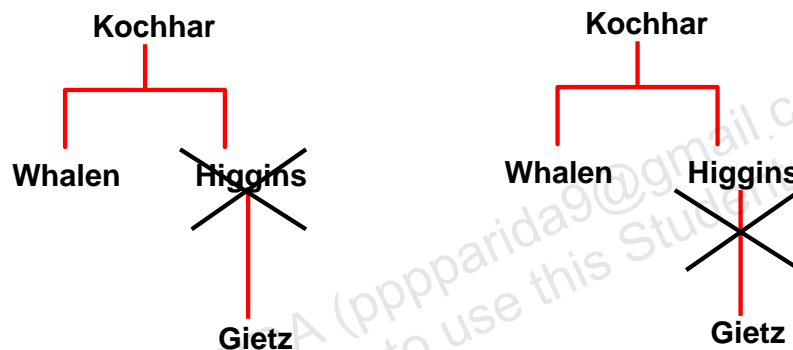
100		____Abel
101		____Hutton
102		____Taylor
103		____Livingston
104		____Grant
105		____Johnson
106		____Hartstein
107		____Fay

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

Copyright © 2009, Oracle. All rights reserved.

Pruning Branches

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 lesson, you should have learned that:

- You can use hierarchical queries to view a hierarchical relationship between rows in a table
- You specify the direction and starting point of the query
- You can eliminate nodes or branches by pruning

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Summary

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.

Practice 7: Overview

This practice covers the following topics:

- Distinguishing hierarchical queries from nonhierarchical queries
- Walking through a tree
- Producing an indented report by using the `LEVEL` pseudocolumn
- Pruning the tree structure
- Sorting the output

The Oracle logo, consisting of the word "ORACLE" in a bold, sans-serif font, is positioned on the right side of a red horizontal bar.

Copyright © 2009, Oracle. All rights reserved.

Practice 7: Overview

In this practice, you gain practical experience in producing hierarchical reports.

Note: Question 1 is a paper-based question.

Practice 7

1. Look at the following output examples. Are they the result of a hierarchical query? Explain why or why not.

Exhibit 1:

	EMPLOYEE_ID	LAST_NAME	MANAGER_ID	SALARY	DEPARTMENT_ID
1	100	King	(null)	24000	90
2	101	Kochhar	100	17000	90
3	102	De Haan	100	17000	90
4	145	Russell	100	14000	80
5	146	Partners	100	13500	80
6	201	Hartstein	100	13000	20
7	205	Higgins	101	12000	110
8	108	Greenberg	101	12000	100
9	147	Errazuriz	100	12000	80
10	168	Ozer	148	11500	80
11	148	Cambrault	100	11000	80
12	174	Abel	149	11000	80
13	114	Raphaely	100	11000	30
14	162	Vishney	147	10500	80
15	149	Zlotkey	100	10500	80

■ ■ ■

Exhibit 2:




	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	100	King	90	Executive
2	101	Kochhar	90	Executive
3	102	De Haan	90	Executive
4	103	Hunold	60	IT
5	104	Ernst	60	IT
6	105	Austin	60	IT
7	106	Pataballa	60	IT
8	107	Lorentz	60	IT
9	108	Greenberg	100	Finance
10	109	Faviet	100	Finance

■ ■ ■

Practice 7 (continued)**Exhibit 3:**

		RANK		LAST_NAME
1		1		Ernst
2		2		Hunold
3		3		De Haan
4		4		King

2. Produce a report that shows an organization chart for Mourgos's department. Print the last names, salaries, and department IDs.

		LAST_NAME		SALARY		DEPARTMENT_ID
1		Mourgos		5800		50
2		Rajs		3500		50
3		Davies		3100		50
4		Matos		2600		50
5		Vargas		2500		50
6		Walsh		3100		50
7		Feeney		3000		50
8		OConnell		2600		50
9		Grant		2600		50

3. Create a report that shows the hierarchy of managers for the employee Lorentz. Display his immediate manager first.

		LAST_NAME
1		Hunold
2		De Haan
3		King

Practice 7 (continued)

4. Create an indented report that shows the management hierarchy starting from the employee whose LAST_NAME is Kochhar. Print the employee's last name, manager ID, and department ID. Give alias names to the columns as shown in the sample output.

	R Z NAME	R Z MGR	R Z DEPTNO
1	Kochhar	100	90
2	__Greenberg	101	100
3	___Faviet	108	100
4	___Chen	108	100
5	___Sciarra	108	100
6	___Urman	108	100
7	___Popp	108	100
8	__Whalen	101	10
9	__Mavris	101	40
10	__Baer	101	70
11	__Higgins	101	110
12	___Gietz	205	110

If you have time, complete the following exercise:

5. Produce a company organization chart that shows the management hierarchy. Start with the person at the top level, exclude all those with a job ID of IT_PROG, and exclude De Haan and those employees who report to De Haan.

	R Z LAST_NAME	R Z EMPLOYEE_ID	R Z MANAGER_ID
1	King	100	(null)
2	Kochhar	101	100
3	Greenberg	108	101
4	Faviet	109	108
5	Chen	110	108
6	Sciarra	111	108
7	Urman	112	108
8	Popp	113	108

...

93	Zlotkey	149	100
94	Abel	174	149
95	Hutton	175	149
96	Taylor	176	149
97	Livingston	177	149
98	Grant	178	149
99	Johnson	179	149
100	Hartstein	201	100
101	Fay	202	201