W4111.001-Introduction to Databases
Spring 2021
Databases Final Project Proposal
Due February 10, 2021 at 11:59 PM
Group number 31

Group Members:

- Matthew Duran (md3420@columbia.edu)
- Meghan Shah (ms5767@columbia.edu)

Application:

- First Proposal: Coronavirus and effect on society
- Final Proposal: Daily operations of a restaurant

First Proposal:

- Our first proposal was trying to see how coronavirus has affected society over the past year, and we found very interesting datasets that would have been cool to populate. However, the overall schema was getting very convoluted, so with the advice of some TAs, we have pivoted to modeling a restaurant.

Final Proposal:

- As avid brunch enthusiasts, we are excited to be able to pivot from more of a research project to a project that models a real world enterprise. This project will be fun and more manageable, allowing us to focus on the database creation and front end development. We are modeling our database after Meghan's favorite Bay Area restaurant, Plow (http://www.eatatplow.com/). With this restaurant as our basis, we plan on modeling its daily operations. At a high level, we are going to track the restaurant's revenues and costs. We have three aggregated relationships in our design: labor costs, food costs, and restaurant revenue. This data will be collected by transaction. Our main entity that we will have to focus on is the orders aggregate, because if we don't generate any revenue then the business will not be able to survive. Then, our next area of focus is supplying the store with the food. Thus, we will model our ingredient stock, where we source our food from, and at what price we acquire the items. Lastly, we will model our labor costs by tracking employees, so we can appropriately staff our restaurant to see how much they work (so they can get paid). By focusing on these three pillars, we will be able to track how a restaurant operates, albeit on a much smaller scale.

<u>Total Entities (8 total):</u>

- Menu items
  - This table will contain all of the menu items that we will sell to customers.
- Ingredients
  - These are the ingredients that comprise the various recipes that we offer.
- Customer
  - Customer table that is used within orders and for signing up for mailing lists.
- Mailing list
  - This table will contain the emails for all of the customers that have signed up for our mailing lists.
- Employee
  - This table will house all of the pertinent information related to employees
- Hours of operation
  - Store hours.
- Enterprise Finances
  - This table will house each transaction from expenses (ingredients and labor) to revenues (food sales). Each transaction will be tied to a unique transaction id.
- Farmers
  - These are the people that we source our ingredients from and we will list the prices of each ingredient.

<u>Total Relationships (8 total):</u>

- Orders
  - Many to many relationship, with total participation on the customer side. This is the case because all customers come to a restaurant to order menu items.
  - This relationship taken in aggregate, allows us to track our food sales using additional attributes that constitute the relationship.
  - Flow of relationship: customer orders menu items.
- Uses
  - Many to many relationship, with total participation on the menu side. This is the case because all menu items use ingredients.
  - Flow of relationship: menu items use ingredients.
- Buys from
  - Many to many relationship, with total participation on the farmer side. All ingredients are sourced from local farmers.
  - This relationship taken in aggregate, allows us to track our ingredient expenses using additional attributes that constitute the relationship.
  - Flow of relationship: ingredients are bought from farmers
- Registers for
  - Many to one relationship, with total participation on the mailing list side. Each unique mailing list tuple will map to a particular customer since they signed up for it.

        - Here we have to assume that each customer is unique, and that we can track each customer. (This is one of the real world constraints that have been hard to model).
- Shift schedule
    - Many to many relationship between employees and the hours of operation.
    - All of the days in the hours of operation will need to be included in this relationship since we operate 7 days a week.
    - This relationship taken in aggregate, allows us to track our labor expenses.
- Food sales
    - One to one relationship with total participation on the orders side. Given a unique transaction id in the orders relationship, we will be able to pull in data from that aggregate allowing us to track food sales and place them in our enterprise finances table.
    - This is fun because we could see which items are the best sellers, what total revenue is, and answer more interesting queries.
- Ingredient expenses
    - One to one relationship with total participation on the buys from side. Given a unique transaction id in the buys from relationship we will be able to track all of the ingredient transactions and place them in our enterprise finances table.
    - We will then be able to see which items we spend the most money on, which farmers take our business, and much more.
- Labor expenses
    - One to one relationship with total participation on the labor side. This relationship will be able to pull in labor costs, per day, for all employees using a unique transaction id and place them in the enterprise finances table.
- Using this labor model, we can tie together all of our finances, and calculate profits and losses over the whole enterprise.

<u>Modelling Challenges & Constraints:</u>

- We tried really hard to model the dynamic nature of restaurants, such as: how menu items change seasonally, and if the component ingredients are in stock. However, these types of relationships require embedding SQL functions into tables (which we will cover very late in the semester).
    - We solved this problem by implicitly assuming that customers can only order items that were in stock. Thus, if an order was made, the item was in stock.
- We wanted to be able to track customers by different groups (adults, children, etc.) and employees (waiter, chef, hostess, etc.) but these types of inheritance relationships were hard to manage. The E-R diagram became very cumbersome and confusing, so we kind of grouped most of those subclasses together.
    - This is a constraint that we could not manage.
- Since we are modeling on by transaction, we attempted to associate waiters with a given order. This relationship would allow us to track tips, but again added to the complexities of the design.

    - This is a constraint that we could not manage.
- We found it very difficult to track an order in its entirety, thus we will break the order down into an itemized list. However, this rule has an exception, if a customer orders more than one of the same item on a list.
        - e.g., a customer orders two servings of guacamole and one burrito
            - Transaction one: two servings of guacamole
            - Transaction two: burrito
- Tracking customers is hard, because in the real world we do not need to track them to have them order food.
    - To fix this we created an orders relationship, where we have a purchase id tied to a transaction. We also place an assumption that each customer tells us their name.
    - If we had an online ordering or reservation system we would need to rework the diagram.

Part 3 Front-End Option:

- We plan on doing the front-end option, which will be a webpage that will allow us to query our database and interact with different slices of our data.
    - Users should be able to query the database and answer many questions regarding our restaurant, such as:
        - Which menu items are in stock.
        - profits/expenses on different days.
        - Look at employee schedules, and see how much they are owed.
        - What food items are most popular, from a total price perspective and a total order perspective.
    - These will most likely be static charts that use predefined queries for production, where users can choose from a drop down menu.
    - The goal would also be to serve dynamic content.
        - We will refine this plan much more when we get to this part of the project/semester and have more SQL/Python experience.

Data Plan:

- We will look to Plow's menu as inspiration for our different dishes, drinks, hours of operation, price offerings, etc.
- For other entities, like employees, jobs, farmers (ingredient source), etc., we will just fabricate the data with entries in the database. We will provide sufficient rows so that we can provide different data slices in the front end option.
- We will also have to add in a few customer sales, employee shifts, and ingredient transactions, so that we can display some daily sales and expenses to track how our business is doing.

Contingency:

- We don't plan on dropping the course, but if something were to happen I would downgrade the web front-end option to host only static content. Meaning, given a drop down menu, chose the slice of data that you would like to see. I feel like the dynamic part would require an inordinate amount of error checking and development to tackle solo.
- The DB will then be downgraded to 3-7 entity sets and 3-7 relationship sets.

PDF Documents Attached Separately:

- Entity - Relationship Diagram
- SQL Schema