Adatbázisrendszerek 2 második beadandó

Készítette: Nagy Bence FVIQLY

Csoport: Hétfő 10:00

Feladatleírás:

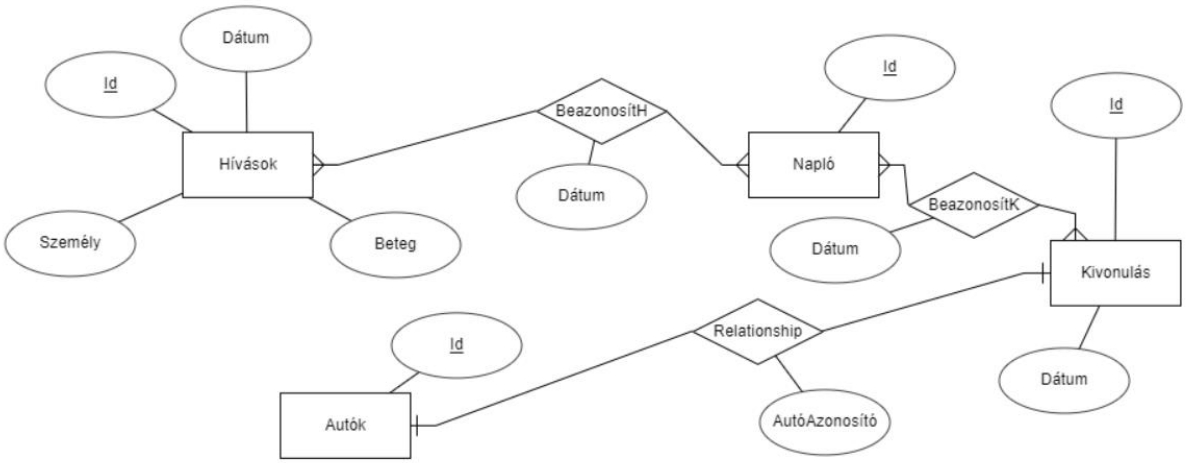
Készítsen nyilvántartást egy mentőszolgálat részére.

A feladat a DBMS szerver oldali elemek megtervezése és megvalósítása.

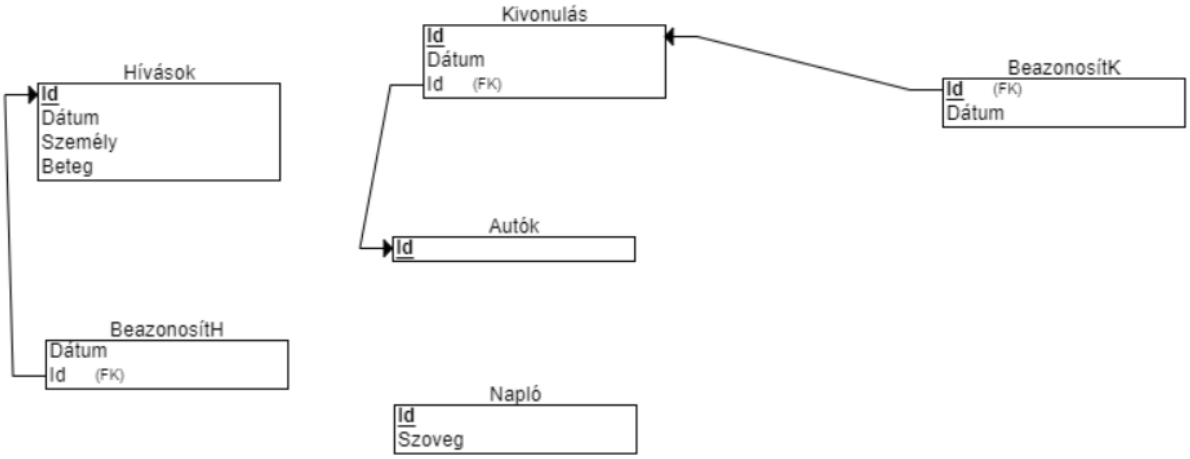
A feladat kötelező lépései:

- relációs modell megtervezése és megvalósítása
- PL/SQL csomag az alábbi funkciókkal:
    - tábla feltöltés véletlen elemekkel
    - tábla feltöltés állományból
    - új hívás felvitele
    - hívás kezelés adminisztrálása
    - kivonulási napló feltöltése
    - függvény, mely egy paraméterként adott naphoz megadja a hívások számát.
    - eddig hívások lekérdezése személyre vagy időszakra, betegre szűrve
- JDBC/ADO alkalmazás készítése a kidolgozott funkciók bemutatására.

ER model:



Relációs séma:



Java Kód:

```java
package DataBases2;

import java.sql.*;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Program {

    public static void main(String[] args) {
        try{
                Class.forName("oracle.jdbc.driver.OracleDriver");
                String url = "jdbc:oracle:thin:@193.6.5.58:1521:XE";
                Connection conn = DriverManager.getConnection(url, "H23_FVIQLY", "FVIQLY_A9");
                System.out.println( "Connection established" );
                Statement stmt;
                //Tábla létrehozása: Hívások:
                UploadTable(conn, "create table Hivasok(Id int primary key, Szemely varchar2(20), Beteg varchar2(20), Datum date)");
                //Tábla létrehozása: Autok:
                UploadTable(conn, "create table Autok(Id int primary key)");
                //Tábla létrehozása: Kivonulás:
                UploadTable(conn, "create table Kivonulas(Id int primary key,  Datum date, Aid int foreign key references Autok(Id))");
                //Tábla létrehozása: Naplo:
                UploadTable(conn, "create table Naplo(Id int primary key, szoveg varchar2(100))");
```

```
        //Tábla létrehozása: BeazonositK:
        UploadTable(conn, "create table BeazonositK(kId int foreign key references Kivonulas(Id), nId foreign key references Naplo(Id), Datum date)");
        //Tábla létrehozása: BeazonositH:
        UploadTable(conn, "create table BeazonositH(hId int foreign key references Hivasok(Id), nId foreign key references Naplo(Id), Datum date)");

        UploadRandom(conn);

        CreatePackage(conn);

        BufferedReader reader = new BufferedReader( new InputStreamReader(System.in));
        String name = reader.readLine();
        GetCallByName(conn, name);

        reader = new BufferedReader( new InputStreamReader(System.in));
        String beteg = reader.readLine();
        GetCallByPatient(conn, beteg);

        reader = new BufferedReader( new InputStreamReader(System.in));
        String datum = reader.readLine();
        GetCallByPatient(conn, datum);


        reader = new BufferedReader( new InputStreamReader(System.in));
        datum = reader.readLine();
```

```
48          datum = reader.readLine();
49          System.out.println(CountOfCallByDate(conn, datum));
50
51          UploadFromFile(conn, "HivasAdatok.txt");
52          UploadFromFile(conn, "KivonulasAdatok.txt");
53
54      }catch (Exception exception){
55          exception.printStackTrace();
56      }
57
58
59  ic static void UploadRandom(Connection conn) {
60  Upload(conn,"INSERT INTO Hivasok VALUES (1, 'Sárosi Péter', 'Kerekesné Mariann', '02/11/2022')");
61  Upload(conn,"INSERT INTO Hivasok VALUES (2, 'Berki Balázs', 'Farkas Valéria', '13/05/2010')");
62  Upload(conn,"INSERT INTO Hivasok VALUES (3, 'Szepesi Dávid', 'Vecser Béla', '21/07/2017')");
63  Upload(conn,"INSERT INTO Hivasok VALUES (4, 'Kerekes Sándor', 'Keresztes Sándor', '24/10/2006')");
64  Upload(conn,"INSERT INTO Hivasok VALUES (5, 'Fekete András', 'Kerekesné Mariann', '22/04/2013')");
65  Upload(conn,"INSERT INTO Hivasok VALUES (6, 'Mézes Endre', 'Édes Virás', '13/06/2018')");
66
67  Upload(conn, "INSERT INTO Autok VALUES (1)");
68  Upload(conn, "INSERT INTO Autok VALUES (2)");
69  Upload(conn, "INSERT INTO Autok VALUES (3)");
70  Upload(conn, "INSERT INTO Autok VALUES (4)");
71
72  Upload(conn,"INSERT INTO Kivonulas VALUES (1, '02/11/2022', 2)");
73  Upload(conn,"INSERT INTO Kivonulas VALUES (2,  '13/05/2010', 4)");
74  Upload(conn,"INSERT INTO Kivonulas VALUES (3,  '21/07/2017', 1)");
75  Upload(conn,"INSERT INTO Kivonulas VALUES (4, '24/10/2006', 3)");
76  Upload(conn,"INSERT INTO Kivonulas VALUES (5,  '22/04/2013', 3)");
77  Upload(conn,"INSERT INTO Kivonulas VALUES (6, '13/06/2018', 2)");
78
79
80
81  ic static void CreatePackage(Connection conn) {
82  try {
83          Statement stmt = conn.createStatement();
84          String createPackageQuery = "create or replace package Szolgalat is\r\n"
```

```
    Statement stmt = conn.createStatement();
    String createPackageQuery = "create or replace package Szolgalat is\r\n"
            + "    procedure FeltoltHivas;\r\n"
            + "    FUNCTION GetCountOfCalls( kdate DATE) RETURN NUMBER;\r\n"
            + "    TRIGGER trCall;\r\n"
            + "    procedure GetCallByName(name varchar2) as SYS_REFCURSOR;\r\n"
            + "    PROCEDURE GetCallByName(name VARCHAR2) AS SYS_REFCURSOR;\r\n"
            + "    PROCEDURE GetCallByDate(datum DATE) AS SYS_REFCURSOR;\r\n"
            + "    TRIGGER KivonulasAdmin;\r\n"
            + "    procedure UploadFromFile(filename varchar2)\r\n"
            + "    \r\n"
            + "end;\r\n"
            + "\r\n"
            + "create or replace package body Szolgalat is\r\n"
            + "create sequence HSEQ;\r\n"
            + "create or replace procedure FeltoltHivas(Id int, nev varchar2, beteg varchar2, datum date) is\r\n"
            + "MID int;\r\n"
            + "AID int;\r\n"
            + "begin\r\n"
            + "    select MAX(Id) INTO MID from Hivasok;\r\n"
            + "    select HSEQ.NEXTVAL into AID from DUAL;\r\n"
            + "    while AID <= MID loop\r\n"
            + "        select HSEQ.NEXTVAL into AID from DUAL;\r\n"
            + "    end loop;\r\n"
            + "    insert into Hivasok values(AID, nev, beteg, datum);\r\n"
            + "commit;\r\n"
            + "end;\r\n"
            + "\r\n"
            + "CREATE OR REPLACE PROCEDURE GetCallByName(name VARCHAR2) AS SYS_REFCURSOR\r\n"
            + "  kurzor SYS_REFCURSOR;\r\n"
            + "BEGIN\r\n"
            + "  OPEN kurzor FOR  SELECT * FROM Hivasok WHERE Hivasok.Szemely = name;\r\n"
            + "  RETURN v_cursor;\r\n"
            + "END;\r\n"
            + "\r\n"
            + "DECLARE\r\n"
            + "  KeresesEredmenyek SYS_REFCURSOR;\r\n"
            + "BEGIN\r\n"
```

```
+ "CREATE OR REPLACE PROCEDURE GetCallByName(name VARCHAR2) AS SYS_REFCURSOR\r\n"
+ "  kurzor SYS_REFCURSOR;\r\n"
+ "BEGIN\r\n"
+ "  OPEN kurzor FOR  SELECT * FROM Hivasok WHERE Hivasok.Szemely = name;\r\n"
+ "  RETURN v_cursor;\r\n"
+ "END;\r\n"
+ "\r\n"
+ "DECLARE\r\n"
+ "  KeresesEredmenyek SYS_REFCURSOR;\r\n"
+ "BEGIN\r\n"
+ "  KeresesEredmenyek := GetCallByName('ertek');\r\n"
+ "END;\r\n"
+ "\r\n"
+ "CREATE OR REPLACE PROCEDURE GetCallByPatient(pname VARCHAR2) AS SYS_REFCURSOR\r\n"
+ "  kurzor SYS_REFCURSOR;\r\n"
+ "BEGIN\r\n"
+ "  OPEN kurzor FOR  SELECT * FROM Hivasok WHERE Hivasok.Beteg = pname;\r\n"
+ "  RETURN v_cursor;\r\n"
+ "END;\r\n"
+ "\r\n"
+ "DECLARE\r\n"
+ "  KeresesEredmenyek SYS_REFCURSOR;\r\n"
+ "BEGIN\r\n"
+ "  KeresesEredmenyek := GetCallByName('ertek');\r\n"
+ "END;\r\n"
+ "\r\n"
+ "CREATE OR REPLACE PROCEDURE GetCallByDate(datum DATE) AS SYS_REFCURSOR\r\n"
+ "  kurzor SYS_REFCURSOR;\r\n"
+ "BEGIN\r\n"
+ "  OPEN kurzor FOR  SELECT * FROM Hivasok WHERE Hivasok.datum = datum;\r\n"
+ "  RETURN v_cursor;\r\n"
+ "END;\r\n"
+ "\r\n"
+ "--ezeket nem itt kell meghívni, majd a Java kódban packagenaem.procedurename(para)\r\n"
+ "DECLARE\r\n"
+ "  KeresesEredmenyek SYS_REFCURSOR;\r\n"
+ "BEGIN\r\n"
```

```
+ "\r\n"
+ "--ezeket nem itt kell meghívni, majd a Java kódban packagenaem.procedurename(para)\r\n"
+ "DECLARE\r\n"
+ "  KeresesEredmenyek SYS_REFCURSOR;\r\n"
+ "BEGIN\r\n"
+ "  KeresesEredmenyek := GetCallByDate('YYYY-MM-DD');\r\n"
+ "END;\r\n"
+ "\r\n"
+ "CREATE OR REPLACE FUNCTION GetCountOfCalls( kdate DATE) RETURN NUMBER\r\n"
+ "IS\r\n"
+ "  rekordok NUMBER;\r\n"
+ "BEGIN\r\n"
+ "  SELECT COUNT(*) INTO rekordok FROM Hivasok WHERE Hivasok.datum = kdate;\r\n"
+ "  RETURN v_count;\r\n"
+ "END;\r\n"
+ "\r\n"
+ "SELECT GetCountOfCalls(TO_DATE('datum', 'YYYY-MM-DD')) AS rekordok FROM dual;\r\n"
+ "\r\n"
+ "create sequence HASEQ;\r\n"
+ "CREATE OR REPLACE TRIGGER trCall AFTER INSERT OR DELETE OR UPDATE ON Hivasok\r\n"
+ "DECLARE\r\n"
+ "szoveg CHAR(100);\r\n"
+ "MID int;\r\n"
+ "AID int;\r\n"
+ "BEGIN\r\n"
+ "IF INSERTING THEN\r\n"
+ "szoveg := 'New call registered' || USER || TO_CHAR(sysdate,'YYYY.MM.DD');\r\n"
+ "END IF;\r\n"
+ "IF DELETING THEN\r\n"
+ "szoveg := 'Call deleted' || USER || TO_CHAR(sysdate,'YYYY.MM.DD');\r\n"
+ "END IF;\r\n"
+ "IF UPDATING THEN\r\n"
+ "szoveg := 'Call was modified' || USER || TO_CHAR(sysdate,'YYYY.MM.DD');\r\n"
+ "END IF;\r\n"
+ "\r\n"
+ "    select MAX(Id) INTO MID from Naplo;\r\n"
+ "    select HASEQ.NEXTVAL into AID from DUAL;\r\n"
```

```
+ "    select MAX(Id) INTO MID from Naplo;\r\n"
+ "    select HASEQ.NEXTVAL into AID from DUAL;\r\n"
+ "    while AID <= MID loop\r\n"
+ "        select HASEQ.NEXTVAL into AID from DUAL;\r\n"
+ "    end loop;\r\n"
+ "\r\n"
+ "INSERT INTO Naplo VALUES (AID, szoveg);\r\n"
+ "END;\r\n"
+ "\r\n"
+ "CREATE SEQUENCE KASEQ;\r\n"
+ "CREATE OR REPLACE TRIGGER KivonulasAdmin AFTER INSERT ON dolgozo\r\n"
+ "DECLARE\r\n"
+ "szoveg CHAR(100);\r\n"
+ "MID int;\r\n"
+ "AID int;\r\n"
+ "BEGIN\r\n"
+ "IF INSERTING THEN\r\n"
+ "szoveg := 'New operation registered' || USER || TO_CHAR(sysdate,'YYYY.MM.DD');\r\n"
+ "END IF;\r\n"
+ "\r\n"
+ "\r\n"
+ "    select MAX(Id) INTO MID from Naplo;\r\n"
+ "    select HASEQ.NEXTVAL into AID from DUAL;\r\n"
+ "    while AID <= MID loop\r\n"
+ "        select HASEQ.NEXTVAL into AID from DUAL;\r\n"
+ "    end loop;\r\n"
+ "\r\n"
+ "INSERT INTO Naplo VALUES (AID, szoveg);\r\n"
+ "END;\r\n"
+ "\r\n"
+ "grant read on directory Beadando2 to public;\r\n"
+ "create or replace procedure UploadFromFile(filename varchar2) as\r\n"
+ "  F1 UTL_FILE.FILE_TYPE;\r\n"
+ "  SOR varchar2(200);\r\n"
+ "  \r\n"
+ "    str VARCHAR2(100) := 'Apple,Banana,Cherry,Durian'; \r\n"
+ "  valaszto CHAR := ','; \r\n"
+ "  arr DBMS_SQL.VARCHAR2_TABLE;  \r\n"
```

```
+ "  valaszto CHAR := ','; \r\n"
+ "  arr DBMS_SQL.VARCHAR2_TABLE;  \r\n"
+ "  i INTEGER := 1;\r\n"
+ "  \r\n"
+ "begin\r\n"
+ "  F1 := UTL_FILE.FOPEN('Beadando2',filename,'R');\r\n"
+ "  begin\r\n"
+ "    loop\r\n"
+ "       UTL_FILE.GET_LINE(F1,sor,100);\r\n"
+ "        WHILE (INSTR(sor,delim) > 0) LOOP\r\n"
+ "    arr(i) := SUBSTR(sor,1,INSTR(sor,valaszto)-1);\r\n"
+ "    sor := SUBSTR(sor,INSTR(sor, valaszto)+1);\r\n"
+ "    i := i + 1;\r\n"
+ "  END LOOP;\r\n"
+ "  arr(i) := sor;  \r\n"
+ "  if filename = \"HivasAdatok.txt\" then\r\n"
+ "    FOR j IN 1..i LOOP\r\n"
+ "            insert into Hivasok values(arr(0), arr(1), arr(2), arr(3));\r\n"
+ "            commit;\r\n"
+ "         END LOOP;\r\n"
+ "  end if;\r\n"
+ "  \r\n"
+ "  if filename = \"KivonulasAdatok.txt\" then\r\n"
+ "    FOR j IN 1..i LOOP\r\n"
+ "            insert into Hivasok values(arr(0), arr(1), arr(2));\r\n"
+ "            commit;\r\n"
+ "         END LOOP;\r\n"
+ "  end if;\r\n"
+ "        \r\n"
```

```
                + "            commit;\r\n"
                + "         END LOOP;\r\n"
                + "  end if;\r\n"
                + "  \r\n"
                + "  if filename = \"KivonulasAdatok.txt\" then\r\n"
                + "    FOR j IN 1..i LOOP\r\n"
                + "            insert into Hivasok values(arr(0), arr(1), arr(2));\r\n"
                + "            commit;\r\n"
                + "         END LOOP;\r\n"
                + "  end if;\r\n"
                + "        \r\n"
                + "    end loop;\r\n"
                + "   exception\r\n"
                + "      when others then \r\n"
                + "         null;\r\n"
                + "  end;\r\n"
                + "  UTL_FILE.FCLOSE(F1);\r\n"
                + "end;\r\n"
                + "\r\n"
                + "set serveroutput on\r\n"
                + "\r\n"
                + "begin\r\n"
                + "  UploadFromFile;\r\n"
```

```
    public static void GetCallByName(Connection conn, String name) {
        try {
             Statement stmt = conn.createStatement();
             stmt.execute("Szolgalat.GetCallByName('"+name+"')");
             stmt.close();

         } catch (SQLException e) {
             e.printStackTrace();
         }
    }

    public static void GetCallByPatient(Connection conn, String patient) {
        try {
             Statement stmt = conn.createStatement();
             stmt.execute("Szolgalat.GetCallByPatient('"+patient+"')");
             stmt.close();

         } catch (SQLException e) {
             e.printStackTrace();
         }
    }

    public static void GetCallByDate(Connection conn, String Date) {
        try {
             Statement stmt = conn.createStatement();
             stmt.execute("Szolgalat.GetCallByDate('"+Date+"')");
             stmt.close();

         } catch (SQLException e) {
             e.printStackTrace();
         }
    }
```

```
    public static int CountOfCallByDate(Connection conn, String Date) {
        int result = 0;
        try {
            CallableStatement stmt = conn.prepareCall("Szolgalat.GetCallByDate('"+Date+"')");
              stmt.registerOutParameter(1, Types.INTEGER);
              stmt.execute();
              result = stmt.getInt(1);
             stmt.close();
         } catch (SQLException e) {
             e.printStackTrace();
         }
        return result;
    }

    public static void UploadFromFile(Connection conn, String filename) {
         try {
             Statement stmt = conn.createStatement();
             stmt.executeUpdate("Szolgalat.UploadFromFile('"+filename+"')");
             stmt.close();

         } catch (SQLException e) {
             e.printStackTrace();
         }
    }

    public static void Upload(Connection conn, String parancs) {

         try {
             Statement stmt = conn.createStatement();
             System.out.println(parancs);
             stmt.executeUpdate(parancs);
             stmt.close();

         } catch (SQLException e) {
             e.printStackTrace();
         }
    }
    public static void UploadFromFile(Connection conn, String filename) {
         try {
             Statement stmt = conn.createStatement();
             stmt.executeUpdate("Szolgalat.UploadFromFile('"+filename+"')");
             stmt.close();

         } catch (SQLException e) {
             e.printStackTrace();
         }
    }

    public static void Upload(Connection conn, String parancs) {

         try {
             Statement stmt = conn.createStatement();
             System.out.println(parancs);
             stmt.executeUpdate(parancs);
             stmt.close();

         } catch (SQLException e) {
             e.printStackTrace();
         }
    }

    public static void UploadTable(Connection conn, String parancs) {
        try {
             Statement stmt = conn.createStatement();
             System.out.println(parancs);
             stmt.executeUpdate(parancs);
             stmt.close();
        }catch(SQLException e) {
            e.printStackTrace();
        }

    }

}
```

PL/SQL kódok a csomagokhoz:

```
create or replace package Szolgalat is
    procedure FeltoltHivas;
    FUNCTION GetCountOfCalls( kdate DATE) RETURN NUMBER;
    TRIGGER trCall;
    procedure GetCallByName(name varchar2) as SYS_REFCURSOR;
    PROCEDURE GetCallByName(name VARCHAR2) AS SYS_REFCURSOR;
    PROCEDURE GetCallByDate(datum DATE) AS SYS_REFCURSOR;
    TRIGGER KivonulasAdmin;
    procedure UploadFromFile(filename varchar2)

end;

create or replace package body Szolgalat is
create sequence HSEQ;
create or replace procedure FeltoltHivas(Id int, nev varchar2, beteg varchar2, datum date) is
MID int;
AID int;
begin
    select MAX(Id) INTO MID from Hivasok;
    select HSEQ.NEXTVAL into AID from DUAL;
    while AID <= MID loop
        select HSEQ.NEXTVAL into AID from DUAL;
    end loop;
    insert into Hivasok values(AID, nev, beteg, datum);
commit;
end;

CREATE OR REPLACE PROCEDURE GetCallByName(name VARCHAR2) AS SYS_REFCURSOR
  kurzor SYS_REFCURSOR;
BEGIN
  OPEN kurzor FOR  SELECT * FROM Hivasok WHERE Hivasok.Szemely = name;
  RETURN v_cursor;
END;

DECLARE
  KeresesEredmenyek SYS_REFCURSOR;
```

```
DECLARE
  KeresesEredmenyek SYS_REFCURSOR;
BEGIN
  KeresesEredmenyek := GetCallByName('ertek');
END;

CREATE OR REPLACE PROCEDURE GetCallByPatient(pname VARCHAR2) AS SYS_REFCURSOR
  kurzor SYS_REFCURSOR;
BEGIN
  OPEN kurzor FOR  SELECT * FROM Hivasok WHERE Hivasok.Beteg = pname;
  RETURN v_cursor;
END;

DECLARE
  KeresesEredmenyek SYS_REFCURSOR;
BEGIN
  KeresesEredmenyek := GetCallByName('ertek');
END;

CREATE OR REPLACE PROCEDURE GetCallByDate(datum DATE) AS SYS_REFCURSOR
  kurzor SYS_REFCURSOR;
BEGIN
  OPEN kurzor FOR  SELECT * FROM Hivasok WHERE Hivasok.datum = datum;
  RETURN v_cursor;
END;

--ezeket nem itt kell megh?vni, majd a Java k?dban packagenaem.procedurename(para)
DECLARE
  KeresesEredmenyek SYS_REFCURSOR;
BEGIN
  KeresesEredmenyek := GetCallByDate('YYYY-MM-DD');
END;
```

```
CREATE OR REPLACE FUNCTION GetCountOfCalls( kdate DATE) RETURN NUMBER
IS
  rekordok NUMBER;
BEGIN
  SELECT COUNT(*) INTO rekordok FROM Hivasok WHERE Hivasok.datum = kdate;
  RETURN v_count;
END;

SELECT GetCountOfCalls(TO_DATE('datum', 'YYYY-MM-DD')) AS rekordok FROM dual;

create sequence HASEQ;
CREATE OR REPLACE TRIGGER trCall AFTER INSERT OR DELETE OR UPDATE ON Hivasok
DECLARE
szoveg CHAR(100);
MID int;
AID int;
BEGIN
IF INSERTING THEN
szoveg := 'New call registered' || USER || TO_CHAR(sysdate,'YYYY.MM.DD');
END IF;
IF DELETING THEN
szoveg := 'Call deleted' || USER || TO_CHAR(sysdate,'YYYY.MM.DD');
END IF;
IF UPDATING THEN
szoveg := 'Call was modified' || USER || TO_CHAR(sysdate,'YYYY.MM.DD');
END IF;

    select MAX(Id) INTO MID from Naplo;
    select HASEQ.NEXTVAL into AID from DUAL;
    while AID <= MID loop
        select HASEQ.NEXTVAL into AID from DUAL;
    end loop;

INSERT INTO Naplo VALUES (AID, szoveg);
END;
```

```
CREATE SEQUENCE KASEQ;
CREATE OR REPLACE TRIGGER KivonulasAdmin AFTER INSERT ON dolgozo
DECLARE
szoveg CHAR(100);
MID int;
AID int;
BEGIN
IF INSERTING THEN
szoveg := 'New operation registered' || USER || TO_CHAR(sysdate,'YYYY.MM.DD');
END IF;


    select MAX(Id) INTO MID from Naplo;
    select HASEQ.NEXTVAL into AID from DUAL;
    while AID <= MID loop
        select HASEQ.NEXTVAL into AID from DUAL;
    end loop;

INSERT INTO Naplo VALUES (AID, szoveg);
END;

grant read on directory Beadando2 to public;
create or replace procedure UploadFromFile(filename varchar2) as
  F1 UTL_FILE.FILE_TYPE;
  SOR varchar2(200);

    str VARCHAR2(100) := 'Apple,Banana,Cherry,Durian';
  valaszto CHAR := ',';
  arr DBMS_SQL.VARCHAR2_TABLE;
  i INTEGER := 1;
```

```
begin
  F1 := UTL_FILE.FOPEN('Beadando2',filename,'R');
  begin
    loop
      UTL_FILE.GET_LINE(F1,sor,100);
       WHILE (INSTR(sor,delim) > 0) LOOP
    arr(i) := SUBSTR(sor,1,INSTR(sor,valaszto)-1);
    sor := SUBSTR(sor,INSTR(sor, valaszto)+1);
    i := i + 1;
  END LOOP;
  arr(i) := sor;
  if filename = "HivasAdatok.txt" then
    FOR j IN 1..i LOOP
            insert into Hivasok values(arr(0), arr(1), arr(2), arr(3));
            commit;
          END LOOP;
  end if;

  if filename = "KivonulasAdatok.txt" then
    FOR j IN 1..i LOOP
            insert into Hivasok values(arr(0), arr(1), arr(2));
            commit;
          END LOOP;
  end if;

    end loop;
   exception
      when others then
        null;
  end;
  UTL_FILE.FCLOSE(F1);
end;

set serveroutput on
```

```
 set serveroutput on

 begin
   UploadFromFile;
 end;

 end Szolgalat;
```

SQL kódok szeparált formája az SQLscripts mappában