Carpooling Management System Database Schema

By Prahlad Koratamaddi, UNI: pk2743 and Prateek Jain, UNI: pj2399

```
CREATE Table cars(
      vehicle_id        CHAR(8),
      car_type          VARCHAR(5) NOT NULL,
      seats             VARCHAR(6) NOT NULL,
      baggage           INTEGER NOT NULL,
      PRIMARY KEY(vehicle_id)
);
```

```
CREATE Table drivers(
      user_id           VARCHAR(6),
      passwd            VARCHAR(8) NOT NULL,
      full_name         VARCHAR(50) NOT NULL,
      gender            VARCHAR(10),
      phone             CHAR(10) NOT NULL,
      wallet            NUMERIC(10,2) NOT NULL,
      licence_id        CHAR(9) unique NOT NULL,
      rating            REAL NOT NULL,
      licence_expiry    DATE NOT NULL,
      PRIMARY KEY(user_id)
);
```

```
CREATE Table drives(
      user_id           VARCHAR(6),
      vehicle_id        VARCHAR(8),
      active            BOOL NOT NULL,
      PRIMARY KEY(user_id, vehicle_id),
      FOREIGN KEY(user_id) REFERENCES drivers
      ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY(vehicle_id) REFERENCES cars
      ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE Table passengers(
      user_id                       VARCHAR(6),
      passwd                        VARCHAR(8) NOT NULL,
      full_name                     VARCHAR(50) NOT NULL,
      gender                        VARCHAR(10),
      phone                         CHAR(10) NOT NULL,
      wallet                        NUMERIC(10,2) NOT NULL,
      default_car_type              VARCHAR(5),
      default_passenger_count       INTEGER NOT NULL,
      default_baggage               INTEGER NOT NULL,
      default_special_needs         BOOL NOT NULL,
      home_location                 NUMERIC(5,0),
      work_location                 NUMERIC(5,0),
      PRIMARY KEY(user_id),
      FOREIGN KEY(home_location) REFERENCES locations
      ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY(work_location) REFERENCES locations
      ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE Table locations(
      pincode NUMERIC(5,0),
      PRIMARY KEY(pincode)
);
```

```
CREATE Table links(
      start_pin NUMERIC(5,0),
      end_pin   NUMERIC(5,0),
      PRIMARY KEY (start_pin, end_pin),
      FOREIGN KEY (start_pin) REFERENCES locations
      ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY (end_pin) REFERENCES locations
      ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE Table is_in(
      user_id           VARCHAR(6),
      vehicle_id        VARCHAR(8),
      pincode           NUMERIC(5, 0),
      PRIMARY KEY(user_id, vehicle_id),
      FOREIGN KEY(user_id, vehicle_id) REFERENCES drives
      ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY(pincode) REFERENCES locations
      ON DELETE CASCADE ON UPDATE CASCADE
);
```

IMPORTANT NOTE: the below merged table “requests_converts_into” also merges “starts” and “ends” Relationship Sets from the ER Diagram. We chose not to mention them in the name of the table for brevity.

```
CREATE Table requests_converts_into(
      user_id           VARCHAR(6),
      request_id        NUMERIC(6, 0),
      ride_id           NUMERIC(16, 0),
      start_location    NUMERIC(5,0) NOT NULL,
      end_location      NUMERIC(5,0) NOT NULL,
      time_requested    TIMESTAMP WITHOUT TIME ZONE NOT NULL,
      car_type          VARCHAR(5),
      passenger_count   INTEGER NOT NULL,
      baggage           INTEGER NOT NULL,
      special_needs     BOOL NOT NULL,
      request_status    VARCHAR(10) NOT NULL,
      fare              NUMERIC(10, 2),
      PRIMARY KEY(user_id, request_id),
      FOREIGN KEY(ride_id) REFERENCES rides_travels
      ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY(user_id) REFERENCES passengers
      ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY(start_location) REFERENCES locations
      ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY(end_location) REFERENCES locations
      ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE Table rides_travels(
      ride_id           NUMERIC(16,0),
      ongoing           BOOLEAN NOT NULL,
      revenue           NUMERIC(10,2) NOT NULL,
      start_time        TIMESTAMP WITHOUT TIME ZONE NOT NULL,
      end_time          TIMESTAMP WITHOUT TIME ZONE NOT NULL,
      user_id           VARCHAR(6) NOT NULL,
      vehicle_id        VARCHAR(8) NOT NULL,
      seats_available   INTEGER NOT NULL,
      ride_rating       REAL NOT NULL,
      PRIMARY KEY (ride_id),
      FOREIGN KEY (user_id, vehicle_id) REFERENCES drives
      ON DELETE CASCADE ON UPDATE CASCADE
);
```

NOTE: Participation Constraint on **rides_travels** as part of the **converts_into** relationship can not be enforced in our SQL Schema, based on the rules provided in class so far.

Constraints not modeled by our SQL Schema (yet!):

drives

-> Only one driver-car pair (d,c) can have active = true among all possible driver-car pairs which has either the driver d or the car c
-> All driver-car pairs with active = true must be present in the **is_in** table
-> Driver-car pairs which have active = false must not be present in the **is_in** table
-> Driver-car pair can only have active = true if the driver's licence isn't expired

rides_travels

-> If a **rides_travels** instance has ongoing = true, then the instance of **drives** it is associated with must have active = true
-> Two **rides_travels** instances with overlapping durations can't have the same **drives** instance
-> In every **rides_travels** instance, start_time < end_time

requests_converts_into

-> To avoid unending rides, drivers can only approve requests in **requests_converts_into** whose requested_time is less than Ride's start_time + 4 hours
-> A passenger can’t create another request in **requests_converts_into** until all previous requests associated with that passenger have request_status as either "Completed" or “Failed”
-> Only requests in **requests_converts_into** which have request_status = “Accepted”, “Completed” can have a non-NULL ride_id
-> Only requests in **requests_converts_into** with request_status=”Requested”, “Approved” and “Failed” can have a NULL ride_id
-> **rides_travels** with ongoing=false can be associated with only request_status = “Completed” requests in **requests_converts_into**
-> start_location and end_location can not be the same in any request in **requests_converts_into**

links

-> The two participating **locations** (start_pin and end_pin) in **links** can not be the same

Other Constraints on Attributes (not mapped (yet!)):

drivers:
-> user_id: Minimum 6 characters
-> passwd: Minimum 8 characters
-> full_name: Only alphabets
-> gender: “Male”, “Female”, “Non-Binary”
-> phone: 10 digits
-> wallet: >=0 dollars, stores up to cents
-> rating: can only be between 0 to 5

cars:
-> vehicle_id: enforcing format - "[3 upper case letters] - [4 digits]"
-> car_type: only these types: “Mini”, “Sedan”, or “SUV”
-> seats: 3 for "Mini", 4 for "Sedan", 6 for "SUV"
-> baggage: 2 for "Mini", 3 for "Sedan", 4 for "SUV"

requests_converts_into:
-> requested_time <= Current database server time
-> passenger_count: upper bounded by car type’s seats (1 by default).
If no preference for car_type, then <= 6.
-> baggage_count: upper bounded by car type’s baggage (0 by default).
If no preference for car_type, then <= 4.
-> request_status: “Requested”, “Approved”, “Accepted”, “Failed”, “Completed”. The only possible flow of values is:

Initially	*-> Requested*
If Driver approves	*-> Approved, else -> Failed*
If User pays	*-> Accepted, else -> Failed*
If Ride completes	*-> Completed*

-> fare: >0 dollars.

passengers:
-> user_id: Minimum 6 characters
-> passwd: Minimum 8 characters
-> full_name: Only alphabets
-> gender: “Male”, “Female”, “Non-Binary”
-> phone: 10 digits
-> wallet: >=0 dollars, stores up to cents
-> default_car_type: if populated, it can only be either “Mini” or “Sedan” or “SUV”.
-> default_passenger_count: 1 by default
-> default_baggage: 0 by default
-> default_special_needs: false by default

rides_travels:
-> start_time <= Current database server time
-> revenue > 0 dollars, stores up to cents
-> seats_available: upper bounded by **drives** car_type
-> ride_rating: 0 to 5 (default is driver's current rating at the start of the ride)