Pet Portal Database

A SIMPLIFIED PET MANAGEMENT SYSTEM

David Jensen

CIS 451 | FINAL PROJECT

TABLE OF CONTENTS

URL

The PetPortal Database is currently hosted on the University of Oregon ix servers. A link to the main page can be found here: http://ix.cs.uoregon.edu/~djensen/cis451/project/

SUMMARY

When it comes to pets, it's easy to forget about that once-a-year vaccine or yearly vet check-up. When was the last time you applied flea medication to a particular cat? Was it Erin or Chloe who is allergic to Frontline? It's easy to forget some of the details, especially when there's a lot of pets. The Pet Portal was designed to eliminate some of the guess work and simplify the pet owning experience.

My girlfriend Megan works at a vet clinic and occasionally brings home foster animals. Additionally, we stumbled across a situation where someone needed to re-home 16 cats this last summer! Trying to keep all the pet information (who's been spayed, neutered, vaccinated, microchipped, tested, etc...) up to date and in one place was quite a challenge, especially whilst trying to keep on top of our personal pets' needs. We needed a way to manage all of this data, and this project is the perfect time to compile all of our information! Although those with multiple pets will gain the most use from the Pet Portal database, anyone can use the Pet Portal database design to simplify the pet owning experience!

Features:

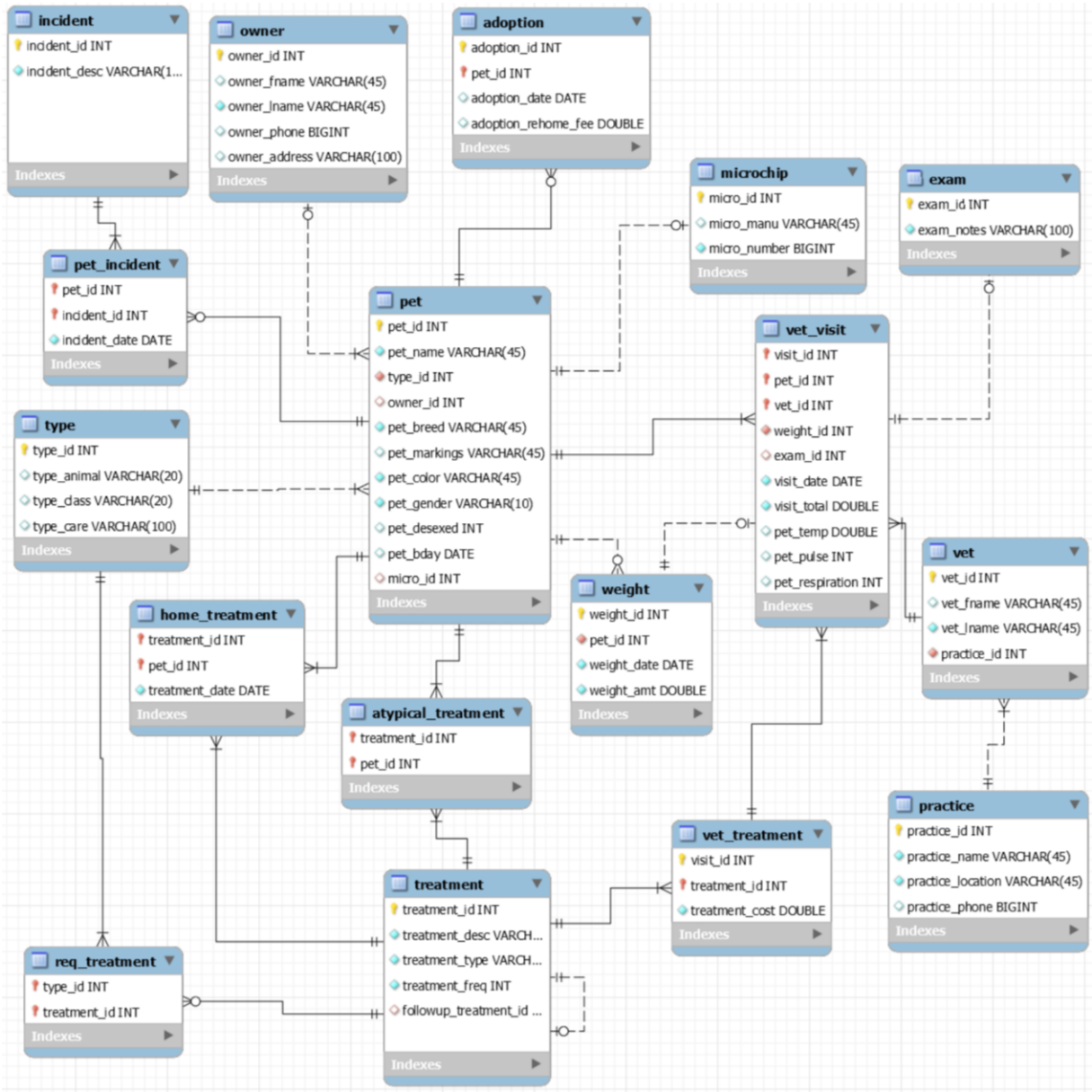
The PetPortal Database is designed to store information regarding your pets. The database allows for multiple owners to own multiple pets. A pet can be of any type, big or small. In a perfect world, every pet would have a home. Since this is not the case, my database supports foster care pets, for when an owner is unknown. Some of the information that is tracked includes required treatments, atypical treatments, vet visit history, vet treatment history, and pet weight history. All the data in the PetPortal Database is real, accurate data.

Functional Units:

By leveraging the features present in the database, a treatment report can be generated to display a list of treatments a pet is overdue for. This is very handy for tracking regular treatment applications like a flea medication or longer duration treatments like a 3 year rabies vaccine. The database can also generate a treatment history report which can be pivotal in providing a new vet with adequate pet treatment history. Seeing a vet can be very costly. It is possible to generate a report of vet treatment history to see exactly how expensive. Or generate a list of your most costly pets so you can scold them appropriately. Monitoring your pets' weight can also provide insights into the health of the pet. Update the database with periodic pet weights and query the information later for analysis.

LOGICAL DESIGN

The logical design of the database was created using MySQL Workbench. The ERD is in crowsfoot notation.



PHYSICAL DESIGN

The database is divided into many tables, and each table has a specific purpose. The entire database is in BCNF except where explicitly stated. We will first look at the data heavy tables in alphabetical order, then we will look at the bridge tables in alphabetical order to see how functional units can be created from this data.

Data Heavy Tables:

Data heavy tables do a lot of the heavy lifting in the database. They contain the bulk of the information stored in the database and act as link points for functional tables. Every data heavy table should contain its own primary key which can uniquely identify data in the table without the need for a foreign key or composite key.

ADOPTION:

The ADOPTION table maintains information pertaining to a pet adoption event. When an adoption event occurs, the date of adoption (adoption_date), the pet being adopted (pet_id) and any applicable rehoming fees (adoption_rehome_fee) are recorded. The adopting owner can be found by joining the PET and OWNER tables on their respective keys. Note that the owner_id in the PET table will need to be updated in conjunction with an adoption event.

Relationships:

M:1 adoption | pet FK: pet_id

Table Field	Data Type	Description
adoption_id	INT, NN, AI, PK	Primary key identifier, auto-increments
pet_id	INT, NN, FK	Foreign key to PET table, cannot be null
adoption_date	DATE	Records the adoption date, if known
adoption_rehome_fee	DOUBLE	Records the rehoming fee, if any

EXAM:

The EXAM table contains notes for a pet exam. Some argument could be made to merge this table with the VET_VISIT table and allow NULL values (a pet is not required to have an exam if seen within a particular time period). However, multiple cats can have the same chart notes (exam_notes), especially if multiple cats are seen for the same procedure (spay/neuters require a brief exam).

Relationships:

0:1 exam | vet_visit FK: exam_id

Table Field	Data Type	Description
exam_id	INT, NN, AI, PK	Primary key identifier, auto-increments
exam_notes	VARCHAR(100)	Records any notes or comments from the vet

INCIDENT:

The INCIDENT table contains a description field for an incident. An incident is an event which can affect the pet's health. An incident could range from something small (noticed ear mites on a cat) to something more significant (cat or dog got into a fight). The incident description is generated only once, and applied to a pet on the PET_INCIDENT table.

Relationships:

1:M incident | pet_incident FK: incident_id

Table Field	Data Type	Description
incident_id	INT, NN, AI, PK	Primary key identifier, auto-increments
incident_desc	VARCHAR(100)	Records any information about an incident

MICROCHIP

The MICROCHIP table contains information about who made a microchip (micro_manu) and the assigned microchip number (micro_number). The microchip manufacturer is important because each manufacturer has a different website to update personal contact information on. Pet microchips are important because they provide a method for returning a lost pet to an owner. This table is in 2NF as the micro_manu field is duplicated unnecessarily. A new table to hold the manufacturer information would make this table BCNF yet it didn't seem pivotal at the present.

Relationships:

0:1 microchip | pet FK: micro_id

Table Field	Data Type	Description
micro_id	INT, NN, AI, PK	Primary key identifier, auto-increments
micro_manu	VARCHAR(100)	Records the microchip manufacturer information
micro_number	BIGINT, NN	The microchip number stored as a big integer

OWNER:

The OWNER table contains contact information for a pet owner. This contact information includes the owner's name (owner_fname, owner_lname), phone number (owner_phone), and address (owner_address). This information is stored in the event that an owner needs to be contacted for some reason. For example, if a pet is lost and the microchip information has not been updated on the website to reflect the new owners contact information, we can attempt to contact the owner ourselves or take steps toward an appropriate action if necessary.

Relationships:

0:M owner | pet FK: owner_id

Table Field	Data Type	Description
owner_id	INT, NN, AI, PK	Primary key identifier, auto-increments
owner_fname	VARCHAR(45)	Records the owner's first name
owner_lname	VARCHAR(45)	Records the owner's last name
owner_phone	BIGINT	Big integer used to store an owners phone number
owner_address	VARCHAR(100)	Records the owners full address

PET:

The PET table contains a lot of information that is unique to each pet. Each pet requires a name (pet_name), an animal type (type_id), a breed (pet_breed), primary color (pet_color), and a gender (pet_gender). Optional fields include the owner (owner_id), identifying pet markings (pet_markings), whether the animal has been spayed or neutered (pet_desexed), the pet's date of birth, if known (pet_bday), and the microchip identifier, if applicable (micro_id). The PET table is a central hub for a significant number of other tables, which all reference the pet identifier (pet_id).

Relationships:

M:1 pet | owner FK: pet_id
1:M pet | adoption FK: pet_id
1:1 pet | microchip FK: pet_id
1:M pet | vet_visit FK: pet_id

1:M pet | weight FK: pet_id
1:M pet | atypical_treatment FK: pet_id
1:M pet | home_treatment FK: pet_id
1:1 pet | type FK: pet_id
1:M pet | pet_incident FK: pet_id

Table Field	Data Type	Description
pet_id	INT, NN, AI, PK	Primary key identifier, auto-increments
pet_name	VARCHAR(45), NN	Records the pets name, cannot be NULL
type_id	INT, NN, FK	Foreign key to TYPE table, cannot be NULL
owner_id	INT, FK	Foreign key to OWNER table
pet_breed	VARCHAR(45), NN	Records the pet breed
pet_markings	VARCHAR(45)	Records any significant markings the pet may have
pet_color	VARCHAR(45), NN	Records the primary pet colors
pet_gender	VARCHAR(10), NN	Records the pet gender
pet_desexed	INT	Records if the pet has been spayed/neutered (1) or not (0)
pet_bday	DATE	Records the pets date of birth, if known
micro_id	INT, FK	Foreign key to MICROCHIP table

PRACTICE:

The PRACTICE table contains information about a veterinary office. This information includes the name of the practice (practice_name), the location (practice_location), and the phone number (practice_phone). This is useful as it will link a vet to a practice which can be useful to obtain vet records.

Relationships:

1:M practice | vet FK: practice_id

Table Field	Data Type	Description
practice_id	INT, NN, AI, PK	Primary key identifier, auto-increments
practice_name	VARCHAR(45)	Records the name of the veterinary practice
practice_location	VARCHAR(45)	Records the veterinary office location
practice_phone	BIGINT	Big integer used to store the phone number

TREATMENT:

The TREATMENT table contains different types of treatments that can be performed at home or at the vet's office. The VET_TREATMENT and HOME_TREATMENT tables both link to the same TREATMENT table to generate line items for treatment history. The TREATMENT table contains a treatment description (treatment_desc), a treatment type (treatment_type), the frequency of treatment (treatment_freq), and a follow up treatment, if necessary (followup_treatment_id).

The treatment type is used for differentiating treatment types (vaccine, flea prevention, etc). The treatment frequency contains information about how frequently a treatment needs to be performed (365 = 1 year). The followup_treatment_id is populated by a circular lookup on the treatment_id column and includes the next treatment id. This is especially useful for vaccines that follow a sequence (Lepto #1 -> Lepto #2 > Lepto 1yr -> Lepto 1yr…). This table does not necessarily contain prescription information, but it could be added here.

Relationships:

1:M treatment | atypical_treatment FK: treatment_id
1:M treatment | vet_treatment FK: treatment_id

1:M treatment | treatment FK: treatment_id=followup_treatment_id
1:M treatment | req_treatment FK: treatment_id
1:M treatment | home_treatment FK: treatment_id

Table Field	Data Type	Description
treatment_id	INT, NN, AI, PK	Primary key identifier, auto-increments
treatment_desc	VARCHAR(45), NN	Records a description of the treatment
treatment_type	VARCHAR(45), NN	Records the type of treatment
treatment_freq	INT, NN	Records the frequency of repeating treatments
followup_treatment_id	INT	Records the next treatment in sequence, or self if repeating

TYPE:

The TYPE table contains information about the type of a pet. This includes information such as the animal type (type_animal), the classification of the pet (type_class), and a brief description about the type of care a particular type of animal needs (type_care). Some examples of animal types are feline and canine. Some examples of classifications are reptiles and mammals. UVB bulbs for turtles and vaccinations are some examples of types of care a particular animal needs.

Relationships:

1:M type | pet FK: type_id
1:M type | req_treatment FK: type_id

Table Field	Data Type	Description
type_id	INT, NN, AI, PK	Primary key identifier, auto-increments
type_animal	VARCHAR(20)	Records the type of animal
type_class	VARCHAR(20)	Records the classification of an animal
type_care	VARCHAR(100)	Contains a description of required animal care

VET:

The VET table contains information about the vet. This includes information such as the veterinarian's name (vet_fname, vet_lname) and where the vet works (practice_id).

Relationships:

1:M vet | vet_visit FK: vet_id
M:1 vet | practice FK: practice_id

Table Field	Data Type	Description
vet_id	INT, NN, AI, PK	Primary key identifier, auto-increments
vet_fname	VARCHAR(45)	Records the first name of the vet
vet_lname	VARCHAR(45)	Records the last name of the vet
practice_id	INT, NN, FK	Foreign key to PRACTICE table, where the vet works

WEIGHT:

The WEIGHT table contains information about a pet's weight. This includes the pet (pet_id), the date the pet was weighed (weight_date) and the recorded weight of the pet (weight_amt). By monitoring pet weight, one can gain some insights into potential issues with a pet. For example, a sudden decrease in pet weight could indicate kidney issues whereas a sudden increase in pet weight could indicate thyroid problems.

Relationships:

M:1 weight | pet FK: pet_id

1:1 weight | vet_visit FK: weight_id

Table Field	Data Type	Description
weight_id	INT, NN, AI, PK	Primary key identifier, auto-increments
pet_id	INT, NN, FK	Foreign key to PET table, the pet to be weighed
weight_date	DATE, NN	The date the pet was weighed
weight_amt	DOUBLE, NN	The amount the pet weighed

Functional Tables:

Functional tables are significantly more interesting. Functional tables (bridge tables) link two data heavy tables together to provide additional functionality. These tables do not have a self-identifying primary key, so they rely on foreign keys from other tables to identify the data. These tables are a great way to correlate data from one table to another. The functional tables will be listed below in alphabetical order. All of these tables are in BCNF.

ATYPICAL_TREATMENT

The ATYPICAL_TREATMENT table contains information about optional or different types of treatments a particular pet is receiving. For example, there are multiple types of flea medication, but only 1 type needs to be applied every month. This table links the PET and TREATMENT tables together to generate a correlation between a pet and a treatment. For example, not all pets need a Sentry Calming Collar, but a particular cat (Kaylin) needs one every month as she has anxiety issues. If untreated, Kaylin's anxiety will lead to being extra noisy around other cats, or as a worst case, urinating in undesignated areas.

Relationships:

M:1 atypical_treatment | pet FK: pet_id
M:1 atypical_treatment | treatment FK: treatment_id

Table Field	Data Type	Description
treatment_id	INT, NN, FK	Foreign key to TREATMENT table, links treatment to pet
pet_id	INT, NN, FK	Foreign key to PET table, links a pet to a treatment

HOME_TREATMENT

The HOME_TREATMENT table contains information about treatments (treatment_id) a pet (pet_id) received at home on a particular date (treatment_date). This table links the PET and TREATMENT tables to generate a correlation between a pet and a treatment they received. For example, the cat Chloe receives Revolution flea medication on the first of this month. The treatment table will also indicate that Revolution has a 30 day (treatment_freq) follow-up with another dose of Revolution (followup_treatment_id). This information can then be used to generate a report of required treatments and their due dates (based on the date a treatment was last received).

Relationships:

M:1 home_treatment | pet FK: pet_id
M:1 home_treatment | treatment FK: treatment_id

Table Field	Data Type	Description
treatment_id	INT, NN, FK	Foreign key to TREATMENT table, links a treatment to a pet
pet_id	INT, NN, FK	Foreign key to PET table, links a pet to a treatment

treatment_date	DATE	The date a treatment was applied to a pet

PET_INCIDENT

The PET_INCIDENT table contains information about which incidents (incident_id) happened to which pets (pet_id) on a particular date (incident_date). This table links the INCIDENT and PET tables to generate a correlation between an incident and a pet. For example, it was noticed that the cat Cordelia (pet_id) had eye discharge (incident_id) on Oct 17th, 2017 (incident_date).

Relationships:

M:1 pet_incident | pet_id FK: pet_id
M:1 pet_incident | incident_id FK: incident_id

Table Field	Data Type	Description
pet_id	INT, NN, FK	Foreign key to PET table, links a pet an incident
incident_id	INT, NN, FK	Foreign key to INCIDENT table, links an incident to a pet
incident_date	DATE	The date an incident occurred

REQ_TREATMENT

The REQ_TREATMENT table contains information about treatments (treatment_id) an animal (type_id) is required to have. This table links the TYPE and TREATMENT tables together to generate a correlation between a type of animal and the treatment they need to receive. For example, all dogs (type_id) need a rabies vaccine (treatment_id). The treatment table will also indicate that Rabies 1yr vaccine must be followed up with a Rabies 3yr vaccine. So the Rabies 3yr vaccine will also be added to the required treatment table until a loop is found (Rabies 3yr -> Rabies 3yr). Note that this is manually entered and not done programmatically.

This functional table can also be compared with the union of the VET_TREATMENT and HOME_TREATMENT tables to see if a particular treatment that a type of pet needs has been administered to a pet of that type. For example, if all cats need a Rabies 1yr vaccine (treatment_id), we can look to see if a cat (type_id) Nitro has received the vaccine at home (HOME_TREATMENT) or at the vets office (VET_TREATMENT).

Relationships:

M:1 req_treatment | type_id FK: type_id
M:1 req_treatment | treatment_id FK: treatment_id

Table Field	Data Type	Description
type_id	INT, NN, FK	Foreign key to TYPE table, links type to a treatment
treatment_id	INT, NN, FK	Foreign key to TREATMENT table, links treatment to a type

VET_TREATMENT

The VET_TREATMENT table contains information about treatments (treatment_id) performed at the vet's office and the cost of the treatment (treatment_cost). This table can be thought of as a line item table for a vet visit. This table links the TREATMENT and VET_VISIT tables to generate a correlation between a vet visit that a pet went to, a treatment they received, along with the associated cost of the treatment. For example, a vet visit (visit_id) consisted of a spay (treatment_id) for $80 (treatment_cost), and a microchip implant (treatment_id) for $35 (treatment_cost). Here, the visit_id is a composite key with treatment_id. The cost

field is not redundant as different vet offices could (and do!) charge different prices for the same procedures. Prices can also change at the same vet office due to promotions or inventory changes.

Relationships:

M:1 vet_treatment | vet_visit FK: visit_id
M:1 vet_treatment | treatment FK: treatment_id

Table Field	Data Type	Description
visit_id	INT, NN, CK	Composite key, links a visit to a treatment
treatment_id	INT, NN, FK	Foreign key to TREATMENT table, links treatment to visit
treatment_cost	DOUBLE	The cost of the treatment

VET_VISIT

The VET_VISIT table contains information about a pet (pet_id) who received a vet_treatment (visit_id) by a vet (vet_id) on a particular date (visit_date) which incurred a total visit cost (visit_cost). The VET_TREATMENT table contains all the specifics of the treatments received and the line item costs for each individual treatment. The VET_VISIT table also includes some visit specific information including the weight (weight_id) of the animal, the exam notes (exam_id) if any, and pet TPR information (pet_temp, pet_pulse, pet_respiration). The VET_VISIT table contains the most foreign keys in the database as a lot of the information is extrapolated from other tables.

Relationships:

M:1 vet_visit | exam FK: exam_id
M:1 vet_visit | vet FK: vet_id
M:1 vet_visit | vet_treatment FK: visit_id
M:1 vet_visit | weight FK: weight_id
M:1 vet_visit | pet FK: pet_id

Table Field	Data Type	Description
visit_id	INT, NN, FK	Foreign key to VET_TREATMENT, links to line items
pet_id	INT, NN, FK	Foreign key to PET, links pet to a vet visit
vet_id	INT, NN, FK	Foreign key to VET, links to vet who was at the visit
weight_id	INT, NN, FK	Foreign key to WEIGHT, links weight on visit to weight table
exam_id	INT, FK	Foreign key to EXAM, links exam notes to vet visit
visit_date	DATE, NN	Date the pet saw the vet
visit_total	DOUBLE, NN	The total cost of the vet visit
pet_temp	DOUBLE	The temperature of the pet
pet_pulse	INT	The pulse of the pet
pet_respiration	INT	The respiration of the pet

LIST OF APPLICATIONS

The program is divided into many different types of applications. The more interesting types of applications are those that span across multiple tables. On the website that has been created, the functionality has been divided into many different pages which are accessed via the menu on the left. Each page invokes two different types of functionality. More detail about each page will be listed below. To simplify the testing process, an initial value has been loaded into each field, where applicable.

Home:

The “Home” page provides a brief outline of the project and contains a drop down menu to list/view the content from every table in the database.

Pets by Type:

In the PetPortal Database, all animals are organized into categories. Each type of animal has different types of care, which is reflected in the REQUIRED_TREATMENT table. The “Pets by Type” page executes some simple queries to familiarize visitors with the myriad of pet types and pet names that are present in the database. Any of these pet names can be used in any query that requires a pet name. However, the Felines and Canines will have the most information present in the database. The following functionality is on this page:

Q1A: This query is used to generate a drop down list of different types of animals. This query leverages the following tables: [TYPE]

Q1B: This query piggybacks off of Q1A to return a list of animals that are of the chosen type. This query leverages the following tables: [PET, TYPE]

Treatment History:

In the PetPortal Database, pet treatment information is either performed at home (HOME_TREATMENT) or at the vet’s office (VET_TREATMENT). Typical home treatments include flea medication or other simple pet care treatments. Vet treatments are typically more advanced and can contain treatments such as vaccinations or surgery (although there are no limits on the type of treatments that can be added to the database). The “Treatment History” page concatenates the vet and home treatment information into a single result set. This can be useful if you need to take a pet to a new vet – all the history is in the same place! The following queries are on this page:

Q2A: Executes when the page is loaded. This query counts the number of treatments each pet has received and displays the results on the right hand side of the page. Any of the pet names in this table can be used to generate applicable information for the pet. This query leverages the following tables: [HOME_TREATMENT, TREATMENT, PET] U [VET_TREATMENT, TREATMENT, VET_VISIT, PET]

Q2B: Once a pet name is submitted, this query will list all vet or home treatments received for a specific pet. This query leverages the following tables: [HOME_TREATMENT, TREATMENT, PET] U [VET_TREATMENT, TREATMENT, VET_VISIT, PET]

Required Treatments:

In the PetPortal Database, each type of animal is linked to specific treatments that are required (REQ_TREATMENT) for a specific type of animal. For example, all cats need the FeLV sequence, FVRCP sequence, and Rabies sequence of vaccines. Additionally, certain pets require additional specific treatments (ATYPICAL_TREATMENT) that may be unique to a particular pet. These include treatments such as a Sentry

Calming Collar (for cats with mild to moderate behavioral issues/quirks) or a different brand of flea medication (due to allergic reactions with a particular brand). As a last example, the feline FVRCP vaccine comes in two flavors: the 1 year and 3 year FVRCP vaccination. For higher risk cats (outdoor or indoor/outdoor or cats that come into frequent contact with dogs who are outside more frequently), the 1 year vaccine is recommended. For low risk cats (older, indoor only cats), the 3 year is adequate. This page leverages both tables (REQ_TREATMENT, ATYPICAL_TREATMENT) to determine which treatments need to be applied to a designated animal. The following queries are on this page:

Q3A: Generates a list of required and atypical treatments that have already been implemented along with the date of implementation. This is displayed on the right hand side of the page after a pet name has been submitted. This query leverages the following tables: { [REQ_TREATMENT, TREATMENT, TYPE, PET] U [ATYPICAL_TREATMENT, PET, TREATMENT], [HOME_TREATMENT, TREATMENT, PET] U [VET_TREATMENT, TREATMENT, VET_VISIT, PET], TREATMENT, PET }

Q3B: Generates a list of required or atypical treatments that are NOT implemented. This is displayed in the center of the page. This query leverages the following tables: { [REQ_TREATMENT, TREATMENT, TYPE, PET] U [ATYPICAL_TREATMENT, PET, TREATMENT], [HOME_TREATMENT, TREATMENT, PET] U [VET_TREATMENT, TREATMENT, VET_VISIT, PET], TREATMENT, PET }

Treatments Due:

In the PetPortal Database, most types of treatments are recurring. The TREATMENT table tracks the repeat cycles of all recurring treatments. The “Treatments Due” page utilizes the TREATMENTS table in conjunction with treatments that are already in place (VET_TREATMENTS, HOME_TREATMENTS) to compile a list of due dates for recurring treatments. These are sorted in reverse order – so the treatment that should be applied next (future or past) is listed at the top. This is especially useful to determine when the next flea medication or 3 year vaccine needs to be applied. The following queries are on this page:

Q4A: Generates a list of required treatments that are NOT implemented (similar to 3QB). This is displayed on the right hand side of the page. This is useful as treatments that have not been applied do not have a “due date” for repeat treatment. However, vaccines are still important, so we want users to be aware that this list is not comprehensive. This query leverages the following table relationships: { [REQ_TREATMENT, TREATMENT, TYPE, PET] U [ATYPICAL_TREATMENT, PET, TREATMENT], [HOME_TREATMENT, TREATMENT, PET] U [VET_TREATMENT, TREATMENT, VET_VISIT, PET], TREATMENT, PET }

Q4B: Generates a list of vaccine and flea medication due dates (where a treatment has been applied) for a selected pet. This query leverages the following tables: [HOME_TREATMENT, TREATMENT, PET] U [VET_TREATMENT, TREATMENT, VET_VISIT, PET], PET, TREATMENT

Weight History:

The PetPortal Database allows for pet weight tracking. Monitoring a pets weight is important because a pet that is under or over weight can signify pet health concerns. As with people, a pet that is overweight is at

increased risk for developing disorders such as diabetes, arthritis, kidney disease, cancer and other life threatening diseases. Underweight animals can signify issues such as thyroid problems, intestinal parasites, worms and many other issues. Although the "Weight History" page doesn't leverage extremely beefy queries, the information is still vital to track and review. The following queries are on this page:

Q5A: This query executes when the page is loaded and provides are weight record count on the right hand side of the page. Pets in this table can be used for query Q5B. This query leverages the following tables: [WEIGHT, PET]

Q5B: This query loads when a valid pet has been submitted and displays all weight history records for the selected pet. Since the WEIGHT table does not specify a measurement type, we need to check the animal type to determine the units of measure. Felines, Canines, and Rabbits are all weighed in pounds. Turtles, Geckos, and Tortoises are all weighed in grams. This query leverages the following tables: [WEIGHT, PET, TYPE]

Microchip Lookup:

The PetPortal Database maintains a record of pet microchip information. If a pet is lost (or stolen I guess), the microchip information is used locate the owners of the pet. By maintaining a list of owner information and pet microchip data, we can easily look the microchip up in the database to give the owner a call. The following queries are on this page:

Q6A: Generates a list of microchip numbers that are in our database for testing purposes. This information is displayed on the right hand side of the page. This query utilizes the following tables: [MICROCHIP]

Q6B: Assuming the pet has an owner (we do rescue work, so we allow NULL values), the pet owners' contact information is displayed when the microchip number is submitted. As we are using actual user data, the last 4 digits of the phone number have been obfuscated for owner privacy for the scope of this project. This query utilizes the following tables: [MICROCHIP, PET, OWNER]

Adoptions:

The PetPortal Database was designed because we love doing pet rescue work. Seeing our furry (and not so furry) animals that have been adopted to good homes is its own reward. This page does not perform any custom queries, but our adoption information will be displayed here. If we are able to get all of our current information in, expect to see at least 20 records here (from the last summer alone). The following queries are present on this page:

Q7A: Counts the total number of adoption records present in the database. This is the simplest query on our page. This table uses the following table: [ADOPTION]

Q7B: Returns the pet adoption records for all pets in our database. Future implementations may limit the results to the 20 most recent records. This query leverages the following tables: [ADOPTION, PET, OWNER]

Pet Owners:

This page provides a list of all pets associated to a particular owner. This will be more useful as more owners are added to the database.

Q8A: This query counts the number of animals per animal type for the selected owner and displays the counts on the right hand side of the page. This query uses the following tables: [PET, TYPE, OWNER]

Q8B: This query returns the list of animals for the selected pet owner. The following tables are leverages to gather these results: [PET, OWNER, TYPE]

Sources:

This page provides a list of sources. All the files are in plain text so the links can be accessed directly in the browser. The source list provides links for the CSS, JavaScript, and PHP files.

USERS GUIDE

The simplest way to utilize the PetPortal Database is via the web interface. This is where forms, functions and queries have been written to streamline the retrieval process. A user will need to manually enter any new data into the database via MySQL Workbench or another similar application. The web interface contains a menu on the left hand side which allows a user to select the type of query they wish to perform. More information about each type of query can be found in the "List of Applications" section. The webpage uses XML to populate a region of the page, so a single HTML file can be used for multiple MySQL queries.

The following menu options are located on the left hand side of the page and can be clicked on to load the appropriate forms and data fields:

Pets By Type:

Allows a user to display all animals of a particular animal type (Feline, Turtle, Canine, etc) by selecting the appropriate option in the drop down list. This list is dynamically generated, so adding a new type of animal to the database will add the option to the drop down list on the webpage. After making a selection, the pets in the database of the selected type will be displayed with the following fields: {Pet Name, Breed, Color, Gender}.

Treatment History:

Allows a user to view pet treatment history via a union on the VET_TREATMENT and HOME_TREATMENT tables. A list of available pets with a record count is displayed on the right. While any pet name can be used in the pet lookup field, using one of the animals on the right hand side will return the corresponding number of records. An initial value of "Nacho" has been filled in the text field. A user can use this value or specify their own. When the submit button is clicked, the treatment records are returned with the following fields: {Pet Name, Treatment, Treatment Type, Date}.

Req Treatments:

Allows a user to view a list of required treatments for a particular pet. A list of available pets with a record count is not displayed on this page, so a user will either need to return to the “Pets by Type” page to locate potential animal names or use the provided pet name “Stanley”. When the submit button is clicked, a list of required treatments that have not been implemented are returned with the following fields: {Treatment, Treatment Type}. Another table is generated on the right which lists the required treatments that have already been implemented with the following fields: {Treatment, Treatment Type, Date Applied}.

Treatments Due:

Allows a user to view a list of recurring treatments for a particular pet and when the next application of a treatment is due. A list of available pets is not displayed on this page, so a user will either need to return to the “Pets by Type” page to locate potential animal names or use the provided pet name “Nitro”. When the submit button is clicked, a list of required treatments that have not been implemented are returned with the following fields: {Treatment, Treatment Type, Treatment Due}. Since this table only lists treatments that have been applied at least once, another table is generated on the right which lists the required treatments that have NOT been implemented. The fields on this table are: {Treatment, Treatment Type}.

Weight History:

Allows a user to view a list of recorded pet weights. A list of available pets and their corresponding number of weight records is displayed on the right. The fields on this table are {Pet Name, Pet Type, Records}. While any pet name can be used, if a pet does not have any records, no results will be returned. An initial value of “NotStan” has been provided. When the submit button is clicked, a list of pet weights sorted by most recent is returned. The fields on this table are {Pet, Weight Date, Weight Amount}. Note that the “Weight Amount” values will change depending on animal type. Most reptiles are weighted in grams, while mammals are weighed in pounds.

Microchip Lookup:

Allows a user to lookup pet owner information via the microchip number. A list of available microchip numbers is displayed on the right hand side with the following field: {Microchip Numbers}. This allows a user to copy and paste a value into the form for testing purposes. An initial microchip number has been filled in. This value can be used, or another one selected from the list on the right. When the submit button is clicked, the pet and pet owner information is retrieved. The following fields are returned: [Pet Name, Owner, Phone, Manufacturer, Microchip Number}. This will help to expedite the process of returning a lost pet.

Adoptions:

Allows a user to view pet owner information for animals that have been adopted. No custom queries are generated for this page. A running tally of pet adoptions is displayed on the right hand side of the page with the following field: {Adoptions}. The primary results are returned in the center of the screen with the following fields: {Adopter, Pet Name, Pet Breed, Pet Gender, Adoption Date}

Pet Owners:

Allows a user to look up pet information by owner name. The query returns results based on either the owner's first or last name. The owner's full name will not return any results. Once an owner's name is selected, the pet results are returned in the center of the screen with the following headers: {Pet Name, Animal Type, Pet Breed, Pet Gender}. A summary of animal types owned by this person are displayed on the right hand side with the following fields: {Animal Type, Number}. The default owner lookup is set to "Megan" as she has a significant number of adopted animals as she does foster care for International Reptile Rescue and frequently fosters animals who are in poor living conditions.

TABLE DATA

The contents of each table are displayed on the "Home" page on the website. The dropdown can be accessed to display all records for a particular table. Note that although there are not over 1,000 records in a single table, MySQL may limit return results to a maximum of 1,000 records.

IMPLEMENTATION CODE

The implementation code for the data retrieval process is divided into several steps to form a single cohesive process. The process starts on the HTML page, which executes JavaScript to generate the page content. Once the content is generated, a second layer of JavaScript code executes the desired PHP function. The PHP function queries the PetPortal Database and returns the results to the calling JavaScript function, which then updates the page. All the code for individual source files and SQL queries can be found on the "Sources" website tab.

HTML:

The index.html page is a very lightweight implementation of HTML that contains nested div blocks to generate a layout of the page. Each div block has an id and class. The id is used for locating the element to insert data into later. The class is used as a method of classifying types of data so a style can be applied to the block.

JavaScript:

There are three JavaScript files in this project: main.js, data.js and pet_queries.js. The source code for these files can be found on the "Sources" tab on the website.

DATA.JS:

The data.js file contains all the text that is displayed on the screen as well as a function to return the appropriate data to the main.js file.

MAIN.JS:

The main.js file is responsible for the flow of information on the page. The main.js will display information on the page, process user input, and pass user input to the pet_queries.js file.

PET_QUERIES.JS:

The pet_queries.js file is the primary event handler. User input and database output is processes in this file. This is where the XML requests are created and processed.

PHP | SQL:

The PHP files is where all the magic happens. This is where user input is interjected into a SQL query and the results returned to the page. The 16 PHP files are all in a similar format: create a database connection, get the user input from the form, pass the input into a prepared statement, and return the results. The source code for each PHP file is on the 'Sources' page along with a txt file which contains all the SQL queries.

CONCLUSION

The scope of this project includes the creation of the PetPortal Database and the page website. It includes a primary HTML file, three JavaScript files, and a sixteen PHP files. All data present in the database is 100% legitimate data. The plan is to continue to load the database with previous and future pet records to flesh the database out. As of now, there are a few apparent issues that will need additional consideration.

Scalability:

Pet Names: Pet names are currently treated as a unique value. As more pets are added, this may not always be the case. It is possible that multiple owners could have a pet with the same name. I will need to implement more advanced queries to narrow our scope down to the correct pet. We may even need an intermediate table which lists the owners name and the pet name. Eventually this may not be enough either, and we would need another semi-unique value to query to pull appropriate records. Many databases use the phone number as a pseudo-unique value.

Pet Status: It is assumed that all pets in the database are currently alive. I will need to add some functionality to remove dead pets from the queries. The simplest way is to add a column in the pet table and filter out if the pet is dead or not.

Database Updating:

Insert Into: There is not an easy way to insert new records into the database. While it may be sufficient for me to update the database manually, other users will need a better approach. Additional pages will need to be created which perform the necessary inserts into the database. Additionally, some consideration for data validation and foreign key constraint checking while updating the database from a web browser will make this a challenging design feature.