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

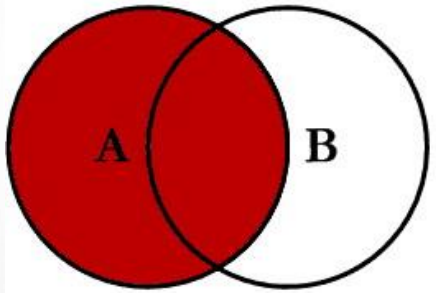
- CONSTRUCT A JOIN TO DISPLAY A LIST OF ALL EMPLOYEES, THEIR CURRENT JOB_ID AND ANY PREVIOUS JOBS THEY MAY HAVE HELD.
- THE EMPLOYEES TABLE CONTAINS LAST_NAME AND JOB_ID
- THE JOB_HISTORY TABLE CONTAINS JOB_ID AND EMPLOYEE_ID OF AN EMPLOYE'S PREVIOUS JOBS AND THE END_DATE OF THAT JOB.

```
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);
```

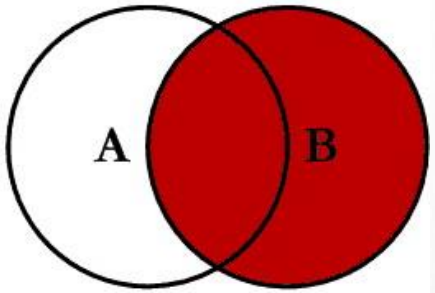
Results | Explain | Describe | Saved SQL | History

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	-	-

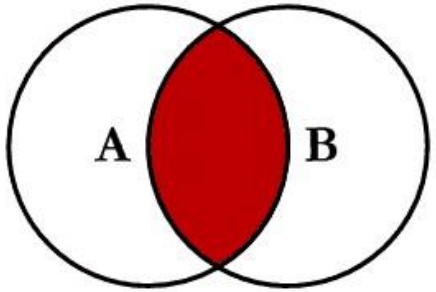
SQL JOINS



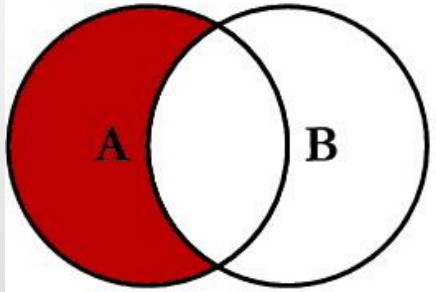

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



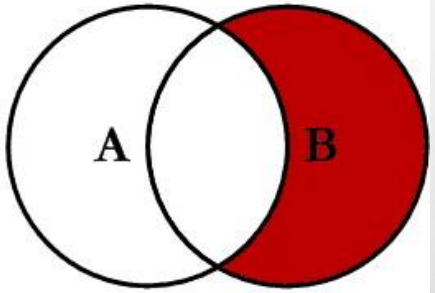

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



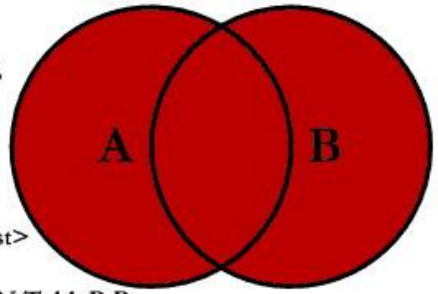

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



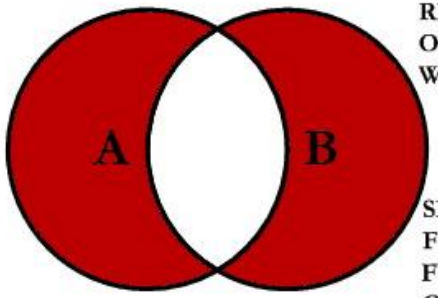

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



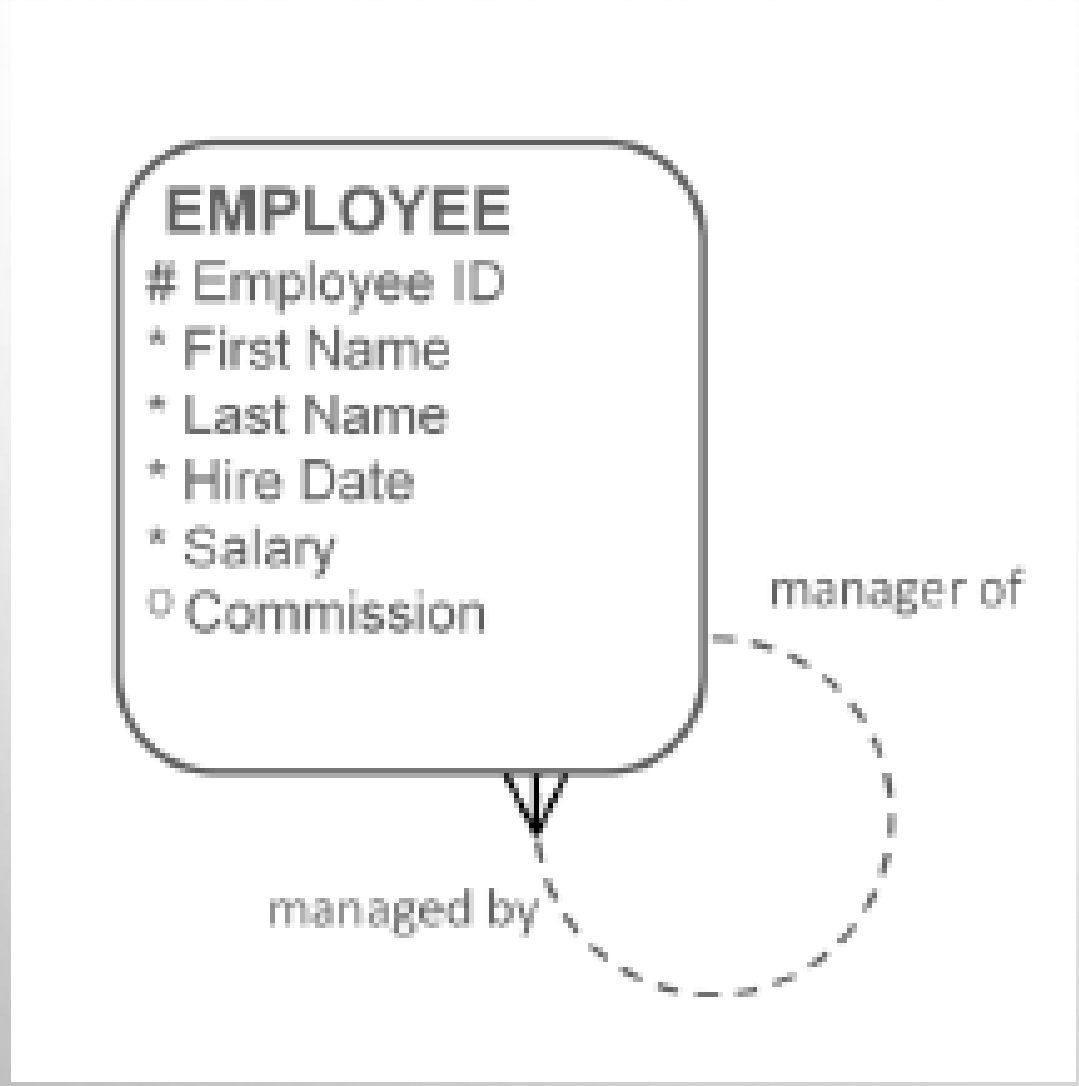

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
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
```



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.