Grocery Store App

Huminiuc Simona
University of “Gheorghe Asachi” Iasi
Faculty of Automatic Control and Computer Engineering

1. Introduction. App usage

The purpose of this application is to provide users with a platform to view and purchase products available in an online store. The application includes product listing functionalities, checkout and payment process, as well as an administration interface for product management.

In the application there are two types of users, each with different roles, who can do certain actions on the database:

- Admin
- Buyer

a. Admin

The user with the name Admin has the possibility of several actions as follows: visualisation/addition/addition/deletion/modification of elements in the database tables such as for example, adding a new product category, changing the price of a product, deleting a supplier or visualise the transactions made by the buyer.

b. Buyer

The user with the name Buyer is the customer who can "buy" products that are available in the shop. The customer can purchase products, see the total price of the purchases and finalise the purchase by selecting the payment method. The customer cannot visualise other tables in the database other than the one with products, he cannot add a product, but he can modify indirectly the quantity after it has been purchased.

General functionalities:

- Stock products management: add/remove/modify products, detailed display
- Sale and transaction management: customers can add products to shopping cart,
update stocks, manage all transactions
- Inventory management: visualisation of product stocks, quantity sold
- Manage product categories and suppliers:
Adding/deleting/modifying/viewing product categories and supplier information
- Managing payment methods: add/update payment method

2. Technologies used

a. Front-end: HTML and CSS

Html (Hyper Text Style Mark-up Language) is a language used for creating and structuring web pages. In this application this language has been used to create a simple and user-friendly user interface. Several such "pages" have been created for almost every action we want to perform on the database. For example in the Admin interface there are different buttons that redirect you to different pages to view a particular table in the database.

CSS (Cascading Style Sheets) is a stylization language used to manage the way elements look on web pages. The main function of CSS is to separate the HTML content from the visual layout of the page. This allows developers to create styles such as fonts, colours, borders, sizes and layouts. It provides an efficient way to manage and continuously update the layout of a website. CSS uses selectors to identify HTML elements and then applies appropriate style rules to those elements. It is vital to web development and is often used in conjunction with HTML and JavaScript to create interactive and engaging web experiences.

b. Back-end: Python(Flask)

Python is a versatile programming language in which you can develop a multitude of applications, most often used for the logic behind it, offering multiple functionalities.

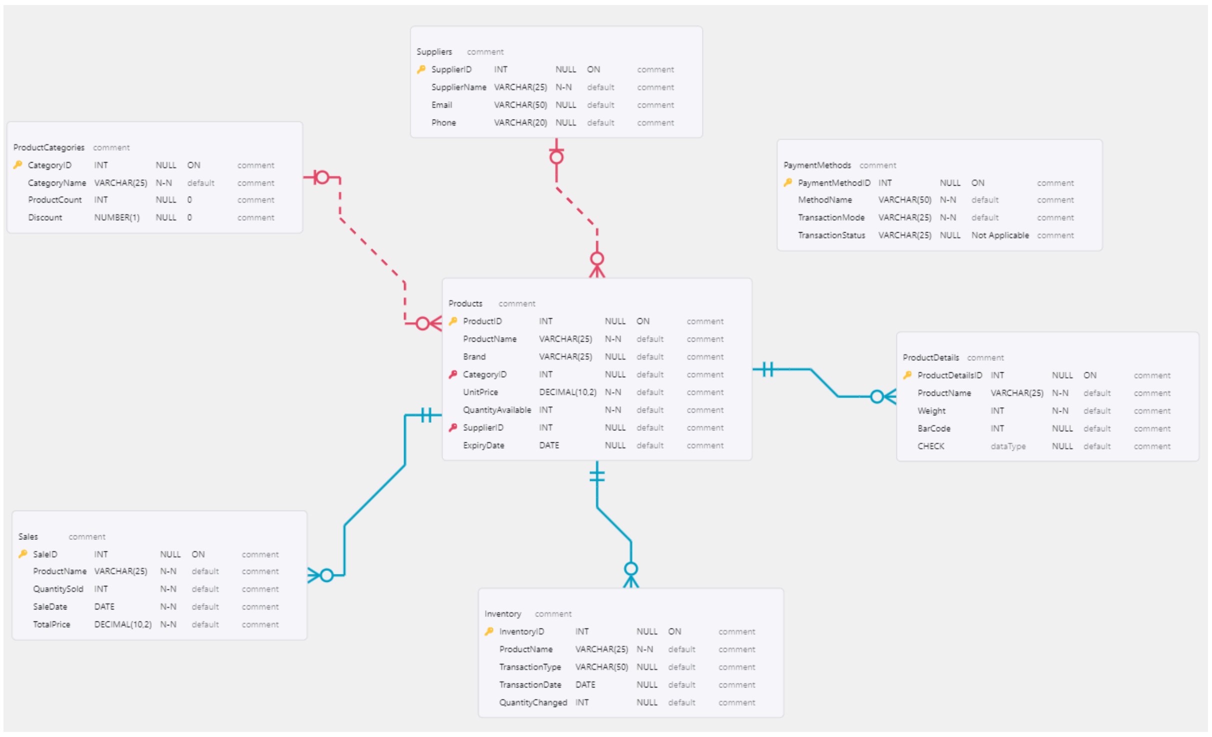
Flask is a web framework for Python that facilitates the creation of web applications, handling various interactions between a database, URL routing, HTTP request handling, etc.

c. Data base: Oracle SQL

Oracle SQL is a relational database management system used to manage data and database operations. Oracle provides powerful support for transactions, performance and scalability. The database was used to enter tables with information about different store areas. These are linked together by certain keys and can be modified according to the user's wishes.

These technologies together allow the user to create interactive and functional web applications, enabling data manipulation and storage, managing user interactions and presenting information in an accessible way.

3. Structure and Inter-relation of tables



Suppliers	comment			
SupplierID	INT	NULL	ON	comment
SupplierName	VARCHAR(25)	N-N	default	comment
Email	VARCHAR(50)	NULL	default	comment
Phone	VARCHAR(20)	NULL	default	comment

ProductCategories	comment			
CategoryID	INT	NULL	ON	comment
CategoryName	VARCHAR(25)	N-N	default	comment
ProductCount	INT	NULL	0	comment
Discount	NUMBER(1)	NULL	0	comment

PaymentMethods	comment			
PaymentMethodID	INT	NULL	ON	comment
MethodName	VARCHAR(50)	N-N	default	comment
TransactionMode	VARCHAR(25)	N-N	default	comment
TransactionStatus	VARCHAR(25)	NULL	Not Applicable	comment

Products	comment			
ProductID	INT	NULL	ON	comment
ProductName	VARCHAR(25)	N-N	default	comment
Brand	VARCHAR(25)	NULL	default	comment
CategoryID	INT	NULL	default	comment
UnitPrice	DECIMAL(10,2)	N-N	default	comment
QuantityAvailable	INT	N-N	default	comment
SupplierID	INT	NULL	default	comment
ExpiryDate	DATE	NULL	default	comment

ProductDetails	comment			
ProductDetailsID	INT	NULL	ON	comment
ProductName	VARCHAR(25)	N-N	default	comment
Weight	INT	N-N	default	comment
BarCode	INT	NULL	default	comment
CHECK	dataType	NULL	default	comment

Sales	comment			
SaleID	INT	NULL	ON	comment
ProductName	VARCHAR(25)	N-N	default	comment
QuantitySold	INT	N-N	default	comment
SaleDate	DATE	N-N	default	comment
TotalPrice	DECIMAL(10,2)	N-N	default	comment

Inventory	comment			
InventoryID	INT	NULL	ON	comment
ProductName	VARCHAR(25)	N-N	default	comment
TransactionType	VARCHAR(50)	NULL	default	comment
TransactionDate	DATE	NULL	default	comment
QuantityChanged	INT	NULL	default	comment

Fig 1. ER Diagram

Products and ProductsCategories

Relationship type: one-to-many (1:N)

Each product belongs to only one product category, but a product category can have several products. This relationship is established by the CategoryName in the Products table, which refers to the CategoryName in the ProductCategories table.

Products and Suppliers

Relationship type: one-to-many (1:N)

Each product has only one supplier, while a supplier may provide multiple products. This relationship is established by the SupplierName in the Products table, which references the SupplierName in the Suppliers table.

Products and Sales

Relationship type: one-to-many (1:N)

Each sale involves one product, and one product can be associated with multiple sales. This relationship is established by the ProductName in the Sales table, which references the ProductName in the Products table.

Products and Inventory

Relationship type: one-to-many (1:N)

Each product can have multiple inventory records, while each belongs to a specific product. This relationship is established by the ProductName in the Inventory table, which refers to the ProductName in the Products table.

ProductsCategories and Sales (relationship inferred by Products)
Relationship type: one-to-many (1:N)
Indirectly, through the CategoryName field in the Products table, there is an association between the categories of products and sales. However, this relationship is mediated by the Products table.

Products and ProductDetails
Relationship type: one-to-one (1:1)
Each product may have specific details stored in the ProductDetails table. This relationship is established through the ProductName, acting as a common key between the Products and ProductDetails tables.

Normalise

1. Eliminate redundancy: check for duplicate data that can be split into separate tables
2. Identification of primary and foreign keys: defining and using primary keys to uniquely identify each record and define foreign keys to establish relationships between different tables.
3. Creating relationships: ensuring that tables are correctly related to each other through foreign keys.
4. Checking normal forms: analysing whether tables fulfil the requirements of the normal
5. Optimise table structure: analyse and adjust the table structure in order to maximise database efficiency.

4. Description of constraints used

Products:
Primary Key: ProductID
Foreign Keys: CategoryName, SupplierName
Unique Constraint: ProductName
Check Constraint: QuantityAvailable (needs to be positive)
Not NULL Constraint: Productname, UnitPrice, QunatityAvailable

ProductCategories:
Primary Key: CategoryID
Not NULLConstraint : CategoryName

Suppliers:
Primary Key: SupplierID
Not NULL Constraint: SupplierName

Sales:
Primary Key: SaleID
Foreign Keys: ProductName Check Constraint: QuantitySold (needs to be positive)
Not NULL Constraint: ProductID, QuantitySold, SaleDate, TotalPrice

Inventory:
Primary Key: InventoryID
Foreign Keys: ProductName

PaymentMethods:
Primary Key: PaymentMethodID
Not NULL Constraint: MethodName, TransactionMode, TransactionStatus

ProductDetails:
Primary Key: ProductDetails
Foreign Keys: ProductName Check Constraint: BarCode (needs to be 12 digits), Weight (needs to be positive)
Not NULL Constraint: Weight