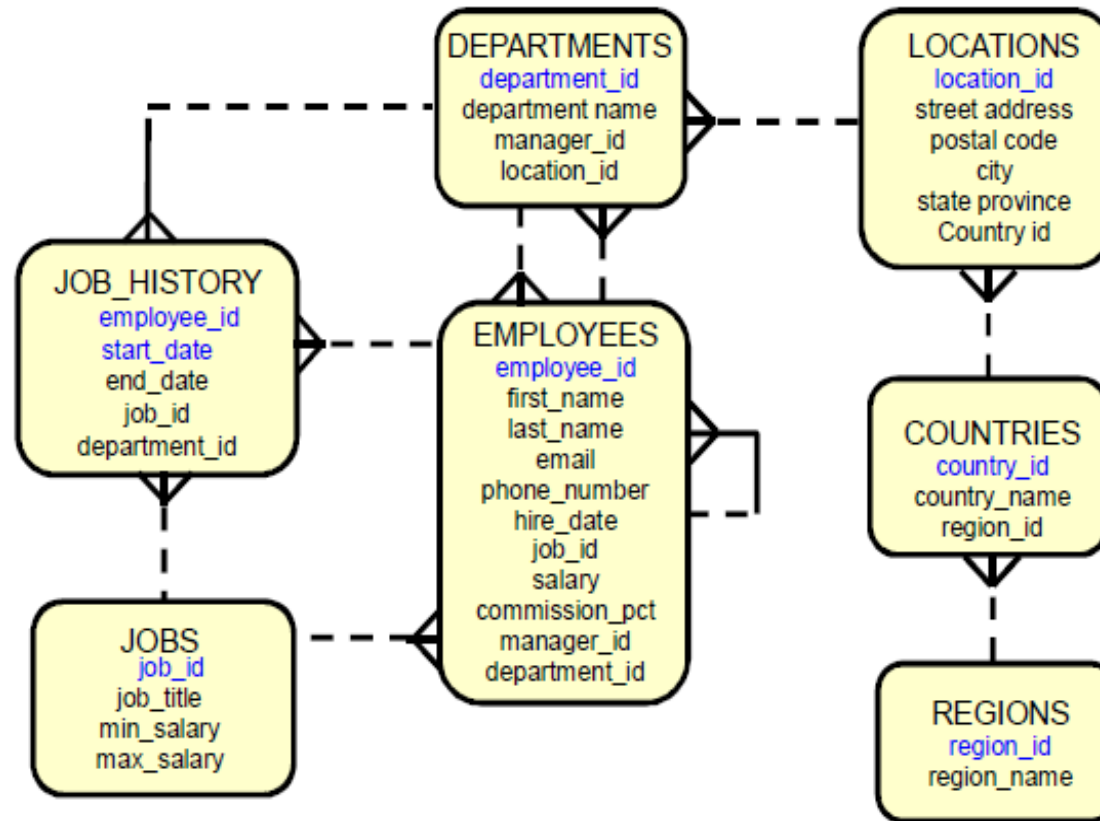




# Displaying Data from multiple tables using joins

## Human Resources (HR) Schema



## Obtaining Data from Multiple Tables

EMPLOYEES

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	200	Whalen	10
2	201	Hartstein	20
3	202	Fay	20
...			
18	174	Abel	80
19	176	Taylor	80
20	178	Grant	(null)

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

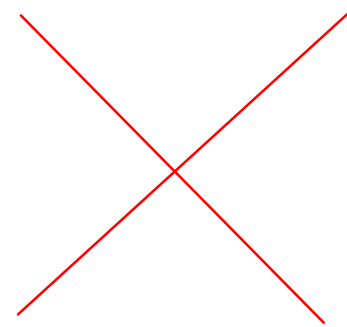
↓ ↓

	EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	200	10	Administration
2	201	20	Marketing
3	202	20	Marketing
4	124	50	Shipping
...			
18	205	110	Accounting
19	206	110	Accounting

## Cartesian Products

- A Cartesian product is formed when:
  - A join condition is omitted
  - A join condition is invalid
  - All rows in the first table are joined to all rows in the second table
- To avoid a Cartesian product, always include a valid join condition in a WHERE clause.

```
SELECT EMPLOYEES.EMPLOYEE_ID ,  
EMPLOYEES.FIRST_NAME,  
EMPLOYEES.DEPARTMENT_ID,  
DEPARTMENTS.DEPARTMENT_NAME  
FROM EMPLOYEES,  
DEPARTMENTS  
ORDER BY EMPLOYEE_ID;
```



## Types of Joins

### Oracle Proprietary Joins (8i and prior):

- Equijoin
- Nonequijoin
- Outer join
- Self join

### SQL: 1999 Compliant Joins:

- Cross joins
- Natural joins
- Using clause
- Full or two sided outer joins
- Arbitrary join conditions for outer joins

## Joining Tables Using Oracle Syntax

Use a join to query data from more than one table.

```
SELECT    table1.column, table2.column
FROM      table1, table2
WHERE     table1.column1 = table2.column2;
```

- Write the join condition in the WHERE clause.
- Prefix the column name with the table name when the same column name appears in more than one table.

### Guidelines

- When writing a SELECT statement that joins tables, precede the column name with the table name for clarity and to enhance database access.
- If the same column name appears in more than one table, the column name must be prefixed with the table name.
- To join  $n$  tables together, you need a minimum of  $n-1$  join conditions. For example, to join four tables, a minimum of three joins is required. This rule may not apply if your table has a concatenated primary key, in which case more than one column is required to uniquely identify each row.

## What Is an Equijoin?

EMPLOYEES

EMPLOYEE_ID	DEPARTMENT_ID
200	10
201	20
202	20
124	50
141	50
142	50
143	50
144	50
103	60
104	60
105	60
106	60
107	60
108	60
109	60
110	60
111	60
112	60
113	60
114	60
115	60
116	60
117	60
118	60
119	60
120	60
121	60
122	60
123	60
124	60
125	60
126	60
127	60
128	60
129	60
130	60
131	60
132	60
133	60
134	60
135	60
136	60
137	60
138	60
139	60
140	60
141	60
142	60
143	60
144	60
145	60
146	60
147	60
148	60
149	60
150	60
151	60
152	60
153	60
154	60
155	60
156	60
157	60
158	60
159	60
160	60
161	60
162	60
163	60
164	60
165	60
166	60
167	60
168	60
169	60
170	60
171	60
172	60
173	60
174	60
175	60
176	60
177	60
178	60
179	60
180	60
181	60
182	60
183	60
184	60
185	60
186	60
187	60
188	60
189	60
190	60
191	60
192	60
193	60
194	60
195	60
196	60
197	60
198	60
199	60
200	60
201	60
202	60
203	60
204	60
205	60
206	60
207	60
208	60
209	60
210	60
211	60
212	60
213	60
214	60
215	60
216	60
217	60
218	60
219	60
220	60
221	60
222	60
223	60
224	60
225	60
226	60
227	60
228	60
229	60
230	60
231	60
232	60
233	60
234	60
235	60
236	60
237	60
238	60
239	60
240	60
241	60
242	60
243	60
244	60
245	60
246	60
247	60
248	60
249	60
250	60
251	60
252	60
253	60
254	60
255	60
256	60
257	60
258	60
259	60
260	60
261	60
262	60
263	60
264	60
265	60
266	60
267	60
268	60
269	60
270	60
271	60
272	60
273	60
274	60
275	60
276	60
277	60
278	60
279	60
280	60
281	60
282	60
283	60
284	60
285	60
286	60
287	60
288	60
289	60
290	60
291	60
292	60
293	60
294	60
295	60
296	60
297	60
298	60
299	60
300	60
301	60
302	60
303	60
304	60
305	60
306	60
307	60
308	60
309	60
310	60
311	60
312	60
313	60
314	60
315	60
316	60
317	60
318	60
319	60
320	60
321	60
322	60
323	60
324	60
325	60
326	60
327	60
328	60
329	60
330	60
331	60
332	60
333	60
334	60
335	60
336	60
337	60
338	60
339	60
340	60
341	60
342	60
343	60
344	60
345	60
346	60
347	60
348	60
349	60
350	60
351	60
352	60
353	60
354	60
355	60
356	60
357	60
358	60
359	60
360	60
361	60
362	60
363	60
364	60
365	60
366	60
367	60
368	60
369	60
370	60
371	60
372	60
373	60
374	60
375	60
376	60
377	60
378	60
379	60
380	60
381	60
382	60
383	60
384	60
385	60
386	60
387	60
388	60
389	60
390	60
391	60
392	60
393	60
394	60
395	60
396	60
397	60
398	60
399	60
400	60
401	60
402	60
403	60
404	60
405	60
406	60
407	60
408	60
409	60
410	60
411	60
412	60
413	60
414	60
415	60
416	60
417	60
418	60
419	60
420	60
421	60
422	60
423	60
424	60
425	60
426	60
427	60
428	60
429	60
430	60
431	60
432	60
433	60
434	60
435	60
436	60
437	60
438	60
439	60
440	60
441	60
442	60
443	60
444	60
445	60
446	60
447	60
448	60
449	60
450	60
451	60
452	60
453	60
454	60
455	60
456	60
457	60
458	60
459	60
460	60
461	60
462	60
463	60
464	60
465	60
466	60
467	60
468	60
469	60
470	60
471	60
472	60
473	60
474	60
475	60
476	60
477	60
478	60
479	60
480	60
481	60
482	60
483	60
484	60
485	60
486	60
487	60
488	60
489	60
490	60
491	60
492	60
493	60
494	60
495	60
496	60
497	60
498	60
499	60
500	60
501	60
502	60
503	60
504	60
505	60
506	60
507	60
508	60
509	60
510	60
511	60
512	60
513	60
514	60
515	60
516	60
517	60
518	60
519	60
520	60
521	60
522	60
523	60
524	60
525	60
526	60
527	60
528	60
529	60
530	60
531	60
532	60
533	60
534	60
535	60
536	60
537	60
538	60
539	60
540	60
541	60
542	60
543	60
544	60
545	60
546	60
547	60
548	60
549	60
550	60
551	60
552	60
553	60
554	60
555	60
556	60
557	60
558	60
559	60
560	60
561	60
562	60
563	60
564	60
565	60
566	60
567	60
568	60
569	60
570	60
571	60
572	60
573	60
574	60
575	60
576	60
577	60
578	60
579	60
580	60
581	60
582	60
583	60
584	60
585	60
586	60
587	60
588	60
589	60
590	60
591	60
592	60
593	60
594	60
595	60
596	60
597	60
598	60
599	60
600	60
601	60
602	60
603	60
604	60
605	60
606	60
607	60
608	60
609	60
610	60
611	60
612	60
613	60
614	60
615	60
616	60
617	60
618	60
619	60
620	60
621	60
622	60
623	60
624	60
625	60
626	60
627	60
628	60
629	60
630	60
631	60
632	60
633	60
634	60
635	60
636	60
637	60
638	60
639	60
640	60
641	60
642	60
643	60
644	60
645	60
646	60
647	60
648	60
649	60
650	60
651	60
652	60
653	60
654	60
655	60
656	60
657	60
658	60
659	60
660	60
661	60
662	60
663	60
664	60
665	60
666	60
667	60
668	60
669	60
670	60
671	60
672	60
673	60
674	60
675	60
676	60
677	60
678	60
679	60
680	60
681	60
682	60
683	60
684	60
685	60
686	60
687	60
688	60
689	60
690	60
691	60
692	60
693	60
694	60
695	60
696	60
697	60
698	60
699	60
700	60
701	60
702	60
703	60
704	60
705	60
706	60
707	60
708	60
709	60
710	60
711	60
712	60
713	60
714	60
715	60
716	60
717	60
718	60
719	60
720	60
721	60
722	60
723	60
724	60
725	60
726	60
727	60
728	60
729	60
730	60
731	60
732	60
733	60
734	60
735	60
736	60
7	



## Qualifying Ambiguous Column Names

- Use table prefixes to qualify column names that are in multiple tables.
- Improve performance by using table prefixes.
- Distinguish columns that have identical names but reside in different tables by using column aliases.

```
SELECT  
EMPLOYEES.EMPLOYEE_ID ,  
EMPLOYEES.FIRST_NAME,  
EMPLOYEES.DEPARTMENT_ID,  
DEPARTMENTS.DEPARTMENT_NAME  
FROM  
EMPLOYEES,  
DEPARTMENTS  
WHERE EMPLOYEES.DEPARTMENT_ID=DEPARTMENTS.DEPARTMENT_ID  
ORDER BY EMPLOYEE_ID;
```



## Using Table Aliases

- Simplify queries by using table aliases
- Improve performance by using table prefixes

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e, departments d  
WHERE  e.department_id = d.department_id;
```

### Guidelines

- Table aliases can be up to 30 characters in length, but the shorter they are the better.
- If a table alias is used for a particular table name in the FROM clause, then that table alias must be substituted for the table name throughout the SELECT statement.
- Table aliases should be meaningful.
- The table alias is valid only for the current SELECT statement.

## Joining More than Two Tables

EMPLOYEES

LAST_NAME	DEPARTMENT_ID
King	90
Kochhar	90
De Haan	90
Hunold	60
Ernst	60
Lorentz	60
Grant	10
Whalen	10
Hartstein	20
Fay	20
Higgins	110
Gietz	110

20 rows selected.

DEPARTMENTS

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.

LOCATIONS

LOCATION_ID	CITY
1400	Southlake
1500	South San Francisco
1700	Seattle
1800	Toronto
2500	Oxford

To join  $n$  tables together, you need a minimum of  $n-1$  join conditions. For example, to join three tables, a minimum of two joins is required.

## Nonequijoins

**EMPLOYEES**

LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000
Hunold	9000
Ernst	6000
Lorentz	4200
Mourgos	5800
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500

**JOB\_GRADES**

GRA	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e, job_grades j
WHERE  e.salary BETWEEN j.lowest_sal AND j.highest_sal;
```

Salary in the **EMPLOYEES** table must be between lowest salary and highest salary in the **JOB\_GRADES** table.

### Nonequijoins

A nonequijoin is a join condition containing something other than an equality operator.

The relationship between the **EMPLOYEES** table and the **JOB\_GRADES** table has an example of a nonequijoin. A relationship between the two tables is that the **SALARY** column in the **EMPLOYEES** table must be between the values in the **LOWEST\_SALARY** and **HIGHEST\_SALARY** columns of the **JOB\_GRADES** table. The relationship is obtained using an operator other than equals (=).



# Outer Join

Main table

EMP		
EMP ID	Name	dept_id
1	khaled	10
2	ali	20
3	samer	30
4	Hassan	30
5	nader	

DEPT	
dept_id	name
10	Accounting
20	sales
30	marketing
40	HR

- Emp ID 5 has no department  
So if you used natural join `emp.dept_id=dept.dept_id`, then he will not appear
- So we have to use outer join (+), always place it on the side that have the missing data  
`where emp.dept_id=dept.dept_id(+)`
- If a FK in table1 has null values and it is the main table that you want to display the data from, then the outer join should be table2 side which has the reference from table1

## Outer Joins Syntax

- You use an outer join to also see rows that do not meet the join condition.
- The outer join operator is the plus sign (+).

```
SELECT table1.column, table2.column  
FROM   table1, table2  
WHERE  table1.column(+) = table2.column;
```

```
SELECT table1.column, table2.column  
FROM   table1, table2  
WHERE  table1.column = table2.column(+);
```



# Outer Join

EMP		
EMP ID	Name	dept_id
1	khaled	10
2	ali	20
3	samer	30
4	Hassan	30
5	nader	

DEPT	
dept_id	name
10	Accounting
20	sales
30	marketing
40	HR

Where emp.dept\_id=dept.dept\_id(+)

emp.EMP ID	emp.Name	emp.dept_id	dept.name
1	khaled	10	Accounting
2	ali	20	sales
3	samer	30	marketing
4	Hassan	30	marketing
5	nader		

Where emp.dept\_id(+)=dept.dept\_id

emp.EMP ID	emp.Name	dept.dept_id	dept.name
1	khaled	10	Accounting
2	ali	20	sales
3	samer	30	marketing
4	Hassan	30	marketing
		40	HR

## Joining a Table to Itself

EMPLOYEES (WORKER)

EMPLOYEE_ID	LAST_NAME	MANAGER_ID
200	Whalen	101
201	Hartstein	100
202	Fay	201
205	Higgins	101
206	Gietz	205
100	King	(null)
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103

...

EMPLOYEES (MANAGER)

EMPLOYEE_ID	LAST_NAME
200	Whalen
201	Hartstein
202	Fay
205	Higgins
206	Gietz
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst

...

MANAGER\_ID in the WORKER table is equal to  
EMPLOYEE\_ID in the MANAGER table.

SELECT

worker.EMPLOYEE\_ID,  
WORKER.FIRST\_NAME,  
WORKER.MANAGER\_ID,

manager.first\_name

FROM

EMPLOYEES WORKER,

EMPLOYEES MANAGER

WHERE WORKER.MANAGER\_ID=MANAGER.EMPLOYEE\_ID;



# Joining Tables Using SQL: 1999 Syntax

Use a join to query data from more than one table.

```
SELECT  table1.column, table2.column
FROM    table1
[CROSS JOIN table2] |
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
  ON (table1.column_name = table2.column_name)] |
[LEFT|RIGHT|FULL OUTER JOIN table2
  ON (table1.column_name = table2.column_name)];
```

## Defining Joins

Using the SQL: 1999 syntax, you can obtain the same results as what was shown in the prior pages.

In the syntax:

<code>table1.column</code>	Denotes the table and column from which data is retrieved
<code>CROSS JOIN</code>	Returns a Cartesian product from the two tables
<code>NATURAL JOIN</code>	Joins two tables based on the same column name
<code>JOIN table</code> <code>USING column_name</code>	Performs an equijoin based on the column name
<code>JOIN table ON</code> <code>table1.column_name</code> <code>= table2.column_name</code>	Performs an equijoin based on the condition in the ON clause
<code>LEFT/RIGHT/FULL OUTER</code>	



## Creating Cross Joins

- The **CROSS JOIN** clause produces the cross-product of two tables.
- This is the same as a Cartesian product between the two tables.

```
SELECT last_name, department_name  
FROM   employees  
CROSS JOIN departments;
```

This is the same but in old join format

```
SELECT last_name, department_name  
FROM employees, departments;
```



## Creating Natural Joins

- The **NATURAL JOIN** clause is based on all columns in the two tables that have the same name.
- It selects rows from the two tables that have equal values in all matched columns.
- If the columns having the same names have different data types, then an error is returned.

### 1999 Format

```
SELECT department_id, department_name,  
       location_id, city  
FROM   departments  
NATURAL JOIN locations;
```

### Old Format as Equijoin

The natural join can also be written as an equijoin:

```
SELECT department_id, department_name,  
       departments.location_id, city  
FROM   departments, locations  
WHERE  departments.location_id = locations.location_id;
```

## Creating Joins with the USING Clause

- If several columns have the same names but the data types do not match, the `NATURAL JOIN` clause can be modified with the `USING` clause to specify the columns that should be used for an equijoin.  
Note: Use the `USING` clause to match only one column when more than one column matches.
- Do not use a table name or alias in the referenced columns.
- The `NATURAL JOIN` and `USING` clauses are mutually exclusive.

```
SELECT e.employee_id, e.last_name, d.location_id
FROM employees e JOIN departments d
USING (department_id);
```

This can also be written as an equijoin:

```
SELECT employee_id, last_name,
       employees.department_id, location_id
FROM employees, departments
WHERE employees.department_id = departments.department_id;
```



## Creating Joins with the ON Clause

- The join condition for the natural join is basically an equijoin of all columns with the same name.
- To specify arbitrary conditions or specify columns to join, the ON clause is used.
- Separates the join condition from other search conditions.
- The ON clause makes code easy to understand.

1999 format

```
SELECT
EMPLOYEES.EMPLOYEE_ID ,
EMPLOYEES.FIRST_NAME,
DEPARTMENTS.DEPARTMENT_ID, ---here prefix should be use
DEPARTMENTS.DEPARTMENT_NAME
FROM EMPLOYEES join
DEPARTMENTS
ON (EMPLOYEES.DEPARTMENT_ID=DEPARTMENTS.DEPARTMENT_ID)
ORDER BY EMPLOYEE_ID;
```

Old Format

```
SELECT
EMPLOYEES.EMPLOYEE_ID ,
EMPLOYEES.FIRST_NAME,
EMPLOYEES.DEPARTMENT_ID,
DEPARTMENTS.DEPARTMENT_NAME
FROM EMPLOYEES,
DEPARTMENTS
WHERE EMPLOYEES.DEPARTMENT_ID=DEPARTMENTS.DEPARTMENT_ID
ORDER BY EMPLOYEE_ID;
```

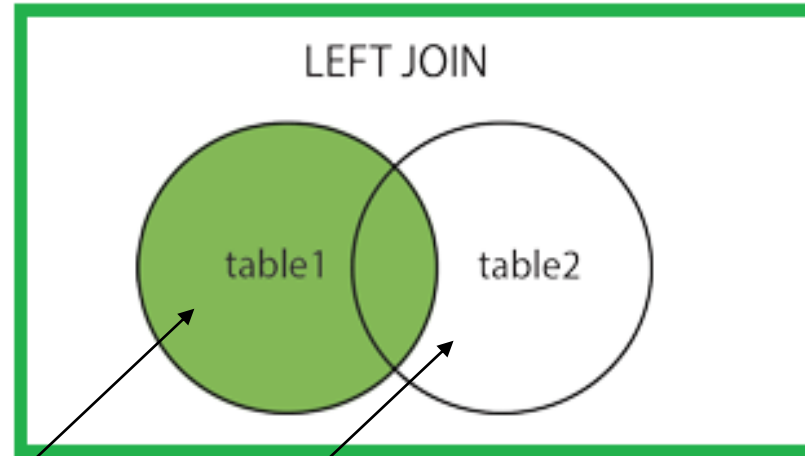
## Creating Three-Way Joins with the ON Clause

```
SELECT employee_id, city, department_name
FROM   employees e
JOIN   departments d
ON     d.department_id = e.department_id
JOIN   locations l
ON     d.location_id = l.location_id;
```

### Three-Way Joins

A three-way join is a join of three tables. In SQL: 1999 compliant syntax, joins are performed from left to right, so the first join to be performed is `EMPLOYEES JOIN DEPARTMENTS`. The first join condition can reference columns in `EMPLOYEES` and `DEPARTMENTS` but cannot reference columns in `LOCATIONS`. The second join condition can reference columns from all three tables.

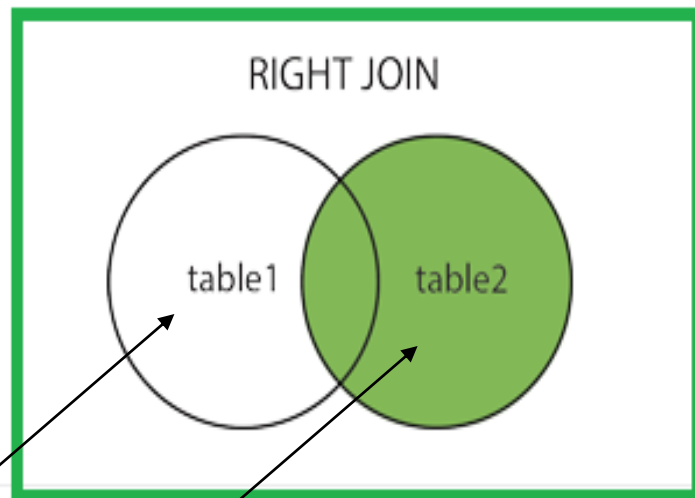
# Left Outer Join



```
SELECT
EMPLOYEES.EMPLOYEE_ID ,
EMPLOYEES.FIRST_NAME,
EMPLOYEES.DEPARTMENT_ID,
DEPARTMENTS.DEPARTMENT_NAME
FROM EMPLOYEES
left OUTER JOIN DEPARTMENTS
on( EMPLOYEES.DEPARTMENT_ID=DEPARTMENTS.DEPARTMENT_ID)
ORDER BY EMPLOYEE_ID;
```

```
SELECT
EMPLOYEES.EMPLOYEE_ID ,
EMPLOYEES.FIRST_NAME,
EMPLOYEES.DEPARTMENT_ID,
DEPARTMENTS.DEPARTMENT_NAME
FROM EMPLOYEES,
DEPARTMENTS
WHERE EMPLOYEES.DEPARTMENT_ID=DEPARTMENTS.DEPARTMENT_ID(+)
ORDER BY EMPLOYEE_ID;
```

# Right Outer Join

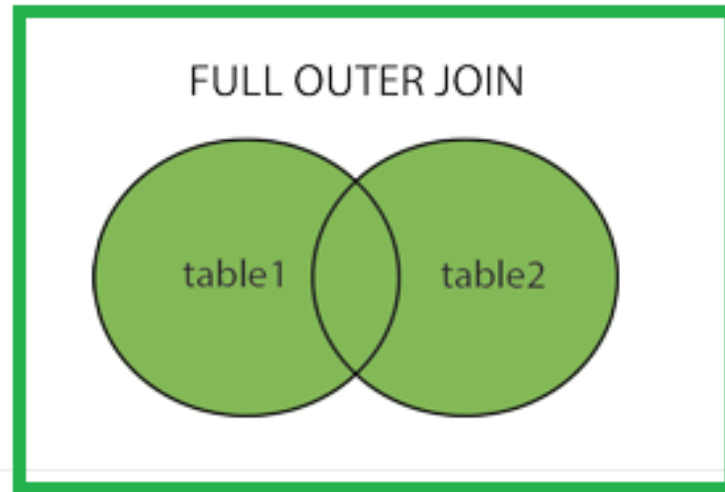


```
SELECT
EMPLOYEES.EMPLOYEE_ID ,
EMPLOYEES.FIRST_NAME,
EMPLOYEES.DEPARTMENT_ID,
DEPARTMENTS.DEPARTMENT_NAME
FROM EMPLOYEES
right OUTER JOIN DEPARTMENTS
ON( EMPLOYEES.DEPARTMENT_ID=DEPARTMENTS.DEPARTMENT_ID)
ORDER BY EMPLOYEE_ID;
```

```
SELECT
EMPLOYEES.EMPLOYEE_ID ,
EMPLOYEES.FIRST_NAME,
EMPLOYEES.DEPARTMENT_ID,
DEPARTMENTS.DEPARTMENT_NAME
FROM EMPLOYEES,
DEPARTMENTS
WHERE EMPLOYEES.DEPARTMENT_ID(+) =DEPARTMENTS.DEPARTMENT_ID
ORDER BY EMPLOYEE_ID;
```



# full Outer Join



```
SELECT  
EMPLOYEES.EMPLOYEE_ID ,  
EMPLOYEES.FIRST_NAME,  
EMPLOYEES.DEPARTMENT_ID,  
DEPARTMENTS.DEPARTMENT_NAME  
FROM EMPLOYEES  
full OUTER JOIN DEPARTMENTS  
ON( EMPLOYEES.DEPARTMENT_ID=DEPARTMENTS.DEPARTMENT_ID)  
ORDER BY EMPLOYEE_ID;
```





# Thank You