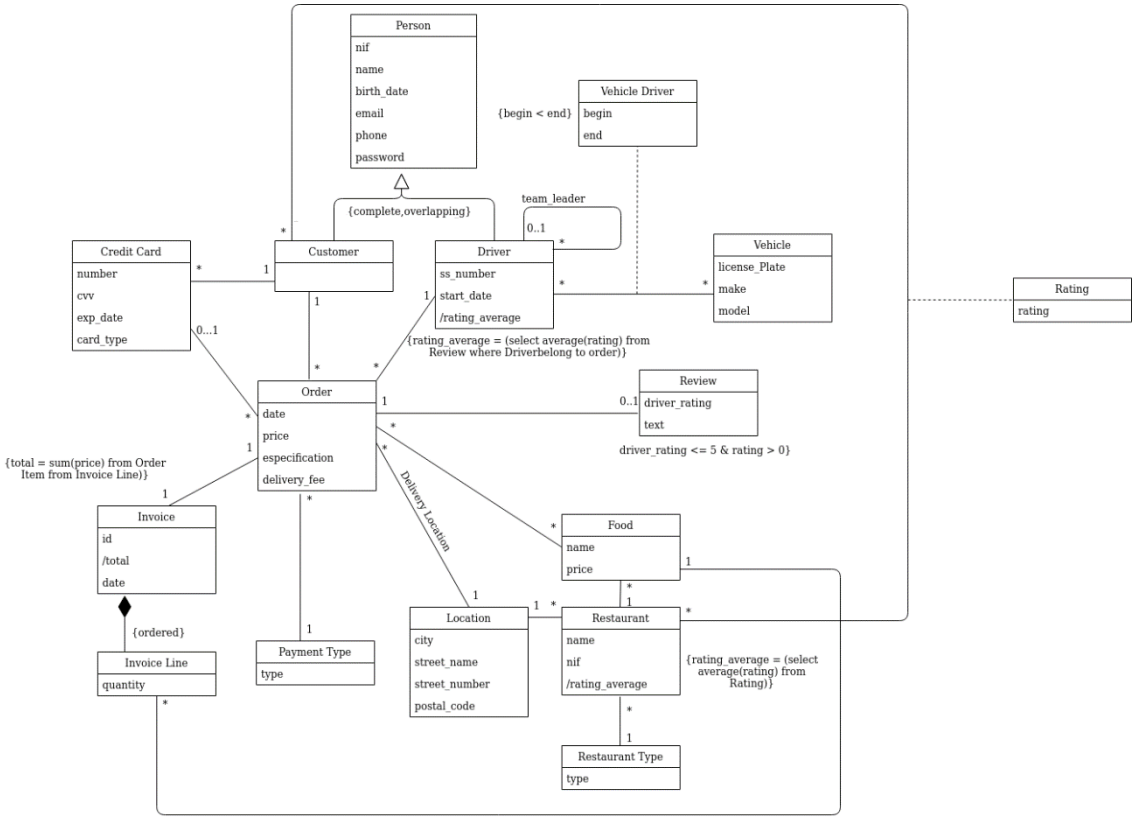
BDAD – Relatório: Empresa de Entregas

João Martins, José Miguel Maçães e Miguel Charchalis

Esquema UML:



Contexto:

De cada cliente interessa saber o nome, o NIF, o email, a data de nascimento, o número de telemóvel e a palavra passe. Isto tudo também é informação necessária relativa a cada condutor, para além do número de segurança social e a data em que começou a trabalhar na plataforma. Para além disso interessa saber as horas de início e de fim da sessão de trabalho do condutor.

Cada cliente necessita de ter um cartão de crédito, do qual deve ser conhecido o número, o CVV, a data de validade e a rede (VISA, MasterCard, etc.)

Cada condutor deve ter a si associado um veículo, identificado pela matrícula, do qual interessa saber a marca e o modelo. Um condutor pode ser chefe de equipa, sendo responsável por outros condutores.

O cliente pode efetuar o pedido, que será entregue por um condutor e é constituído por uma data, um modo de pagamento e um preço, calculado a partir do preço da comida e da taxa de entrega. Para além disso terá uma avaliação que terá uma classificação, entre 1 e 5, e poderá ou não ter um texto. As classificações são usadas para calcular a média de classificações do condutor. O pedido estará também associado a um local de entrega, caracterizado pela cidade, nome da rua, número da rua, código postal.

O Restaurante tem como elementos identificativos o nome, o tipo, a localização, da qual interessa saber o mesmo que o local de entrega, e a classificação, é a média de classificações de clientes. Disponibiliza pratos, cada um com um nome e preço, que constituem os itens pedidos, juntamente com a quantidade de cada prato. Estes itens fazem parte do pedido e estão presentes na fatura, também associada ao pedido composta por um número identificativo, os itens pedidos e a sua quantidade, o preço total e a data.

Modelo Relacional:

Primary key de cada relação está sublinhada.

Person (NIF, name, birth_date, email, phone, password)

{NIF} -> {name, birth_date, email, phone, password}

Não viola BCNF porque todos os atributos à direita são não primos.

NIF, birth_date, phone, password não podem ser nulos.

Cumprimento de phone maior ou igual a 9.

NIF tem de ter tamanho 9.
Não podem haver 2 person com NIF igual.

Customer (customerNIF -> Person)

Só um atributo, não viola BCNF.

Não podem haver 2 customer com customerNIF igual.

Driver (driverNIF -> Person, ss_number, start_date, /rating_average)

{driverNIF} -> {ss_number, start_date, /rating_average}

Não viola BCNF porque todos os atributos à direita são não primos.

driverNIF, ss_number, start_date, password não podem ser nulos.

Não podem haver 2 driver com driverNIF igual.

Comprimento ss_number maior ou igual a 11.

Vehicle (license_plate, make, model)

{license_plate} -> {make, model}

Não viola BCNF porque todos os atributos à direita são não primos.

License_plate, make, model não podem ser nulos.

Não podem haver 2 driver com license_plate igual.

Tamanho de license_plate igual a 8 (a contar com “-”).

VehicleDriver (driverNIF -> Driver, vehicle_license_plate -> Vehicle, begin, end)

{driverNIF,vehicle_license_plate} -> {begin, end}

Violação: Usar o “driverNIF” e “vehicle_license_plate” como primary key leva à repetição da primary key e a mesma key se relacionar com diferentes valores. Uma possível solução é trocar “vehicle_license_plate” com “begin”, pois é impossível o mesmo condutor começar a conduzir duas vezes no mesmo tempo, torna a primary key única.

VehicleDriver (driverNIF -> Driver, vehicle_license_plate -> Vehicle, begin, end)

{driverNIF, begin} -> {vehicle_license_plate, end}

{driverNIF} -> {end}

{begin} -> {vehicle_license_plate}

{driverNIF, begin} -> {vehicle_license_plate, end}

Não viola BCNF porque nenhum atributo no lado direito depende de outro no lado direito: end não depende de vehicle_license_plate, nem vice-versa. Apesar da chave ser única e respeitar a normalização, não impede que um condutor e um carro tenha períodos sobrepostos.

Nenhum atributo pode ser nulo.

Não podem haver 2 vehicleDriver com driverNIF e begin iguais.

Team (driverNIF -> Driver, leaderNIF -> Driver)

{diverNIF} -> {leaderNIF}

Não viola BCNF porque só há um atributo de cada lado.

No pode haver driverNIF repetidos, cada driver só tem um leader. Ou nenhum, se for ele um leader.

CreditCard (number, cvv, exp_date, card_type, customerNIF -> Customer)

{number, customerNIF} -> {cvv, exp_date, card_type}

Viola BCNF pois duas pessoas podem usar o mesmo cartão, então a informação está repetida. Neste caso não é pretendido o update nem delete geral devido à proteção de dados: se um cliente mudar algum dado, por exemplo o cvv, não se pretende que mude também para outros clientes que estão a usar o mesmo cartão noutra conta, pois devem ser os mesmos a mudar.

Nenhum atributo pode ser nulo.

Não pode existir dois cartões iguais ligados ao mesmo customer.

Comprimento de number maior ou igual a 13 e menor ou igual a 19.

Comprimento de cvv igual a 3.

Demand (demandID, date, /price, specification, delivery_fee, customerNIF -> Customer, driverNIF -> Driver, locationID -> Location, paymentTypeID -> PaymentType, creditCardID -> CreditCard)

{demandID} -> {date, /price, specification, delivery_fee, customerNIF, driverNIF, locationID, paymentTypeID, creditCardID}

{date} -> {delivery_fee}

A segunda DF viola a BCNF porque o lado esquerdo da DF (date) não é uma superchave, ou seja, o seu fecho não contém a totalidade dos atributos da relação.

Viola BCNF, mas não viola 3NF porque existe atributos à direita que dependem de outra key.

Date, delivery_fee, price, customerNIF, driverNIF, locationID e paymentTypeID não podem ser nulos.

Não pode existir 2 demand com o mesmo demandID.

delivery_fee maior ou igual a 0.

price maior ou igual 0 (não inclui delivery_fee).

PaymentType (paymentTypeID, type)

FD = {paymentTypeID -> type}

{paymentTypeID} -> {type}

Não viola BCNF porque todos os atributos à direita são não primos e só tem um atributo de cada lado.

Nenhum atributo pode ser nulo.

Não pode haver elementos repetidos.

Review (reviewID, rating, text, demandID -> Demand)

{reviewID} -> {rating, text, demandID}

Não viola BCNF porque só pode existir uma review por demand e se a demand for apagada então a review também devia ser.

Pode não existir uma review numa demand, mas se existir não pode ter atributos nulo.

Não pode haver mais que uma review por demand, nem reviewID duplicados.

Rating maior ou igual a 1 e menor ou igual a 5.

Rating (ratingID, rating, restaurantID -> Restaurant) - Class associação.

{ratingID} -> {rating, restaurantID}

Não viola BCNF porque todos os atributos à direita são não primos.

Pode não existir um rating para um restaurant, mas se existir não pode ter atributos nulo.

Não pode haver mais que uma rating por restaurant, nem ratingID duplicados.

Rating maior ou igual a 1 e menor ou igual a 5.

Food (foodID, name, price, restaurantID -> Restaurant)

{foodID} -> {name, price, restaurantID}

Viola BCNF porque o Restaurant, name e price dependem todos da foodID, mas não viola 3NF porque existe atributos à direita que são parte de outra key.

foodID, name, price e restaurantID não podem ser nulos.

price tem de ser maior que 0.

Não pode haver 2 Restaurant com o mesmo restaurantID.

Demanded (DemandedID, foodID -> Food, demandID -> Demand, quantity)

{ DemandedID} -> {foodID, demandID, quantity}

Viola BCNF porque a quantity, foodID e demandID dependem da demandedID, mas não viola 3NF porque existe atributos à direita que são parte de outra key.

Restaurant (restaurantID, name, NIF, locationID ->Location, /rating_average, restaurantTypeID -> RestaurantType)

{restaurantID} -> {name, NIF, locationID, /rating_average, restaurantTypeID}

Não viola BCNF porque todos os atributos à direita dependem do restaurantID.

restaurantID, name, NIF, locationID e restaurantTypeID não podem ser nulos.

NIF tem de ter tamanho 9.

Não pode haver 2 Restaurant com o mesmo restaurantID.

RestaurantType (restaurantTypeID, type)

{restaurantTypeID} -> {type}

Não viola BCNF porque todos os atributos à direita são não primos e só tem um atributo de cada lado.

restaurantTypeID e type não podem ser nulos.

Não pode haver 2 RestaurantType com o mesmo restaurantTypeID.

Location (locationID, city, street_name, street_number, postal_code)

{locationID} -> {city, street_name, street_number, postal_code}

Não viola BCNF porque todos os atributos à direita são não primos.

locationID, city, street_name e postal_code não podem ser nulos.

Não pode haver 2 Location com o mesmo locationID.

Invoice (id, /total, date, demandID -> Demand)

{id} -> {/total, date, demandID}

Viola BCNF porque a date depende da demandID, mas não viola 3NF porque existe atributos à direita que são parte de outra key.

invoiceID, total, date e demandID não podem ser nulos.

Não pode haver duas Invoice com o mesmo invoiceID.

InvoiceLine (invoice_lineID, quantity, demandedID -> Demanded, invoiceID -> Invoice)

{invoice_lineID} -> {quantity, demandedID , invoiceID}

Viola BCNF porque a quantity depende de demandedID , mas não viola 3NF porque existe atributos à direita que são parte de outra key.

invoice_lineID, demandedID e invoiceID não podem ser nulos.

Não pode haver duas InvoiceLine com o mesmo invoice_lineID