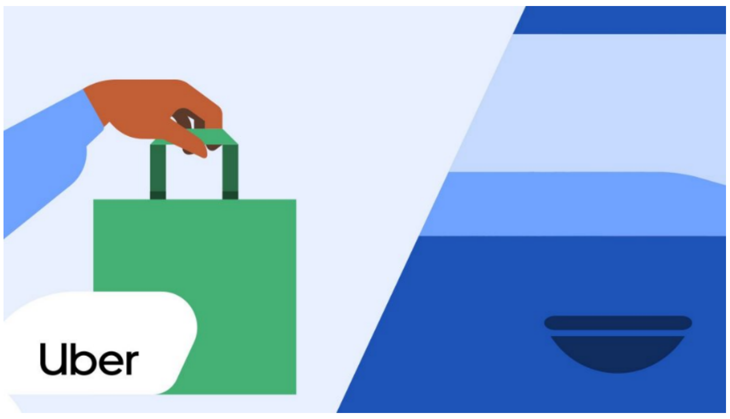
U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Base de Dados

Serviços Ober



MIEIC – BDad - Grupo 706
Luísa Marques up201907565@fe.up.pt
Margarida Raposo up201906784@fe.up.pt
Marta Mariz up201907020@fe.up.pt
april 2021

Index

Description

The goal is to save all the information regarding the services provided by the chain **Ober** that works like a broker, allowing the contact between the clients and the drivers to create different services associated with the transportation of people and packages.

The clients sign up to the app to use the available services. From each **client** it is necessary to save the name, contact, e-mail, credit card information and the NIF. A client is able to request a **service** and make the respective **payment**, which will have its specific id, the amount and the chosen payment method (Visa, Mastercard or American Express).

Two types of services are provided: Ober Ride (a people transportation service) and Ober Package (package delivery). In both cases, it's necessary to register the initial and final time and also the initial and final location, saving each's address, longitude and latitude.

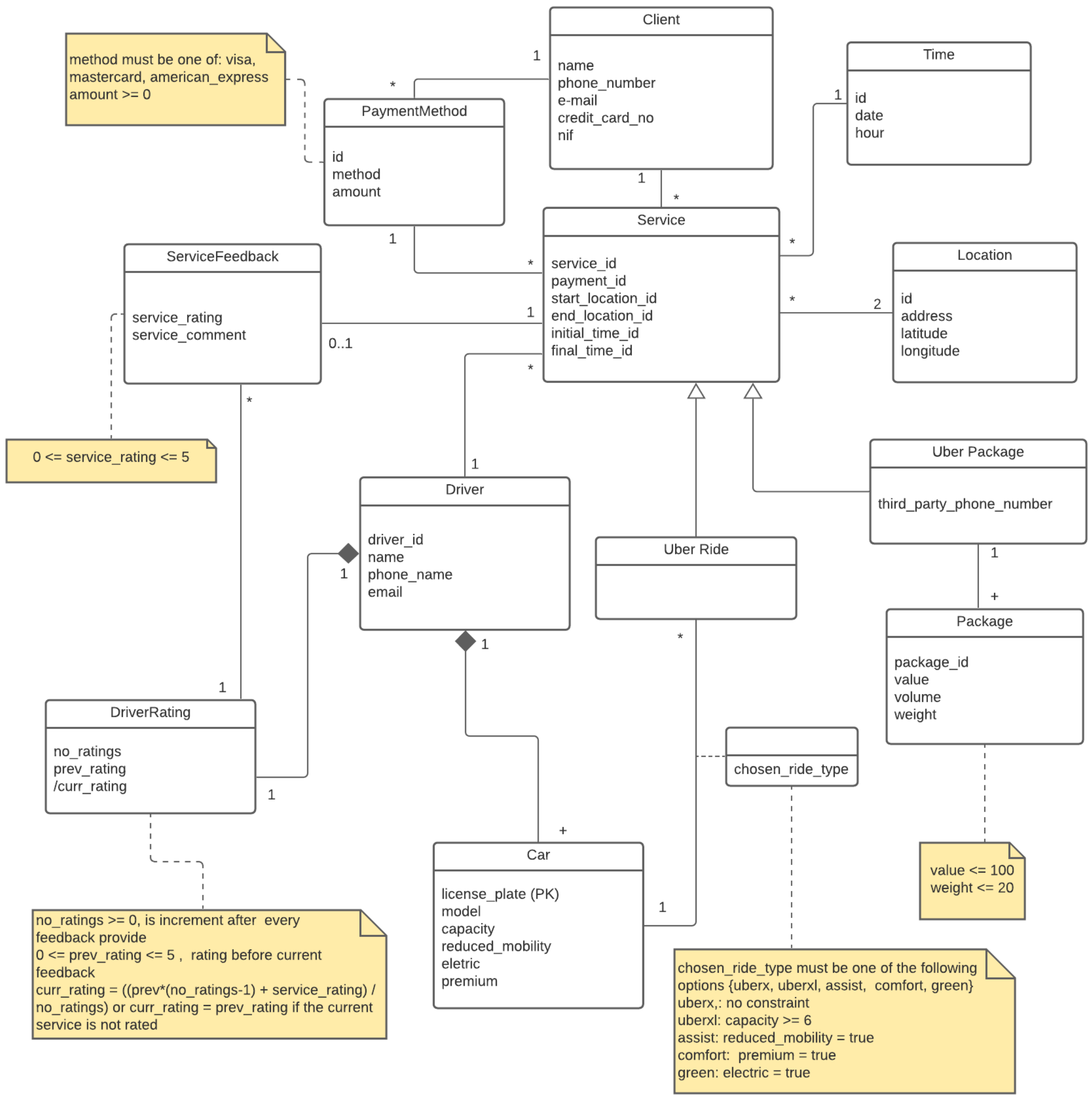
The **Ober package** service allows clients to request a trip that will take their order to a provided location. For safety reasons it's necessary to save the third party's contact, to whom the package will be delivered, and the **package**´s information: id, value, volume and weight. Only packages with value not superior to 100€ and weight equal or inferior to 20 kg will be accepted and posteriorly delivered.

Ober Ride is a private people transportation service, it can be of **different types**: OberX, Green, OberXL, Comfort and Assist. A car can provide these services if it has the necessary characteristics to do so.

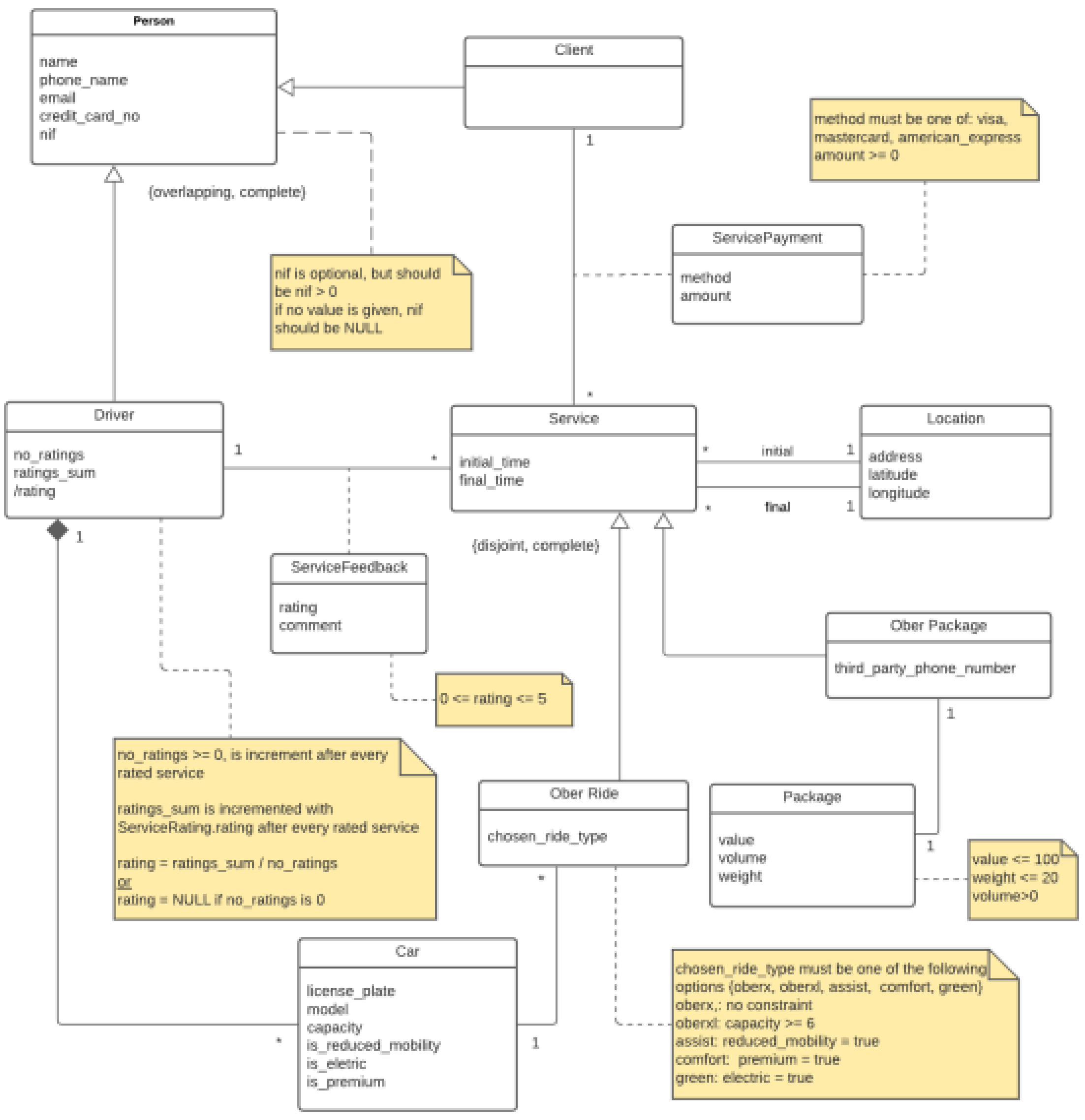
Each service has an associated **driver**, from which the id, the name, the phone contact and the email will be registered. A driver can have multiple **cars**, from which it's necessary to save its license plate, its model, and the specific characteristics that will specify the type of service that this type of vehicle is capable of providing: type of fuel, capacity and if it is appropriate to transport people with reduced mobility. In an OberGreen service the associated vehicle has to be electrical, in an OberXL the associated vehicle has to be capable of transporting at least 6 clients, in an Ober Comfort service the associated vehicle has to be previously evaluated and considered to be premium, and lastly in an OberX service the associated vehicle is only required to have the basic conditions: be able to transport at least 4 clients per trip.

After using the service, the client will be asked to provide **feedback**, it will be composed of a service evaluation (from 0 to 5) and a comment. Each driver has saved the number of feedback evaluations received and its sum, with each feedback gotten, the average rating will be recalculated taking into account all the feedback previously received.

Initial Conceptual Model



Updated Conceptual Model



Relational Model and Functional Dependencies

Client(client_id, email, name, nif, phone_num, credit_card_num)

email→ client_id, name, nif, phone_num, credit_card_num
client_id→ email, name, nif, phone_num, credit_card_num
nif→ client_id, name, phone_num, credit_card_num
email, client_id and nif are keys
client_id is the primary key

ServicePayment(service_id→Service, client_id→Client, method, amount)

service_id, client_id → method, amount
service_id and client_id are the composite primary key
service_id and client_id are foreign keys

Driver(driver_id, email, name, nif, phone_num, credit_card_num, no_ratings, rating, ratings_sum)

driver_id → email, name, nif, phone_num, credit_card_num, no_ratings, ratings_sum, rating
no_ratings, ratings_sum → rating
driver_id is the primary key
no_ratings and ratings_sum are a composite key (BCNF violation)

ServiceFeedback(rating, comment, driver_id→ Driver, service_id→Service)

driver_id, service_id → rating, comment
driver_id and service_id are the primary composite key
driver_id and service_id are foreign keys

Car(license_plate, model, capacity, reduced_mobility, electric, premium, driver_id→Driver)

license_plate → model, capacity, reduced_mobility, electric, premium, driver_id
license_plate is the primary key
driver_id is a foreign key

Location(location_id, latitude, longitude, address)

location_id → latitude, longitude, address
location_id is the primary key

Service(service_id, inital_time, final_time, initial_location_ id→Location, final_location_id→Location, driver_id→ Driver, client_id→Client)

service_id→ initial_time, final_time, comment, initial_location_ id, final_location_id, driver_id, client_id

OberRide(service_id→Service, chosen_ride_type)

service_id → chosen_ride_type
service_id is a foreing key
service_id is the primary key

OberPackage(service_id→Service, third_party_phone_number)

service_id → third_party_phone_number
service_id is a foreign key
service_id is the primary key

Package(package_id, value, volume, weight, service_id→OberPackage)

package_id → value, volume, weight, service_id
package_id is the primary key
service_id is a foreign key

Normal Forms Analysis

For a relational model to be in the Boyce-Codd Normal Form (BCNF) and, consequently, in the 3rd Normal Form (3NF) in each of the relations, the left side of the functional dependency has to be a key for that relation.

In the relations described above, there is one BCNF violation in the Driver:

{no_ratings, ratings_sum}+ = {no_ratings, ratings_sum, rating}

The attributes no_ratings and ratings_sum form a composite key which closure does not include all the attributes in that class. It only includes "rating" because it is a derived attribute which is a mean calculated using the attributes in the composite key.
Knowing that BCNF requires trivial functional dependencies or the left-hand side of a functional dependency to be a superkey, this FD represents a violation:

- It is not trivial - the right-hand side is not a subset of the left-hand side;
- It does not have a superkey on the left-hand side - {no_ratings, ratings_sum} is not a superkey.

According to the requirements of this 2nd delivery, we have decided to keep this violation for two reasons. Firstly, creating a new class would be unnecessary since the only attribute left in Driver would be driver_id and the organization of the database would be jeopardized. Secondly, if we had decided to create a new class, we would have to insert a new id or the driver_id as a foreign key. This is because the number and sum of ratings do not provide sufficient information to identify the driver, which would create the same problem we had before.

Note: We will be able to remove this violation in the next delivery by implementing queries.

Restrictions

Form restrictions (primary keys, foreign keys, unique, check and not null) have been implemented to ensure the maintenance of the integrity of the database and additional security.

Service		
service_id	Primary key representative id of a service. If it exists, it cannot be null.	service_id INTEGER PRIMARY KEY
initial_time	Instant the service is requested by the user. It cannot be null. Automatically generated by the system.	initial_time DATETIME NOT NULL
final_time	Moment the service finishes. It cannot be null. Automatically generated by the system.	final_time DATETIME NOT NULL
initial_location_id	Foreign Key associated with Location class. Location the client indicates when requesting the service.	initial_location_id TEXT NOT NULL REFERENCES Location(location_id)
final_location_id	Foreign Key associated with Location class.Location the client indicates when requesting the service.	final_location_id TEXT NOT NULL REFERENCES Location(location_id)
driver_id	Foreign key associated with Driver class. Used since the service is performed by a driver.	driver_id INTEGER NOT NULL REFERENCES Driver(driver_id)
client_id	Foreign key associated with Client class. Used since the user requests and enjoys the service.	client_id INTEGER NOT NULL REFERENCES client(client_id)
derived classes	Service has two derived classes: Ober Package and Ober Ride. Both have a foreign key associated with this class.	service_id INTEGER REFERENCES service(service_id)

Ober Ride		
ober_ride_id	Unique representative id of ober ride. If it exists, it cannot be null.	ober_ride_id INTEGER PRIMARY KEY NOT NULL
service_id	Foreign Key associated with Service class. Used since Ober Ride is a class derived from Service	service_id INTEGER NOT NULL REFERENCES service(service_id)
license_plate	Foreign Key associated with Car class. Used since Ober Ride Service uses a car.	license_plate TEXT NOT NULL REFERENCES car(license_plate)
chosen_ride_type	chosen_ride_type can be one of: oberx, oberxl, assist, confort, green. If an Ober Ride is required, chosen_ride_type must be given.	chosen_ride_type TEXT NOT NULL CHECK(chosen_ride_type in (“oberx”, “oberxl”, “assist”, “confort”, “green”))

Ober Package		
ober_package_id	Unique representative id of ober package. If it exists, it cannot be null.	ober_package_id INTEGER PRIMARY KEY NOT NULL
service_id	Foreign Key associated with Service class. Used since Ober Package is a class derived from Service	service_id INTEGER NOT NULL REFERENCES service(service_id)
third_party_phone_num	If an Ober Package service is requested, a third_party_phone_num has to be given.	INTEGER NOT NULL

Client		
client_id	PK representative id of a client. If it exists, it cannot be null.	client_id INTEGER PRIMARY KEY
email	Unique representative email of a client. If a client exists, he has to have an email.	email TEXT UNIQUE NOT NULL
name	Name of a client. If a client exists, he has to have a name.	name TEXT NOT NULL
nif	Unique representative nif of a client. NIF is not mandatory.	nif INTEGER UNIQUE
phone_num	Unique number of a client. If a client exists, he has to have a phone number.	phone_num INTEGER UNIQUE NOT NULL
credit_card_num	Unique credit card number of a client. Used as one of the payment methods' requirements.	credit_card_num INTEGER UNIQUE

Driver		
driver_id	Unique PK representative id of a driver. If it exists, it cannot be null.	driver_id INTEGER PRIMARY KEY
email	Unique representative email of a driver. If a driver exists, he has to have an email.	email TEXT UNIQUE NOT NULL
name	Name of a driver. If a driver exists, he has to have a name.	name TEXT NOT NULL
nif	Unique representative nif of a driver. NIF is not mandatory.	nif INTEGER UNIQUE
phone_num	Unique phone number of a driver. If a driver exists, he has to have a phone number.	phone_num INTEGER UNIQUE NOT NULL
credit_card_num	Unique credit card number of a driver. Used as one of the payment methods' requirements.	credit_card_num INTEGER UNIQUE
num ratings	Total number of feedbacks associated with the driver. If feedback exists, num_ratings cannot be null or less than zero.	(using triggers, to be implemented futurely)
ratings_sum	Sum of all the feedback ratings associated with the driver. If feedback exists, ratings_sum cannot be null.	(using triggers, to be implemented futurely)
rating	rating is calculated dividing ratings_sum by the num_ratings	(using triggers, to be implemented futurely)

Car		
license_plate	Unique PK representative license plate of a car. If it exists, the license plate cannot be null.	license_plate TEXT PRIMARY KEY
model	If a car exits, its model has to be given.	model TEXT NOT NULL
capacity	If a car exists, capacity cannot be null and it must be greater than 1.	capacity INT NOT NULL CHECK(capacity > 1)
is_reduced_mobility	A car can be suitable to transport people with reduced mobility. This value is 0 by default.	is_reduced_mobility INTEGER NOT NULL DEFAULT 0 CHECK(is_reduced_mobility IN (0,1))
is_electric	A car can be electric. This value is 0 by default.	is_electric INTEGER NOT NULL DEFAULT 0 CHECK(is_electric IN (0,1))
is_premium	A car can be premium if its capacity is greater than or equal to 6. This value is 0 by default.	is_electric INTEGER NOT NULL DEFAULT 0 CHECK(is_electric IN (0,1))
driver_id	Foreign Key associated with Driver class. Used since the feedback is given to a specific driver.	driver_id INTEGER NOT NULL REFERENCES driver(driver_id)

Package		
package_id	Unique PK representative id of a package. Cannot be null.	package_id INTEGER PRIMARY KEY
value	Value of a package. It cannot be null or less than 0 or greater than 100.	value INT NOT NULL CHECK(value > 0 AND value <= 100)
weight	Weight of a package. Cannot be null or greater than 20.	weight DOUBLE NOT NULL CHECK(weight <= 20)
ober_package_id	Foreign Key associated with ober_package class. Used since the package is delivered in a specific service.	service_id INTEGER NOT NULL REFERENCES ober_package(ober_package_id)

Location		
location_id	Unique PK representative id of a location resulting from the joining of latitude, longitude and address. If it exists, the license plate cannot be null.	location_id INTEGER PRIMARY KEY
latitude	Unique representative latitude of a location. Cannot be null.	latitude DOUBLE NOT NULL
longitude	Unique representative latitude of a location. Cannot be null.	longitude DOUBLE NOT NULL
address	Unique representative string address of a location. Cannot be null.	address TEXT UNIQUE NOT NULL

Service Payment		
service_id	Foreign Key associated with Service class. Used since it is the payment of a specific service.	service_id INTEGER NOT NULL REFERENCES service(service_id)
client_id	Foreign Key associated with Client class. Used since a payment is made by a customer.	cllient_id INTEGER NOT NULL REFERENCES client(client_id)
method	Payment method can be one of: visa, mastercard, american_express. If a payment is made, a method is required.	method TEXT NOT NULL CHECK(method in (“visa”, “mastercard”, “american_express”))
amount	Amount paid for the service. It cannot be null or greater than 0.	amount INT NOT NULL CHECK(amount > 0)
service_id, client_id	Composite primary key	PRIMARY KEY(service_id, client_id)

Service Feedback		
rating	If Feedback exits, cannot be null and must be greater than 0.	rating INT NOT NULL CHECK(rating > 0)
comment	Comment is not mandatory.	comment TEXT
driver_id	Foreign Key associated with Driver class. Used since the feedback is given to a specific driver.	driver_id INTEGER NOT NULL REFERENCES driver(driver_id)
service_id	Foreign Key associated with Service class. Used since the feedback is given to a specific service.	service_id INTEGER NOT NULL REFERENCES service(service_id)
service_id, driver_id	Composite primary key	PRIMARY KEY(service_id, driver_id)

Queries

We have implemented all 10 queries which are presented and described below in natural language.

Query 1
Lists every Client identifier number and name which has joined the app and created an account, but who has not yet required any service. This query may be useful to understand if there is a significant number of clients who end up never using the app after downloading it.
Query 2
Lists every provided service along with the Client who received it, the Driver who delivered it, the Car used for the transportation and the duration of the trip.
Query 3
Shows every electric Car which is available at the moment to perform a Service.
Query 4
Displays all Drivers and indicates if they can or cannot perform a specific kind of service due to the type of car they own.
Query 5
Lists by method of payment which is the total transferred value, the number of services used by that method and the mean value spent by trip.
Query 6
Shows a "calendar" with the addresses where services began or ended on a certain date.
Query 7
Displays information about Services which were evaluated by the Clients with less than 3 (in a scale of 1 to 5). This information includes the names of the Client and the Driver and the respective rating given to that Service.
Query 8
Lists the name, email address and phone number of the people who are simultaneously Drivers and Clients.
Query 9
Checks all users and finds out if they have already been part of a Service Ober Package as a receiver (third party).
Query 10
Obtains the maximum value ever paid for a Service.

Triggers

We have implemented all 3 triggers which are presented and described below in natural language.

Trigger 1

When any new Service feedback is added to the database, the sum of the driver rating is recalculated (the rating given is added) and the total number of driver ratings is incremented.

Trigger 2

Checks if the driver of a particular car exists in the database.

Trigger 3

Prevents a service from occurring when the client and the driver are the same person (when they have the same id).