Smart Traffic Management Database

Chrysikos Christos, 9432 | Mavridis Antonis, 9563 | Gitopoulos Giorgos, 9344

Databases @ ECE AUTh, November 2021

1. Introduction

This project deals with the theoretical design of a database of a proposed application. After the description of the application and its data requirements, we will present the structure of the database and its user types. Moreover, some examples will be used in order to explain the database functionality.

1.1. Application Description

The proposed application concerns *traffic management* in a *smart city*. A *smart city* is a technologically modern urban area that uses different types of electronic methods, voice activation methods and sensors to collect specific data. That data is used to improve the operations across the city, such as traffic management.
The purpose of the application is to match each vehicle with the fastest feasible route in order to reach the desirable destination and help the drivers to locate available parking slots easier. Also, it aims to detect traffic violations and store them. First of all, we assume a computing cloud that will perform all the necessary computations for the application. In addition, a database will be needed to store all the collected data.
Concerning the application, the exact position of each vehicle will be collected by GPS, it will be stored in the database and a traffic metric will be computed for each area of the city according the number of vehicles in the area. Every driver will select its destination and the cloud will export the optimal route for this vehicle, using a *Shortest Path Algorithm* (i.e. *Dijkstra's Algorithm*) and taking into account the neighboring areas, the traffic metric of each area, the traffic lights and the tolls. Also, vision sensors will be placed in every parking slot and the data of their status will be stored to the database. So, the drivers will be able to find the empty parking slots in a specific area using the application. About the traffic violations, cameras will be placed through the city streets and in the traffic lights too, in order to detect speed limit or road signs violations and violations of red traffic lights respectively.

1.2. Data Requirements

Some of the data that the database should store, with the corresponding estimations of storage size per day, are:

- the users with their id, name, age and gender (500,000 $\sim$ 10,000,000 users),
- the vehicles with their licence plate, type, driver id, location, destination and route ($\sim$ 1 vehicle per user),
- the areas of the city with their id, name and traffic metric (20 $\sim$ 50 areas),
- the couples of neighbouring areas in order to declare the routes ($\sim$ 3 neighbouring areas per area),
- the routes that are exported by the cloud (as a sequence of areas *) with their id and duration ($\sim$ 3 routes per vehicle),
- the traffic lights with their details (5,000 $\sim$ 20,000 traffic lights),
- the tolls with their details (3 $\sim$ 10 tolls),
- the parking slots with their details (5,000,000 $\sim$ 20,000,000 parking slots)
- the traffic violations and their details (5,000 $\sim$ 20,000 violations)

* Note that the smaller each area is, the more precise the route is. For simplicity we assume larger areas of a city.

2. User Categories & Requirements

In this section, we will present the types of users of the database and their corresponding data access requirements.

- **End user:** The end users of the application are the drivers of the vehicles. They can access all the data of their own vehicle, their own route, the data of every area of the city, the tolls, the traffic lights and the data of every parking slot. They are able to write in the database just their own destination, while all the other data are read - only for them.
- **Administrator:** The administrators can read and modify all the data of the database. They are responsible for the proper operation of the system and they should take action if they detect any type of problem.
- **Sensor - Embedded system:** The several sensors with their embedded systems will be considered as users of the database, as they are able to write and modify data. For example, a system of a parking slot sensor updates the status of the slot in the database, every time that it changes and the embedded system of a camera writes down the plate number of a vehicle and its fee in the violations table, when it detects that it violated the speed limit. The embedded systems of the sensors have no access to other data and cannot read data at all. They can just write in or modify some specific memory addresses.

- **Cloud:** The computing cloud has access to all the data concerned to the route computation. It can read the current location of every vehicle to compute the traffic metric of every area as a fuzzy variable and writes it in the table of the areas. Also, it reads the location of the traffic lights, the tolls, the destination of every vehicle, the neighbouring areas and the computed traffic metric of each area in order to compute the fastest route for each vehicle. When it does that, it writes the route id in the corresponding line of the vehicle table, the route id and its duration in the route table and the areas that the route passes through in the route - area table (Note that the tables will be presented in depth in section 4).

3. Entity – Relationship Model

3.1. Description

The **entities** of the **Entity - Relationship** model are: *User*, *Area*, *Route*, *Vehicle*, *Violations*, *Tolls*, *Parking Slot*, and *Traffic Light*. For every *User* there is at least one *Vehicle*, each *Vehicle* can only be located in one *Area* at a time and can follow only one *Route* at time, each *Vehicle* must have a starting and a destination *Area*, each *Route* is consisted of many different *Areas* and each *Area* can be a part of many different *Routes*. An *Area* can have *Parking Slots*, *Tolls* and *Traffic Lights*. Also, each *Area* has its own neighboring *Areas* and each *User* can commit one or more *Violations*.
The description of the **attributes** of our **entities** follows:

- For every ***User*** there must be a *name*, a *gender*, a unique *id* which is the primary key, an *age* and a *mobile number*.
- Every ***Vehicle*** must have a unique *license plate* and a *type* which defines the type of the car (SUV, small Car, large Car, Limo, etc).
- Every ***Area*** has a unique *id* which is the primary key, a *name* and must have a *traffic metric* which shows the traffic situation in the *Area*.
- Every ***Route*** has a unique *id* which is the primary key and a *duration* which determines how long the *Route* is estimated to last (in minutes).
- Every ***Violation*** has a unique *id*, a *fee* and a *type* (red light, stop sign, speeding etc…).
- A ***Parking Slot*** has a unique *id*, an exact *location* and a *status* which shows if it is occupied or not.
- ***Traffic Light*** has a unique *id*, a *status* (red or green), an exact *location* and a *duration* (in minutes).
- ***Tolls*** have an *id*, a *location* and a *toll* which defines the price of the *Tolls*.

Some **assumptions** that will be made are:

- Every *Vehicle* has a unique *User*. There can be no *Vehicle* with two different owners.
- A *Vehicle* can have alternating locations depending on time. For example, if a Vehicle traverses through a *Route* it will change from two to multiple locations in order to arrive in the final destination.
- An *Area* can have multiple *Vehicles* simultaneously.
- The current location of each *Vehicle* is being updated by a *GPS tracking unit* in a fixed time interval schedule.
- The *Route* of each *Vehicle* is decided by the *computing cloud* that uses a *Shortest Path Algorithm*, taking into account the *traffic metrics*, the *Traffic lights* and the *Tolls* of the *Areas*. For example, the *User* with *id* = 1, starts from *Area* with *id* = 3, chooses the destination *Area* with *id* = 10, the *computing cloud*, taking into consideration the parameters that were mentioned previously, decides that the optimal *Route* is the one with *id* = 2, which passes through *Areas* with *id* = 1, 4, 6, 10 and each *Area* is neighboring to the previous and the next one.
- Two different *Vehicles* can be matched with the same *Route*.
- The *traffic metric* of an *Area* is computed as a fuzzy value after counting the *Vehicles* that are located in the *Area* and is updated every a standard amount of minutes.
- The *status* of every *Parking Slot* is being updated by the embedded systems of the *sensors* that are placed in the *Parking Slot*.
- The *Violations* are recorded by *cameras* and are automatically written to the *database*.

3.2. Entities

In the next tables, our **entities** are explained further:

Entity Name	User
Description	Entity where we store every user in the Database
Properties	Strong Entity
Attributes	• id • name • age • gender • mobile_number

Entity Name	Vehicle
Description	Entity where we save every Vehicle in the Database
Properties	Weak Entity - Cannot exist without a User
Attributes	• license_plate • type

Entity Name	Area
Description	Entity where we store every Area in the Database
Properties	Strong Entity
Attributes	• id • name • traffic_metric

Entity Name	Route
Description	Entity where we store every Route in the Database
Properties	Strong entity
Attributes	• id • destination

Entity Name	Violation
Description	Entity where we save every traffic Violation in the Database
Properties	Weak Entity - Cannot exist without User
Attributes	• id • fee • type

Entity Name	Parking Slot
Description	Entity where we store every Parking Slot in the Database
Properties	Weak Entity - Demands an Area
Attributes	• id • location • status

Entity Name	Traffic Light
Description	Entity where we store every Traffic Light in the Database
Properties	Weak Entity - Demands an Area
Attributes	• id • status • location • duration

Entity Name	Tolls
Description	Entity where we save every Toll in the Data Base
Properties	Weak Entity - Demands an Area
Attributes	• id • location • toll

Figure 1: *Entities*

3.3. Relationships

The **relationships** between the defined *entities* are presented below:

Relationship Name	User - own - Vehicle
Description	Every user must have 1 to many Vehicles and every vehicle must have a unique user as owner
Properties	Owns, Identifying
Cardinality	1:1...N
Dependency	User fully dependant, vehicle partial dependant
Attributes	-

Entity Name	Vehicle - follow - Route
Description	Many Vehicles can share the shame Route.
Properties	Follow, Identifying
Cardinality	N:1
Dependency	Vehicle fully dependant, route partial dependant
Attributes	-

Relationship Name	Vehicle - is located in - Area
Description	There can be many Vehicles in the same Area
Properties	Loacated in, Identifying
Cardinality	N:1
Dependency	Vehicle fully dependant, Area partial dependant
Attributes	-

Relationship Name	Vehicle - has a starting and a destination - Area
Description	Many different Vehicles can share the same starting and destination Area.
Properties	has-A, Identifying
Cardinality	N:1
Dependency	Vehicle fully dependant, Area partial dependant
Attributes	-

Relationship Name	Routes - pass through - Areas
Description	Many Areas can be a part of a Route and and many routes can share the same Area
Properties	Pass through, Identifying
Cardinality	M:N
Dependency	Route fully dependant, Area partial dependant
Attributes	-

Relationship Name	Is - neighboring - to
Description	Every Area can be a neighbor to any other Area
Properties	Neighboring, Unary
Cardinality	M:N
Dependency	Area partial dependant
Attributes	-

Figure 2: *Relationships (i)*

Relationship Name	Area - has - Parking Slot
Description	Every Area can have from 0 to many Parking Slots
Properties	has-A, Identifying
Cardinality	1:N
Dependency	Area partial dependant, Parking Slot fully dependant
Attributes	-

Entity Name	Area - has - Tolls
Description	Every Area can have from 0 to many Tolls
Properties	has-A, Identifying
Cardinality	1:N
Dependency	Area partial dependant, Tolls fully dependant
Attributes	-

Entity Name	Area - has - Traffic Light
Description	Every Area can have from 0 to many Traffic Light
Properties	has-A, Identifying
Cardinality	1:N
Dependency	Area partial dependant, Traffic Light fully dependant
Attributes	-

Relationship Name	User - commit - Violation
Description	Every user can have 0 to many Violations
Properties	Commits, Identifying
Cardinality	1:N
Dependency	Violation fully dependant, User partial dependant
Attributes	-

Figure 3: *Relationships (ii)*

3.4. Entity – Relationship Diagram

The **Entity - Relationship** diagram of the *database* based on the defined *entities* and *relationships* is the following:

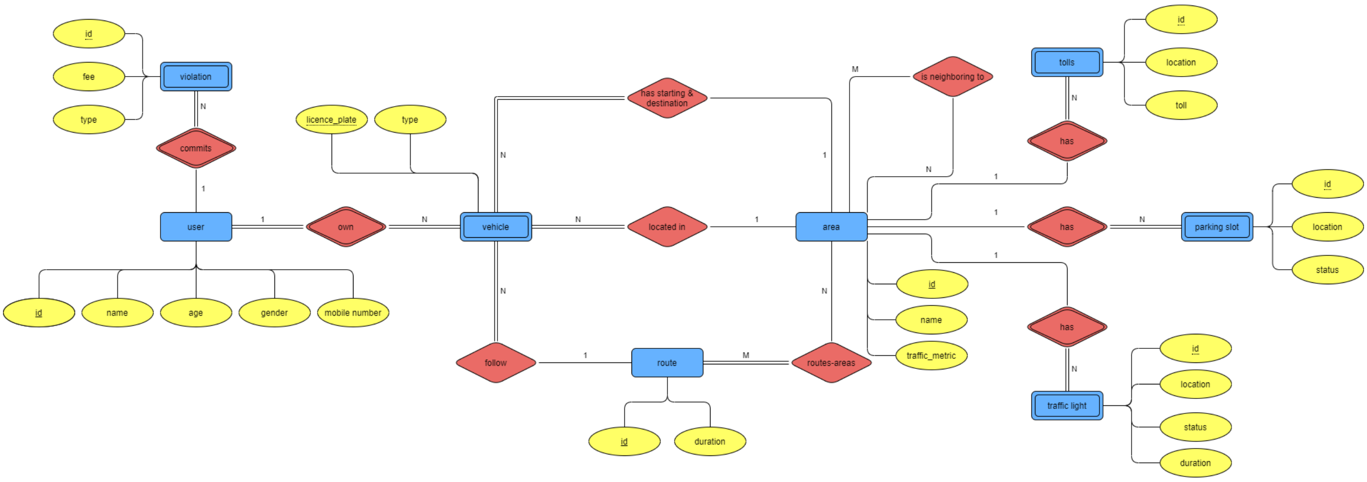

Figure 4: *Entity – Relationship Diagram*

4. Relational Model

4.1. Data Domains

4.2. Relations

4.3. Relational Diagram

4.4. Views

5. Examples

5.1. Table Examples

5.2. Relational Algebra Examples