# JEGYZŐKÖNYV

Adatbázisrendszerek II.

Mobiltelefon Nyilvántartás

Készítette: **Csomor Bence Patrik**

Neptunkód: **TVIK4I**

**Feladat leírása:**

A feladatban egy web-es telefon nyilvántartást modelljét fogom bemutatni. Három tábla szerepel az adatbázisban:

- WebShop
- Costumer
- Phone

A Costumer tábla összeköttetésben áll a Phone táblával az „id”-n keresztül.

**A táblák létrehozása szolgáló SQL parancsok:**

A létrehozásnál ügyelni kell a sorrendre, először azokat a táblákat kell létrehozni, amelyekben nincs idegen kulcs, és ezután azokat, amelyekben van, hiszen az idegen kulcsnak a már létrehozott táblára kell mutatnia. Az idegen kulcsot tartalmazó mezők típusának meg kell egyeznie a referenciaként szolgáló, másik táblában találhatókulcsmező típusával.

```
public static void StatikusTablaLetrehozas() {
    String sqlp_phone = "create table phone "
            + "("
            + "id int not null, "
            + "type char(10) not null, "
            + "color char(10) default 'white', "
            + "release date, "
            + "price integer not null, "
            + "condition char(10), "
            + "primary key (id)"
            + ")";
    String sqlp_webShop = "create table webShop "
            + "("
            + "id numeric(3) not null, "
            + "name char(30) not null, "
            + "address char(30), "
            + "url char(40), "
            + "foundation date, "
            + "primary key (id)"
            + ")";

    String sqlp_costumer = "create table costumer "
            + "("
            + "identityN int not null, "
            + "name char(30) not null, "
            + "address char(30), "
            + "salary int, "
            + "birthday date, "
            + "primary key (identityN)"
            + ")";
```

```
    if (conn != null) {
        try {
            s = conn.createStatement();
            s.executeUpdate(sqlp_phone);
            System.out.println("Phone table was created\n");
            s.executeUpdate(sqlp_webShop);
            System.out.println("WebShop table was created\n");
            s.executeUpdate(sqlp_costumer);
            System.out.println("Costumer table was created\n");
            s.close(); //erőforrás felszabadítása
        } catch(Exception ex) {
            System.err.println(ex.getMessage());
        }
    }
}
```

**A táblák feltöltésére szolgáló SQL parancsok:**

Feltölti a létrehozott tábláka adatokkal és megadott tömbökben tárolja azokat. Sikeres felöltés esetén kiírja, hogy a rekord sikeres volt.

```
public static void StatikusAdatfelvitel() {
    if (conn != null) {
        String[] sqlp_ws = {
                "insert into webShop values(123, 'Phone Shop', '1st street', 'https://www.phoneshop.com', to_date('2010.07.10', 'yyyy.mm.dd'))",
                "insert into webShop values(456, 'Mobile Buys', '2nd street', 'https://www.mobilebuys.com', to_date('2014.04.14', 'yyyy.mm.dd'))"
        };

        String[] sqlp_phone = {
                "insert into phone values(1, 'Samsung', 'red', to_date('2020.01.01', 'yyyy.mm.dd'), 165000, 'new')",
                "insert into phone values(2, 'IPhone', 'black', to_date('2016.10.21', 'yyyy.mm.dd'), 280000, 'used')",
                "insert into phone values(3, 'Huawei', 'white', to_date('2021.05.17', 'yyyy.mm.dd'), 150000, 'new')",
                "insert into phone values(4, 'LG', 'blue', to_date('2022.08.10', 'yyyy.mm.dd'), 300000, 'new')"
        };

        String[] sqlp_costumer = {
                "insert into costumer values(12345678, 'Kiss Béla', 'Budapest, Pöttyös utca 69', 1200000, to_date('1990.04.12', 'yyyy.mm.dd'), 1)",
                "insert into costumer values(87654321, 'Nagy Ferenc', 'Miskolc, Kockás utca 420', 3400000, to_date('1994.12.05', 'yyyy.mm.dd'), 2)",
                "insert into costumer values(12398746, 'Kovács Dániel', 'Debrecen, Hős utca 84', 5400000, to_date('1985.03.24', 'yyyy.mm.dd'), 3)",
                "insert into costumer values(54637281, 'Horváth Bence', 'Gyál, Damjanich utca 10', 2300000, to_date('1997.07.17', 'yyyy.mm.dd'), 4)"
        };
```

```
            for (int i=0; i<sqlp_ws.length; i++) {
                try {
                    s = conn.createStatement();
                    s.executeUpdate(sqlp_ws[i]);
                    System.out.println("webShop recorded\n");
                    s.close();
                } catch(Exception ex) {
                    System.out.println(ex.getMessage());
                }
            }
            for (int i=0; i<sqlp_phone.length; i++) {
                try {
                    s = conn.createStatement();
                    s.executeUpdate(sqlp_phone[i]);
                    System.out.println("Phone recorded\n");
                    s.close();
                } catch(Exception ex) {
                    System.out.println(ex.getMessage());
                }
            }
            for (int i=0; i<sqlp_costumer.length; i++) {
                try {
                    s = conn.createStatement();
                    s.executeUpdate(sqlp_costumer[i]);
                    System.out.println("Costumer recorded\n");
                    s.close();
                } catch(Exception ex) {
                    System.out.println(ex.getMessage());
                }
            }
        }
    }
```

**Program funkciók:**

**Driver regisztrálása és kapcsolódás a szerverhez:**

Regisztrálja a Drivert és csatlakoztatja a szerverhez a projektet. Ezzel biztosítjuk a kapcsolatot az adatbázissal, ez nagyon fontos lépés a program elején.

```
public static void DriverReg() {
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        System.out.println("Sikeres driver regisztrálás\n");
    } catch (Exception e) {
        System.err.println(e.getMessage());
    }
}

public static void Connect() {

    try {
        conn = DriverManager.getConnection(url, user, pwd);
        System.out.println("Sikeres kapcsolódás\n");
    } catch(Exception e) {
        System.err.println(e.getMessage());
    }
}
```

```
Sikeres kapcsolódás

Sikeres driver regisztrálás
```

**Táblák törlése:**

Letörli a korábban létrehozott táblákat az adatbázisból. Ez azért szükséges a program indítása elején, mert ha megpróbálunk olyan táblát létrehozni, ami már szerepel az adatbázisban, akkor hibát kapunk.

```
public static void TablaTorlese() {
    try {
        String sqlp_webShop = "DROP TABLE webShop";
        String sqlp_costumer = "DROP TABLE costumer";
        String sqlp_phone = "DROP TABLE phone";
        s = conn.createStatement();
        s.executeUpdate(sqlp_webShop);
        System.out.println("WebShop table deleted!\n");
        s.executeUpdate(sqlp_costumer);
        System.out.println("Costumer table deleted!\n");
        s.executeUpdate(sqlp_phone);
        System.out.println("Phone table deleted!\n");
        s.close();
    } catch(Exception ex) {
        System.err.println(ex.getMessage());
    }
}
```

```
WebShop table deleted!

Costumer table deleted!

Phone table deleted!
```

**Statikus tábla módosítása:**

Módosítja a Costumer táblát. Hozzáadja referenciaként a Phone id adattagját. Ezáltal kapcsolat lép fel a két tábla között. Tehát a két tábla törlésénél ügyelni kell a sorrendre, mert a kapcsolat miatt csak a megfelelő sorrendben lehet a táblákat örölni.

```
public static void StatikusTablaModositas() {
    //eltárol
    if (conn != null) {
        try {
            String sqlp = "alter table costumer add(id references phone)";
            s = conn.createStatement();
            s.executeUpdate(sqlp);
            System.out.println("Phone table modified\n");
            s.close();
        } catch(Exception ex) {
            System.err.println(ex.getMessage());
        }
    }
}
```

```
Phone table modified
```

**Statikus adattörlés:**

Bekér egy id-t és ha tartozik hozzá egy vásárló, akkor az adott id-hez tartozó vásárlót törli. Majd kiírja, hogy az adott id-vel rendelkező vásárló törölve lett.

```
public static void StatikusAdattorles() {
    System.out.println("Costumer to delete: ");
    String id = sc.next();
    String sqlp = "delete from costumer where id like'"+id+"'";
    if (conn != null) {
        try {
            s = conn.createStatement();
            s.executeUpdate(sqlp);
            s.close();
            System.out.println("Cosumer with " + id + " id was deleted\n");
        } catch(Exception ex) {
            System.err.println(ex.getMessage());
        }
    }
}
```

```
Costumer to delete:
2
Cosumer with 2 id was deleted
```

**Statikus adatlekérdezés:**

Bekér egy táblanevet. Ha a megadott tábla neve létezik, akkor az adott táblában lévő összes adatot kiírja tabulátorokkal tagolva.

```
public static void StatikusLekerdezes() {
    System.out.println("Melyik tábla adatait szeretnéd látni? ");
    String table = sc.next().trim();
    if (conn != null) {
        if(table.equals("phone")) {
            String  sqlp = "select * from phone";
            System.out.println("ID"+  "\t\t" +" Type"+"\t\t"+" Color"+"\t\t"+" Release"+"\t\t"+" Price"+"\t\t"+" Condition");
            try {
                s = conn.createStatement();
                s.executeQuery(sqlp);
                rs = s.getResultSet();
                while(rs.next()) {
                    //A get metódusoknak a megfelelő típusú táblamezőket kell megadni
                    int id = rs.getInt("id");
                    String type = rs.getString("type");
                    String color = rs.getString("color");
                    String release = rs.getString("release");
                    Integer price = rs.getInt("price");
                    String condition = rs.getString("condition");
                    System.out.println(id + "\t\t" + type +  "\t" + color + "\t" +
                    release + "\t" + price + "\t\t" + condition);
                }
                rs.close();
            } catch(Exception ex) {
                System.err.println(ex.getMessage());
            }
```

```
        } else if(table.equals("costumer")) {
            String  sqlp = "select * from costumer";
            System.out.println("IdentityN"+  "\t\t" +" Name"+"\t\t\t\t"+" Address"+"\t\t\t"+" Salary"+"\t\t"+" Birthday");
            try {
                s = conn.createStatement();
                s.executeQuery(sqlp);
                rs = s.getResultSet();
                while(rs.next()) {
                    //A get metódusoknak a megfelelő típusú táblamezőket kell megadni
                    int identityN = rs.getInt("identityN");
                    String name = rs.getString("name");
                    String address = rs.getString("address");
                    int salary = rs.getInt("salary");
                    String birthday = rs.getString("birthday");
                    System.out.println(identityN + "\t\t" + name +  "\t" + address + "\t" +
                            salary + "\t" + birthday);
                }
                rs.close();
            } catch(Exception ex) {
                System.err.println(ex.getMessage());
            }
```

```
        } else if(table.equals("webShop")) {
            String  sqlp = "select * from webShop";
            System.out.println("ID"+  "\t\t" +" Name"+"\t\t"+" Address"+"\t\t"+" URL"+"\t\t"+" Foundation");
            try {
                s = conn.createStatement();
                s.executeQuery(sqlp);
                rs = s.getResultSet();
                while(rs.next()) {
                    //A get metódusoknak a megfelelő típusú táblamezőket kell megadni
                    long id = rs.getLong("id");
                    String name = rs.getString("name");
                    String address = rs.getString("address");
                    String url = rs.getString("url");
                    String foundation = rs.getString("foundation");
                    System.out.println(id + "\t\t" + name +  "\t" + address + "\t" +
                            url + "\t" + foundation);
                }
                rs.close();
            } catch(Exception ex) {
                System.err.println(ex.getMessage());
            }
        } else {
            System.out.println("Ilyen tábla nincs!");
            StatikusLekerdezes();
        }
    }
}
```

```
Melyik tábla adatait szeretnéd látni?
phone
ID              Type            Color           Release                 Price           Condition
1               Samsung         red             2020-01-01 00:00:00.0   330000          new
2               IPhone          black           2016-10-21 00:00:00.0   280000          used
3               Huawei          white           2021-05-17 00:00:00.0   150000          new
4               LG              blue            2022-08-10 00:00:00.0   300000          new
```

**Módosítható kurzor:**

Bekér egy színt a felhasználótól és ha létezik az adatbázisban a megadott szín, akkor az összes ahhoz tartozó telefon árát megduplázza. Majd kiírja, hogy az adott színnel rendelkező telefon ára duplázva lett.

```
public static void ModosithatoKurzor() {
    System.out.println("Color: ");
    String color = sc.next().trim();
    String sqlp = "select price from phone where color= '"+color+"'";
    if(conn != null) {
        try {
            s=conn.createStatement(ResultSet.TYPE_FORWARD_ONLY, ResultSet.CONCUR_UPDATABLE);
            rs=s.executeQuery(sqlp);
            while(rs.next()) {
                int oldPrice = rs.getInt("price");
                rs.updateInt("price", (oldPrice*2));
                rs.updateRow();
            }
        }catch(Exception ex) {
            System.err.println(ex.getMessage());
        }
    }
    System.out.println("Phone price with "+color+" color was doubled");
}
```

```
Color:
red
Phone price with red color was doubled
```

**Dinamikus lekérdezés:**

Dinamikusan lekérdezi a megadott WebShop kódja alapján annak nevét.

```
public static void DinamikusLekerdezes() {
    System.out.println("WebShop ID-je: ");
    String id = sc.next().trim();
    String sqlp = "select name from webShop where id like '"+id+"'";
    if(conn != null) {
        try {
            s=conn.createStatement();
            s.executeQuery(sqlp);
            rs=s.getResultSet();
            while(rs.next()) {
                    String name = rs.getString("name");

                    System.out.println("Name: "+name);
            }
            rs.close();
        }catch(Exception ex) {
            System.out.println(ex.getMessage());
        }
    }
}
```

```
WebShop ID-je:
123
Name: Phone Shop
```

**Dinamikus adattörlés:**

Dinamikusan letörli a megadott Customert az adatbázisból. Bekérjük az egyik vásárló hoz tartozó id-t és az ahoz az id-hez tartozó vásárlót töröljük.

```
public static void DinamikusAdattorles() {
    System.out.println("Costumer to delete: ");
    String id = sc.next();
    //Az sql parancsban a ? helyére kerülnek a paraméterek
    String sqlp = "delete from " + user + ".COSTUMER where id=?";
    if (conn != null) {
        try {
            ps = conn.prepareStatement(sqlp);
            ps.setString(1, id);
            ps.executeUpdate();
            ps.close();
            System.out.println("costumer with "+id+" id was deleted dynamically\n");
        } catch(Exception ex) {
            System.err.println(ex.getMessage());
        }
    }
}
```

**String dátummá alakítása:**

A megadott Sringet Date formátummá alakítja, hogy fel lehessen venni az adatbázisba.

A dátummá alakítás azért kell, mert így egyszerűen meg lehet adni szövegesen a dátumot és ezzel a metódussal dátum lesz belőle.

```
public static Date StringToDate(String release) throws ParseException {
    //Instantiating the SimpleDateFormat class
    SimpleDateFormat formatter = new SimpleDateFormat("yyyy.MM.dd");
    //Parsing the given String to Date object
    Date releaseDate = formatter.parse(release);
    System.out.println("Date object value: " + releaseDate);
    return releaseDate;
 }
```

**Bejelentkezési felület:**

Bekéri a felhasználótól az url-t, a felhasználó nevet és a jelszót. Ha ezek az értékek validak, akkor a kapcsolat létrejön és kiírja, hogy sikeres kapcsolódás.

```
public static void Bejelentkezes() {
    try {
        System.out.println("Kérem az url-t: ");
        String url1 = sc.next().trim();
        System.out.println("Kérem az user-t: ");
        String user1 = sc.next().trim();
        System.out.println("Kérem a pwd-t: ");
        String pwd1 = sc.next().trim();
        conn = DriverManager.getConnection(url1, user1, pwd1);
        System.out.println("Sikeres kapcsolódás\n");
    } catch(Exception e) {
        System.err.println(e.getMessage());
    }
}
```