CS 340-400

Brittany Dunn

Project Step 1: Final

Project + Database Outline

Feedback From Peer Reviewers:

Comments:

Outline makes sense and is understandable why you would create a database. There are enough entities but I don't see a reason why addresses need to be their own entity. It would probably be fine to just attach it to the restaurant entity. Review entity is somewhat confusing. If a customer enters a restaurant, will they automatically be set to like it? A record could be created if a user clicks a like, with a person id and a restaurant id in the record. If recording both likes and dislikes, the boolean attribute can stay in the record as well.

Jonathan Abantao, Oct 1 at 11:18pm

Hi Brittany, It is strange my database is based on menus for restaurants and the other student I was assigned is also doing a food related database. I am not sure of your intentions for this database but for the sake of the project you could probably simplify this a bit. You could have the restaurants location reduced to latitude and longitude or just use a generic "location ID". If this is a database to track restaurants that the user likes perhaps a personal note or favorite meal would be helpful to remember. I know I rarely know what I ordered but just that I had a good meal.

Jacob Powers, Oct 2 at 4:36pm

Actions Based on Feedback:

I am leaving location as an entity because location is a thing which is made up of attributes and will be stored together in a table. Another reason I am leaving location as an entity is location should be unique and restaurants cannot be at the same location. The restaurant entity table will show the one-to-one relationship between the location entity and restaurant entity. In this table I would use a location id (as mentioned by the reviewer) and restaurant id. I do not want to change location to latitude and longitude as I want location to be something that can easily be looked up and used by a customer for locating a restaurant.

I tried to clarify how the review process would work in the project outline and I changed the review to not have a default choice. If a customer selects a restaurant to review, they will then choose between a like (1) and dislike (0), but their entry cannot be null. This database is less to track a customer's restaurants they like and more to help customers choose a restaurant based on their cuisine preference, location, and other customer reviews.

Upgrades to Draft Version:

I added more detail to the project outline to better explain how I would like this database to work as there seemed to be a bit of confusion from the reviewers. I also changed reviews to an attribute instead of an entity because I realized it was an attribute of the relationship between restaurants and customers. While working on project step 2 I realized there was not a Boolean data type, so I changed this to a bit which cannot be null.

Project Outline

This database will represent a review and information system for restaurants and their customers. Customers can review restaurants which will generate a rating for the restaurant. Customers must add their information in order to add a review. Restaurants will have information about their name, location, cuisine, and rating.

This database would work as follows:

- customers would create an account with their information (name, email, birthdate, and preferences)
- restaurants would be added to the database by customers and would include the restaurant information (name, location, rating, and cuisine)
    - name of restaurant would be entered by customer
    - location would be added when a restaurant is added, but it cannot be the same as a location already in the database
    - rating would default to 0 at creation because no reviews exist
    - cuisine can be chosen from the already existing cuisines or a new cuisine type can be created
- Customers can select an existing restaurant and choose either like or dislike. Restaurants are reviewed after creation and never during. During creation the rating (which is made up of reviews) is defaulted to 0.
- Customers would be able to find all restaurants matching their cuisine preference or a certain rating
- When a customer looks up a restaurant they would receive its information including location, rating (which is the average like rate from customers who have left reviews), and cuisine type

The focus of this database is to allow customers to give their opinion of a restaurant in the form of a like or dislike which in turn will help other customers choose a restaurant. Customers searching for a restaurant narrow down their options by location, cuisine preferences, and a restaurant's rating.

Database Outline

Entities and Attributes:

- **Restaurants**
    - **Id:** This is an auto-incremented, not null, and unique number used to identify a restaurant. This is the primary key.
    - **Name:** This is the restaurant's name. This is a string with a max of 200 characters. It cannot be blank or null.
    - **Rating:** This is an average rating based on customer reviews. This is a decimal number to two decimal places to represent a percentage. It cannot be blank or null. The numbers will be calculated as an average of likes (value of 1) and dislikes (value of 0) based on the number of customers who left reviews. When a restaurant is first added to the database the default is 0.

    - **Location:** This is the physical location of the restaurant. This will be an id of the location entity. Only one restaurant can be at any given location, so the id must be unique. Location cannot be null or blank. (This is a foreign key)
    - **Cuisine:** This is the cuisine which the restaurant serves. This will be an id of the cuisine entity. The cuisine does not have to be unique. (This is a foreign key)
- **Location**
    - **Id:** This is an auto-incremented, not null, and unique number used to identify a location. This is the primary key.
    - **Address**
      This is the physical address of a restaurant.
        - **Street:** This is the street name of the restaurant. This is a string with a max of 255 characters. This cannot be null or blank.
        - **Suite number:** This is the suite number of the restaurant. This can be blank.
        - **City:** This is the city where the restaurant is located. This cannot be null or blank. This is a string with a max of 100 characters.
        - **State:** This is the state abbreviation where the restaurant is located. This cannot be null or blank. This is a string with a max of 2 characters.
        - **Zip code:** This is the zip code of where the restaurant is located. This is an integer with 5 numbers. This cannot be blank or null.
- **Customers**
    - **Id:** This is an auto-incremented, not null, and unique number used to identify a customer. This is the primary key.
    - **First name:** This is the customer's first name. This is a string with the max of 100 characters. This cannot be blank or null.
    - **Last name:** This is the customer's last name. This is a string with the max of 100 characters. This cannot be blank or null.
    - **Email:** This is the customer's email. This is a string with the max of 100 characters. This cannot be blank or null. This must be unique.
    - **Birthdate:** This is the customer's birthdate. This is a date in the format YYYY-MM-DD.
    - **Preferences:** This is the customer's cuisine preference. This will be an id of the cuisine entity. Customers are not required to have a cuisine preference. (This is a foreign key)
- **Reviews (This is an attribute of the relationship between restaurants and customers. It will be used to derive the rating attribute of restaurants)**
    - **Like/Dislike:** This represents the customers review of a restaurant. This is a bit. 1 for like and 0 for dislike. This cannot be null. To clarify reviews are only likes or dislikes they will not include comments.
- **Cuisine**
    - **Id:** This is an auto-incremented, not null, and unique number used to identify a cuisine. This is the primary key.
    - **Type:** This is a description of the type of food a restaurant serves. This is a string with a max of 100 characters.

Relationships:

- **Restaurants serve customers:** Many restaurants have many customers. This is a many-to-many relationship
    - **And these customers generate reviews:** This is an attribute of the restaurant-customer relationship
    - **And these reviews create a rating:** A restaurant's rating is calculated from reviews left by its customer
    - **Example of this relationship:**

Restaurant ID	Customer ID	Review
ABC	123	1
DEF	123	0
ABC	456	0

Restaurant- ID: ABC **Rating:** 50%

- **Restaurants are at locations:** Every restaurant has a single location which belongs to it. This is a one-to-one relationship.
- **Restaurants specialize cuisine:** A restaurant serves one cuisine, but a cuisine can be served at many restaurants. This is a one-to-many relationship.
- **Customers have cuisine preferences:** Every customer has a single cuisine preference, but a cuisine can be a preference for many customers. This is a one-to-many relationship.

CS 340-400

Brittany Dunn

Project Step 2: Final

ERD + Schema +DDQ

Feedback by the peer reviewer:

The best peer review for ERD would answer all of the following questions.
Are the attributes for each entity in the ERD same as that described in the database outline?
Yes, the attributes in the ERD match those in the outline.
Is the participation of entities in the relationships same as that described in the outline?
Yes, the participation of restaurants to customers, restaurants to locations, and restaurants to cuisine all have correct participation. I am confused about the customers to cuisine relationship. If a customer will have only one preference, then must they have a preference? If a customer doesn't have to have a preference, then the ERD is correct.
Is the cardinality of entities in the relationships same as that described in the outline?
Yes, the cardinality of the entities all match.
Is there something that could be changed/improved in the E R Diagram and/or the overall database design?
No, I think it looks really well thought-out. I particularly like how the customers and cuisine entities are linked logically.

The best peer review for a Schema would answer all of the following questions:
Are the relationship tables present where required and correctly defined, when compared with the database outline?
Yes, all the relationship tables are present and correctly defined.
Are foreign keys present where required and correctly defined, when compared with the database outline?
Is the location attribute in the Restaurants entity a foreign key? Even though it is not explicitly defined as one in the outline, it seems like it would be.
Do the entity attributes match those described in the outline?
Yes, all the entity attributes match those in the outline.
Is there something that could be changed/improved in the Schema and/or the overall database design?
No, I think it is a very clean and efficient design that is well integrated.

The ideal peer review for a DDQ file would answer all of the following questions:
Is the SQL file syntactically correct? This can be easily verified by importing/copy-pasting it in phpmyadmin. (Do not forget to take backup of your own database before you do this!)
Yes.
Are the data types appropriate considering the description of the attribute in the database outline?
Yes.
Are the foreign keys correctly defined when compared to the Schema?
Yes.
Are relationship tables present when compared to the ERD/Schema?
Yes, reviews_restaurants_customers is present and populated.

David Rider, Oct 10 at 7:23am

Hi Brittany, the changes you made based on previous reviewers were good. I like your ERD and Schema diagrams, thanks for using blue arrows (made it easier to see and distinguish from table borders). My only confusion is with the reviews. Once a restaurant entry is created, you say the review value defaults to 0. However, it's also stated that the review is based off 1 (like) vs 0 (dislike), so wouldn't it be unfair to new restaurants that haven't had any reviews yet? Unless you somehow "subtract" that initial 0 value from the average rating. Your SQL queries look nice, thanks for spacing them out and using several samples/examples. I like your use of the AVG function. Only thing I noticed was you don't drop any tables if it exists, so if you were to import this twice, some tables may be overwritten while others aren't. I was thinking perhaps you could put in DROP TABLE IF EXISTS so that you have a fresh table every time you import this file in.

Hyung Jun Kim, Oct 10 at 10:48am

Actions based on the feedback:

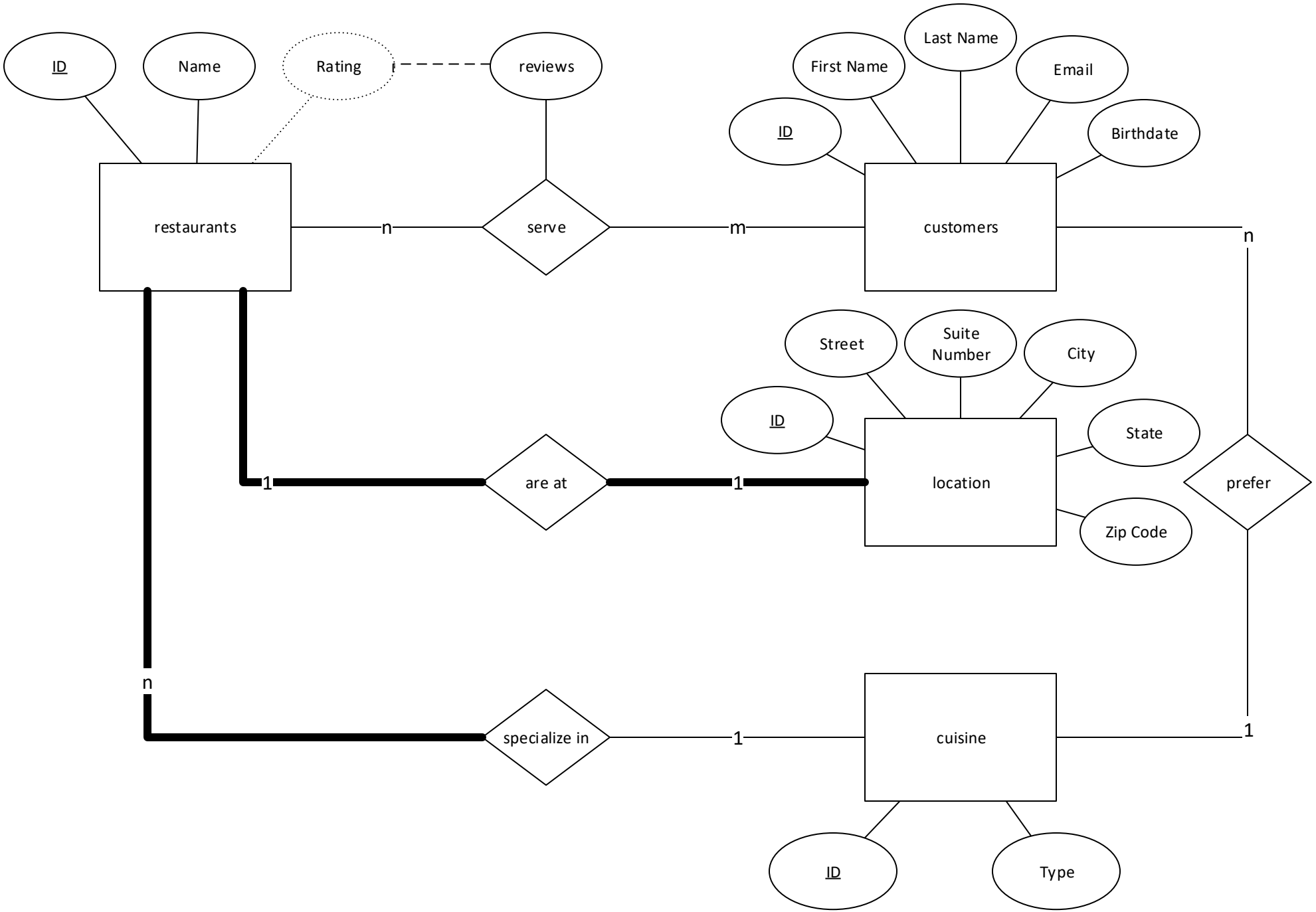
I went back in the outline and clarified that it is not required for a customer to have a cuisine preference. I made no changes to the ERD since it is already correct. I went back in the outline and labeled all the foreign keys, so it would be clearer.

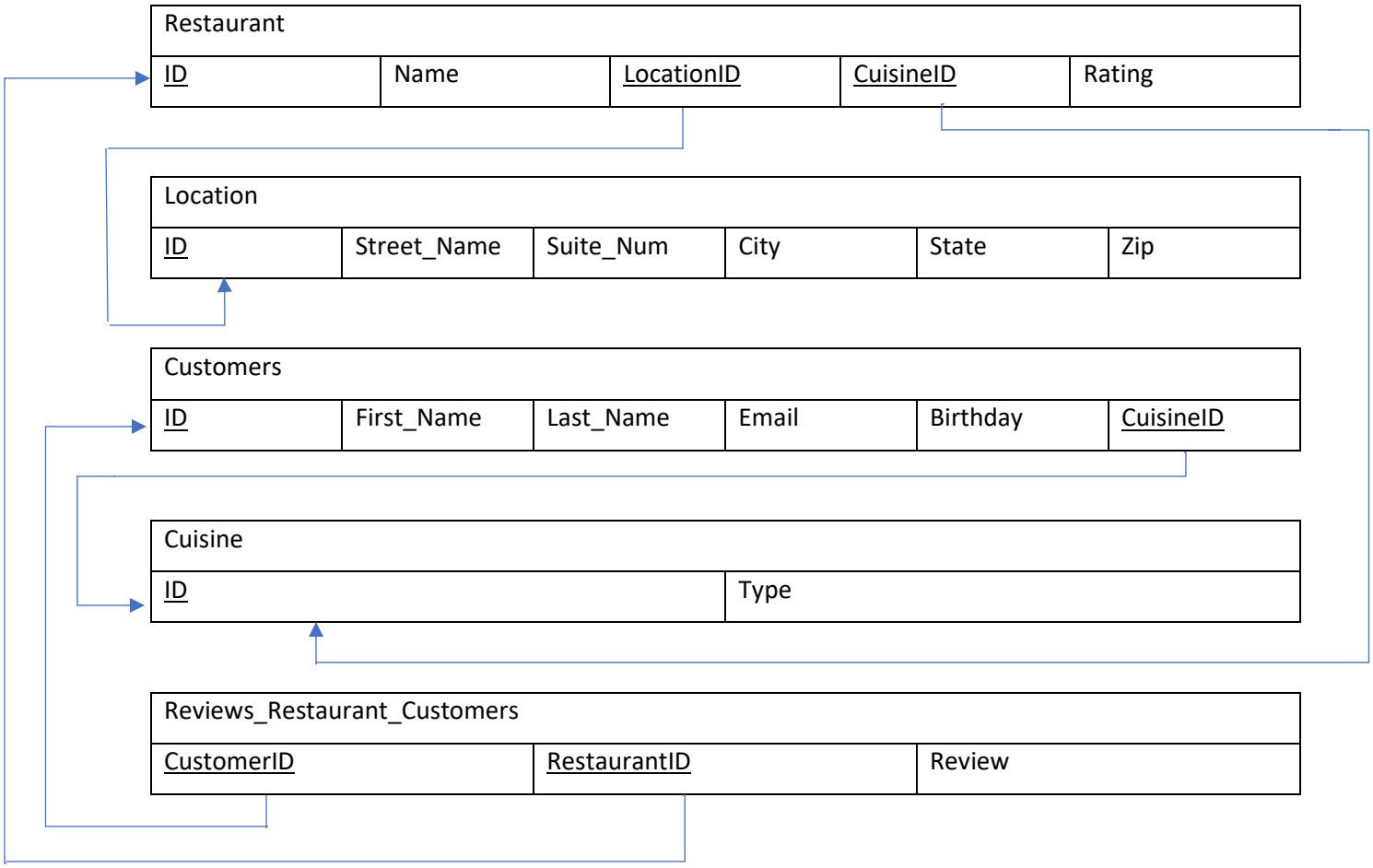
I did not change the default rating of 0 even though it may seem unfair. There are a few reasons I did not change this. One I want it to be more obvious to the customer that a restaurant has no reviews yet. When this database is more fully implemented, I hope to set it up so if a customer is viewing the rating they can easily access the number of reviews and how they are divided among likes and dislikes. A customer is more likely to question why a restaurant has a 0 than a 100 which makes them more likely to look at the number of reviews. The second reason is I do not expect the default rating to last very long. Customers are the ones who will be adding the restaurants to the database, so I would expect that if a customer is going through the process of adding the restaurant it is because they are wanting to leave their review. The idea is the customer would add the restaurant and then review it shortly after. Finally, the default value will be deleted by the average function as soon as the first review is added so that the initial 0 value is not skewing the results. To clarify the default value of 0 in the rating attribute when the table is created is just a placeholder until the first review is added and then rating becomes a calculated value based on the reviews relationship table.

I did add drop table statements as suggested. I originally did not because I was worried about a peer reviewer accidently deleting their tables when importing mine in case they were named the same. I did see the instructions for peer reviewing do warn us about backing up our own databases, so drop table statements should be okay.

Upgrades to the Draft version:

When checking the foreign keys in the schema after peer comments, I noticed I forgot to underline CusinieID under the customer entity to indicate it is a key. I fixed this mistake.


ID
Name
Rating
reviews
restaurants
n
serve
m
customers
First Name
Last Name
Email
ID
Birthdate
n
Street
Suite Number
City
ID
State
location
Zip Code
1
are at
1
prefer
n
specialize in
1
cuisine
1
ID
Type


Restaurant
ID
Name
LocationID
CuisineID
Rating
Location
ID
Street_Name
Suite_Num
City
State
Zip
Customers
ID
First_Name
Last_Name
Email
Birthday
CuisineID
Cuisine
ID
Type
Reviews_Restaurant_Customers
CustomerID
RestaurantID
Review