Grocery Store Application

Proiect la disciplina

Baze de date

Student: Huminiuc Simona

Grupa: 1309A

Cadru didactic Indrumator: Catalin Mironeanu

1. Introducere. Scopul aplicatiei

Scopul acestei aplicatii este sa ofere utilizatorilor o platforma pentru a vedea si a cumpara produse disponibile in magazinul online. Aplicatia include functionalitati de listare a produselor, proces de checkout si plata, precum si o interfata de administrare pentru gestionarea produselor.

In aplicatie exista doua tipuri de utilizatori, fiecare avand diferite roluri, care pot face anumite actiuni asupara bazei de date:

- Admin
- Buyer

1. Admin

Utilizatorul cu numele Admin beneficiaza de posibilitatea mai multor actiuni dupa cum urmeaza: vizualizare/adaugare/stergere/modificare a elementelor din tabelele din baza de date spre exemplu adaugarea unei noi categorii de produse, modificarea pretului unui produs, stergerea unui furnizor sau vizualizarea tranzactiilor efectuate de cumparator.

2. Buyer

Utilizatorul cu numele de Buyer este clientul care poate "cumpara" produse din cele care sunt disponibile in magazin. Clientul poate achizitiona produse, poate vedea pretul total al cumparaturilor si poate finaliza achizitia selectand metoda de plata. Clientul nu poate vizualiza alte tabele din baza de date decat cel cu produse, nu poate adauga un produs, dar il poate modifica indirect cantitatea dupa ce acesta a fost cumparat.

Funtionalitati generale:

- Administrare produse din stoc: adaugare/stergere/modificare produse, afisare detaliata
- Vanzare si gestionarea tranzactiilor: clientii pot adauga in cosul de cumparaturi produse, actualizare stocuri, gestionarea tuturor tranzactiilor
- Administrarea unui inventar: vizualizarea stocurilor de produse, cantitatea vanduta
- Administrarea categoriilor de produse si furnizori: adaugare/stergere/modificare/vizualizare categorii de produse si informatiile furnizorilor
- Gestionarea metodelor de plata: adaugare/actualizare metoda de plata

2. Tehnologii folosite

1. Front-end: HTML si CSS

Html (Hyper Text Style Mark-up Language) este un limbaj utilizat pentru crearea si structurarea paginilor web. In aceast aplicatie a fost utilizat acest limbaj pentru a creea o interfata simpla si prietenoasa cu utilizatorul. S-au creat mai multe "pagini" de acest fel pentru aproape fiecare actiune pe care dorim sa o executam asupra bazei de date. Spre exemplu in interfata Admin

exista diferite butoane care te redirectioneaza pe diferite pagini pentru a vizualiza un anumit tabel din baza de date.

CSS (Cascading Style Sheets) este un limbaj de stilizare folosit pentru a gestiona modul în care elementele arata pe paginile o web. Principala functie a CSS este de a separa continutul HTML de aspectul vizual al paginii. Acest lucru le permite dezvoltatorilor sa creeze stiluri precum fonturi, culori, margini, dimensiuni și layout-uri. Acesta ofera o modalitate eficienta de a gestiona si actualiza continuu aspectul unui site web. CSS utilizeaza selectori pentru a identifica elementele HTML si apoi aplica regulile de stil adecvate pentru aceste elemente. Este vital pentru dezvoltarea web contemporana si este adesea folosit împreună cu HTML si JavaScript pentru a crea experiente web interactive si captivante.

2. Back-end: Python (Flask)

Python este un limbaj de programare versatil in care se pot dezvolta o multitudine de aplicatii, utilizat cel mai des pentru logica din spatele lui, oferind functionalitati multiple.

Flask este un framework web pentru Python care faciliteaza crearea aplicatiilor web, gestionand diferite interactiuni dintre o baza de date, rutele URL, gestionarea cererilor HTTP etc.

3. Baza de date: Oracle SQL

Oracle SQL este un sistem de gestionare a bazelor de date relationale, utilizat pentru a gestiona datele si operatiile cu baza de date. Oracle ofera suport puternic pentru tranzactii, performanta si scalabilitate. Baza de date s-a utilizat pentru introducerea unor tabele cu informatii despre diferite arii ale magazinului. Acestea se leaga intre ele prin anumite chei si pot fi modificate in functie de dorintele utilizatorului.

Aceste tehnologii impreuna permit utilizatorulor sa creeze aplicatii web interactive si functionale, permitand manipularea si stocarea datelor, gestionand interactiuni utilizatorilor si prezentarea informatiilor intr-un mod accesibil.

3. Structura si Inter-Relationarea tabelelor

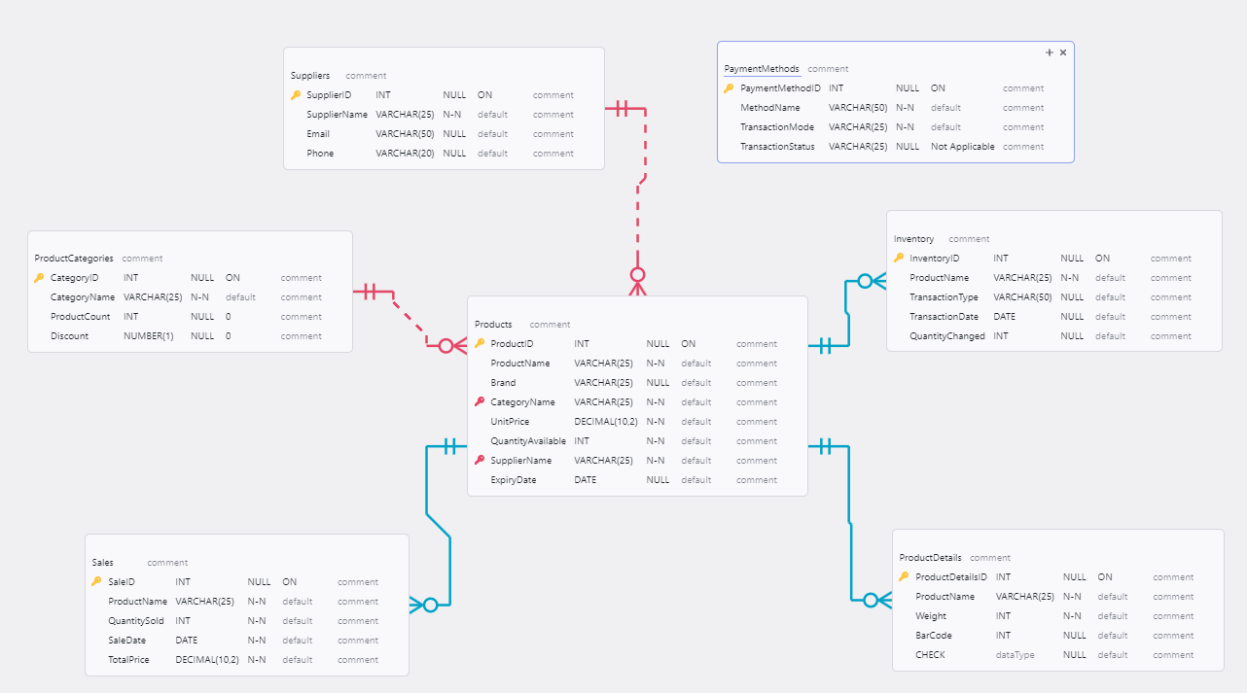

Fig. 3.1 Diagram ER

Products si ProductsCategories

Tip de relatie: unu-la-mai multi (1:N)

Fiecare produs apartine unei singure categorii de produse, dar o categorie de produse poate avea mai multe produse. Aceasta relatie este stabilita prin CategoryName din tabelul Products, care face referire la CategoryNamedin tabelul ProductCategories.

Products si Suppliers

Tip de relatie: unu-la-mai multi (1:N)

Fiecare produs are un singur furnizor, in timp ce un furnizor poate furniza mai multe produse. Aceasta relatie este stabilita prin SupplierName din tabelul Products, care face referire la SupplierName din tabelul Suppliers.

Products si Sales

Tip de relatie: unu-la-mai multi (1:N)

Fiecare vanzare implica un produs, iar un produs poate fi asociat cu vanzari multiple. Aceasta relatie este stabilita prin ProductName din tabelul Sales, care face referire la ProductName din tabelul Products.

Products si Inventory

Tip de relatie: unu-la-mai multi (1:N)

Fiecare produs poate avea mai multe inregistrari de inventar, in timp ce fiecare inregistrare de inventar apartine unui anumit produs. Aceasta relatie este stabilita prin ProductName din tabelul Inventory, care face referire la ProductName din tabelul Products.

ProductsCategories si Sales (relatie dedusa prin Products)
Tip de relatie: unu-la-mai multi (1:N)

Indirect, prin campul CategoryName din tabelul Products, exista o asociere intre categoriile de produse si vanzari. Cu toate acestea, aceasta relatie este mediata de tabelul Produse.

Products si ProductDetails
Tip de relatie: unu-la-unu (1:1)

Fiecare produs poate avea detalii specifice stocate in tabelul ProductDetails. Aceasta relatie este stabilita prin ProductName, actionand ca o cheie comuna intre tabelele Products si ProductDetails.

Normalizre

1. **Eliminare redundantei:** verificarea daca exista date duplicate care pot fi impartite in tabele separate
2. **Identificarea cheilor primarte si straine:** definirea si utilizarea cheilor primare pentru identificarea unica a fiecarei inregistrari si definirea cheilor straine pentru a stabili relatii intre diferite tabele.
3. **Crearea relatiilor:** asigurarea ca tabelel sunt corect legate intre ele prin intermediul cheilor straine.
4. **Verificarea formelor normale:** analiza daca tabelele indeplinesc cerintele formelor normale
5. **Optimizarea structurii tabelelor:** analizarea si ajustarea structurii tabelelor pentru a maximiza eficienta bazei de date.

4. Descrierea constrangerilor folosite

Products:
Primary Key: ProductID
Foreign Keys: CategoryName, SupplierName
Constrangere Unique: ProductName
Constrangere Check: QuantityAvailable (cantitatea disponibila trebuie sa fie pozitiva)
Constrangere Not NULL: Productname, UnitPrice, QunatityAvailable

ProductCategories:
Primary Key: CategoryID
Constrangere Not NULL: CategoryName

Suppliers:
Primary Key: SupplierID
Constrangere Not NULL: SupplierName

Sales:

Primary Key: SaleID
Foreign Keys: ProductName
Constrangere Check: QuantitySold (cantitatea cumparata trebuie sa fie pozitiva)
Constrangere Not NULL: ProductID, QuantitySold, SaleDate, TotalPrice

Inventory:
Primary Key: InventoryID
Foreign Keys: ProductName

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

ProducDetails:
Primary Key: ProductDetalisID
Foreign Keys: ProductName
Constrangere Check: BarCode (codul de bare trebuie sa aiba exact 12 cifre), Weight (greutatea trebuie sa fie pozitiva)
Constrangere Not NULL: Weight

5. Descrierea modaliatatii de conectare la baza de date

Conectarea la baza de date s-a facut prin intermediul librariei numite cx_Oracle.

```
from flask import Flask, render_template, request, redirect
import cx_Oracle

app = Flask(__name__)


cx_Oracle.init_oracle_client(lib_dir=r"instantclient-basic-windows.x64-21.12.0.0.0dbru.zip")
connection = cx_Oracle.connect("bd093", "bd093", "bd-dc.cs.tuiasi.ro:1539/orcl")
```

In fiecare functie conceputa pentru acest proiect au fost utilizate si urmatoarele linii de cod pentru a extrage/introduce informatii din baza de date:

ID

```
connection.commit()

cursor.close()          cursor = connection.cursor()
```

6. Capturi de ecran

Interfata

```
        }
        select {
            width: 100%;
            padding: 10px;
            border: 1px solid #b99470;
            border-radius: 5px;
            box-sizing: border-box;
            margin-bottom: 20px;
            font-size: 16px;
            color: #783d19;
        }
            input[type="submit"] {
                background-color: #c4661f;
                color: #fefae0;
                border: none;
                padding: 12px 20px;
                border-radius: 5px;
                cursor: pointer;
                font-size: 16px;
                width: 100%;
                display: inline-block;
            }
            input[type="submit"]:hover {
                background-color: #783d19;
            }
        </style>
    </head>
<body>
    <h1>Modify Product</h1>
    <form action="/admin/modify_product" method="post">
        <label for="product_name">Product Name:</label>
        <select id="product_name" name="product_name">
            {% for product in products %}
                <option value="{{ product.ProductName }}">{{ product.ProductName }}</option>
            {% endfor %}
        </select><br>

        <label for="new_name">New Name:</label>
        <input type="text" id="new_name" name="new_name"><br>

        <label for="new_brand">New Brand:</label>
        <input type="text" id="new_brand" name="new_brand"><br>
```