PROJECT REPORT DOCUMENT

DS 5420 - DATA MANAGEMENT SYSTEMS

PROJECT 2

Project:
Austin Animal Center Database

Team:
Soyeon Park
Meena Muthusubramanian

April 26, 2021

Introduction

Dataset

The dataset which we decided to use for this project was created by Austin Animal Center (AAC), which is the largest shelter in the United States that provides care and shelter to over 18,000 animals each year. The organization has accumulated the data containing intakes and outcomes of animals entering it from the beginning of October 2013. The data is available on the official City of Austin open data portal and are updated daily. The size of data is 30MB, consisting of 41 attributes and 79,672 rows. The dataset is available here. There are 3 datasets in the website, and we used ***'aac_intakes_outcomes.csv'***.

There are 2 main reasons for choosing this dataset. The first reason is that we wanted our application to be used for a good purpose. Austin Animal Center's goal is to place all adoptable animals in forever homes through adoption, foster care or rescue partner groups. If they can have an application allowing them to organize and manage animal's data in a database, they can operate the center better with a data-driven decision. Another reason is that the dataset consists of a variety of data types such as string, numbers and datetime. By using this dataset, we can get experience handling many diverse data types.

Application

This application is developed for employees working in the Austin Animal Center. They can save animals' traits information like type, breed, color and date of birth. Also, the application allows users to save animals' intake and outcome history. Users can not only insert new data, but also update and delete the data. Whenever users update the animal's information, the application saves the new and old data together in the auditing table. The purpose of this function is that they can find old data easily when they update wrong data by mistake.

Another good function of this application is that they can access the list of special animals easily; the animals which experienced their gender change, adopted animals and transferred animals. These animals require more caution and attention than other animals. The application has separate buttons to see the tables.

The last function of this application is summary. The employees in Austin Animal Center can see the number of total animals in the center, the number and ratio of animal type, and animal's outcome situation. This statistics function will help the employees monitor the situation of the center and take care of the animals better.

Final Implementation

Database Design

The normalization process was operated with 5 steps; moving unnecessary attributes to an extra table, 1st normal form, 2nd normal form, 3rd normal form, and placing together attributes sharing the same characteristics.

1) Moving unnecessary attributes to an extra table
Before starting normalization, we created a data table called 'no_use'. We moved all unnecessary attributes to the table. They are either ones in which all values are the same or ones overlapping with another attribute.

All values in the column 'count' are the same as 1. 'animal_id_outcome' overlaps with 'animal_id_intake'. Since the 'date_of_birth' column already includes the same information as 'dob_monthyear', 'dob_year', and 'dob_month', they are also moved to the 'no_use' table. Same thing happened to 'intake_monthyear', 'intake_year', 'intake_month', 'intake_hour', 'outcome_monthyear', 'outcome_year', 'outcome_month', and 'outcome_hour' for the same reason. As 'time_in_shelter' also overlaps with 'time_in_shelter_days', it was also moved to the 'no_use' table. We also moved 'age_upon_intake_age_group' and 'age_upon_outcome_age_group', because their values are unnecessary for our application. Finally, 'no_use' table includes 'age_upon_intake' and 'age_upon_outcome' as well, because they are duplicated rows.

2) 1st Normal form
We checked if all attributes follow the rules of 1st normal form. There are 3 rules for the 1st normal form. ① Each column should contain atomic values. ② A column should contain values of the same type. ③ Each column should have a unique name. Since all attributes in the mega table satisfy these three conditions, the table is already in 1NF.

3) 2nd Normal Form
To be in 2NF, a table should not have any partial dependency. Partial dependency means that a non-prime attribute is functionally dependent on part of a candidate key. In the table "in_out_mega", the combination of 'animal_id_intake' and 'intake_number' is the candidate key.

{animal_id_intake, intake_number}+ = {animal_type, breed, color, date_of_birth, intake_datetime, intake_weekday, age_upon_intake_years, age_upon_intake_days, sex_upon_intake, intake_type, intake_condition, found_location, outcome_datetime, outcome_weekday, age_upon_outcome_years, age_upon_outcome_days, sex_upon_outcome, outcome_type, outcome_subtype, time_in_shelter}

However, ‘animal_type’, ‘breed’, ‘color’ and ‘date_of_birth’ are functionally dependent on ‘animal_id_intake’ by itself, a part of the candidate key of this mega table.

{animal_id_intake}+ = {animal_type, breed, color, date_of_birth}

Therefore, we divided our table into two tables as follows:
- *Table A*: {animal_id_intake, animal_type, breed, color, date_of_birth}
- *Table B*: {animal_id_intake, intake_number, intake_datetime, intake_weekday, age_upon_intake_years, age_upon_intake_days, sex_upon_intake, intake_type, intake_condition, found_location, outcome_datetime, outcome_weekday, age_upon_outcome_years, age_upon_outcome_days, sex_upon_outcome, outcome_type, outcome_subtype, time_in_shelter}

Since there is no more partial dependency, both tables are in 2NF.

4) 3rd Normal Form
If a table has a transitive dependency, it violates 3NF. A transitive dependency occurs when a non-prime attribute is dependent on another non-prime attribute. As both tables do not have any transitive dependency, they are in 3NF as well.

5) Placing together attributes sharing the same characteristics
Even though the 2 tables are in 3NF, we decomposed the tables more to place together attributes sharing the same characteristics. We divided table B into 3 tables. Their candidate key is all the same as the combination of ‘animal_id_intake’ and ‘intake_number’.

The first table contains attributes related to only intake information. ‘animal_id_intake’, ‘intake_number’, ‘intake_datetime’, ‘intake_weekday’, ‘age_upon_intake_years’, ‘age_upon_intake_days’, ‘sex_upon_intake’, ‘intake_type’, ‘intake_condition’, and ‘found_location’ belong to this table. The second table has attributes regarding only outcome information. ‘animal_id_intake’, ‘intake_number’, ‘outcome_datetime’, ‘outcome_weekday’, ‘age_upon_outcome_years’, ‘age_upon_outcome_days’, ‘sex_upon_outcome’, ‘outcome_type’, and ‘outcome_subtype’ are in the second table. Since the ‘time_in_shelter_days’ attribute is related to intake and outcome together, we created a different table for the attribute. The last table consists of ‘animal_id_intake’, ‘intake_number’ and ‘time_in_shelter’.

Here are our final decomposed tables in 3NF. In the database 'austin_animal_center’, there are 4 tables as follows. The candidate keys for each table are in bold type.

- animal_info
{**animal_id_intake**, animal_type, breed, color, date_of_birth}

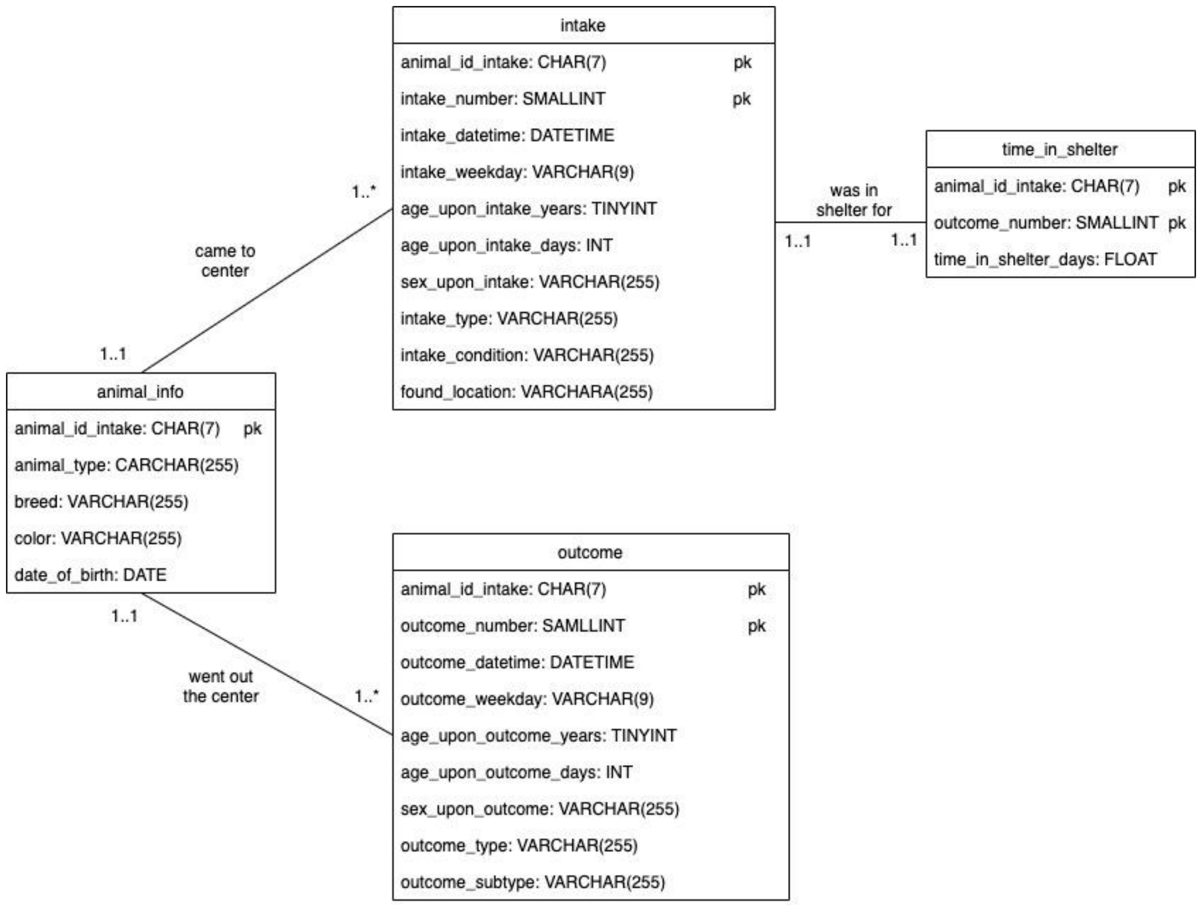
- intake
{**animal_id_intake, intake_number,** intake_datetime, intake_weekday, age_upon_intake_years, age_upon_intake_days, sex_upon_intake, intake_type, intake_condition, found_location}

- outcome
{**animal_id_intake, intake_number**, outcome_datetime, outcome_weekday, age_upon_outcome_years, age_upon_outcome_days, sex_upon_outcome, outcome_type, outcome_subtype}

- time_in_shelter
{**animal_id_intake, intake_number**, time_in_shelter_days}

* no_use
{animal_id_outcome, count, dob_monthyear, age_upon_intake_age_group, age_upon_outcome_age_group, dob_year, dob_month, intake_monthyear, intake_year, intake_month, intlae_hour, age_upon_intake, outcome_monthyear, outcome_month, outcome_hour, age_upon outcome, time_in_shelter}

Unified Modeling Language (UML)


intake
animal_id_intake: CHAR(7) pk
intake_number: SMALLINT pk
intake_datetime: DATETIME
intake_weekday: VARCHAR(9)
age_upon_intake_years: TINYINT
age_upon_intake_days: INT
sex_upon_intake: VARCHAR(255)
intake_type: VARCHAR(255)
intake_condition: VARCHAR(255)
found_location: VARCHARA(255)
time_in_shelter
animal_id_intake: CHAR(7) pk
outcome_number: SMALLINT pk
time_in_shelter_days: FLOAT
was in shelter for
1..1
1..1
came to center
1..*
1..1
animal_info
animal_id_intake: CHAR(7) pk
animal_type: CARCHAR(255)
breed: VARCHAR(255)
color: VARCHAR(255)
date_of_birth: DATE
1..1
went out the center
1..*
outcome
animal_id_intake: CHAR(7) pk
outcome_number: SAMLLINT pk
outcome_datetime: DATETIME
outcome_weekday: VARCHAR(9)
age_upon_outcome_years: TINYINT
age_upon_outcome_days: INT
sex_upon_outcome: VARCHAR(255)
outcome_type: VARCHAR(255)
outcome_subtype: VARCHAR(255)

Data used for testing

We used the Austin Animal Center Shelter Intakes and Outcomes data for the application. Since the data is not too big, we inserted all data including 41 attributes and 79,672 rows into our database, except one row. The row's primary key was the same as that of another row, so it prevents the primary key in the table from being unique. For this reason, we deleted the row.

For test data, we selected 2 data. First test data is animal_id that already exists in the database, another test data is different animal_id that does not exist in the database yet. For the first situation, we used 'A178569'. Since this data already exists in the database, when we search the ID in the [Animal Info Table], it shows a row displaying the animal's type, breed, color and date of birth. Since animal ID is the primary key for the table, we can't add another row of which animal_id is 'A178569' to [Animal Info Table]. When we insert new information in either [Intake Info Table] and [Outcome Info Table] using the animal ID 'A178569', the intake number is offered automatically, in this case the number is 2, using the maximum number of pre-existing numbers. After inserting a new row to [Intake Info Table] and [Outcome Info Table], we can see the data in the merged datatable [Intake & Outcome Table]. We can also update 'A178569's data in [Animal Info Table]. After that, we can see the old data and new data together in the table [Animal Change History].

For the case when we use an animal's ID that does not exist in the database yet, we used animal ID 'A090909'. Since the data does not exist in the database, when we search the ID in the [Animal Info Table], it shows an error message saying that no rows returned for the search parameters. For this case, we can add the ID in [Animal Info Table]. After registering the animal ID in [Animal Info Table], we can insert a new row in either [Intake Info Table] and [Outcome Info Table]. As both tables do not have any data related to 'A090909' yet, the system gives 1 for 'intake_number' column automatically when we insert a new row using the animal ID. After that, any row related to 'A090909' can be updated or deleted like 'A178569' can do. Also, whenever there is an update in any row, we can see the old data and new data together in an auditing table.

Summary of implemented use cases

- Search animal's trait/intake/outcome information
- Insert/update/delete animal's bio/intake/outcome data
- View specific groups of animals
  (animals which experienced their gender transformation, adopted animals and transferred animals)
- View data change history
  (animal traits, intake and outcome table)
- View statistics related to the animals in the center

Illustration of Functionality

Our application has 7 major functionalities : search, insert, update, delete, display, audit, and summary and below are the use cases for each functionality.

Pre-conditions :

- Database server should be up and running
- Backend NodeJS and frontend application should be started by following the steps in the readme document.

1. **Use case for ‘Search’**

   Use case name : Search Animal Data

   Search works for the following tables with the given search parameters

Table	Search Parameter
Animal Info Table	Animal ID
Intake Info Table	Animal ID/Intake Date
Outcome Info Table	Animal ID/Outcome Date
Intake & Outcome Table	Animal ID/Outcome Date/Intake Date

   Step 1 : Click on any of the above table from Tables Menu

   Step 2 : Enter existing Animal ID/Outcome Date/Intake Date in search bar

   - Application validates the input
   - Application shows details corresponding to the particular Animal ID

   Enter non-existing Animal ID/Outcome Date/Intake Date in search bar

   - Application returns an error message

   Sample Search Examples :

   - [Tables] > [Animal Info Table]

Type ‘A006100’ in the animal ID search bar and enter	Animals ID, type, breed, color and date of birth is displayed
Type ‘A090909’ in the animal ID search bar and enter	ERROR BOX is displayed : No rows returned for the search parameters

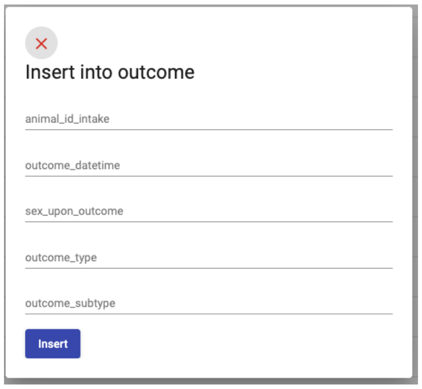
   - [Tables] > [Intake Info Table]

Type ‘2015-05-20’ in the intake date search bar and enter	Details of animal intake for the particular date is displayed
Type ‘1998-09-10’in the intake date search bar and enter	ERROR BOX is displayed : No rows returned for the search parameters

2. **Use case for 'Insert'**

Use case name : Insert animal information
Insert works for the following tables

- Animal Info Table
- Intake Info Table
- Outcome Info Table



Insert into outcome

animal_id_intake

outcome_datetime

sex_upon_outcome

outcome_type

outcome_subtype

Step 1 : Click on any of the above table from tables menu
Step 2 : Click on insert row button and enter the information regarding the animal

Sample Insert Examples :

- [Tables] > [Animal Info Table]

Inserting new data	animal_id_take : C010101 animal_type : Dog breed : Husky color : White Date_of_birth :1996-01-01	Data is successfully entered and can be viewed by searching the same animal ID in the search bar
Inserting existing data	animal_id_take : A006100 animal_type : Dog breed : Husky color : White Date_of_birth :1996-01-01	ERROR BOX displays Animal ID already exist

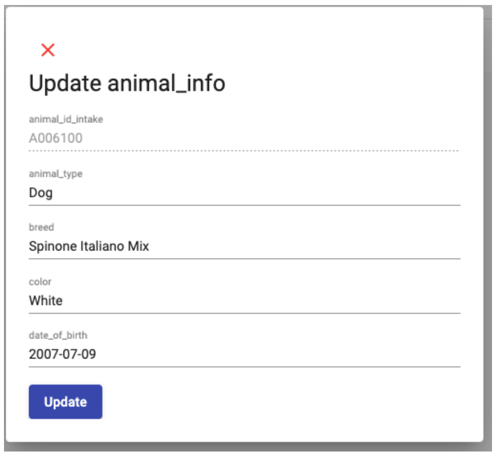
- [Tables] > [Outcome Info Table]

Insert data of already existing animal	animal_id_take : C010101 outcome_Datetime : 2021-01-01 Sex_upon_outcome : Male Outcome_type : Transfer Outcome_subtype : Friend	Data is successfully entered and can be viewed by searching the same animal ID in the search bar

3. **Use case for 'Update'**

Use case name : Insert animal information
Update works for the following tables

- Animal Info Table
- Intake Info Table
- Outcome Info Table



Update animal_info

animal_id_intake: A006100

animal_type: Dog

breed: Spinone Italiano Mix

color: White

date_of_birth: 2007-07-09

Step 1 : Click on any of the above table from tables menu
Step 2 : Click on the three dots next to each row where update is present
Step 3 : Click on update as the update form pops up
Step 4 : Certain attributes cannot be updated because it's a primary reference.

Sample Search Examples :

- [Tables] > [Animal Info Table]

Click on three dots and click update corresponding to animal id A006100	Update form is displayed
Change the animal color to 'White'	Search for the animal id in the search box and the updated value is displayed.

- [Tables] > [Update Intake table]

Click on three dots and click update corresponding to animal id A141142	Update form is displayed
Change the found location to 'TN''	Search for the animal id in the search box and the updated value is displayed.

4. **Use case for 'Delete'**

Use case name : Delete animal information
Delete works for the following tables

- Animal Info Table
- Intake Info Table
- Outcome Info Table

Step 1 : Click on any of the above table from tables menu
Step 2 : Click on the three dots next to each row where update is present
Step 3 : Click on delete

- [Tables] > [Delete Outcome table]

Click on three dots and click delete corresponding to animal id A165752	The entire row of values corresponding to the animal ID is deleted.

- [Tables] > [Animal Info Table]

Click on three dots and click delete corresponding to animal id A191351	The entire row of values corresponding to the animal ID is deleted.

5. **Use cases for viewing**

- 'See all' button helps to view the entire table with all the data. It is present in all the tables.
- There are the views under 'Views' in the menu which we created for specific functionalities which help to view the data of animals who have changed gender, adopted list and transferred animals list. We cannot perform any operation but it's for a viewing purpose.
- There are also the audit functionalities under Audit in the menu which we created which helps to initiate a trigger and store the change history for the animal intake and outcome.
- 'Summary' menu queries through the backend and displays useful animal information in a tabular format and also calculates percentage.

Summary Discussion

Our project is fully complete with many functionalities which makes use of Stored Procedures, Views, Triggers and includes DML operations. Fetching the entire data in one page with around 70k rows was hard and figuring out backend pagination was done with select statements and including limit parameters with offset and page limit. This was done to enhance the user experience and clear application presentation.

Since back-end and front-end operations go together, we initially wanted to make sure both the developers split the work equally and spent equal time to stay connected to help each other if there were any roadblocks. We split the task into smaller subtasks which helped in co-ordinating and met weekly deadlines. Finalizing the dataset, thinking through the structure of UML, finalizing the functionalities and creating weekly progress reports were done together. Once we were ready to start working the data , person 1 took the responsibility to load in the data, create tables, decompose and write queries regarding advanced features in MySql while the other person started working on establishing the front-end connection, installing dependencies, identifying front-end framework and importing the queries via back-end. By constantly reminding the team of the objectives we were trying to achieve as a team, we fairly achieved and developed the application with equal contribution. We always made sure we were on the same page, shared and updated our work in real time and by acknowledging each other's work, we did justice to what we were doing both individually and as a team. We acknowledge each other's work and agree that it was a fair split.