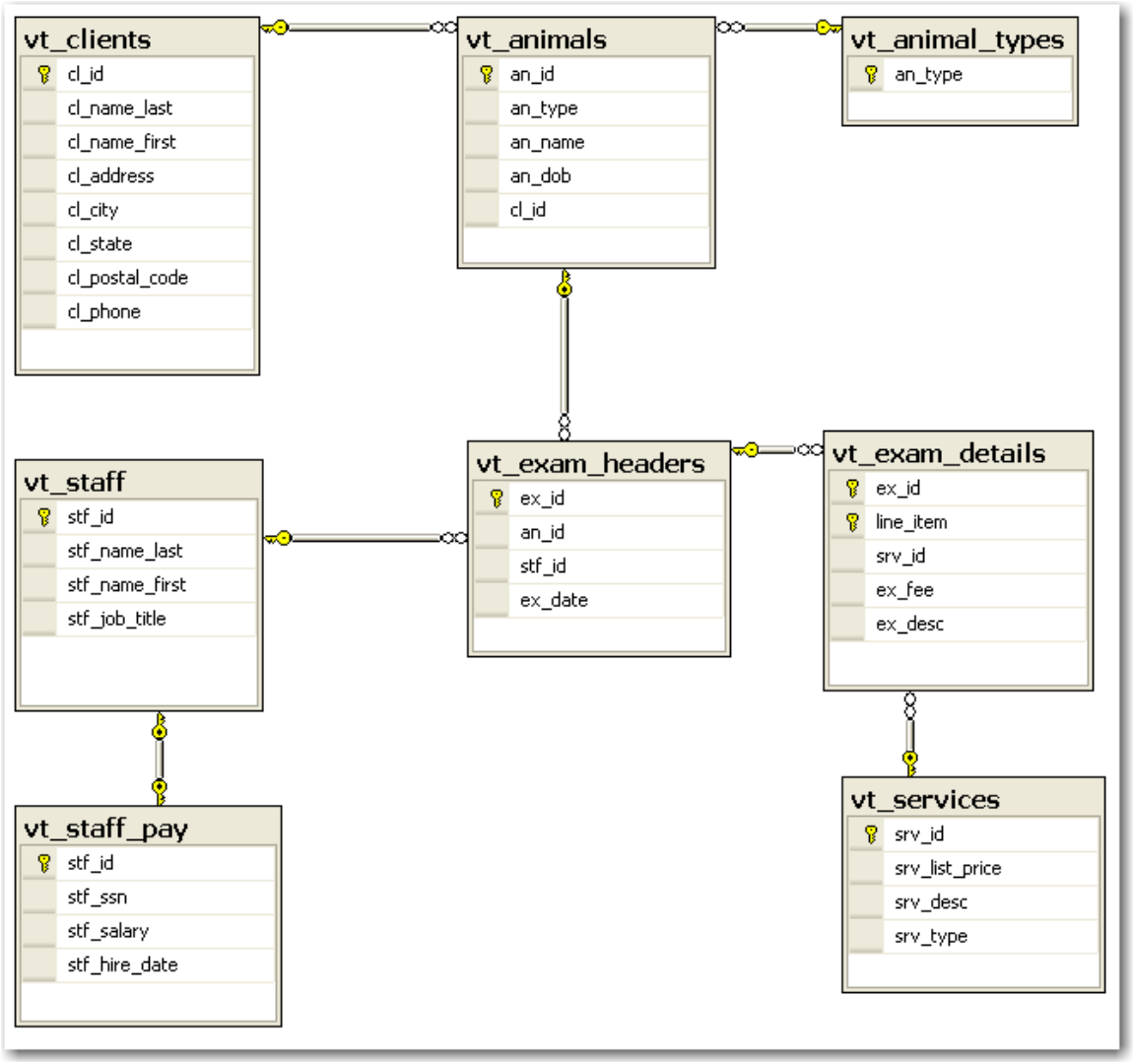
Table of Contents

1. Tables in the vets database

Starting with Assignment 2, you will need to use tables from the vets series of tables. With our Oracle accounts, all of our tables are stored in the same schema so I use a prefix of vt_ for all of the tables for this series of tables.

The following is a graphical display of the tables and how they are related. This diagram was set up in SQL Server. (It does very nice diagrams.)

For assignment 2 and 3, each query deals with a single table.



2. Business Rules:

- Each animal seen at the Vet clinic is the responsibility of a single client.
- We need name and address information for the clients.
- The vets treat only certain type of animals.
- Some animal types are categorized as reptiles; these are: snake, chelonian, crocodilian, lizard.
  Some animal types are categorized as rodents; these are: hamster, capybara, porcupine , dormouse.
- For the staff at the clinic, there are two staff tables- one is vt_staff with their public information and a second table, vt_staff_pay, with information that would be more private.
- We have a list of the various services that the clinic can provide and we have a list price for each service.
- When an animal comes in for an exam, we keep track of some of the exam data in this database- primarily the data that might be used for billing. The exam_details table includes the actual amount the client was charged for services.
- This database does not handle payment of the bills.

For this database you may assume that each exam_header row has at least one associated exam detail row. The database does not enforce that rule.

3. SQL Discussion:

This discussion will take you through some of the details in a Create Table statement. You do not need to remember all of these details yet but you should be able to understand these. It will help you understand the tables if you read through the Create statements.

Note that when you actually create these tables, you need to follow the order of statements in the script. For example, the client table has to be created before the animals table.

The best way to do this is to run the script that I have provided- do not try to copy and create the table one table at a time.

3.1. vt_animals

Let's start with the animal table. For each animal, we want to know what type of animal it is- a cat, dog, lizard etc. We also want to know its name- but we do not insist that it has a name. We want a date of birth for each animal to calculate its age. If necessary, the vet will estimate this date. Since we are storing data for a veterinary clinic, we need to know who will pay the vet bills and make the treatment decisions- so for each animal we will also store the id of the client responsible for this animal. And we need an identifier for the animals.

This is the create table statement for the animals table. I added line numbers to make this easier to discuss but the line numbers are not part of the SQL.

```
1.   create table vt_animals(
2.     an_id           number(6,0)  constraint vt_animals_pk primary key
3.   , an_type         varchar2(25) not null
4.   , an_name         varchar2(25) null
5.   , an_dob          date         not null
6.   , cl_id           number(6,0)  not null
7.   , constraint vt_animals_animal_types_fk
          foreign key(an_type)  references vt_animal_types
8.   , constraint vt_animals_clients_fk
          foreign key (cl_id) references vt_clients
9.   , constraint an_id_range check (an_id >0)
10.  ,  constraint an_dob_ck check (trunc(an_dob) = an_dob)
11.  );
```

The statement starts by saying we want to create a table and supplies a name for the table (line 1)

The statement then lists the various attribute names and their data type (lines 2-6). " number(6,0) " is an integer number with up to 6 digits; "varchar2" is used for character data and includes a maximum length- in this case the animal name must be no more than 25 characters long; "date" is used for date values.

We also indicate if a value is required to be entered for each attribute. "not null" means that we have to supply a value and "null" means that the value is optional.

Then we have some constraints defined for the various attributes.

The phrase "primary key" on line 2 is used to indicate that this column- an_id- is the identifier for the table. Designating a column as a primary key means that it cannot be null and it must be unique in this table. This means that every animal has an an_id value and every animal has a different value for the an_id. It also means that we can use this value to identify the animal in another table- such as the exam headers table.

There are two "foreign key … references" phrases.

```
1. , constraint vt_animals_animal_types_fk
        foreign key(an_type)  references vt_animal_types
2.   , constraint vt_animals_clients_fk
          foreign key (cl_id) references vt_clients
```

The vet is not willing to treat all types of animals- for example, our vet will not treat fish. So we have a table vt_animal_types that includes the allowed values for an_type. The references phrase in the animals tables enforces the rule that any value entered for the an_type column must be a value in the vt_animal_types table. If the vet later decides to treat fish, we can add that value to the vt_animal_types table. This has a second purpose in that we can restrict the spelling of the values entered- so a dog gets the value "dog"- not "dogs", "puppy", "canine", "poodle" or other terms. This helps keep our data more organized.

The other references clause is used to be certain that each animal does have an associated client. This also means that you have to create tables in a specific order.

Line 9 adds a check that the an_id values have to be positive values.

I decided that the animal dob value should not have a time components. For the purpose of the database I might want to know that the animal was born on April 12, 2011but I do not care to record that the animal was born at 3:05 a.m. We will see the trunc function later, but this rule says that all dob values are recorded with a time component of midnight.

Now let's think about bringing an animal into the vet clinic for an exam. We will need to have data for the animal and its client in our tables- this is sometimes called registering the animal and client. The vet will keep some information about the exam- such as the date and time of the exam, which staff person did the exam and what was done for the animal. These tables are not keeping all of the exam details but it is quite likely that an exam will include more than one treatment- and more than one billing item.

We want to keep track of the various treatments the animal has on an exam and a standard way of dealing with this is to have two tables: one table for the exam includes those attributes that occur only once per exam (the animal identifier and the date) and another table that can have one row for each treatment provided. Then we need to associate those two tables.

3.2. vt_exam_headers and vt_exam_details

This is the Create Table statement for the exam header table which stores the information for the exam. We need an identifier for the exam (line 2) and each exam is for one animal (line 3) - that uses the an_id which must refer to an animal that we have in the animal table (line 6). We also have a value for the staff person who did the exam which refers to a person in the staff table (line 7). The exam date is stored as a date field and includes a time component. The exam date has to be no earlier than Jan 1, 2010- the date that the clinic opened (line 9). All of these fields need to have an entry (not null).

```
1. create table vt_exam_headers(
2.     ex_id           number(6,0)  constraint vt_exam_headers_pk primary key
3.   , an_id           number(6,0)  not null
```

```
4.    , stf_id          number(6,0)  not null
5.    , ex_date         date         not null
6.    , constraint vt_exam_headers_animal_fk
         foreign key(an_id) references vt_animals
7.    , constraint vt_exam_headers_staff_fk
         foreign key(stf_id) references vt_staff
8.    , constraint ex_id_range check (ex_id >0)
9.    , constraint exam_date_range  check (ex_date > date '2010-01-01')
10.   );
```

This is the Create Table statement for the exam detail table which stores the detailed information for the exam. You should identify the rules in the create table statement as you read this discussion.

We have one row in this table for each treatment that was done during an exam. So we need to be able to associate a row in the table with the exam it belongs to; this is done by including the exam identifier in this table and referencing that attribute to the exam header table.

But now we cannot make that attribute the primary key for this table since we will generally have multiple rows for each exam. In this table design we are including a second attribute - the line_item which is part of the primary key. We expect that the first item for an exam will be line_item 1 and the second line_item 2- but the design does not enforce that rule. We are declaring the combination of these two attributes (ex_id and line_item) to be the primary key (line 7)- that the values exist and that the pair of values (ex_id, line_item) will be unique in the table.

We include the treatment identifier- the service id- in the detail table and associate it with the services table. The exam detail table includes an attribute for actual fee charged. This could be the same as the list price in the services table- but the vet could choose to charge less than the list fee- or more. The table includes a check rule that the value for the ex_fee must be a non-negative value.

```
1. create table vt_exam_details(
2.      ex_id           number(6,0)  not null
3.    , line_item       number(6,0)  not null
4.    , srv_id          number(6,0)  not null
5.    , ex_fee          number(6,2)  not null
6.    , ex_desc         varchar2(50) not null
7.    , constraint evt_exam_details_pk primary key (ex_id, line_item )
8.    , constraint evt_exam_details_headers_fk
         foreign key (ex_id) references vt_exam_headers (ex_id)
9.    , constraint evt_exam_details_services_fk
         foreign key (srv_id) references vt_services (srv_id)
10.   , constraint line_item_range check (line_item >0)
11.   , constraint ex_fee_range   check (ex_fee  >= 0)
12.);
```

This shows two possible rows from the exam headers tables and the related rows from the exam details table.

Exam Headers

EX_ID	AN_ID	STF_ID	EX_DATE
1982	21001	15	23-OCT-11
1988	15001	25	09-NOV-11

Exam Details

EX_ID	LINE_ITEM	SRV_ID	EX_FEE	EX_DESC
1982	1	523	60	yearly checkup
1982	2	110	50	(null)
1988	1	104	30	check up
1988	2	1002	25	(null)
1988	3	687	45	shell softening

3.3. vt_services :

The Create Table statement for the services has two additional rules: There is a column srv_type and we want to restrict the values to 4 possible values. In this table this is done with a check constraint which lists the allowed values (line 9). This is a different technique than was used for the animal types and is useful when we have a short list of possible values that is not apt to change frequently. You can see this technique also used in the staff table. The second rule is that the values in srv_desc must all be different; it would be confusing to have two rows for a service which have the same name; this is done with a **Unique** constraint.

```
1. create table vt_services(
2.     srv_id          number(6,0)
3.          constraint vt_services_pk primary key
4.   , srv_list_price  number(6,2)  not null
5.   , srv_desc        varchar2(50) not null Unique
6.   , srv_type        varchar2(25) not null
7.   , constraint srv_id_range check (srv_id >0)
8.   , constraint srv_list_price_range  check (srv_list_price >= 0)
9.   , constraint srv_type_values
              check (srv_type in ('office visit', 'consult', 'medicine', 'treatment'))
10.);
```

3.4. Inserting data

Because our tables are related to each other, the inserts have to be done in a specific order. Suppose you are a new client to the vet clinic and you are bringing in your dog "mittens" for a checkup. The person registering you will need to store your client information first in the vt_clients table before they can store the animal's information. This is because in the vt_animals table, there is a required cl_id value that has to match a cl_id value in the client's table. And the exam data cannot be stored until there is a related animal row. The exam header row needs to be entered before the exam detail rows.

The clinic would be using an application that handles these details and the person at the registration desk would not need to know those details. But when we enter the data using our inserts statements, we have to insert these in the proper order. The inserts script uses the following order for the inserts

1. staff
2. staff_pay
3. services
4. animal types
5. clients
6. animals
7. exam headers
8. exam details

Take a look at the delete statements at the top of the script and you can see that these delete statements are written in the opposite order. This is because the way the database is set up, we cannot delete a parent row if there are related child rows. For example, we cannot delete a client if the client has animals in the animals table. In our database we do not want to store data about animals for which we do not have a client to pay the bills!

3.5. Commit

The script with the inserts ends with a commit statement. In Oracle we use this statement at the end of the set of Inserts to tell the Oracle dbms to save these new rows in a persistent form.