Table of Contents

I showed you the syntax of the insert statement in unit 01.What I want to discuss here is something of the logic on inserting rows into a table. Your first question might be why would you want to do that since I provide the inserts for all of the demo and assignment tables.

1. Testing your query

One time that you might want to do inserts is to test your query. Suppose I ask you to write a query that finds all of the dogs in the animals table that have the name 'Buddy'. We have not discussed all of the following row filters, but you should be able to see that this query does that task.

```
select *
from a_vets.vt_animals
where an_type = 'dog' and an_name = 'Buddy';
```

But if you run this query, the result set is empty; we do not currently have any dogs named Buddy. But that does not give you any reassurance that the query is correct. There are a lot of queries that you could write that return an empty result set.

So you might want to add a row to the animals table that adds a dog named Buddy. Following the model in the inserts script, you might try the following:

```
insert into a_vets.vt_animals ( an_id, cl_id, an_name, an_type, an_dob)
  values(15165, 42, 'Buddy', 'dog', '2007-07-07');
```

The syntax is valid, but that insert won't run. If you review the document on the vets database you can detect two errors in the insert.

1) The column an_id is the primary key for this table and we already have an animal row using the an_id value 15165. So we would need to use a value for an_id that does not match any of our current rows.
2) The column cl_id is the client id and it is the foreign key to the clients table, so that value has to match a client we have in the clients table.

We could do the following insert:

```
insert into a_vets.vt_animals ( an_id, cl_id, an_name, an_type, an_dob)
  values(15, 5689, 'Buddy', 'dog', '2007-07-07' );
```

Now if we run the first query again- to find dogs named Buddy, we get our new row in the result set.

1.1. Gotchas for testing your query

One thing that can get confusing is that you cannot prove your query is correct by examining the result set. For example, I could write the following query- which returns the same results set but which is not a vaid query to find dogs named Buddy.

```
select *
from a_vets.vt_animals
where an_dob = '2007-07-07';
```

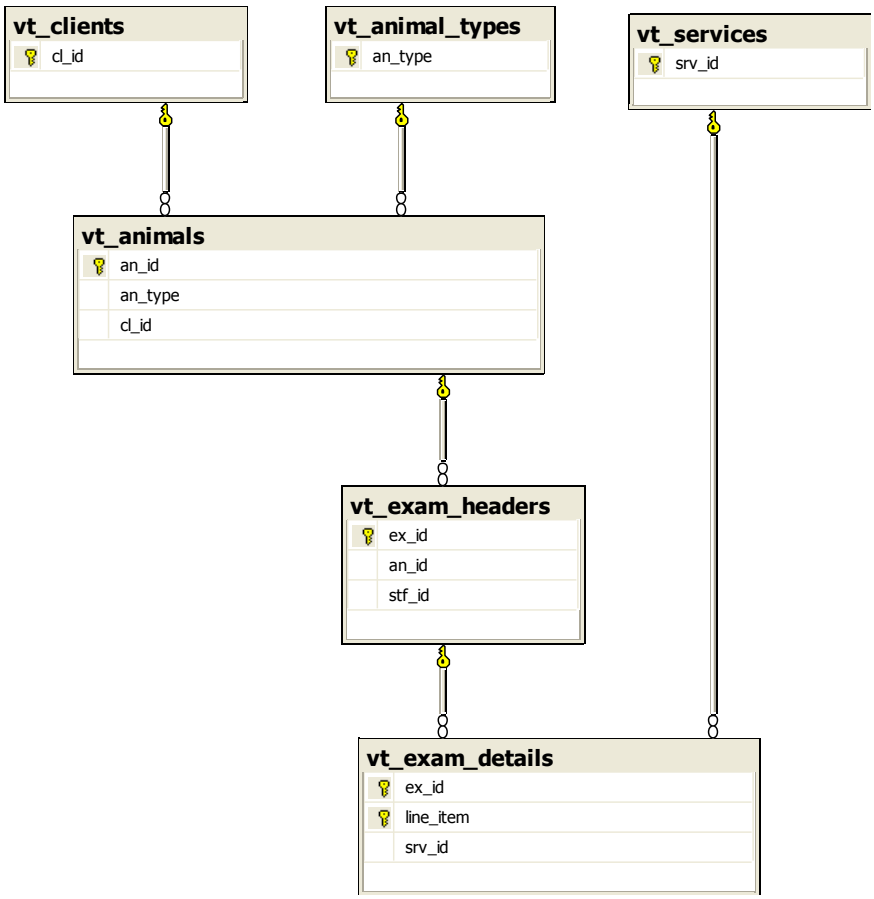
You can demonstrate that your query is wrong if it returns an incorrect result set- if your query returns cats then your query is incorrect. But you cannot prove your query is correct by examining the result set.

But it can help to add test rows.

1.1. Parent and child rows

The following is a diagram of some of the tables in the vets databases (with slightly different names) that show the tables and their PK and FK columns. Note that the tables for clients, animal types and services have no FK columns. You can add columns to these tables without worrying about FK- without worrying about the parent table rows.

But if you add a row to the animals table, you need to worry about the an_type (it needs to match a row in the animal_types table) and about the cl-id (which needs to match a row in the cleitns table).



Suppose you want to insert a row in the exam_details table; in that case you need to have FK matches for the other 5 tables shown. The srv_id needs to match a srv_id in the services table; the ex_id needs to match a row in the exam_headers table. The an_id value in the exam_headers table needs to match an an_id in the animals table. The cl_id value in the animals table needs to match an cl_id in the clients table. The an_type value in the animals table needs to match an an_type in the animal_types table.

You could either build on the existing rows to add additional rows to the tables, starting with the insert for the parent before the insert for the child.

2. Resetting the tables

When you run the assignment script to turn in, you need to use the original inserts that I provided- this should not include any test data that you inserted, You did the test inserts to test your queries- but not for grading.

If you did a simple one row add- such as in the start of the discussion, you could do a simple delete based on the pk.

```
delete from a_vets.vt_animals where an_id = 15;
```

But if you have added several rows, it can be quicker to simply rerun the inserts script before you run the assignment script for grading.