Table of Contents

1. What is an outer join?

In a previous unit, we discussed inner joins. We use joins to bring data together from two or more tables. With an inner join between two tables, we see data from both tables only when there are matching rows in the two tables. the matching is often on the primary key of one table and the foreign key of the other. For example, we might want a list of clients and their order information, showing **only clients with orders**- we join on the cl_id which occurs in both tables..

Sometimes we want to see all of the data from one of the tables even if there is no matching data in the other table. For example, we might want a list of clients and their order information, including clients with orders and clients without orders. That requires an outer join.

When we discuss outer joins, we may use the term "preserved" table. That is the table which has all of its rows returned even if there are no matching rows in the other table.

.

2. Tables for this demo

For this discussion I wanted a small set of tables. These are the tables we used last time for the Cartesian product demo-the cross join. Note that we have employees with no projects and a department with no employees and employees with no department. Although these tables did not have a primary key defined, there is only one row with the same d_id in the z_em_dept table, only one row with the same e_id in the z_em_emp table, and only one row with the same (p_id,e_id) combination in the z_em_empproj table. This is a simplification but most business tables should have a pk. (You can experiment with this by adding another row with d_id = 100 in the z_em_dept table.)

z_em_dept

D_ID	D_Name
100	Manufacturing
150	Accounting
200	Marketing
250	Research

z_em_emp

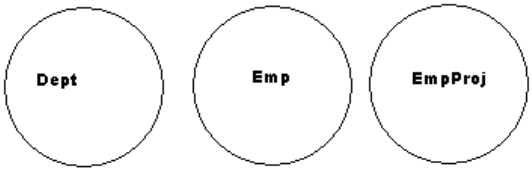
E_ID	E_Name	D_ID
1	Jones	150
2	Martin	150
3	Gates	250
4	Anders	100
5	Bossy	Null
6	Perkins	Null

z_em_empproj

P_ID	E_ID
ORDB-10	3
ORDB-10	5
Q4-SALES	2
Q4-SALES	4
ORDB-10	2
Q4-SALES	5

In this discussion, I want to talk about the joins as concepts and results; we will get to the syntax in the next document.

Sometimes graphical representations help. Each of the tables is represented by a circle. (These are not actually Venn diagrams and this does not exactly follow set theory. Use the diagrams if they help.)



3. Join Types

We will mostly look at just the z_em_dept table and the z_em_emp table.

3.1. INNER JOIN

An inner join between these two tables using the d_id attribute will result in rows for employees who have a department and departments who have an employee. This includes the rows shown here.

fromDept	D_Name	E_ID	E_Name	fromEmp
100	Manufacturing	4	Anders	100
150	Accounting	1	Jones	150
150	Accounting	2	Martin	150
250	Research	3	Gates	250

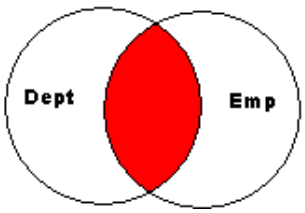
With an inner join you have the same result if you do the join in either order.

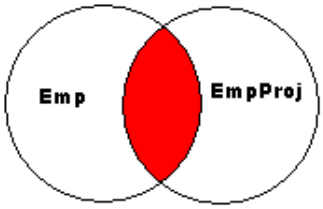
```
z_em_Dept   INNER JOIN z_em_Emp
z_em_Emp    INNER JOIN z_em_Dept
```

The red section of the diagrams represents the data returned by inner joins. From left to right:

Departments with employees

Employees assigned to projects



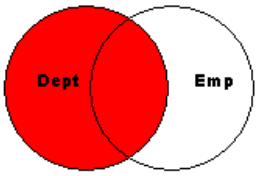


3.2. LEFT JOIN

This is a left join: z_em_Dept LEFT JOIN z_em_Emp

The result is all departments and any employees matched. We get one row each for the Manufacturing department and the Research department which each have one employee and two rows for Accounting which has two employees. The Marketing department has no employees and gets one row filled with nulls for the missing employee data.

fromDept	D_Name	E_ID	E_Name	fromEmp
100	Manufacturing	4	Anders	100
150	Accounting	1	Jones	150
150	Accounting	2	Martin	150
200	Marketing	NULL	NULL	NULL
250	Research	3	Gates	250

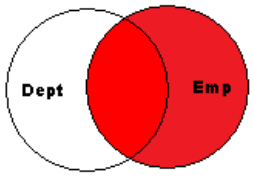


With an outer join, you get a different result if you do the join with the tables listed in the other order.

This is a left join: z_em_Emp LEFT JOIN z_em_Dept

The result is all employees and any departments matched. We have two employees without a department. (5, Bossy, null) and (6, Perkins, null). The null in the last column for these rows is the null from the table. The nulls in the first two columns for these rows are the nulls filled in by the outer join.

fromDept	D_Name	E_ID	E_Name	fromEmp
NULL	NULL	5	Bossy	NULL
NULL	NULL	6	Perkins	NULL
100	Manufacturing	4	Anders	100
150	Accounting	1	Jones	150
150	Accounting	2	Martin	150
250	Research	3	Gates	250



3.3. RIGHT JOIN

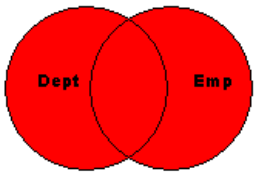
This is a right join: Dept RIGHT JOIN Employees. Note that the result set is the same as with : z_em_Emp LEFT JOIN z_em_Dept. You can use either syntax but you may find it easier at first to stick with either the Left join or the Right join.

fromDept	D_Name	E_ID	E_Name	fromEmp
NULL	NULL	6	Perkins	NULL
NULL	NULL	5	Bossy	NULL
100	Manufacturing	4	Anders	100
150	Accounting	1	Jones	150
150	Accounting	2	Martin	150
250	Research	3	Gates	250

3.4. FULL JOIN

Another join is a full join which returns all departments and the matching employees and all employees and the matching departments. (Note we almost never have a task in the assignments where a Full join is the way to solve the task. It is included here for completeness.)

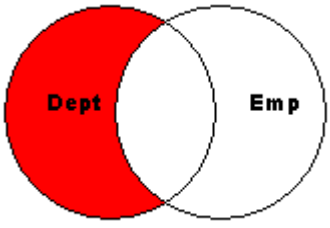
D_ID	D_Name	E_ID	E_Name
100	Manufacturing	4	Anders
150	Accounting	1	Jones
150	Accounting	2	Martin
200	Marketing	**NULL**	**NULL**
250	Research	3	Gates
NULL	**NULL**	5	Bossy
NULL	**NULL**	6	Perkins

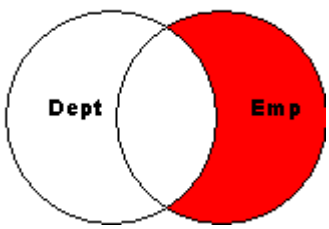


4. Unmatched joins

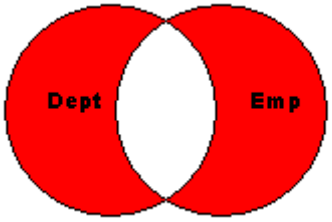
This is a common usage for the outer joins. It is not a different type of join but a way to use outer joins.

The first diagram represents departments without employees; the second diagram represents employees without departments.





The next represents departments with no employees and employees with no departments. This is not very common but the diagram is interesting.



5. Joins and the vets database

If you looked carefully at the create table statements for the tables z_em_dept, z_em_emp tabl, and z_emp_empproj, you might have noticed that there were no foreign keys created. We can still write joins because we have data values in common between the tables.

But most collections of tables are created with referential integrity and foreign keys - generally the foreign keys are set as Not Null. These settings help enforce business rules. The tables in the vets database do have referential integrity built into the table definitions. Let's consider the joins in the vets set of tables looking at just the tables for client and animals.

The clients table has a PK of cl_id. The animals table has a pk of an_id and a not null FK of cl_id which is tied to the clients table. That means that we cannot enter a row into the animals table unless that animal is associated with a client we already have in the client tables. (That is how the vet knows who is going to pay the bills- a good business rule.)

An inner join between clients and animals returns rows for clients who have animals. If a client does not have any animals currently(that means there is no row in the animals table with that cl_id), that client is not returned. It is also true that with the inner join an animal also has to have a client-but that rule is already built into the animals table.

If we do an outer join using `clients left join animals`, then the table expression includes clients with animals and clients without animals. The table expression includes a row for each valid combination of client and animal- if a client has 4 animals, that client gets 4 rows in the table expression. And the table expression includes one row for each client who has no animals. (The client is the preserved table.) One way to think of an outer join from the clients table to the animals tables is as a two step process- these two steps are done for us when you run the query.

First we get the clients with animals. Then the dbms will add in rows for the clients with no animals- clients where there is no matching row in the animals table. What should we display for the an_type column if the client has no animals? We might want to display a message- but what the dbms does for those rows is return a null in that attribute. The null is what SQL commonly uses for a situation where there is no data.

Now consider the opposite join- `animals left join clients`. The table expression includes animals with clients and animals without a client. We have animals with clients, but the rules for the animal table says that we do not have any animals without a client. So in this case the table expression is just the same as an inner join- clients with animals. You should not write the join animals left join clients, since an outer join may be less efficient than an inner join and with our tables the result will always be the inner join.

Sometimes people write outer join and then include a Where clause filter that eliminates all of the null-added rows, ending up with an inner join. That is also not efficient and needs to be avoided.

Use outer joins when you need them, but only when you need them.