G

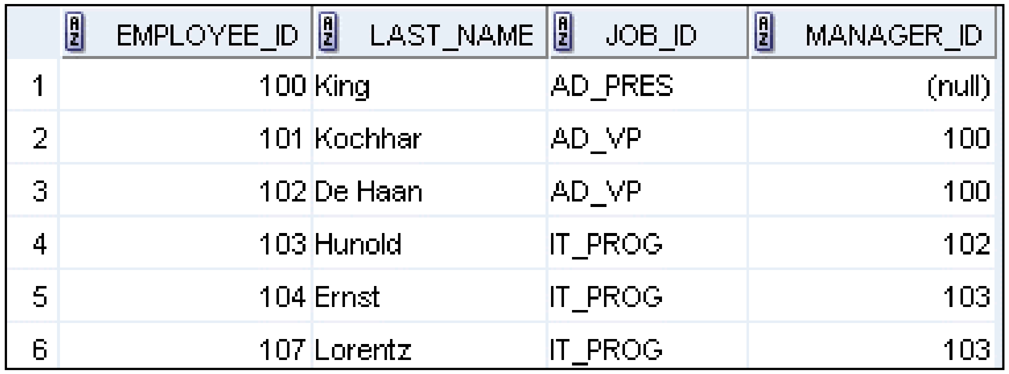
Hierarchical Retrieval

Objectives

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

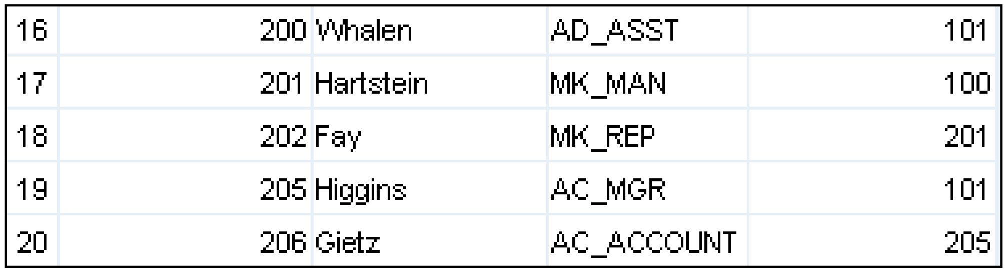
- Interpret the concept of a hierarchical query
- Create a tree-structured report
- Format hierarchical data
- Exclude branches from the tree structure

Sample Data from the `EMPLOYEES` Table



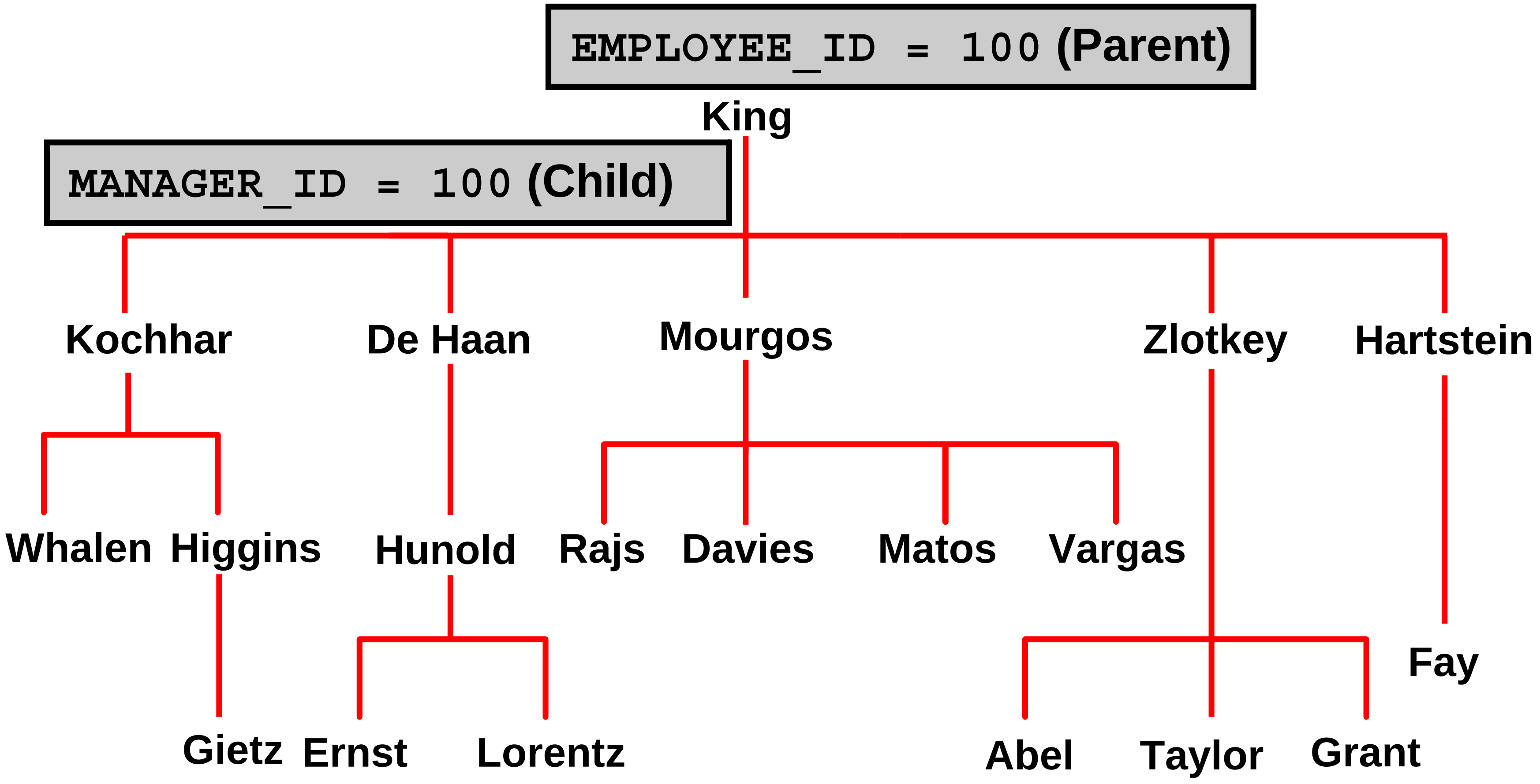
	EMPLOYEE_ID	LAST_NAME	JOB_ID	MANAGER_ID
1	100	King	AD_PRES	(null)
2	101	Kochhar	AD_VP	100
3	102	De Haan	AD_VP	100
4	103	Hunold	IT_PROG	102
5	104	Ernst	IT_PROG	103
6	107	Lorentz	IT_PROG	103

…



16	200	Whalen	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

Natural Tree Structure



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
```

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'
```

Walking the Tree

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

Walk from the top down, using the EMPLOYEES table.

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

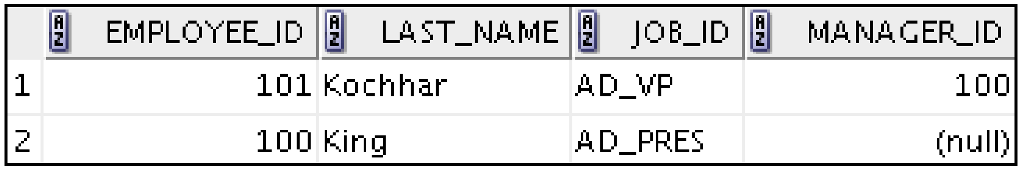
Direction

$$\text{Top down} \longrightarrow \begin{array}{l}\text{Column1 = Parent Key}\\\text{Column2 = Child Key}\end{array}$$

$$\text{Bottom up} \longrightarrow \begin{array}{l}\text{Column1 = Child Key}\\\text{Column2 = Parent Key}\end{array}$$

Walking the Tree: From the Bottom Up

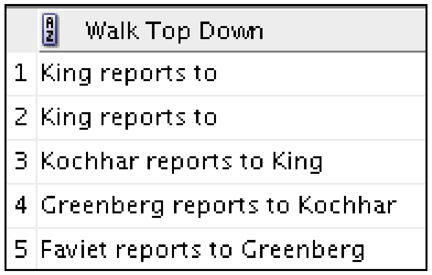
```
SELECT employee_id, last_name, job_id, manager_id
FROM   employees
START  WITH  employee_id = 101
CONNECT BY PRIOR manager_id = employee_id ;
```



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

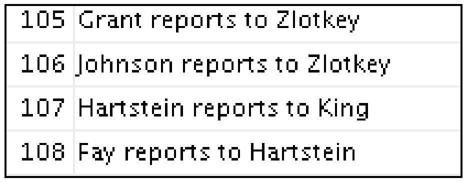
Walking the Tree: From the Top Down

```
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 ;
```



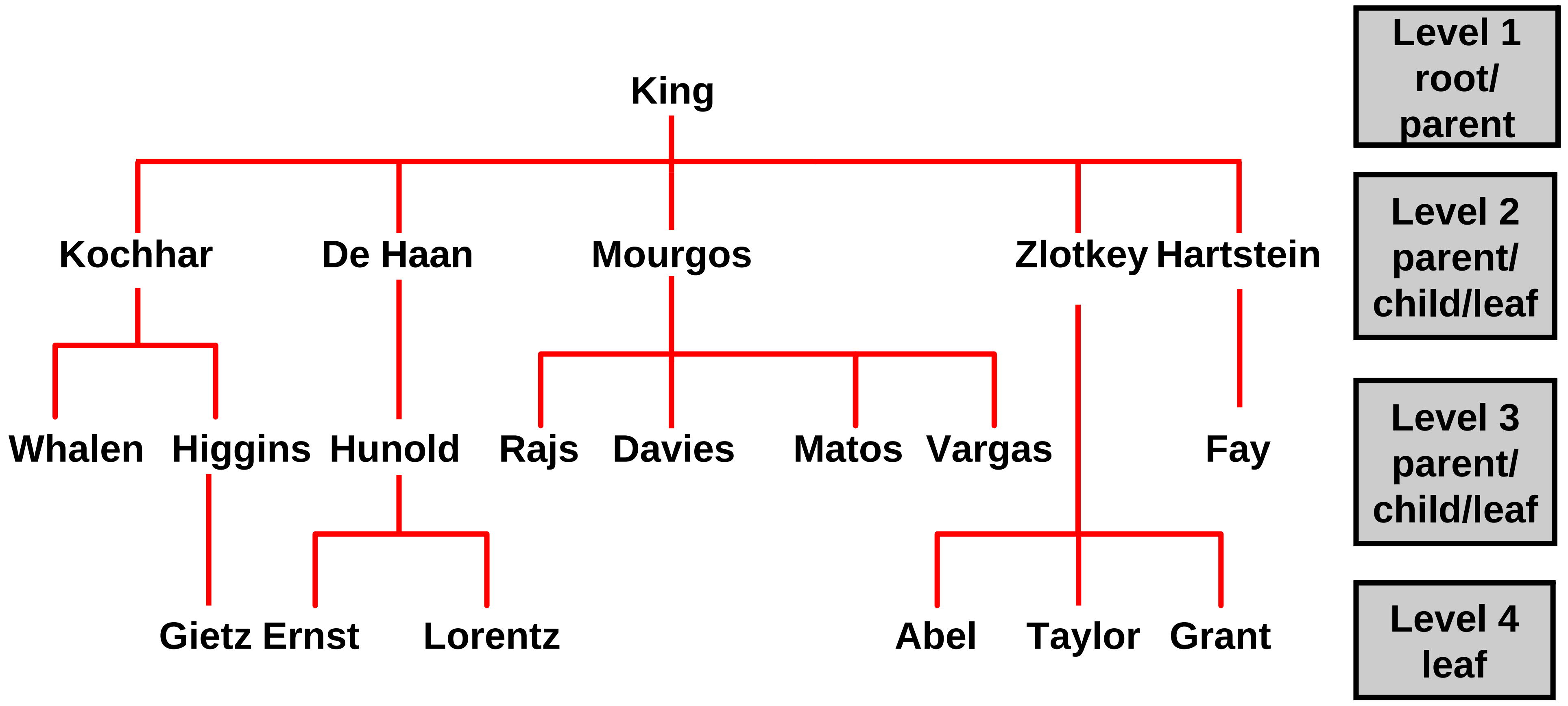
	Walk Top Down
1	King reports to
2	King reports to
3	Kochhar reports to King
4	Greenberg reports to Kochhar
5	Faviet reports to Greenberg

...



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

Ranking Rows with the `LEVEL` Pseudocolumn



Formatting Hierarchical Reports Using `LEVEL` and `LPAD`

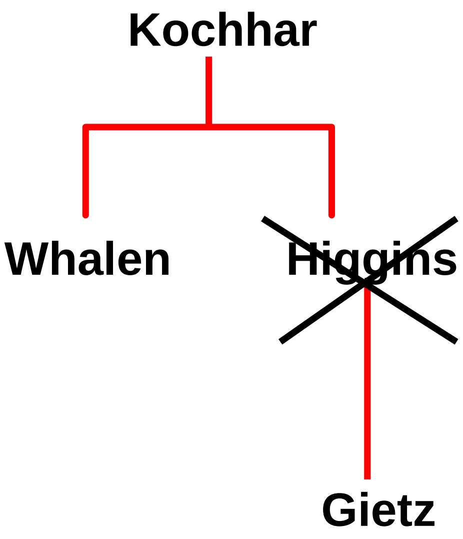
Create a report displaying company management levels, beginning with the highest level and indenting each of the following levels.

```
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
```

Pruning Branches

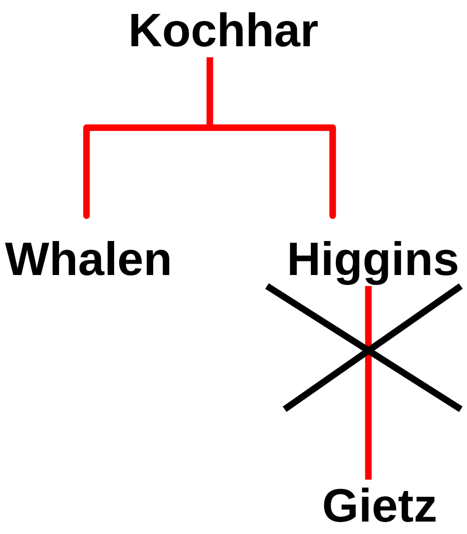
Use the `WHERE` clause to eliminate a node.

```
WHERE last_name != 'Higgins'
```



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

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



Summary

In this appendix, you should have learned that you can:

- 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