Table of Contents

1. What is an outer join? 1
2. Tables for this demo 2
3. Join Types 2
3.1. INNER JOIN 2
3.2. LEFT JOIN 2
3.3. RIGHT JOIN 3
3.4. FULL JOIN 3

1. What is an outer join?

In a previous unit, we discussed inner joins. We use inner joins to bring data together from two or more tables. In the vets database, we might want to see information about a client and about the client's animals. We have a table for clients and another table for animals. The animals table includes an attribute cl_id which refers back to the client table and that is how we do a join.

With an inner join between two tables, we see data from both tables only when there are matching rows. If we do an inner join between the clients table and the animals table using the cl_id attribute, we will see only rows for clients who have animals. If we do an inner join between the animals table and the exam headers table using the an_id attribute, we will see only rows for animals that have exams. That is the way that an inner join works. 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. We might want a list of animals that include both animals that have exams and animals that do not have exams. 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.

One way to think of an outer join from the animals table to the exam tables is as a two step process- these two steps are done for us when you run the query.

First we get the animals with exams- these are a few of those rows. Some animals have one exam, some animals have multiple exams.

```
+-------+-----------------+---------------------+
| an_id | an_name         | ex_date             |
+-------+-----------------+---------------------+
| 15001 | Big Mike        | 2011-12-09 09:00:00 |
| 15001 | Big Mike        | 2011-12-22 09:00:00 |
| 15002 | George          | 2012-06-13 10:45:00 |
| 15401 | Pinkie          | 2012-06-06 10:30:00 |
| 16002 | Fritz           | 2011-12-23 12:15:00 |
| 16002 | Fritz           | 2012-01-03 14:30:00 |
| 16002 | Fritz           | 2012-01-06 10:45:00 |
| 16002 | Fritz           | 2012-01-16 09:15:00 |
| 16002 | Fritz           | 2012-02-03 14:30:00 |
```

Then the dbms will add in rows for the animals with no exams- for animals where there is no matching row in the exam_headers table. What should we display for the ex_date column if the animal has no exams. 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. These are a few of the rows including the added null rows. Now we get at least one row for every animal in the animals table.

```
+-------+-----------------+---------------------+
| an_id | an_name         | ex_date             |
+-------+-----------------+---------------------+
| 10002 | Gutsy           | NULL                |
| 11015 | Kenny           | NULL                |
| 11025 | NULL            | NULL                |
| 11029 | NULL            | NULL                |
| 12035 | Mr Peanut       | NULL                |
| 12038 | Gutsy           | NULL                |
| 15001 | Big Mike        | 2011-12-09 09:00:00 |
| 15001 | Big Mike        | 2011-12-22 09:00:00 |
```

```
| 15002 | George          | 2012-06-13 10:45:00 |
| 15165 | Burgess         | NULL                |
| 15401 | Pinkie          | 2012-06-06 10:30:00 |
| 16002 | Fritz           | 2011-12-23 12:15:00 |
| 16002 | Fritz           | 2012-01-03 14:30:00 |
| 16002 | Fritz           | 2012-01-06 10:45:00 |
| 16002 | Fritz           | 2012-01-16 09:15:00 |
| 16002 | Fritz           | 2012-02-03 14:30:00 |
```

If we have an animal with no exams- such as an_id 10002, the row has a null for the ex_date.

If we have an animal with exactly one exam- such as an_id 15002, the row has a value for the ex_date.

If we have an animal with multiple exams- such as an_id 16002, there will be multiple rows with values for the ex_date.

Note that in the exam_headers table, the ex_date attribute cannot be null but in the result set returned by the outer join the ex_date column can be nulled.

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.

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	
6	Perkins	

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

3. Join Types

We will start with 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.

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

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
```

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. We also get one row for the Marketing department which has no employees; this row has nulls filled in for the missing data from the employee side of the join.

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

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.

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

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.

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

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.

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