EPPS 6354.001 - Final Project Report:

A Veterinary Clinic Database

Tyler Merrill
The University of Texas at Dallas
tjm130330@utdallas.edu

Report Abstract

The purpose of this report is to present the rationale behind the development of a custom database and an application built to access it. Motivation and the schematic basis of the database model will be covered first. The design of the end-user application will be explicated, along with a focus on the tools used to accomplish this task. A brief summary and reflections on potential scalability options will then be considered.

Motivational Background

Community-tier veterinary clinics are institutions intended to serve the communities they inhabit by providing for the welfare of animals owned by local residents. However, in recent years there has been a shortage in veterinarians across the country (Kogut et al. 2024). Any tool that could help increase the efficiency of a given clinic could help ameliorate the effects of this shortage (Ouedraogo et al. 2023). Databases (DBs) offer the potential to reduce the administrative and logistical burdens and smooth the operations of clinics that adopt them. It is hoped that a well-designed DB will be demonstrably useful to small-scale organizations on account of their effectiveness in this regard.

The DB pursed in the project was envisioned to work for a hypothetical, community-sized veterinary clinic; most likely occupying a small-commercial sized office space with a staff of four to eight people in total (including clinical and non-clinical personal). This clinic is assumed to be a non-livestock clinic, with a focus on residential pets only - also known as a companion-animal clinic (SVM 2025). For modeling purposes, it was considered best to focus on the ‘appointment flow’ of animals brought into rotation with a hypothetical veterinary clinic’s services. This is because a DB tailored to the check-in process would be an intuitive and effective way to reduce logistic overhead, leaving more time among staff to focus on providing the intimate care their clients expect.

Users and Data Simulacrums

The end-users of this DB are the staff that will check-in clients visiting for an appointment, and the veterinarians (or appropriate clinical staff) conducting consultations with client's pets. The goals for these two use-cases diverge in terms of what they will requestion from the given DB.

The check-in process will necessitate appointment information such as the dates, times, clients, and reasons for any appointment being made. Consultation information will involve the recording of who undertook them, and what was done. In this context, it is assumed that taking biometrical readings will be the focus of appointment visits. Further steps after consultation, such as possible medical interventions, will be discussed in the last section concerning scalability potential.

For this project, all data was simulated. This includes names, addresses, check-in dates, vet salaries, and biometrics. Veterinarian earnings data was based off the national median annual salary as reported by the Bureau of Labor Statistics (BLS). This was $119,100 USD as of May of 2023 (BLS 2024). Red-blood cell count (x10^6/µL) and white-blood cell count (x10^3/µL), taken from ranges reported by the Clinical Pathology Laboratory of the Animal Health Diagnostic Center at the College of Veterinary Medicine at Cornell University (CUCVM 2024). Blood glucose levels (mg/dL) were based off of ranges reported by the American Animal Hospital Association (AAHA 2019). Blood pH levels were taken from ranges reported in Ochoa 2005.

Weight ranges were taken for the San Diego Humane Society (SDHS 2020). Respiratory and pulse ranges were extrapolated from data given by the Texas A&M AgriLife Extension (ALE 2009).

Random samples from these intervals were taken as actual measurements of animal biometrics and used as tuple values in the database.

E-R Schema

After deciding on who the users of the database will be, the next stage in database design is to lay out a schema of the objects in the real world to be described (entities) and the relationships among those objects that are pertinent for a database to manage (Ramakrishnan and Gerhrke 2002, 25-32).

To capture the full ‘appointment flow’ of a hypothetical community clinic, five entities were deemed necessary to capture: owners, their pets, staff (veterinarians and non-veterinarians), the appointments themselves, and the bio-records captured during those appointments.

The nine relationships of various cardinalities that were deemed important were: that between owners and their animals (‘owns’). A one-to-many relationship because an owner can own multiple animals, but a single animal can only have one owner. The relationship between owners and the appointment they make (‘makes’). Also, one-to-many. The relation between a staff member (vet) and an appointment they conduct (‘conducts’). This is one-to-many because a single vet will conduct many appointments, but it is an implicit constraint that appointments may only be conducted by one staff member.

Furthermore, the many-to-one relationship between a vet and the vital records they store ('records'). The one-to-one relationship between each appointment and the vital records that results from it ('tracks'). The one-to-many relationship between an animal and the records they have ('possesses'). This is because each animal may have multiple vital screenings, but each screening will only concern a single animal. The many-to-many relationships between an animal and vets ('has'). A single pet can be seen by multiple vets, and a single vet can see multiple pets. The one-to-many relationship between animals and their appointments ('attends').

Finally, the many-to-many relationship between owners and vets ('meets'). Because a single owner can meet with many vets over the course of many appointments, and a single vet will conduct appointments on the behalf of many owner's pets.

All entities possess a single-valued primary key, with the exception of 'appointment', whose primary key is a composite key composed of 'date' and 'time' attributes. The attributes were chosen to reflect the most pertinent facets of those objects for modeling the 'appointment flow' of a hypothetical clinic.

The E-R schema itself is visible in 'Crow's Foot' notation (Silberschatz et al. 2020, 316) in 'Figure 1', below:

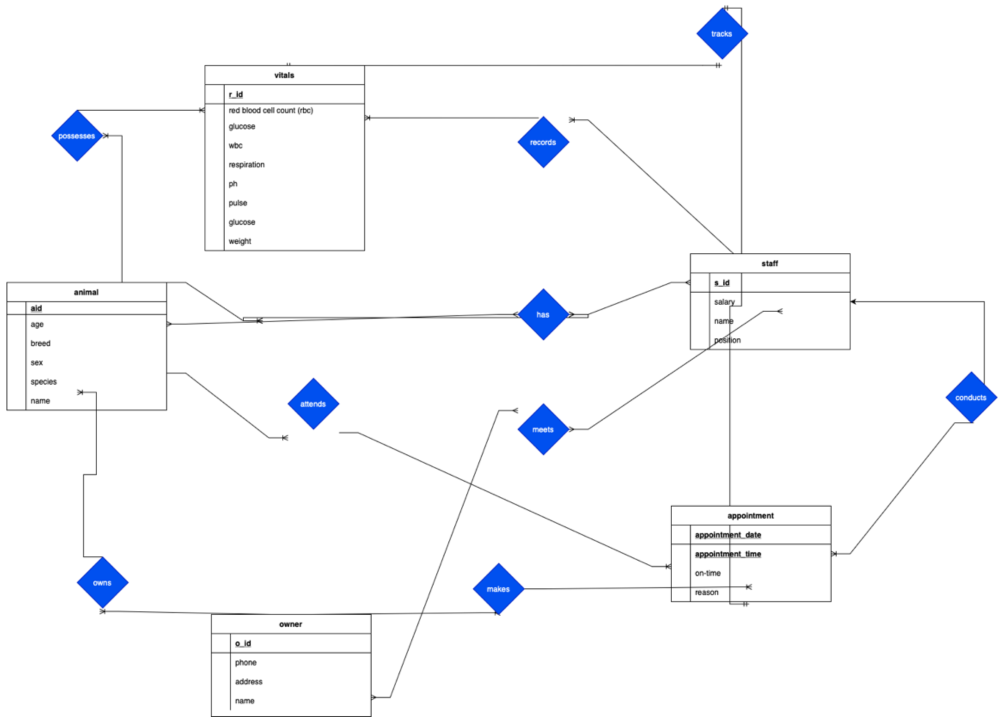


Figure 1

From the viewpoint of a hypothetical community clinic, the check-in receptionist would need to know what owner/animal pairing is coming in at what date/time. Additionally, having a brief reason for an appointment ‘on file’ would be helpful. Each of these facets were added as attributes in the appropriate ‘appointment’ entity. It is assumed that this hypothetical clinic does not accept walk-ins, and that only one pet can be covered by a single appointment. Such constraints can accord with the limited manpower of community-tier clinics.

For the clinical staff (veterinarians and veterinary technicians), being able to record and retrieve what vitals were read for a given appointment would be paramount. The ability to do this was implemented via attributes across the ‘appointment’ and ‘vital’ entities.

Relational Implementation

In order to translate this schema into a functional, physical, instantiation of a database accessible via PostgreSQL or any other query language, it was first necessary to recompose the E-R schema into a relational model. This next move functioned as a wrap up to the data-definition language (DDL) stage of the database design (Silberschatz et al. 2020, 42). Roughly, this involves defining the structure of the columns in the relational database. This was done using ‘SQLite’ and saved as a ‘.sql’ file. A copy of this is available in my GitHub repository entitled ‘merrill_project.sql’.

Perhaps the most significant changes made at this juncture was the addition of two cross-reference tables to better capture the model’s two many-to-many relationships. These two tables allow the relations between ‘animal’ and ‘staff’ and between ‘owner’ and ‘staff’ to be captured in the model. Note, the tables are titled ‘animal_vet’ and ‘owner_vet’ in the model. Their only attributes are comprised of the primary keys of their respective component tables.

All the other relationships between entity sets were captured as foreign key constraints in the appropriate relations. These relations can be seen mapped in ‘Figure 2’ below.

own		
PK	o_id	VARCHAR(255) NOT NULL
	name	VARCHAR(255) NOT NULL
	address	VARCHAR(255) NOT NULL
	phone	VARCHAR(255) NOT NULL

staff		
PK	s_id	VARCHAR(255) NOT NULL
	name	VARCHAR(255) NOT NULL
	salary	INTEGER NOT NULL
	position	VARCHAR(255) NOT NULL

vital		
PK	r_id	VARCHAR(255) NOT NULL
	weight	NUMERIC(5, 2)
	rbc	NUMERIC(5, 2)
	wbc	NUMERIC(5, 2)
	respiration	NUMERIC(5, 2)
	ph	NUMERIC(5, 2)
	pulse	NUMERIC(5, 2)
	glucose	NUMERIC(5, 2)
FK	s_id	VARCHAR(255) NOT NULL
FK	aid	VARCHAR(255) NOT NULL

animal_vet		
PK	aid	VARCHAR(255) NOT NULL
PK	s_id	VARCHAR(255) NOT NULL
FK	aid	VARCHAR(255) NOT NULL
FK	s_id	VARCAHR(255) NOT NULL

owner_vet		
PK	o_id	VARCHAR(255) NOT NULL
PK	s_id	VARCHAR(255) NOT NULL
FK	o_id	VARCHAR(255) NOT NULL
FK	s_id	VARCAHR(255) NOT NULL

animal		
PK	aid	VARCHAR(255) NOT NULL
	name	VARCHAR(255) NOT NULL
	age	INTEGER NOT NULL
	breed	VARCHAR(255)
	sex	VARCHAR(255) NOT NULL
	species	VARCHAR(255) NOT NULL
FK	o_id	VARCHAR(255) NOT NULL

appointment		
PK	appointment_date	DATE NOT NULL
PK	appointment_time	VARCHAR(255) NOT NULL
	reason	VARCHAR(255)
	ontime	VARCHAR(255)
FK	s_id	VARCHAR(255) NOT NULL
FK	aid	VARCHAR(255) NOT NULL
FK	o_id	VARCHAR(255) NOT NULL
FK	r_id	VARCHAR(255) NOT NULL

Figure 2

Each attribute in the model is atomic, meaning the possible inputs for every tuple are not composed of sub-units that can be broken down. At least not by query transactions alone. This means that the relational model is in first-normal form (Silberschatz et al. 2020, 371). In each of these tables, there are not any non-key columns which depend on a proper part (or subset) of their key column. Additionally, no non-key columns are *functionally* dependent upon other non-key columns in the 'own', 'staff', 'animal', 'vital', 'animal_vet' and 'owner_vet' relations. Which means you cannot derive one of the non-key column values when you know the value of any other (Painter-Wakefield 2023). Putting these relations in third-normal form.

In the 'appointment' relation, 'reason' and 'on-time' *are* fully dependent on the composite primary key ('appointment_date', 'appointment_time'). Because these non-key columns are non-transitively related to the composite primary key (e.g. they are not tied to the composite key via another dependency), this relation can be considered to be in third-normal form (Silberschatz et al. 2020, 346-348, Painter-Wakefield 2023). Therefore, this relational model is in third-normal form, or 3NF.

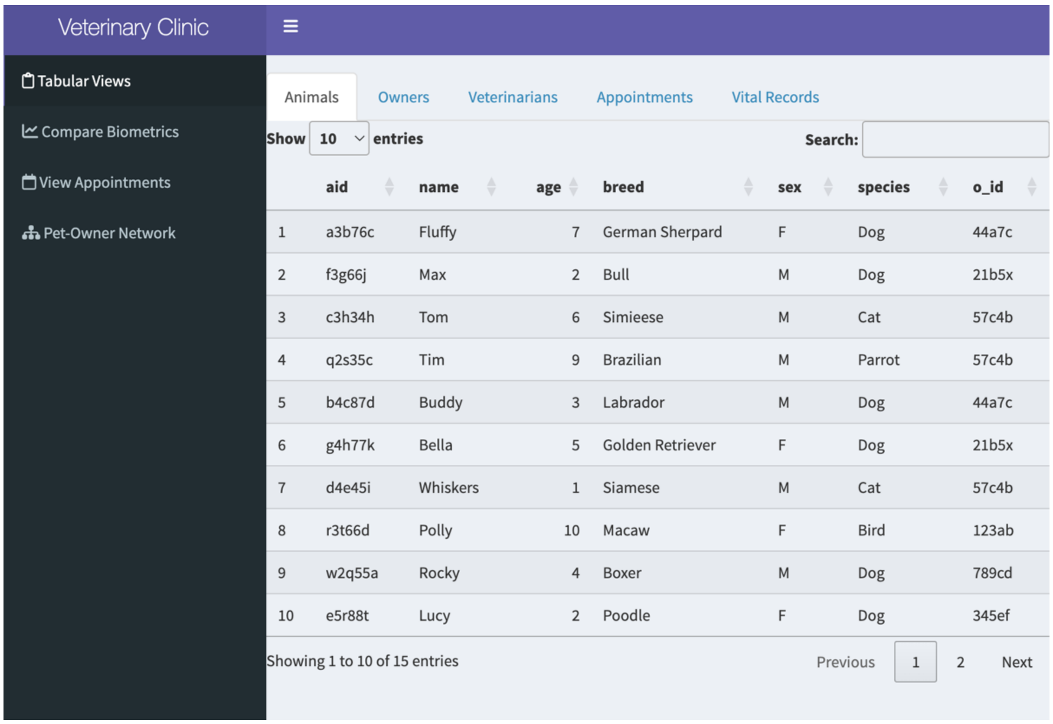
Finally, in order to use this relational database, fifteen 'animal' records (tuples) were simulated along with various vet, owner, vital, and appointment *n*-tuples. An additional relational schema generated in 'PGAdmin 4' (Frost et al. 2024) can be seen in 'Figure 3' below.

Figure 3

Application Design

In order to efficiently build an application that could allow users to easily select pre-specified 'views' of the database, it was decided (in accordance with syllabus specifications) to use 'R' in order to produce a 'shiny' (Chang et al. 2024) end-user application with pre-programmed assets and functions meant to compose an interactive program. Such a setup, involving an end-user UI accessing a database in order to navigate the database pursuant of certain user selections, closely represents a classic 'two-tier' network architecture: with the caveat that for this project, the database itself will not be hosted live and will be packaged as a '.db' (made using 'DB Browser for SQLite', Piacentini et al. 2024) file with the live application.

The final application allows end-users to access and search the entire relational database ('Figure 4'). This is perhaps the most fundamental collation of database storage and query potential. Any recorded information can be accessed and viewed in order to inform decision making and logistical planning, such as weekly staffing, overhead cost allocation (if different vets pay disproportionate shares of rent), etc. This was accomplished with the ‘RSQLite’, ‘DBI’, and ‘RPostgres’ packages (Müller et al. 2025; Müller 2024; Wickham et al. 2025).



	aid	name	age	breed	sex	species	o_id
1	a3b76c	Fluffy	7	German Sherpard	F	Dog	44a7c
2	f3g66j	Max	2	Bull	M	Dog	21b5x
3	c3h34h	Tom	6	Simieese	M	Cat	57c4b
4	q2s35c	Tim	9	Brazilian	M	Parrot	57c4b
5	b4c87d	Buddy	3	Labrador	M	Dog	44a7c
6	g4h77k	Bella	5	Golden Retriever	F	Dog	21b5x
7	d4e45i	Whiskers	1	Siamese	M	Cat	57c4b
8	r3t66d	Polly	10	Macaw	F	Bird	123ab
9	w2q55a	Rocky	4	Boxer	M	Dog	789cd
10	e5r88t	Lucy	2	Poodle	F	Dog	345ef

Figure 4

A graphic allowing users to compare select bio-metric markers (blood ph, rbc, wbc, glucose, weight, pulse) was also incorporated into the app (‘Figure 5’). This was accomplished by adding interactively to a ‘ggplot’ (Wickham 2015) graph with the ‘plotly’ package in ‘R’ (Sievert et al 2024). Selectable fields were also included to allow user to decide what bio-marker to choose to array the x and y axes and scale the scatter-dot size and hue.

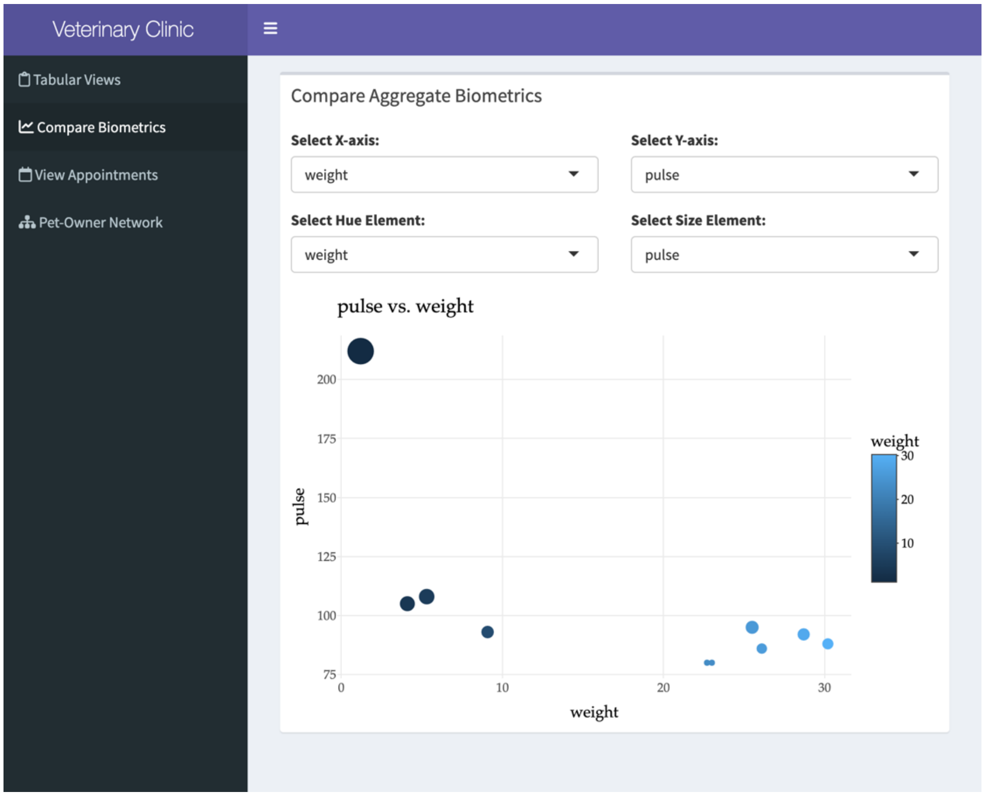


Figure 5

Appointment data was also visualized as a calendar composed of geometric tiles, similar to a 'heat-map' visual ('Figure 6'). Specifically, days were visualized as 'ggtile' tile objects in a 'ggplot' (Wickham 2015) graph with interactively provided via the 'plotly' library (Sievert et al 2024). This calendar allows a user to see quickly which days have appointments booked, and to view some information about them, such as the date/time and reason for the visit. This core of this visual was developed as the result of a 'conversation' with the chatbot model 'Gemini', hosted by Google (Google 2025). The transcript of this is available in my GitHub repository, entitled 'merrill_project_transcript_with_GEMINI'. Utilization of the 'dplyr' (Wickham 2023) and 'lubridate' (Spinu et al. 2024) packages was also made for this visual.

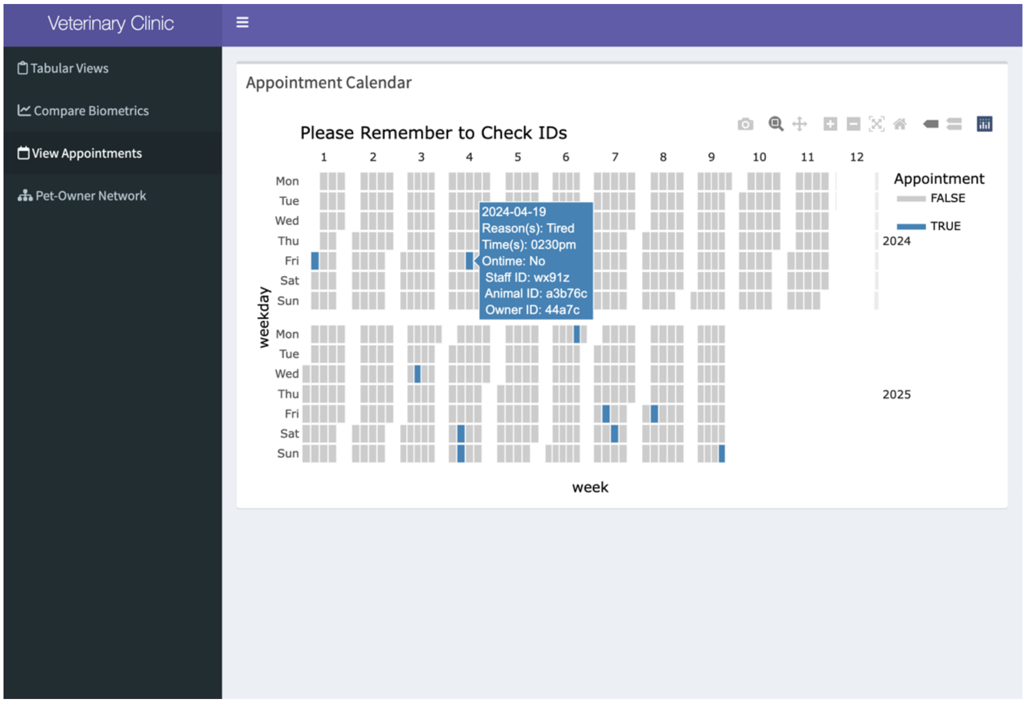


Figure 6

Finally, a network graphic was added to show visually which owner owned what animal(s). This was done with the 'igraph' (Csárdi et al. 2025) and 'ggraph' (Pedersen 2024) packages in 'R'. This simple visual ('Figure 7') can show at a glance the entire client distribution of patients the clinic serves.

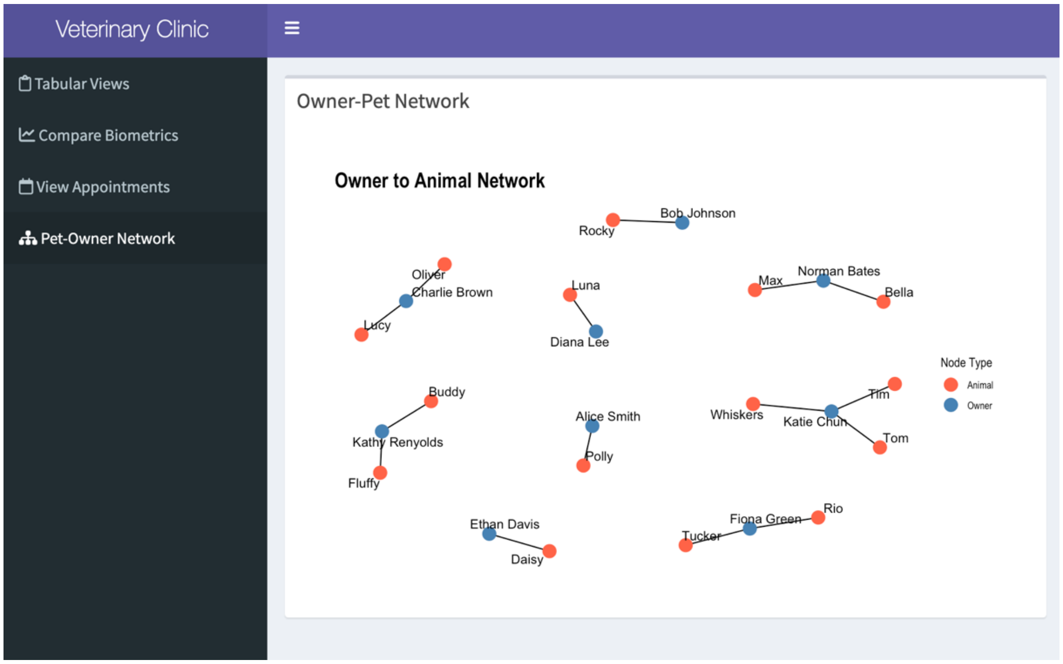


Figure 7

Additional functionality, such as the collapsible sidebar, was provided by the ‘shiny dashbaord’ package (Chang and Borges 2025). A copy of the ‘R’ script used to build the application is hosted on my GitHub repository, entitled ‘merrill_project_app.R’.

Scaling-Up

In the future, there are a number of explorable directions for capabilities expansion with both this database and its associated application. For one, this database is presumed to work for only a single community-tier veterinary clinic. Adding in a ‘branch’ relation would enable this database to cover a network of such clinics. This would allow important information to be transferred across locations, allowing owners and vets to attend different branches as their requirements may dictate.

Better bio-metric aggregation across species would be another avenue to fine-tune the application. Currently, only clinic-level biometrics can be observed with the bio-metric graphic.

Incorporating customization based upon species and gender comparisons would be a natural next step in integrating more realistic comparative visualization modality.

Another possibility would be to add more interactivity to the calendar visual. Perhaps emphasizing the breakdown of daily appointment flows with another visual layer. Yet another extension would be to make a mobile version of this application, although this possibility is currently constrained by the possibilities inherit in shiny-based applications. Although this could quickly change in the near future as developments are enrolled into the package.

As it is, the current version of the database/application pair provide a baseline functionality for information retrieval, managing, and comparison in the delimited context it was intended to do so. Its potential is nearly as malleable as that of other relational databases, which historically has proven to be quite robust to needs of users.

References

AAHA. 2019. “Blood Glucose Curves.” https://www.aaha.org/resources/2018-aaha-diabetes-management-guideline-for-dogs-and-cats/blood-glucose-curves/ (April 27, 2025).

ALE. 2009. *Normal Physiological Values for Select Animals*. Texas AgriLife Extension Service.

BLS. 2024. “29-1131 Veterinarians.” *Bureau of Labor Statistics*. https://www.bls.gov/oes/2023/may/oes291131.htm (April 26, 2025).

Chang, Winston, and Barbara Ribeiro Borges. 2025. “Shinydashboard: Create Dashboards with ‘Shiny.’” : 0.7.3. doi:10.32614/CRAN.package.shinydashboard.

Chang, Winston, Joe Cheng, Jj Allaire, Carson Sievert, Barret Schloerke, Yihui Xie, Jeff Allen, et al. 2024. “Shiny: Web Application Framework for R.” : 1.10.0. doi:10.32614/CRAN.package.shiny.

Csárdi, Gábor, Tamás Nepusz, Vincent Traag, Szabolcs Horvát, Fabio Zanini, Daniel Noom, and Kirill Müller. 2025. “Igraph: Network Analysis and Visualization.” : 2.1.4. doi:10.32614/CRAN.package.igraph.

CUCVM. 2024. “Hematology (Advia 2120).” *Routine Hemogram Reference Intervals*. https://www.vet.cornell.edu/animal-health-diagnostic-center/laboratories/clinical-pathology/reference-intervals/hematology-advia-2120 (April 27, 2025).

Frost, Stephen, Andrew Gierth, and Micheal Meskes. 2024. “PGAdmin 4.”

“Google ‘Gemini.’” 2025.

Kogut, Susanne, Meredith L Montgomery, and Julie K Levy. 2024. “The Nonprofit Veterinarian Shortage:Who Will Care for the Pets Most in Need?”

Müller, Kirill. 2024. “DBI: R Database Interface.”

Müller, Kirill, Hadley Wickham, David A. James, and Seth Falcon. 2025. "RSQLite: SQLite Interface for R." : 2.3.9. doi:10.32614/CRAN.package.RSQLite.

Ochoa, Luis Núñez. 2005. *Acid-Base Principles and Practical Interpretation in Small Animals*. World Small Animal Veterinary Association World Congress Proceedings, 2005. https://www.vin.com/doc/?id=6694579.

Ouedraogo, Frederic B., Peter Weinstein, and Sandra L. Lefebvre. 2023. "Increased Efficiency Could Lessen the Need for More Staff in Companion Animal Practice." *Journal of the American Veterinary Medical Association* 261(9): 1357–62. doi:10.2460/javma.23.03.0163.

Painter-Wakefield, Christopher. 2023. "3.3. Normalization — A Practical Introduction to Databases." In *A Practical Introduction to Databases*, https://runestone.academy/ns/books/published/practical_db/PART3_RELATIONAL_DATABASE_THEORY/03-normalization/normalization.html (April 28, 2025).

Pedersen, Thomas Lin. 2024. "Ggraph: An Implementation of Grammar of Graphics for Graphs and Networks." : 2.2.1. doi:10.32614/CRAN.package.ggraph.

Piacentini, Mauricio, Seong Tae Jeong, Justin Clift, and Chris Locke. 2024. "DB Browser for SQLite."

Ramakrishnan, Raghu, and Johannes Gehrke. 2002. *Database Managment Systems, Third Edition*. Third. McGraw-Hill.

SDHS. 2020. "Animal Size and Weight Chart." *San Diego Humane Society*. https://resources.sdhumane.org/Programs_and_Services/Adoption_Resources/Animal_Size_and_Weight_Chart (April 27, 2025).

Sievert, Carson, Chris Parmer, Toby Hocking, Scott Chamberlain, Karthik Ram, Marianne Corvellec, and Pedro Despouy. 2024. "Plotly: Create Interactive Web Graphics via 'Plotly.Js.'" : 4.10.4. doi:10.32614/CRAN.package.plotly.

Silberschatz, Abraham, Henry F. Korth, and S. Sudarshan. 2020. *Database System Concepts, Seventh Edition*. Seventh. New York, NY: McGraw-Hill.

Spinu, Vitalie, Garrett Grolemund, and Hadley Wickham. 2024. "Lubridate: Make Dealing with Dates a Little Easier." : 1.9.4. doi:10.32614/CRAN.package.lubridate.

SVM. 2025. "5 Types of Veterinarians and What They Do." *Veterinary Blog*. https://www.sgu.edu/blog/veterinary/types-of-veterinarians-and-what-they-do/ (May 2, 2025).

Wickham, Hadley. 2015. *Ggplot2 - Elegant Graphics for Data Analysis*. Springer.

Wickham, Hadley. 2023. "Dplyr: A Grammar of Data Manipulation."

Wickham, Hadley, Jeroen Ooms, and Kirill Müller. 2025. "RPostgres: C++ Interface to PostgreSQL." : 1.4.8. doi:10.32614/CRAN.package.RPostgres.