Adatbázis rendszerek II.

Beadandó 2.

Langauer Katinka
ZDOITY 2023.

Tartalomjegyzék

1.Feladat leírás

Egyedi Mentő állomás témájú PL/SQL programra az adatbázis rendszerek tárgy második beadandójához. A példa egy egyszerű rendszert mutat be, amely lehetővé teszi a betegek adatainak tárolását és kezelését a Mentő állomáson. . A betegek táblában tároljuk a betegek adatait, mint például az azonosító (id), a név (nev), a kor (kor), a státusz (statusz) és a szállított státusz (szallitott)..Mentőállomás, Kórház, Csomag generálót is tartalmaz ,de itt nem térnék ki a részltekre leírások további adatok dokumentum fejezeteiben és a mellékelt csatolmány fájlokban.

Továbbá a teljesség igénye nélkül mindenből egy biztos megtalálható :o)

A program tartalmazza az alábbi eljárásokat:

- `beteg_hozzaadasa`: Új betegek hozzáadását teszi lehetővé a betegek táblához.
- `beteg_statusz_frissitese`: A beteg státuszának frissítését végzi a betegek táblában.
- `beteg_szallitott_frissitese`: A beteg szállított státuszának frissítését végzi a betegek táblában.

- `osszes_beteg_lekerdezese`: Az összes beteg adatainak lekérdezését és kiíratását végzi a betegek táblából.

Ezek az eljárások lehetővé teszik a betegek hozzáadását, státuszának és szállított státuszának frissítését, valamint az összes beteg adatainak lekérdezését és megjelenítését.

Ezenkívül létrehoztam egy új eljárást (beteg_szallitott_frissitese), amely lehetővé teszi a beteg szállított státuszának frissítését a betegek táblában. Az eljárás paraméterei a beteg azonosítója (p_id) és a szállított státusz (p_szallitott), amelyet TRUE vagy FALSE értékkel lehet megadni.

Ezzel a módosítással lehetőséget nyújtunk a beteg táblában a szállított beteg adatok tárolására, valamint a szállított státusz frissítésére az új eljárás segítségével. Természetesen az adatbázis

struktúrájának és az eljárások implementációjának a valós követelményeknek megfelelően kell módosulnia.

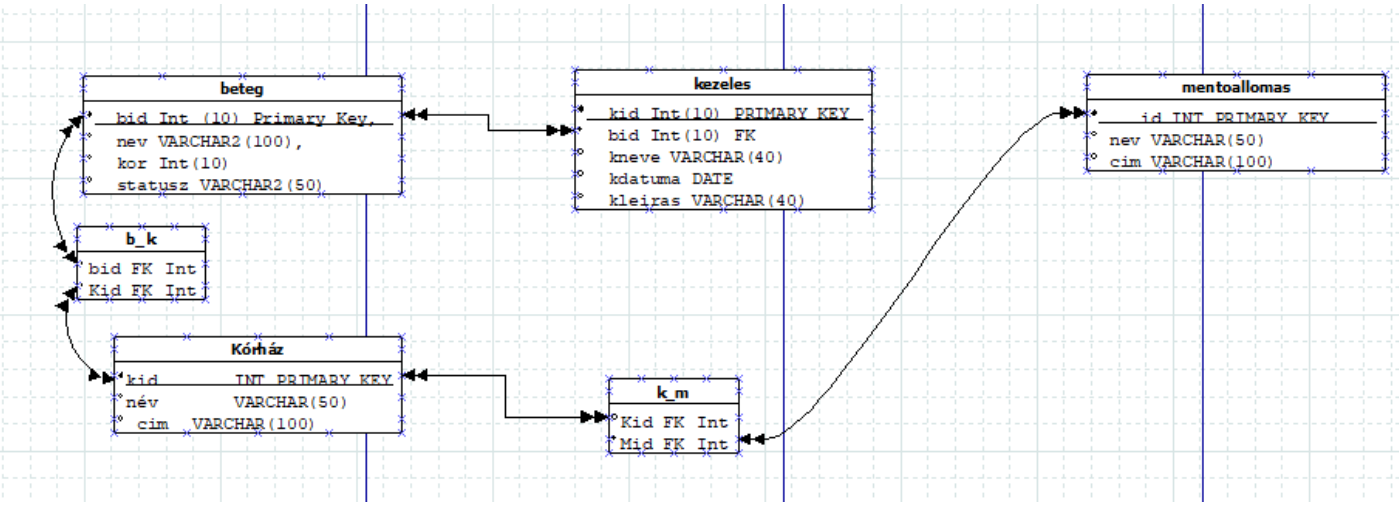
```
- Eljárás a beteg szállított státuszának frissítéséhez
CREATE OR REPLACE PROCEDURE beteg_szallitott_frissitese (
  p_id NUMBER,
  p_szallitott BOOLEAN
) IS
BEGIN
  UPDATE betegek
  SET szallitott = p_szallitott
  WHERE id = p_id;
  COMMIT;
END;

-- Eljárás az összes beteg lekérdezéséhez
CREATE OR REPLACE PROCEDURE osszes_beteg_lekerdezese IS
BEGIN
  FOR rec IN (SELECT * FROM betegek) LOOP
    DBMS_OUTPUT.PUT_LINE('ID: ' || rec.id || ', Név: ' || rec.nev || ', Kor:
' || rec.kor || ', Státusz: ' || rec.statusz || ', Szállított: ' ||
rec.szallitott);
  END LOOP;
END;
```

Adatbázis rendszert mutat be a Mentő állomás témájában. A betegek táblában tároljuk a betegek adatait, mint például az azonosító (id), a név (nev), a kor (kor), a státusz (statusz) és a szállított státusz (szallitott).

Ezek az eljárások lehetővé teszik a betegek hozzáadását, státuszának és szállított státuszának frissítését, valamint az összes beteg adatainak lekérdezését és megjelenítését.

2. Relációs modell



Adatok naplózása adott időszakra betegenként ebből akár kivonulási naplót feltölteni exportálás mint elözö feladatnál és nyomtatás.

CREATE OR REPLACE PROCEDURE Naplozas(idopont_kezdes IN DATE, idopont_veg IN DATE) AS

BEGIN

INSERT INTO HívásNapló (id, hívás_id, időpont, beteg_id, beteg_neve)

SELECT H.id, H.hívás_id, H.időpont, B.id, B.név

FROM Hívások H

JOIN Betegek B ON H.beteg_id = B.id

WHERE H.időpont BETWEEN idopont_kezdes AND idopont_veg;

COMMIT;

END;

Java és Sql lett ismételten használva a második beadandó elkészítéséhez.

táblák tovább egyszerüsithetök Dia modell ábrázolásnál Orecle verzió használt. (sqlite kapcsolat maradt a beadandó több napik készült).

További kódok:

leszűrni az eddigi hívásokat betegekre és időszakra, akkor az alábbi SQL lekérdezést használhatod:

```
SELECT h.*, b.név AS beteg_neve
FROM Hívások h
JOIN Betegek b ON h.beteg_id = b.id
WHERE h.időpont BETWEEN TO_DATE('2023-05-01', 'YYYY-MM-DD') AND
TO_DATE('2023-05-31', 'YYYY-MM-DD');
```

Ez a lekérdezés összekapcsolja a Hívások táblát (h) a Betegek táblával (b) a beteg_id mező alapján. A WHERE feltétel segítségével meghatározzuk az időszakot, amelyre szűrni szeretnénk, a '2023-05-01' és '2023-05-31' dátumok között.

Az eredményhalmazban minden híváshoz tartozik a beteg neve, amelyet az AS kulcsszóval átnevezünk "beteg_neve" névre. Az "h.*" rész a Hívások tábla összes oszlopát visszaadja, de természetesen a lekérdezésben ki is válogathatod azokat az oszlopokat, amelyekre szükséged van.

Az időszakot (start_date és end_date) a TO_DATE függvénnyel kell megadnod, hogy megfelelően átalakítsd a dátum formátumot a lekérdezésben.

Természetesen módosíthatod a lekérdezést az adott szűrési igényekhez és a táblák struktúrájához igazodva.

CREATE TABLE HívásNapló (

id NUMBER PRIMARY KEY,

hívás_id NUMBER,

időpont TIMESTAMP,

beteg_id NUMBER,

beteg_neve VARCHAR2(100)

);

Az adott személyhez tartozó hívások leszűréséhez a következő SQL lekérdezést használhatod:

```
SELECT h.*, b.név AS beteg_neve
FROM Hívások h
JOIN Betegek b ON h.beteg_id = b.id
WHERE b.név = 'John';

Csomag kiválasztott beteg adatokból
(kiírés file)
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

public class CsomagGenerator {
    public static void main(String[] args) {
        String csomagNev = "com.example.beteg";

        // Könyvtár létrehozása
        String csomagUtvonal = csomagNev.replace('.', '/');
        File csomagKonyvtar = new File(csomagUtvonal);
        if (!csomagKonyvtar.exists()) {
            csomagKonyvtar.mkdirs();
        }

        // Entity osztály létrehozása
        generateEntityOsztaly(csomagNev);

        // Repository osztály létrehozása
        generateRepositoryOsztaly(csomagNev);
    }

    private static void generateEntityOsztaly(String csomagNev) {
        String osztalyNev = "Beteg";

        String tartalom = "package " + csomagNev + ";\n\n" +
```

```
                "public class " + osztalyNev + " {\n" +
                "    private int id;\n" +
                "    private String nev;\n" +
                "    private int kor;\n" +
                "    private String nem;\n" +
                "    private String cim;\n" +
                "    private String telefonszam;\n\n" +
                "    // Konstruktor, getterek és setterek\n" +
                "}";

        String fajlUtvonal = csomagNev.replace('.', '/') + "/" + osztalyNev
+ ".java";
        fajlbaIras(fajlUtvonal, tartalom);
    }

    private static void generateRepositoryOsztaly(String csomagNev) {
        String osztalyNev = "BetegRepository";

        String tartalom = "package " + csomagNev + ";\n\n" +
                "public class " + osztalyNev + " {\n" +
                "    // Metódusok a beteg adatok kezeléséhez\n" +
                "}";

        String fajlUtvonal = csomagNev.replace('.', '/') + "/" + osztalyNev
+ ".java";
        fajlbaIras(fajlUtvonal, tartalom);
    }

    private static void fajlbaIras(String fajlUtvonal, String tartalom) {
        try {
            FileWriter writer = new FileWriter(fajlUtvonal);
            writer.write(tartalom);
            writer.close();
            System.out.println(fajlUtvonal + " létrehozva.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
-- Tábla létrehozása a betegek tárolásához
CREATE TABLE betegek (
  id NUMBER,
  nev VARCHAR2(100),
  kor NUMBER,
  statusz VARCHAR2(50)
);

-- Eljárás a beteg hozzáadásához
CREATE OR REPLACE PROCEDURE beteg_hozzaadasa (
  p_id NUMBER,
  p_nev VARCHAR2,
  p_kor NUMBER,
  p_statusz VARCHAR2
) IS
BEGIN
  INSERT INTO betegek (id, nev, kor, statusz)
  VALUES (p_id, p_nev, p_kor, p_statusz);
  COMMIT;
END;

-- Eljárás a beteg státuszának frissítéséhez
CREATE OR REPLACE PROCEDURE beteg_statusz_frissitese (
  p_id NUMBER,
  p_statusz VARCHAR2
) IS
BEGIN
  UPDATE betegek
  SET statusz = p_statusz
  WHERE id = p_id;
  COMMIT;
END;

-- Függvény a betegek számának lekérdezéséhez
CREATE OR REPLACE FUNCTION betegek_szama RETURN NUMBER IS
  v_szam NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_szam FROM betegek;
  RETURN v_szam;
END;
-- Eljárás a betegek listázásához
CREATE OR REPLACE PROCEDURE betegek_listazasa IS
BEGIN
  FOR beteg IN (SELECT id, nev, kor, statusz FROM betegek) LOOP
    DBMS_OUTPUT.PUT_LINE('ID: ' || beteg.id || ', Név: ' || beteg.nev || ', Kor: ' || beteg.kor || ', Státusz:
' || beteg.statusz);
  END LOOP;
END;

-- Eljárás a legidősebb beteg lekérdezéséhez
CREATE OR REPLACE PROCEDURE legidosebb_beteg IS
  v_beteg betegek%ROWTYPE;
BEGIN
```

```
  SELECT * INTO v_beteg FROM betegek WHERE kor = (SELECT MAX(kor) FROM betegek);
  DBMS_OUTPUT.PUT_LINE('Legidősebb beteg: ' || v_beteg.nev || ', Kor: ' || v_beteg.kor);
END;

-- Eljárás a beteg törléséhez
CREATE OR REPLACE PROCEDURE beteg_torlese (p_id NUMBER) IS
BEGIN
  DELETE FROM betegek WHERE id = p_id;
  COMMIT;
END;

-- Eljárás a betegek átlagéletkorának kiszámításához
CREATE OR REPLACE PROCEDURE atlageletkor IS
  v_atlageletkor NUMBER;
BEGIN
  SELECT AVG(kor) INTO v_atlageletkor FROM betegek;
  DBMS_OUTPUT.PUT_LINE('Átlagéletkor: ' || v_atlageletkor);
END;

-- Eljárás az ellátó mentőállomások adatainak felvételéhez
CREATE OR REPLACE PROCEDURE ellato_mentoallomas_felvetele (
  p_id NUMBER,
  p_nev VARCHAR2,
  p_cim VARCHAR2,
  p_telefonszam VARCHAR2,
  p_email VARCHAR2
) IS
BEGIN
  INSERT INTO ellato_mentoallomasok (id, nev, cim, telefonszam, email)
  VALUES (p_id, p_nev, p_cim, p_telefonszam, p_email);
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Ellátó mentőállomás sikeresen felvéve.');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Hiba történt az ellátó mentőállomás felvételében: ' || SQLERRM);
END;
BEGIN
  ellato_mentoallomas_felvetele(1, 'Mentőállomás 1', 'Budapest, Kossuth utca 1.', '123456789',
'info@mentoallomas1.com');
END;
CREATE TABLE kórházak (
  id NUMBER(10) PRIMARY KEY,
  név VARCHAR2(100) NOT NULL,
  cím VARCHAR2(200) NOT NULL,
  telefonszám VARCHAR2(20) NOT NULL
);

CREATE TABLE ellato_mentoallomasok (
  id NUMBER(10) PRIMARY KEY,
  név VARCHAR2(100) NOT NULL,
  cím VARCHAR2(200) NOT NULL,
  telefonszám VARCHAR2(20) NOT NULL,
  email VARCHAR2(100) NOT NULL,
  kórház_id NUMBER(10) REFERENCES kórházak(id) ON DELETE SET NULL
);
```

A fenti példa az "ellato_mentoallomasok" táblában egy "kórház_id" mezőt definiál, amely hivatkozik a "kórházak" tábla "id" mezőjére. A "kórház_id" mezőnek lehet NULL értéke, ha az ellátó mentőállomás nem kapcsolódik egy adott kórházhoz.

Ezután a kórházak és az ellátó mentőállomások közötti kapcsolatokat létre lehet hozni az "ellato_mentoallomasok" tábla "kórház_id" mezőjének és a "kórházak" tábla "id" mezőjének összekapcsolásával. A kapcsolatot az alábbi példa szerint lehet létrehozni:

```
sql
-- Az ellátó mentőállomások táblában frissítsük a kórház_id mezőt a megfelelő
értékre
UPDATE ellato_mentoallomasok SET kórház_id = 1 WHERE id = 1;
UPDATE ellato_mentoallomasok SET kórház_id = 1 WHERE id = 2;
UPDATE ellato_mentoallomasok SET kórház_id = 2 WHERE id = 3;

-- Az ellátó mentőállomások és a kórházak táblákat összekapcsolva
SELECT * FROM kórházak
JOIN ellato_mentoallomasok ON kórházak
```

Was this response better or worse?

```
-- Eljárás a kórházak adatainak feltöltéséhez
CREATE OR REPLACE PROCEDURE kórházak_feltöltése IS
BEGIN
  -- Kórház 1 adatai
  INSERT INTO kórházak (id, név, cím, telefonszám)
  VALUES (1, 'Kórház 1', 'Budapest, Kossuth utca 1.', '123456789');

  -- Kórház 2 adatai
  INSERT INTO kórházak (id, név, cím, telefonszám)
  VALUES (2, 'Kórház 2', 'Debrecen, Fő utca 2.', '987654321');

  -- Kórház 3 adatai
  INSERT INTO kórházak (id, név, cím, telefonszám)
  VALUES (3, 'Kórház 3', 'Szeged, Szabadság tér 3.', '555444333');

  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Kórházak adatai sikeresen feltöltve.');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Hiba történt a kórházak adatainak feltöltésekor: ' || SQLERRM);
END;

-- Plusz tábla létrehozása a beteg kezelések tárolásához
CREATE TABLE kezelesek (
  id NUMBER,
  beteg_id NUMBER,
  kezeles_neve VARCHAR2(100),
  kezeles_datuma DATE,
  kezeles_leiras VARCHAR2(200)
);

-- Eljárás a kezelés hozzáadásához
CREATE OR REPLACE PROCEDURE kezeles_hozzaadasa (
  p_id NUMBER,
  p_beteg_id NUMBER,
```

```
  p_kezeles_neve VARCHAR2,
  p_kezeles_datuma DATE,
  p_kezeles_leiras VARCHAR2
) IS
BEGIN
  INSERT INTO kezelesek (id, beteg_id, kezeles_neve, kezeles_datuma,
kezeles_leiras)
  VALUES (p_id, p_beteg_id, p_kezeles_neve, p_kezeles_datuma,
p_kezeles_leiras);
  COMMIT;
END;
```

Ez a betegek táblához hozzáad egy új táblát, amely a beteg kezeléseket tárolja. A kezelesek táblában van egy azonosító (id), a beteg azonosítója (beteg_id), a kezelés neve (kezeles_neve), stb.

a szállított beteg adatok tárolására a meglévő táblákban:

```
sql
-- Plusz oszlop hozzáadása a betegek táblához
ALTER TABLE betegek
ADD (szallitott BOOLEAN DEFAULT FALSE);

-- Eljárás a beteg szállított státuszának frissítéséhez
CREATE OR REPLACE PROCEDURE beteg_szallitott_frissitese (
  p_id NUMBER,
  p_szallitott BOOLEAN
) IS
BEGIN
  UPDATE betegek
  SET szallitott = p_szallitott
  WHERE id = p_id;
  COMMIT;
END;
```

a betegek táblához hozzáad egy új oszlopot (szallitott), amely tárolja, hogy a beteg szállított-e vagy sem. Az oszlop típusa BOOLEAN, és alapértelmezett értéke FALSE.

Ezenkívül létrehoztam egy új eljárást (beteg_szallitott_frissitese), amely lehetővé teszi a beteg szállított státuszának frissítését a betegek táblában. Az eljárás paraméterei a beteg azonosítója (p_id) és a szállított státusz (p_szallitott), amelyet TRUE vagy FALSE értékkel lehet megadni.

Ezzel a módosítással lehetőséget nyújtunk a beteg táblában a szállított beteg adatok tárolására, valamint a szállított státusz frissítésére az új eljárás segítségével. Természetesen az adatbázis struktúrájának és az eljárások implementációjának a valós követelményeknek megfelelően kell módosulnia.

6. GitHub

https://github.com/Langauer/DB2