DATABASE DESIGN & IMPLEMENTATION

Objectives

- Inner and outer joins

Purpose

- All joins up until now have returned data that matched the join condition.
- Sometimes, however, we want to retrieve both the data that meets the join condition, and the data that does not meet the join condition.
- The outer joins in ANSI-99 SQL allow this functionality.

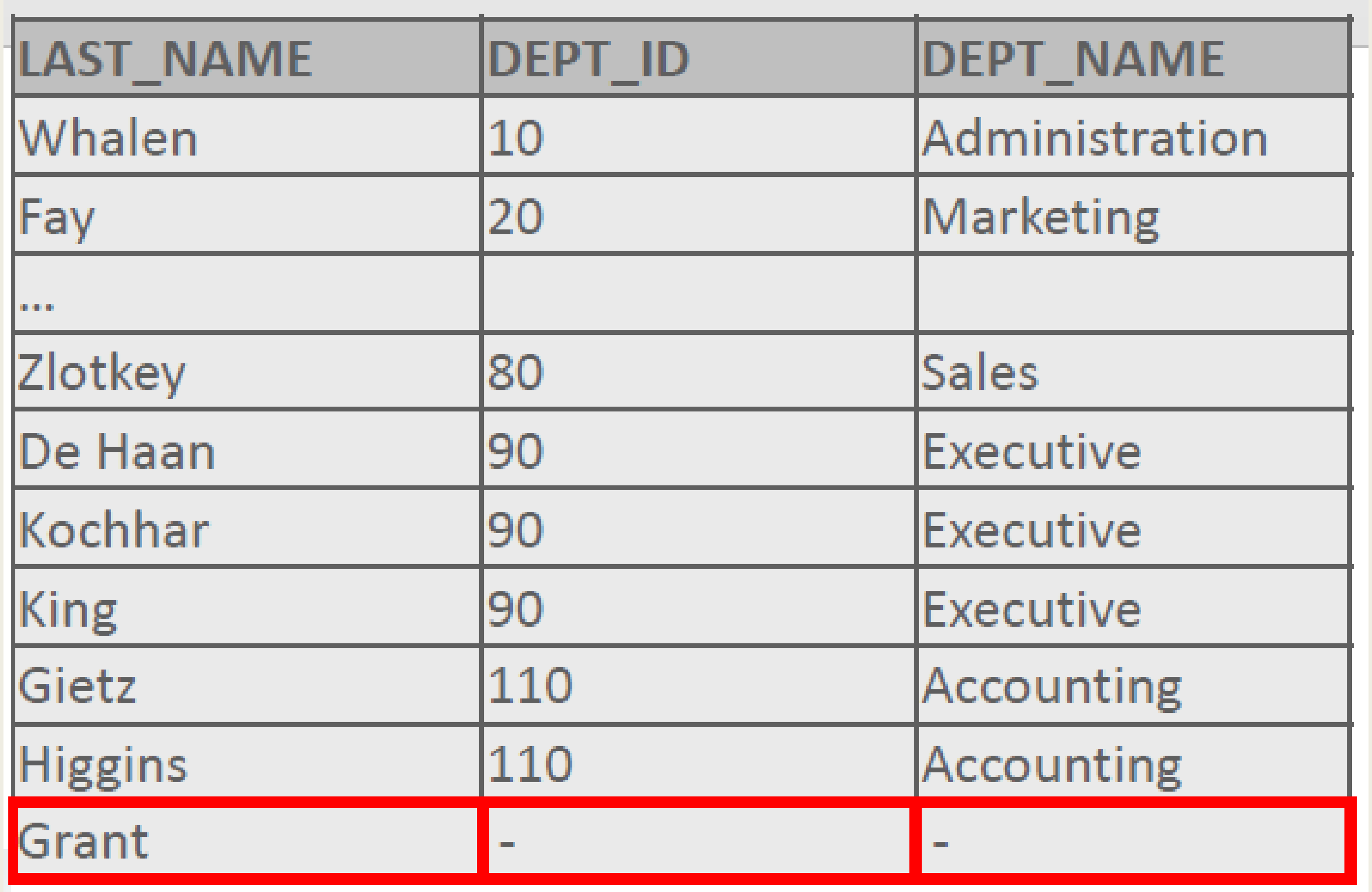
INNER and OUTER joins

- In ANSI-99 SQL , a join of two or more tables that returns only the matched rows is called an inner join.
- When a join returns the unmatched rows as well as the matched rows, it is called an outer join.
- Outer join syntax uses the terms “left, full, and right”
- These names are associated with the order of the table names in the FROM clause of the SELECT statement.

LEFT and RIGHT OUTER joins

```
SELECT e.last_name, d.department_id, d.department_name
FROM employees e
LEFT OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

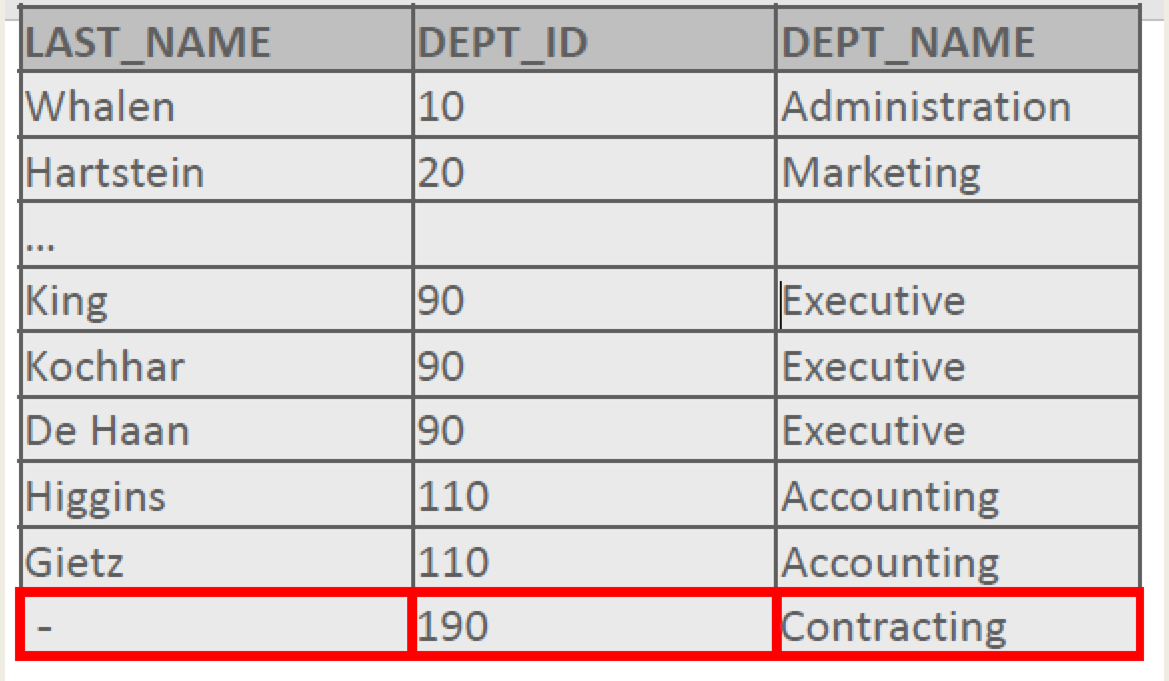
The table name listed to the left of the words “left outer join” is referred to as the left table, in this case that is employees.



LAST_NAME	DEPT_ID	DEPT_NAME
Whalen	10	Administration
Fay	20	Marketing
…		
Zlotkey	80	Sales
De Haan	90	Executive
Kochhar	90	Executive
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant	-	-

LEFT and RIGHT OUTER joins

- This query will return all employees last names, both those that are assigned to a department and those that are not.
- What do you think would be returned if we changed the outer join to be a right outer join?

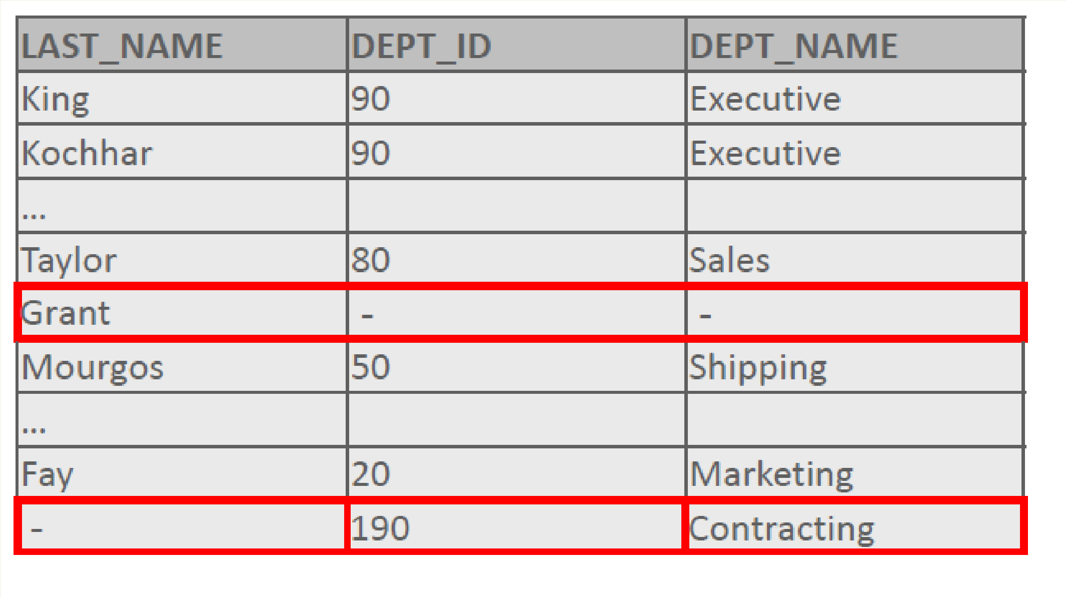


LAST_NAME	DEPT_ID	DEPT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
…		
King	90	Executive
Kochhar	90	Executive
De Haan	90	Executive
Higgins	110	Accounting
Gietz	110	Accounting
-	190	Contracting

FULL OUTER join

- It is possible to create a join condition to retrieve all matching rows and all unmatched rows from both tables.
- Using a full outer join does this.
- The result set of a full out join includes all rows from a left outer join and all rows from a right outer join combined together without duplication.

FULL OUTER join example



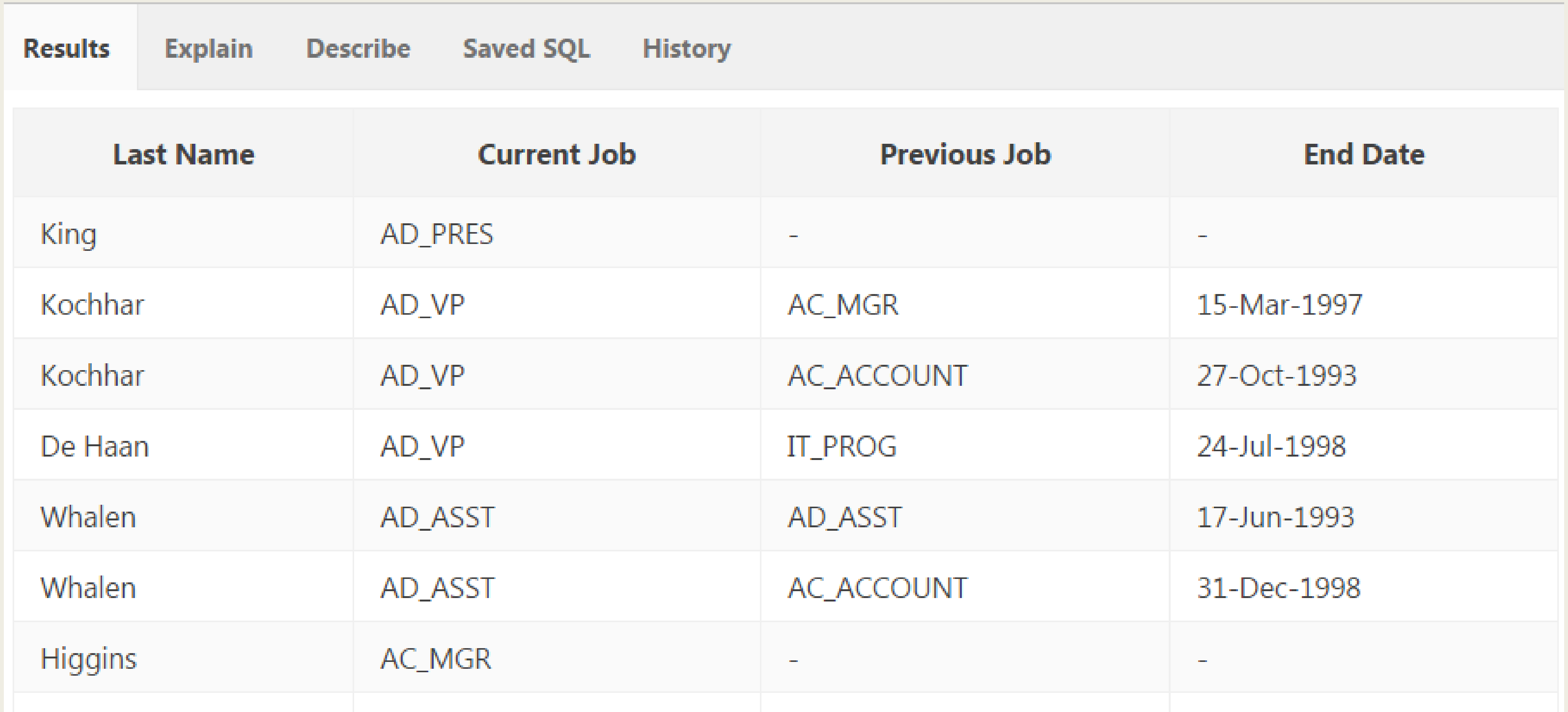
LAST_NAME	DEPT_ID	DEPT_NAME
King	90	Executive
Kochhar	90	Executive
…		
Taylor	80	Sales
Grant	-	-
Mourgos	50	Shipping
…		
Fay	20	Marketing
-	190	Contracting

```
SELECT e.last_name,
d.department_id,
d.department_name

FROM employees e

FULL OUTER JOIN departments d

ON (e.department_id =
d.department_id);
```

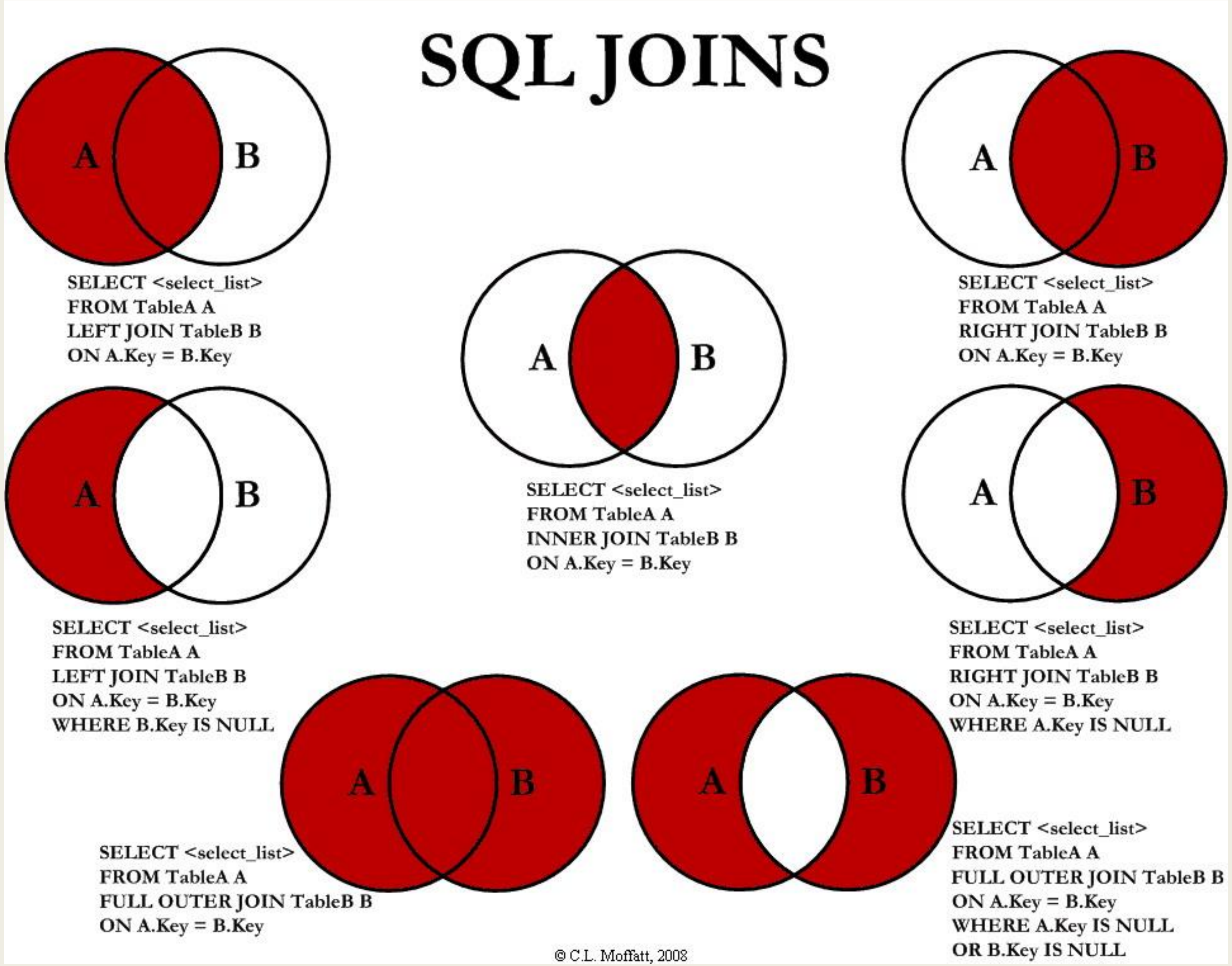
JOIN Scenario

- Construct a join to display a list of all employees, their current job_id and any previous jobs they may have held.
- The employees table contains last_name and job_id
- The job_history table contains job_id and employee_id of an employe’s previous jobs and the end_date of that job.

```
SELECT e.last_name as “Last Name”, e.job_id as
“Current Job”, jh.job_id as “Old Job”, jh.end_date
as “End Date”
FROM employees e
LEFT OUTER JOIN job_history jh
ON (e.employee_id = jh.employee_id);
```



Last Name	Current Job	Previous Job	End Date
King	AD_PRES	-	-
Kochhar	AD_VP	AC_MGR	15-Mar-1997
Kochhar	AD_VP	AC_ACCOUNT	27-Oct-1993
De Haan	AD_VP	IT_PROG	24-Jul-1998
Whalen	AD_ASST	AD_ASST	17-Jun-1993
Whalen	AD_ASST	AC_ACCOUNT	31-Dec-1998
Higgins	AC_MGR	-	-



SQL JOINS

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

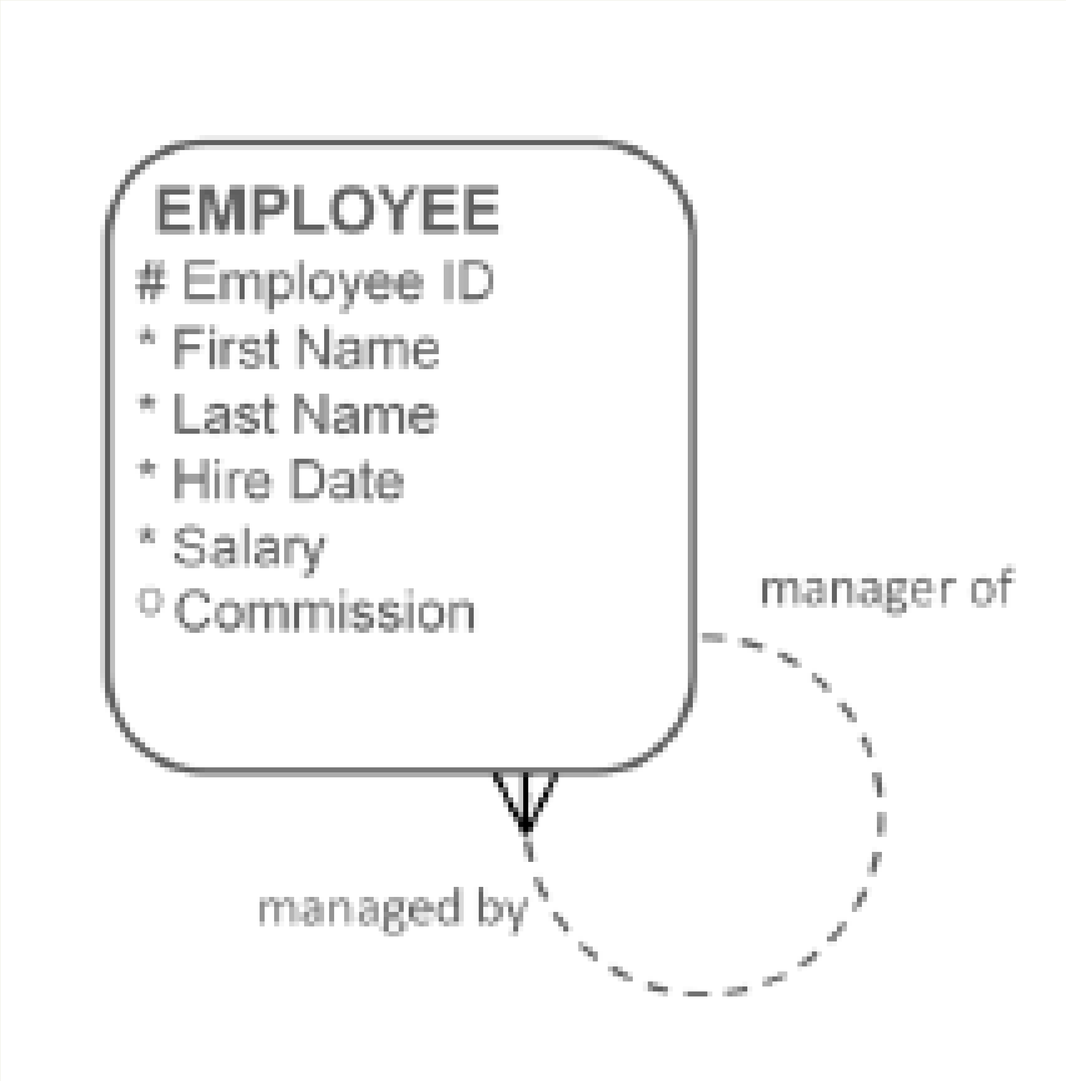
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

© C.L. Moffatt, 2008



SELF join

- In data modelling, it was sometimes necessary to show an entity with a relationship to itself.
- For example an employee can also be a manager.
- We showed this using the recursive relationship.

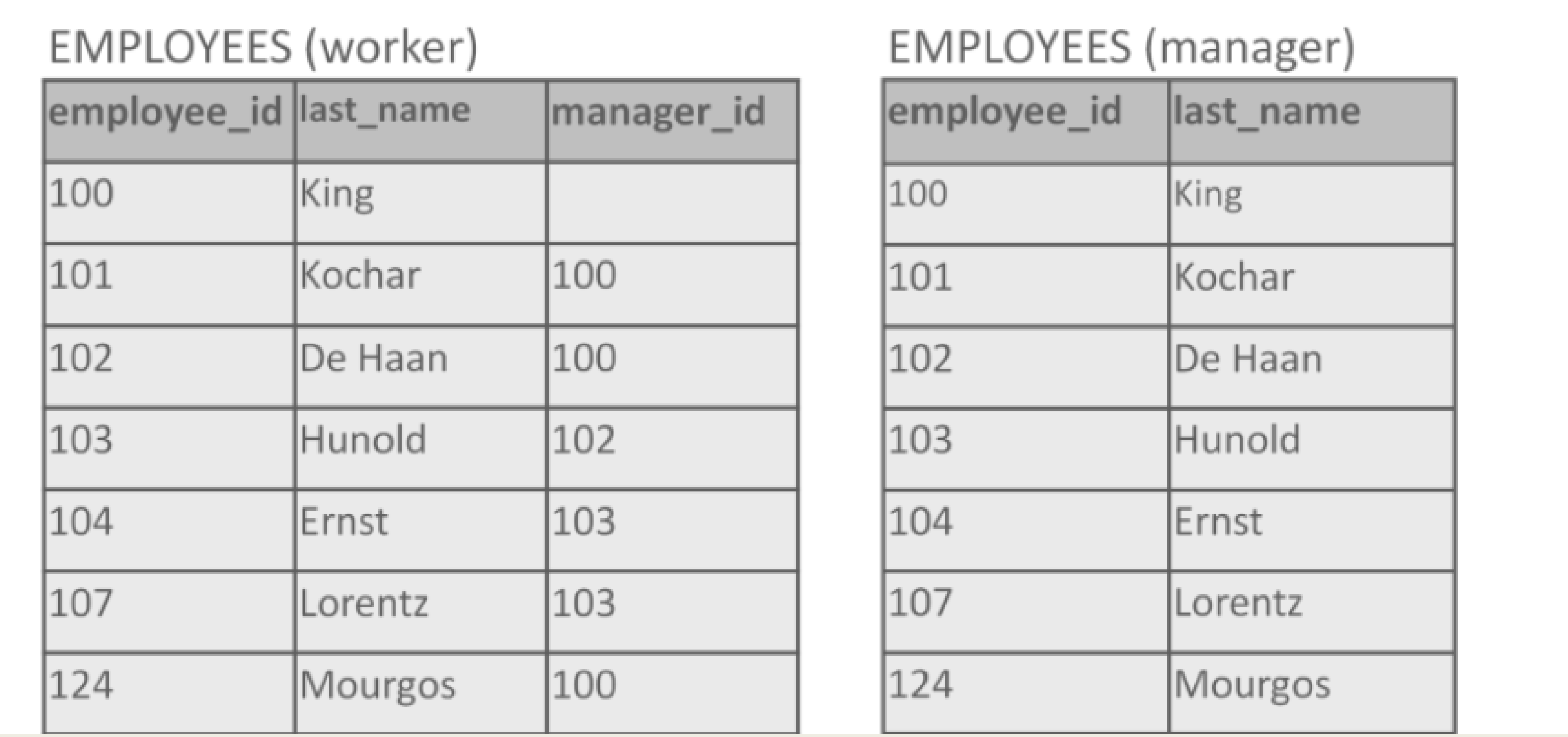
Self Join

- Once we create our employees table a special kind of join called a self join is required to access this data.
- A self join is used to join a table to itself as if it was two tables.

```
SELECT worker.last_name || ‘ works for ‘ ||
manager.last_name as “Works For”

FROM employees worker

JOIN employees manager

ON (worker.manager_id = manager.employee_id);
```

Self Join

- To join a table to itself, the table is given two aliases. This makes the database think that there are two tables.
- Manager id in the worker tables is equal to employee id in the manager table.



EMPLOYEES (worker)

employee_id	last_name	manager_id
100	King	
101	Kochar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
107	Lorentz	103
124	Mourgos	100

EMPLOYEES (manager)

employee_id	last_name
100	King
101	Kochar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos

Self Join

- Choose alias names that relate to the data’s association with that table.

Practice

- Display the employee’s last name and employee number along with the managers last name and manager number.
- Modify the previous to display those that do not have managers and order the output by the manager’s name.
- Display the names and hire dates for all employees who were hired before their managers along with their managers names and hire dates. Label the columns appropriately.