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 | Requirements and Assumptions | |  | 3 | |
| 2 | Enhanced Entity-Relation Diagram | |  | 5 | |
| 3 | Mapping EER Diagram to Relational Schema and Normalization | | | 6 | |
| 4 | SQL |  |  | 11 | |
| 5 | Trigger | |  | 17 | |
|  | 5.1 | Insert/Update of Route Id . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 17 | |
|  | 5.2 | Log all the changes about Customer . . . . . . . . . . . . . . . . . . . . . . . | | 17 | |
| 6 | Procedure | |  | 18 | |
|  | 6.1 | Fetch Tolls for a given company. . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 18 | |
|  | 6.2 | Find stops for a Vehicle . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 19 | |

* 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.

* Enhanced Entity-Relation Diagram

The EER is shown in fig. 1.

Figure 1: EER for Transportation System

A picture containing text

Description automatically generated

* 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

A close up of a map

Description automatically generated

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(Ticket\_id,Quantity,Source,Destination,SSN,Vehicle\_Id,Registration\_Number)**

**Travel\_Amount(Source, Destination, 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(Company\_name, Street, County)**

**Address(County, Street, 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

**A close up of a map

Description automatically generated**

* 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;

* 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;

* 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;