

# JEGYZŐKÖNYV

Adatbázisrendszerek II.

Mobiltelefon Nyilvántartás

Készítette: **Csomor Bence Patrik**

Neptunkód: **TVIK4I**

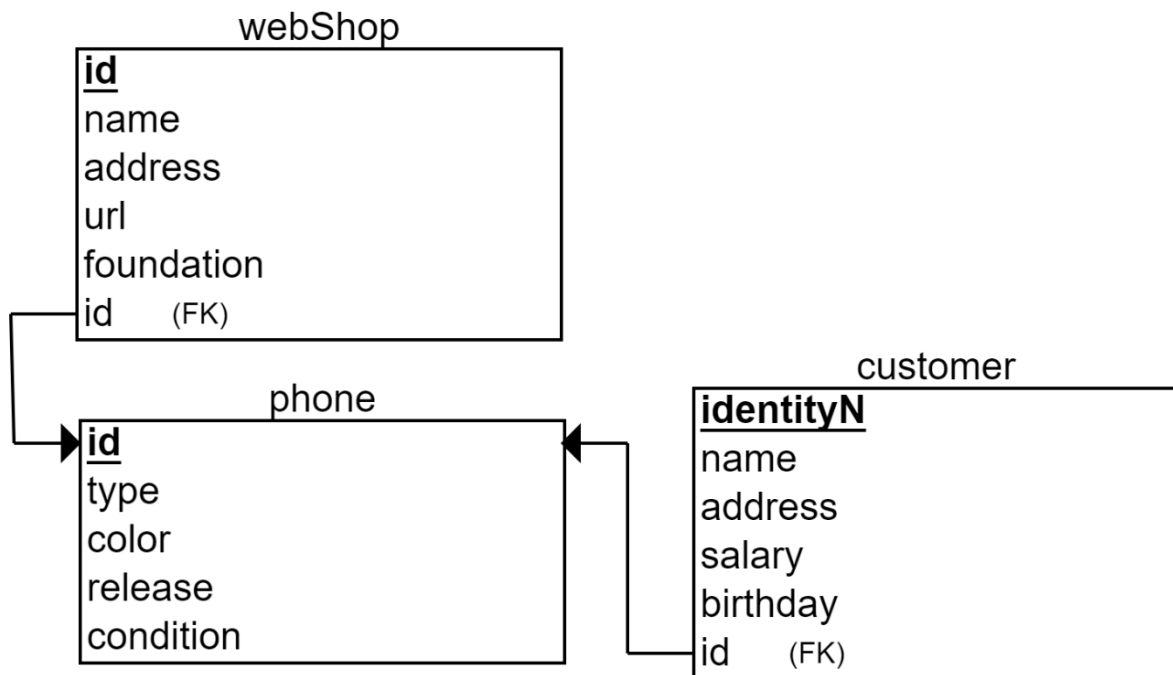
### Feladat leírása:

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.

### Schema ábra:



### 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 található kulcsmező 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 táblák adataival és megadott tömbökben tárolja azokat. Sikeres feltöltés 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\t" + color + "\t\t" + release + "\t\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\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\t" + address + "\t\t" + salary + "\t\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\t" + address + "\t\t" + url + "\t\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 adattörölés:

Dinamikusan törölni a megadott Customert az adatbázisból. Bekérjük az egyik vásárlóhoz tartozó id-t és az ahhoz 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 Stringet 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 módszerrel 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 nevét és a jelszót. Ha ezek az értékek validak, akkor a kapcsolat létrejön és kiírja, hogy sikeres kapcsolódás.

```
public static void Bejelentkezés() {  
    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());  
    }  
}
```

## Dinamikus adatfelvitel:

Dinamikusan bekéri az adatokat. Ha nem jó akkor újra kéri. A megadott adatokat bekéri, a dátumot String bekérése után dátummá alakítja. Ha sikeres akkor kiírja, hogy felvéve.

```
public static void DinamikusAdatfelvitel() throws ParseException {
    if (conn != null) {
        //Az SQL parancsban a ? helyére kerülnek a paraméterek
        String sqlp = "insert into phone (id, type, color, release, price, condition)" +
            "values (?, ?, ?, ?, ?, ?)";

        System.out.println("Kérem az id-t: ");
        String id = sc.next().trim();
        System.out.println("Kérem a típust: ");
        String type = sc.next().trim();
        System.out.println("Kérem a színt: ");
        String color = sc.next().trim();
        System.out.println("Kérem a megjelenési dátumot: (yyyy.mm.dd) ");
        String release = sc.next().trim();
        Date releaseDate = TVIK4I_DB2.StringToDate(release);
        java.sql.Date sqlDate = new java.sql.Date(releaseDate.getTime());
        System.out.println("Kérem a árat: ");
        float price = sc.nextFloat();
        System.out.println("Kérem az állapotát: ");
        String condition = sc.next().trim();
        try {
            ps = conn.prepareStatement(sqlp);
            ps.setString(1, id);
            ps.setString(2, type);
            ps.setString(3, color);
            ps.setDate(4, sqlDate);
            ps.setFloat(5, price);
            ps.setString(6, condition);
            ps.executeUpdate();
            ps.close();
            System.out.println("Telefon felvéve\n");
        } catch (Exception ex) {
            System.err.println(ex.getMessage());
            System.out.println("Az ID létezik, adjon meg mást!");
            DinamikusAdatfelvitel();
        }
    }
}
```

## 2. Feladat:

PL/SQL szerver oldali tárolt programok:

Új telefon létrehozása. Tárolt eljárással felvisszük az új telefon adattagjait. Először megadjuk a telefon elemeit és azok típusait, majd ezeket beillesztjük a telefon értékeihez.

A létrehozás kódja:

```
create or replace procedure UjPhone(id number, type char, color char,  
release date, price number, condition char) as  
begin  
insert into phone values(id, type, color, release, price, condition);  
end;
```

A futtatás kódja:

```
begin  
UjPhone(6, 'Huawei', 'blue', TO_DATE('2010.04.14','yyyy.MM.dd'), 210000, 'new');  
end;
```

Új vásárló létrehozása. Tárolt eljárással felvisszük az új vásárló adattagjait. Először megadjuk a vásárló elemeit és azok típusait, majd ezeket beillesztjük a vásárló értékeihez.

A létrehozás kódja:

```
create or replace procedure UjCustomer(identityN number, cname char, address char,  
salary number, birthday date, pid number) as perror exception;  
begin  
if salary <= 200000 then  
raise perror;  
else  
insert into customer values(identityN, cname, address, salary, birthday, pid);  
end if;  
exception  
when perror then  
dbms_output.put_line('A fizetés túl alacsony!');  
end;
```

A futtatás kódja:

```
begin  
UjCustomer(32145698, 'Valaki Gergő', 'Eger, Egri utca 44', 500000, TO_DATE('2000.10.02','yyyy.MM.dd'), 5);  
end;
```

A telefonok árának növelése egy adott értékkel. Megnöveljük a telefonok árát 20000Ft-vel. Ha nem talál egy elemet sem, akkor kiírja, hogy egyik ár sem frissült, ha talál akkor frissíti őket.

A létrehozás kódja:

```
create or replace procedure incPrice as
allRows integer;
begin
update phone set price = price+20000;
if sql%notfound then
dbms_output.put_line('Nem frissült egy ár sem');
elsif sql%found then
allRows:=sql%rowcount;
dbms_output.put_line( allRows || ' darab phone frissítve lett ');
end if;
end;
```

A futtatás kódja:

```
begin
    incPrice;
end;
```

## Naplózások:

Létrehozunk egy naplózás táblát, mellyel az esemény mivoltát, a változtatott adatot és a változtatás dátumát tőgzíthetjük.

Naplózás tábla létrehozása:

```
create table Naplozas(
Esemeny varchar2(20),
Adat varchar2(200),
Datum timestamp(6));
```

Naplózás feltöltésének kódja:

```
create or replace trigger feltolt after insert on phone for each row
BEGIN
insert into Naplozas values('Beszúrás', :new.ID||'_'||:new.TYPE||'_'||:new.COLOR||'_'||:new.RELEASE||'_'||:new.PRICE||'_'||:new.CONDITION, sysdate);
END;
```

### Naplózás módosításának a kódja:

```
create or replace trigger modositas after update on phone for each row
begin
    insert into Naplozas values('Módosítás', :old.ID||'_'||:old.TYPE||'_'||:old.COLOR||'_'||:old.RELEASE||'_'||:old.PRICE||'_'||:old.CONDITION
    ||'_'||:new.ID||'_'||:new.TYPE||'_'||:new.COLOR||'_'||:new.RELEASE||'_'||:new.PRICE||'_'||:new.CONDITION, sysdate);
end;
```

### Naplózás törlésének a kódja:

```
create or replace trigger torles after delete on phone for each row
begin
    insert into Naplozas values('Törlés', :old.ID||'_'||:old.TYPE||'_'||:old.COLOR||'_'||:old.RELEASE||'_'||:old.PRICE||'_'||:old.CONDITION, sysdate);
end;
```

Kontrollált módosítást is végre tudunk hajtani. Abban az esetben, ha az új telefon megjelenési dátuma 1970.01.01 előtt volt vagy lett megadva, akkor nem fogadja el az új értéket.

### Módosítás kontrollálása:

```
create or replace trigger modositas_kontrollalas
before update on phone for each row
declare
begin
    if :new.release not between to_date('1970.01.01','yyyy.MM.dd') and
    sysdate then
        dbms_output.put_line('Nem helyes dátum');
        :new.release:=:old.release;
    end if;
end;
```

A telefon megjelenési dátumát állítjuk egy adott értékre. Ezt id alapján el tudjuk érni.

### A kód:

```
update phone set release = to_date('1984.01.15','yyyy.MM.dd') where id = 3;
```

### Vásárló törlésének a kódja:

```
create or replace procedure customerDel(iddel int) as
begin
    delete from customer where identityN=iddel;
end;
```

A vásárló törlés futtatásának kódja:

```
begin
    customerDel(12398746);
end;
```

A vásárló nevének lekérésének a kódja:

```
create or replace function getName(getidentityN in integer) return
varchar as
x customer.cname$type;
y integer;
begin
select count(*) into y from customer where getidentityN=identityN;
if(y<1) then
x:='Nem létezik a kód';
else
select cname into x from customer where getidentityN = identityN;
end if;
return x;
end;
```

A vásárló nevének lekérésének futtatása, adott vásárló nevének lekérése:

```
select getName(12345678) from dual;
```

Telefon árának módosítása:

```
create or replace procedure PhoneModositas(pid number, ujPrice
number) as
begin
update phone set price = ujPrice where id = pid;
end;
```

Ki tudjuk számolni az adott kritériumoknak megfelelő elemek adott értékeinek átlagát.

Az adott márkájú telefonok átlagárát kiszámító kód:

```
create or replace function GetAvgPrice(typein char) return
char as
x integer;
y integer;
ni char(100):= 'Nem létezik ilyen típus';
begin
select count(*) into y from phone where type = typein;
if y >= 1 then
select avg(price) into x from phone where type = typein;
ni:='Az átlagár: '||x;
end if;
return ni;
end;
```

Az átlagár kiszámításának futtatási kódja:



```
select getavgprice('LG') from dual
```

Az adatokat külön létrehozott csomagokban is tudjuk tárolni. Ehhez létrehozzuk a csomag1 nevű csomagot és a funkciókat itt hívjuk meg.

Tárolt csomag létrehozása:

```
create or replace package csomag1 as
  procedure UjPhone(pid number, ptype char, color char,
    prelease date, price number, condition char);
  procedure UjCustomer(identityN number, cname char, address char,
    salary number, birthday date, pid number);
  function GetName(getidentityN in integer) return varchar;
  function GetAvgPrice(typein char) return char;
end;
```

Tárolt csomag testének létrehozása:

```
create or replace package body csomag1 as
procedure UjPhone(pid number, ptype char, color char,
prelease date, price number, condition char) as
begin
insert into phone values(pid, ptype, color, prelease, price, condition);
end;
procedure UjCustomer(identityN number, cname char, address char,
salary number, birthday date, pid number) as perror exception;
begin
if salary <= 200000 then raise perror;
else
insert into customer values(identityN, cname, address, salary, birthday, pid);
end if;
exception
when perror then
dbms_output.put_line('A fizetés túl alacsony!');
end;
function GetName(getidentityN in integer) return varchar as
x customer.cname%type;
y integer;
begin
select count(*) into y from customer where getidentityN=identityN;
if(y<1) then
x:='Nem létezik a kód';
else
select cname into x from customer where getidentityN = identityN;
end if;
return x;
end;
function GetAvgPrice(typein char) return
char as
x integer;
y integer;
ni char(100):= 'Nem létezik ilyen típus';
begin
select count(*) into y from phone where ptype = typein;
if y >= 1 then
select avg(price) into x from phone where ptype = typein;
ni:='Az átlagár: '||x;
end if;
return ni;
end;
end;
```

Új telefon hozzáadása a csomaghoz:

```
begin
csomag1.UjPhone(7, 'LG', 'black', TO_DATE('2020.08.18','yyyy.MM.dd'), 240000, 'new');
end;
```

Új vásárló hozzáadása a csomaghoz:

```
begin
csomag1.ujcustomer(32145438, 'Más Jenő', 'Pécel, Péceli utca 84', 450000, TO_DATE('1989.03.12','yyyy.MM.dd'));
end;
```

A csomagból történő vásárló nevének lekérdezése.

A futtató kód:

```
select csomag1.getName(12398746) from dual;
```

A csomagból történő adott típusú telefonok átlagárának lekérdezése.

A futtató kód:

```
select csomag1.getAvgPrice('LG') from dual;
```

Automatikus kód beállítás létrehozása:

```
create sequence seq1
start with 8;
create trigger autokod before insert on phone for each row
begin
:new.id := seq1.nextval;
end;
```

Automatikus kód futtatása:

```
insert into phone(id, type, color, release, price, condition)
values(9, 'LG', 'green', TO_DATE('2014.03.08','yyyy.MM.dd'), 360000, 'new');
```

A három tábla tartalma:

Itt látható a létrehozott táblák módosítások utáni tartalma.

Phone table:

	id	type	color	release	price	condition
1	1	Samsung	red	20-JAN. -01	165000	new
2	2	IPhone	black	16-OKT. -21	280000	used
3	3	Huawei	white	21-MÁJ. -17	150000	new
4	4	LG	blue	22-AUG. -10	300000	new
5	5	LG	red	10-MÁRC. -14	210000	used
6	6	Huawei	blue	10-ÁPR. -14	210000	new

### Customer table:

1	12345678 Kiss Béla	Budapest, Pöttyös utca 69	1200000 90-ÁPR. -12	1
2	87654321 Nagy Ferenc	Miskolc, Kockás utca 420	3400000 94-DEC. -05	2
3	54637281 Horváth Bence	Gyál, Damjanich utca 10	2300000 97-JÚL. -17	4
4	32145698 Valaki Gergő	Eger, Egri utca 44	500000 00-OKT. -02	5

### Webshop table:

1	123 Phone Shop	1st street	<a href="https://www.phoneshop.com">https://www.phoneshop.com</a>	10-JÚL. -10
2	456 Mobile Buys	2nd street	<a href="https://www.mobilebuys.com">https://www.mobilebuys.com</a>	14-ÁPR. -14

Továbbá itt a naplózás táblának a tartalma, amely feljegyezte az összes táblákon végrehajtott változtatást.

### Naplózás tábla tartalma:

1	Beszúrás	6_Huawei	_blue	_10-ÁPR. -14_210000_new					22-MÁJ. -03 19.05.34,000000000
2	Módosítás	4_LG	_blue	_22-AUG. -10_300000_new	4_LG	_blue	_22-AUG. -10_400000_new		22-MÁJ. -03 19.48.46,000000000
3	Módosítás	6_Huawei	_blue	_10-ÁPR. -14_210000_new	6_Huawei	_blue	_10-ÁPR. -14_230000_new		22-MÁJ. -03 20.37.57,000000000
4	Módosítás	1_Samsung	_red	_20-JAN. -01_330000_new	1_Samsung	_red	_20-JAN. -01_350000_new		22-MÁJ. -03 20.37.57,000000000
5	Módosítás	2_iPhone	_black	_16-OKT. -21_280000_used	2_iPhone	_black	_16-OKT. -21_300000_used		22-MÁJ. -03 20.37.57,000000000
6	Módosítás	3_Huawei	_white	_21-MÁJ. -17_150000_new	3_Huawei	_white	_21-MÁJ. -17_170000_new		22-MÁJ. -03 20.37.57,000000000
7	Módosítás	4_LG	_blue	_22-AUG. -10_400000_new	4_LG	_blue	_22-AUG. -10_420000_new		22-MÁJ. -03 20.37.57,000000000
8	Módosítás	5_LG	_red	_11-FEBR. -12_680000_used	5_LG	_red	_11-FEBR. -12_700000_used		22-MÁJ. -03 20.37.57,000000000
9	Módosítás	3_Huawei	_white	_21-MÁJ. -17_170000_new	3_Huawei	_white	_84-JAN. -15_170000_new		22-MÁJ. -04 00.05.53,000000000