Table of Contents

About this time in the semester, people realize that they need to understand joins and subqueries better. The rest of the semester assumes that you know the difference between inner and outer joins; how to write subqueries that perform much of the work of an inner or outer join; how to write subqueries to use in filters.

And a bit more practice and what is mostly a repeat of previous explanations can help. There is no assignment based directly on this and therefore some people won't read this- but it would be a good idea to think about this before the exam.

This is going to use just two tables: clients and animals.

The database rules - set up in the tables- include the following

- Every client has a unique value for client id in the clients table
- Every animal has a unique value for animal id in the animals table
- Every animal has a non-null value for client id in the animals table; this is not unique since a client can have more than one animal. This also means that we do not have any animals without a value for a valid client id.
- We can have clients with no animals.
- Every animal has a non-null value for animal type in the animals table

1. Types of Joins

First you need to understand the two most common basic join types_ inner joins and outer joins.

With an **inner join** between two tables, the result set only contains rows where there was a match between the two tables. Since we are talking about clients and animals, the match will be on the client id value in the animals table matching the client id in the clients table. With an inner join, we get back rows on clients and the client's animals. If a client does not have an animal, that client is not in the result set of an inner join.

If we do that inner join and display the client id, the animal id and the animal type, those columns will never contain a null because every client has a client id, every animal has an animal id and a not null value for cl_id and an_type from the animals table.

Therefore it makes no sense with this inner join to test if the an_id is not null or if an_type is not null- they are never not null in this join with these tables.

With an **outer join** between two tables, the result set will contains rows where there was a match between the two tables and also rows that are based on one of the table even if there are no matching rows in the second table.

If we do vt_clients left join vt_animals, then we get a row for each client- animal match and we get a single row for each client that has no animals. In that case the columns for the data from the animals tables will be null for clients with no animals (because they do not have an animal).

If we do vt_clients right join vt_animals, then we get a row for each client- animal match and we get a single row for each animal that has no associated client. But we do not have any animals without a client so in this case that join simply becomes an inner join. (But you have made the system do more work to figure that out.)

2. Unmatched queries

These are used to find rows in one table that do not have an associated row in another table. Such as client with no animals.

For that you use vt_clients left join vt_animals and test that the value for an_id is null. You could also test that an_type is null but that is not as efficient (it is easier for the dbms to test the pk than another column) and is just makes more sense. We use an_id to identify an animal, so a null an_id means there is no animal.

3. Using subqueries

You can use a subquery to do the work of a join for many queries. If I just want to see data from the clients table, I can use a subquery to see clients with animals and I can use a subquery to see clients without animals. Note that the subquery for these tasks is not testing anything about the animal. We just want to know if that cl_id value exists in the animals table which would mean that client has at least one animal.

```
select cl_id, cl_name_last
from vt_clients
where cl_id in (
    select cl_id
    from vt_animals)
;

select cl_id, cl_name_last
from vt_clients
where cl_id not in (
    select cl_id
    from vt_animals)
;
```

If I want to select both client data and animal data, then this subquery does not work. I can only display data from the tables listed in the main query's From clause.

4. More complex tests

These are some of the rows returned by an inner join between clients and animals.

Client 1825 has several animals. Each of these animals gets its own row and there is one value for an_type for each row. Client 6897 has only one animal and that animal gets a row in this result set. A client with no animals does not appear in this result set- this is from an inner join.

```
      cl_id       an_id an_type
----------- ----------- -------------------------
       1825       16002 porcupine
       1825       16003 cat
       1825       21002 hedgehog
       1825       21005 dormouse
       1825       21006 hamster
       1852       21007 snake
…
       6897       16004 cat
       7152       17026 lizard
       7152       17027 lizard
```

Some tasks might ask you to display clients who have a snake or a cat. That uses a simple filter Where an_type in ('cat', 'snake'). We can determine that a row is in the result set by looking at a single row. (We cannot tell that a client **is not** in the result set by looking at one row, but we can tell that the client **is** in the result set by looking at one row.) SQL does its tests on a row by row basis. It looks at one row, does the test and decides if that row gets into the result. If we look at the first row

```
       1825       16002 porcupine
```

We cannot tell if client 1825 will be in the final result. But we can tell this when we look at the next row.

```
       1825       16003 cat
```

Based on the above rows, we would get the rows shown here

```
      cl_id       an_id an_type
----------- ----------- -------------------------
       1825       16003 cat
       1852       21007 snake
       6897       16004 cat
```

And if we displayed only the client id and used distinct, we would get only two rows.

```
      cl_id
----------- -
       1825
       6897
```

But if we asked who has **both** a cat and a snake- that is a different type of question, Since we look at one row at a time, there is no way to do this with a simple filter. So the query becomes more complex. We will find several ways to do this. For now just stick with this approach.

This uses a subquery to find all clients with an animal- any type of animal

```
  select CL.cl_id
  from vt_clients  CL
  where
    CL.cl_id in (
     select  AN.cl_id
     from vt_animals AN
     );
```

This filters the subquery so it only returns client ids if the client has a snake. So this returns clients who have a snake.

```
  select CL.cl_id
  from vt_clients  CL
  where
    CL.cl_id in (
     select  AN.cl_id
     from vt_animals AN
     where AN.an_type = 'snake'
     );
```

Now you can add another test to check that that client id is also in the set of clients who have a cat.

```
  select CL.cl_id
  from vt_clients  CL
  where
    CL.cl_id in (
     select  AN.cl_id
     from vt_animals AN
     where AN.an_type = 'snake'
     )
  and
    CL.cl_id in (
     select  AN.cl_id
     from vt_animals AN
     where AN.an_type = 'cat'
     );
```

Now you can add another test to check that that client id is **not** in the set of clients who have a cat.

```
  select CL.cl_id
  from vt_clients  CL
  where
    CL.cl_id in (
     select  AN.cl_id
     from vt_animals AN
     where AN.an_type = 'snake'
     )
  and
    CL.cl_id NOT in (
     select  AN.cl_id
     from vt_animals AN
     where AN.an_type = 'cat'
     );
```

Note that these queries are *not* doing an inner join and then a set of AND tests - that approach does not solve these questions because that would look for a row that has the value 'snake' and 'cat' the same time ; remember the inner join between clients and animals returns one row per animal and an animal has only one value for an_type.

This is not particularly efficient but it is flexible. What do the following return?

```
select CL.cl_id
from vt_clients  CL
where
  CL.cl_id in (
   select  AN.cl_id
   from vt_animals AN
   where AN.an_type = 'hamster'
   )
and
  CL.cl_id   in (
   select  AN.cl_id
   from vt_animals AN
   where AN.an_type = 'dormouse'
   )
and
  CL.cl_id   in (
   select  AN.cl_id
   from vt_animals AN
   where AN.an_type = 'hedgehog'
   )
```

```
select CL.cl_id
from vt_clients  CL
where
  CL.cl_id in (
   select  AN.cl_id
   from vt_animals AN
   where AN.an_type = 'hamster'
   )
and
  CL.cl_id  not  in (
   select  AN.cl_id
   from vt_animals AN
   where AN.an_type = 'dormouse'
   )
and
  CL.cl_id not  in (
   select  AN.cl_id
   from vt_animals AN
   where AN.an_type = 'hedgehog'
   )
```