

COM1025 Web and Database Systems

Coursework Assignment

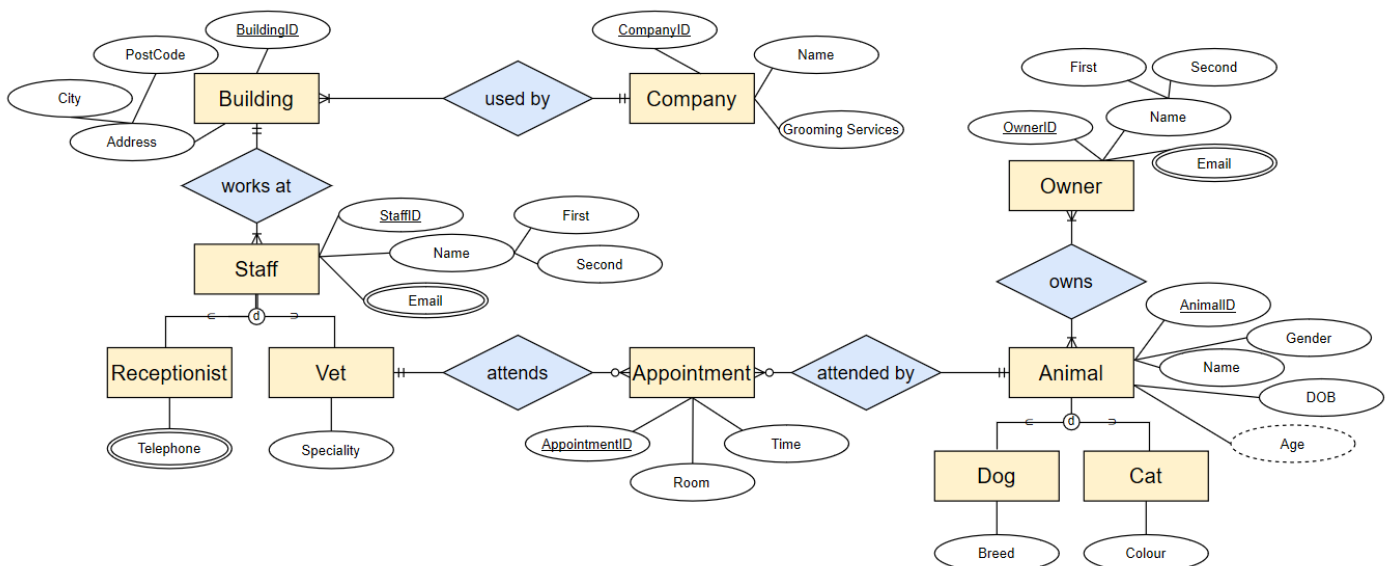
13/01/2020

I.hay

1 Developing and Testing Environment

EER Diagram Drawing Software – draw.io**MySQL Server Version** – XAMPP v7.4.1 (used to create and test MySQL Database)**MySQL Client-Side Tool** – mysql.exe (used to test MySQL Database)**Web Server and PHP Model** – XAMPP v7.4.1/ PHP v7.4.1 (used to test website)**Web Browsers** – Google Chrome v79.0.3945.88, Microsoft Edge v 44.17763.831.0**Operating System** – Windows 10 Home (used during development and testing)

2 EER Data Model and Diagram



The EER Data Model and Diagram I created is based off a database for storing information on vet clinics. I drew this model using the software draw.io. The 10 entities are represented by rectangles, the 5 binary relationships are represented by diamonds and the attributes of each entity are represented by ovals connected to their appropriate entity by a single straight line (edges). I colour coded the diagram to make it easier to interpret when developing the schema and database. This EER Model is justified as it uses the notations taught in this module and includes all required entities, attributes and relationships,

Other Attributes Types:

- Composite Attributes – can be broken down into smaller attributes e.g. Name (in Owner and Staff) and Address (in Building). Represented in my diagram by multiple attributes connected to one attribute which is connected to the entity.
- Derived Attributes – value derived from other attributes e.g. Age (in Animal). Represented in my diagram by broken line oval connected to entity.
- Multivalued Attributes – can have multiple values e.g. Email (in Owner and Staff) and Telephone (in Receptionist). Represented by double lined oval.
- Identifier Attribute – selected to uniquely identify entity. Represented by underlining text in oval e.g. CompanyID (in Company).

Binary Relationships:

- Building(mandatory) used by Company (mandatory) = *many to one cardinality ratio*
- Staff(mandatory) works at Building(mandatory) = *many to one cardinality ratio*
- Vet(mandatory) attends Appointment(optional) = *one to many cardinality ratio*
- Appointment(optional) attended by Animal(mandatory) = *many to one cardinality ratio*
- Owner(mandatory) owns Animal(mandatory) = *many to many cardinality ratio*

Mandatory = total participation

Optional = partial participation

Supertype-Subtype Relationships:

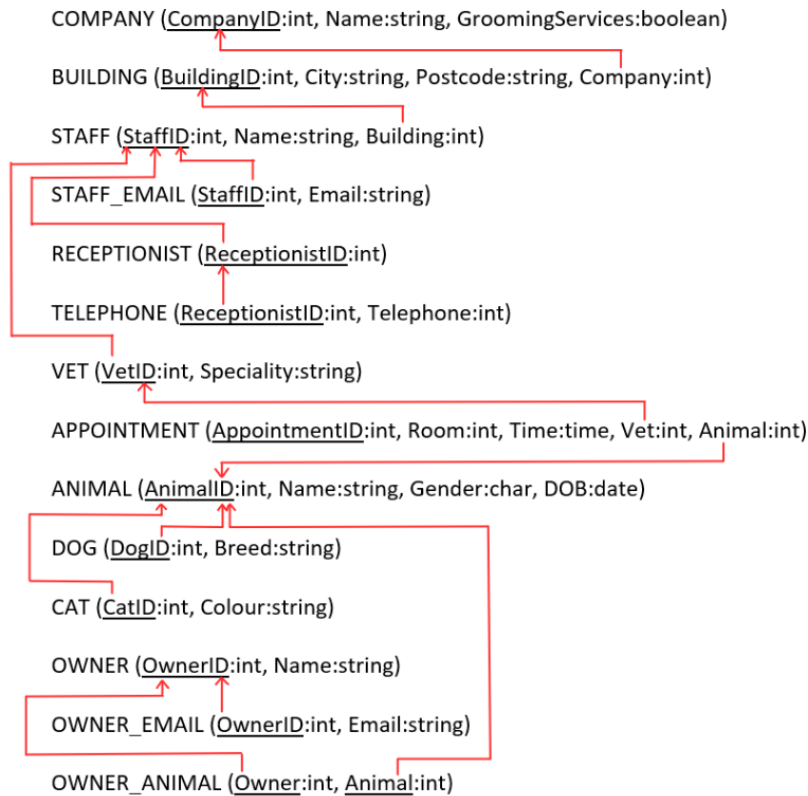
1. Receptionist + Vet Subtypes of Staff Supertype-
 - Disjoint constraint so no overlapping (entity can only be one subtype)
 - Total participation so every Staff entity must be one of its subtypes
2. Dog + Cat Subtypes of Animal Supertype-
 - Disjoint constraint so no overlapping (entity can only be one subtype)
 - Partial participation so Staff entities do not need to be one of its subtypes

Self-Assessment:

I have completed all the main requirements for this section since my EER Model has at least one supertype with two subtypes and constraints (Animal supertype of Dog and Cat). It has multiple additional entity types and all entity types have attributes. There is a many to many and multiple one to many relationships all marked with crows foot notation.

3 Conceptual/Logical Relational Database Schema

To convert the EER diagram I created into a conceptual relational database schema I had to convert multi-valued attributes into a new relation with two attributes including a foreign key referring to the entity type relation which is also the primary key of the new relationship e.g. STAFF_EMAIL, OWNER_EMAIL and TELEPHONE. I then removed derived attributes. I also had to create a new relation OWNER_ANIMAL for the many to many relationship so there are not multi-valued attributes in both relations; this created two new one to many relationships. For all the one to many relationships I had to put a foreign key on the 'many side' relation. This schema is justified since it clearly shows the layout and relations in the database and covers all requirements.



Self-Assessment:

I have completed all the main requirements for this section since my relational database schema has translated every supertype and their subtypes and all the attributes included in my EER Model. The primary keys are shown by underlines and the relationships shown by arrows pointing from foreign keys to the primary keys they are referring to.

4 MySQL Code

Creating Database:

```

1 DROP DATABASE vet_db; -- to prevent errors incase database already exists
2 CREATE DATABASE vet_db; -- creates the sql database that I will use
3
4 USE vet_db; -- selects this database so tables are added to it

```

These statements remove any already existing databases with the same name so that a new database can be created. 'USE' ensures all tables are created/edited within this database.

```

7  DROP TABLE IF EXISTS Company; -- prevents errors if relation already exists in database
8  CREATE TABLE Company -- creates company relation and defines its attributes/domain
9  (
10     CompanyID INT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE, -- cannot be negative or empty, must be unique
11     Name VARCHAR(255) NOT NULL,
12     GroomingServices ENUM ('True', 'False') DEFAULT 'False', -- must be either true or false, false by default
13     PRIMARY KEY (CompanyID) -- defines primary key of company relation
14 );

```

These statements delete the table 'Company' if it already exists to prevent errors it then defines the attributes for company and their datatypes. 'NOT NULL' is used if a field is required when making a new record. All the other relations in my conceptual relational database schema are created using a similar layout but a different name and domain. The 'UNIQUE' constraint ensures that all column values are different which is essential for a primary key which is why 'AUTO_INCREMENT' is used so that a new number can be generated automatically when a new record is inserted into a table. The 'DEFAULT' constraint means that if no value of 'GroomingServices' is inputted it will be recorded as 'False' automatically.

```

53 DROP TABLE IF EXISTS Receptionist; -- prevents errors if relation already exists in database
54 CREATE TABLE Receptionist -- creates receptionist relation and defines its attributes/domain
55 (
56     ReceptionistID INT UNSIGNED NOT NULL,
57     Telephone VARCHAR(255) NOT NULL,
58     PRIMARY KEY (ReceptionistID),
59     FOREIGN KEY (ReceptionistID) REFERENCES Staff(StaffID) -- foreign key is primary key since subset of staff
60 );

```

This relation is created with the same basic layout as 'Company' but it uses a foreign key 'ReceptionistID' which refers to 'StaffID' in the 'Staff' relation. In this case the foreign key is the primary key since receptionist is a subset of staff.

```

143 DROP TABLE IF EXISTS Owner_Animal; -- prevents errors if relation already exists in database
144 CREATE TABLE Owner_Animal -- creates owner to animal relation and defines its attributes/domain
145 (
146     Owner INT UNSIGNED NOT NULL,
147     Animal INT UNSIGNED NOT NULL,
148     PRIMARY KEY (Owner, Animal), -- two primary keys since links two relations that have many to many relationship
149     FOREIGN KEY (Owner) REFERENCES Owner(OwnerID),
150     FOREIGN KEY (Animal) REFERENCES Animal(AnimalID)
151 );

```

This relation has two primary keys which are both foreign keys and no other attributes since it is used to translate a many to many relationship to two, one to many relationships. The values of primary keys cannot be negative so 'UNSIGNED' is used to keep it positive.

Populating Database:

```

45 INSERT INTO Staff VALUES (1, 'Charlotte Harman', 'chmiah@gmail.com', 001); -- adds records/tuples to staff relation
46 INSERT INTO Staff VALUES (2, 'Hazel Parker', 'hparker@yahoo.com', 001);
47 INSERT INTO Staff VALUES (3, 'Marnie Trewern', 'marnie3@yahoo.com', 001);
48 INSERT INTO Staff VALUES (4, 'Charlie Harrison', 'work@har.cx', 002);
49 INSERT INTO Staff VALUES (5, 'Ellie Vosper', 'elliev@hotmail.com', 002);
50 INSERT INTO Staff VALUES (6, 'Alice Gilbert', 'alice.gilbert@gmail.com', 002);

```

```

91 INSERT INTO Animal VALUES (01, 'Muffin', 'F', '2005-05-16'); -- adds records/tuples to animal relation
92 INSERT INTO Animal VALUES (02, 'Jerry', 'M', '2018-12-01');
93 INSERT INTO Animal VALUES (03, 'Max', 'M', '2014-10-10');
94 INSERT INTO Animal VALUES (04, 'Voltaire', 'M', '2010-12-26');
95 INSERT INTO Animal VALUES (05, 'Portia', 'F', '2005-05-16');
96 INSERT INTO Animal VALUES (06, 'Widget', 'F', '2020-01-07');

```

These statements are used to populate the database. For other relations 'Staff' or 'Animal' is replaced with the relation name and the data is put in the brackets separated by commas in the order defined in the relation schema. To populate the database, I used dummy data I got from multiple sample excel spreadsheets online and also some I created myself.

Embedded MySQL:

```

43 <?php
44 $result = mysqli_query($conn, //querys mysql database
45 "SELECT Company.Name, Building.City, Building.Postcode, Building.BuildingID
46 FROM Building
47 INNER JOIN Company ON Building.Company=Company.CompanyID"); //selects records with matching values in both tables

```

This query uses an 'INNER JOIN' to select records from both tables that have matching values. 'Company.Name' means name attribute from the 'Company' relation.

```

75 $result = mysqli_query($conn, //querys mysql database
76 "SELECT Name, Email, Building
77 FROM Staff
78 ORDER BY Name ASC"); //displays slected records in alphabetical order of name

```

This query uses a basic 'SELECT' statement to retrieve the name, email and building of staff and orders the displayed information in alphabetical order by name.

```

102 $result = mysqli_query($conn, //querys mysql database
103 "SELECT Appointment.Time, Appointment.Room, Staff.Building, Staff.Name, Animal.Petname
104 FROM Appointment
105 INNER JOIN Vet ON Appointment.Vet=Vet.VetID
106 INNER JOIN Staff ON Appointment.Vet=Staff.StaffID
107 INNER JOIN Animal ON Appointment.Animal=Animal.AnimalID
108 ORDER BY Time ASC LIMIT 5"); //only displays next 5 appointments

```

This query uses multiple 'JOIN' clauses to connect relation instances. Orders by time and limits to 5 so that only the next 5 appointments will be displayed.

```

151 $result = mysqli_query($conn, //querys mysql database
152 "SELECT Owner.Name, Animal.Petname
153 FROM Owner_Animal
154 INNER JOIN Owner ON Owner_Animal.Owner=Owner.OwnerID
155 INNER JOIN Animal ON Owner_Animal.Animal=Animal.AnimalID
156 WHERE Owner.Name LIKE ".$_POST["owner"].""); //where name starts with entered value so don't need to enter fullname

```

This query uses a 'WHERE' clause to filter records based on user input (\$_POST["owner"]). The 'LIKE' operator means it will filter records where name starts with what the user inputted, like a search bar.

```

195 $result = mysqli_query($conn,
196 "SELECT Animal.Petname, Animal.Gender, Animal.DOB, Cat.Colour
197 FROM Animal
198 INNER JOIN Cat ON Animal.AnimalID=Cat.CatID
199 ORDER BY DOB DESC"); //ordered by DOB in descending order

```

This query selects animals that are cats using an 'INNER JOIN'.

```

212 $result = mysqli_query($conn,
213 "SELECT Animal.Petname, Animal.Gender, Animal.DOB, Dog.Breed
214 FROM Animal
215 INNER JOIN Dog ON Animal.AnimalID=Dog.DogID
216 ORDER BY DOB DESC"); //ordered by DOB in descending order

```

This query selects animals that are dogs using an 'INNER JOIN'.

```

230 "SELECT Petname, Gender, DOB
231 FROM Animal
232 WHERE AnimalID NOT IN ((SELECT CatID FROM Cat) UNION (SELECT DogID FROM Dog))
233 ORDER BY DOB DESC"); //uses subquery and union to select all animals not in Cat or Dog tables

```

This query selects animals that are neither dogs or cats using two subqueries and 'UNION' which combines both subqueries.

Key Issues:

When writing the MySQL code for implementing the relational database schema I designed and for reading and writing data from the MySQL Database I came across some key issues. An example of this was when joining two tables that had identical column names. I overcame this by using aliases to give the column/attribute a temporary name allowing my program to distinguish between the two. I also had trouble using multiple constraints and subqueries in one complex select statement, but I fixed this with correct use of brackets. When implementing the database, I had to manually

delete my database or certain tables every time I edited the SQL statements. To overcome this, I added 'DROP DATABASE vet_db;' at the start of my .sql file and before creating each table I used 'DROP TABLE IF EXISTS tablename' to make this more efficient and to improve error handling.

Self-Assessment/Justification:

I have completed all the main requirements for this section since I have implemented all relations and their primary and foreign keys. The statements in my .sql file have executed successfully to create a database and the sql embedded in my php scripts contains a sub-query, select statement and join statement as seen above.

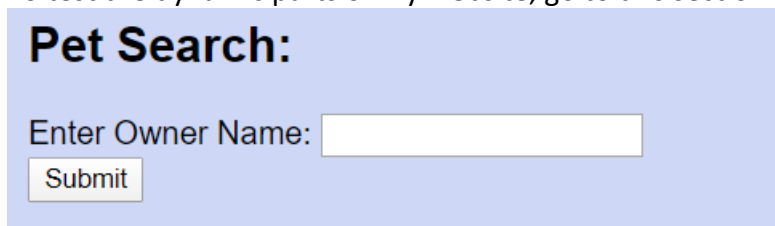
5 Website Working with MySQL Database

Website Files:

- cat.jpg – image used in HTML section of php file
- main.css – defines style of HTML section of php file
- VetWebsite.php – contains HTML of webpage and php script used to access and query MySQL database

Website Testing:

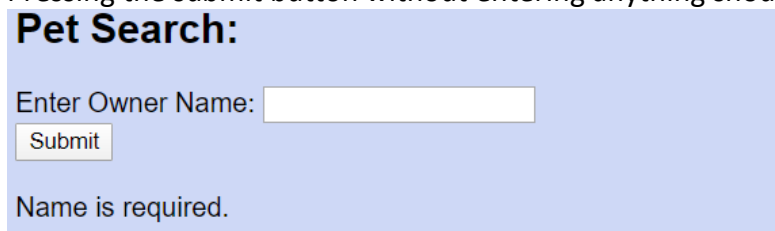
To test the dynamic parts of my website, go to this section of the webpage-



Pet Search:

Enter Owner Name:

Pressing the submit button without entering anything should display this-

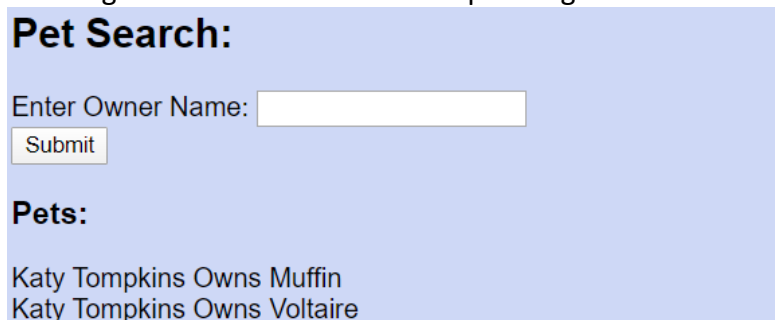


Pet Search:

Enter Owner Name:

Name is required.

Entering 'ka' into the text box then pressing submit should display this-



Pet Search:

Enter Owner Name:

Pets:

Katy Tompkins Owns Muffin
Katy Tompkins Owns Voltaire

Entering 'Arno' into the text box then pressing submit should display this-

Pet Search:

Enter Owner Name:

Pets:

0 results

To test the dropdown box, go to this section of webpage-

Patients:

Select Pet Type:

Selecting 'Dog' then pressing submit should display this-

Dogs:

Widget F 2020-01-07 Chow Chow
Max M 2014-10-10 Labrador
Voltaire M 2010-12-26 French Bulldog

Selecting 'Other' then pressing submit should display this-

Other Animals:

Jerry M 2018-12-01

Self-Assessment/Justification:

Evidence that all basic requirements of php section have been completed-

Connecting to MySQL Database

```
4 <?php
5 $conn = mysqli_connect("localhost", "root", "", "vet_db"); //connects to mysql database
6 if (! $conn){ //incase of error accessing mysql database
7     die("Connection Failed: ". mysqli_connect_error());
8 }
9 ?>
```

Querying MySQL Database

```
44 $result = mysqli_query($conn, //querys mysql database
45 "SELECT Company.Name, Building.City, Building.Postcode, Building.BuildingID
46 FROM Building
47 INNER JOIN Company ON Building.Company=Company.CompanyID"); //selects records with matching values in both tables
```

Reading Data out of result of MySQL Query

```

80 if(mysqli_num_rows($result) > 0){ //only does this if data exists to prevent errors
81     while ($r = mysqli_fetch_assoc($result)) { //repeats for each selected record
82         echo "<tr><strong><td>",$r["Name"],"</td></strong><td>    "; //reads data out of result of mysql query
83         echo $r["Email"],"</td><td>    Building:";
84         echo $r["Building"],"</td><br></tr>";
85     }
86 } else {
87     echo "0 results";
88 }

```

Freeing Results of MySQL Query

```

89 mysqli_free_result($result); //frees result of mysql query so more queries can be carried out

```

Releasing Connection to MySQL Database

```

260 <?php
261 mysqli_close($conn);
262 ?>

```

Dynamically Constructed MySQL Statement

```

156 WHERE Owner.Name LIKE '". $_POST["owner"]."%'"; //where name starts with entered value so don't need to enter fullname

```

Error Handling

```

6 if (!$conn){ //incase of error accessing mysql database
7     die("Connection Failed: ". mysqli_connect_error());
8 }

```

```

49 if(mysqli_num_rows($result) > 0){ //only does this if data exists to prevent errors

```

6 Advanced Tasks

EER Model:

I created a more complicated EER model than what was required which involved additional entities and therefore more relationships and attributes. I also included a second supertype-subtype relationship which allowed me to use more than one type of participation. This can be seen in my EER diagram in section 2. I did this to make it more reflective of a real EER Model that would be used in this scenario.

Relational Database Schema:

I translated everything in my EER model into a logical relational database schema which is more than what was required. This meant I used more relations and foreign keys, had to optimise more constraints and had to transfer multi-valued attributes into the form of the schema. This can be seen in my diagram in section 3. I did this to make it more a more interesting and complex design and so that I could use more complex queries for my webpage.

MySQL Database:

I used more of my logical relational database schema than what was required when creating my SQL Database. This meant I used more optimised relations, constraints and foreign keys. I also used all components of the implemented MySQL Database in my website. This can be seen in section 4. I did this to make it more a more interesting and complex design and so that I could use more complex queries for my webpage.

Normalisation:

My database is in third normal form since in every relation there are no multiple columns that you could use to get the same information, each tuple is always unique, there are no partial dependencies and no transitive dependencies. I did this to improve overall database organisation and data consistency and to make it a more flexible design with less redundant data.

MySQL Statements:

I used some slightly more advanced MySQL keywords, operators and constraints than what was required such as 'NOT IN', 'UNION', 'UNIQUE', 'AUTO_INCREMENT', 'LIMIT' and 'DEFAULT' as seen in section 4. I used these to improve my knowledge of SQL allowing me to make more complex and interesting SQL statements.

PHP:

I used php scripts to validate data entered from forms to prevent errors meaning I could implement more advanced database queries.

```

146 <?php
147 if ($_SERVER["REQUEST_METHOD"] == "POST"){ //checks if form submitted (validation)
148     if (empty($_POST["owner"])){ //checks if text box empty (validation)
149         echo "<br>Name is required.";
150     } else {
151         $result = mysqli_query($conn, //queries mysql database
152             "SELECT Owner.Name, Animal.Petname
153             FROM Owner_Animal
154             INNER JOIN Owner ON Owner_Animal.Owner=Owner.OwnerID
155             INNER JOIN Animal ON Owner_Animal.Animal=Animal.AnimalID
156             WHERE Owner.Name LIKE '$_POST[\"owner\"].%'; //where name starts with entered value so don't need to enter fullname

```

I used the superglobal '\$_SERVER["PHP_SELF"]' which returns the file name of the current script so error messages display on this page. htmlspecialchars prevents attackers from exploiting code by injecting HTML or Javascript by converting special characters to HTML entities like '<' to '<' so they cannot be used to inject. I did this to make the webpage more secure and convenient.

```

139 <form id="owners" method="post" onsubmit="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>" >

```

7 References

<https://www.draw.io> -creating EER Diagram

<https://www10.lunapic.com> -creating relational database schema

<https://www.wisdomaxis.com/technology/software/data/for-reports/> -dummy data for populating database