

How do you contribute to the environment?

——CarbonFootprint

Name: Zike Tang

Table of Contents

Project Direction Overview	2
Use Cases and Fields	3
Structural Database Rules	7
Conceptual Entity-Relationship Diagram	10
Full DBMS Physical ERD.....	11
Stored Procedure Execution and Explanation	17
Question Identification and Explanations.....	20
Index Identification and Creations	23
History Table Demonstration	26
Data Visualizations	29
Summary and Reflection	32

Project Direction Overview

Why I'm interested in

New energy vehicles, such as electric vehicles and solar powered vehicles, are rushing into the car industry nowadays. Tesla, one of the biggest electric vehicle and clean energy company, is famous for its alternating-current power system. It occurs to me that the environmentally friendly energy powered transportation is emerging and how people contribute to the environmental protection matters to the environment at the same time. Therefore, I intend to design an application called *CarbonFootprint* to measure how individual contributes to the environmental protection by carbon dioxide level released by traditional transportation methods, such as flight, train, and car. It is designed to encourage people to take a green way to get around. The application can be downloaded in phone or be plugged on the website of user's laptop to track the person's transportation histories.

Database used for

The database is designed for company concerning about improving the environment especially for some non-profit organizations. Gathering general information of users, together with their transportation information from websites and applications, data can be analyzed to see how people choose on transportation for different activities.

Additionally, *CarbonFootprint* can show users how they contribute to the environment by calculating carbon dioxide emissions of different ways of transportation.

Use example

Here is a simple example indicating how a person can use this application. Jimmy drives his own car to work during weekdays. Last week, he went to Washington D.C. to enjoy the cherry blossom by flight and came back by train. The *CarbonFootprint* application automatically recorded his flight and train order information through plugin on websites or applications synchronization on phone. He also went to supermarket by bus on weekends because his car has been sent for repairment on weekends. The food delivery ordered by Jimmy at weekends is also included in the application from the takeaway platforms or apps. He also manually recorded the date he took the bus in *CarbonFootprint* application. The carbon dioxide level of all the means of transport will be calculated and presented as a green energy in *CarbonFootprint*. The company can analyze how people choose on transportation and Jimmy can get a sense of how he contributes to the green transportation so that he could choose a better transportation the next time.

Data contained

Database includes the basic information of individuals, transportation, and the gas emissions under different means of transport. More specifically, people's general information be collected when people first create the account. The company will record person's flight and train bookings on websites and mobile applications. Bus transport need to be manually recorded in *CarbonFootprint* application. Car driving, car-hailing and car using of food delivery will be updated in the database from navigation, car-hailing apps and delivery services respectively. Data regarding carbon dioxide emission level of different means of transport will also be included.

Use Cases and Fields

1. Account Signup/Installation Use Case

The first usage of the database is when a person signs up for an account on the website or in the *CarbonFootprint* application.

Here are the use case details:

1. The person visits the *CarbonFootprint's* website or app store and installs the application.
2. For new user, the person is asked to create a new account. Basic information is required to be fulfilled and it will be added to the database storing user's information.
3. When either an application account or a website account is setup, the person can login the account by both website and application.

Basic information collected when creating new account is listed below:

Field	What it Stores	Why it's Needed
Id	This is the identification number for each person who agrees to share their transportation and delivery information.	It can be used to identify individual user so that duplicates will not be made in case of there are people with the same other basic information.
FirstName	This is the first name of the person who agrees to share personal information on transportation and delivery.	It is necessary for displaying the person's name and contacting in the future if there is any data access problem.
LastName	This is the last name of the person who agrees to share personal information on transportation and delivery.	It is necessary for displaying the person's name and contacting in the future if there is any data access problem.
Gender	This specifies the gender of the person.	It is useful for further analysis, for instance, to find whether there is a difference between different gender on transportation and delivery.
Career	This is the career of the person.	It is useful for total carbon footprint comparison because some occupations are required to go on a business frequently, which will lead to more flight bookings.
Since	This stores the date of the account signed up by a person.	It records how long a person's information is agreed to be used for environmental analysis, which can also be considered as an indicator to show the person's contributions to environment improvement.

2. Automatic Flight or Train Track Use Case

The second usage of the database is when a flight is booked on the website or application and automatically recorded in the *CarbonFootprint* application.

Here are the use case details:

1. A person books a flight/train on the official flight/train website or in mobile flight/train application.
2. The application in phone or the plugin in website detects the booked flight or the train tickets and records related information in the database.

Either information of flight or train will be included in this single database because types of information concerning train and flight booking are almost the same. For example, the person's identification, departure date, start location and destination are required when booking a flight or train.

Information collected from flight or train bookings are detailed below:

Field	What it Stores	Why it's Needed
Id	This is the identification number for each person who agrees to share their transportation and delivery information.	It can be used to identify individual user so that duplicates will not be made in case of there are people with the same other basic information.
FlightorTrain	This specifies the type of transportation, indicating it is a flight or a train data.	As both train and flight information are included in the single database, specifying the specific types can be used to calculate gas elimination later for train and flight separately.
Date	This stores the date of the specific transportation.	It can be used to calculate the frequency of taking flight or train by a person on a monthly or yearly basis.
Departure	This is the start point of the transportation.	It can be used to make comparison of gas emission among different transportation methods.
Destination	This is the destination of the transportation.	It can be used to make comparison of gas emission among different transportation methods.
Miles	This is the mileage between the departure point and the destination by flight or train.	It can be used to calculate the gas elimination by a flight or a train.

3. Manual Bus Track Use Case

The third usage of the database is when a person goes by bus and the person is required to update the bus record in *CarbonFootprint* application.

Here are the use case details:

1. A person takes a bus and scan the transport card on the bus.
2. The person manually recorded the information of bus trip in the application.

Information collected from bus transport card are detailed below:

Field	What it Stores	Why it's Needed
BusDate	This stores the date of the bus taken.	It can be used to calculate the frequency of traveling by bus by a person on a monthly or yearly basis.
BusId	This is the number of bus transport card of a person.	It is used to record a person's bus use. One bus card might be shared by several people.
BusFrom	This is the first stop of the bus trip.	It can be used to calculate the gas elimination by a bus for a given route.
BusTo	This is the destination of the bus trip.	It can be used to calculate the gas elimination by a bus for a given route.

4. Automatic Car Track Use Case

The fourth usage of the database is when a car driving record, or a car-hailing order is, or a takeaway order is made and automatically recorded in the *CarbonFootprint* application.

Here are the use case details:

1. A person drives a car and use navigation to record the route.
2. A person uses car-hailing services.
3. A person orders a takeaway on food delivery platforms or applications.
4. The *CarbonFootprint* application gets access to the navigation, car-hailing application as well as delivery applications and automatically records related information in the database.

The route information created when a person drives a car, takes a taxi, or made a takeaway order will be obtained from navigation application, car-hailing apps, and delivery application respectively. As in the use case 2, information of driving a person's own car, taking a taxi, and making take-out delivery order is included in this single database because types of information concerning these three situations are almost the same. For example, the person's identification, date of driving a car, taking a taxi or making takeout order, start location and destination are all the similar types of information we could obtain from navigation, car-hailing apps and delivery services.

Information collected from navigation and car-hailing applications are detailed below:

Field	What it Stores	Why it's Needed
Id	This is the identification number for each person agree to share their transportation information.	It can be used to identify individual user so that duplicates will not be made in case of there are people with the same other basic information.
DriveType	This specifies the type of car driving, indicating whether it is a car driving by a person's own car, or a taxi driving, or a food delivery by car.	Although individual car, taxi information and food delivery are included in the single database, specifying the specific types can be used to make comparison between these three types of car driving.
CarDate	This stores the date of the car driving or date of using car-hailing.	It can be used to calculate the frequency of driving car or using car-

		hailing service by a person on a monthly or yearly basis.
Start	This specifies the starting point of a driving a car or taking a taxi.	It can be used to calculate the gas elimination by driving a car or taking a taxi based on the route distances between two locations.
WhereTo	This specifies the destination of a driving a car or taking a taxi.	It can be used to calculate the gas elimination by driving a car or taking a taxi based on the route distances between two locations.
CarMiles	This is the mileage between the departure point and the destination by driving a car or taking a taxi which can be obtain from the navigation or car-hailing application.	It can be used to calculate the gas elimination by driving a car or taking a taxi.

5. Emissions Use Case

The fifth usage of the database is when the application would like to calculate the gas emission under different methods of transportation.

Information of gas emissions are detailed below:

Field	What it Stores	Why it's Needed
Transportation	This is the transportation methods including airline, train, car, and bus driving.	It can be used to differentiate the gas emission condition among different transportation methods.
GasEmissions	This indicates the unit gas emissions of specific transportation methods.	It can be used to calculate the overall gas emission by a specific transportation method and compare it to other transportation methods.

Structural Database Rules

1. *Use case1: Account Signup/installation*

- a) The person visits the *CarbonFootprint*'s website or app store and installs the application.
- b) For new user, the person is asked to create a new account. Basic information is required to be fulfilled and it will be added to the database storing user's information.
- c) When either an application account or a website account is setup, the person can login the account by both website and application.

Components included in this use case are the *CarbonFootprint* application, the new user, and the database. Focusing on the structural database rules, **account** is the only one entity in this use case. It makes sense because there is no other entity or any relationship existing when a person creates a new account on *CarbonFootprint*. However, an account will have other connections with other entities in the following use cases when there are more entities come out.

2. *Use case2: Automatic Flight or Train Track*

- a) A person books a flight/train on the official flight/train website or in mobile flight/train application.
- b) The application in phone or the plugin in website detects the booked flight or the train tickets and records related information in the database.

In use case 2, **booking** and **tickets transport** (go around by flight or by train) are new entities in this use case. *CarbonFootprint* can track the bookings made on the website or in the phone application, including information on the departure date, the start point and destination. It makes sense that each booking is associated with an account since we need to match the booking to the person. Start from use case 2, I can create some structural database rules together with the entity in use case 1. Here follows the structural database rules:

1. Each account may be associated with many bookings and each booking is associated with an account.

Obviously, each person can make multiple flight or train bookings. In other words, one account corresponds to multiple orders on flight or train. On the other hand, each order can only be made by one account. It is also possible that an account exists in the database without any booking being made, if a person installs and just first runs the *CarbonFootprint* application and does not make any flight or train booking.

2. Each booking is made from a means of transport, and each means of transport has one to many bookings.

This rule indicates that every booking is associated with a kind of means of transport, and each kind of transportation have multiple bookings. I made booking to means of transport mandatory since a means of transport will not be stored in the database unless a booking is made on website or in phone application. There are multiple orders for one kind of means of transport. In the real world, the flight can be assigned to different airlines, but here we do not focus on the airline company because what matters is the carbon emission by plane.

3. *Use case3: Manual Bus Track*

- a) A person takes a bus and scan the transport card on the bus.
- b) The person manually recorded the information of bus trip in the application.

There is a new entity in this use case: **bus use**. The person is required to login in to *CarbonFootprint* application and manually record the date and other related information when the person takes a bus. Here follows the structural database rule in this use case:

1. Each account may be associated with many records of taking a bus and each bus record is associated with an account.

This rule indicates that each account can have many records of taking bus since a person can take a bus many times within one day or during a week. At the same time, each bus taken record is associated with one account because one record can only be made by one person. I made account to bus record optionally because it is possible that a person never takes the bus, or a person does not want to include the bus record because the person takes a bus not that often.

4. *Use case4: Automatic Car Track*

- a) A person drives a car and use navigation to record the route.
- b) A person uses car-hailing services.
- c) The application gets access to the navigation and car-hailing application and automatically records related information in the database.

There are two new entities in this use case: **car use** though driving a person's own car or using car-hailing service, and **car type** including private car driving, taxi driving and food delivery driving. CarbonFootprint application will tracks the route information, including date, starting point, and destination, when a person drives a car or uses car-hailing service. The entity in this use case will also be connected to the entity in the previous use cases. Here follows the structural database rules in this use case:

1. Each account may be associated with many car using records, and each car using records is associated with an account.

Each record of car using, whether by driving private car or using car-hailing service, is tied to one account because no record will be made without an account. For each account, or for each person, there is possibility that there is no car using account, which could happen when the person, especially when the user is a middle or high school student, does not own a car and does not use car-hailing service.

2. Each car using record results from one type of car driving, and each car driving type has one to many car using records.

This rule indicates that every car using record is associated with one kind of specific car driving situation, and each kind of car using situation have many car using records associated to it since a person can drives the private car, uses car-hailing service and food delivery application at the same time. I make the type of car using mandatory to car using records because types of car driving will not be stored in the database unless a car using is recorded.

5. Use case5: Emissions

This is a table includes the level of carbon emission of different means of transport including flight, train, bus, and car. The rule is obvious as follows:

1. One kind of means of transport corresponds one level of carbon emission and one level of carbon emission is associated with one kind of transportation.

The unit of carbon emission will be defined later and since different kinds of transport are classified, the level of carbon emission will differ among different transportation.

Specialization-Generalization rules

1. A booking is a flight booking or a train booking.

Kinds of booking includes flight or train. The relationship is a complete and disjoint in this use case, which means the kind of booking in this use case is either flight or train. I did not put any of the verbiage such as “several of these” or “none of these” since the rule is totally complete and disjoint.

2. A car type is private car, a car use in car-hailing or in food delivery, or none of these.

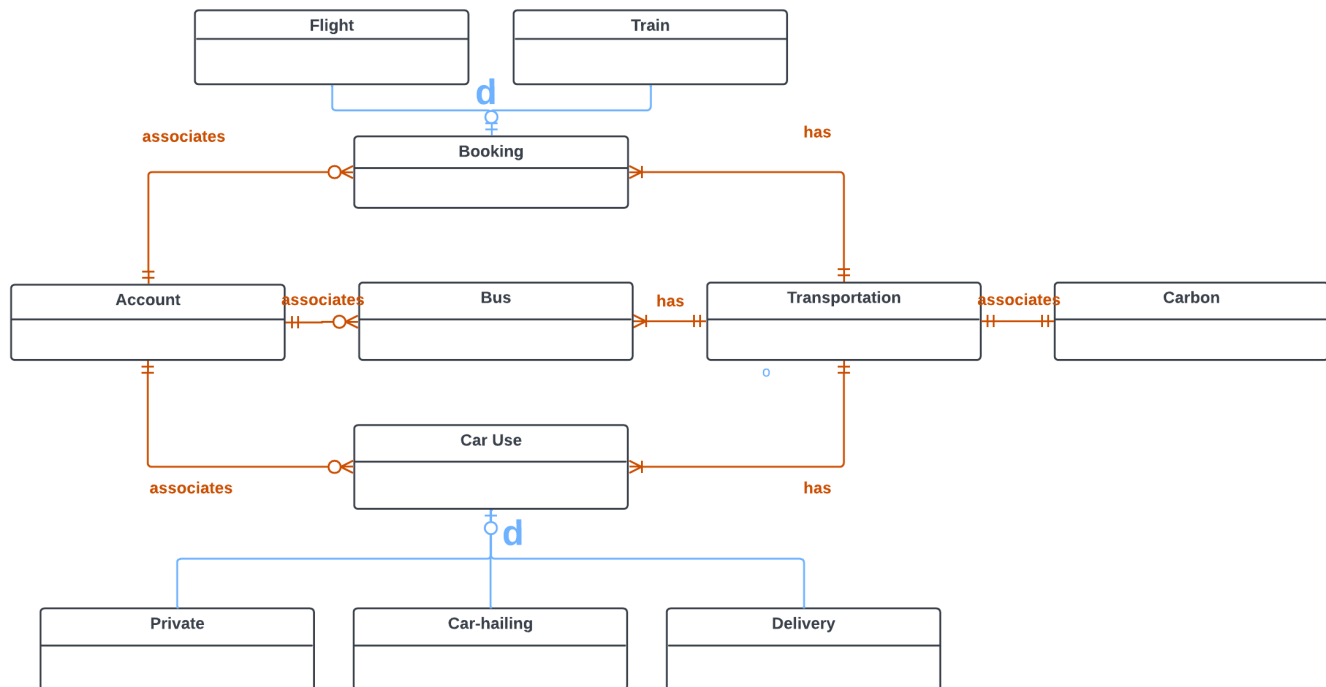
The relationship allows my database to have car use either for private use, or car-hailing, or food delivery. The food delivery indicates that the person order take-out food, which is also an indirect way of car using because food is delivered by car driving. The above three ways mentioned are what I am concerning about in my case. Therefore, I do not use the phrase “several of these”. I phrase “or none of these” because a person can also use a car in other ways which I do not focus here. For instance, a person might rent a car when the person takes a journey.

Here is a summary of the structured database rules and specialization-generalization rules mentioned before:

1. Each account may be associated with many bookings and each booking is associated with an account.
2. Each booking is made from a means of transport, and each kinds of transport has one to many bookings.
3. Each account may be associated with many records of taking a bus and each bus record is associated with an account.
4. Each account may be associated with many car using records, and each car using records is associated with an account.
5. Each car use record results from one type of car driving, and each car driving type has one to many car using records.
6. One kind of means of transport has one level of carbon emission and one level of carbon emission is associated with one kind of transportation.
7. A booking is a flight booking or a train booking.
8. A car type is private car, a car use in car-hailing or in food delivery, or none of these.

Conceptual Entity-Relationship Diagram

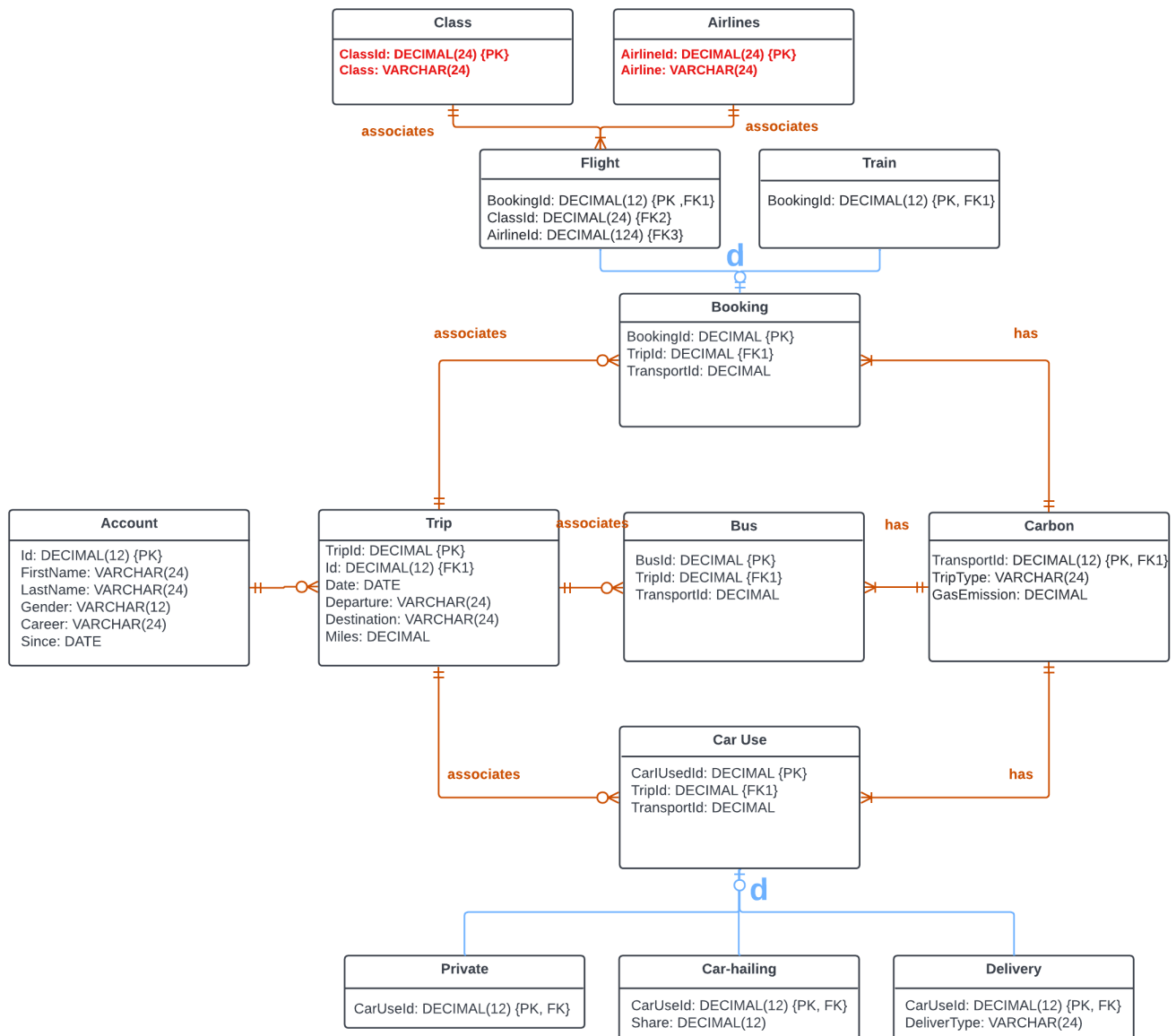
Here follows the ERD I came up with for the structural database rules using Crow's Foot symbol:



The Account entity is associated with three entities: bookings, bus record and car using record. At the same time, booking is related with means of transport, from flight to train, and car using records is relative with type of car driving, involving private car, car-hailing driving, and food delivery car driving. The participation and plurality constraints indicate how entity relates to each other in the structural database rules. For example, the circle near inside the end of the line indicates that the relationship is optional and the single bar inside the end of the line indicates that the relationship is mandatory. Crow's Foot uses a single bar to refer to singular relationship and use a fork to indicate plural relationship at the end of the line outside. In this ERD, for instance, the symbols between account and the bus record indicates that each account may be associated with many records of bus and each bus record is associated with an account, which reflects the business rules.

Full DBMS Physical ERD

Here is my full DBMS physical ERD with attributes included:



ERD with Specialization-Generalization

There are new entities for booking and car use. A booking is a train booking or a flight booking, which indicates that the relationship is mandatory and disjoint. I define this kind of relationship because flight and train are the only two ways of transportation that requires ticket in my application. Similarly, car use involves private car, car-hailing, and delivery car use, or none of these, which indicates that the relationship is mandatory and disjoint.

Relationship Classification and Associative Mapping

The associative relationships in my firstly draft conceptual ERD are *Account/Booking*, *Account/Bus*, *Account/Car Use*,

The *Account/Booking* relationship is 1:M. An account can make several bookings, but each booking is tied to one account.

The *Account/Bus* relationship is 1:M. An account can take bus for multiple times, but each use of bus is tied to one account.

The *Account/Car Use* relationship is 1:M. An account can use car for multiple times, but each car use is tied to one account.

Relationship between *Carbon* and *Booking/Bus/Car Use* is all 1:1. One kind of use of transport mentioned corresponds to one level of gas emission level.

The DBMS physical ERD created from the conceptual ERD is included in the above graph.

Id is the primary key for account entity which is also the foreign key for *Booking*, *Bus* and *Car Use* entity. Records of ticket transport, bus taking, and car use can be traced through account identification, which makes calculation of individual carbon emission possible.

TransportId, which is defined as the primary key for *Transportation* and as the foreign key for *Booking*, *Bus* and *Car Use*, refers to different kinds of transportation in this case, including flight or train, bus, and car. It is also the primary key for *Carbon* because it can be used to list out the unit gas emission for different transportation.

The DECIMAL and VARCHAR allow me to fill enough data in the database with individual data in proper length. Date data is also defined to fill in with date form, which allows me to make further date calculation and summary.

Specialization-Generalization Mapping

I had two specialization-generalization relationships in my conceptual ERD, for *Booking* and *Car Use*. The additional entities under *Booking* are *Flight* and *Train*, each of which has a primary and foreign key of *TransportId* which reference the primary key of *Transportation*. The additional entities under *Car Use* are *Private*, *Ca-hailing* and *Delivery* each of which has a primary and foreign key of *TransportId* which reference the primary key of *Car Type*. There is a new field for car-hailing named *Share* which is used to indicate the total number of people taking the car at a time. In this case, the level of carbon emission can be divided by the total number of people taking the car when calculating the level of carbon emission for each account.

Attributes explanation

Table	Attribute	Datatype	Reasoning
Account	Id	DECIMAL(12)	Each person has an identification which will be created when the individual creates an account in <i>CarbonFootprint</i> application. I allow identification number to be up to 12 decimals.
	FirstName	VARCHAR(24)	This is the first name of the account user, up to 24 characters is enough to store encrypted text.

	LastName	VARCHAR(24)	This is the last name of the account user, up to 24 characters is enough to store encrypted text.
	Gender	VARCHAR(12)	This is the gender of the account user, up to 12 characters is enough to store the gender information.
	Career	VARCHAR(24)	This is the career description of the account user, up to 24 characters is enough to store the occupation information.
	Since	DATE	The specifies the date when the new account for individual is created. Date type is identified to calculate the length of existence of the account.
Booking	Date	DATE	A booking is made on a specific date.
	Departure	VARCHAR(24)	When a person makes a flight or train booking, information of the departure location is required which will also be stored in the application. I allow for 24 characters to store the departure name.
	Destination	VARCHAR(24)	When a person makes a flight or train booking, information of the destination location is required which also will be stored in the application. I allow for 24 characters to store the destination name.
	Miles	DECIMAL	When booking a flight or train, miles between the departure and the destination will be informed in the booking details.
Bus	BusDate	DATE	A person takes a bus on a specific date.
	BusFrom	VARCHAR(24)	When a person takes a bus, there will be a starting point/starting bus station which needed to be filled in by user manually. 24 characters are long enough to store the starting station information.
	BusTo	VARCHAR(24)	When a person takes a bus, there will be an end point/ending bus station which needed to be filled in by user manually. 24 characters are long enough to store the terminal information.

CarUse	CarDate	DATE	This indicates the date when the account user has a car consumption.
	Start	VARCHAR(24)	When there is a car consumption, there will be a starting point where the car starts off. I allow 24 characters to store the information of the departure.
	WhereTo	VARCHAR(24)	When there is a car consumption, there will be an ending point where the car arrives. I allow 24 characters to store the information of the departure.
	CarMiles	DECIMAL	This indicates the distance between the starting point and the ending point for a specific car route. No upper limit is set for the miles.
Transportation	TransportId	CHAR(1)	This specifies the type of the transportation focused by CarbonFootprint when a person make a booking, use a car, or take a bus. Flight, train, bus, and car use are represented by 1, 2, 3 and 4 respectively. This attribute is the subtype discriminator indicating which it is.
	TransportType	VARCHAR(24)	There are four types of transportations focused by <i>CarbonFootprint</i> application, as mentioned before, including flight, train, bus, and car. 24 characters are enough to store the description of transportation types.
Carbon	GasEmission	DECIMAL	Each transportation corresponds to a specific level of carbon emission. No upper limit is set for the level of gas emission.
Car-hailing	Share	DECIMAL	When using a car-hailing service, it is possible that a person will share the trip with other people, meaning that the carbon level for this individual will be the carbon emission of this trip divided by the total number of people.

CODE FOR CREATING TABLES IN SQL SERVER

```
DROP TABLE Account;
DROP TABLE Trip;
```

```

DROP TABLE Booking;
DROP TABLE Flight;
DROP TABLE Class;
DROP TABLE Airlines;
DROP TABLE Train;
DROP TABLE Bus;
DROP TABLE CarUse;
DROP TABLE Private;
DROP TABLE Car_hailing;
DROP TABLE Delivery;
DROP TABLE Carbon;

DROP SEQUENCE account_seq;
DROP SEQUENCE trip_seq;
DROP SEQUENCE booking_seq;
DROP SEQUENCE class_seq;
DROP SEQUENCE airline_seq;
DROP SEQUENCE bus_seq;
DROP SEQUENCE car_seq;

CREATE TABLE Account (
    Id DECIMAL(12) NOT NULL PRIMARY KEY,
    FirstName VARCHAR(24) NOT NULL,
    LastName VARCHAR(24) NOT NULL,
    Gender VARCHAR(12) NOT NULL,
    Career VARCHAR(24),
    Since DATE);

CREATE TABLE Trip (
    TripId DECIMAL NOT NULL PRIMARY KEY,
    Id DECIMAL(12) FOREIGN KEY REFERENCES Account (Id),
    Date DATE,
    Departure VARCHAR(24),
    Destination VARCHAR(24),
    Miles DECIMAL);

CREATE TABLE Booking (
    BookingId DECIMAL NOT NULL PRIMARY KEY,
    TripId DECIMAL FOREIGN KEY REFERENCES Trip (TripId),
    TransportId DECIMAL);

CREATE TABLE Flight (
    BookingId DECIMAL NOT NULL PRIMARY KEY,
    ClassId DECIMAL(24),
    AirlineId DECIMAL(24));

CREATE TABLE Class (
    ClassId DECIMAL(24) PRIMARY KEY,
    Classname VARCHAR(24));

CREATE TABLE Airlines (
    AirlineId DECIMAL(24) PRIMARY KEY,
    Airline VARCHAR(24));

CREATE TABLE Train (
    BookingId DECIMAL NOT NULL PRIMARY KEY);

CREATE TABLE Bus (
    BusId DECIMAL NOT NULL PRIMARY KEY,
    TripId DECIMAL FOREIGN KEY REFERENCES Trip (TripId),
    TransportId DECIMAL);

```

```
CREATE TABLE CarUse (  
    CarUseId DECIMAL NOT NULL PRIMARY KEY,  
    TripId DECIMAL FOREIGN KEY REFERENCES Trip (TripId),  
    TransportId DECIMAL);
```

```
CREATE TABLE Private (  
    CarUseId DECIMAL NOT NULL PRIMARY KEY);
```

```
CREATE TABLE Car_hailing (  
    CarUseId DECIMAL NOT NULL PRIMARY KEY,  
    Share DECIMAL(12));
```

```
CREATE TABLE Delivery (  
    CarUseId DECIMAL NOT NULL PRIMARY KEY,  
    DeliverType VARCHAR(24));
```

```
CREATE TABLE Carbon (  
    TripTypeId DECIMAL(12) PRIMARY KEY,  
    TransportType VARCHAR(24),  
    GasEmission DECIMAL);
```

I include DROP TABLE command at the top to make the scripts overall re-runnable. From now on, six tables are created, and more data is waiting to be filled in.

Stored Procedure Execution and Explanation

1. Account Signup/installation Use Case

The first use case for *CarbonFootprint* is the account signup use case listed below.

1. The person visits the *CarbonFootprint's* website or app store and installs the application.
2. For new user, the person is asked to create a new account. Basic information is required to be fulfilled and it will be added to the database storing user's information.
3. When either an application account or a website account is setup, the person can login the account by both website and application.

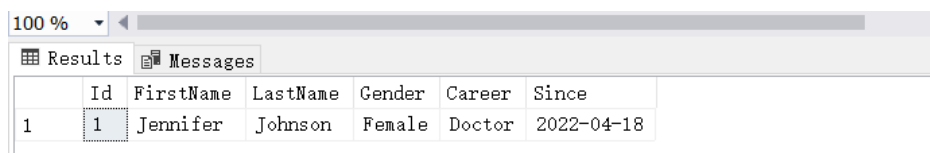
For this use case, I will implement a transaction that creates an account, using SQL Server . Here is a screenshot of my stored procedure definition.

```
CREATE PROCEDURE add_account
    @FirstName VARCHAR(24),
    @LastName VARCHAR(24),
    @Gender VARCHAR(12),
    @Career VARCHAR(24),
    @Date DATE
AS
BEGIN
    INSERT INTO Account (Id, FirstName, LastName, Gender, Career, Since, CarbonTotal)
    VALUES (NEXT VALUE FOR account_seq, @FirstName, @LastName, @Gender, @Career, @Date, 0);
END;
GO
```

I name the stored procedure “add_account”, and provide related parameters related to the Account table. ‘Since’ is the current date when the account was created, so it is unnecessary to provide a parameter for the date. Instead, a GETDATE() function is used. Inside the stored procedure, there will be new account created and be inserted into the Account table.

I add a person named Jennifer Johnson with other fictional but proper information. Here is a screenshot of my stored procedure execution.

```
BEGIN TRANSACTION add_account;
EXECUTE add_account 'Jennifer', 'Johnson', 'Female', 'Doctor', '12/20/2020';
COMMIT TRANSACTION;
SELECT * FROM Account;
```



Results		Messages				
	Id	FirstName	LastName	Gender	Career	Since
1	1	Jennifer	Johnson	Female	Doctor	2022-04-18

2. Automatic Flight Track Use Case

The second usage of the database is when a flight is booked on the website or application and automatically recorded in the *CarbonFootprint* application.

Here are the use case details:

1. A person books a flight on the official flight website or in mobile flight application.
2. The application in phone or the plugin in website detects the booked flight tickets and records related information in the database.

For this use case, I will implement a transaction that creates a flight, using SQL Server .
Here is a screenshot of my stored procedure definition.

```
CREATE PROCEDURE add_flight
    @FirstName VARCHAR(24),
    @LastName VARCHAR(24),
    @Date DATE,
    @Departure VARCHAR(24),
    @Destination VARCHAR(24),
    @Miles DECIMAL,
    @Class VARCHAR(24),
    @Airline VARCHAR(24)
AS
BEGIN
    DECLARE @v_Id DECIMAL(12);
    DECLARE @trip_seq INT = NEXT VALUE FOR trip_seq;
    DECLARE @booking_seq INT = NEXT VALUE FOR booking_seq;

    SELECT @v_Id = Id
    FROM Account
    WHERE FirstName = @FirstName and LastName = @LastName;

    INSERT INTO Trip (TripId, Id, Date, Departure, Destination, Miles)
    VALUES (@trip_seq, @v_Id, @Date, @Departure, @Destination, @Miles);

    INSERT INTO Booking (BookingId, TripId, TransportId)
    VALUES (@booking_seq, @trip_seq, 1);

    IF NOT EXISTS (SELECT * FROM Class
                   WHERE Classname = @Class)
    BEGIN
        DECLARE @class_seq INT = NEXT VALUE FOR class_seq;
        INSERT INTO Class (ClassId, Classname)
        VALUES (@class_seq, @Class)
    END

    IF NOT EXISTS (SELECT * FROM Airlines
                   WHERE Airline = @Airline)
    BEGIN
        DECLARE @airline_seq INT = NEXT VALUE FOR airline_seq;
        INSERT INTO Airlines (AirlineId, Airline)
        VALUES (@airline_seq, @Airline)
    END

    DECLARE @class_seq_v DECIMAL(12) = (SELECT ClassId FROM Class WHERE Classname = @Class);
    DECLARE @airline_seq_v DECIMAL(12) = (SELECT AirlineId FROM Airlines WHERE Airline =
@Airline)
    INSERT INTO Flight (BookingId, ClassId, AirlineId)
    VALUES (@booking_seq, @class_seq_v, @airline_seq_v)

END;
```

I name the stored procedure “add_flight”, and provide related parameters related to the Booking, Flight and Trip table. Inside the stored procedure, there will be new trip record created and be inserted into the Trip table and new flight booking records being added to the Booking table. New information about flight class, airline will also be created.

I add a flight record for the person named Jennifer Johnson I created before. Here is a screenshot of my stored procedure execution.

```
GO
BEGIN TRANSACTION add_flight;
EXECUTE add_flight 'Jennifer', 'Johnson', '01/20/2021', 'Boston', 'Chicago', 867, 'economy',
'JetBlue';
COMMIT TRANSACTION;
```

3. Automatic Car Track Use Case

The third usage example is a car driving. Take private car driving as an example here.

Here are the use case details:

1. A person drives a car and use navigation to record the route.
2. A person uses car-hailing services.
3. A person orders a takeaway on food delivery platforms or applications.
4. The *CarbonFootprint* application gets access to the navigation, car-hailing application as well as delivery applications and automatically records related information in the database.

For this use case, I will implement a transaction that creates a private car use by SQL Server . Here is a screenshot of my stored procedure definition.

```
CREATE PROCEDURE add_bus
    @FirstName VARCHAR(24),
    @LastName VARCHAR(24),
    @Date DATE,
    @Departure VARCHAR(24),
    @Destination VARCHAR(24),
    @Miles DECIMAL
AS
BEGIN
    DECLARE @v_Id DECIMAL(12);
    DECLARE @trip_seq INT = NEXT VALUE FOR trip_seq;
    DECLARE @bus_seq INT = NEXT VALUE FOR bus_seq;

    SELECT @v_Id = Id
    FROM Account
    WHERE FirstName = @FirstName and LastName = @LastName;

    INSERT INTO Trip (TripId, Id, Date, Departure, Destination, Miles)
    VALUES (@trip_seq, @v_Id, @Date, @Departure, @Destination, @Miles);

    INSERT INTO Bus (BusId, TripId, TransportId)
    VALUES (@bus_seq, @trip_seq, 2);

END;
```

I name the stored procedure “add_private_car”, and provide related parameters related to the private car using. Inside the stored procedure, there will be new trip record created and be inserted into the Trip table and new private car use records being added to the CarUse table.

I add a private car use record for the person named Jennifer Johnson as before. Here is a screenshot of my stored procedure execution.

```
GO
```

```
BEGIN TRANSACTION add_bus;
EXECUTE add_bus 'John', 'Ferguson', '01/19/2021', 'Chicago-bus1', 'Chicago-bus10', 13;
COMMIT TRANSACTION;
```

Question Identification and Explanations

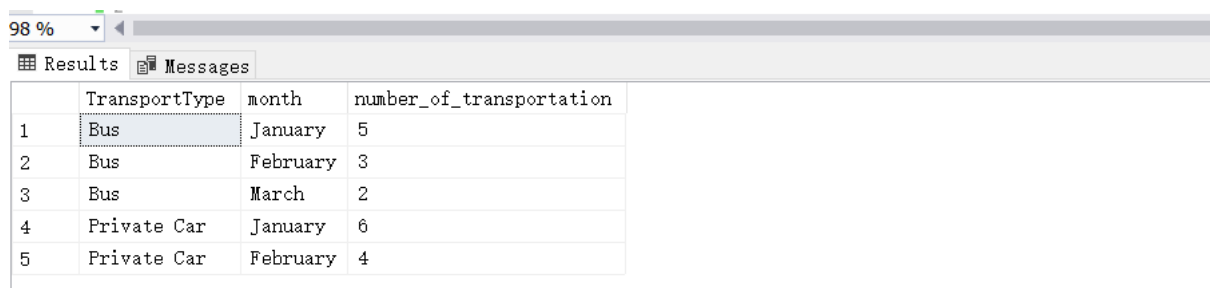
Monthly Carbon Emission

Here is a question useful to the important part of CarbonFootprint application: How people choose between private car driving and taking a bus, shown in different months?

This question gives the frequency that people choose to drive their own car or take a bus when going around. There is a possible situation that people can not drive a car but have to book the flight, but in most cases, people can choose to take a bus instead of driving their own car, which releases much higher level of carbon dioxide. This question helps me to further analyze the user's preference between driving private car and taking a bus in different month.

Here is a screenshot of the query I use.

```
SELECT TransportType,
       CASE MONTH(Date) WHEN 1 THEN 'January' WHEN 2 THEN 'February' ELSE 'March' END
month,
       COUNT (t.TripId) number_of_transportation
FROM Trip t JOIN
  (SELECT c. TransportId, TripId
   FROM Carbon c
   INNER JOIN Bus bu ON c.TransportId = bu.TransportId
   UNION ALL
   SELECT c.TransportId, TripId
   FROM Carbon c
   INNER JOIN CarUse cu ON c.TransportId = cu.TransportId) tb
ON t. TripId = tb. TripId
LEFT JOIN Account a ON a. Id = t. Id
LEFT JOIN Carbon c ON tb.Transportid=c.Transportid
WHERE YEAR (Date) = '2021'
GROUP BY TransportType, MONTH(Date)
ORDER BY TransportType;
```



	TransportType	month	number_of_transportation
1	Bus	January	5
2	Bus	February	3
3	Bus	March	2
4	Private Car	January	6
5	Private Car	February	4

Flight Number

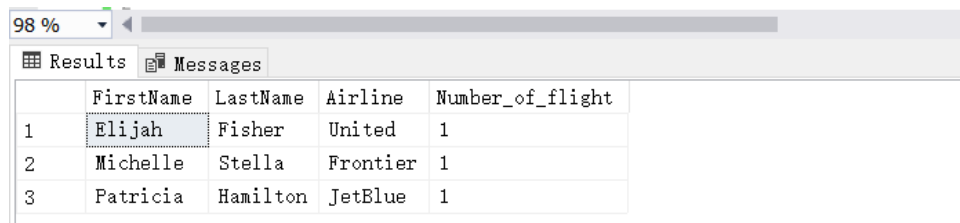
Here is a question useful to the CarbonFootprint application: For all the users, how many times do they take a flight in a specific period, and do they have preference on specific airline company? Here I choose to figure out the frequency of flight booking for individual in February in 2021.

Traveling by flight releases a high level of carbon dioxide, which makes the question useful for CarbonFootprint application to get a sense of at how frequently people travel by flight in a specific period. Also, considering that CarbonFootprint is also faced by users, each account user could see how

frequent they choose to travel by flight. Also, it is useful especially when there is a peak season when people are more likely to choose to book flights. The number of flights is group by different airline company, which is also useful to analyze whether there is a different preference among individual on airlines.

Here is a screenshot of the query I use.

```
SELECT FirstName, LastName, Airline, COUNT(a.Id) AS Number_of_flight
FROM Account a
LEFT JOIN Trip t ON a.Id = t.Id
LEFT JOIN Booking b ON t.TripId = b.BookingId
LEFT JOIN Flight f ON f.BookingId = b.BookingId
LEFT JOIN Airlines air ON air.AirlineId = f.AirlineId
WHERE Airline IS NOT NULL
      AND Date BETWEEN '02/01/2021' AND '03/01/2021'
GROUP BY FirstName, LastName, Airline
ORDER BY Number_of_flight DESC;
```



	FirstName	LastName	Airline	Number_of_flight
1	Elijah	Fisher	United	1
2	Michelle	Stella	Frontier	1
3	Patricia	Hamilton	JetBlue	1

Individual Carbon Emission

Here is a question useful to the important part of CarbonFootprint application: How much carbon dioxide is released by each individual, which is the significant data that this program focuses on.

This question gives the value of total level of carbon released by each account, which will also present to users. I also use order by to rank the total carbon level, which specifies the users who have most carbon emissions. This helps me to further analyze the user's transportation preference since I can divide the total level of carbon emission for individual into different kinds of transportation in the next steps to find out which kind of transportation contributes to the most part of carbon emission.

Here is a screenshot of the query I use.

```
CREATE OR ALTER VIEW individual_total_carbon
AS
SELECT FirstName, LastName, SUM(Miles * GasEmission) carbon_emission
FROM Trip t JOIN
  (SELECT c.TransportId, TripId
   FROM Carbon c
   INNER JOIN Booking b ON c.TransportId = b.BookingId
   UNION ALL
   SELECT c.TransportId, TripId
   FROM Carbon c
   INNER JOIN Bus bu ON c.TransportId = bu.TransportId
   UNION ALL
   SELECT c.TransportId, TripId
   FROM Carbon c
   INNER JOIN CarUse cu ON c.TransportId = cu.TransportId) tb
ON t.TripId = tb.TripId
LEFT JOIN Account a ON a.Id = t.Id
```

```
LEFT JOIN Carbon c ON tb.TransportId = c.TransportId
GROUP BY FirstName, LastName;
```

```
SELECT * FROM individual_total_carbon
ORDER BY carbon_emission DESC;
```

98 %

Results				Messages
	FirstName	LastName	carbon_emission	
1	Michelle	Stella	824250	
2	Elijah	Fisher	369363	
3	Paula	Reyes	339672	
4	Natalie	Robinson	278262	
5	Jennifer	Johnson	267204	
6	Patricia	Hamilton	229935	
7	John	Ferguson	86790	
8	Jasmine	James	2625	
9	Samuel	Patterson	1995	

Index Identification and Creations

As far as primary keys which are already indexed, here is the list.

Account.Id
Trip.TripId
Booking.BookingId
Flight.BookingId
Train.BookingId
Class.ClassId
Airlines.AirlineId
Bus.BusId
CarUse.CarUseId
Carbon.TransportId
Private.CarUseId
Car_hailing.CarUseId
Delivery.CarUseId

As far as foreign keys, I know all of them need an index. Below is a table identifying each foreign key column, whether or not the index should be unique or not, and why.

Column	Unique?	Description
Trip.Id	Not unique	The foreign key in Trip referencing Account is not unique because there can be many trips to an account.
Booking.TripId	Not unique	The foreign key in Booking referencing Trip is not unique because there can be many bookings to one trip.
Flight.BookingId	Unique	The foreign key in Flight referencing Booking is unique because one booking is only corresponds to one flight.
Flight.ClassId	Not unique	The foreign key in Flight referencing Class is not unique because there can be many flights to one class.
Flight.AirlineId	Not unique	The foreign key in Flight referencing Airline is not unique because there can be many flights in one airline.
Train.BookingId	Unique	The foreign key in Train referencing Booking is unique because there is only one train booking correspond to one booking.

Bus.TripId	Not unique	The foreign key in Bus referencing Trip is not unique because there can be many bus takings in one trip.
CarUse.TripId	Not unique	The foreign key in CarUse referencing Trip is not unique because there can be many car-uses to one trip.
Private.CarUseId	Unique	The foreign key in Private referencing CarUseId is unique because one car use id refers to private car use.
Car_hailing.CarUseId	Unique	The foreign key in Car_hailing referencing CarUseId is unique because one car use id refers to car-hailing.
Delivery.CarUseId	Unique	The foreign key in Delivery referencing CarUseId is unique because one car use id refers to delivery car use.
Carbon.TransportId	Unique	The foreign key in Carbon referencing TransportId is unique because there is one level of carbon for each kind of transportation.

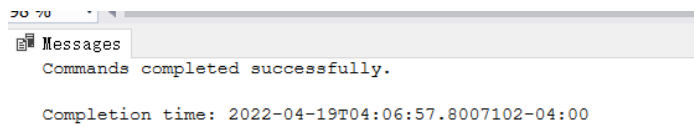
I firstly choose individual_total_carbon.carbon_emission which is the view I created in the last query to index because it is reasonable that there will be many queries that limit by the total value of carbon emission. This would be a non-unique index because many people can have the same level of carbon emission.

It is also reasonable that the date of trip will be a limiting column in queries because people have the habit to see whether they spent more on their commute regularly. So I choose Trip,Date to index. Carbon emission calculated in month can be used to compare among different seasons in a year. This would be a non-unique index because many trips could happen on the same day.

Lastly, I select Account.Career to index because there is possible to have queries limiting on career of users to see whether there are occupations which have more business trips by flights. It is a non-unique index because there can be multiple accounts for one specific career.

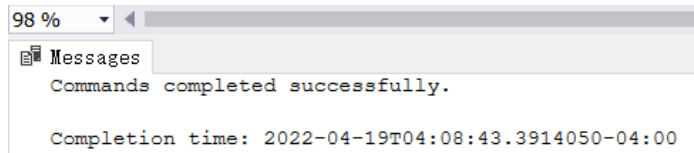
Here is the screenshot showing the creation of a foreign key index for CarbonFootprint, for the foreign key between Trip and Date.

```
CREATE INDEX DateIndex
ON Trip(Date);
```

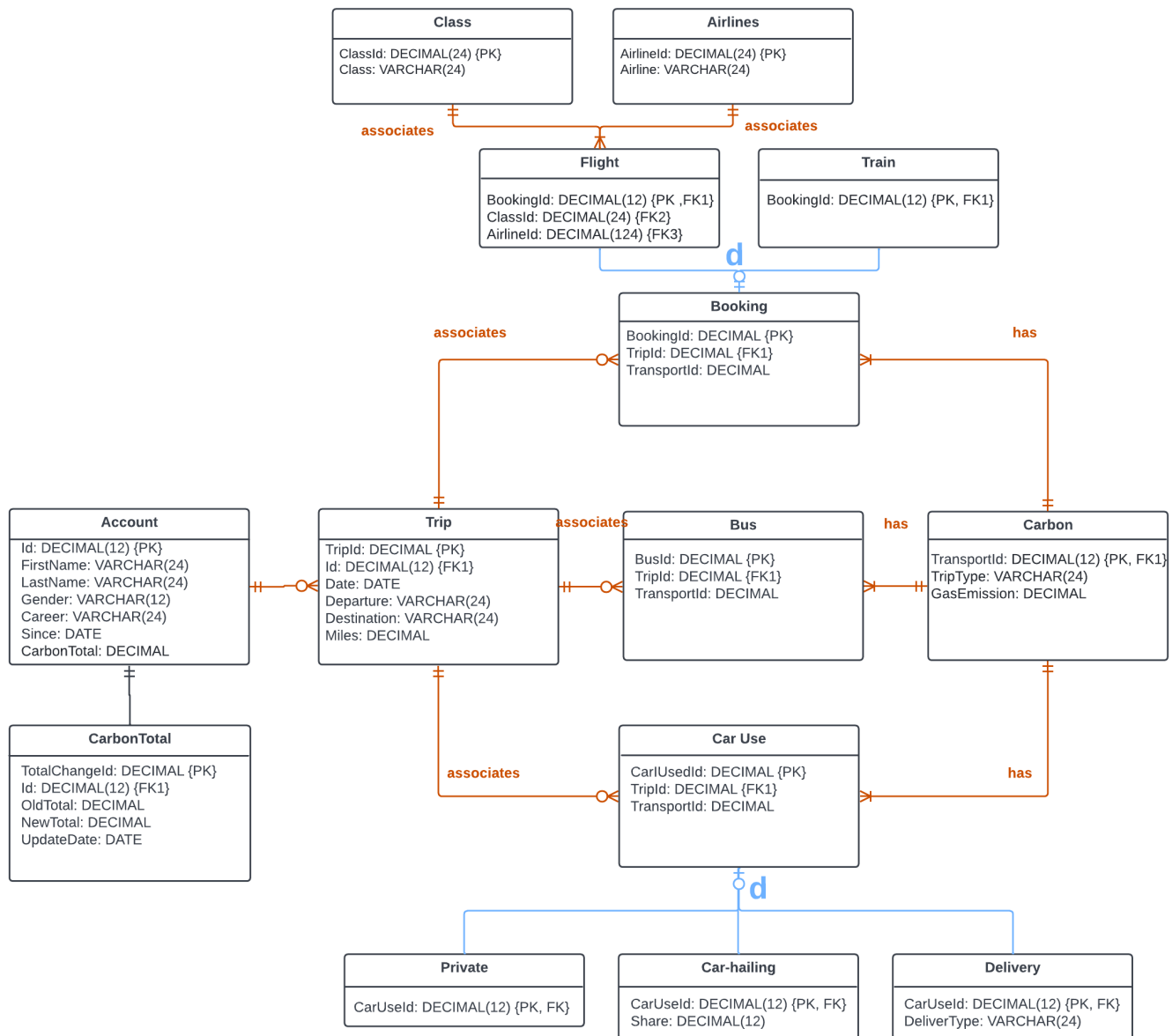
Here is the screenshot showing the creation of a foreign key index for CarbonFootprint, for the foreign key between Trip and Date.

```
CREATE INDEX CareerIndex  
ON Account(Career);
```



History Table Demonstration

For *CarbonFootprint* application, the entity that will benefit from a historical record is the total level of carbon of individual account. Such a history can provide users a glance at their carbon emission, which will help them to see how they choose on transportation when they go around. If they saw that the change of the carbon level is drastic, which indicating that they use flight or car more at that period, they might pay more attention to choosing more environmentally friendly transportation in the future. Each Account has many total carbon value changes, and each total carbon level is for an account. My updated DBMS physical ERD is below:



The CarbonTotal entity is present and linked to Account. Below are the attributes I added and why.

Attribute	Description
TotalChangeId	This is the primary key of the history table. It is a DECIMAL to allow for values.

Id	This is the foreign key to Account table, a reference to the account that had the change in the carbon total level.
OldTotal	This is the carbon level before change. The datatype mirrors the carbon total value datatype in Account table.
NewTotal	This is the carbon level after change. The datatype mirrors the carbon total value datatype in Account table.
UpdateDate	This is the date the carbon total level change occurred, with a DATE datatype.

Here is a screenshot of my table and sequence creation, which has all the same attributes and datatypes as indicated in the DBMS physical ERD.

```
CREATE TABLE CarbonTotal (
    TotalChangeId DECIMAL NOT NULL PRIMARY KEY,
    Id DECIMAL(12) NOT NULL FOREIGN KEY REFERENCES Account(Id),
    OldTotal DECIMAL NOT NULL,
    NewTotal DECIMAL NOT NULL,
    UpdateDate DATE NOT NULL
);
```

```
CREATE SEQUENCE TotalChangeSeq START WITH 1;
```

Here is a screenshot of my trigger creation which will maintain the CarbonTotal table.

```
CREATE TRIGGER CarbonTotalTrigger
ON Account
AFTER UPDATE
AS
BEGIN

    DECLARE @OldTotal DECIMAL = (SELECT CarbonTotal FROM DELETED);
    DECLARE @NewTotal DECIMAL = (SELECT CarbonTotal FROM INSERTED);

    IF (@OldTotal <> @NewTotal)
        INSERT INTO CarbonTotal (TotalChangeId, Id, OldTotal, NewTotal, UpdateDate)
        VALUES (NEXT VALUE FOR TotalChangeSeq, (SELECT Id FROM INSERTED),
                @OldTotal, @NewTotal, GETDATE());

END;
```

I explain it here line by line.

CODE	DESCRIPTION
CREATE TRIGGER CarbonTotalTrigger ON Account AFTER UPDATE	This starts the definition of the trigger and names it "CarbonTotalTrigger". The trigger is linked to the Account table, and is executed after any updated to that table.
AS BEGIN	This is part of the syntax starting the trigger block.

<pre> DECLARE @OldTotal DECIMAL = (SELECT CarbonTotal FROM DELETED); DECLARE @NewTotal DECIMAL = (SELECT CarbonTotal FROM INSERTED); IF (@OldTotal <> @NewTotal) </pre>	This saves the old and new balances by referencing the DELETED and INSERTED pseudo tables, respectively.
<pre> INSERT INTO CarbonTotal (TotalChangeId, Id, OldTotal, NewTotal, UpdateDate) VALUES (NEXT VALUE FOR TotalChangeSeq, (SELECT Id FROM INSERTED), @OldTotal, @NewTotal, GETDATE()); </pre>	This inserts the record into the CarbonTotal table. The primary key is set by using the TotalChangeSeq sequence. The old and new balances are used from the variables. The Id is obtained from the INSERTED pseudo table. The date of the change is obtained by using the built-in GETDATE function.
<pre> END; </pre>	This ends the trigger definition.

All new account's total carbon level starting with 0.

```

UPDATE Account
SET CarbonTotal = (SELECT SUM(Miles * GasEmission) carbon_emission
FROM Trip t JOIN
    (SELECT c.TransportId, TripId
    FROM Carbon c
    INNER JOIN Booking b ON c.TransportId = b.TransportId
    UNION ALL
    SELECT c.TransportId, TripId
    FROM Carbon c
    INNER JOIN Bus bu ON c.TransportId = bu.TransportId
    UNION ALL
    SELECT c.TransportId, TripId
    FROM Carbon c
    INNER JOIN CarUse cu ON c.TransportId = cu.TransportId)
    ON t.TripId = tb.TripId
LEFT JOIN Account a ON a.Id = t.Id
LEFT JOIN Carbon c ON tb.TransportId = c.TransportId
WHERE FirstName = 'Jennifer' AND LastName = 'Johnson')
SELECT * FROM CarbonTotal;

```

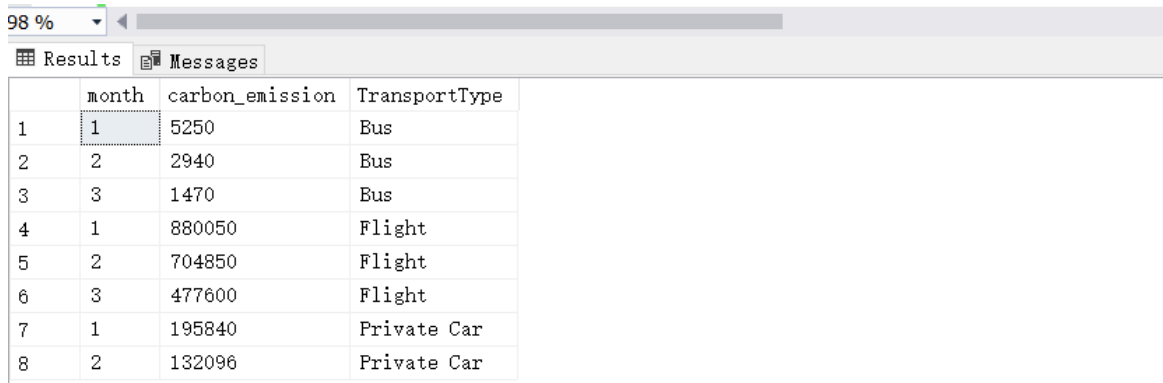
My update is based on new records added in Trip table. When there is a new trip created, the total value of carbon for that person increases, which leads to the update in the CarbonTotal table. At the same time, the total level of carbon emission will be calculated based on the transportation type and the corresponding carbon emission level.

Data Visualizations

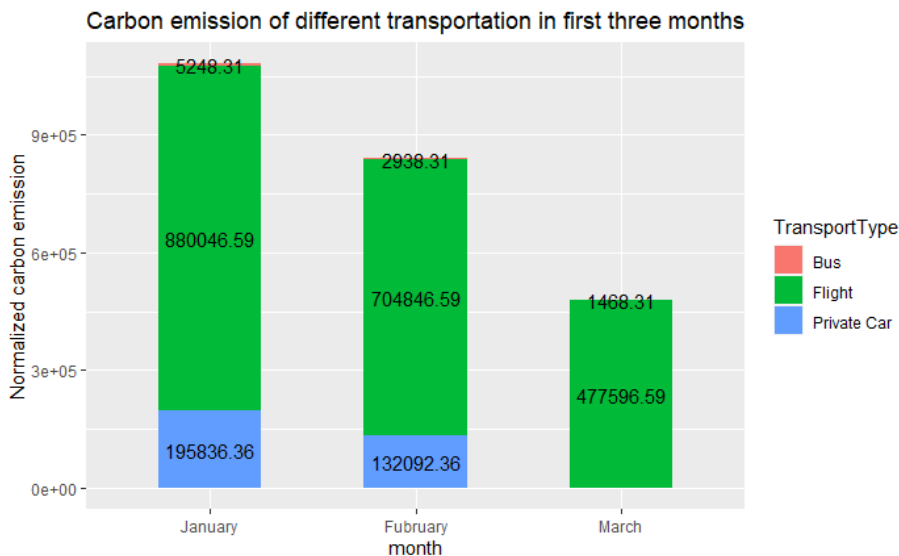
1. Carbon emission of different transportation

The more that is known about how users choose on the different transportation, the more effectively that people will raise their awareness of choosing more environmentally friendly way of transportation. So, it is useful to look at the monthly carbon emission level of different transportation in each month. The comparison between different kinds of transportation can be made. To list out the result, I develop and execute a query that summarize the monthly total carbon emission of different transportation. This query and results are shown below.

```
SELECT MONTH(Date) month, SUM(Miles * GasEmission) carbon_emission, c.TransportType
INTO graph2
FROM Trip t JOIN
    (SELECT c.TransportId, TripId, c.TransportType
    FROM Carbon c
    INNER JOIN Booking b ON c.TransportId = b.TransportId
    UNION ALL
    SELECT c.TransportId, TripId, c.TransportType
    FROM Carbon c
    INNER JOIN Bus bu ON c.TransportId = bu.TransportId
    UNION ALL
    SELECT c.TransportId, TripId, c.TransportType
    FROM Carbon c
    INNER JOIN CarUse cu ON c.TransportId = cu.TransportId) tb
    ON t.TripId = tb.TripId
LEFT JOIN Account a ON a.Id = t.Id
LEFT JOIN Carbon c ON tb.TransportId = c.TransportId
GROUP BY MONTH(Date), c.TransportType;
SELECT * FROM graph2;
```



	month	carbon_emission	TransportType
1	1	5250	Bus
2	2	2940	Bus
3	3	1470	Bus
4	1	880050	Flight
5	2	704850	Flight
6	3	477600	Flight
7	1	195840	Private Car
8	2	132096	Private Car



The data was normalized, and we can make a conclusion that it seems people release carbon mostly on flight travel on their trips, which can be told from the main green part of the stack bar plot. Also, the total carbon level decreases for the given dataset from January to March. Flight releases more carbon overall, which might result from the long miles for flight. The comparison between bus and car can be further investigated.

2. Individual monthly carbon emission

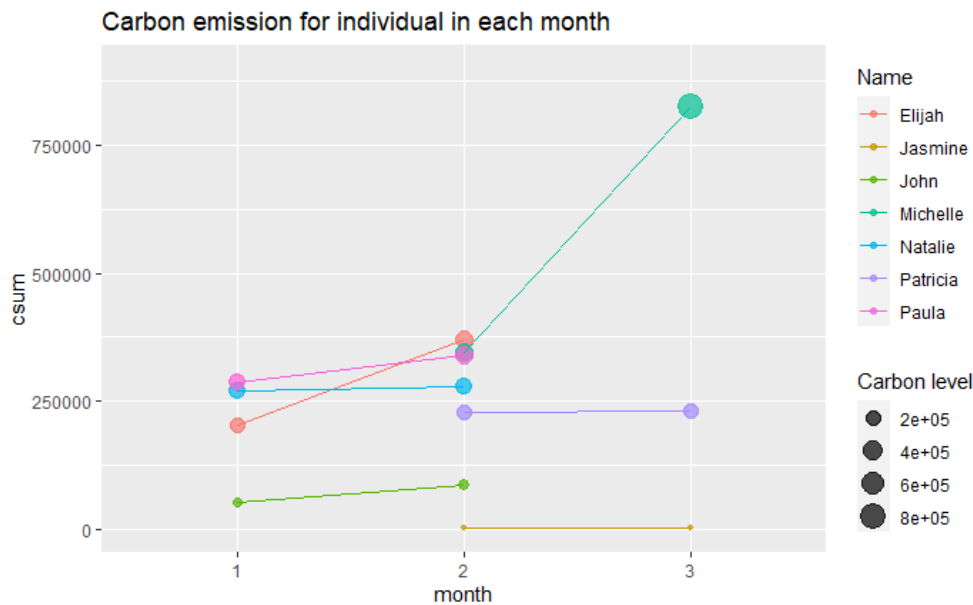
For individual, users also care about their personal total carbon emission. The total carbon emission level recorded starts when their account was created. Each account can have a look at the cumulative carbon level, which is the key information stored in the history table. To list out the result, I develop and execute a query that summarize the cumulative carbon emission of for every account in the dataset. This query and results are shown below.

```
SELECT FirstName, LastName, MONTH(Date) month, SUM(Miles * GasEmission) carbon_emission
INTO graph3
FROM Trip t JOIN
  (SELECT c.TransportId, TripId
   FROM Carbon c
   INNER JOIN Booking b ON c.TransportId = b.TransportId
  UNION ALL
   SELECT c.TransportId, TripId
   FROM Carbon c
   INNER JOIN Bus bu ON c.TransportId = bu.TransportId
  UNION ALL
   SELECT c.TransportId, TripId
   FROM Carbon c
   INNER JOIN CarUse cu ON c.TransportId = cu.TransportId) tb
ON t.TripId = tb.TripId
LEFT JOIN Account a ON a.Id = t.Id
LEFT JOIN Carbon c ON tb.TransportId = c.TransportId
GROUP BY FirstName, LastName, MONTH(Date);
```

98 %

Results Messages

	FirstName	LastName	month	carbon_emission
1	Elijah	Fisher	1	203601
2	Elijah	Fisher	2	165762
3	Jasmine	James	2	1995
4	Jasmine	James	3	630
5	Jennifer	Johnson	1	267204
6	John	Ferguson	1	51846
7	John	Ferguson	2	34944
8	Michelle	Stella	2	346650
9	Michelle	Stella	3	477600
10	Natalie	Robinson	1	269814
11	Natalie	Robinson	2	8448
12	Patricia	Hamilton	2	229095
13	Patricia	Hamilton	3	840
14	Paula	Reyes	1	286680
15	Paula	Reyes	2	52992
16	Samuel	Patte...	1	1995



The size of the points indicates the value of the total carbon emission. We can see that Michelle reached the most total carbon level in March and Jasmine is at the bottom, indicating that Michelle might spend more on carbon-releasing transportation. The CarbonFootprint can message Michelle in March to tell him to pay more attention to his transportation. At the same time, the application could give compliments to those accounts reaching lower total carbon level.

Summary and Reflection

My database is designed for a company which design an application to analyze individual's contribution to the environment by calculating the gas emissions level of different means of transport. Miles, date, and transportation data can be used to calculate the gas emission, thereby evaluating people's contribution to environment. Feedbacks can also be shown to users to let them have a sense of how they choose to use the green transport when they go around. *CarbonFootprint* will potentially function as increasing people's awareness of environmental protection starting by choosing the better transportation.

I still need to assess the reasonableness of the classification of use cases. It seems the second to the fifth use cases relate to the first use cases separately, without direct connections among the second to the fifth use cases. In other words, the database does not have multiple levels of connections. This is what I am concerning about, and I am appreciated and looking forward to any feedbacks.

I built ERD for the entities I built based on my use cases. The ERD gives a clear logit between different entities and shows how the application can run. I have a question about the difference between **entity** needed for building structural database rules and **fields** defined in the use cases?

The structural database rules and conceptual ERD for my database design contain the important entities of Account, Booking, Car Use, Bus and Carbon, as well as relationships between them. The design contains a hierarchy of Booking/Flight and Booking/Train to reflect the two kinds of booking recorded in this database. The design also contains a hierarchy of Car Use/Private, Car Use/Car-hailing and Car Use/Delivery to indicate three main situations people use their car and make carbon emission.

I have added attributes to each of the entity, with most of the fields taking decimal, varchar, or date form, which is adequate to store information I need in my *CarbonFootprint* application.

I polished my database by inserting data to the tables I created through creating procedures which only require parameters to inserting new data. I also made some queries to filter out useful data frame for my CarbonFootprint application.

History table recording the total carbon due to the new trip records was created by trigger. Carbon emission of different transportation and individual cumulative total carbon level were visualized.