University College of Northern Denmark

Technology and Business

AP Graduate in Computer Science
dmaj0914

Persistence Workshop

Database creation, Database manipulation

Authors:

Blagovest Bozhinov
Georgi Karavasilev
Denis Nikolov
Ivan Manev
Mihail Tsanov

Supervisors:

Kis Boisen Hansen
Gianna Belle

Abstract

The purpose of this workshop is to develop user friendly software capable of handling database that contains product information, customer information and supplier information, the software must also handle sales of products. The software also includes user friendly GUI.

Table of Contents

I. Introduction

Group-based problem solving is the standart teaching style in Denmark. Throughout the semester there are workshops, their purpose is to strengthen this standart. The goal of this exact workshop is to develop and learn how to use databases. The software mus have MVC (Model, View, Control Layers), also in this workshop there is a fourth layer called DB Layer that serves the purpose of storing information about the database and its variables.

II. Analysis

During the development of this software, the layout and the GUI were discussed and edited based off a now four layer architecture, consisting of Model Layer, View Layer, Control Layer and Database Layer. While developing the software, we've put a lot of consideration about the end-user, we've also tought about how to make the GUI as simple as possible so the end-user is not confused.

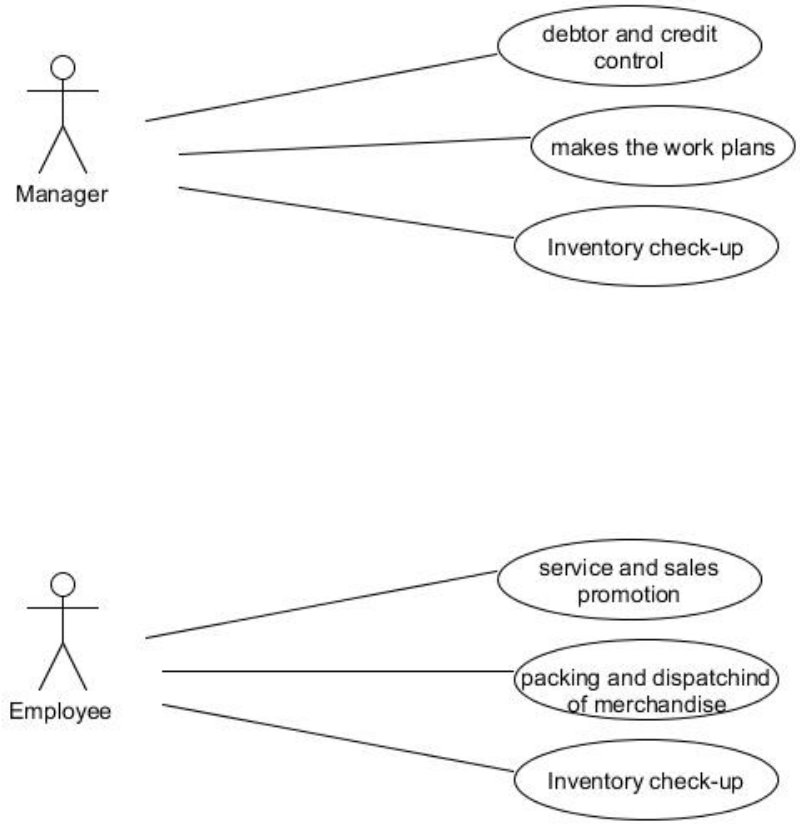
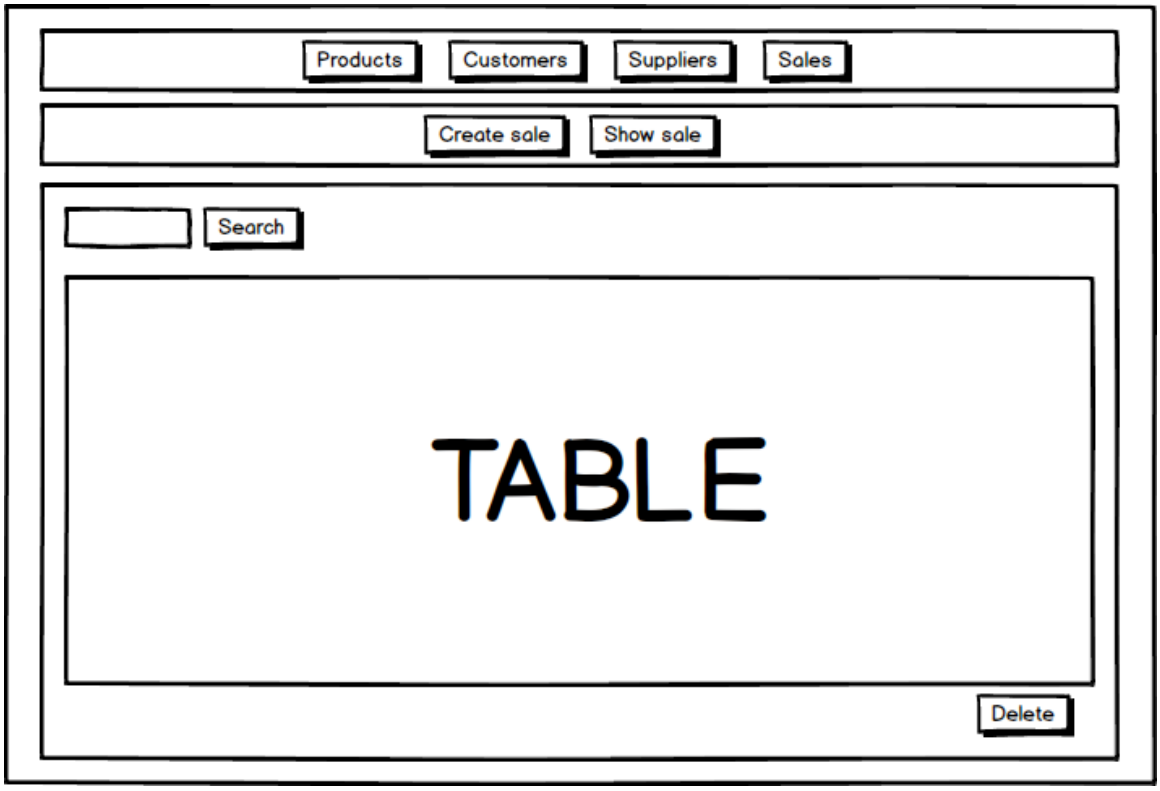


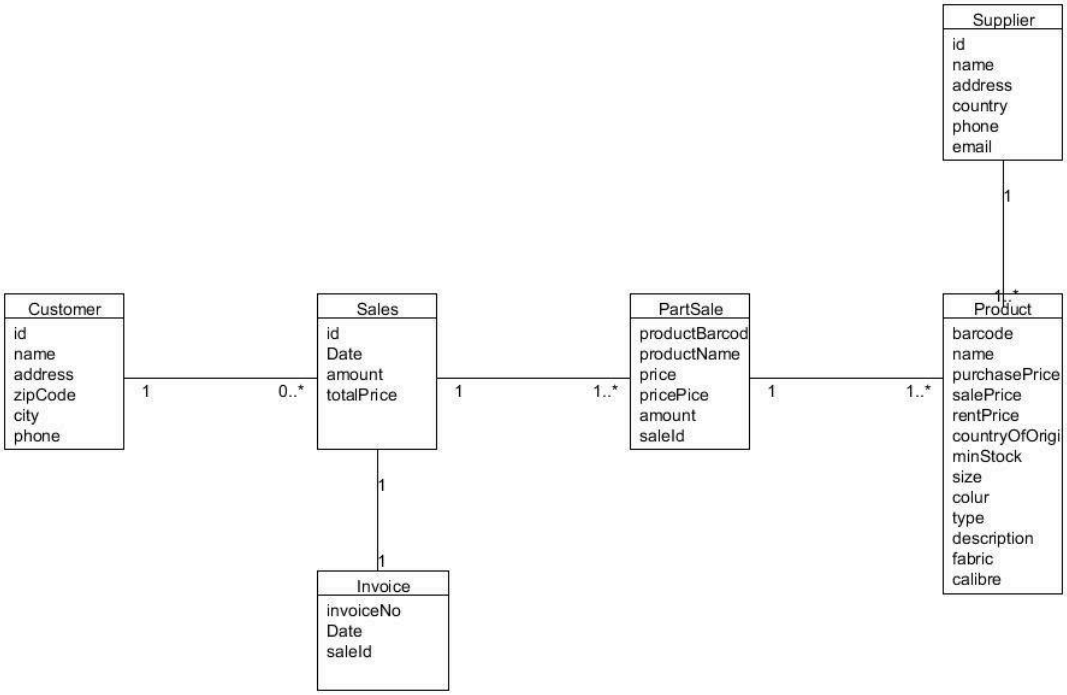
Figure 1. Use-Cases for Western Style

III. Mock-up



After discussion, we've decided to make the GUI as simple as possible and as user-friendly as it can be, with all functionality that the software is required to have. As we can see, on the first layer of buttons we have four different buttons that serve different purpose, namely Products, Customers, Suppliers and Sales. When clicked on them, another layer of buttons are shown Show and Create, which by the name ,I t can be judged that the Create button opens a table that you can create a new entry for either product, customer, supplier or a sale. When clicked on Show, we get a list of all products, customers, suppliers and sales made. The software also has a Search field and a Delete button.

IV. Domain Model



*In our domain model we have class Customer with variables **id**, **name**, **address**, **zipCode**, **city** and **phone**. The **Customer** class is connected with **sale** with one to many connection. In the **sale** we have **id**, **date**, **amount**, and **totalPrice**, also the **sales** class is connected with the **Invoice** class which have variables **invoiceNo**, **saleDate** and **amount**. The **sale** is also related with the **parthSale** class and the **contractor** class. The **product** has **barcode**, **name**, **purchasePrice**, **salesPrice**, **rentPrice**, **countryOfOrigin**, **minStock** fields. The **Product** class is connected with the **supplier** class, which supplier class has fields that are **id**, **name**, **address**, **country**, **phone** and **email**.*

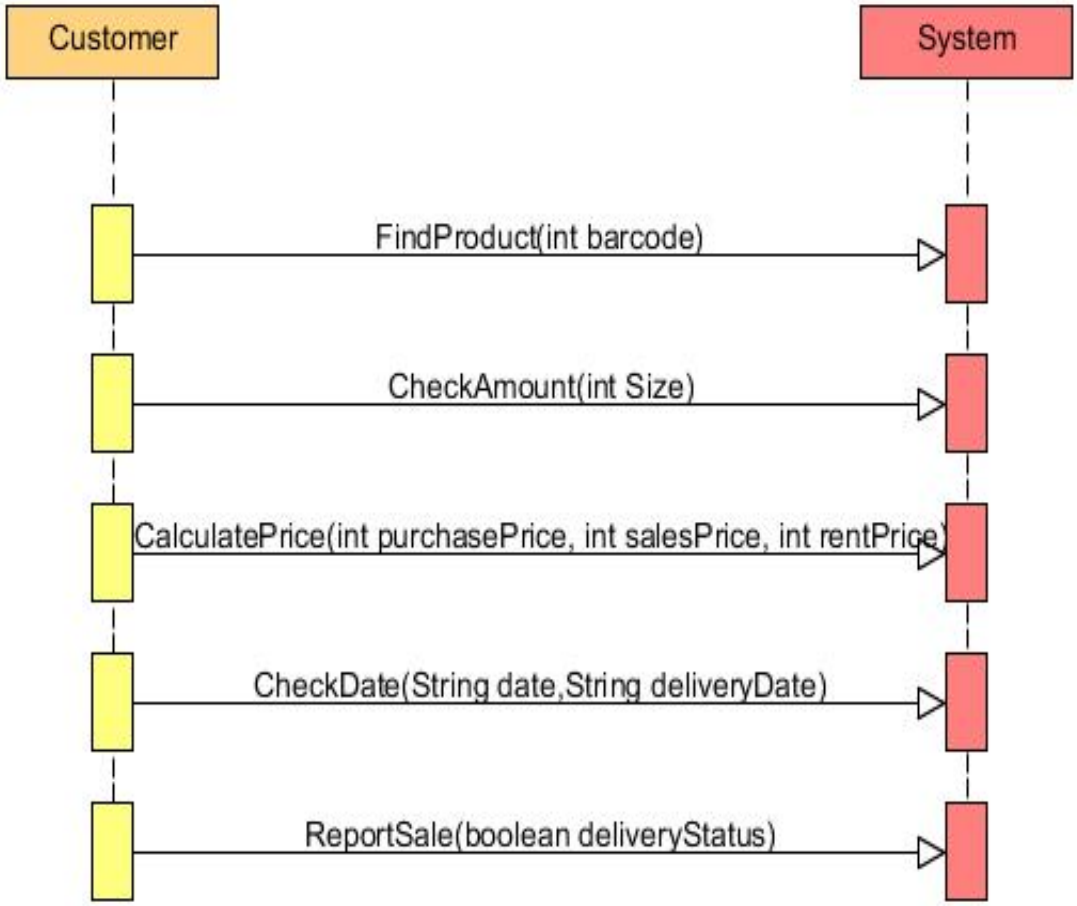
V. Fully Dressed Use-Cases

Use case: ***Product is sold***
Fully dressed use case

Use case name	Product is sold	
Actors	Employee and Customer	
Pre-conditions	Product are available	
Post-conditions	Product is sold and is not anymore in stock	
Frequency	Every time it needs	
Main Success Scenario (Flow of events)	**Actor(Action)**	**System(Response)**
	1.Customer chooses a product and a quantity of it	-
	2.Employee types product's barcode	3.System finds the product
	4.Employee types the amount customer wants to buy	5.System calculates total price for current product
	6.Employee types the customer id	7.System finds customer and calculates total price
		8. The system get the current date automatically
	9.Employee finishes the sale	10.System reports that the sale is created
Alternative flows	2a. Employee types invalid barcode 3a. The system returns no products 6a. Employee types invalid customer's id 11a System reports that a sale is not created	

Fully dressed table about the product is sold. The main actors here are employees and customer. Pre-conditions are available. Post-conditions are when every time it needs.

VI. System sequence diagram



*System sequence diagram about the product is sold. The main actors here are employees and customer. And the main methods are **FindProduct**, **CheckAmount**, **CalculatePrice**, **CheckDate** and **ReportSale**.*

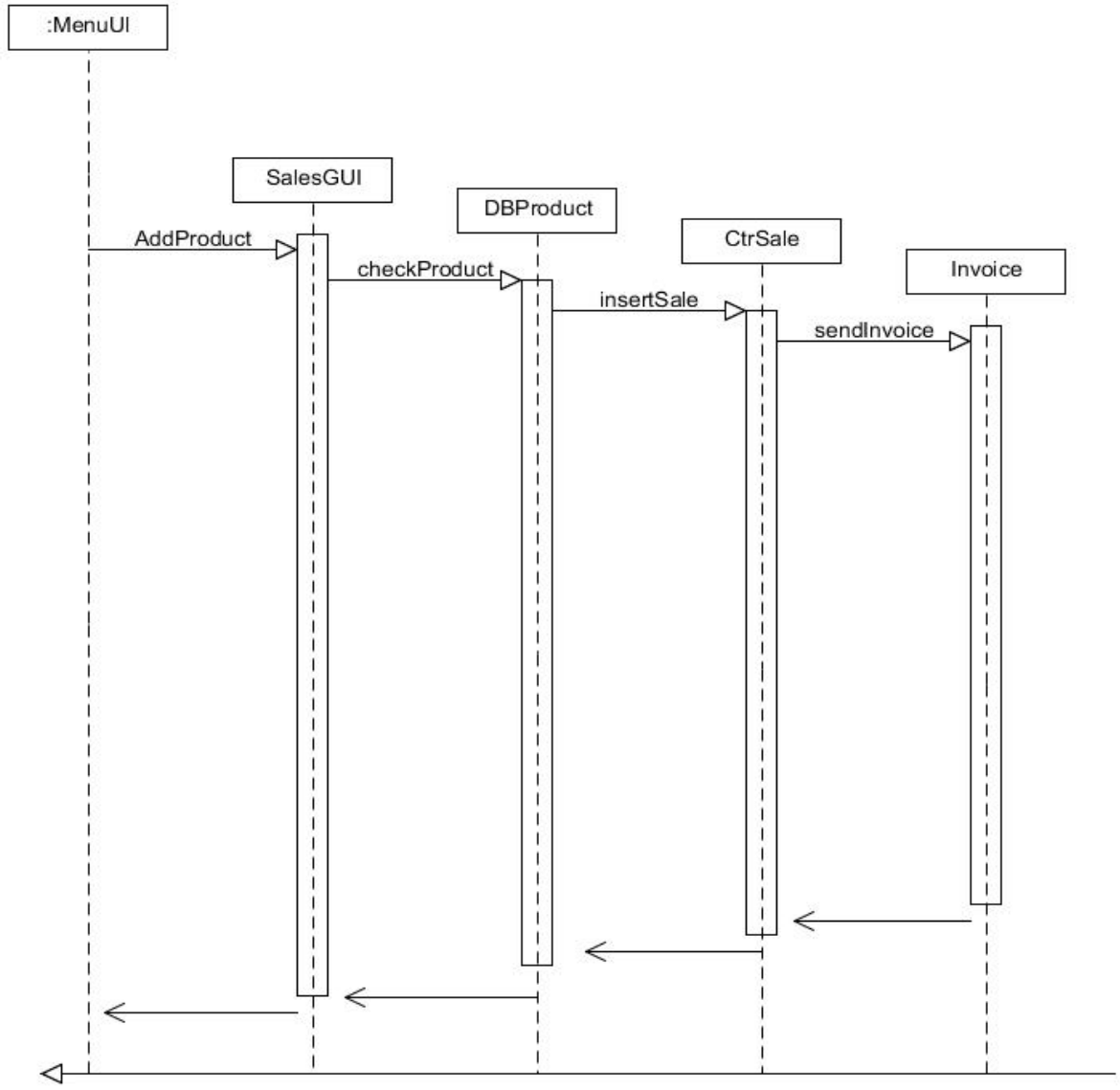
VII. Operation contracts:

Operation: **FindProduct** Use case: Product is sold Pre-condition: Product is in stock Post-condition: Product is found by the system Product is sold and is not anymore in stock
Operation: **CalculatePrice** Use case: Product is sold Pre-condition: Product is in stock Post-condition: Product is found by the system Total price is calculated Product is sold and is not anymore in stock
Operation: **ReportSale** Use case: Product is sold Pre-condition: Product is in stock Post-condition: Product is reported by the system that it crated sale Product is sold and is not anymore in stock

Operation: **CheckAmount** Use case: Product is sold Pre-condition: Product is in stock Post-condition: Amount is check by the system if it is available or not Product is sold and is not anymore in stock
Operation: **CheckDate** Use case: Product is sold Pre-condition: Product is in stock Post-condition: Date is checked by the system is it valid or not Product is sold and is not anymore in stock

*Operation contract about the product is sold. The are four main operation which are **FindProduct**, **CheckAmount**, **CalulatePrice**, **CheckDate** and **ReportSale**.*

VIII. Interaction Diagram



Interaction diagrams are used to visualize the interactive behavior of the system. Now visualizing interaction is a complex task. Our solution is to use different types of models to capture the different aspects of the interaction. In this interaction diagram, we have shown our sale. First we go to our ***menuUI*** *, afterwards we are adding a* ***product****, after we are done adding the product, the* ***Sales*** *class checks the product in the* ***DBproduct*** *class, we are inserting sale from* ***CtrSale****, and* ***sending invoice****, after this step is completed, we can say that everything is done.*

IX. Transformation. Domain model to Relational Model

Customer

Id	Name	Address	Zipcode	City	Phone
1	Jhon	Lulin 8	1000	Sofia	0878110930
2	Sara	Mladost 1	1000	Sofia	0873264591

*We have the following fields for the customers. **ID** that is a primary key and int, **Name** and **Adress** that are varchar, **Zipcode** that is int, also we have **City** and **Phone** that are varchar.*

Sale

Id	Date	customerId	totalPrice
1	02.03.2015	2	150.00
2	03.15.2015	1	1000.0

*In this table we have **ID** that is again a primary key, **Date** that is type date, **customerID** that is int, and **totalPrice** which is double.*

Invoice

saleId	saleDate	invoiceNo
2	02.12.2015	15
1	03.13.2015	16

*In the Invoice table we have the following fields. **saleID** that is int, **saleDate** is type date as in Sale, **invoiceNo** is type int.*

partSale

saleId	productBarcode	Amount	productName	pricePerPiece	price
1	15215	2	Hats	20	40
2	15216	20	Guns	10	60

***Many to many** relation between **Sale** and **Product**, with fields that are **saleID** that is from type int, **productBarcode** from type int and **amout** that is from type int, **productName** is varchar, **pricePerPiece** and price are int.*

Product

Barcode	Name	purchasePrice	Sale Price	rent Price	countryOf Origin	min Stock	size	supplierId	Colou
15215	Hat	10	20	5	Bulgaria	10	5	2	Red
15216	Mask	5	15	10	Bulgaria	20	10	1	Blue

*In Product we have the following fields, **barcode** that is from type int, **name** that is varchar, **purchasePrice**, **salePrice**, **rentPrice** are from type int, **countryOfOrigin** is varchar, **minStock** size and **supplierId** are int, **color** is varchar.*

Supplier

Id	Name	Address	Country	Phoneno	Email
1	JumboStore	Luluin 8	Bulgaria	088152612	jumbo@abv.bg
2	FunZone	Mladost	Bulgaria	084651354	funzone@abv.bg

*In this table we have **ID** that is primary key and int, **name**, **address** and **country** are varchar , **phoneno** and **email** are varchar.*

X. SQL Scripts for Creation

```
CREATE TABLE Customer(
        id int NOT NULL,
        name varchar(50) NOT NULL,
        address varchar(50) NOT NULL,
        zipcode int NOT NULL,
        city varchar(50) NOT NULL,
        phone varchar(50) NOT NULL,
 CONSTRAINT PK_Customer PRIMARY KEY (id)
);



CREATE TABLE Invoice(
        saleId int NOT NULL,
        saleDate date NOT NULL,
        invoiceNo int NOT NULL
);


CREATE TABLE partSale(
        saleId int NOT NULL,
        productBarcode int NOT NULL,
        productName varchar(50) NOT NULL,
        pricePerPiece float NOT NULL,
        amount int NOT NULL,
        price float NOT NULL
);
```

XI. SQL Scripts for Insertion

INSERT Customer (id, name, address, zipcode, city, phone) VALUES (0, 'unknown', 'none', 0, 'none', '0')

INSERT Customer (id, name, address, zipcode, city, phone) VALUES (1, 'Ivan', 'Havekrogen', 9000, 'Aalborg', '123123')

INSERT Invoice (saleId, saleDate, invoiceNo) VALUES (1, CAST('2015-03-21' AS Date), 1)

INSERT Invoice (saleId, saleDate, invoiceNo) VALUES (2, CAST('2015-03-21' AS Date), 2)

INSERT Invoice (saleId, saleDate, invoiceNo) VALUES (3, CAST('2015-03-21' AS Date), 3)

INSERT Invoice (saleId, saleDate, invoiceNo) VALUES (4, CAST('2015-03-21' AS Date), 4)

INSERT partSale (saleId, productBarcode, productName, pricePerPiece, amount, price) VALUES (1, 1, '1', 1, 11, 1)

INSERT partSale (saleId, productBarcode, productName, pricePerPiece, amount, price) VALUES (2, 1, '1', 1, 1, 1)

INSERT partSale (saleId, productBarcode, productName, pricePerPiece, amount, price) VALUES (2, 2, '2', 2, 2, 2)

INSERT partSale (saleId, productBarcode, productName, pricePerPiece, amount, price) VALUES (3, 3, '3', 3, 333333, 3333333)

INSERT partSale (saleId, productBarcode, productName, pricePerPiece, amount, price) VALUES (4, 1, '1', 1, 1, 1)

INSERT partSale (saleId, productBarcode, productName, pricePerPiece, amount, price) VALUES (4, 1, '1', 1, 1, 1)

INSERT partSale (saleId, productBarcode, productName, pricePerPiece, amount, price) VALUES (4, 1, '1', 1, 1, 1)

INSERT partSale (saleId, productBarcode, productName, pricePerPiece, amount, price) VALUES (4, 1, '1', 1, 1, 1)

INSERT partSale (saleId, productBarcode, productName, pricePerPiece, amount, price) VALUES (4, 1, '1', 1, 1, 1)

INSERT partSale (saleId, productBarcode, productName, pricePerPiece, amount, price) VALUES (4, 1, '1', 1, 1, 1)

INSERT partSale (saleId, productBarcode, productName, pricePerPiece, amount, price) VALUES (4, 1, '1', 1, 1, 1)

INSERT partSale (saleId, productBarcode, productName, pricePerPiece, amount, price) VALUES (4, 1, '1', 1, 1, 1)

INSERT partSale (saleId, productBarcode, productName, pricePerPiece, amount, price) VALUES (4, 1, '1', 1, 1, 1)

INSERT partSale (saleId, productBarcode, productName, pricePerPiece, amount, price) VALUES (4, 1, '1', 1, 1, 1)

INSERT partSale (saleId, productBarcode, productName, pricePerPiece, amount, price) VALUES (4, 1, '1', 1, 1, 1)

INSERT partSale (saleId, productBarcode, productName, pricePerPiece, amount, price) VALUES (4, 1, '1', 1, 1, 1)

INSERT partSale (saleId, productBarcode, productName, pricePerPiece, amount, price) VALUES (4, 1, '1', 1, 1, 1)

INSERT Product (barcode, name, purchasePrice, salesPrice, rentPrice, countryOfOrigin, minStock, size, colour, type, description, fabric, calibre, supplierId) VALUES (1, '1', 1, 1, 1, '1', 1, '1', '11', '1', '1', '1', 1, 1)

INSERT Product (barcode, name, purchasePrice, salesPrice, rentPrice, countryOfOrigin, minStock, size, colour, type, description, fabric, calibre, supplierId) VALUES (2, '2', 2, 2, 2, '22', 22, '22', '22', '2', '2', '2', 2, 1)

INSERT Product (barcode, name, purchasePrice, salesPrice, rentPrice, countryOfOrigin, minStock, size, colour, type, description, fabric, calibre, supplierId) VALUES (3, '3', 3, 3, 3, '3', 3, '3', '3', '3', '3', '33', 3, 1)

INSERT Sale (id, date, customerId, totalPrice) VALUES (1, CAST('2015-03-21' AS Date), 1, 1)

INSERT Sale (id, date, customerId, totalPrice) VALUES (2, CAST('2015-03-21' AS Date), 1, 123)

INSERT Sale (id, date, customerId, totalPrice) VALUES (3, CAST('2015-03-21' AS Date), 1, 333333)

INSERT Sale (id, date, customerId, totalPrice) VALUES (4, CAST('2015-03-21' AS Date), 1, 1)

INSERT Supplier (id, name, address, country, phoneno, email) VALUES (1, 'Georgi', 'Zarka', 'Bulgaria', '798456', 'sho@gmai.com')

XII. PRIMARY KEYS AND CONSTRAINTS

```
ALTER TABLE Invoice  WITH CHECK ADD  CONSTRAINT FK_Invoice_Sale
FOREIGN KEY(saleId)

REFERENCES Sale (id)


ALTER TABLE Invoice CHECK CONSTRAINT FK_Invoice_Sale


ALTER TABLE partSale  WITH CHECK ADD  CONSTRAINT FK_partSale_Product
FOREIGN KEY(productBarcode)

REFERENCES Product (Barcode)


ALTER TABLE partSale CHECK CONSTRAINT FK_partSale_Product


ALTER TABLE partSale  WITH CHECK ADD  CONSTRAINT FK_partSale_Sale
FOREIGN KEY(saleId)

REFERENCES Sale (id)


ALTER TABLE partSale CHECK CONSTRAINT FK_partSale_Sale


ALTER TABLE Product  WITH CHECK ADD  CONSTRAINT FK_Product_Supplier
FOREIGN KEY(supplierId)

REFERENCES Supplier (id)


ALTER TABLE Product CHECK CONSTRAINT FK_Product_Supplier


ALTER TABLE Sale  WITH CHECK ADD  CONSTRAINT FK_Sale_Customer
FOREIGN KEY(customerId)

REFERENCES Customer (id)


ALTER TABLE Sale CHECK CONSTRAINT FK_Sale_Customer
```