# Database Model for the carpooling application

29 July 2022

## Abstract

This document presents a first proposition of the model for the carpooling application. Its purpose is to present the different entities of the model and their attributes.

## Editors

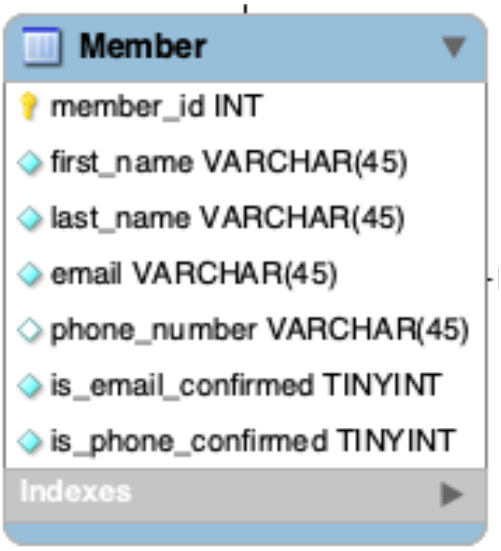
| Editor | Date | Changelog |
|---|---|---|
| Hamza ABOUHANIFA | 29/07/2022 | First draft |

# Entities

## Member

Every user that interacts with the carpooling platform needs to be an authenticated user, therefore the system has to store his information on the database. For This reason, we create the `Member` table :

- `member_id`: (PK) Identifier of the member
- `first_name`: First name of the member
- `last_name`: Last name of the member
- `email`: Email of the member
- `phone_number`: Phone number of the member
- `is_email_confirmed`: 1 if the email is verified else 0
- `is_phone_confirmed`: 1 if the phone is verified else 0

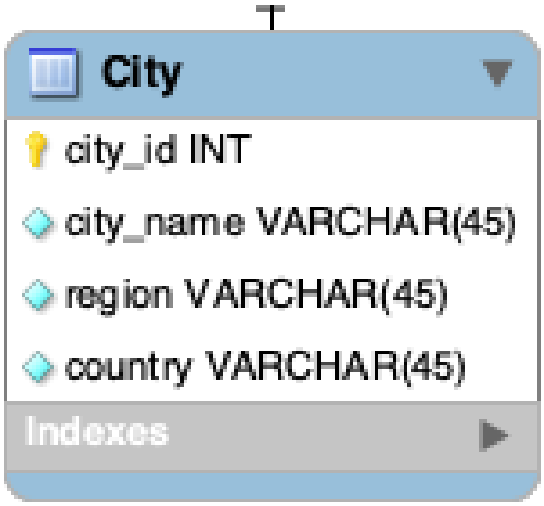


## City

When a user wants to publish a trip, he needs to choose an origin and a destination, ideally from a drop-down, which means that those locations must be stored in our database.

By creating a `city` table we have a list of cities to choose from:

- `city_id`: (PK) Identifier of the city
- `city_name`: Name of the city
- `region`: region where the city is located
- `state`: state where the city is located
- `country`: Country where the city is located



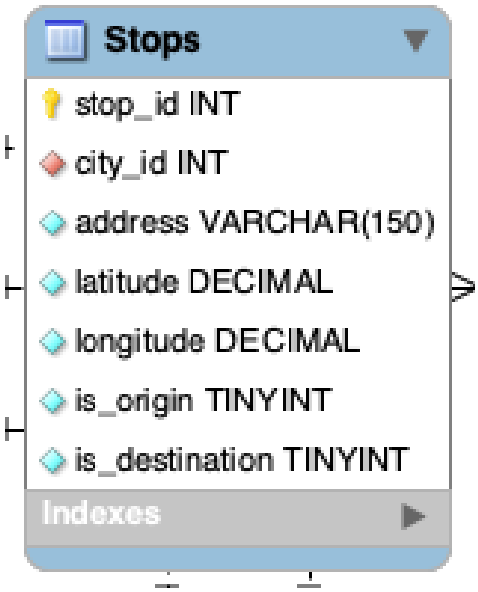

## Stops

The origin and destination are basically mandatory stops for the journey, therefore they will be stored as such in the `Stops` table, a table that will also store additional stops if the

driver chooses to add some to his journey. Once the user has chosen the origin and the destination of the trip, the application suggests the possible routes to take to get from point A to point B, and then the possible stops if the user chooses to make a stop in a particular city.

A table `Stops` stores these details:

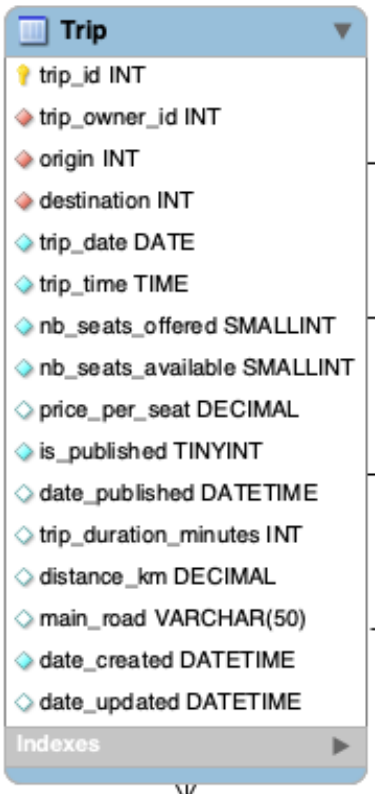
- `stop_id`: (PK) Identifier of the stop
- `city_id`: (FK) Identifier of the city
- `address`: Address of the stop
- `latitude`: Latitude of the stop point
- `longitude`: Longitude of the stop point
- `is_origin`: 1 if the stop is the starting point else 0
- `is_destination`: 1 if the stop is the destination else 0



# Trip

After confirming the route, the user then chooses the date and time of departure, he will also add the number of seats availble, these details would be stored in the `Trip` table:

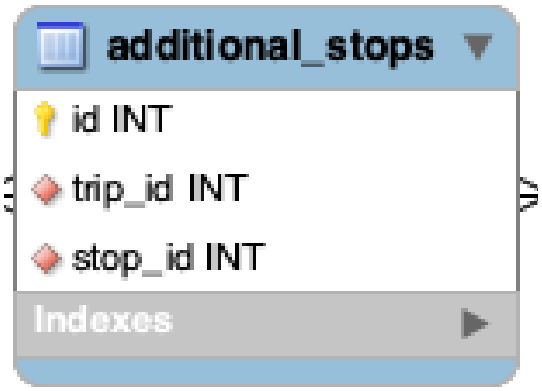
- `trip_id`: (PK) Identifier of the trip
- `trip_owner_id`: (FK) Identifier of the driver (Refrences `Member` table)
- `origin`: (FK) Identifier of the origin point (Refrences `Stops` table)
- `destination`: (FK) Identifier of the destination point (Refrences `Stops` table)
- `trip_date`: Date of the trip
- `trip_time`: Time of the trip
- `nb_seats_offered`: Number of seats the driver offers
- `nb_seats_available`: Number of seats available
- `price_per_seat`: Suggested price of the seat
- `is_published`: 1 if the trip is published else 0
- `date_published`: Date of the publication of the trip
- `trip_duration_minutes`: Duration of the trip in minutes
- `distance_km`: Overall distance of the trip in kilometers
- `main_road`: Name of the main road followed during the journey (Ex: Paris-Lyon via A6)
- `date_created`: Technical date of creation of the trip
- `date_updated`: Technical date of when the trip is updated




## Additional stops

The trip could have multiple additional stops, we will store them on a table of their own:

- `id`: (PK) Identifier of the additional stop
- `trip_id`: (FK) Identifier of the trip (Refrences `Trip` table)
- `stop_id`: (FK) Identifier of the stop point (Refrences `Stops` table)

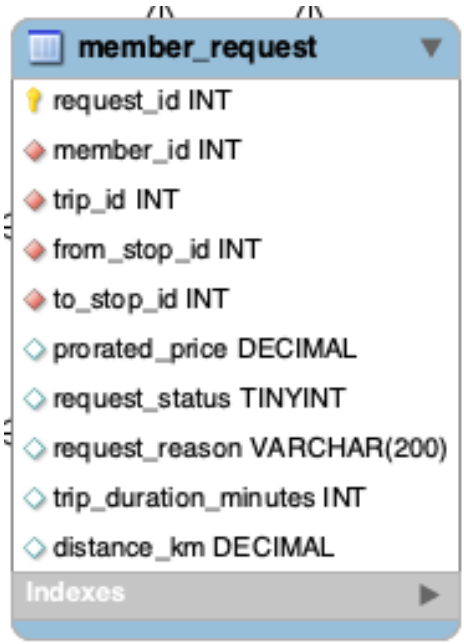



## Member Request

Once a trip is published, other users could request to join, and choose one of the stops published by the driver as their starting point or destination, the sub-trip would have a prorated price, and the driver could accept/reject their request.

A table `member_equest` stores these details:

- `request_id`: (PK) Identifier of the request
- `member_id`: (FK) Identifier of the member (Refrences `Member` table)
- `trip_id`: (FK) Identifier of the trip (Refrences `Trip` table)
- `from_stop_id`: (FK) Identifier of the starting point (Refrences `Stops` table)
- `to_stop_id`: (FK) Identifier of the destination point (Refrences `Stops` table)
- `prorated_price`: prorated price that the user have to pay for the trip
- `request_status`: 1 if the request is accepted by the driver, 0 if it is rejected
- `request_reason`: if the request is rejected, the driver could leave a custom message to explain the reason
- `trip_duration_minutes`: Duration of the sub-trip in minutes
- `distance_km`: Distance of the sub-trip in kilometers


member_request
request_id INT
member_id INT
trip_id INT
from_stop_id INT
to_stop_id INT
prorated_price DECIMAL
request_status TINYINT
request_reason VARCHAR(200)
trip_duration_minutes INT
distance_km DECIMAL
Indexes

# Full Data Model