Cameron Cole
Professor DiPippo
CSC 436
Independent Project

Implementation of Food Delivery Database

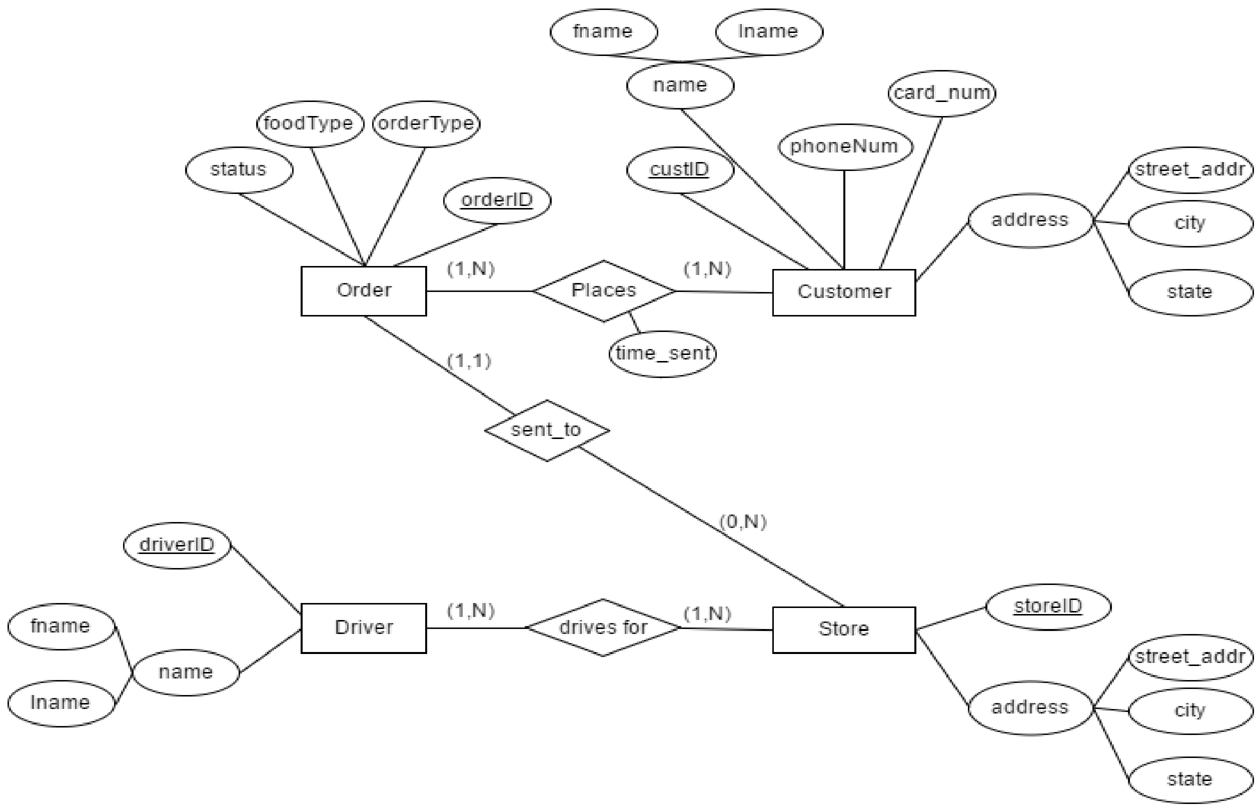
Application Description

An online food ordering system, such as that one may see for a pizza restaraunt chain, requires that a customer input information regarding their name, address, creditcard information, username, password, and also the food order information. This site may allow a customer to log in and save all this information for future transactions. Also, when a customer inputs this information, it may be important for the website to determine which store is closest to the customer and which delivery drivers are available, and the status of the transaction. By storing this information in a database, it will be easy to store and re-access information for a customer based on previous transactions. Queries to the database could return many kinds of useful information including the delivery driver who is closest to the customer, the driver with the least amount of queued deliveries to make, and the driver who has driven the most miles on a certain day. Additionally, this application provides a clear basis for a web application implementation.

Data Description

Starting with the customer, the application will keep track of the customer's ID (custID), username, password, first name, last name, street address, city, state, phone number, and credit card. Another relation called store will hold variables for the store's ID (storeID), name, street address, city, and state. In a relation called driver there will be an ID (driverID), first name, last name, and a foreign key to the store that the driver is associated with. To simplify the fact that a restaurant can have many types of foods, the orders relation will simply be an ID (orderID), food type, order type(pickup or delivery), the time the order is sent, the status of the order, and foreign keys that will point to the customer that placed the order and the store that the order has been sent to. In this case, the food and pricing related to food would probably require a whole different database. A delivery queue will have the driver's ID as the primary key and the order ID as the foreign key. Having a delivery queue solves the problem where a driver may have multiple deliveries to make.

Conceptual Database Design



Notable ER Design Features:

- The delivery queue is a feature that is not best represented in the ER design, but essentially occurs when the order is sent to the store. The store receives an order, chooses one of the drivers (probably the one with least amount of deliveries queued) and assigns them that delivery.
- Not all orders will be deliveries. If orderType is "pickup", then no delivery driver will be assigned that order.
- The min for orders being placed and sent is 1 because otherwise no order would exist.
- The relationship on the bottom assumes that a store for this company always provides delivery as a service and therefore must have at least 1 driver.

Relational Design

The following is the initial (1NF) relational form of the database:

CUSTOMER(CUSTID, USERNAME, PASSWORD, FNAME, LNAME, PNUM, CREDITCARD, STADDR, CITY, STATE)
ORDERS(ORDERID, STOREID, CUSTID, ORDERTYPE, FOODTYPE, TIMESENT, STATUS)
STORE(STOREID, NAME, PNUM, STADDR, CITY, STATE)
DRIVER(DRIVERID, STOREID, FNAME, LNAME)
DELIVERY_QUEUE(DRIVERID, ORDERID, STATUS)

Mapping from the ER diagram to the relational design was fairly straight forward. I was able to implement a relation for the delivery queue, which was not shown in the ER diagram. Also, I was able to make the following foreign keys apparent:

- CustID in Orders. A order is attached to the customer who placed it.
- StoreID in Driver. A driver is identified both by his ID and the store he drives for.

- OrderID in Delivery_Queue. A queued order is identified by its driver and order id.

Relational Analysis

As shown above, my initial relational design is already in 1NF.

Also if we observe this relational design, we can see its also in 2NF, because every relation is fully functionally dependent on its primary key(s). Since a driver may driver for multiple stores, both driverID and storeID are listed as primary keys.

CUSTOMER(<u>CUSTID</u>, USERNAME, PASSWORD, FNAME, LNAME, PNUM, CREDITCARD, STADDR, CITY, STATE)
ORDERS(<u>ORDERID</u>, STOREID, CUSTID, ORDERTYPE, FOODTYPE, TIMESENT, STATUS)
STORE(<u>STOREID</u>, NAME, PNUM, STADDR, CITY, STATE)
DRIVER(<u>DRIVERID</u>, <u>STOREID</u>, FNAME, LNAME)
DELIVERY_QUEUE(<u>DRIVERID, ORDERID</u>, STATUS)

Lastly, since there are no transitive properties in any of the relations, this is already in 3NF. The simplicity of the relations and my initial thoughts when creating them made it so my design was already in 3NF from the beginning.

Query Description

First, I created the tables for each relation in the database:

CREATE TABLE customer(custID varchar(7) NOT NULL, fname char(50), lname char(50), pnum varchar(15), creditcard varchar(30), staddr varchar(100), city varchar(50), state varchar(30), PRIMARY KEY(custID))

CREATE TABLE orders(orderID varchar(6) NOT NULL, storeID varchar(5) NOT NULL, custID varchar(7) NOT NULL, orderType varchar(8), foodType varchar(100), timeSent datetime, status varchar(50), PRIMARY KEY(orderID), FOREIGN KEY(storeID) REFERENCES store(storeID), FOREIGN KEY(custID) REFERENCES customer(custID))

CREATE TABLE store(storeID varchar(5), name varchar(200), pnum varchar(15), staddr varchar(100), city varchar(50), state varchar(30), PRIMARY KEY(storeID))

CREATE TABLE driver(driverID varchar(5), storeID varchar(5), fname char(50), lname char(50), PRIMARY KEY(driverID, storeID), FOREIGN KEY(storeID) REFERENCES store(storeID))

CREATE TABLE delivery_queue(driverID varchar(5), orderID varchar(6), status varchar(50), PRIMARY KEY(driverID, orderID), FOREIGN KEY(driverID) REFERENCES driver(driverID), FOREIGN KEY(orderID) REFERENCES orders(orderID))

The database was populated using phpMyAdmin insert functionality.

Code and Final Details

See attached zip file containing index.php and twig files that were used to implement the web application for the database. I was only able to implement the order page, login, and registration. I had a good idea of where I would have gone with the rest of it, but due to personal issues I was not able to finish implementing. Had I finished, I would have included a page to check statuses of orders, see store information, see driver information, and more features similar to the streamTV project.