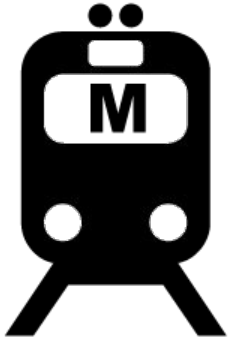
ADM Project:

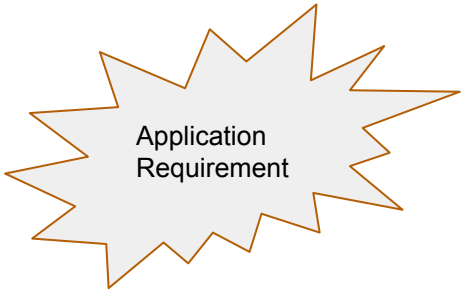
The design and the implementation of the distributed data management layer for the metro mobile application



Written By:

- ANKUR KUMAR
- NARGES ROKNI
- ANASTASIYA BOGUSLAVSKYA
- SOUAD BOUTANE

Introduction



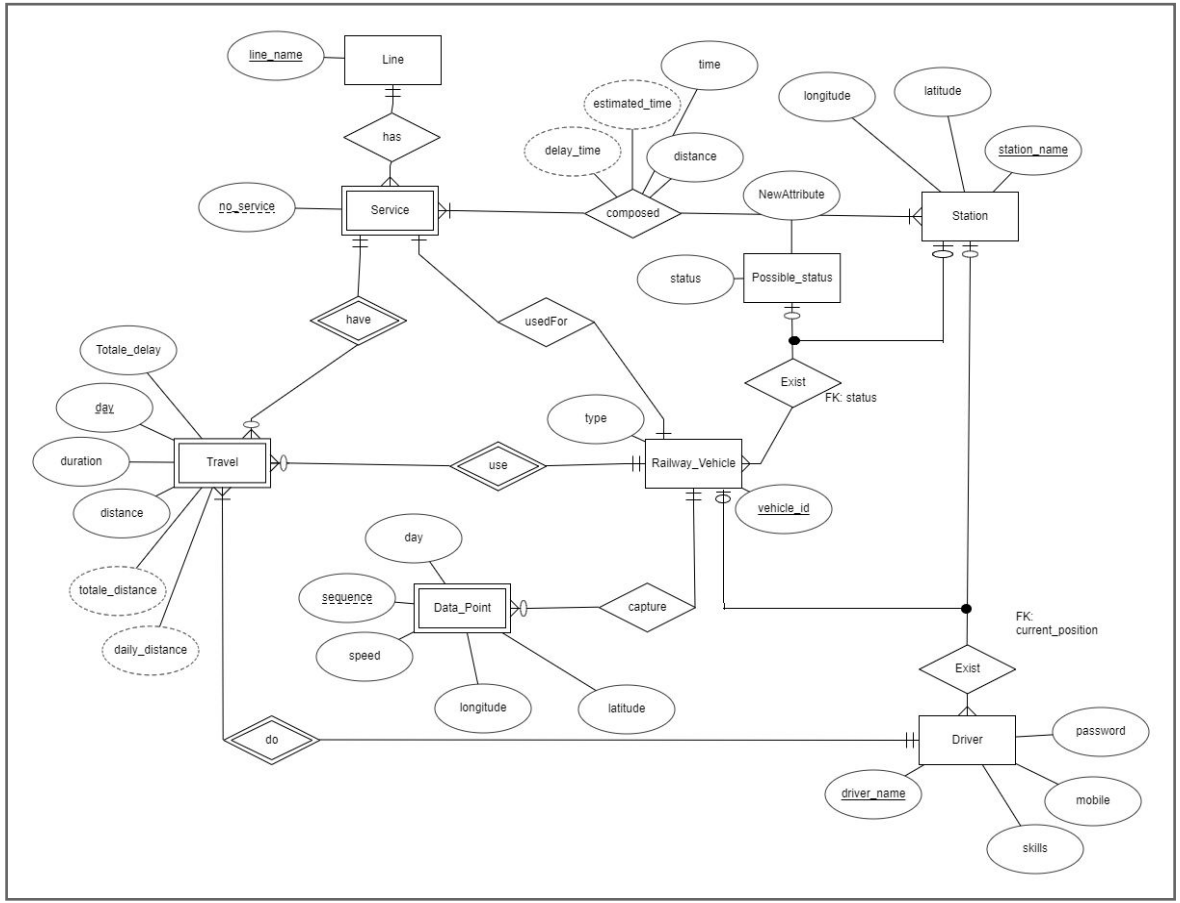
Plan

1. Conceptual Schema
2. Workload
3. Logical Schema
4. Database generation
5. CQL implementation

Conceptual Schema

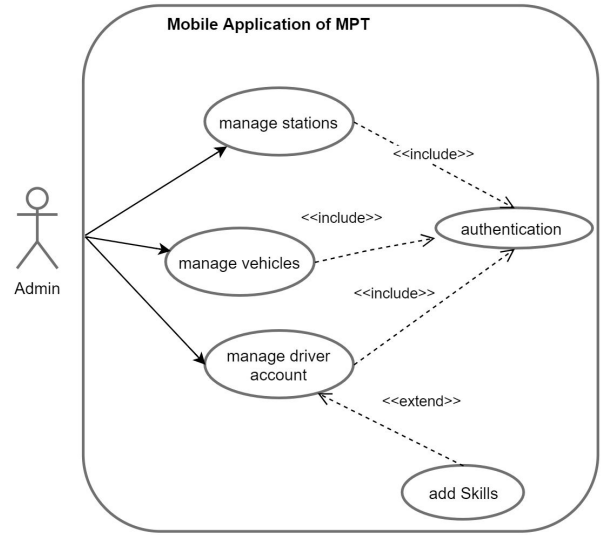
Conceptual Schema

To identify the problem and its specifications has been created ERD schema



Workload

WorkLoad 1: Administrator

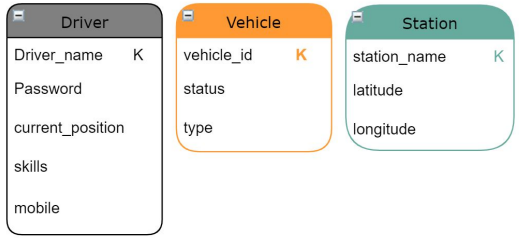


Use case for Admin

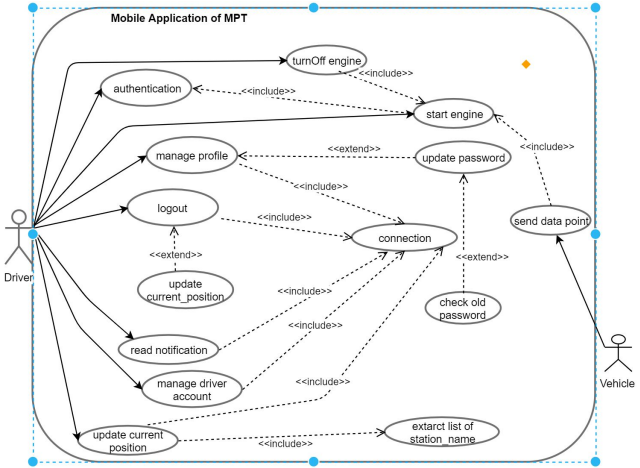
Queries according Admin role:

Class	**Queries**
Queries A for Station	Insert/remove/update a Station
Queries B for Vehicle	Insert/remove/update a Vehicle
Queries C for Driver	Insert/remove/update(skills) a Driver

First version of logical schema

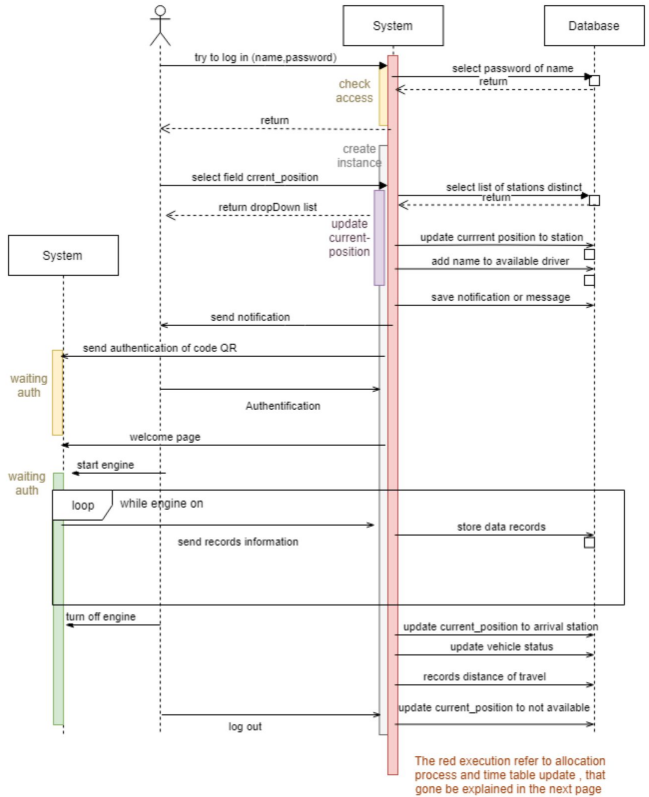


WorkLoad 2: Driver



Use case for Admin and vehicle

Sequence Diagram

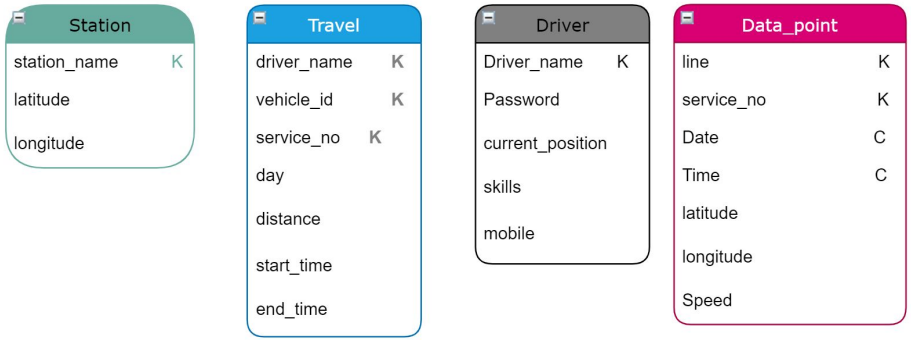


WorkLoad 2: Driver

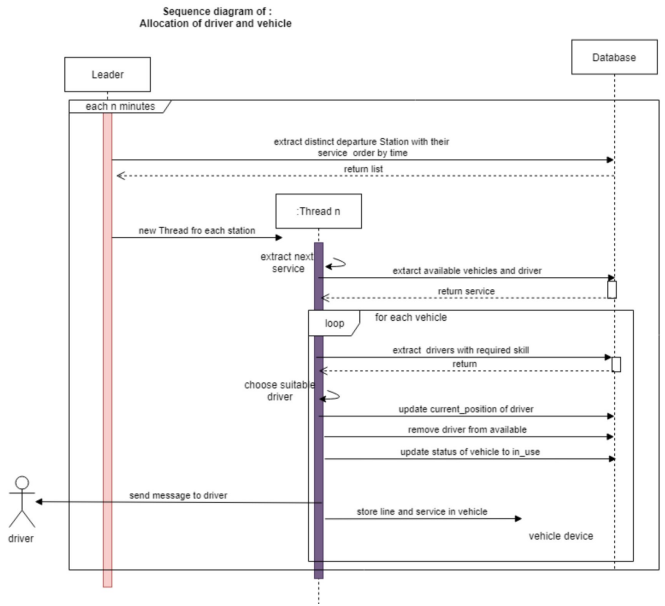
executed over different tables

Queries group	list of queries
Queries C on driver	1. select password of driver by his name 2. update current_position to choose station and 3. add notification
Queries A on station	1. select list of Station distinct
Queries E on data Point	1. insert data records
Queries D on Travel	1. record travel information

Second version of logical schema

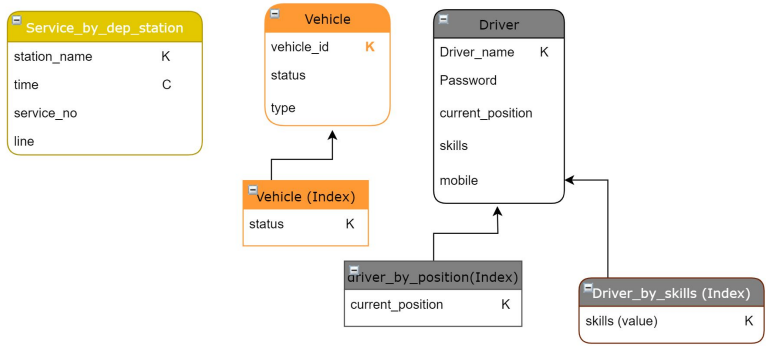


WorkLoad 3: Allocation of Vehicle and Driver

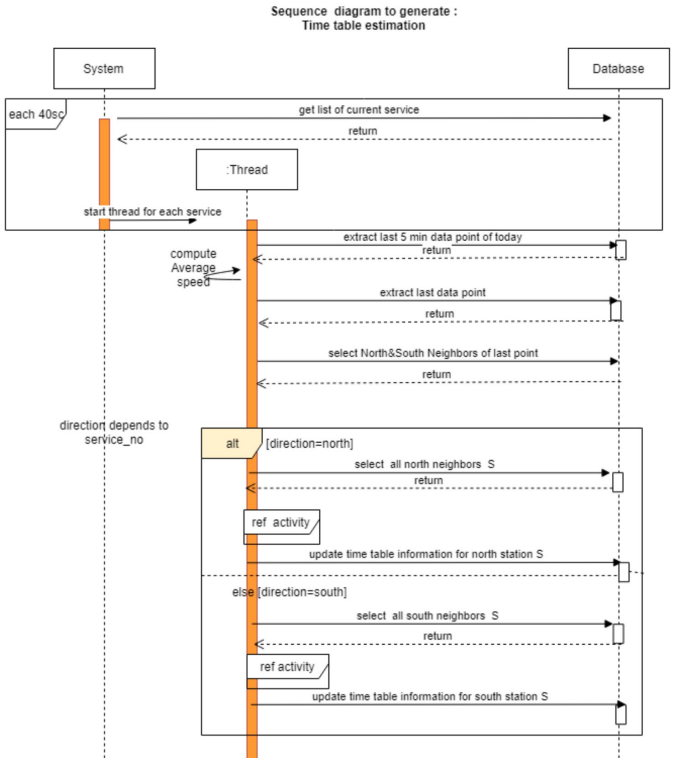


Queries group	list of queries
Queries F : service_by_dep_station	1. select list of distinct station with their services ordered by time departure.
Queries C : Driver with index on current position	1. Extract available drivers in a station 2. Extract the drivers that can use a specific vehicle 3. update current position of driver to vehicle_id
Queries C : Vehicle index on status	1. Extract available vehicle in a station 2. Update Vehicle_id to in_use

Third version of logical schema

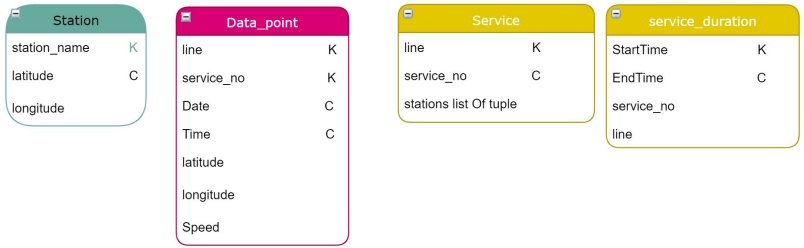


WorkLoad 4: Timetable upload



Queries group	list of queries
Queries F : service_duration	1. select list of current services
Queries: C on data Point	1. select last data point of a service 2. select data points of time interval on a day
Queries F: on Service	1. select total distance of a given service 2. select stations of a service
Queries A on station	1. select the//all north neighbors of a given latitude from list of station 2. select the/all south neighbors of a given latitude from list of station

Last version of logical schema



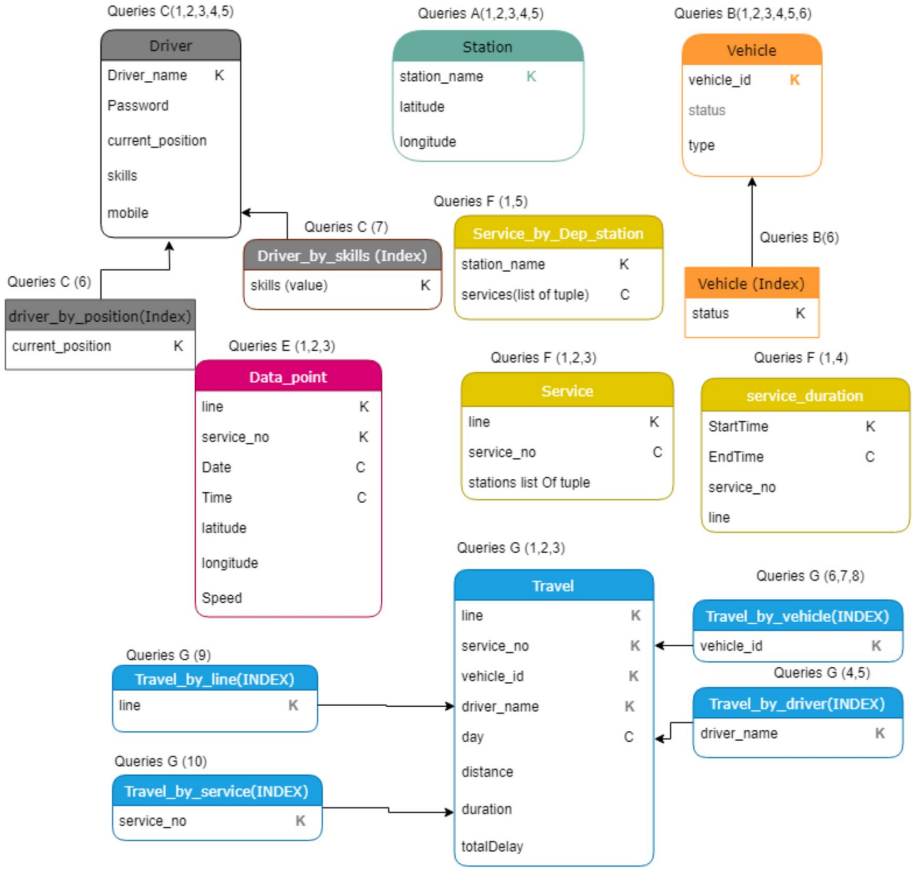
Logical Schema

Global Logical Schema

Based on the conceptual schema and the required workloads, the global logical schema schema for database layer after adding other queries for analysing the data base:

A 1. Insert a station by admin 2. Remove a station by admin 3. Select list of Station 4. select north neighbor 5. select south neighbor	**B** 1. Insert vehicle by Admin 2. Remove vehicle by Admin 3. Update vehicle by Admin 4. Update status 5. Select vehicle Type 6. Select Available vehicle in a given station (Index in Status
C 1. Insert driver by admin 2. Remove driver by admin 3. Update driver'skills by admin 4. Select password of driver by his name 5. Update current_position to selected station or a vehicle_id 6. Select Available driver in a given station (index) 7. Extract the drivers that can use a specific vehicle (index)	**E** 1. Insert data records 2. Select data points for a given Line_service in time interval of a day 3. Select last data points for a given Line,_service and day

F 1. Insert new service 2. Select distance of a service 3. Select list of stations of a service 4. Select current service with current time bigger than departure_time (distance=0) and small than arrivel/departure_time(distance !=0) => Service_duration(Index) 5. Select list of distinct station with their services ordered by time departure.	**G** 1. Insert new records after selecting (QueryF.2,QueryD.5) 2. Select total number travels What does it mean total travels 3. Select average delay What is Average delay 4. Select total delay of a driver on a day interval index (travel_by_driver) 5. Select total work time of a driver on a day interval index (travel_by_driver) 6. Select total delay for a given vehicle on a day interval index (travel_by_vehicle) 7. Select total distance for a given vehicle on a day interval index (travel_by_vehicle) 8. Select average daily distance for a given vehicle index (travel_by_vehicle) 9. Select total travel of a service (travel_by_service) 10. Select total travel of a line (travel_by_line)



Data Generation

Data Generation

Based on Real use case, based on London underground train, we collected station names and position, we divide them into line. Then we created all services , we implemented the simulation in order to compute all position .

80 drivers

167 000 data_points

217 stations

10 lines

80 vehicles

1000 services

CQL IMPLEMENTATION

CQL Implementation

Queries A on Station Table

```
CREATE TABLE Station (
    station_name text PRIMARY KEY,
    latitude float,
    longitude float
);
```

Queries B on Vehicle

```
CREATE TABLE Vehicle (
   vehicle_id text PRIMARY KEY,
   status text,
   type text
);
CREATE INDEX vehicle_status_idx ON ks_user27.vehicle (status);
```

Queries C on Driver Table

```
CREATE TABLE Driver (
      driver_name text PRIMARY KEY,
      password text,
      current_position text,
      skills set<text>,
      mobile txt,
);
CREATE INDEX driver_current_position_idx ON Driver (current_position);
CREATE INDEX driver_by_skill ON Driver (values(skills));
```

Queries F on services tables

```
CREATE TABLE Service_duration (
   starttime int,
   endtime int,
   line text,
   service_no smallint,
   PRIMARY KEY (starttime, endtime)
) WITH CLUSTERING ORDER BY (endtime ASC);

CREATE TABLE Service (
    line text,
    service_no smallint,
    stations list<tuple<int,text,float>>,
    PRIMARY KEY (line, service_no)
) WITH CLUSTERING ORDER BY (service_no ASC);

CREATE TABLE Service_by_dep_station (
    station_name text,
    services list<tuple<time,int,txt>>,
    PRIMARY KEY (station_name)
);
CREATE OR REPLACE FUNCTION lastOfList (input list<text>)
CALLED ON NULL INPUT
RETURNS float LANGUAGE java AS
'return Integer.valueOf(input.get(input.size()-1));
';
CREATE OR REPLACE FUNCTION tuple_distance(input tuple<int,text,float>)
CALLED ON NULL INPUT
RETURNS float LANGUAGE java AS
'return Integer.valueOf(input.get(input.size()-1));
';
```

Queries G on Travel table

```
CREATE TABLE ks_user27.travel (
    line text,
    service_no smallint,
    vehicle_id text,
    driver_name text,
    day int,
    distance float,
    duration int,
    totaldelay int,
    PRIMARY KEY ((line, service_no, vehicle_id, driver_name), day)
) WITH CLUSTERING ORDER BY (day ASC);

CREATE INDEX travel_by_line ON ks_user27.travel (line);
CREATE INDEX travel_by_vehicle ON ks_user27.travel (vehicle_id);
CREATE INDEX travel_by_driver ON ks_user27.travel (driver_name);
CREATE INDEX travel_by_service ON ks_user27.travel (service_no);
```

Queries E on Data_point tables

```
CREATE TABLE Data_point (
    line text,
    service_no smallint,
    date int,
    time int,
    latitude float,
    longitude float,
    speed int,
    vehicle_id text,
    PRIMARY KEY ((line, service_no), date, time)
) WITH CLUSTERING ORDER BY (date ASC, time ASC);
```

Thank you