MOHAMMAD ATIF HUSSAIN atifhussain@utdallas.edu

**TAXI SERVICE**

**CS/SE 6360.003- FINAL PROJECT REPORT**

**TAXI SERVICE**

**CS/SE 6360.003- FINAL PROJECT REPORT**

**Table of Contents**

Requirements2

Modelling of Requirements as E R Diagram3

Mapping of E R Diagram in Relational Schema5

Applying Normalization6

Final Relational Schema7

SQL Statements to Create Relations8

PL/SQL Triggers12

Trigger-I12

Trigger-II12

PL/SQL Procedures13

Procedure-I13

Procedure-II14

**REQUIREMENTS**

A **Taxi Service** includes a **taxicab**, also known as a **taxi** or a **cab**, which is a type of [vehicle for hire](https://en.wikipedia.org/wiki/Vehicle_for_hire) with a driver, used by a single passenger or small group of passengers, often for a non-shared ride. A taxi service conveys passengers between locations of their choice. This differs from other modes of [public transport](https://en.wikipedia.org/wiki/Public_transport) where the pick-up and drop-off locations are determined by the service provider, not by the passenger, although [demand responsive transport](https://en.wikipedia.org/wiki/Demand_responsive_transport) and [share taxis](https://en.wikipedia.org/wiki/Share_taxi) provide a hybrid bus/taxi mode.

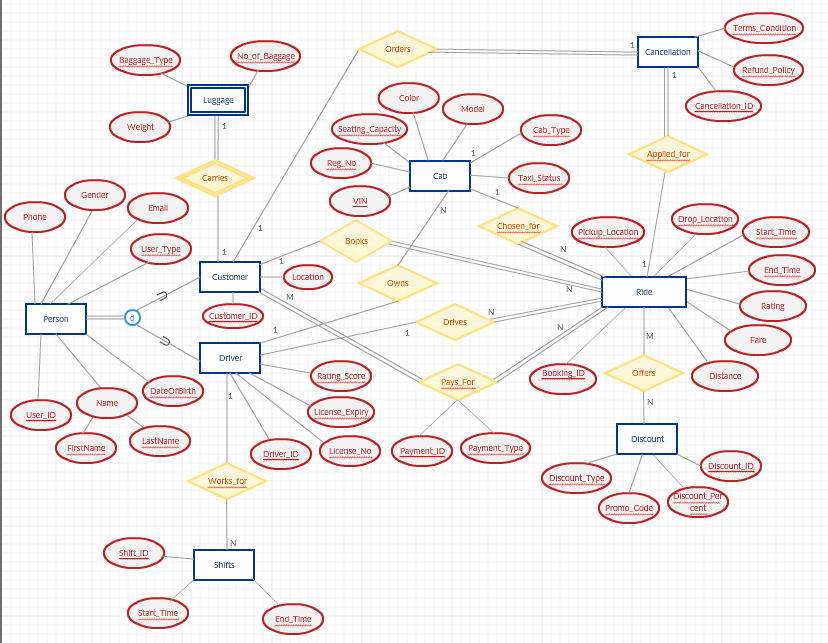
The basic requirement for the existence of the taxi service is a proper link between a customer and a driver. It may also be possible that a driver can be a customer for the next available driver in some situations. With the rapid growth of taxi service providing firms such as Uber, Lyft, etc. we have come up with gathering most of the necessary functions used to run this business. Firstly, the most important entities involved in such scenario are the 1. users, 2. cab and the 3. ride.

1. A user can have information like his ID, name, Birth date, address, phone number and email. Also, the user can either be a driver or a customer depending on different situations. In case he is a driver, he has to have a valid driving license that is unexpired along with the vehicle ID that he decides to use during his ride. His driving ethics and other information like his driving shift timings and ratings can also be included for a better customer service. In case the user is a customer, then he has to have a proper location to be picked up from.

2. Meanwhile, the driver can own multiple cabs with different vehicle ID, Registration number, color, model, seating capacity, type of cab service and the status of the cab.

3. A customer can book a ride from the cab and this ride can include multiple features such as the booking ID, Pickup Location, Drop Location, Start and end time of the ride, Rating, fare and the distance covered during the ride. Only one cab and a driver can be chosen for a ride. Once the ride is completed, the customer has to make payment to the driver of cab depending on the mode of payment of his choice. The ride charges between to same places may vary according to the discount and surge prices during that time. These may be offered by applying some promotional codes or discount percentage to a loyal existing customer.

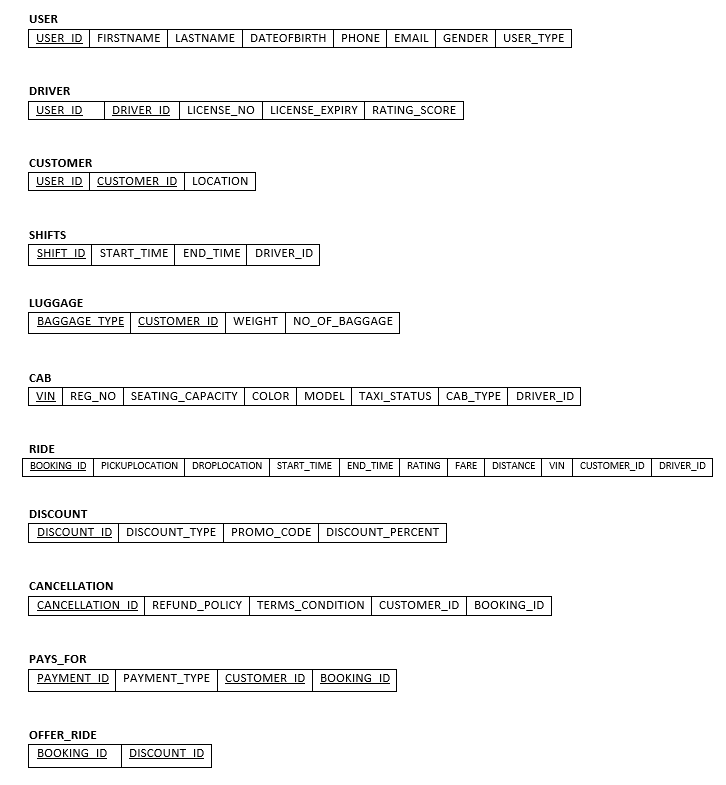
**MODELLING OF REQUIREMENTS AS ER-DIAGRAMS**



The requirements derived from ER Diagram are -

* A Customer can book 1 ride at a time and a ride can be booked by only one customer (1:1).
* A Driver can work for many shifts and only 1 shift can be taken by a driver at a time (1:N).
* Only 1 cab can be chosen for a ride and one ride can have only 1 cab at a time. (1:1).
* A Driver can own N cabs and 1 cab cannot be owned by more than 1 driver at once (1:N).
* A driver can drive only 1 ride at a time and only 1 driver can be used for a specific ride (1:1).
* A customer can pay for any rides and a ride can be paid by any customers (M:N).
* A ride can be chosen for any amount of discount and a discount can be offered to any ride (M:N).
* One ride can be applied for cancellation and only one ride can be cancelled at a time(1:1).

**MAPPING OF ER-DIAGRAM IN RELATIONAL SCHEMA**



**APPLYING NORMALIZATION**

**i) Partial dependency exists on CUSTOMER & DRIVER, Applying 2NF**

**CUSTOMER: {CUSTOMER\_ID 🡪 LOCATION}**

**DRIVER: {DRIVER\_ID 🡪 LICENSE\_NO, LICENSE\_EXPIRY, RATING\_SCORE}**

**CUSTOMER**

|  |  |
| --- | --- |
| CUSTOMER\_ID | LOCATION |

**DRIVER**

|  |  |  |  |
| --- | --- | --- | --- |
| DRIVER\_ID | LICENSE\_NO | LICENSE\_EXPIRY | RATING\_SCORE |

**USER\_DRIVER**

|  |  |
| --- | --- |
| USER\_ID | DRIVER\_ID |

**USER\_CUSTOMER**

|  |  |
| --- | --- |
| USER\_ID | CUSTOMER\_ID |

**ii) Applying 2NF to PAYS\_FOR,**

**PAYS\_FOR: {PAYMENT\_ID 🡪 PAYMENT\_TYPE}**

**PAYMENT**

|  |  |
| --- | --- |
| PAYMENT\_ID | PAYMENT\_TYPE |

**PAYMENT\_BOOKING**

|  |  |  |
| --- | --- | --- |
| CUSTOMER\_ID | PAYMENT\_ID | BOOKING\_ID |

**iii) Applying 3NF to DRIVER,**

**DRIVER: {LICENSE\_NO 🡪 LICENSE\_EXPIRY}**

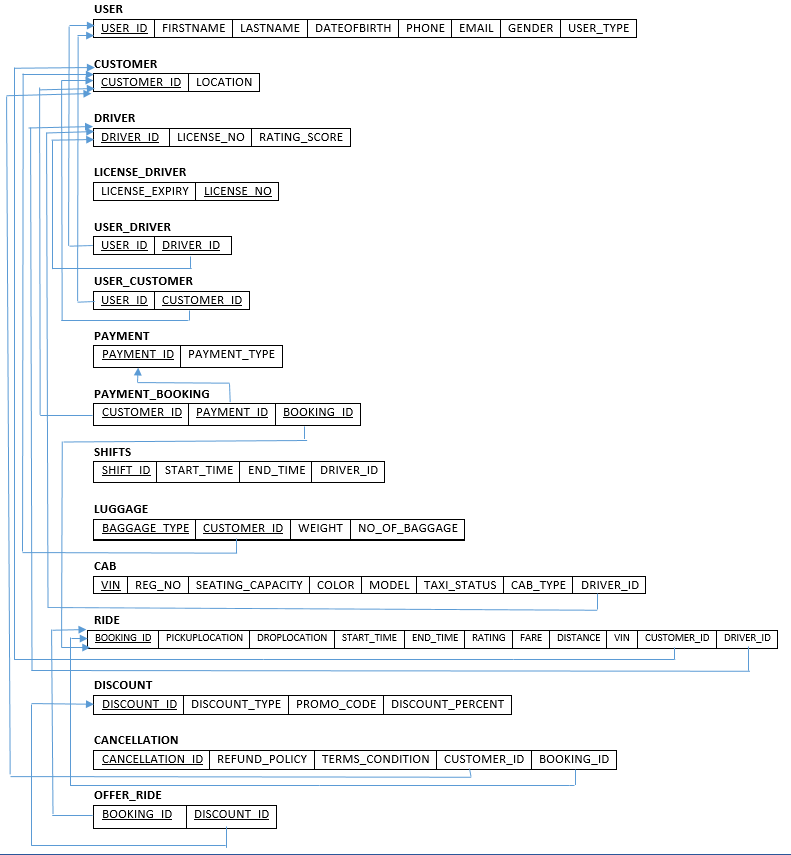
**DRIVER**

|  |  |  |
| --- | --- | --- |
| DRIVER\_ID | LICENSE\_NO | RATING\_SCORE |

**LICENSE\_DRIVER**

|  |  |
| --- | --- |
| LICENSE\_EXPIRY | LICENSE\_NO |

**FINAL RELATIONAL SCHEMA**



**SQL STATEMENTS TO CREATE RELATIONS**

1. CREATE TABLE PERSON

(

USER\_ID int NOT NULL PRIMARY KEY,FIRSTNAME varchar(255),

LASTNAME varchar(255), DATEOFBIRTH varchar(255), PHONE INT,EMAIL varchar(255),

GENDER varchar(255),USER\_TYPE varchar(255)

)

1. CREATE TABLE CUSTOMER

(

CUSTOMER\_ID int NOT NULL PRIMARY KEY,

LocationValue varchar(255)

)

1. CREATE TABLE DRIVER

(

DRIVER\_ID int NOT NULL PRIMARY KEY,

LICENSE\_NO INT NOT NULL,

RATING\_SCORE varchar(255)

)

1. CREATE TABLE LICENSE\_DRIVER

(

LICENSE\_NO INT NOT NULL PRIMARY KEY REFERENCES DRIVER(LICENSE\_NO),

LICENSE\_EXPIRY varchar(255)

)

1. CREATE TABLE USER\_DRIVER

(

USER\_ID int NOT NULL,

DRIVER\_ID int NOT NULL PRIMARY KEY REFERENCES driver(DRIVER\_Id),

FOREIGN KEY (USER\_ID) REFERENCES Person(user\_Id)

)

1. CREATE TABLE USER\_CUSTOMER

(

USER\_ID int NOT NULL,

CUSTOMER\_ID int NOT NULL PRIMARY KEY REFERENCES customer(customer\_Id),

FOREIGN KEY (USER\_ID) REFERENCES Person(user\_Id)

)

1. CREATE TABLE PAYMENT

(

PAYMENT\_ID int NOT NULL PRIMARY KEY,

PAYMENT\_TYPE varchar(255)

)

1. CREATE TABLE PAYMENT\_BOOKING

(

PAYMENT\_ID int NOT NULL PRIMARY KEY REFERENCES PAYMENT(PAYMENT\_ID),

CUSTOMER\_ID int NOT NULL,

BOOKING\_ID int NOT NULL,

FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMER(CUSTOMER\_ID),

FOREIGN KEY (BOOKING\_ID) REFERENCES RIDE(BOOKING\_ID)

)

1. CREATE TABLE SHIFTS

(

SHIFTS\_ID int NOT NULL PRIMARY KEY,

START\_TIME VARCHAR(255),

END\_TIME VARCHAR(255),

DRIVER\_ID int NOT NULL,

FOREIGN KEY (DRIVER\_ID) REFERENCES DRIVER(DRIVER\_Id)

)

1. CREATE TABLE LUGGAGE

(

CUSTOMER\_ID int NOT NULL PRIMARY KEY,

BAGGAGE\_TYPE int NOT NULL,

WEIGHT VARCHAR(255),

NUMBER\_OF\_BAGGAGE VARCHAR(255),

FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMER(CUSTOMER\_ID)

)

1. CREATE TABLE CAB

(

VIN INT NOT NULL PRIMARY KEY,

REG\_NO VARCHAR(255),

SEATING\_CAPACITY VARCHAR(255), COLOR VARCHAR(255), MODEL VARCHAR(255), TAXI\_STATUS VARCHAR(255),

CAB\_TYPE VARCHAR(255), DRIVER\_ID INT NOT NULL,

FOREIGN KEY (DRIVER\_ID) REFERENCES DRIVER(DRIVER\_ID)

)

1. CREATE TABLE RIDE

(

BOOKING\_ID INT NOT NULL PRIMARY KEY,

PICKUPLOCATION VARCHAR(255),

DROPLOCATION VARCHAR(255), START\_TIME VARCHAR(255), END\_TIME VARCHAR(255), RATING VARCHAR(255),

FARE VARCHAR(255),

DISTANCE NUMBER(5,2) DEFAULT 0.00 CHECK (DISTANCE>=0.00 AND DISTANCE<=100),

VIN INT NOT NULL, CUSTOMER\_ID INT NOT NULL,DRIVER\_ID INT NOT NULL,

FOREIGN KEY (VIN) REFERENCES CAB(VIN),FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMER(CUSTOMER\_ID),

FOREIGN KEY (DRIVER\_ID) REFERENCES DRIVER(DRIVER\_ID)

)

1. CREATE TABLE DISCOUNT

(

DISCOUNT\_ID INT NOT NULL PRIMARY KEY,

DISCOUNT\_TYPE VARCHAR(255),

PROMO\_CODE VARCHAR(255), DISCOUNT\_PERCENT VARCHAR(255)

)

1. CREATE TABLE OFFER\_RIDE

(

BOOKING\_ID INT NOT NULL PRIMARY KEY REFERENCES RIDE(BOOKING\_ID),

DISCOUNT\_ID INT NOT NULL,

FOREIGN KEY (DISCOUNT\_ID) REFERENCES DISCOUNT(DISCOUNT\_ID)

)

1. CREATE TABLE CANCELLATION

(

CANCELLATION\_ID INT NOT NULL PRIMARY KEY,

REFUND\_POLICY VARCHAR(255), TERMS\_CONDITIONS VARCHAR(255),

CUSTOMER\_ID INT, BOOKING\_ID INT,

FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMER(CUSTOMER\_ID),

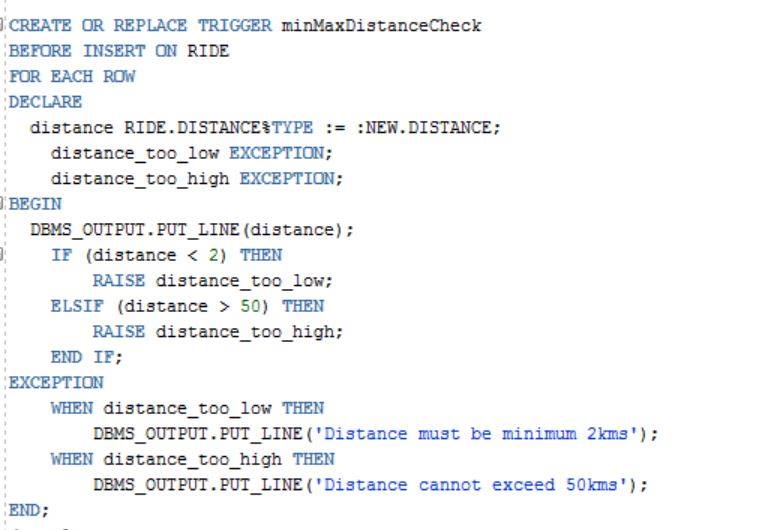
FOREIGN KEY (BOOKING\_ID) REFERENCES RIDE(BOOKING\_ID)

)

**PL/SQL-TRIGGERS**

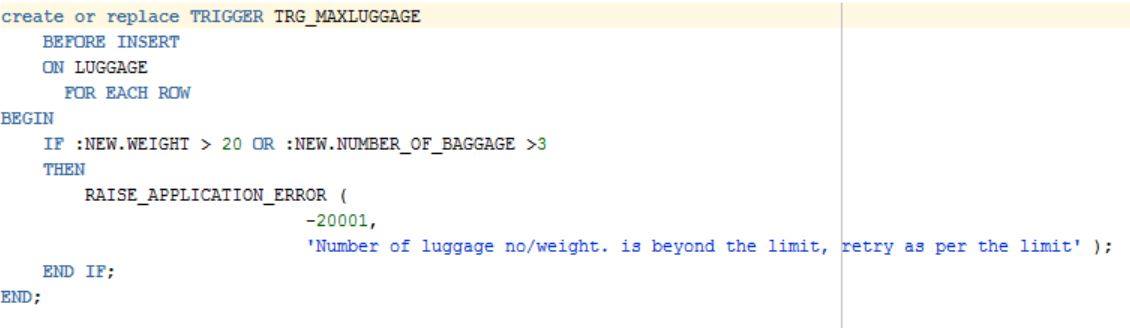
**Triggers -I**

Trigger is executed when an entry with distance less than 2kms or more than 50kms is being made to insert in Ride table



**Triggers -II**

Trigger prompts application error when luggage total weight exceeds 20kgs or no. of baggage is more than 3

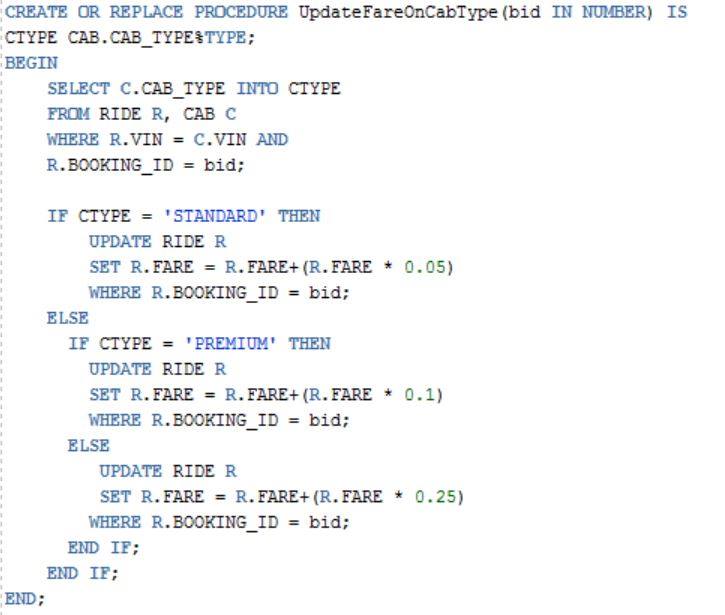


**PL/SQL-PROCEDURES**

**Procedures-I**

This procedure hikes the fare price in accordance with the type of car chosen for the ride.

* Argument: (Booking\_ID)



**Procedures-II**

This procedure updates the ride fare by 10% for customers with luggage weight more than 20kgs & distance greater than 5kms.

* Argument: None

