**Databases Assignment**

**Pet Shop Case Study**

**Daragh Browne - C15396706**

**Dmiter Dinkov- C15334276**

**Naif Alamoudi - D14127025**

**Thought Process for ERD**

When creating the ERD we put ourselves in the shoes of Peter and his pet shop employees. We attempted to make it as realistic and efficient as possible for keeping track of important data for the shop.

As a group, we landed on the consensus that we would need 11 tables.

The Species table contains all the information about all the species that you can buy in Pete’s pet shop.It hold information like the species type,the species price,the instructions for handling the pet which are given to the customer that purchases an animal of that species and the cage size the animal must be kept in. In this table you cannot have more than one of the same type of species.

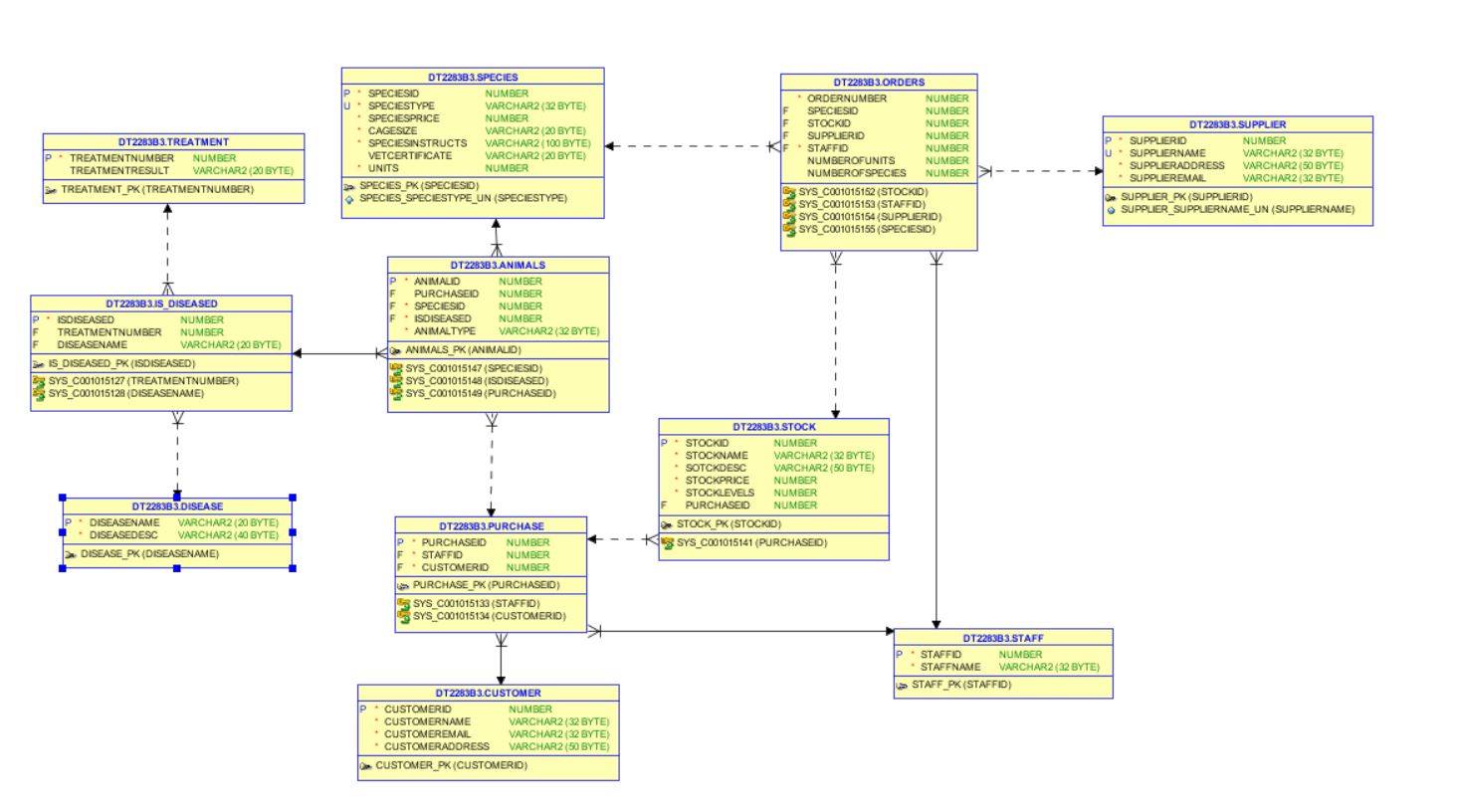
The species table extends a one to many relationship with the animals table since you can have more than one animal of the same species, but you cannot have one animal of many species. The animal table contains all the information about all the animals that are contained in the shop. The information in the animal table is the animal type, the purchase ID(which if is null means that this animal hasn’t been purchased), isDiseased (which if it is 0 then it means that the animal has no diseases, but if its greater than 0 the animal has is diseased) and a few other information fields.

The animal table has a many to one relationship with the is\_diseased table, because you can have an animal which is either diseased or not, but you can’t have an animal which is diseased and not diseased at the same time. The is\_diseased table has a many to one relationship with the treatment table and the disease table.The is\_diseased table has an id a treatment id(which if null means that no treatment has been applied to the animal yet) and a disease name. The treatment table has a treatment number and a treatment result which holds information about the outcome of the treatment(wether it was positive or negative). The disease table holds a list of all the known diseases of animals. It has a disease name column and a disease description column.

The supplier table holds all the information about a supplier. It extends a one to many relationship with the orders table. The orders table in turn extends a many to one relationship with the species and stock tables so if an order is placed for either more units of a specific stock or a specific ID, then a the corresponding foreign key id is filled in for either the species or the stock in the orders table. If an order is placed for only species or only stock then the other foreign key is just left as null.

The staff table contains all the information about a staff member. The staff table extends a one to many relationship to both the order and purchase tables since a staff can handle multiple orders or purchases.

Finally the purchase table extends a many to one relationship to the customer table as a customer can make more than one purchase, but a purchase cannot have more than one customers. The customer table a list of information about customers.



**Thought process of designing Transactions (PL/SQL)**

When creating my Pl/Sql I choose to make a function that would allow a user to enter in a disease into the list of all diseases. It asks the user to enter a disease name and then a disease description. The function then inserts the information the user entered into the disease table.

**Thought process for the sql queries:**

1.SELECT \* FROM ANIMALS;   
This querie displays all the information from the animals table and outputs it to the user.  
  
 2. SELECT stockPrice,stockName from stock;  
This querie shows the use of projection by selecting the stock price and stock name from the stock table.  
3. SELECT SUM(speciesPrice) as Total from species  
join animals using(speciesID)  
join purchase using(animalID)  
where speciesPrice >10 ;  
  
  
The following querie demonstrates aggregation with filters by getting the sum of all species sold which cost more than 10.  
  
4. SELECT speciestype,speciesId from species  
where speciesID =1004  
UNION  
SELECT animalType, speciesID from animals  
where speciesID = 1004;  
  
This querie demonstrates union by displaying all the information from species and the animals table which have a speciesID of 1004.  
  
4. SELECT orderNumber FROM orders  
JOIN stock using(stockID)  
JOIN species using(speciesID)  
where stockPrice > 1 and speciesPrice > 100  
minus  
select supplierID from supplier  
where supplierID = 101;  
  
Demonstrates minus. Selects all the order numbers from the order table that have a stock price greater than one and a species price greater than 100 , but don’t have a supplier ID of 101.  
  
5. SELECT \* FROM SPECIES  
WHERE speciesPrice <100  
intersect  
SELECT \* from species  
where cageSize = 'Medium';  
  
Select all the information from species for species which have a price over 100 and need a medium cage size. Demonstrates the use of Intersects.  
6. SELECT \* from animals  
inner join species using(speciesID)  
where speciesPrice >20;  
The following querie shows the use of inner joins. All the information is displayed from animals (join species table) which have a species price greater than 20.  
  
7. SELECT animalType,purchaseID from animals  
FULL OUTER JOIN purchase using(purchaseID);  
Select the all the animal from the animal table and all the purchaseID from the purchase table, demonstrating the use of full outer joins.  
  
8. SELECT animalType from animals  
where purchaseID in   
(select purchaseID from purchase);  
  
Select all the animal types from the animal table which have not been bought. Demonstrates semi-joins.  
  
9. SELECT animalType from animals  
where NOT EXISTS(  
SELECT purchaseID from purchase  
where animals.purchaseID = purchase.purchaseID);  
Selects all animals which are not bought. Demonstrates anti-joins.  
  
10. SELECT \* from purchase  
where customerID=(  
Select customerID from customer  
where customerName = 'Mary Black');  
  
Return all the purchases of Mary Black . Demonstrating the use of correlational queries.