Introduction to Databases

Stage 4 Report:

Demo and Final Report

For Del Delivery Drivers Web Application

By: Morgan Houston

May 03rd, 2019

Introduction:

The application I have created compares the total price between different delivery companies. I chose this application to help people see the price difference between each company. I have experience with the delivery business, as I used to work for Door Dash. I always wondered if other companies were cheaper. I use delivery food services sometimes, so my motivation for this project is to see if I can save a few dollars.

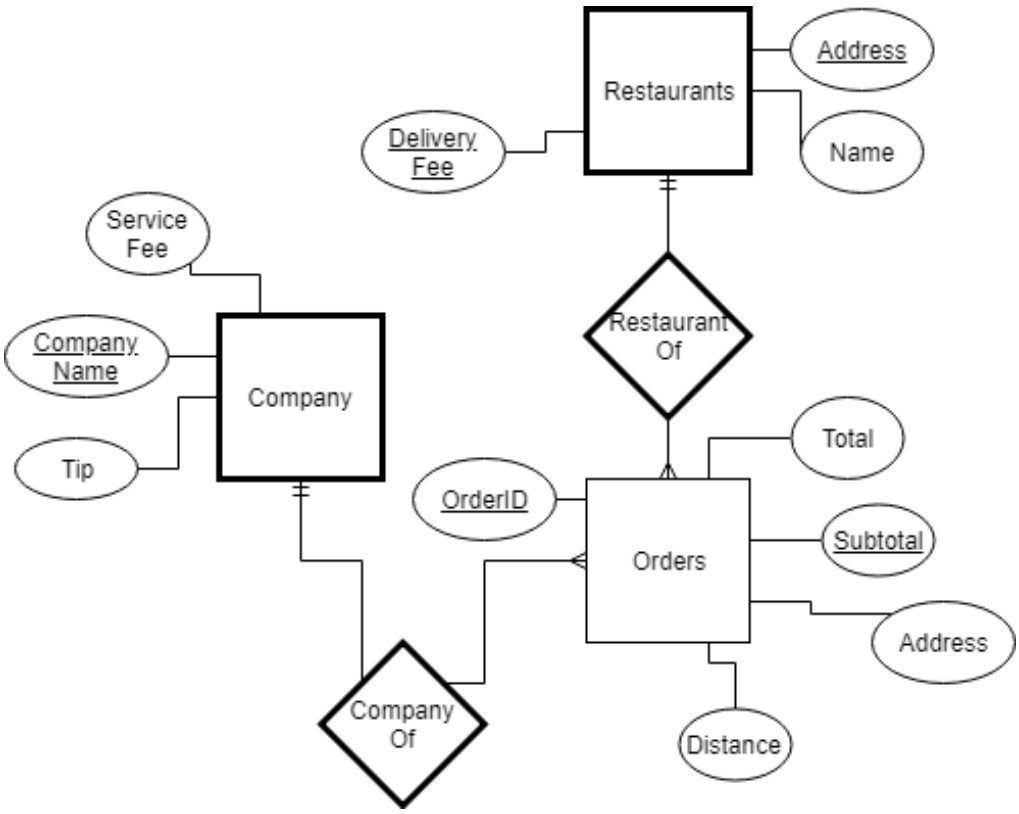
Del Delivery Drivers Database is a very useful application. There is no need to sign-in or create an account. Compare prices from different delivery services instantly. My project has a google map element to give a visual view of restaurants in the area. I have added an advanced function that when you select a restaurant, it will automatically insert the restaurant name and address into the submission form. Then all you must enter is your address and the orders subtotal for which you want to compare. This will help determine the delivery fee (if any) as well as the service fee and a %20 tip for Door Dash and Grub Hub orders. Uber Eats only lets you tip with cash. On my website you will also be able to delete and update queries.

Database Details:

I thought of the three main components needed to design the database. That is an order itself, the Restaurant whom is making the order, and the company who will be delivering the order. These are the entities needed to construct the database. Your address and the subtotal for your order along with the restaurant name and address will find the values in each of the table to get the total amount each company will charge you to deliver it.

ER Diagram:

The following is an entity relation model for my database. Orders is a strong entity, while restaurants and company are weak entities.



Relational Schema:

Restaurants	
	Name
PK1	Address
PK2	Delivery Fee
FK	OID

Orders	
PK1	OrderID
	Address
FK1	Subtotal
	Distance
PK2	Total

Company	
PK1	CompanyName
PK2	Service Fee
PK3	Tip
FK	OID

Functional Dependency:

If there is an order entity there must be a Restaurant and Company entity as well. The delivery fee, service fee and tip all rely on the subtotal. I believe my tables are in the Third-Normal Form.

- OrderID -> r.OID, c.OID
- OrderID -> o.Address, Subtotal, Distance, Total
- Subtotal -> Delivery Fee, Service Fee, Tip
- r.OID, o.Address, Subtotal -> r.Address, Name, Delivery Fee
- c.OID, o.Address, Subtotal -> Company Name , Service Fee, Tip
- OrderID, Subtotal, o.Address -> r.Address, r.Name, Distance, Total, Delivery Fee, Service Fee, Tip, Company Name

Functionality Details:

My database comes with a lot of basic features as well as some features that are advanced. For the basic ones, I made it where a user can insert data to the database. With each submission you will get three queries returned to display in a table on the website. You can then individually update and delete each one. One will compare the company Door Dash, one for Grub Hub and the last one for Uber Eats. I chose these companies because I feel they are most popular and do not require a subscription. The user can also sort the table by a list of different attributes.

For my advanced functionality, I have added a google maps API that helps the user select and find restaurants they want to compare the prices of. Another advanced function I implemented is determining the cheapest company for you. At the bottom of the website, there is a compare button. When you have multiple queries in the table and it is clicked, it will find the average total for each company and compare them with one another to find the cheapest one usually.

Implementation Details:

I chose to use XAMPP to create and host my webserver. It comes with Apache, which I believe does the hosting part, and MySQL for the database. I used HTML, CSS, PHP, and some JavaScript for the languages to implement the front-end Web Interface and the applications logic. The front-end web interface is stored in an 'index.php' file styled with a style.css file to create the text and tables on the webpage. JavaScript was used to make the map. PHP was used for the server side of things. When you click any of the buttons, it connects to the 'server.php' file, which is connected to myphpadmin on the server 127.0.0.1 in a database called 'database'. Here is where all the main logic is handled for the application database.

GitHub Repository: https://github.com/mmhousto/DeliveryDatabase

Experiences:

I have learned a lot over the course of this project. I started the project with three group members, now it is just me. This was a hard obstacle to overcome, I had to redesign the project in a way that a single person could complete. Before it was advanced and would have taken a lot more work, but I am very happy with the outcome and it feels good knowing I did it all by myself, with the help of the internet of course. Getting the google map API to work how I wanted to, was a hard problem for me and took awhile to implement. I have never coded in PHP, so it was nice learning the basics of another computer language, as well as working with MySQL. In the past, I have heard about MySQL but did not really know what it was. This project has really taught me a lot. To extend my project to an advanced system, I would need more people helping and a lot more time. We could do our groups original idea, to basically create our own delivery company, and have all the items from each restaurant in the database along with a description. There would also be a delivery bar on the webpage telling you how your order is going. I think this would be a fun project to work on in the future.

References:

I have used the following resources to help construct my webpage and database:

Awa Melvine: https://www.youtube.com/watch?v=mjVuBlwXASo

Johan Godinho: https://www.youtube.com/watch?v=YXCK03O-wjM &

https://www.youtube.com/watch?v=gsNF7trOtU8

Codex World: https://www.codexworld.com/distance-between-two-addresses-google-maps-api-php/

W3schools.com: https://www.w3schools.com/

Google Maps API: https://developers.google.com/places/web-service/intro