



Universitatea Tehnică "Gheorghe Asachi" din Iași

Facultatea de Automatică și Calculatoare

Baze de date-Proiect

Baze de date-Proiect

Student: Beșliu Cătălina

Grupa 1305A

1.Descrierea temei

Proiectul dat are ca scop gestiunea unei baze de date a meniului unui restaurant.

Tabelele folosite în cadrul proiectului sunt:

- Pizza
- Pizzatopping
- Burgers
- Sauce
- Drinks

Scopul aplicației este de a utiliza limbajul de manipulare a datelor(DML): funcțiile de inserare, actualizare a rândurilor dorite și de ștergere cât și afișarea acestor date pe un exemplu de interfață creată în limbaj Java.

2.Instalare aplicații necesare

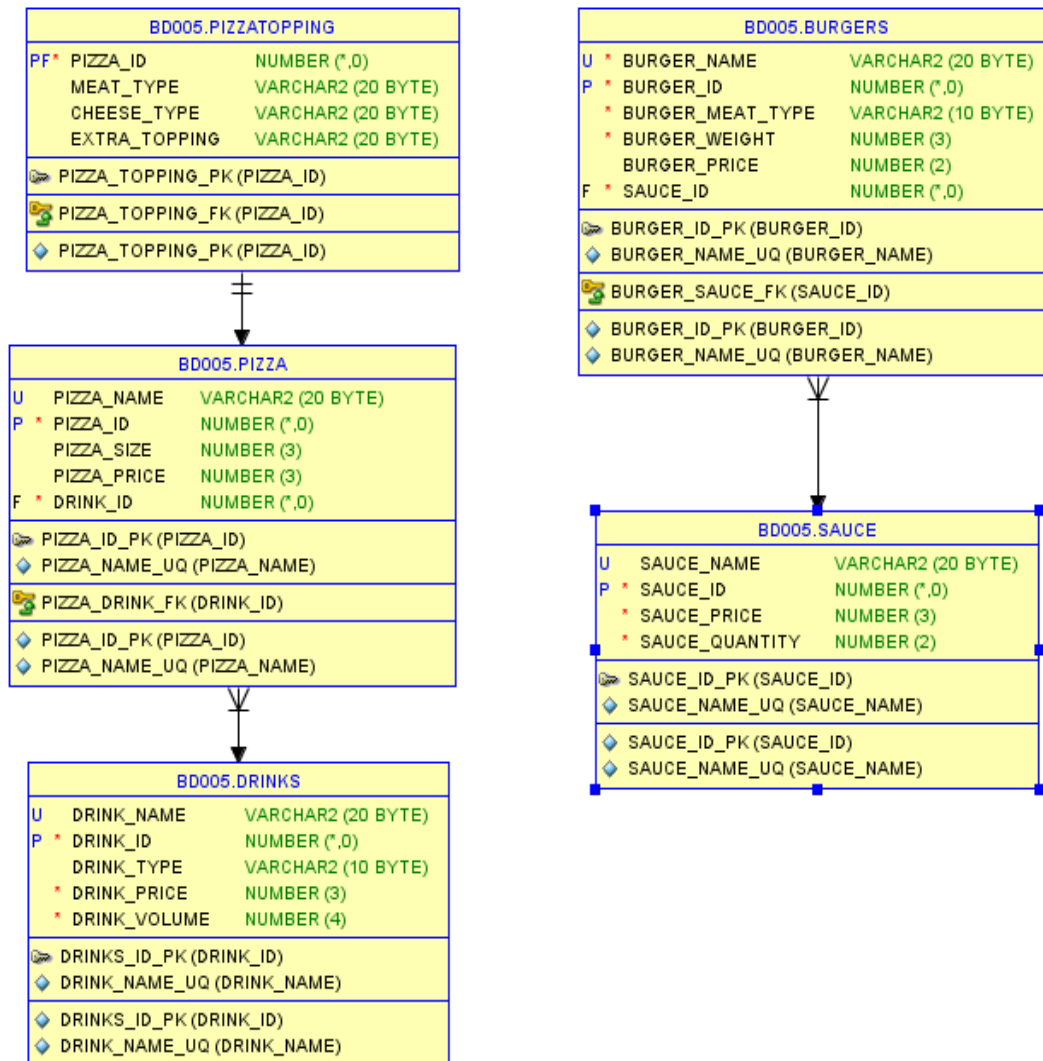
La crearea interfeței cu utilizatorul a fost utilizată aplicația IntelliJ IDEA ce poate fi descărcată de pe site-ul <https://www.jetbrains.com/idea/>, și la fel este obligatorie descărcarea a Java JDK de pe <https://www.oracle.com/java/technologies/javase-downloads.html> și adăugarea acestuia în "Environment Variables" a calculatorului.

La crearea componentelor GUI(Graphic User Interface) am utilizat framework-ul Java Swing, cadru cu un set de clase ce oferă anumite facilități când vine vorba de interfața grafică. Am utilizat butoane, casete de selectare, etichete, tabele și panouri. Conexiunea între baza de date și interfață are loc prin următoarele rânduri de cod:

```
String dbURL="jdbc:oracle:thin:@bd-dc.cs.tuiasi.ro:1539:orcl";
String username="bd005";
String password="bd005";
try {
    Connection connection = DriverManager.getConnection(dbURL, username, password);
```

Metoda DriverManager.getConnection stabilește o conexiune la baza de date și ca parametri necesită o adresă URL a bazei de date, username-ul și parola.

3.Data Modeler



4.Constrângeri

Constrângerile forțează reguli la nivelul tabelului. Am utilizat următoarele constrângeri:

- NOT NULL
- CHECK
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY

1)NOT NULL:

Această constrângere se folosește atunci când se dorește impunerea faptului ca unei coloane să nu i se atribuie valori nule. Am utilizat aceasta constrângerea la crearea tabelului indicând că prețul, dimensiunea, ID-ul obiectului nu pot avea valori nule.

2)CHECK

CHECK permite aplicarea oricărei condiții ce va fi satisfăcută. În cadrul tabelelor am utilizat-o asupra coloanelor de tip varchar în combinație cu expresiile regulate astfel impunând restricții ca nume de produse să nu conțină cifre.

Ex: ALTER TABLE sauce

```
ADD CONSTRAINT sauce_name_check CHECK (regexp_like(sauce_name,'^[a-zA-Z]*$'));
```

3)UNIQUE

UNIQUE impune ca valorile unei coloane (sau a mai multor coloane) să fie unice pentru fiecare înregistrare din tabel. Am folosit această constrângere asupra coloanelor numelui produselor astfel îndeplinindu-i scopul deoarece nu ar trebui să existe înregistrări copii.

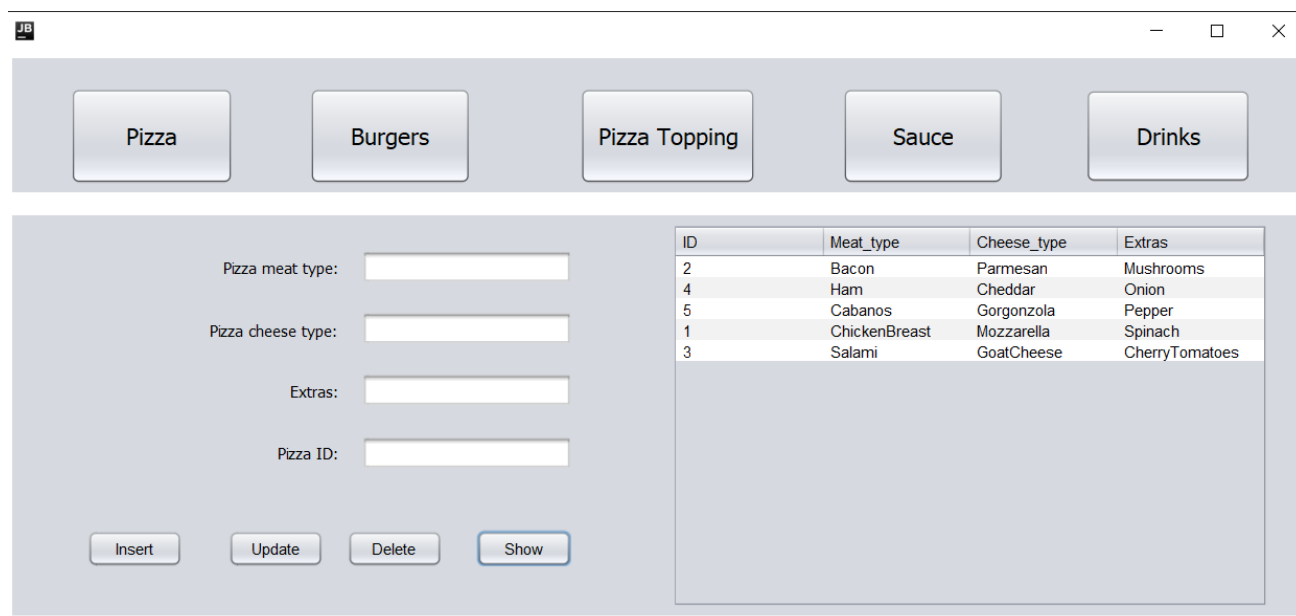
4)PRIMARY KEY

Primary key identifică unic fiecare înregistrare și astfel am folosit-o la coloanele ID a tabelelor.

5)FOREIGN KEY

Cu ajutorul constrângerii foreign key am facut relații de one-to-many între tabela "burgers" și tabela "sauce", tabela "pizza" și tabela "drinks", precum și relație de one-to-one între tabela "pizza" și "pizzatopping" (fiecărei pizza îi este atribuit un singur tip de topping).

5.Instrucțiuni SQL folosite



ID	Meat_type	Cheese_type	Extras
2	Bacon	Parmesan	Mushrooms
4	Ham	Cheddar	Onion
5	Cabanos	Gorgonzola	Pepper
1	ChickenBreast	Mozzarella	Spinach
3	Salami	GoatCheese	CherryTomatoes

Fig.1 Interfața proiectului

Conform fig.1 în interfața proiectului, fiecare tabel a bazei de date deține propriul său buton și panou de afișare. La fel, în cadrul fiecărui panou se află 4 butoane ce implementează clasa ActionListener, butonul "Show"-pentru afișarea datelor, "Insert"-pentru inserarea unei noi înregistrări, "Update"-actualizează înregistrarea specificată prin ID-ul dat și "Delete"-ștergere la fel pe bază de ID.

- SELECT

Sintaxa: SELECT [DISTINCT] {*,column [alias],...} FROM table;

Exemplu de cod:

```
Connection connection = DriverManager.getConnection(dbURL, username, password);
Statement statement=connection.createStatement();
String query="SELECT * FROM burgers";
ResultSet rs=statement.executeQuery(query);
Burger burger;
while(rs.next()){
    burger=new Burger(rs.getString( s: "burger_name"),rs.getInt( s: "burger_id"),
        rs.getString( s: "burger_meat_type"),rs.getInt( s: "burger_weight"),
        rs.getInt( s: "burger_price"),rs.getInt( s: "sauce_id"));
    burgersList.add(burger);
}
```

Rezultat:

Name	ID	Meat_type	Weight	Price	Sauce_ID
Cheesebur...	14	beef	460	75	1
ChickenBu...	6	chicken	530	85	1
PulledPork...	7	pork	600	89	4
RoyalBurger	9	beef	650	99	2
AmericanB...	11	beef	300	49	5
KingBurger	13	lamb	620	95	7

Instrucțiunea afișează coloanele selectate din tabela menționată. În cadrul programului este apelată la click-ul butonului "show" și afișează toate înregistrările din tabelul respectiv.

- INSERT

Sintaxa: INSERT INTO table [(column [,column...])] VALUES (value [,value...]);

Adaugă noi înregistrări într-un tabel la apăsarea butonului "Insert", luând date din casetele de text respectiv completate cu date valide pentru a fi preluate de aplicație.

Exemplu de cod:

```

String burgerName = burgerNameField.getText();
String burgerMeatType = burgerMeatTypeField.getText();
String burgerWeight = burgerWeightField.getText();
String burgerPrice = burgerPriceField.getText();
String burgerSauceID = burgerSauceIDField.getText();
if ((burgerName.length() > 0) && (burgerMeatType.length() > 0) && (Integer.parseInt(burgerWeight) > 0)
    && Integer.parseInt(burgerPrice) > 0 && Integer.parseInt(burgerSauceID) > 0) {

    try {
        Connection connection = DriverManager.getConnection(dbURL, username, password);
        Statement statement = connection.createStatement();
        String sql = "INSERT INTO burgers(burger_name,burger_meat_type,burger_weight,burger_price,sauce_id) VALUES('" + burgerName
            + "','" + burgerMeatType + "','" + Integer.parseInt(burgerWeight) + "','" + Integer.parseInt(burgerPrice)
            + "','" + Integer.parseInt(burgerSauceID) + "')";
        int result = statement.executeUpdate(sql);
        if (result > 0) {
            System.out.println("Row inserted");
        }

        connection.close();
    } catch (SQLException e) {
        System.out.println("Not connected to database");
        e.printStackTrace();
    }
} else
    System.out.println("reintroduceti datele");
}

```

- UPDATE

Sintaxa: UPDATE table SET column= (value) [,column...]] [WHERE condition];

Exemplu de cod:

```

String pizzaName = pizzaNameField.getText();
String pizzaSize = pizzaSizeField.getText();
String pizzaPrice = pizzaPriceField.getText();
String pizzaDrinkID = pizzaDrinkIDField.getText();
String pizzaID = pizzaIDField.getText();
String dbURL = "jdbc:oracle:thin:@bd-dc.cs.tuiasi.ro:1539:orcl";
String username = "bd005";
String password = "bd005";
if ((pizzaName.length() > 0) && Integer.parseInt(pizzaID) > 0
    && (Integer.parseInt(pizzaSize) > 0) && Integer.parseInt(pizzaPrice) > 0
    && Integer.parseInt(pizzaDrinkID) > 0) {
    try {
        Connection connection = DriverManager.getConnection(dbURL, username, password);
        Statement statement = connection.createStatement();
        String sql = "UPDATE pizza SET pizza_name='" + pizzaName + "',pizza_size="
            + Integer.parseInt(pizzaSize) + ",pizza_price=" + Integer.parseInt(pizzaPrice)
            + ",drink_id=" + Integer.parseInt(pizzaDrinkID) + " WHERE pizza_id=" + Integer.parseInt(pizzaID);

        int result = statement.executeUpdate(sql);
        if (result > 0) {
            System.out.println("Row updated");
        }

        connection.close();
    } catch (SQLException e) {
        System.out.println("Not connected to database");
        e.printStackTrace();
    }
} else
    updatePizzaButtonActionPerformed()

```

La click-ul butonului "Update" se actualizează înregistrarea cu ID-ul meționat cu date valide, preluate din casetele de text.

- DELETE

Sintaxa: DELETE FROM tabel WHERE condition;

Șterge înregistrarea selectată.

```
String burgerID=burgerIDField.getText();
if (Integer.parseInt(burgerID)>0 ) {

    try {
        Connection connection = DriverManager.getConnection(dbURL, username, password);
        Statement statement = connection.createStatement();
        String sql="DELETE FROM burgers WHERE burger_id="+burgerID;

        int result = statement.executeUpdate(sql);
        if (result > 0) {
            System.out.println("Row deleted");
        }

        connection.close();
    } catch (SQLException e) {
        System.out.println("Not connected to database");
        e.printStackTrace();
    }
} else
    System.out.println("reintroduceti datele");
```