

# 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 and construction of a database of a proposed application. After the description of the application and its data requirements, we will present the user types and the structure of the database (*Entity - Relationship model* and *Relational model*). Moreover, some examples will be used in order to explain the database functionality. The repository of the project can be found [here](#).

### 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 to 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, gender and mobile number (500,000 ~ 10,000,000 users),
- the vehicles with their licence plate, type, driver id, location, destination and route (~ 1 vehicle per user),
- the areas of the city with their id, name and traffic metric (20 ~ 50 areas),
- the couples of neighboring areas in order to declare the routes (~ 3 neighboring areas per area),
- the routes that are exported by the cloud (as a sequence of areas \*) with their id and duration (~ 3 routes per vehicle),
- the traffic lights with their details (5,000 ~ 20,000 traffic lights),
- the tolls with their details (3 ~ 10 tolls),
- the parking slots with their details (1,000,000 ~ 20,000,000 parking slots)
- the traffic violations and their details (5,000 ~ 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 their personal data, the data of their own vehicle, their own route, the data of every area of the city, the tolls and the data of every empty parking slot. They are able to write in the database just their own data and their vehicle data, while all the other data that were mentioned 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.
- **Violation officer:** Police officers responsible for traffic violations will have read and write rights on the traffic violations and read rights on the information of the users who commit the violations and their vehicles.

- **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 driver id (automatically found by the plate number of the 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 value 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 neighboring 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* consists 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 vehicle (car, motorcycle, truck, 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*.
- The *Neighboring Areas* (see section 4) define all the couple of *Areas*, so that a *Vehicle* can pass from the first to the second and are used by the *computing cloud* to define valid *Routes*. For example, a *Route* can start from *Area* with *id* = 1, pass through *Area* with *id* = 2 and reach *Area* with *id* = 3, only if there are the *Neighboring Areas* couples 1-2 and 2-3 in the respective table.
- If the *cloud* computes that the optimal *Route* for a *Vehicle* is the one mentioned in the previous bullet, it will declare an *id* for that *Route* (i.e. *id* = 105), it will write that *id* in the table of *Vehicles* next to the specific *Vehicle* and it will also add the *id* of the *Route* with all the *ids* of the *Areas* it passes through in a *Route - Areas* table (it will be described in section 4). So, it will write the couples (105, 1), (105, 2), (105, 3).
- As noted in subsection 1.2 too, we consider large *Areas* for simplicity. The optimal approach would be to consider short pieces of streets as *Areas* in order to increase the precision of the *Routes*.

### 3.2. Entities

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

<b>Entity Name</b>	User	<b>Entity Name</b>	Vehicle
	Entity where we store every user in the Database		Entity where we save every Vehicle in the Database
	Strong Entity		Weak Entity - Cannot exist without a User
	<ul style="list-style-type: none"> <li>• <u>id</u></li> <li>• name</li> <li>• age</li> <li>• gender</li> <li>• mobile_number</li> </ul>		<ul style="list-style-type: none"> <li>• <u>license_plate</u></li> <li>• type</li> </ul>
<b>Entity Name</b>	Area	<b>Entity Name</b>	Route
	Entity where we store every Area in the Database		Entity where we store every Route in the Database
	Strong Entity		Strong entity
	<ul style="list-style-type: none"> <li>• <u>id</u></li> <li>• name</li> <li>• traffic_metric</li> </ul>		<ul style="list-style-type: none"> <li>• <u>id</u></li> <li>• destination</li> </ul>
<b>Entity Name</b>	Violation	<b>Entity Name</b>	Parking Slot
	Entity where we save every traffic Violation in the Database		Entity where we store every Parking Slot in the Database
	Weak Entity - Cannot exist without User		Weak Entity - Demands an Area
	<ul style="list-style-type: none"> <li>• <u>id</u></li> <li>• fee</li> <li>• type</li> </ul>		<ul style="list-style-type: none"> <li>• <u>id</u></li> <li>• location</li> <li>• status</li> </ul>
<b>Entity Name</b>	Traffic Light	<b>Entity Name</b>	Tolls
	Entity where we store every Traffic Light in the Database		Entity where we save every Toll in the Data Base
	Weak Entity - Demands an Area		Weak Entity - Demands an Area
	<ul style="list-style-type: none"> <li>• <u>id</u></li> <li>• status</li> <li>• location</li> <li>• duration</li> </ul>		<ul style="list-style-type: none"> <li>• <u>id</u></li> <li>• location</li> <li>• toll</li> </ul>

Figure 1: *Entities*

### 3.3. Relationships

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

<b>Relationship Name</b>	User - own - Vehicle	<b>Entity Name</b>	Vehicle - follow - Route
<b>Description</b>	Every user must have 1 to many Vehicles and every vehicle must have a unique user as owner	<b>Description</b>	Many Vehicles can share the same Route
<b>Properties</b>	Owes, Identifying	<b>Properties</b>	Follow, Identifying
<b>Cardinality</b>	1:N	<b>Cardinality</b>	N:1
<b>Dependency</b>	User fully dependant, vehicle partial dependant	<b>Dependency</b>	Vehicle fully dependant, route partial dependant
<b>Attributes</b>	-	<b>Attributes</b>	-

<b>Relationship Name</b>	Vehicle - is located in - Area	<b>Relationship Name</b>	Vehicle - has a starting and a destination - Area
<b>Description</b>	There can be many Vehicles in the same Area	<b>Description</b>	Many different Vehicles can share the same starting and destination Area
<b>Properties</b>	Located in, Identifying	<b>Properties</b>	has-A, Identifying
<b>Cardinality</b>	N:1	<b>Cardinality</b>	N:2
<b>Dependency</b>	Vehicle fully dependant, Area partial dependant	<b>Dependency</b>	Vehicle fully dependant, Area partial dependant
<b>Attributes</b>	-	<b>Attributes</b>	-

<b>Relationship Name</b>	Routes - pass through - Areas	<b>Relationship Name</b>	Is - neighboring - to
<b>Description</b>	Many Areas can be a part of a Route and many routes can share the same Area	<b>Description</b>	Every Area can be a neighbor to any other Area
<b>Properties</b>	Pass through, Identifying	<b>Properties</b>	Neighboring, Unary
<b>Cardinality</b>	M:N	<b>Cardinality</b>	M:N
<b>Dependency</b>	Route fully dependant, Area partial dependant	<b>Dependency</b>	Area partial dependant
<b>Attributes</b>	-	<b>Attributes</b>	-

Figure 2: Relationships (i)

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

<b>Entity Name</b>	Area - has - Traffic Light	<b>Relationship Name</b>	User - commit - Violation
<b>Description</b>	Every Area can have from 0 to many Traffic Light	<b>Description</b>	Every user can have 0 to many Violations
<b>Properties</b>	has-A, Identifying	<b>Properties</b>	Commits, Identifying
<b>Cardinality</b>	1:N	<b>Cardinality</b>	1:N
<b>Dependency</b>	Area partial dependant, Traffic Light fully dependant	<b>Dependency</b>	Violation fully dependant, User partial dependant
<b>Attributes</b>	-	<b>Attributes</b>	-

Figure 3: Relationships (ii)

### 3.4. Entity – Relationship Diagram

The **entity - relationship** diagram of the *database* based on the defined *entities* and *relationships*, according to *Chen's notation* is the following:

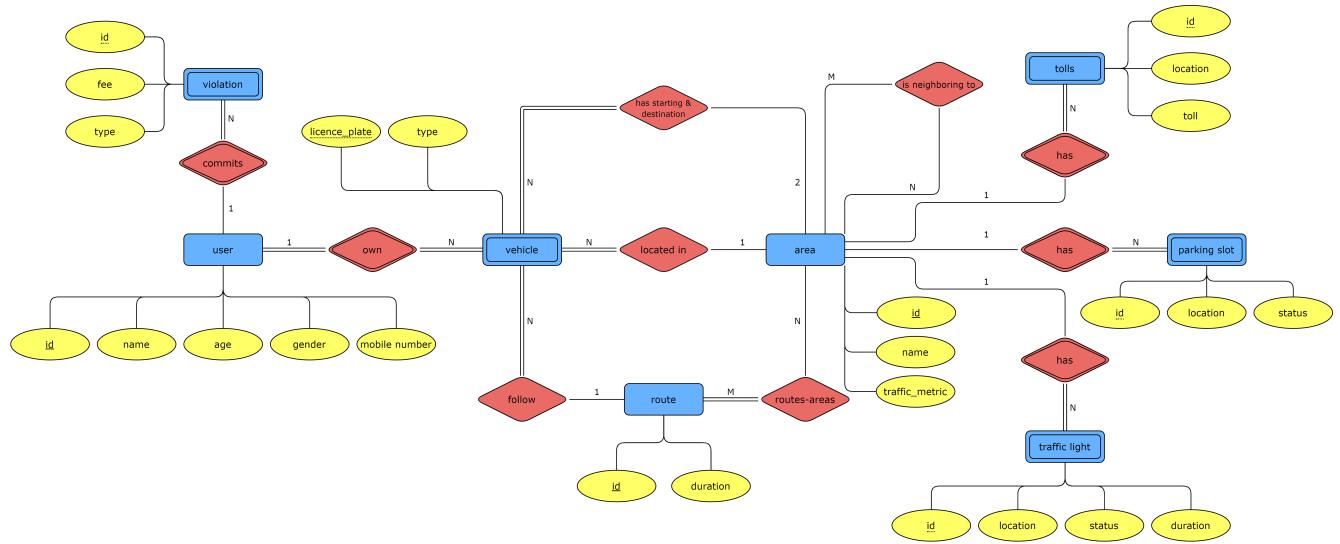


Figure 4: Entity – Relationship Diagram

## 4. Relational Model

### 4.1. Data Domains

The **data domains** of the *relational model* are presented below:

Domain	Data Type
id	CHAR (10)
name	VARIABLE_CHAR (30)
age	INTEGER_RANGE [18,99]
gender	ENUMERATED {MALE, FEMALE}
mobile_number	CHAR (10)
integer	INT
violation_type	ENUMERATED {TRAFFIC_LIGHT, SPEED_LIMIT, TRAFFIC_SIGN}
vehicle_type	ENUMERATED {CAR, MOTORCYCLE, TRUCK}
money	FLOAT
metric	ENUMERATED {VERY_LOW, LOW, NORMAL, HIGH, VERY_HIGH}
duration	INT
location	CHAR (24)
parking_slot_status	ENUMERATED {EMPTY, NOT_EMPTY}
traffic_light_status	ENUMERATED {GREEN, RED}

Figure 5: Data Domains

## 4.2. Relations

The **relations** of the *relational model* are shown in the next tables:

User		Vehicle	
attributes :		attributes :	
Name	Domain	Name	Domain
id	id	licence_plate	id
name	name	type	vehicle_type
age	age	user_id	id
gender	gender	route_id	id
mobile_number	mobile_number	current_area_id	id
constraints :		constraints :	
Primary Key	id	Primary Key	licence_plate & user_id
Foreign Key	-	Foreign Key	user_id ~> User route_id ~> Route current_area_id ~> Area starting_area_id ~> Area destination_area_id ~> Area

Area		Route	
attributes :		attributes :	
Name	Domain	Name	Domain
id	id	id	id
name	name	duration	integer
traffic_metric	metric	constraints :	
Primary Key	id	Primary Key	id
Foreign Key	-	Foreign Key	-

Figure 6: *Relations (i)*

Violation		Traffic Light	
<b>attributes :</b>		<b>attributes :</b>	
<b>Name</b>	<b>Domain</b>	<b>Name</b>	<b>Domain</b>
id	id	id	id
type	violation_type	status	traffic_light_status
fee	money	location	location
user_id	id	duration	integer
<b>constraints :</b>		<b>constraints :</b>	
Primary Key	id & user_id	Primary Key	id & area_id
Foreign Key	user_id ~> User	Foreign Key	area_id ~> Area
Parking Slot		Tolls	
<b>attributes :</b>		<b>attributes :</b>	
<b>Name</b>	<b>Domain</b>	<b>Name</b>	<b>Domain</b>
id	id	id	id
location	location	location	location
status	parking_slot_status	toll	money
area_id	id	area_id	id
<b>constraints :</b>		<b>constraints :</b>	
Primary Key	id & area_id	Primary Key	id & area_id
Foreign Key	area_id ~> Area	Foreign Key	area_id ~> Area
Routes - Areas		Neighboring Areas	
<b>attributes :</b>		<b>attributes :</b>	
<b>Name</b>	<b>Domain</b>	<b>Name</b>	<b>Domain</b>
route_id	id	area1_id	id
area_id	id	area2_id	id
<b>constraints :</b>		<b>constraints :</b>	
Primary Key	route_id & area_id	Primary Key	area1_id & area2_id
Foreign Key	route_id ~> Route area_id ~> Area	Foreign Key	area1_id ~> Area area2_id ~> Area

Figure 7: Relations (ii)

#### 4.3. Relational Diagram

The relational diagram follows:

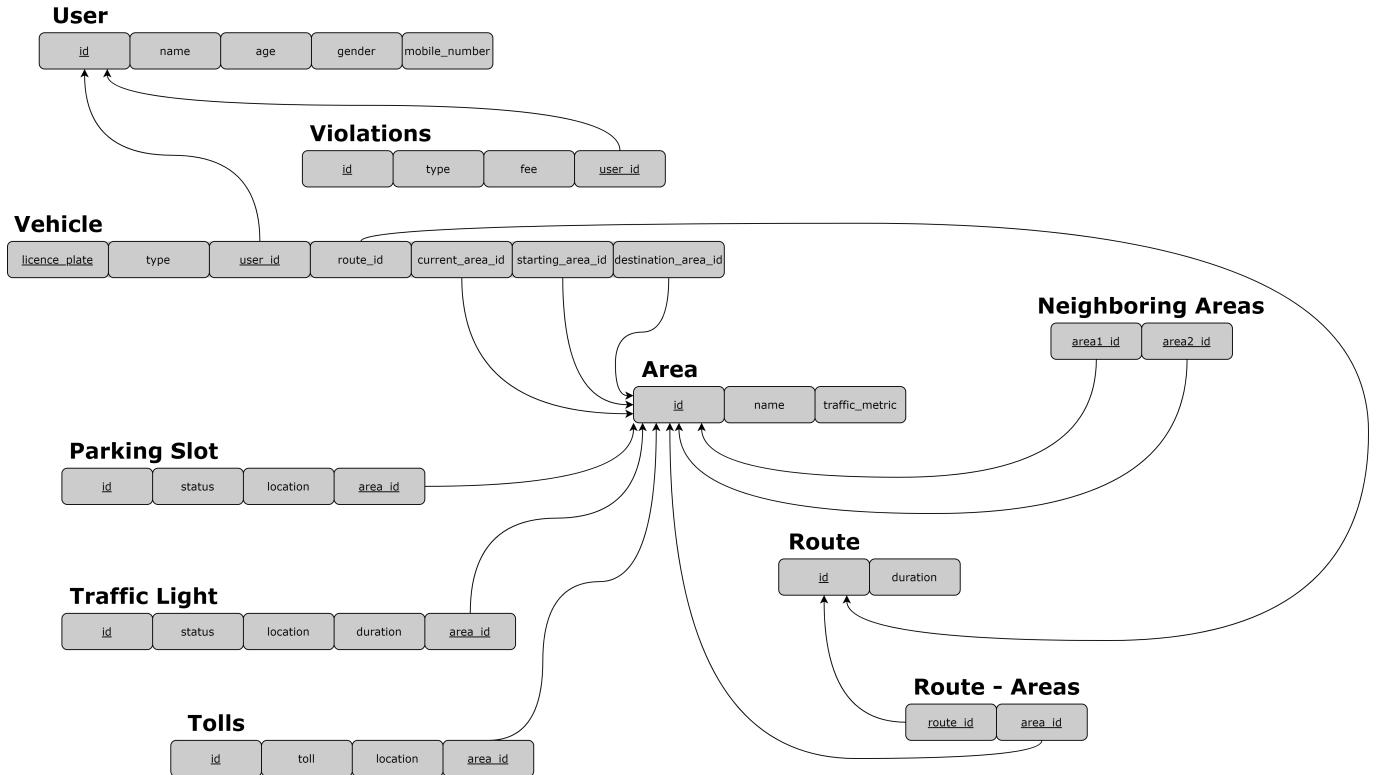


Figure 8: Relational Diagram

#### 4.4. Views

Now, we will define **database views** in order to store some data that will be accessed by the *database users* frequently. An example of instance of the two first views is presented in subsection 5.1.

- The first view will consists of all the *Area names* and the empty *Parking Slots* of the corresponding *Area*. It will be used by the *Users* who are trying to locate an empty *Parking Slot* in a specific *Area*. The view is expressed in *relational algebra* as:

$$\left( \pi_{name,id}(Area) \right) \bowtie \left( \pi_{area\_id,location} (\sigma_{status=EMPTY} (ParkingSlot)) \right)$$

where *id* and *area\_id* are considered as common attributes for the *natural join* operation, or

$$\pi_{name,area\_id,location} \left( \sigma_{id=area\_id} \left( \left( \pi_{name,id}(Area) \right) \times \left( \pi_{area\_id,location} (\sigma_{status=EMPTY} (ParkingSlot)) \right) \right) \right)$$

- Another view to be defined, is the *Users* who commit *Violations* with the *Violation details*. It will be accessed by *Violation officers* in order to see the information of *Users* who committed every *Violation*. In *relational algebra* it is defined as:

$$\pi_{user\_id,name,age,gender,mobile\_number,type,fee} ((User) \bowtie (Violation))$$

where *id* and *user\_id* are considered as common attributes for the *natural join* operation, or

$$\pi_{user\_id,name,age,gender,mobile\_number,type,fee} \left( \sigma_{id=user\_id} \left( (User) \times \pi_{type,fee,user\_id}(Violation) \right) \right)$$

- The last view consists of the *types* of the *Vehicles* next to their *current area id* and *name*. It is useful for the cloud in order to compute the *traffic metric* of the *Areas*:

$$\sigma_{id,name,type} ((Area) \bowtie (Vehicle))$$

## 5. Examples

### 5.1. Table Examples

Subsequently, we will present an example of an **instance** of the *database*, that is all the data stored in the tables at a particular moment of time. The estimation for the stored data size can be found in 1.2.

User				
<b>id</b>	<b>name</b>	<b>age</b>	<b>gendre</b>	<b>mobile_number</b>
AI05069901	Jim Brooks	32	MALE	6972461286
IK13114254	Steve Smith	26	MALE	6921673161
KE01019933	Chloe Brown	36	FEMALE	6981620227
BN08028872	John Davis	52	MALE	6915623763
OM27126296	Marie Miller	21	FEMALE	6993357512

Vehicle						
<b>licence_plate</b>	<b>type</b>	<b>user_id</b>	<b>route_id</b>	<b>current_area_id</b>	<b>starting_area_id</b>	<b>destination_area_id</b>
GRTH639633	CAR	AI05069901	THPAFA3163	THTO920993	THPA237310	THFA021224
GRTH699821	CAR	IK13114254	THPOTO0912	THFA021224	THPO651209	THTO920993
GRTH098331	MOTORCYCLE	KE01019933	THKETO3441	THKE835612	THKE835612	THTO920993
GRTH707342	TRUCK	BN08028872	THPOPA0121	THPO651209	THPO651209	THPA237310
GRTH788242	CAR	OM27126296	THKEPO9884	THKE835612	THKE835612	THPO651209

Violation			
<b>id</b>	<b>type</b>	<b>fee</b>	<b>user_id</b>
PP3781AI01	TRAFFIC_SIGN	20	KE01019933
SA9312IK54	TRAFFIC_LIGHT	300	IK13114254
KF9571KE33	SPEED_LIMIT	90	KE01019933
YT1267BN72	SPEED_LIMIT	60	OM27126296
EZ5205OM96	TRAFFIC_SIGN	40	OM27126296

Figure 9: *Database Instance (i)*

Area			Route	
<b>id</b>	<b>name</b>	<b>traffic_metric</b>	<b>id</b>	<b>duration (min)</b>
THKE835612	Center	VERY_HIGH	THPAFA3163	20
THPA237310	Panorama	LOW	THPOTO0912	25
THTO920993	Toumpa	HIGH	THKETO3441	15
THFA021224	Faliro	NORMAL	THPOPA0121	30
THPO651209	Polichni	VERY_LOW	THKEPO9884	15

Parking Slot			
<b>id</b>	<b>location</b>	<b>status</b>	<b>area_id</b>
THKE412320	41.427462,23.652820	EMPTY	THPA237310
THPA292425	39.523312,24.535225	NOT_EMPTY	THPA237310
THTO421920	42.327402,19.865920	NOT_EMPTY	THTO920993
THFA402225	40.623412,22.955825	EMPTY	THFA021224
THPO282225	38.628712,22.005225	NOT_EMPTY	THTO920993

Traffic Light				
<b>id</b>	<b>status</b>	<b>location</b>	<b>duration (sec)</b>	<b>area_id</b>
THKETL3267	GREEN	40.424462,22.654820	35	THKE835612
THPATL2572	RED	40.887462,21.692820	25	THPA237310
THTOTL1903	RED	39.426662,22.677820	20	THTO920993
THFATL0212	RED	42.427462,24.652880	5	THFA021224
THPOTL9032	GREEN	40.400062,23.699920	30	THPO651209

Figure 10: Database Instance (ii)

Tolls			
<b>id</b>	<b>location</b>	<b>toll</b>	<b>area_id</b>
THKET95127	41.488462,21.659920	1.2	THKE835612
THPAT09978	40.517462,21.882820	0.7	THPA237310
THTOT12842	39.462662,22.677820	1	THTO920993
THFAT09051	42.429962,24.611880	1.1	THFA021224
THPOT65481	41.473062,23.699920	0.9	THPO651209

Neighboring Areas	
<b>area1_id</b>	<b>area2_id</b>
THPA237310	THTO920993
THTO920993	THFA021224
THPO651209	THFA021224
THFA021224	THTO920993
THKE835612	THTO920993
THPO651209	THPA237310
THKE835612	THPO651209

Route - Areas		
<b>route_id</b>	<b>area_id</b>	
THPAFA3163	THFA021224	
THPAFA3163	THPA237310	
THPAFA3163	THTO920993	
THPOTO0912	THPO651209	
THPOTO0912	THTO920993	
THPOTO0912	THFA021224	
THKETO3441	THKE835612	
THKETO3441	THTO920993	
THPOPA0121	THPO651209	
THPOPA0121	THPA237310	
THKEPO9884	THKE835612	
THKEPO9884	THPO651209	

Empty parking slots in Areas (View)		
<b>name</b>	<b>area_id</b>	<b>location</b>
Panorama	THPA237310	41.427462,23.652820
Faliro	THFA021224	40.623412,22.955825

Users that committed Violations (View)						
<b>user_id</b>	<b>name</b>	<b>age</b>	<b>genre</b>	<b>mobile_number</b>	<b>type</b>	<b>fee</b>
KE01019933	Chloe Brown	36	FEMALE	6981620227	TRAFFIC_SIGN	20
IK13114254	Steve Smith	26	MALE	6921673161	TRAFFIC_LIGHT	300
KE01019933	Chloe Brown	36	FEMALE	6981620227	SPEED_LIMIT	90
OM27126296	Marie Miller	21	FEMALE	6993357512	SPEED_LIMIT	60
OM27126296	Marie Miller	21	FEMALE	6993357512	TRAFFIC_SIGN	40

Figure 11: Database Instance (iii)

## 5.2. Useful Queries

Some useful queries examples that can be expressed using *relational algebra* on the tables of the *database* are:

- What are the *locations* of the *empty Parking Slots* in *Area: "Faliro"*? (Useful for a driver who is looking for an *empty Parking Slot*)

-Answer:

$$\pi_{location} \left( \pi_{id} (\sigma_{name=Faliro}(Area)) \bowtie \pi_{area\_id, location} (\sigma_{status=EMPTY}(ParkingSlot)) \right)$$

- What are the *Areas* with *low* or *very\_low traffic\_metric*? (Useful for the *computing cloud* in order to classify the *Areas* based on traffic)

-Answer:

$$\pi_{name} (\sigma_{traffic\_metric=LOW \vee traffic\_metric=VERY\_LOW}(Areas))$$

- What is the *mobile\_number* of the *User* who committed the *Violation* with *id = SA9312IK54*? (Useful for a *police officer* who wants to call a violator)

-Answer:

$$\pi_{mobile\_number} \left( \pi_{id, mobile\_number}(User) \bowtie \pi_{user\_id} (\sigma_{id=SA9312IK54}(Violations)) \right)$$

- What are the *types* of the *Vehicles* that are currently located in *Area* with *id = THKE835612* and are heading to *Area* with *id = THTO920993*? (Useful for the *computing cloud* in order to estimate the future *traffic\_metric* in the *Areas*)

-Answer:

$$\pi_{type} (\sigma_{current\_area\_id=THKE835612 \wedge destination\_area\_id=THTO920993}(Vehicle))$$

- How many *Vehicles* are currently located in *Area* with *id = THKE835612*, but are not motorcycles? (Can be used by the *computing cloud* in order to estimate the *traffic\_metric* of an *Area* with higher precision, as motorcycles should not be taken into account)

-Answer:

$$F_{count(license\_plate)} (\sigma_{current\_area\_id=THKE835612}(Vehicle) - \sigma_{current\_area\_id=THKE835612 \wedge type=MOTORCYCLE}(Vehicle))$$