Universitatea Tehnică "Gheorghe Asachi" din Iași
Facultatea de Automatică și Calculatoare
Specializarea Calculatoare și Tehnologia Informației

Proiect disciplina baze de date

Aplicație pentru gestiunea meniurilor, stocurilor și a comenzilor unui restaurant

Cadru didactic coordonator,
Cristian Buțincu

Student,
Cojocaru Constantin-Cosmin
Mălăncuș Horia
Grupa 1309A

Scopul proiectului:

Analiza și proiectarea unei aplicații care să modeleze activitatea unui restaurant cu privire la crearea unui meniu digital cat si posibilitatea înregistrării comenzilor de produse pe baza ingredientelor disponibile.

Descrierea aplicației:

De multe ori, întocmirea unui meniu de restaurant presupune de multe ori verificarea atentă a tuturor categoriilor, produselor și a ingredientelor componente. Astfel modificarea sau adăugarea ulterioară a produselor sau chiar crearea unui nou meniu devine o operație costisitoare în special de timp. În acest scop, folosirea unei aplicații care are în spate are o baza de date drept mecanism de stocare pentru crearea și gestionarea meniului ar fi soluție eficienta. Astfel, principalul avantaj este modificarea celor mai volatile elemente din meniu în timp util: numele preparatelor, meniului, prețul produselor, șamd... O alta problema a unui restaurant este primirea de comenzi ale unor preparate indisponibile din cauza lipsei de ingrediente disponibile în stoc. Verificarea continuă a ingredientelor pentru furnizarea informațiilor necesare referitoare la stocul lor este consumatoare de timp. Folosirea unei aplicații în situația de fata ar permite modificarea și verificarea stocului produselor în timp real.

Aplicația este concepută pentru a permite folosirea de către 2 categorii de utilizatori:

Administratori:

Aceștia sunt persoane privilegiate cu gestionarea a tot ce înseamnă meniuri, categorii, produse, ingrediente, rețetă, comenzi. Aplicația le permite introducerea, modificarea sau ștergerea acestora. De asemenea aceștia mai pot seta un meniu curent care sa fie folosit de către clienți.

Clienți:

Aceștia au acces doar la cea de-a doua parte a aplicației: meniul propriu-zis de unde pot face o comandă. Aceștia își pot selecta numărul de produse pe care doresc sa le comande în limita stocului disponibil. Clienții pot vizualiza comanda pe care urmează să o plaseze pe un panou pe care le sunt prezentate produsele cât și numărul comandat al fiecărui produs, numărul mesei la care se plasează comanda, cat si opțiunea plății cu cardul sau numerar.

Tehnologiile utilizate:

Întreaga aplicație a fost realizată în IDE-ul **NetBeans** în limbajul de programare Java. IDE-ul facilitează crearea interfeței grafice (GUI) printr-o alta interfața de creare a acesteia.

Front-end:

În realizarea interfeței grafice aferente proiectului pentru meniul digital al unui restaurant s-au utilizat bibliotecile **AWT** si **Swing** din cadrul Java. Astfel, pentru implementarea interfeței grafice au fost folosite numeroase elemente grafice: butoane (clasa JButton), panouri (clasa JPanel), spații pentru text (JTextField si JTextArea), cat și altele. Acestea au fost folosite pentru a crea un

mediu cat mai intuitiv, interactiv și mai ușor de utilizat, atât pentru client, cat si pentru administrator.

Baza de date:

Aplicația folosește o baza de date Oracle 11g, conectarea la aceasta făcându-se prin intermediul driver-ului **Oracle Database 12.1.0.1 JDBC Driver**.

Back-end:

Pentru dezvoltarea aplicației s-a folosit un API care este oferit de Oracle pentru accesarea și procesarea datelor stocate într-o bază de date, numit **java.sql**, utilizând limbajul de programare Java.

Informațiile necesare pentru modelare:

Administratori:

Administratorii sunt responsabili cu gestiunea meniurilor. Despre aceștia este necesar să se cunoască un id unic pentru fiecare, un nume și o parola care este codificata cu cifrul Caesar cu offset-ul 25.

Meniuri:

Crearea de meniuri este primul din cele doua obiective pe care și le propune acest proiect. Un meniu poate fi format din mai multe categorii. Acesta este identificat unic printr-un număr de ordine și va avea un nume unic în baza de date. Se vor înregistra de asemenea, o dată de creare cât și detalii suplimentare opționale pentru descrierea acestora. Meniul poate conține una sau mai multe categorii.

Categorii:

Categoriile sunt identificate printr-un număr unic. Acestea vor avea un nume unic în cadrul fiecărui meniu. Categoriile vor conține o dată opțională de creare cât și detalii suplimentare opționale pentru descrierea acestora. Acestea vor conține produsele de care restaurantul dispune.

Produse:

Produsele sunt identificate printr-un număr unic. Acestea vor avea un nume unic. Produsele se împart în doua tipuri: preparate și băuturi. Fiecărui produs îi este asociat un nume unic, un preț, o dată opțională de creare cât și detalii suplimentare opționale pentru descrierea acestora. Produsele care se doresc sa nu mai fie folosite se vor înregistra ca fiind dezactivate. Acest lucru va permite păstrarea meniurilor, comenzilor și a rețetelor în caz de reactivarea ulterioară.

Produsele pentru care nu se folosesc ingrediente pentru prepararea lor (ex: băuturi sigilate, apa la sticla), vor avea un stoc propriu. Produsele pot fi formate din unul sau mai multe ingrediente.

Ingrediente:

Ingredientele au un nume și o firma producătoare, împreună formând o pereche unică în baza de date. Pentru fiecare ingredient se va tine cont de stocul acestuia. Produsele cat si ingredientele pot fi

de un anumit tip: post, peste, carne, lactat, alcoolice, non-alcoolice etc... Pentru produsele care sunt preparate pe baza ingredientelor, se va ține evidența rețetelor acestora. Acestea conțin ingredientele folosite și cantitatea folosită din ingredientul respectiv.

Comenzi:
Plasarea de comenzi pe baza ingredientelor disponibile sau stocului disponibil este cel de-al doilea obiectiv pe care și-l propune acest proiect. Comenzile sunt identificate printr-un id unic. Produsele comandate vor fi asociate unei comenzi, iar pentru fiecare produs se va specifica numărul de produse comandate. Comenzile conțin o dată de creare, opțional un număr al mesei cat si detalii suplimentare opționale pentru mai multe detalii.

Funcționalitatea aplicației:

Principalele funcții pe care le realizează aplicația:

- Crearea de meniuri care pot fi împărțite pe categorii, acestea conținând produsele care sunt vândute.
- Evidența stocului ingredientelor cât și al stocului produselor care nu folosesc ingrediente.
- Gestionarea comenzilor.

Tabelele din aceasta baza de date sunt:

- Administratori
- Meniuri
- Categorii
- categorii_produse
- Produse
- stoc_produs
- Ingrediente
- Rețete
- tipuri_aliment
- produse_comenzi
- Comenzi

Schema logica a baze de date

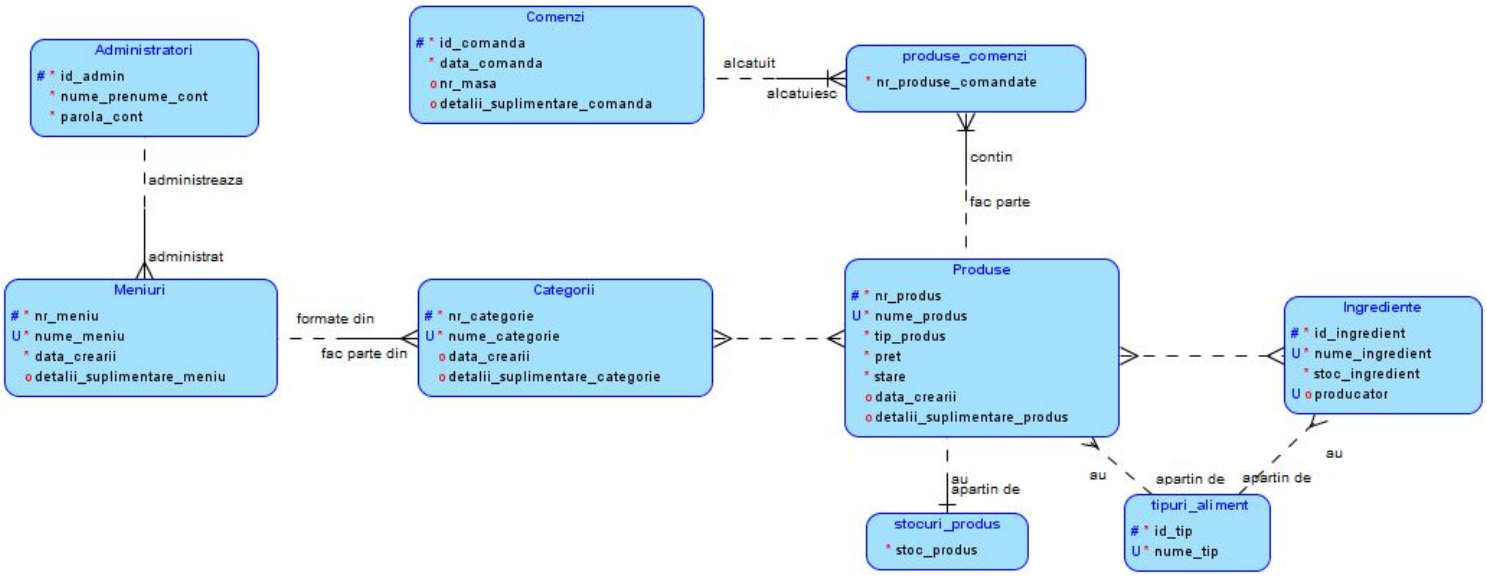


Fig. 1

Descrierea relațiilor între tabele:

În proiectarea acestei baze de date s-au identificat tipurile de relații **1:1**, **1:n**, **n:1**, **n:n**.

Între tabele **Administratori** și **Meniuri** se stabilește o relație **1:n** deoarece un administrator poate avea în gestiune mai multe meniuri, iar meniurile sunt gestionate doar de un singur administrator deoarece este suficient pentru problema pe care aplicația si-o propune.

Între **Meniuri** și **Categorii** se stabilește o relație **1:n** deoarece dorim ca un meniu să fie format din mai multe categorii, iar acestea să aparțină obligatoriu doar de un singur meniu. Apartenența de un singur meniu și nu de mai multe este data de problema modificării unei categorii în cazul în care acesta ar aparține de mai multe meniuri.

Între **Categorii** și **Produse** se stabilește o relație **n:n** deoarece fata de relația dintre Meniuri și Categorii, produsele pot să aparțină de mai multe categorii indiferent dacă rețeta, numele său prețul acestuia se modifică. Pentru a se ajunge la normalizarea 3NF, această relație se împarte în alte două relații **1:n** și **n:1**. Astfel se creează o nouă tabela numita **categorii_produse** în care se vor înregistra apartenența produselor la anumite categorii.

Între **Produse** și **stocuri_produs** se stabilește o relație **1:1** deoarece se dorește evidența stocurilor produselor care nu folosesc ingrediente in prepararea lor: apa la sticla, doză de suc șamd... Așadar în aceasta situație, un singur produs poate avea asociat un singur stoc, implicit doar un singur stoc este obligatoriu asociat doar unui produs.

Între **Produse** și **Ingrediente** se stabilește o relație **n:n** deoarece un produs poate fi preparat din unul sau mai multe ingrediente, iar un ingredient poate fi componenta în prepararea mai multor produse. Pentru a rezolva problema cantității ingredientelor necesare in prepararea produselor și pentru ca un atribut care să specifice aceasta cantitate ar crea probleme de normalizare dacă ar fi introdus în unul din aceste doua tabele, relația n:n se împarte în două relații **1:n** și **n:1**. Astfel se creează o nouă tabelă **Rețete** care va avea ca si atribut suplimentar, cantitatea necesara de ingrediente preparării produsului.

Relațiile **1:n** între **Produse** și **Rețete** și respectiv **n:1** între **Rețete** și **Ingrediente** rezultate după spargerea relației n:n asigura formă normală 3NF deoarece asocierea ingredientelor necesare preparării produselor este independentă de atributele tabelelor Produse și Ingrediente.

Între **Produse** și **tipuri_aliment**, respectiv între **Ingrediente** și **tipuri_aliment** se stabilesc câte o relație **n:1** deoarece produselor cat si ingredientelor li se pot asigna câte un tip de aliment specific. Unui produs sau unui ingredient i se poate asigna doar un singur tip din tabela tipuri_aliment, iar un tip poate fi asignat mai multor produse sau ingrediente.

Tabela **produse_comenzi** modelează nevoia cunoașterii numărului de produse comandate pentru fiecare produs. În aceasta tabela se înregistrează lista de cumpărături ale clientului. Între **Produse** și **produse_comenzi** se stabilește o relație **1:n** deoarece un produs poate face parte din mai multe liste de acest gen, iar o listă conține doar câte o singură instanță al unui produs per comanda. Între **Comenzi** și **produse_comenzi** se stabilește o relație **1:n** deoarece o comanda poate conține unul sau mai multe elemente din lista de produse, iar un element din aceasta lista trebuie să aparțină doar unei singure comenzi.

Constrângeri necesare folosite:

În proiectarea acestei baze de date s-au identificat tipurile de constrângeri **NOT NULL**, **UNIQUE**, **DEFAULT**, **CHECK**, **PRIMARY/FOREIGN KEY** și de tip **DOMAIN**.

NOT NULL: este folosit pentru atributele care trebuie să se cunoască neapărat valorile pentru a se asigura o funcționalitate corectă a bazei de date. Acest tip de constrângere s-a folosit pentru: numele administratorului, meniului, categoriilor, produselor, ingredientelor, deoarece acestea sunt necesare de știut la crearea instanțelor entităților respective. De asemenea s-a folosit și pentru stocurile ingredientelor, produselor, numărul de ingrediente folosit în rețetă cât și pentru prețul și numărul de produse comandate de client.

UNIQUE: este folosit pentru a restrânge adăugarea de elemente multiple care ar crește confuzia relațiilor dintre instanțele entităților. Acest tip de constrângere s-a folosit pentru a asigura un nume unic fiecărui meniul, fiecărei categorii împreuna cu numărul meniului pentru a asigura unicitatea numelui categoriei doar în cadrul fiecărui meniu. De asemenea s-a mai folosit acest tip de constrângere pentru numele și producătorul ingredientelor împreuna pentru a modela o diversitate mai mare de ingrediente care pot fi introduse în baza de date. Față de numele meniurilor,

categoriilor și a ingredientelor, numelui produsului nu ii se va atribuii o astfel de constrângere deoarece în baza de date pot exista mai multe preparate cu nume identic, dar cu rețete diferite.

DEFAULT: În acest proiect constrângerea de tip acest tip s-a folosit pentru introducerea automată a datelor de creare ale instanțelor entităților Meniuri, Categorii, Produse si Comenzi. Valoare înregistrată automat la creare este data curentă.

CHECK: este folosită pentru a limita posibilitatea valorilor introduse în anumite câmpuri din baza de date. Acest tip de constrângere s-a folosit cu două scopuri în baza de date: limitarea valorilor numelor introduse și asigurarea unei valori pozitive (>0) pentru:

- numărul mesei la care se face comanda,
- stocul ingredientelor,
- cantitatea ingredientelor folosite în rețetă,
- prețul produselor,
- stocul produselor,
- numărul produselor comandate;

Limitarea valorilor numelor introduse s-a realizat printr-o expresie regulată pentru:

- **numele și prenumele administratorilor:** [Nume] [Prenume]-[Alt prenume opțional] (doar litere),
- **parola conturilor:** orice șir de caractere care nu conține caractere albe,
- **numele meniurilor:** cuvinte sau numere, primul cuvânt începând cu majusculă, opțional delimitate prin '-',
- **numele categoriilor:** doar cuvinte, primul cuvânt începând cu majusculă, opțional delimitate prin '-', tipul alimentului: un cuvânt sau maxim 2 cuvinte delimitate prin '-',
- **numele produsului:** cuvinte sau numere (opțional și în virgula mobilă), primul cuvânt începând cu majusculă, opțional delimitate prin '-',
- **numele ingredientelor:** cuvinte fără majuscule;

PRIMARY/FOREIGN KEY: constrângerile de tip primary key sunt folosite pentru a asigura existenta unei instanțe unic identificabila în tabele. Constrângerile de tip foreign key sunt folosite pentru a crea relații între cheile primare ale altor tabele.

DOMAIN: sunt folosite pentru respectarea tipurilor de date ale atributelor atunci când sunt introduse valori în baza de date.

Modelul relațional

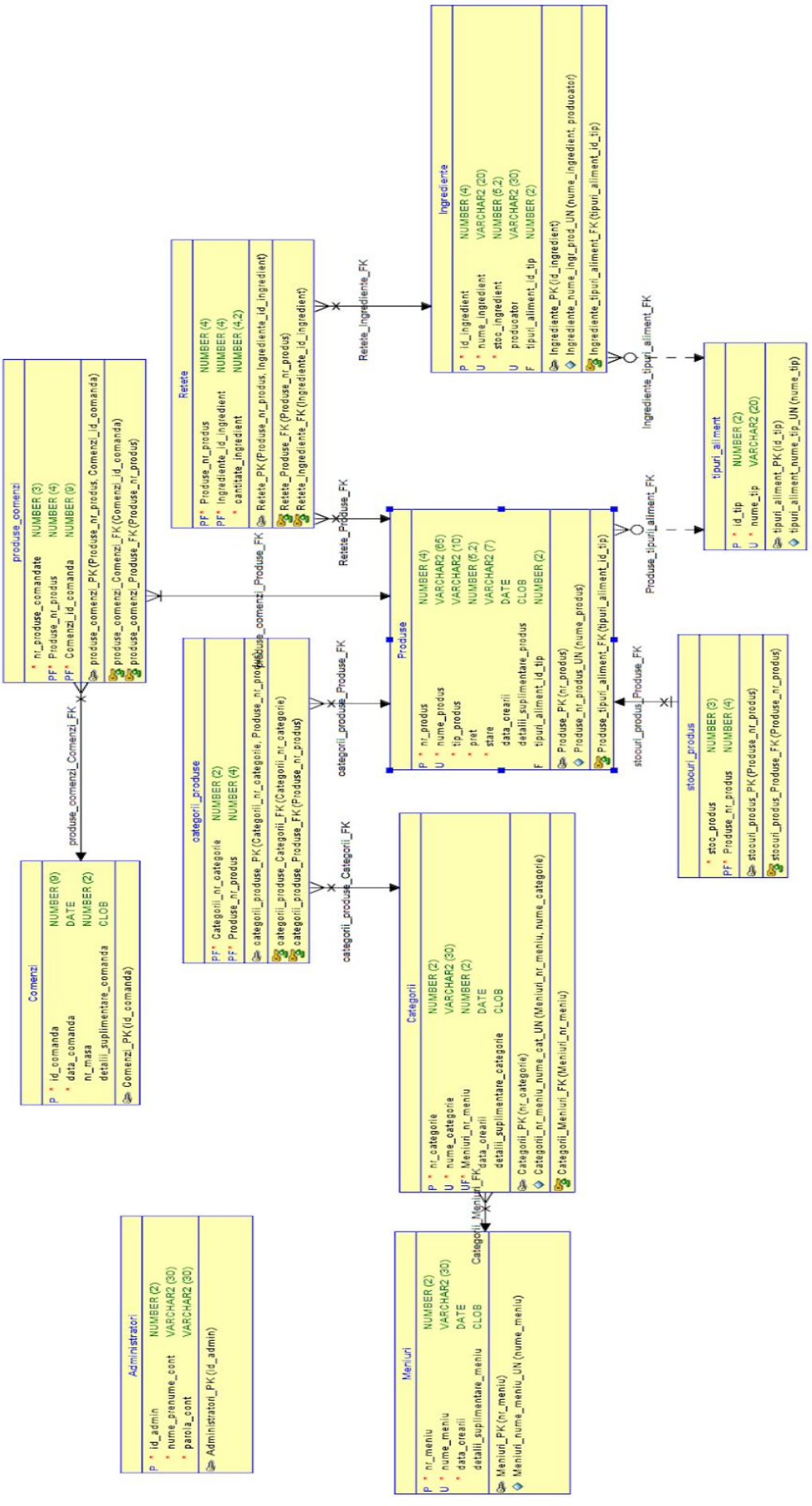
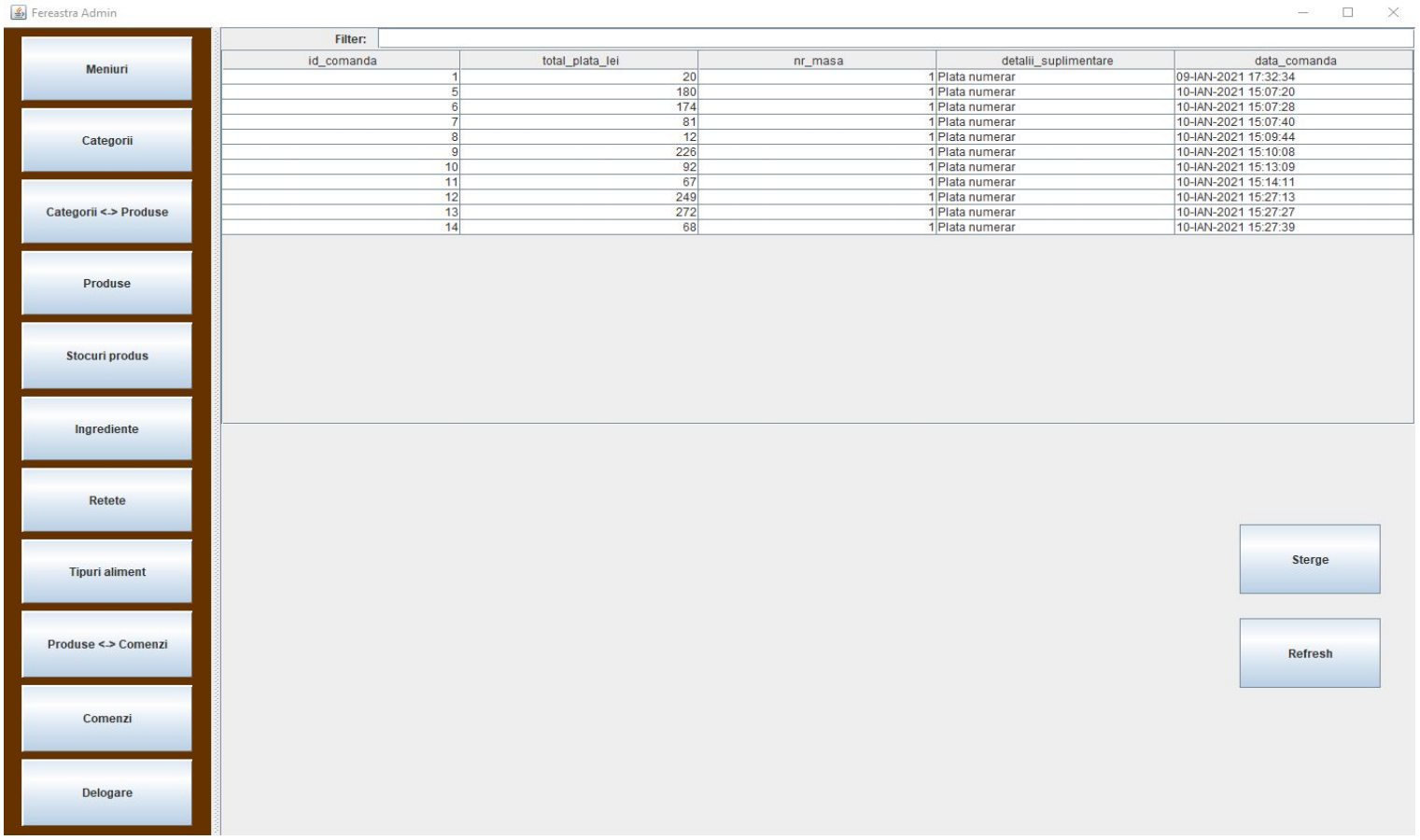


Fig. 2

Instrucțiuni SQL:

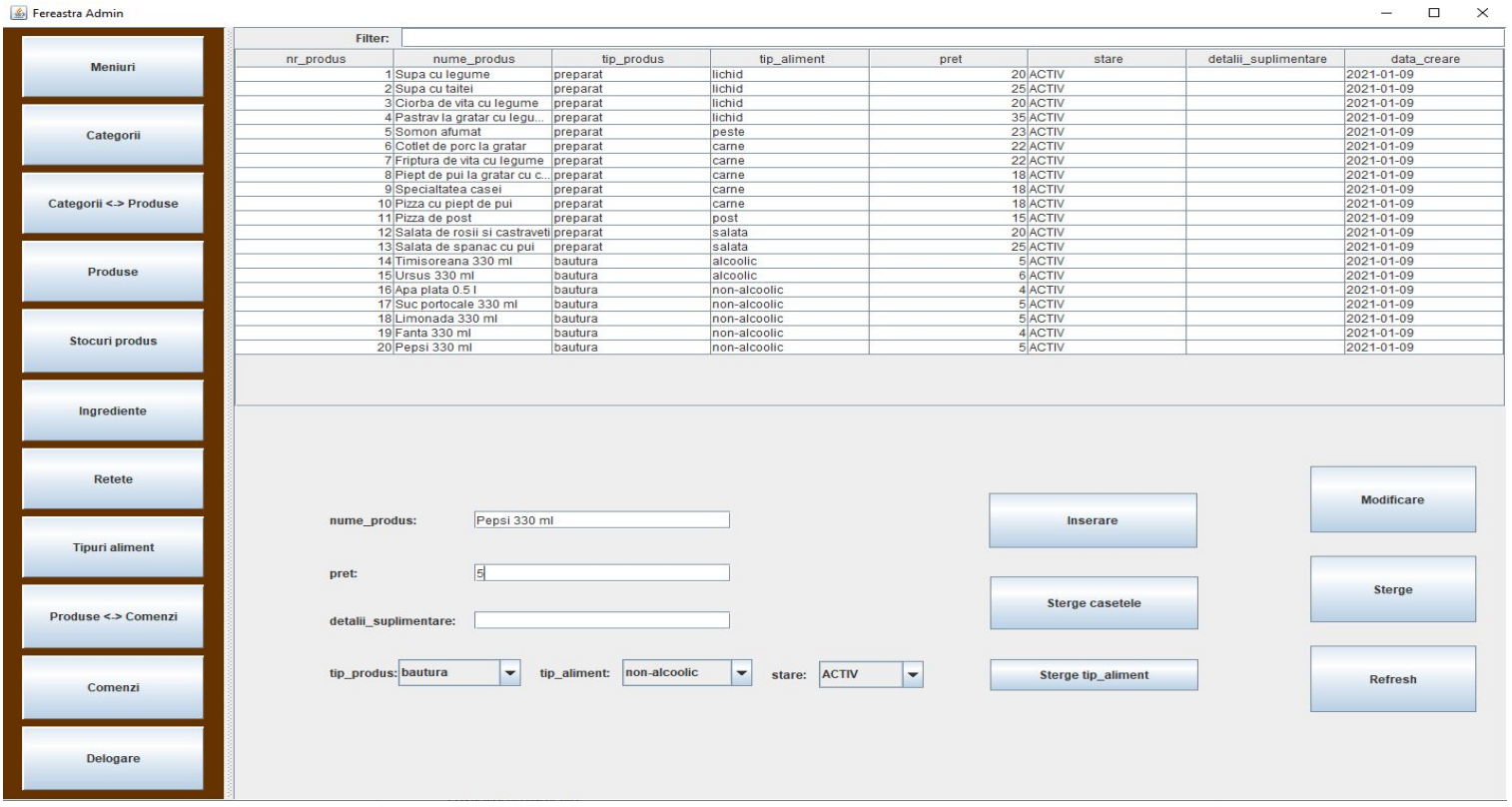
Instrucțiunea **SELECT** s-a folosit în special pentru obținerea informațiilor din baza de date. Spre exemplu, pentru extragerea informațiilor cu privire la una dintre comenzile date de către un client s-a utilizat exemplul de cod de mai jos:

```
ResultSet rs = appWindow.getDataBaseConnection().getConnection().createStatement().executeQuery(
        "SELECT c.id_comanda, SUM(p.pret * pc.nr_produse_comandate)
         + "as total_plata, c.nr_masa,   to_char(c.detalii_suplimentare_comanda ) as detalii_comanda, "
         + " (SELECT  to_char(c.data_comanda,'DD-MON-YYYY HH24:MI:SS') from dual) "
         + "as data_si_ora_comanda FROM Comenzi c, produse_comenzi pc, Produse p\n"
       + "WHERE c.id_comanda = pc.Comenzi_id_comanda and pc.Produse_nr_produs = p.nr_produs\n"
       + "GROUP BY c.id_comanda, c.nr_masa, to_char(c.detalii_suplimentare_comanda), c.data_comanda\n"
       + "ORDER BY c.id_comanda"
    );
```



Pentru introducerea de noi elemente în baza de date s-a utilizat instrucțiunea **INSERT**. Inserarea unui produs poate fi observată în exemplul următor:

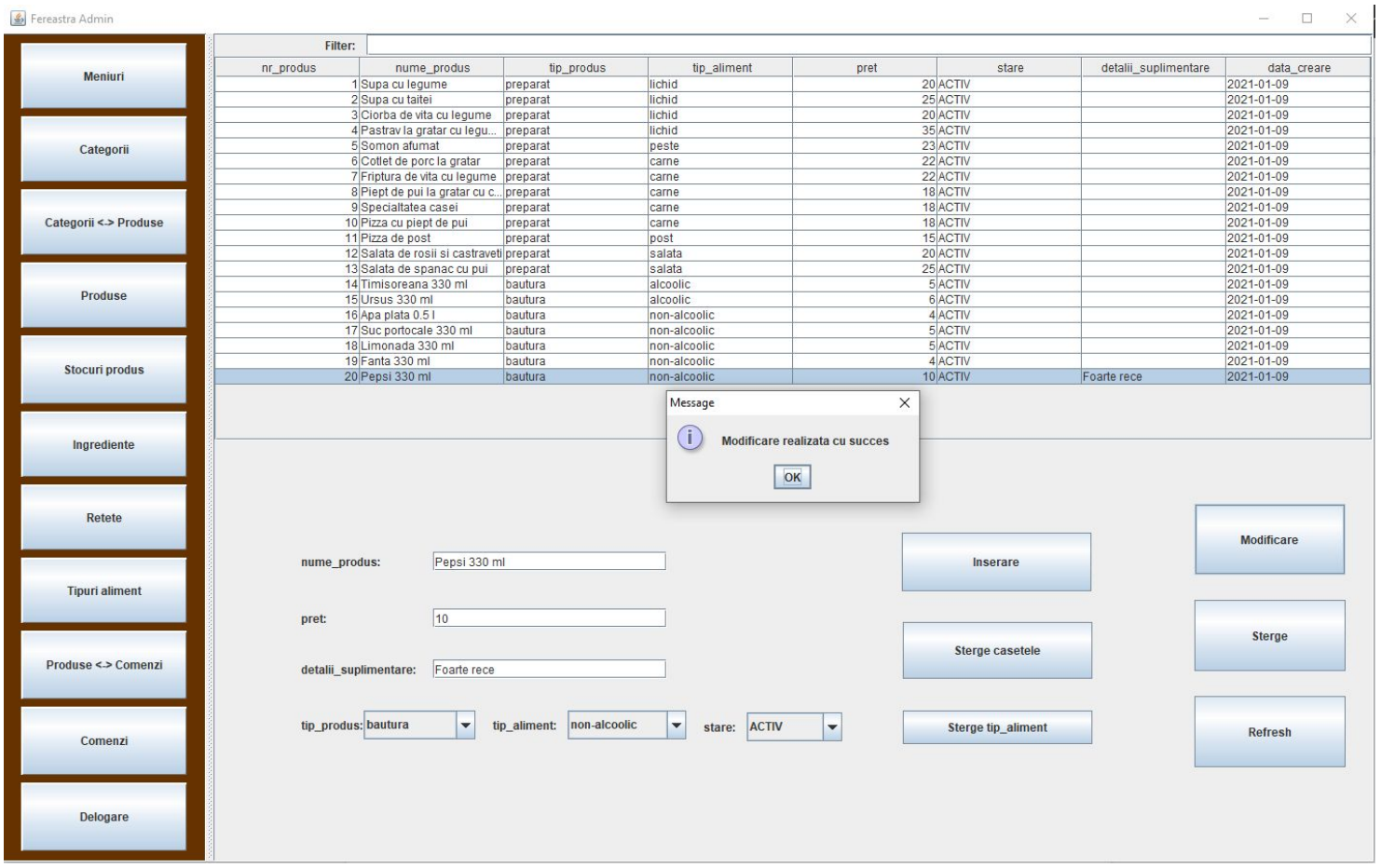
```
 nume_produs = numeProdusTextField.getText();
tip_produs = productTypeCB.getSelectedItem().toString();
tip_aliment = foodTypeCB.getSelectedItem().toString();
pret = priceTextField.getText();
stare = stateCB.getSelectedItem().toString();
detalii_suplimentare = detaliiSuplimentareTextField.getSelectedText();

 try {
  PreparedStatement prepSt = conn.prepareStatement("INSERT INTO Produse(nume_produs,  \n"
                  + "tip_produs, pret, tipuri_aliment_id_tip, stare, detalii_suplimentare_produs)  \n"
                  + "VALUES(?, ?, ?, (SELECT id_tip FROM  tipuri_aliment WHERE nume_tip = ?), ?, ?)"); "
  prepSt.setString(1, nume_produs);
  prepSt.setString(2, tip_produs);
  prepSt.setString(3, pret);
  prepSt.setString(4, tip_aliment);
  prepSt.setString(5, stare);
  prepSt.setString(6, detalii_suplimentare);
  prepSt.execute();

  ResultSet rs = conn.createStatement().executeQuery("SELECT MAX(nr_produs) FROM Produse");
  rs.next();
  nr_produs = rs.getString(1);

      Object tfData[] = {Short.parseShort(nr_produs), nume_produs, tip_produs, tip_aliment, Short.parseShort(pret),          stare,
detalii_suplimentare, currentDate};
  tblModel.addRow(tfData);

  conn.createStatement().execute("commit");
  JOptionPane.showMessageDialog(this, "Inserare realizata cu succes");
 } catch (SQLException ex) {
  JOptionPane.showMessageDialog(this, ex.getMessage());
 }
```



De asemenea, în cadrul gestionării datelor din tabele s-a utilizat și instrucțiunea **UPDATE**, pentru editarea unor linii deja existente în baza de date:

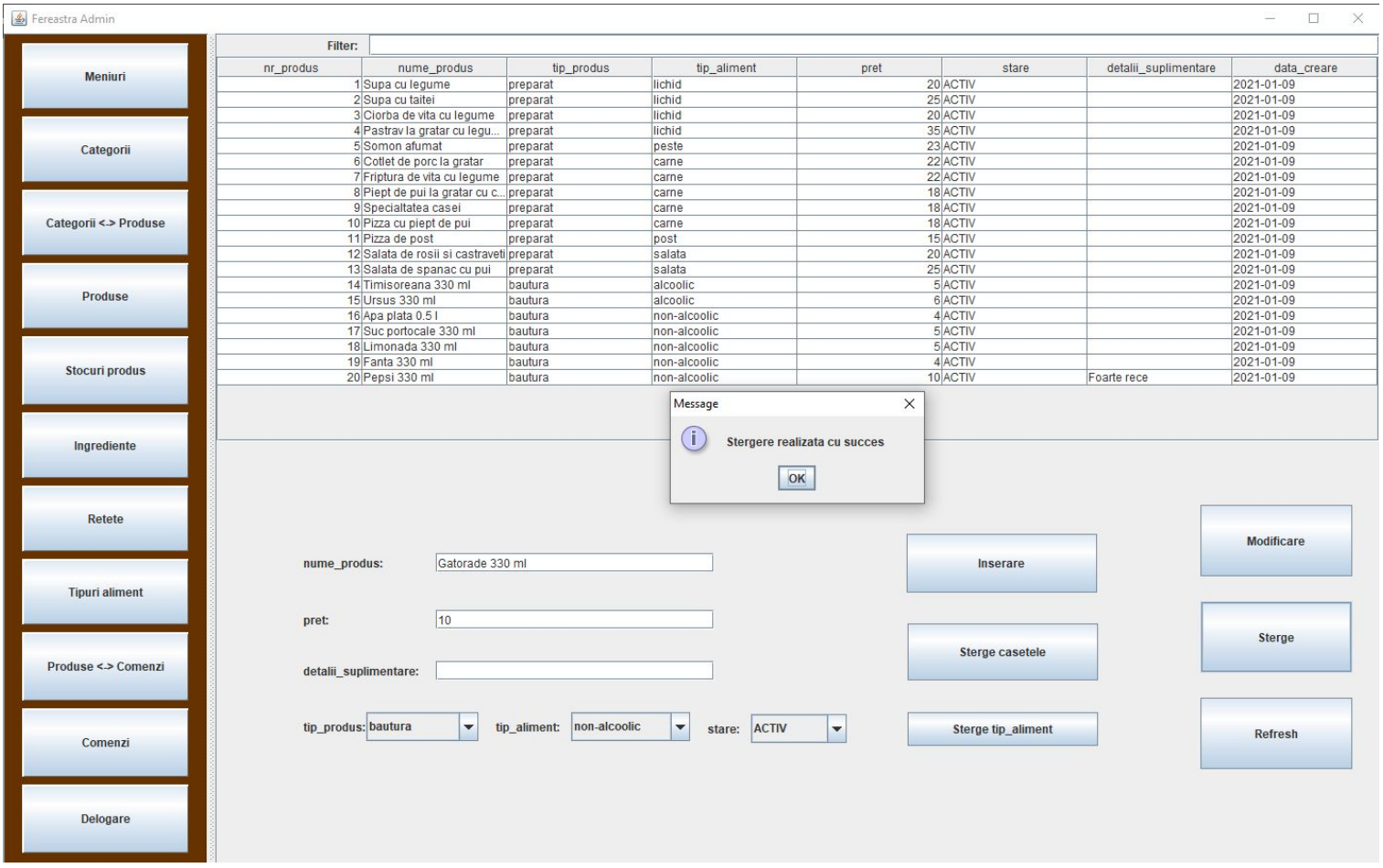
```
if (!nume_produs_mod.equals(resultSelectSet.getString(2))) {
            PreparedStatement prepUpdateSt2 = conn.prepareStatement("UPDATE Produse SET nume_produs = ? WHERE nr_produs = ?");
            prepUpdateSt2.setString(1, nume_produs_mod);
            prepUpdateSt2.setShort(2, Short.parseShort(nr_produs));
            prepUpdateSt2.execute();
          }

          if (!tip_produs_mod.equals(resultSelectSet.getString(3))) {
            PreparedStatement prepUpdateSt2 = conn.prepareStatement("UPDATE Produse SET tip_produs = ? WHERE nr_produs = ?");
            prepUpdateSt2.setString(1, tip_produs_mod);
            prepUpdateSt2.setShort(2, Short.parseShort(nr_produs));
            prepUpdateSt2.execute();
          }

          if (!pret_mod.equals(resultSelectSet.getString(5))) {
            PreparedStatement prepUpdateSt2 = conn.prepareStatement("UPDATE Produse SET pret = ? WHERE nr_produs = ?");
            prepUpdateSt2.setShort(1, Short.parseShort(pret_mod));
            prepUpdateSt2.setShort(2, Short.parseShort(nr_produs));
            prepUpdateSt2.execute();
          }

          if (!stare_mod.equals(resultSelectSet.getString(6))) {
            PreparedStatement prepUpdateSt2 = conn.prepareStatement("UPDATE Produse SET stare = ? WHERE nr_produs = ?");
```

```
            prepUpdateSt2.setString(1, stare_mod);
            prepUpdateSt2.setShort(2, Short.parseShort(nr_produs));
            prepUpdateSt2.execute();
          }

          if (!detalii_suplimentare_mod.equals(resultSelectSet.getString(7))) {
              PreparedStatement prepUpdateSt2 = conn.prepareStatement("UPDATE Produse SET detalii_suplimentare_produs = ? WHERE
nr_produs = ?");
            prepUpdateSt2.setString(1, detalii_suplimentare_mod);
            prepUpdateSt2.setShort(2, Short.parseShort(nr_produs));
            prepUpdateSt2.execute();
          }

          if (!tip_aliment_mod.equals(resultSelectSet.getString(4))) {
              PreparedStatement prepUpdateSt2 = conn.prepareStatement("UPDATE Produse SET tipuri_aliment_id_tip = (SELECT id_tip
FROM tipuri_aliment WHERE nume_tip = ?) WHERE nr_produs = ?");
            prepUpdateSt2.setString(1, tip_aliment_mod);
            prepUpdateSt2.setShort(2, Short.parseShort(nr_produs));
            prepUpdateSt2.execute(); }
```



În cele din urmă, pentru ștergerea datelor din tabele a fost utilizata instrucțiunea **DELETE**, după cum se poate observa în exemplul de mai jos:

```
PreparedStatement prepSt = conn.prepareStatement("DELETE FROM Produse WHERE nr_produs = ?");
    prepSt.setString(1, nr_produs);
    prepSt.execute();

    conn.createStatement().execute("commit");

    tblModel.removeRow(dataTable.convertRowIndexToModel(dataTable.getSelectedRow()));
    JOptionPane.showMessageDialog(this, "Stergere realizata cu succes");
```



Pentru inserarea sau ștergerea unei comenzi noi date de către un client s-au utilizat cate o **tranzacție**. După cum se poate observa în exemplu de mai jos, comanda este înregistrată:

```
String str =
            "DECLARE\n"
          + "   no_stock EXCEPTION;\n"
          + "   error_order EXCEPTION;\n"
          + "   produse_comandate produse_comenzi.nr_produse_comandate%TYPE;\n"
          + "   nr_masa_insert Comenzi.nr_masa%TYPE;\n"
          + "   detalii_suplimentare comenzi.detalii_suplimentare_comanda%TYPE;\n"
          + "   \n"
          + "   produs_in_reteta Ingrediente.id_ingredient%TYPE;\n"
          + "   produs_in_stoc stocuri_produs.stoc_produs%TYPE;\n"
          + "BEGIN\n"
          + "\n"
          + "   SAVEPOINT sp;\n"
          + "\n"
          + "   nr_masa_insert := ?;\n"
          + "   detalii_suplimentare := ?;\n"
                              + "          INSERT INTO Comenzi(id_comanda, data_comanda, nr_masa, detalii_suplimentare_comanda)
VALUES(NULL,SYSDATE,nr_masa_insert,detalii_suplimentare);";

      for (int i = 0; i < produseleMele.size(); i++) {
        str = str
            + "   BEGIN\n"
            + "      produse_comandate := ?;\n"
                              + "          INSERT INTO produse_comenzi(nr_produse_comandate, Produse_nr_produs, Comenzi_id_comanda)
VALUES(produse_comandate, (SELECT nr_produs FROM Produse WHERE nume_produs = ?), (SELECT MAX(id_comanda) FROM Comenzi));\n"
            + "\n"
                  + "       SELECT COUNT(Produse_nr_produs) INTO produs_in_reteta FROM Retete r WHERE r.Produse_nr_produs = (SELECT
nr_produs FROM Produse WHERE nume_produs = ?);\n"
                  + "       SELECT COUNT(Produse_nr_produs) INTO produs_in_stoc FROM stocuri_produs sp WHERE sp.Produse_nr_produs =
(SELECT nr_produs FROM Produse WHERE nume_produs = ?);\n"
            + "\n"
            + "      IF (produs_in_reteta > 0) THEN\n"
            + "         UPDATE Ingrediente i\n"
                  + "         SET stoc_ingredient = stoc_ingredient - produse_comandate * (SELECT r.cantitate_ingredient FROM Retete r WHERE
r.Produse_nr_produs = (SELECT nr_produs FROM Produse WHERE nume_produs = ?) and r.Ingrediente_id_ingredient = i.id_ingredient)\n"
                  + "         WHERE EXISTS (SELECT 1 FROM Retete r WHERE Produse_nr_produs = (SELECT nr_produs FROM Produse WHERE
nume_produs = ?) and r.Ingrediente_id_ingredient = i.id_ingredient);\n"
            + "      ELSIF (produs_in_stoc > 0) THEN\n"
            + "         UPDATE stocuri_produs sp\n"
            + "         SET stoc_produs = stoc_produs - produse_comandate\n"
            + "         WHERE sp.Produse_nr_produs = (SELECT nr_produs FROM Produse WHERE nume_produs = ?);\n"
            + "      ELSE \n"
            + "         RAISE no_stock;\n"
            + "      END IF;\n"
            + "   END;\n";
      }
      str +=
            "COMMIT;\n"
          + "\n"
```

```
    + "   EXCEPTION\n"
    + "      WHEN OTHERS THEN\n"
    + "         ROLLBACK TO sp;\n"
    + "         RAISE error_order;"
    + "END;";
selectProduct = conn.prepareStatement(str);
short val = Short.valueOf(String.valueOf(tableComboBox.getSelectedIndex() + 1));
selectProduct.setShort(1, val);
selectProduct.setString(2, (String) detailsComboBox.getSelectedItem());
for (int i = 0; i < produseleMele.size(); i++) {
  String nume = produseleMele.get(i).get(0);
  String cantitate = produseleMele.get(i).get(2);
  if (index % 7 == 3) {
    selectProduct.setString(index, cantitate);
  }
  selectProduct.setString(index + 1, nume);
  selectProduct.setString(index + 2, nume);
  selectProduct.setString(index + 3, nume);
  selectProduct.setString(index + 4, nume);
  selectProduct.setString(index + 5, nume);
  selectProduct.setString(index + 6, nume);
  index += 7;
  }
```