**Database Project Report**

Taxicab DB System

COMP 7115

Database Systems

Fall 2018

Submitted December 1, 2018

**Introduction**

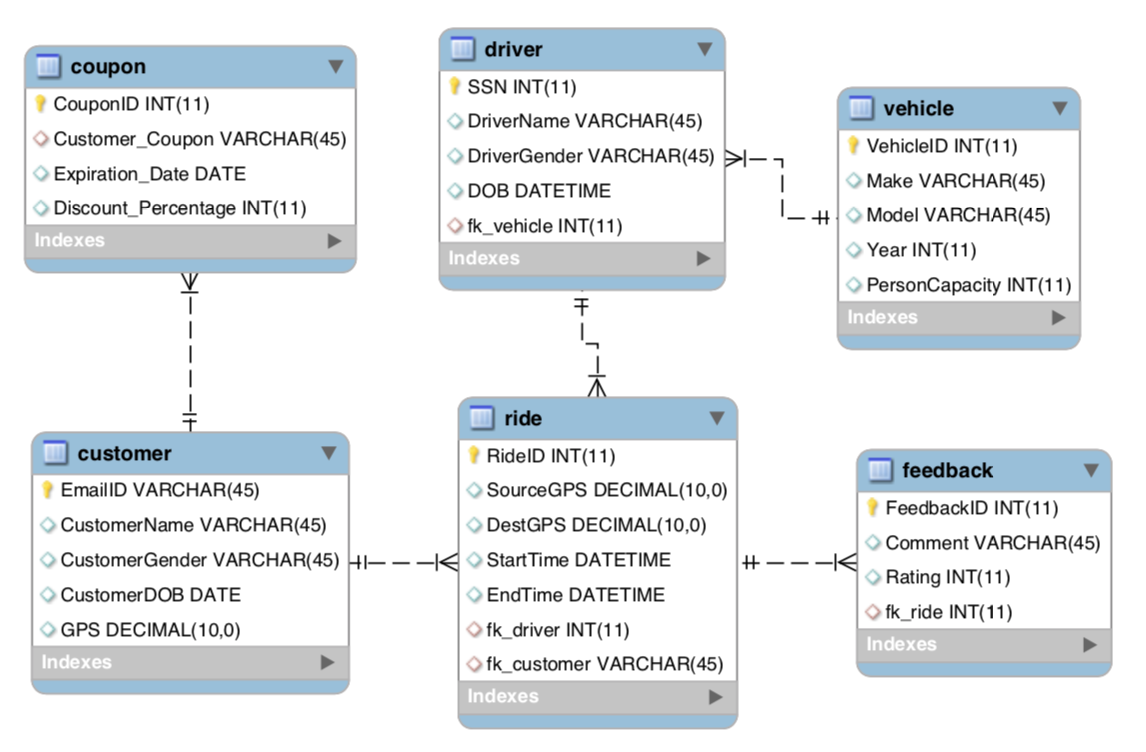
With an introduction to topics of relational database theory this semester, we were tasked to design and implement a practical application of relational databases, namely a simplified taxicab system. Given a list of basic requirements for the system, we brainstormed ways to best implement database models that would efficiently execute queries relevant to a real-world taxicab service such as Uber.

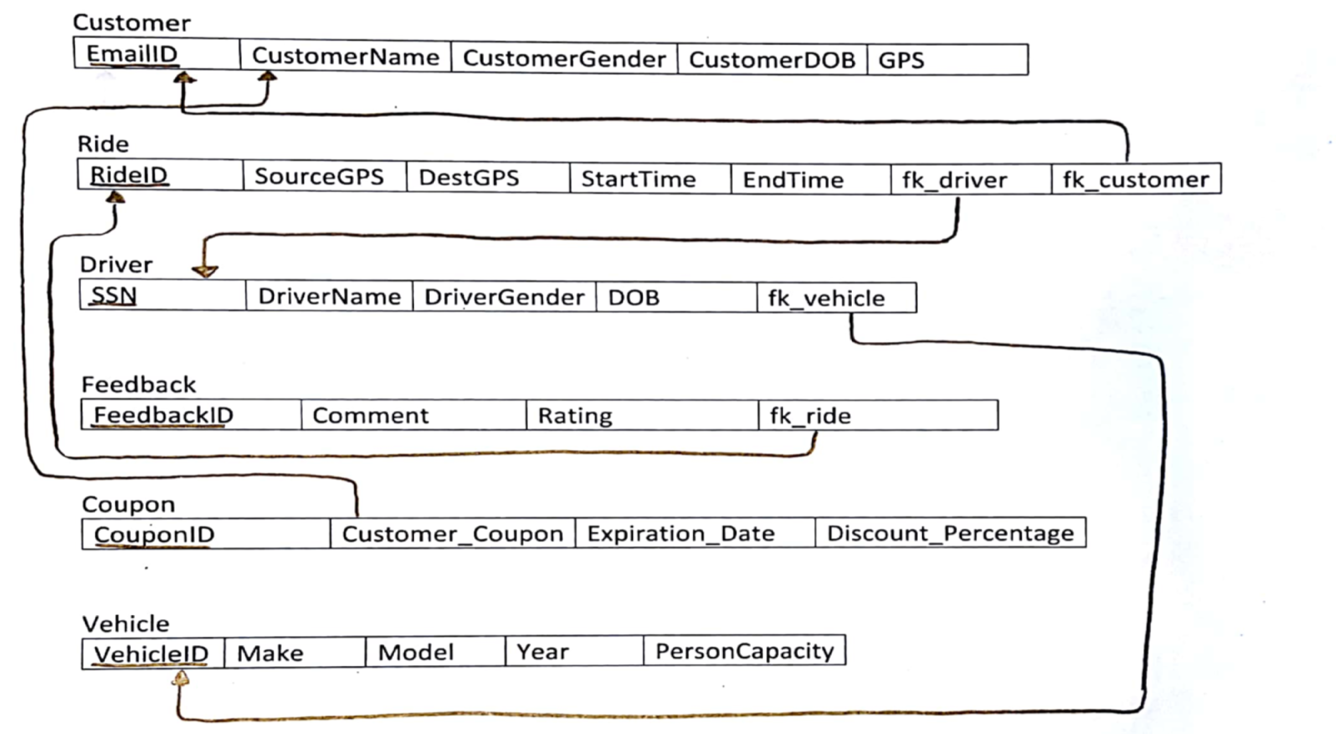
We decided that normalizing the relational models to 3-NF could accomplish these tasks. What this means is that our tables conform to 1-NF (that all attributes are atomic, that no composite or multivalued attributes are allowed), 2-NF (in a given tuple there is no partial dependence on a key), and 3-NF (there is no transitive dependency of non-key attributes on a key). In this way, our tables are well organized but not to such an extent that would create unnecessary lag when querying data.

Once our models were designed to our satisfaction, we could implement the corresponding database in MySQL, seed some data, and test the functionality by performing queries and analyzing the results. Once we were satisfied that our design works in the manner it was intended, we constructed an enhanced entity-relation diagram (documented in this report) and updated our relational model.

Finally, in connecting our database to a simple Python front-end system using the mysql.connector package, we could easily execute queries from the command line to simulate real-world database application.

**Database Design**

Our database relational models, normalized to 3-NF, can be observed in the following enhanced entity-relationship diagram built in MySQL Workbench and its corresponding relational model:



We made an assumption that certain calculations could be computed given data stored in the tables without occupying extra space in the database. For example, the cost of a ride can be computed from the GPS coordinates as well as the start and end times already provided to a “Ride” instance.

**Implementation**

The taxicab system database was created in MySQL Workbench, which, given a list of attributes, datatypes, and relationships between tables, automated much of the table creation process. Then, seeding sample data to the tables was performed by SQL scripting. The MySQL server was configured to host the database on our local machines such that services could be exploited by Workbench or other client programs.

As mentioned in the Introduction section of this document, a front-end system was designed in Python implementing the mysql.connector package, where queries could be easily executed and resulting tables could be displayed to the command line in an attractive manner.

**Queries**

List driver and vehicle information for all rides where the rating was better than average.

SELECT DriverName, DriverGender, DOB, Make, Model, Year

FROM driver INNER JOIN vehicle ON fk\_vehicle = VehicleID

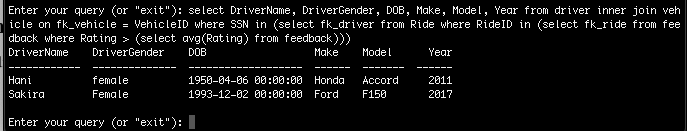
WHERE SSN IN

(SELECT fk\_driver FROM Ride

WHERE RideID IN

(SELECT fk\_ride FROM feedback

WHERE Rating > (SELECT avg(Rating) FROM feedback)))



List driver and feedback info for rides where Hani was the customer.

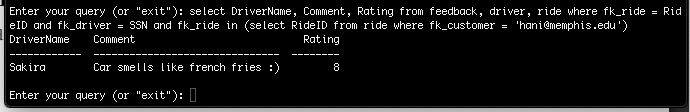
SELECT DriverName, Comment, Rating

FROM feedback, driver, ride

WHERE fk\_ride = RideID AND fk\_driver = SSN AND fk\_ride IN

(SELECT RideID FROM ride

WHERE fk\_customer = 'hani@memphis.edu')



Which driver gave the longest ride during the year 2018, and what was the source and destination GPS of that ride?

SELECT DriverName, SourceGPS, DestGPS

FROM driver, ride

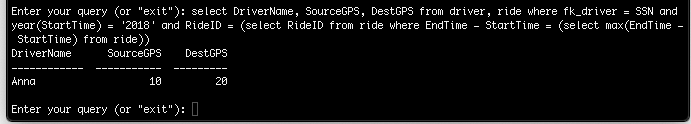
WHERE fk\_driver = SSN AND year(StartTime) = '2018' AND RideID =

(SELECT RideID FROM ride

WHERE EndTime - StartTime =

(SELECT max(EndTime - StartTime)

FROM ride))



How many rides did drivers give to destination GPS 20 and what was each driver’s average rating for those rides, listed in order of highest average rating?

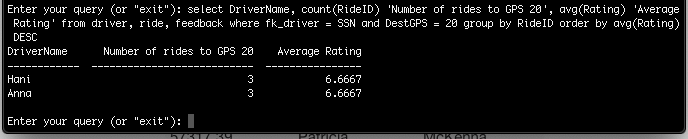
SELECT DriverName, count(RideID) 'Number of rides to GPS 20',

avg(Rating) 'Average Rating' FROM driver, ride, feedback

WHERE fk\_driver = SSN AND DestGPS = 20

GROUP BY RideID

ORDER BY avg(Rating) DESC



**Considerations**

Instead of using proper GPS coordinate system for location, for this database we decided that a decimal number would suffice in its place. In future works it would be cool to interact with Google Maps API in order to get real world tracking data. Additionally, it would be an interesting project to link our database to a model-view-controller framework package like Django or Flask to implement a more functional and attractive web-based taxi cab web site.