Faculdade de Engenharia da Universidade do Porto

Base de dados

Online Retail Database

Carlos Soeiro up201706405
Luís Silva up201808912
Martim Silva up201705205

Relatório elaborado no âmbito da unidade curricular Base de Dados
Mestrado Integrado em Engenharia Informática e de computação

06 de abril de 2019

Introduction

The online retail market has a very relevant impact in the global economic structure. The aim of this project was to create a database for an online store. This database must store the different users. It stores if the user has high access privileges (administrators), their order history, their address, make sure the user is legally an adult (considering 18 years old has reference age), storing the buyers’ opinions on products, checking if there is an sale in any of the products which are being bought amongst other information about the products and the manufacturers.

Project specification’s

A customer must be over 18 years old, has a name, a unique id, an email and a social security number. Their location (postal code and country) is also stored has a foreign key.

The admins, which are also customers, have stored their employee ID, their job title and which sector of the company they work on.

Every product in the store has a name, a unique id, the line of products to which belongs (dairy, tech, gardening…), its price and stock information (Boolean value). The reviews of the products are left by the customers and have a rating (15) and the date of review. A customer can do at most one review for each product.

Products of online stores are often in discount. These offers have a start date and a finish date (always superior to the start date) and the percentage of discount as well ([0, 1]).

A product is made by a company therefore, information about each company must also be stored such has its name, its city (since a company may have different branches on different cities of the same country, so the name is not enough for identification), and its revenue.

Customers’ orders have a unique Id, an order date, a delivery date (always superior to the order date) and the total price of the order (result of the sum of the value of each of the products multiplied by its quantity). The relation of the products with the order must assure that the quantity of the product is superior to zero.

A customer can have multiple orders at the same time in its cart. The cart stores the total price (sum of the prices of the orders). The payment of the cart is done using one of four different methods: PayPal, Visa, MasterCard, Shop Card. When using the Shop Card as payment method you are not required to insert your credit card information.

Relational Model and Functional Dependencies

Customer (id, ssn→CustomerDetails, email)

id → ssn, email

CustomerDetails (ssn, name, birthDate)

ssn → name, birthdate

Location (id, postalCode, country)

id → postalCode, country

Admin (customerId→Customer, employeeId, sector, jobTitle)

customerId → employeeId, sector, jobTitle

Orders (id, orderDate, customerId→Customer, cartId→Cart)

id → orderDate, customerId, cartId

Cart (id, paymentId→Payment)

id → paymentId

Payment (id, paymentMethod, creditCardNumber)

id → paymentMethod, creditCardNumber

Company (id, name, city, revenue)

id → name, city, revenue

Product (id, name, line, price, inStock, discountId→ProductDiscount, companyId→Company)

id → name, line, price, inStock, companyId, discountId

ProductReview (rating, date, customerId→Customer, productId→Product)

customerId, productid → rating, date

ProductDiscount (id, discount, startDate, finishDate

id → discount, startDate, finishDate

QuantityOfProduct (productId→Product, orderId→Order, quantity)

productId, orderId → quantity

CustomerLocation (customerId→Customer, locationId→Location)

Normal Form Analysis

- In all the functional dependencies there is a key on the left side, therefore the relational model is in the Boyce-Codd Normal Form.
- There was a BCNF violation found in Customer table and because of that, it was necessary to separate Customer from CustomerDetails.
- Also, because the model is in the BCNF implies that it is also in 3rd Normal Form.

Restrictions

Customer:

- id is the primary key (key restriction, PRIMARY KEY)
- email is the email of the user and is unique (UNIQUE)
- The user's ssn is unique (UNIQUE)

CustomerDetails:

- ssn is the primary key (key restriction, PRIMARY KEY) name is the user's legal name
- birthDate is the user's birth date which must be more than 18 years before today's date (now()-birthDate > 18)

Location:

- id is the primary key (key restriction, PRIMARY KEY)
- The pair country and postalCode are unique (UNIQUE)

Admin:

- customerId is the primary key (key restriction, PRIMARY KEY) and is a foreign key (referential integrity, FOREIGN KEY)
- employeeId is unique (UNIQUE)
- sector refers to the sector in the company where the employee works

Orders:

- orderId is the primary key (key restriction, PRIMARY KEY)
- customerId is a foreign key (referential integrity, FOREIGN KEY)
- cartId is a foreign key (referential integrity, FOREIGN KEY)

Cart:

- cartId is the primary key (key restriction, PRIMARY KEY)
- paymentId is a foreign key (referential integrity, FOREIGN KEY)

Payment:

- paymentMethod has only 4 options
- creditCardNumber must be only numeric and have 16 digits
- paymentId is the primary key (key restriction, PRIMARY KEY)

Company:

- id is primary key (key restriction, PRIMARY KEY)
- The pair name and city are unique (UNIQUE)

Product:

- productid is the primary key (key restriction, PRIMARY KEY)
- inStock is a Boolean value
- companyId is a foreign key (referential integrity, FOREIGN KEY)
- line is unique (UNIQUE)
- discountId is a foreign key (referential integrity, FOREIGN KEY)

ProductReview:

- rating is a discrete value (between 1 and 5)
- Pair productId, customerId is the primary key (key restriction, PRIMARY KEY)

ProductDiscount:

- id is the primary key (key restriction, PRIMARY KEY)
- discount is a REAL number between 0 and 1

QuantityOfProduct:

- productId is a foreign key (referential integrity, FOREIGN KEY)
- orderId is a foreign key (referential integrity, FOREIGN KEY)
- productId and orderId are a composite primary key (key restriction, PRIMARY KEY)

CustomerLocation:

- customerId and locationId are a composite primary key and foreign keys (key restriction, PRIMARY KEY) (referential integrity, FOREIGN KEY)

Queries

Query 1: Obtain all costumers that made a purchase and the number of orders they made in a given time period. This can be used to check when the sales are greater.

Query 2: Obtain all products and their quantity bought by a customer. Used to recommend products to a customer that are similar to what they usually buy.

Query 3: Obtain the 10 least sold products. Used to know which products to discount.

Query 4: Obtain the customers which benefited from the discounts and how much they saved. Useful to know if customers are taking advantage of the discounts.

Query 5: Obtain the products that have reviews with an average score bigger than 3 and how many times they were reviewed. Checks the best scored products.

Query 6: Obtain the starting and finishing dates of a discount applied to a product and the value of the promotion. Used to inform customers of discounts on certain products.

Query 7: Compare the quantity sold of a product with and without discount. Useful to know if customers are taking advantage of the discounts.

Query 8: Obtain how many orders are done in each country. Used to know which countries are buying from the online store.

Query 9: Obtain the best-selling product in all locations. This information can be good to know which products need more stock.

Query 10: Obtain the 10 customers with more purchases and reviews. Used to know the most active customers on the store.

Triggers

Trigger 1: This trigger updates the order's total price after the user selects another product to buy. It selects the product price, multiplies it buy the quantity that the user wants to buy and verifies if the product has a discount, applying it in case there is.

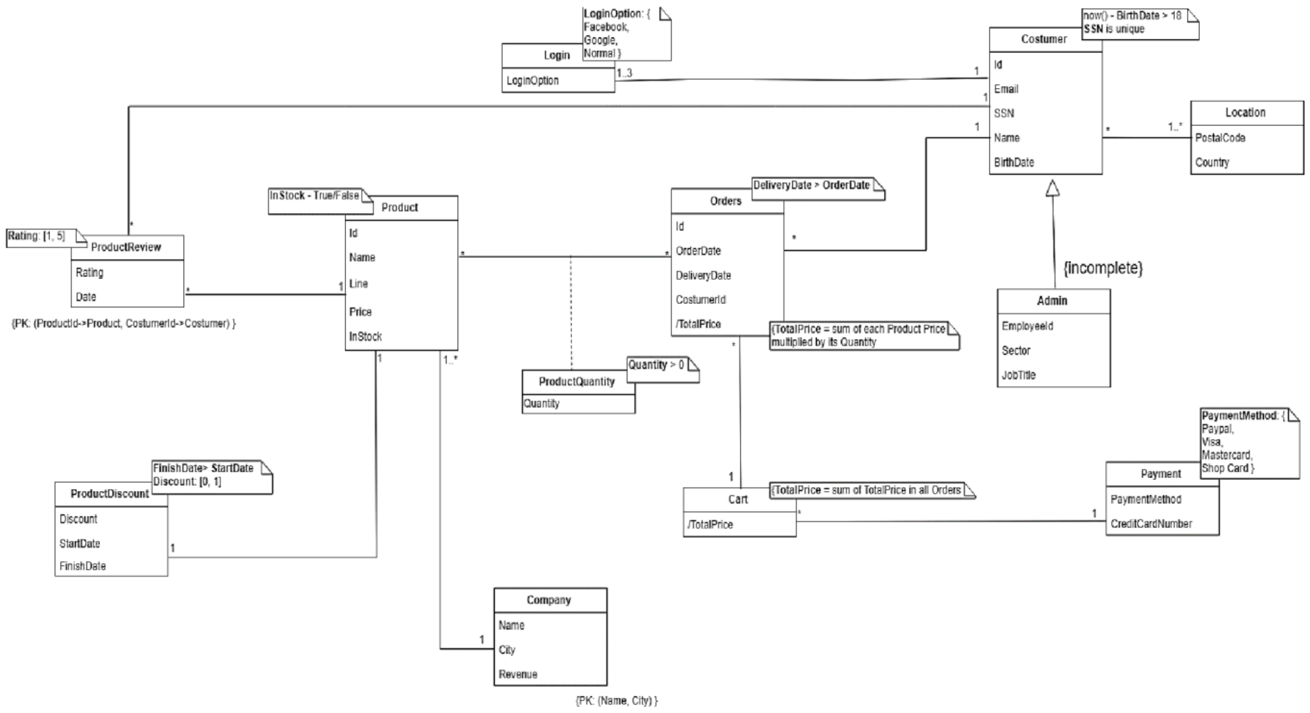
Trigger 2: This trigger updates the cart's total price after there is an insert in the order's table, this is, the total price the costumer must pay.

Trigger 3: This trigger updates the quantity of the product remaining in the warehouse and sets flags in case the store is running out of stock or as no stock left of that product. It is activated after the user defines a new order or the quantity of the product he is buying.

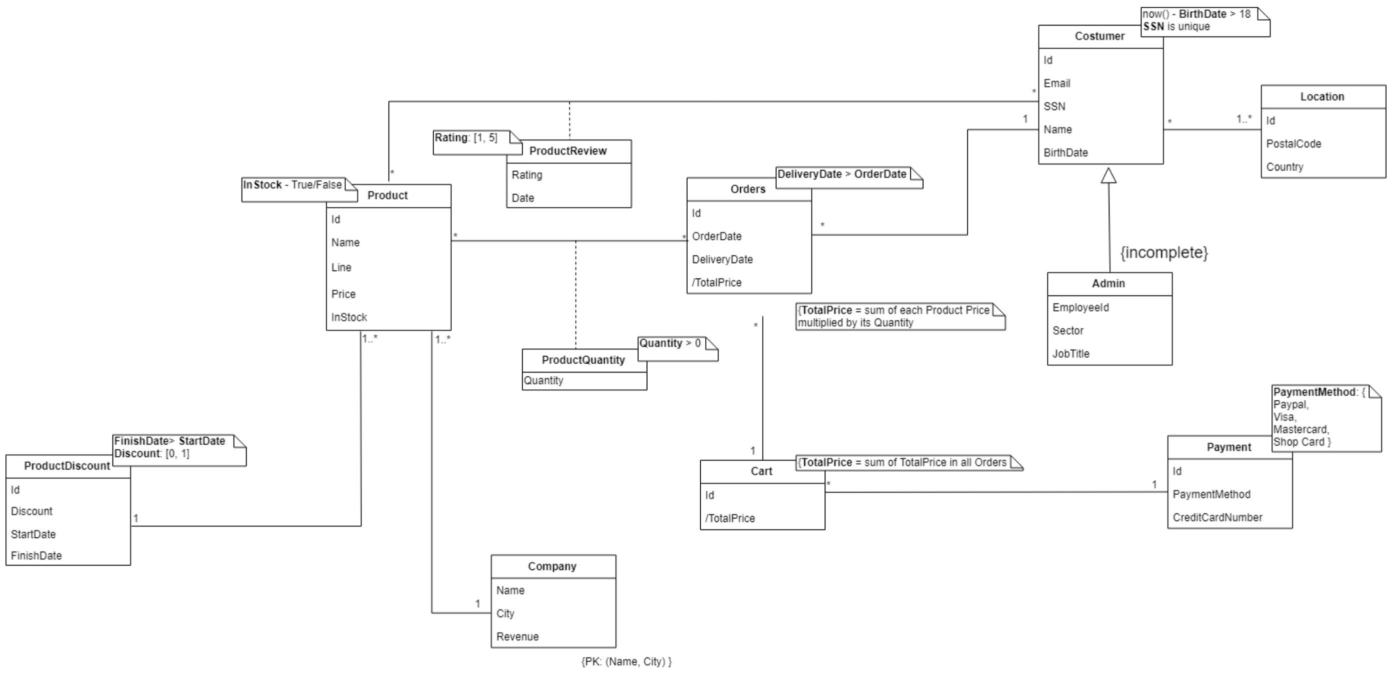
Trigger 4: This trigger checks if the user has bought the product and therefore can submit a review. To do so, after the user tries to submit a review. To do so, it searches for the ids of the orders in which that product was bought, and searches in the orders table which were the orders the costumer submitting the review made. It then finds out if the costumer is in both of these searches. If he isn't, the review is deleted (a rollback is done).

Trigger 5: This trigger checks if there is enough stock to fulfill the client order and if there isn't it removes that entry and raises an exception.

Initial Conceptual Model



Revised Conceptual Model


now() - BirthDate > 18
SSN is unique
Costumer
Id
Email
SSN
Name
BirthDate
*
1
*
1..*
Location
Id
PostalCode
Country
Rating: [1, 5]
ProductReview
Rating
Date
*
InStock - True/False
Product
Id
Name
Line
Price
InStock
DeliveryDate > OrderDate
Orders
Id
OrderDate
DeliveryDate
/TotalPrice
*
*
*
{incomplete}
Admin
EmployeeId
Sector
JobTitle
{TotalPrice = sum of each Product Price multiplied by its Quantity
Quantity > 0
ProductQuantity
Quantity
1..*
1..*
*
FinishDate> StartDate
Discount: [0, 1]
ProductDiscount
Id
Discount
StartDate
FinishDate
1
1
PaymentMethod: { Paypal, Visa, Mastercard, Shop Card }
Payment
Id
PaymentMethod
CreditCardNumber
Cart
Id
/TotalPrice
{TotalPrice = sum of TotalPrice in all Orders
*
1
Company
Name
City
Revenue
1
{PK: (Name, City) }