Database Design
CS 6360.002: Transportation System

Due on Wednesday April 29, 2020 at 11:59pm

Instructor: Nurcan Yuruk

Team 23
Hansika Venkatraman (hxv180008) / Humayoon Akhtar Qaimkhani (hxq190001) / Varadhan Ramamoorthy / (vrr180003)

Contents

1 Requirement

Transportation systems are requisite for all the cities around the world. Currently, there are various transportation systems and they are controlled and managed by various control systems. Within the context of routing schedules and building structures , these are managed separately again. It is reasonable that they should be integrated into a single system. To achieve such integration of transport management , it is necessary to merge their management into one transportation information system and accelerate the collaboration among different types of transport organizations.

1. The system keeps track of each city and its transportation system. Each city will have a unique id (city id) for its identification.

2. Each city shall have multiple Transport Offices located at different stations. Transport Office include Office id, name, Transport Officer (person in charge of the office), address and phone. Each Transport Office must have a unique Office id for its identification.

3. The system shall keep information about structural buildings such as tolls, highways and bridges that belong to a particular city. It also keeps track of all the routes that are part of the city. Each of these buildings or route could be part of multiple cities as well, for example a highway/bridge or a route could be connecting two or multiple cities.

4. Each transportation office shall have multiple transportation vehicle operating under it. Each vehicle include vehicle id, registration (state and reg number) and are categorized based on the transportation medium type (air, water and road).

5. These transportation vehicle shall be owned by a company/owner. A company can own more than one transportation vehicle. Company include company name, and its address.

6. Each transportation vehicle shall be assigned a schedule that includes vehicle frequency, start time, end time and the person who controls the vehicle (driver).

7. Each transportation vehicle shall have a route that it travels. Route includes route id, source and destination locations.

8. The system shall store information about the stops that are part of a route. A transport stop include stop name which will be unique as these stops will be named using primary street followed by the nearest cross street or landmark.

9. The system allows passengers/customer to book ticket for different vehicles. A passenger needs to provide SSN, name ,phone and address to register.

10. The system shall keep information of the tickets a passenger books. A ticket include ticket id, source and the destination location, quantity and price for each ticket.

Assumptions

1. To effectively manage the vehicle resources, we assume that there is only one vehicle functioning in a route at all times. Similarly, a vehicle can only be assigned to single route.

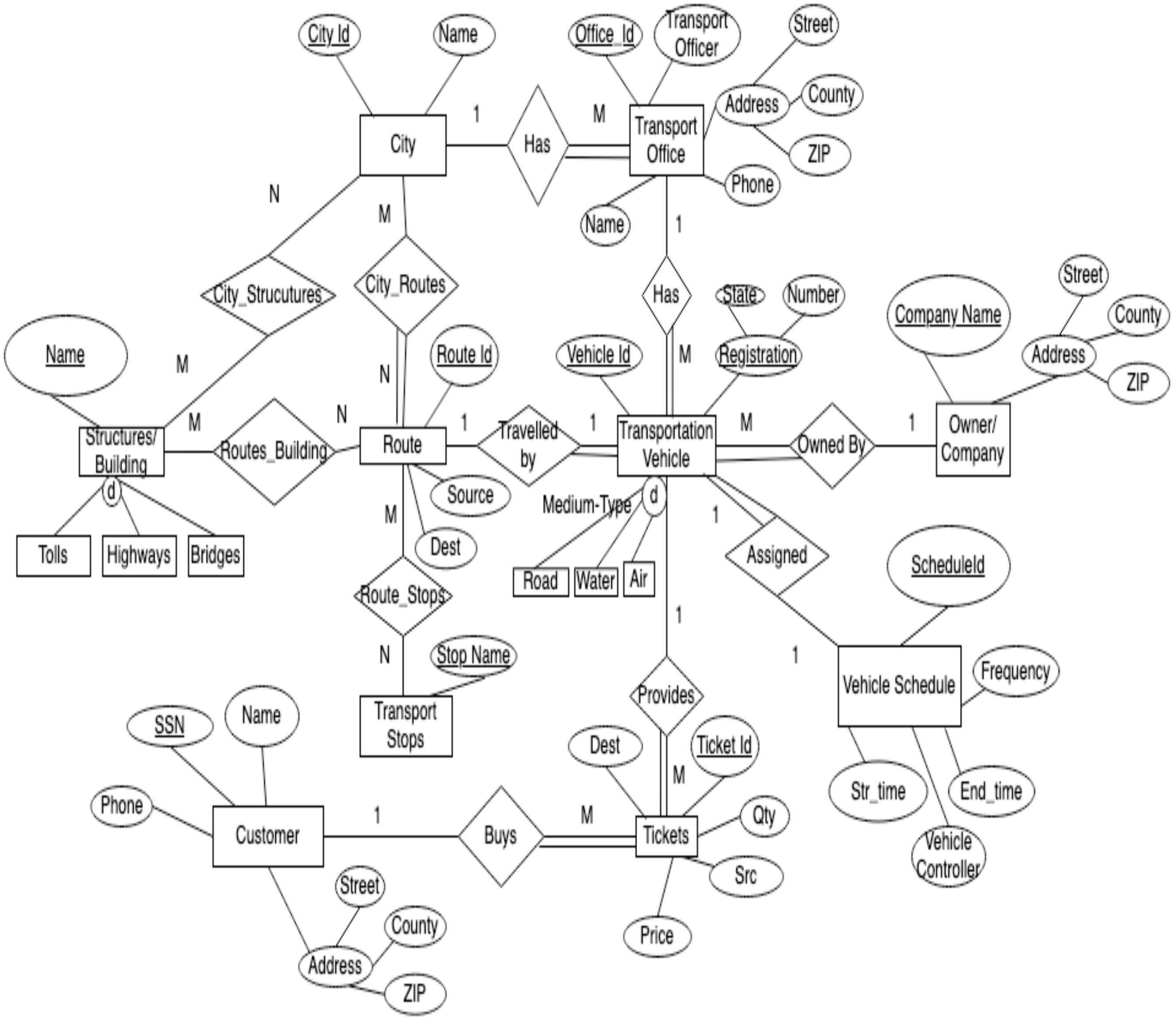
2. To simplify the complexity of the system, we assume that the tickets booked by the passengers cannot be cancelled at all.

3. To productively manage the schedules of vehicles, we assume that a vehicle can be assigned to only one schedule.

2 Enhanced Entity-Relation Diagram

The EER is shown in fig. 1.

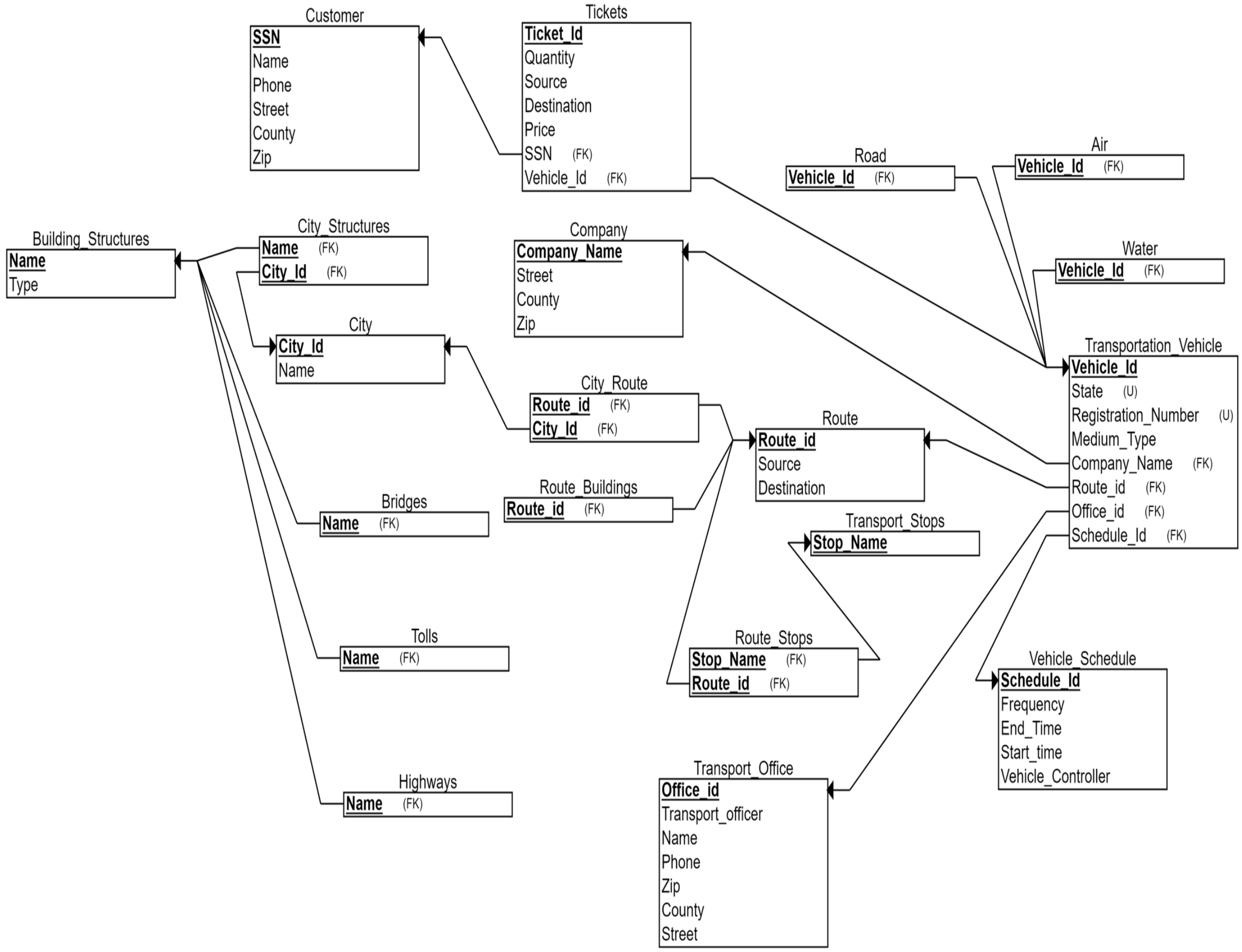
Figure 1: EER for Transportation System



3 Mapping EER Diagram to Relational Schema and Normalization

The relational schema mapped from EER is shown in fig. 2.

Figure 2: Relational Schema after Mapping



The functional dependencies in the schema are the following:

1 . Customer:

In customer Relation ,Since SSN is the primary key ,it can uniquely identify any attribute in the relation. The dependency here is :

SSN -> Name, Phone, Street, County, Zip, City_Id.

.Here, the prime attribute is SSN and Super/Candidate key is also SSN. Since, SSN determines the non-prime attributes , the Relation is already in Second Normal Form since it satisfies first Normal form(no multi-valued attribute) and No Partial Key.

We can also see that the Street and County can determine Zip code

Street, County -> Zip

For a Relation to be in 3rd normal form , the relation should also be in 2nd normal form and it must satisfy either of this condition:

1. The attribute that determines the other attribute should be a super key or
2. The attributes that are determined must be a prime attribute.

Since the above dependency must not satisfy this condition we are making a new relation called as Address with Street and county as Primary key which can determine the Zip code.

Therefore, The Normalized relationship will be :

Customer(SSN, Name, Phone, Street, County)

Address(Street, county, Zip, City_Id)

2. Tickets:

In this relation, the Ticket_Id attribute can determine all the other attributes. Therefore Ticket_Id is the primary key and the table is already in 2nd Normal form since it satisfies the 1st NF and there is no partial Dependency present.

Ticket_Id is the Primary key/super key/candidate key.

The other candidate keys present are

Ticket_Id , Source, Destination(Also satisfies 2nd NF).

There is the functional Dependency present in Tickets relation as the Source and Destination can determine the price of the ticket.

Source, Destination -> Price.

To Convert it to 3rd NF , it must satisfy the condition(as said above).

Since Source and destination aren't a super key and it also doesn't determines prime attributes, We are making a new relationship by naming the relation as Travel_Amount.

Tickets(<u>Ticket_id</u>,Quantity,Source,Destination,SSN,Vehicle_Id,Registration_Number)

Travel_Amount(<u>Source, Destination</u>, Price)

3. Company:

In this relation, same as the Customer relation County and zip can determine the zip code

County, Street → Zip

The Company relation is already in 2nd NF (Company_Name is the primary/candidate key and can determine all the other attribute).

Converting it to 3rd NF:

Company(<u>Company_name</u>, Street, County)

Address(<u>County, Street</u>, Zip, City_Id)

4. Transportation Vehicle:

Here ,The dependencies present are as follows:

Vehicle_id->Medium_Type

Reg_Number ->State,Company_Name, Office_id,Route_Id,Schedule_Id

Here the table is in the 1st NF (No multivalued attributes).

Converting it into 2nd NF:

There is no partial key present as Vehicle_Id can alone determines the other attributes and Reg_Number can also determines the other attributes.

Therefore, Normalizing it to 2nd NF:

Transportation_Vehicle(Vehicle_Id,Registration_Number,State,company_Name,office_id,Route_Id,Schedule_id)

Converting it to 3rdNF:

Here, the table satisfies the 2nd NF and also satisfies the 3rd NF. (SuperKeys only determines the attribute in all the tables)

Transportation_Vehicle(Vehicle_Id,Registration_Numer,Medium_Type)

Registration_Details(Registration_Number,State,company_Name,office_id,Route_Id,Schedule_id)

5. Transport_Office:

The functional Dependencies are :

The office_Id is the primary key. Therefore the table satisfies the 2nf (no partial key present and also in 1st NF).

County,Street -> Zip

Converting to 3rd NF:
We can also see that the Street and County can determine Zip code

Street, County -> Zip

For a Relation to be in 3rd normal form , the relation should also be in 2nd normal form and it must satisfy either of this condition:

1. The attribute that determines the other attribute should be a super key or
2. The attributes that are determined must be a prime attribute.

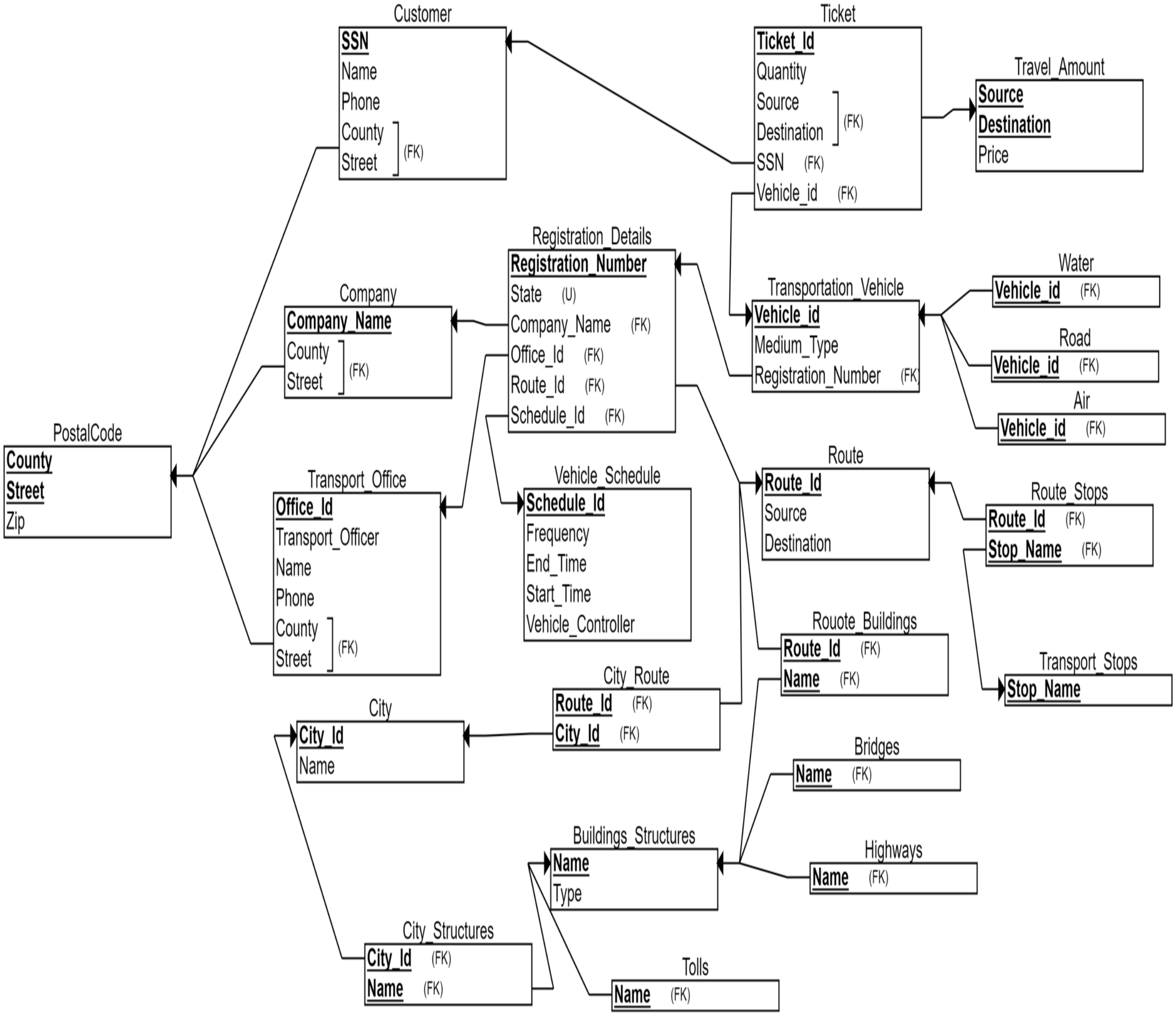
Since the above dependency must not satisfy this condition we are making a new relation called as Address with Street and county as Primary key which can determine the Zip code.

Transport_Office(office_id, Name, phone, Transport_Officer, County, Street)

Address(County, Street, Zip)

The relational schema after Normalization is shown in fig 3.

Fig 3 : Relational schema after Normalization



4 SQL

The CREATE table command is as follow.

```
-- Table creation
CREATE TABLE Customer(
 SSN      CHAR(9)        PRIMARY KEY,
 Name     VARCHAR(100)   NOT NULL,
 Phone    VARCHAR(20)    NOT NULL    UNIQUE,
 County   VARCHAR(50)    NOT NULL,
 Street   VARCHAR(50)    NOT NULL
);

CREATE TABLE Company(
 Company_Name    VARCHAR(100)    PRIMARY KEY,
 County          VARCHAR(50)     NOT NULL,
 Street          VARCHAR(50)     NOT NULL
);


CREATE TABLE PostalCode(
 County      VARCHAR(50),
 Street      VARCHAR(50),
 Zip         INTEGER         NOT NULL,
 City_Id     INTEGER         NOT NULL,
 CONSTRAINT County_Street_PK PRIMARY KEY ( County, Street )
);


CREATE TABLE Transport_Office(
 Office_Id          INTEGER         PRIMARY KEY,
 Transport_Officer   VARCHAR(100)    NOT NULL,
 Name                VARCHAR(50)     NOT NULL,
 Phone               VARCHAR(20)     NOT NULL,
 County              VARCHAR(50)     NOT NULL,
 Street              VARCHAR(50)     NOT NULL
);


CREATE TABLE City(
 City_Id   INTEGER     PRIMARY KEY,
 Name      VARCHAR(50) NOT NULL
);


CREATE TABLE City_Structures(
 City_Id   INTEGER     PRIMARY KEY,
 Name      VARCHAR(50) NOT NULL
);


CREATE TABLE Buildings_Structures(
 Name     VARCHAR(50)    PRIMARY KEY,
 Type     VARCHAR(50)    NOT NULL
);
```

```
CREATE TABLE Tolls(
  Name   VARCHAR(50)  PRIMARY KEY
);


CREATE TABLE City_Route(
  Route_Id    INTEGER   NOT NULL,
  City_Id     INTEGER   NOT NULL,
  CONSTRAINT Route_City_PK PRIMARY KEY ( Route_Id, City_Id )
);


CREATE TABLE Vehicle_Schedule(
  Schedule_Id          INTEGER       PRIMARY KEY,
  Frequency            VARCHAR(20)   NOT NULL,
  End_Time             TIMESTAMP(0)       NOT NULL,
  Start_Time           TIMESTAMP(0)       NOT NULL,
  Vehicle_Controller    VARCHAR(50)   NOT NULL
);


CREATE TABLE Registration_Details(
  Registration_Number   INTEGER       PRIMARY KEY,
  State                 VARCHAR(20)   NOT NULL,
  Company_Name          VARCHAR(100)  NOT NULL,
  Office_Id             INTEGER       NOT NULL,
  Route_Id              INTEGER       NOT NULL,
  Schedule_Id           INTEGER       NOT NULL
);


CREATE TABLE Ticket(
  Ticket_Id            INTEGER       PRIMARY KEY,
  Quantity             INTEGER       NOT NULL,
  Source               VARCHAR(30)   NOT NULL,
  Destination          VARCHAR(30)   NOT NULL,
  SSN                  CHAR(9)       NOT NULL,
  Vehicle_Id           INTEGER       NOT NULL,
  Registration_Number   INTEGER       NOT NULL
);


CREATE TABLE Transportation_Vehicle(
  Vehicle_Id           INTEGER       NOT NULL,
  Registration_Number   INTEGER       NOT NULL,
  Medium_Type          VARCHAR(20)   NOT NULL,
  CONSTRAINT Vehicle_Reg_PK PRIMARY KEY ( Vehicle_Id)
);

CREATE TABLE Route(
  Route_Id       INTEGER       PRIMARY KEY,
  Source         VARCHAR(30)   NOT NULL,
  Destination     VARCHAR(30)   NOT NULL
);
```

```
CREATE TABLE Route_Buildings(
 Route_Id     INTEGER      NOT NULL,
 Name         VARCHAR(50)  NOT NULL,
 CONSTRAINT Route_Name_PK PRIMARY KEY ( Route_Id, Name )
);


CREATE TABLE Bridges(
 Name    VARCHAR(50)   PRIMARY KEY
);


CREATE TABLE Highways(
 Name    VARCHAR(50)   PRIMARY KEY
);


CREATE TABLE Travel_Amount(
 Source       VARCHAR(30)   NOT NULL,
 Destination  VARCHAR(30)   NOT NULL,
 Price        INTEGER       NOT NULL,
 CONSTRAINT Source_Destination_PK PRIMARY KEY ( Source, Destination )
);


CREATE TABLE Water(
 Vehicle_Id          INTEGER   NOT NULL,
 CONSTRAINT Water_Vehicle_Reg_PK PRIMARY KEY ( Vehicle_Id)
);


CREATE TABLE Road(
 Vehicle_Id          INTEGER   NOT NULL,
 CONSTRAINT Road_Vehicle_Reg_PK PRIMARY KEY ( Vehicle_Id)
);


CREATE TABLE Air(
 Vehicle_Id          INTEGER   NOT NULL,
 CONSTRAINT Air_Vehicle_Reg_PK PRIMARY KEY ( Vehicle_Id)
);

CREATE TABLE Route_Stops(
 Route_Id     INTEGER      NOT NULL,
 Stop_Name    VARCHAR(25)  NOT NULL,
 CONSTRAINT Route_Stop_PK PRIMARY KEY ( Route_Id, Stop_Name )

);


CREATE TABLE Transport_Stops(
 Stop_Name    VARCHAR(25)   PRIMARY KEY
);
```

```
-- Constraints
ALTER TABLE Customer
 ADD CONSTRAINT Customer_County_Street_FK FOREIGN KEY ( County, Street )
  REFERENCES PostalCode ( County, Street )
   ON DELETE CASCADE;


ALTER TABLE Company
 ADD CONSTRAINT Company_County_Street_FK FOREIGN KEY ( County, Street )
  REFERENCES PostalCode ( County, Street )
   ON DELETE CASCADE;


ALTER TABLE Transport_Office
 ADD CONSTRAINT Transport_Office_County_FK FOREIGN KEY ( County, Street )
  REFERENCES PostalCode ( County, Street )
   ON DELETE CASCADE;


ALTER TABLE PostalCode
 ADD CONSTRAINT PostalCode_City_Id_FK FOREIGN KEY ( City_Id )
  REFERENCES City ( City_Id )
   ON DELETE CASCADE;


ALTER TABLE City_Structures
 ADD CONSTRAINT City_Structures_City_Id_FK FOREIGN KEY ( City_Id )
  REFERENCES City ( City_Id )
   ON DELETE CASCADE;



ALTER TABLE City_Structures
 ADD CONSTRAINT City_Structures_Name_FK FOREIGN KEY ( Name )
  REFERENCES Buildings_Structures ( Name )
   ON DELETE CASCADE;


ALTER TABLE Registration_Details
 ADD CONSTRAINT Reg_Det_Comp_Name_FK FOREIGN KEY ( Company_Name )
  REFERENCES Company ( Company_Name )
   ON DELETE CASCADE;

ALTER TABLE Registration_Details
 ADD CONSTRAINT Reg_Det_Ofc_Id_FK FOREIGN KEY ( Office_Id )
  REFERENCES Transport_Office ( Office_Id )
   ON DELETE CASCADE;


ALTER TABLE Registration_Details
 ADD CONSTRAINT Reg_Det_Route_Id_FK FOREIGN KEY ( Route_Id )
  REFERENCES Route ( Route_Id )
   ON DELETE CASCADE;


ALTER TABLE Registration_Details
```

```
  ADD CONSTRAINT Reg_Det_Schedule_Id_FK FOREIGN KEY ( Schedule_Id )
    REFERENCES Vehicle_Schedule ( Schedule_Id )
      ON DELETE CASCADE;


ALTER TABLE City_Route
  ADD CONSTRAINT City_Route_Route_Id_FK FOREIGN KEY ( Route_Id )
    REFERENCES Route ( Route_Id )
      ON DELETE CASCADE;


ALTER TABLE City_Route
  ADD CONSTRAINT City_Route_City_Id_FK FOREIGN KEY ( City_Id )
    REFERENCES City ( City_Id )
      ON DELETE CASCADE;


ALTER TABLE Ticket
  ADD CONSTRAINT Ticket_Source_Dest_FK FOREIGN KEY ( Source, Destination )
    REFERENCES Travel_Amount ( Source, Destination )
      ON DELETE CASCADE;


ALTER TABLE Ticket
  ADD CONSTRAINT Ticket_SSN_FK FOREIGN KEY ( SSN )
    REFERENCES Customer ( SSN )
      ON DELETE CASCADE;


ALTER TABLE Ticket
  ADD CONSTRAINT Ticket_Vehicle_Reg_Id_FK FOREIGN KEY ( Vehicle_Id, Registration_Number )
    REFERENCES Transportation_Vehicle ( Vehicle_Id, Registration_Number )
      ON DELETE CASCADE;


ALTER TABLE Transportation_Vehicle
  ADD CONSTRAINT Tran_Vehicle_Reg_No_FK FOREIGN KEY ( Registration_Number )
    REFERENCES Registration_Details ( Registration_Number )
      ON DELETE CASCADE;


ALTER TABLE Water
  ADD CONSTRAINT Water_Vehicle_Reg_Id_FK FOREIGN KEY ( Vehicle_Id, Registration_Number )
    REFERENCES Transportation_Vehicle ( Vehicle_Id, Registration_Number )
      ON DELETE CASCADE;


ALTER TABLE Road
  ADD CONSTRAINT Road_Vehicle_Reg_Id_FK FOREIGN KEY ( Vehicle_Id, Registration_Number )
    REFERENCES Transportation_Vehicle ( Vehicle_Id, Registration_Number )
      ON DELETE CASCADE;


ALTER TABLE Air
  ADD CONSTRAINT Air_Vehicle_Reg_Id_FK FOREIGN KEY ( Vehicle_Id, Registration_Number )
    REFERENCES Transportation_Vehicle ( Vehicle_Id, Registration_Number )
      ON DELETE CASCADE;
```

```
ALTER TABLE Route_Buildings
  ADD CONSTRAINT Route_Buildings_Route_Id_FK FOREIGN KEY ( Route_Id )
    REFERENCES Route ( Route_Id )
      ON DELETE CASCADE;


ALTER TABLE Route_Buildings
  ADD CONSTRAINT Route_Buildings_Name_FK FOREIGN KEY ( Name )
    REFERENCES Buildings_Structures ( Name )
      ON DELETE CASCADE;


ALTER TABLE Bridges
  ADD CONSTRAINT Bridges_Name_FK FOREIGN KEY ( Name )
    REFERENCES Buildings_Structures ( Name )
      ON DELETE CASCADE;


ALTER TABLE Highways
  ADD CONSTRAINT Highways_Name_FK FOREIGN KEY ( Name )
    REFERENCES Buildings_Structures ( Name )
      ON DELETE CASCADE;


ALTER TABLE Tolls
  ADD CONSTRAINT Tolls_Name_FK FOREIGN KEY ( Name )
    REFERENCES Buildings_Structures ( Name )
      ON DELETE CASCADE;


ALTER TABLE Route_Stops
  ADD CONSTRAINT Route_Stops_Route_Id_FK FOREIGN KEY ( Route_Id )
    REFERENCES Route ( Route_Id )
      ON DELETE CASCADE;


ALTER TABLE Route_Stops
  ADD CONSTRAINT Route_Stops_Stop_Name_FK FOREIGN KEY ( Stop_Name )
    REFERENCES Transport_Stops ( Stop_Name )
      ON DELETE CASCADE;
```

5 Trigger

5.1 Insert/Update of Route Id

This trigger will pop up whenever there is an insertion of new route ID or an update of a route ID in Route table.

The trigger is defined as follow:

```
CREATE or REPLACE TRIGGER Route_Changes

BEFORE INSERT or UPDATE OF Route_Id ON Route

FOR EACH ROW

DECLARE

BEGIN
   dbms_output.put_line('Route Id has been added/updated');

END;
```

5.2 Log all the changes about Customer

This trigger will keep track of all the changes made in the customer table. Whenever, a customer changes their information, this trigger will pop up.

The trigger is defined as follow:

```
CREATE TABLE Customer_log (

  Name     VARCHAR(50),
  Phone    INTEGER,
  County   VARCHAR(20),
  Street   VARCHAR(20),
  Log_Date DATE

);


CREATE OR REPLACE TRIGGER Customer_changes

  BEFORE UPDATE OF Name, Phone, County, Street ON Customer
  FOR EACH ROW

BEGIN

  INSERT INTO Customer_log (Name, Phone, County, Street, Log_Date)

  VALUES (:new.Name, :new.Phone, :new.County, :new.Street, SYSDATE);

END;
```

6 Procedure

6.1 Fetch Tolls for a given company

This procedure would take company name in its argument and find all the toll names that the company has to pay for. All the vehicles are registered to some company and vehicles in their daily schedule would pass through several tolls for which they would have to pay, hence, the company would take care of all the payments for their vehicles The procedure is defined as follow:

```
CREATE OR REPLACE PROCEDURE Company_Tolls_Pay ( Comp_Name IN VARCHAR2) AS

    name      Tolls.Name%type;

    CURSOR TollNames is

           SELECT T.Name
           FROM Registration_Details D, Route_Buildings B, Buildings_Structures S, Tolls T
           WHERE D.Company_Name = Comp_Name AND D.Route_Id = B.Route_Id
           AND B.Name = S.Name AND S.Name = T.Name;

    BEGIN

    OPEN TollNames;

    LOOP

           FETCH TollNames INTO name;

           EXIT WHEN (TollNames%NOTFOUND);

           dbms_output.put_line(name);

    END LOOP;

    CLOSE TollNames;

END Company_Tolls_Pay;
```

6.2 Find stops for a Vehicle

This procedure would take vehicle ID in its argument and find all the stop names the vehicle would stop at. This is important for customers who want to stop at certain location when they are traveling on that particular vehicle. The procedure is defined as follow:

```
CREATE OR REPLACE PROCEDURE Vehicle_Stops (
 VID IN INTEGER
) AS
  stops     Route_Stops.Stop_Name%type;

  CURSOR StopNames is

  SELECT S.Stop_Name
  FROM Transportation_Vehicle V, Registration_Details D, Route R, Route_Stops S
  WHERE V.Vehicle_Id = VID AND V.Registration_Number = D.Registration_Number
  AND D.Route_Id = R.Route_Id AND R.Route_Id = S.Route_Id;
```

```
BEGIN
  OPEN StopNames;
  LOOP
        FETCH StopNames INTO stops;
        EXIT WHEN (StopNames%NOTFOUND);
        dbms_output.put_line(stops);
  END LOOP;
  CLOSE StopNames;
END Vehicle_Stops;
```