PÉLDATÁR

NEMZETI ADÓ- ÉS VÁMHIVATAL
Képzési, Egészségügyi és Kulturális Intézete

PL/SQL ALAPOZÓ TANFOLYAM

1.0 verzió

2017

Készítette:

Ender János
Pillér Kft

Lektorálta:

Parditkáné Jakubovics Erzsébet
NAV INIT

Lezárva:
Budapest, 2012. 07. 02.

TARTALOMJEGYZÉK

I. DOLGOZÓK NYILVÁNYTARTÁSA

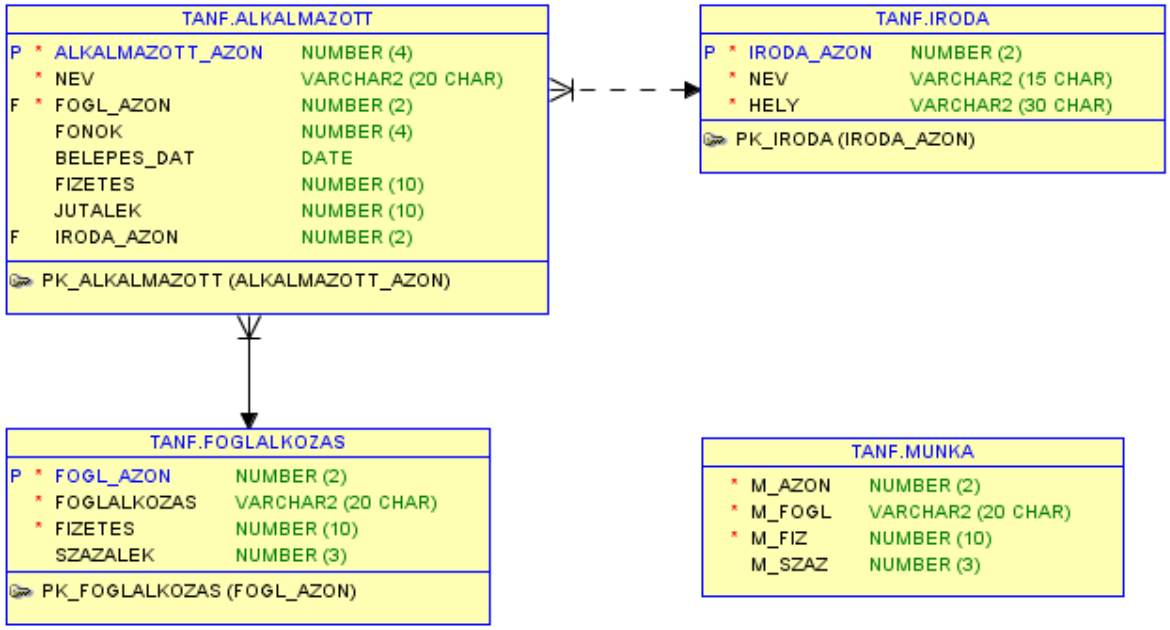
I. 1. Adatmodell

Tanfolyami adatmodellünkben az alábbi adatokat szeretnénk nyilvántartani dolgozókról. Alkalmazott neve, foglalkozása, főnöke, belépés dátuma, fizetése, jutaléka, melyik irodában dolgozik, hol van az iroda, mennyi a foglalkozására jellemző fizetés, ennél a foglalkozásnál a jellemző fizetés hány százalékát kapja egy kezdő.

Ha ezeket az adatokat elemezzük, akkor három részre lehet őket csoportosítani, az alkalmazottra jellemző adatok, az irodára jellemző adatok, és a foglalkozásra jellemző adatok. Ennek megfelelően három táblát hozunk létre, amelyek kapcsolatban állnak egymással. Fontos annak a meghatározása, hogy a főnök nem az iroda főnöke, hanem embereké, ezért az adatmodellben az alkalmazott jellemzőjeként vesszük fel.

Hozzunk létre egy munkatáblát is a foglalkozásoknak. Az ide bekerülő módosításokat bizonyos időközönként fogjuk felhasználni arra, hogy karbantartsuk a foglalkozások jellemzőit.

Az alábbi diagrammal lehet leírni az adatmodellt:



A táblák létrehozó utasításai és kezdeti feltöltésük az III. 1 mellékletben találhatók.

A táblák létrehozásánál és feltöltésénél fontos a sorrend a foreign key kapcsolat miatt. Az alkalmazott táblát csak a másik két tábla létrehozása és feltöltése után lehet feltölteni.

I. 2. Névtelen blokk, host változók kezelése

Készítsünk egy névtelen blokkot, ami megállapítja az ALKALMAZOTT táblából, mennyi a legnagyobb, legkisebb és átlagos fizetés! Mennyi az eltérés közöttük? Írjuk ki SQL*PLUS hozzárendelt változók segítségével az eredményt!

```
--HOST változók létrehozása
VARIABLE g_max NUMBER
VARIABLE g_min NUMBER
VARIABLE g_atl NUMBER
VARIABLE g_max_atl NUMBER
VARIABLE g_min_atl NUMBER
VARIABLE g_max_min NUMBER

-- Névtelen blokk
DECLARE
   v_max     NUMBER;
   v_min     NUMBER;
   v_atl     NUMBER;
   v_max_atl NUMBER;
   v_min_atl NUMBER;
   v_max_min NUMBER;
BEGIN
  SELECT MAX(fizetes),MIN(fizetes),AVG(fizetes)
    INTO v_max, v_min, v_atl
    FROM alkalmazott;

    v_max_atl := v_max - v_atl;
    v_min_atl := v_atl - v_min;
    v_max_min := v_max - v_min;

    -- Értékadás a HOST változóknak
    :g_max := v_max;
    :g_min := v_min;
    :g_atl := v_atl;
    :g_max_atl := v_max_atl;
    :g_min_atl := v_min_atl;
    :g_max_min := v_max_min;
END;
/
-- HOST változók kiírása
PRINT g_max
PRINT g_min
PRINT g_atl
PRINT g_max_atl
PRINT g_min_atl
PRINT g_max_min
```

I. 3. Program készítés, munkatáblából adatfeltöltés

Készítsünk programot (tárolt eljárást), amely a MUNKA tábla adatait összefésüli a FOGLALKOZAS tábla adataival, vagyis a létezőket karbantartja, a nem létezőket beszúrja a táblába.

1. változat: UPDATE – INSERT

A feladatot úgy lehet megoldani, hogy egy kurzorral végigmegyünk a munka táblán. Minden rekordnál meg kell vizsgálni, van-e ilyen rekord, ha igen, akkor UPDATE, ha nem, akkor INSERT következik.

Két megoldást mutatunk be, az elsőben csoportfüggvénnyel ellenőrizzük, hogy van-e rekord, majd IF – ELSE kiértékeléssel döntjük el, milyen utasítással folytatjuk.

```
CREATE OR REPLACE PROCEDURE fogl_upd AS
  db   NUMBER;
BEGIN

  FOR r IN (SELECT * FROM munka)
  LOOP
     BEGIN
        SELECT COUNT(*)
          INTO db
          FROM foglalkozas
         WHERE fogl_azon = r.m_azon;

        IF db = 0 THEN
             INSERT INTO foglalkozas
                       (fogl_azon,foglalkozas,fizetes,szazalek)
               VALUES (r.m_azon,r.m_fogl,r.m_fiz,r.m_szaz);
        ELSE
             UPDATE foglalkozas
                SET foglalkozas = r.m_fogl,
                    fizetes     = r.m_fiz,
                    szazalek    = r.m_szaz
              WHERE fogl_azon = r.m_azon;
        END IF;
        COMMIT;
     end;
   end loop;

END fogl_upd;
/
```

A második esetben megpróbáljuk beszúrni. Mivel egyedi kulcs az azonosító, hibát dob, ha ugyanolyan azonosítójú rekordot akarunk beszúrni. A hibaágon UPDATE-lünk.

```
CREATE OR REPLACE PROCEDURE fogl_upd AS
BEGIN

  FOR r IN (SELECT * FROM munka)
  LOOP
     BEGIN

         INSERT INTO foglalkozas
                     (fogl_azon,foglalkozas,fizetes,szazalek)
              VALUES (r.m_azon,r.m_fogl,r.m_fiz,r.m_szaz);
        COMMIT;

     EXCEPTION
        WHEN DUP_VAL_ON_INDEX THEN
             UPDATE foglalkozas
                SET foglalkozas = r.m_fogl,
                    fizetes     = r.m_fiz,
                    szazalek    = r.m_szaz
              WHERE fogl_azon = r.m_azon;
             COMMIT;
     END;
  END LOOP;

END fogl_upd;
/
```

2. változat: MERGE

A MERGE utasítás használatával nem kell külön foglalkoznunk, van-e ilyen rekord, a MERGE maga megvizsgálja, melyik ágon kell mennie, és végrehajtja a karbantartást.

```
CREATE OR REPLACE PROCEDURE fogl_merge AS
BEGIN

  MERGE INTO foglalkozas f
       USING munka m
          ON (fogl_azon = m_azon)
    WHEN MATCHED THEN
       UPDATE SET foglalkozas=m_fogl,
                  fizetes=m_fiz,
                  szazalek=m_szaz
   WHEN NOT MATCHED THEN
      INSERT (fogl_azon,foglalkozas,fizetes,szazalek)
        VALUES (m_azon,m_fogl,m_fiz,m_szaz);
   COMMIT;

END fogl_merge;
/
```

I. 4. Csomag készítés

Hiba naplózás

Készítsünk egy csomagot, amely tartalmaz egy önálló procedúrát, amely hibát naplóz! A napló tábla létrehozó szkriptje az alábbi:

```
CREATE TABLE HIBANAPLO
 (
  DATUM      DATE               DEFAULT SYSDATE,
  FELHASZN   VARCHAR2(20 CHAR),
  PROGRAM    VARCHAR2(30 CHAR),
  HIBAUZENET VARCHAR2(4000 CHAR)
 );
```

A táblába bekerül a bejegyzés dátuma, ezt a tábla mező DEFAULT SYSDATE része elintézi. Be kell még jegyezni a bejelentkezett felhasználó azonosítóját, a hibát kiváltó program nevét (ez bemenő paraméter), és a hibaüzenetet. A csomag betöltésekor állapítsa meg a felhasználót, és ezt tárolja egy lokális package változóban! A változó feltöltéséhez használjuk a package egyszer futó blokkját! A felhasználót a USER nevű rendszerváltozóból lehet lekérdezni. A hibaüzenetet az SQLERRM értékéből lehet meghatározni. (A hibakódot az SQLCODE adja vissza numerikus értékként, de az SQLERRM is tartalmazza a szövegében.)

```
CREATE OR REPLACE PACKAGE NAPLO AS
   PROCEDURE naplo_ir(p_prg VARCHAR2);
END NAPLO;
/
CREATE OR REPLACE PACKAGE BODY NAPLO AS
  gv_user VARCHAR2(20);

  PROCEDURE naplo_ir(p_prg VARCHAR2) IS
  PRAGMA AUTONOMOUS_TRANSACTION;
     hiba VARCHAR2(4000);
  BEGIN
     hiba := sqlerrm;
     INSERT INTO hibanaplo (felhaszn, program, hibauzenet)
          VALUES (gv_user, p_prg, hiba);
     COMMIT;
  END naplo_ir;

BEGIN
   SELECT user  INTO gv_user FROM dual;
END NAPLO;
/
```

Próbáljuk ki 0-val osztással:

```
DECLARE
   x NUMBER;
BEGIN
   x := 1/0;
   EXCEPTION
     WHEN OTHERS THEN
        naplo.naplo_ir('TESZT');
END;
/
```

Az eredmény a táblában:

DATUM	FELH	PROGRAM	HIBAUZENET
2012.06.08	TANF	TESZT	ORA-01476: divisor is equal to zero

Iroda neve

Készítsünk egy IRODA_CSOMAG csomagot, amiben egy függvény az iroda azonosítója alapján visszaadja a nevét! Ha nem megfelelő iroda azonosítót kap, akkor a neve helyett adja vissza a „Nem létező iroda” szöveget! Ha üres az iroda azonosító, akkor NULL értéket adjon vissza!

```
CREATE OR REPLACE PACKAGE iroda_csomag AS
     FUNCTION iroda_nev(p_iroda in iroda.iroda_azon%TYPE)
                    RETURN VARCHAR2;
END iroda_csomag;
/
CREATE OR REPLACE PACKAGE BODY iroda_csomag AS

  FUNCTION iroda_nev(p_iroda IN iroda.iroda_azon%TYPE)
                RETURN VARCHAR2 AS
     ret_val IRODA.NEV%TYPE;
  BEGIN
    IF p_iroda IS NULL THEN
       RETURN NULL;
    END IF;

    SELECT nev INTO ret_val
    FROM iroda WHERE iroda_azon = p_iroda;

    return ret_VAL;

  EXCEPTION WHEN NO_DATA_FOUND THEN
    return 'Nem létező iroda';
  END iroda_nev;
END iroda_csomag;
/
```

Minimál fizetés

Készítsünk olyan függvényt, amely megadja, hogy az adott dolgozónak mennyi a foglalkozásának megfelelő minimum fizetés. (A FOGLAKOZAS táblában található fizetés ugyanott megadott százaléka szerinti fizetés.)

```
CREATE OR REPLACE PACKAGE iroda_csomag AS
  FUNCTION iroda_nev(p_iroda IN iroda.iroda_azon%TYPE)
              RETURN VARCHAR2;
  FUNCTION min_fiz(p_alk IN alkalmazott.alkalmazott_azon%TYPE)
              RETURN NUMBER;
END iroda_csomag;
/
CREATE OR REPLACE PACKAGE BODY iroda_csomag AS
…
  FUNCTION min_fiz(p_alk IN alkalmazott.alkalmazott_azon%TYPE)
              RETURN NUMBER AS
     ret_val NUMBER;
     v_fogl  NUMBER;
  BEGIN
     SELECT fogl_azon
       into v_fogl
       FROM alkalmazott
      WHERE alkalmazott_azon = p_alk;

     SELECT ROUND(fizetes*szazalek/100,-2)
       INTO ret_val
       FROM foglalkozas
      WHERE fogl_azon = v_fogl;

     RETURN ret_val;

  EXCEPTION WHEN NO_DATA_FOUND THEN
     RETURN NULL;
  END min_fiz;
END iroda_csomag;
/
```

Irodánkénti átlagfizetés számítás

Készítsünk az előző csomagba egy újabb függvényt, ami az átlagfizetést adja vissza! Nem megfelelő azonosítónál NULL legyen a visszatérési érték!

```
CREATE OR REPLACE PACKAGE iroda_csomag AS
  FUNCTION iroda_nev(p_iroda IN iroda.iroda_azon%TYPE)
               RETURN VARCHAR2;
  FUNCTION min_fiz(p_alk IN alkalmazott.alkalmazott_azon%TYPE)
               RETURN NUMBER;
  FUNCTION atl_fiz(p_IRODA IN iroda.iroda_azon%TYPE)
               RETURN NUMBER;
END iroda_csomag;
/
CREATE OR REPLACE PACKAGE BODY iroda_csomag AS
…
  FUNCTION ATL_FIZ(p_iroda IN iroda.iroda_azon%TYPE)
               RETURN NUMBER AS
     ret_val number;
  BEGIN
      SELECT AVG(fizetes) INTO ret_val
        FROM alkalmazott
       WHERE iroda_azon = p_iroda;
      RETURN ret_val;

  EXCEPTION WHEN NO_DATA_FOUND THEN
     RETURN NULL;
  END ATL_FIZ;
END iroda_csomag;
/
```

Fizetésemelés

Készítsünk egy függvényt, amellyel egy iroda minden dolgozójának adott százalék fizetésemelést adunk! A fizetés 100 Ft-ra legyen kerekítve! A függvény adja vissza, mennyi pénz kell a fizetésemeléshez! A függvény ne adjon ki COMMIT-ot, az legyen a hívó feladata!

```
CREATE OR REPLACE PACKAGE iroda_csomag AS
  FUNCTION iroda_nev(p_iroda IN iroda.iroda_azon%TYPE)
               RETURN VARCHAR2;
  FUNCTION min_fiz(p_alk IN alkalmazott.alkalmazott_azon%TYPE)
               RETURN NUMBER;
  FUNCTION atl_fiz(p_IRODA IN iroda.iroda_azon%TYPE)
               RETURN NUMBER;
  FUNCTION emeles(p_iroda IN iroda.iroda_azon%TYPE,
                  p_szazalek IN number)
               RETURN NUMBER;
END iroda_csomag;
/
```

```
CREATE OR REPLACE PACKAGE BODY iroda_csomag AS
…
  FUNCTION emeles(p_iroda IN iroda.iroda_azon%TYPE,
                  p_szazalek IN NUMBER)
                RETURN NUMBER AS
     v_elozo NUMBER;
     v_ujfiz NUMBER;
  BEGIN
     SELECT SUM(fizetes)
       INTO v_elozo
       FROM alkalmazott
      WHERE iroda_azon = p_iroda;

     UPDATE alkalmazott
        SET fizetes = ROUND(fizetes*(100+p_szazalek)/100,-2);
     UPDATE alkalmazott
        SET fizetes = min_fiz(alkalmazott_azon)
      where fizetes < min_fiz(alkalmazott_azon);

     SELECT SUM(fizetes)
       INTO v_ujfiz
       FROM alkalmazott
      WHERE iroda_azon = p_iroda;

     RETURN v_ujfiz-v_elozo;

  END emeles;
END iroda_csomag;
/
```

Tesztelés:

```
VARIABLE x NUMBER
EXECUTE :x:=iroda_csomag.emeles(20,11);
PRINT x
ROLLBACK;
```

Hibaüzenet:

ORA-04091: TANF20.ALKALMAZOTT tábla változtatás alatt áll, trigger/funkció számára nem látható
ORA-06512: a(z) "TANF20.IRODA_CSOMAG", helyen a(z) 25. sornál
ORA-06512: a(z) "TANF20.IRODA_CSOMAG", helyen a(z) 68. sornál
ORA-06512: a(z) helyen a(z) 2. sornál

A hibát az okozza, hogy a where feltételben is használjuk a függvényt, a set-ben is, és a függvény azt az értéket olvassa, amit éppen módosítunk. A hiba elkerülése úgy lehetséges, ha kurzort hozunk létre, előre kiolvassuk a kurzorral és a függvénnyel a szükséges adatokat, és az adatok módosítását már ezekkel a kézben lévő értékekkel hajtjuk végre,
Ez található a következő megoldásban:

```
CREATE OR REPLACE PACKAGE iroda_csomag AS
  FUNCTION iroda_nev(p_iroda IN iroda.iroda_azon%TYPE)
              RETURN VARCHAR2;
  FUNCTION min_fiz(p_alk IN alkalmazott.alkalmazott_azon%TYPE)
              RETURN NUMBER;
  FUNCTION atl_fiz(p_IRODA IN iroda.iroda_azon%TYPE)
              RETURN NUMBER;
  FUNCTION emeles(p_iroda IN iroda.iroda_azon%TYPE,
                  p_szazalek IN number)
              RETURN NUMBER;
  FUNCTION emeles2(p_iroda IN iroda.iroda_azon%TYPE,
                   p_szazalek IN number)
              RETURN NUMBER;
END iroda_csomag;
/

CREATE OR REPLACE PACKAGE BODY iroda_csomag AS
…
  FUNCTION emeles2(p_iroda IN iroda.iroda_azon%TYPE,
                   p_szazalek IN NUMBER)
              RETURN NUMBER AS
     v_elozo   NUMBER;
     v_ujfiz   NUMBER;
     v_minfiz  NUMBER;
     v_alk     alkalmazott.alkalmazott_azon%TYPE;
     v_fizetes alkalmazott.fizetes%TYPE;
     CURSOR v_cr IS
               SELECT alkalmazott_azon,fizetes
                 FROM alkalmazott
                 FOR UPDATE;
  BEGIN
     SELECT SUM(fizetes)
       INTO v_elozo
       FROM alkalmazott
      WHERE iroda_azon = p_iroda;

     UPDATE alkalmazott
        SET fizetes = ROUND(fizetes*(100+p_szazalek)/100,-2);

     OPEN v_cr;
     LOOP
     FETCH v_cr INTO v_alk, v_fizetes;
        EXIT WHEN v_cr%NOTFOUND;
           v_minfiz := min_fiz(v_alk);
        IF v_fizetes < v_minfiz  THEN
           UPDATE alkalmazott
              SET fizetes = v_minfiz
            WHERE CURRENT OF v_cr;
        END IF;
     END LOOP;
```

```
      SELECT SUM(fizetes)
        INTO v_ujfiz
        FROM alkalmazott
       WHERE iroda_azon = p_iroda;

      RETURN v_ujfiz-v_elozo;

  END emeles2;
END iroda_csomag;
/
```

Tesztelés:

```
      VARIABLE x NUMBER
      EXECUTE :x:=iroda_csomag.emeles2(20,11);
      PRINT x
      ROLLBACK;
```

```
anonymous block completed
X
------
119600

rollback complete.
```

I. 5. Kurzor használat

Készítsünk olyan függvényt, amely úgy végzi el a fizetésemelést, hogy akinek kevesebb az emelés utáni fizetése, mint a foglalkozásának megfelelő fizetés százaléka (mindkettő a FOGLALKOZAS tábla adata), az kapja meg ezt a minimumot!

```
CREATE OR REPLACE PACKAGE iroda_csomag AS
  FUNCTION iroda_nev(p_iroda IN iroda.iroda_azon%TYPE)
              RETURN VARCHAR2;
  FUNCTION min_fiz(p_alk IN alkalmazott.alkalmazott_azon%TYPE)
              RETURN NUMBER;
  FUNCTION atl_fiz(p_IRODA IN iroda.iroda_azon%TYPE)
              RETURN NUMBER;
  FUNCTION emeles(p_iroda IN iroda.iroda_azon%TYPE,
                  p_szazalek IN number)
              RETURN NUMBER;
  FUNCTION emeles2(p_iroda IN iroda.iroda_azon%TYPE,
                   p_szazalek IN number)
              RETURN NUMBER;
  FUNCTION emeles_min(p_iroda IN iroda.iroda_azon%TYPE,
                      p_szazalek IN number)
              RETURN NUMBER;
END iroda_csomag;
/
CREATE OR REPLACE PACKAGE BODY iroda_csomag AS
…
  FUNCTION emeles_min(p_iroda IN iroda.iroda_azon%TYPE,
                      p_szazalek IN NUMBER)
              RETURN NUMBER AS
     v_elozo NUMBER;
     v_ujfiz NUMBER;
     v_fizetes alkalmazott.fizetes%TYPE;
     v_alk     alkalmazott.alkalmazott_azon%TYPE;
     v_min_fiz NUMBER;
     v_fiz     NUMBER;
     CURSOR v_crsr IS
                  SELECT fizetes, alkalmazott_azon
                    FROM alkalmazott
                     FOR UPDATE;
  BEGIN
     SELECT SUM(fizetes)
       INTO v_elozo
       FROM alkalmazott
      WHERE iroda_azon = p_iroda;

     OPEN v_crsr;
     LOOP
        FETCH v_crsr INTO v_fizetes, v_alk;
        EXIT WHEN v_crsr%NOTFOUND;

        v_min_fiz :=  min_fiz(v_alk);
```

```
         v_fiz := ROUND(v_fizetes*(100+p_szazalek)/100,-2);

         IF v_fiz < v_min_fiz THEN
            v_fiz := v_min_fiz;
         END IF;

         UPDATE alkalmazott
            SET fizetes = v_fiz
            WHERE CURRENT OF v_crsr;
      END LOOP;
      CLOSE v_crsr;

      SELECT SUM(fizetes)
        INTO v_ujfiz
        FROM alkalmazott
       WHERE iroda_azon = p_iroda;

      RETURN v_ujfiz-v_elozo;

  END emeles_min;
END iroda_csomag;
/
```

Tesztelés:

```
      SELECT nev,fizetes,iroda_azon,
             iroda_csomag.min_fiz(alkalmazott_azon) min_fiz
        FROM alkalmazott
       WHERE fizetes < iroda_csomag.min_fiz(alkalmazott_azon);

      VARIABLE x NUMBER;
      BEGIN
      :x:=iroda_csomag.emeles_min(30,1);
      END;
      /
      PRINT x

      SELECT nev,fizetes,iroda_azon,
             iroda_csomag.min_fiz(alkalmazott_azon) min_fiz
        FROM alkalmazott
       WHERE fizetes < iroda_csomag.min_fiz(alkalmazott_azon);
```

```
NEV                    FIZETES IRODA_AZON    MIN_FIZ
-------------------- ---------- ---------- ----------
CZINEGE                 245000         10     252000

PL/SQL procedure successfully completed
x
---------
12100

no rows selected
```

Készítsünk olyan függvényt, amely egy alkalmazott azonosítója alapján visszaadja a nevét, és „ / ” jelekkel elválasztva a főnökeit hierarchia sorrendben felfelé!

```
CREATE OR REPLACE PACKAGE iroda_csomag AS
  FUNCTION iroda_nev(p_iroda IN iroda.iroda_azon%TYPE)
               RETURN VARCHAR2;
  FUNCTION min_fiz(p_alk IN alkalmazott.alkalmazott_azon%TYPE)
               RETURN NUMBER;
  FUNCTION atl_fiz(p_IRODA IN iroda.iroda_azon%TYPE)
               RETURN NUMBER;
  FUNCTION emeles(p_iroda IN iroda.iroda_azon%TYPE,
                  p_szazalek IN number)
               RETURN NUMBER;
  FUNCTION emeles2(p_iroda IN iroda.iroda_azon%TYPE,
                   p_szazalek IN number)
               RETURN NUMBER;
  FUNCTION emeles_min(p_iroda IN iroda.iroda_azon%TYPE,
                      p_szazalek IN number)
               RETURN NUMBER;
  FUNCTION fonokok(p_alk IN alkalmazott.alkalmazott_azon%TYPE)
               RETURN VARCHAR2;
END iroda_csomag;
/

CREATE OR REPLACE PACKAGE BODY iroda_csomag AS
…
  FUNCTION FONOKOK(P_ALK IN ALKALMAZOTT.ALKALMAZOTT_AZON%TYPE)
               RETURN VARCHAR2 AS
    RET_VAL VARCHAR2(2000) := '';
  BEGIN
    FOR CR IN ( SELECT NEV
                  FROM ALKALMAZOTT
            START WITH ALKALMAZOTT_AZON = P_ALK
      CONNECT BY PRIOR FONOK=ALKALMAZOTT_AZON )
   LOOP
      ret_val := ret_val || cr.nev || ' / ';
   END LOOP;
```

```
   IF LENGTH(RET_VAL) > 3 THEN
      RET_VAL := SUBSTR(RET_VAL,1,LENGTH(RET_VAL)-3);
   END IF;
   RETURN RET_VAL;
  END FONOKOK;
END iroda_csomag;
/
```

Használata:

```
     VARIABLE X VARCHAR2
     EXECUTE :x:=IRODA_CSOMAG.fonokok(7876);
     PRINT X

anonymous block completed
X
--------------------------------
ADAMIS / SÓBRY / KOVÁCS / KÁRMÁN
```

I. 6. Többalakú függvények, rekurzív függvények

Készítsünk 3 függvényt ugyanolyan néven, különböző paraméterezéssel! A függvény állapítsa meg, hogy ki a legjobb igazgató. Lehet az, akinek a közvetlen beosztottai a legtöbbet keresik. Paraméter megadás nélkül ez a függvény működjön. Ha kap egy paramétert, ami „AVG" akkor azt adja vissza, akinek a közvetlen beosztottainak a legnagyobb a kereset átlaga. Ha „COUNT" értéket kap, akkor az a legjobb, akinek a legtöbb a közvetlen beosztottja. Egyéb érték esetén működjön úgy, mint az első függvény.
Egy újabb függvény is készüljön, amelynek a paramétere 0 vagy 1. Ha 0, akkor az első függvény leírása szerint működik, ha 1, akkor nemcsak a közvetlen beosztottakat veszi, hanem az összeset. Ehhez készítsünk egy rekurzív függvényt is, ami összegyűjti az adott igazgató alatti beosztottak keresetét!

```
CREATE OR REPLACE PACKAGE iroda_csomag AS
  FUNCTION iroda_nev(p_iroda IN iroda.iroda_azon%TYPE)
              RETURN VARCHAR2;
  FUNCTION min_fiz(p_alk IN alkalmazott.alkalmazott_azon%TYPE)
              RETURN NUMBER;
  FUNCTION atl_fiz(p_IRODA IN iroda.iroda_azon%TYPE)
              RETURN NUMBER;
  FUNCTION emeles(p_iroda IN iroda.iroda_azon%TYPE,
                  p_szazalek IN number)
              RETURN NUMBER;
  FUNCTION emeles_min(p_iroda IN iroda.iroda_azon%TYPE,
                      p_szazalek IN number)
              RETURN NUMBER;
  FUNCTION emeles2(p_iroda IN iroda.iroda_azon%TYPE,
                   p_szazalek IN number)
              RETURN NUMBER;
  FUNCTION fonokok(p_alk IN alkalmazott.alkalmazott_azon%TYPE)
              RETURN VARCHAR2;
  FUNCTION legjobb_fonok                          RETURN VARCHAR2;
  FUNCTION legjobb_fonok(p_tipus IN VARCHAR2) RETURN VARCHAR2;
  FUNCTION legjobb_fonok(p_tipus IN NUMBER)   RETURN VARCHAR2;
  FUNCTION rekurziv_fv(p_azon IN NUMBER)      RETURN NUMBER;
END iroda_csomag;
/
CREATE OR REPLACE PACKAGE BODY iroda_csomag AS
…
  FUNCTION legjobb_fonok return VARCHAR2 IS
    retval alkalmazott.nev%TYPE;
  BEGIN
     SELECT nev INTO retval FROM (
        SELECT ff.nev,SUM(a.fizetes+NVL(a.jutalek,0)) fiz
          FROM alkalmazott a,alkalmazott ff,foglalkozas f
         WHERE ff.alkalmazott_azon(+) = a.fonok
           AND ff.fogl_azon = f.fogl_azon
           AND f.foglalkozas = 'Igazgató'
         GROUP BY ff.nev
         ORDER BY fiz DESC)
         WHERE ROWNUM=1;
```

```
   RETURN retval;
END legjobb_fonok;

FUNCTION legjobb_fonok(p_tipus in varchar2)
                                    RETURN VARCHAR2 IS
  retval alkalmazott.nev%TYPE;
BEGIN
   IF p_tipus = 'AVG' THEN
      SELECT nev INTO retval FROM (
         SELECT ff.nev,AVG(a.fizetes+NVL(a.jutalek,0)) fiz
           FROM alkalmazott a,alkalmazott ff,foglalkozas f
          WHERE ff.alkalmazott_azon(+) = a.fonok
            AND ff.fogl_azon = f.fogl_azon
            AND f.foglalkozas = 'Igazgató'
          GROUP BY ff.nev
          ORDER BY fiz DESC)
          WHERE ROWNUM=1;
    ELSE
       IF p_tipus = 'COUNT' THEN
         SELECT nev INTO retval FROM (
            SELECT ff.nev,COUNT(*) db
              FROM alkalmazott a,alkalmazott ff,
                   foglalkozas f
             WHERE ff.alkalmazott_azon(+) = a.fonok
               AND ff.fogl_azon = f.fogl_azon
               AND f.foglalkozas = 'Igazgató'
             GROUP BY ff.nev
             ORDER BY DB DESC)
             WHERE ROWNUM=1;
       ELSE
          RETURN legjobb_fonok;
       END IF;
    END IF;
    RETURN retval;

END legjobb_fonok;

FUNCTION legjobb_fonok(p_tipus IN NUMBER) RETURN VARCHAR2 IS
  retval alkalmazott.nev%TYPE;
BEGIN
   IF p_tipus = 0 THEN
      RETURN legjobb_fonok;
   ELSE
      IF p_tipus = 1 THEN
         SELECT nev INTO retval FROM (
            SELECT ff.nev,rekurziv_fv(ff.fonok) mfiz
              FROM alkalmazott ff,foglalkozas f
             WHERE f.foglalkozas = 'Igazgató'
               AND ff.fogl_azon = f.fogl_azon
             ORDER BY mfiz desc)
             WHERE ROWNUM=1;
```

```
        ELSE
           RETURN 'Hibás paraméter érték !';
        END IF;
     END IF;
     RETURN retval;
  END legjobb_fonok;

  FUNCTION rekurziv_fv(p_azon IN NUMBER) RETURN NUMBER IS
     v_fizetes NUMBER;
     v_sumfiz  NUMBER;
     v_alk     NUMBER;
     CURSOR v_crsr(p_alk_azon IN NUMBER) IS
                  SELECT fizetes, alkalmazott_azon
                    FROM alkalmazott
                   WHERE fonok = p_alk_azon;
  BEGIN
      v_sumfiz := 0;
      OPEN v_crsr(p_azon);
      LOOP
         FETCH v_crsr INTO v_fizetes, v_alk;
         EXIT WHEN v_crsr%NOTFOUND;
              v_sumfiz := v_sumfiz + v_fizetes +
                          rekurziv_fv(v_alk);
      END LOOP;
      CLOSE v_crsr;
      RETURN v_sumfiz;
  END rekurziv_fv;
END iroda_csomag;
/
```

Hívása:

```
     VARIABLE x VARCHAR2
     EXECUTE :x:=iroda_csomag.legjobb_fonok(1);
     PRINT x
X
------
KOVÁCS
```

I. 7. Trigger létrehozása

Hozzunk létre az ALKALMAZOTT táblához egy olyan triggert, ami arról gondoskodik, hogy ha egy újonnan felvitt alkalmazottnak nem töltünk ki FIZETES értéket, akkor adjon neki a foglalkozás alapján automatikusan egy kezdőknek járó fizetést 1000 Ft-ra kerekítve. (A foglalkozásra jellemző fizetés adott százalékát. Mindkét adat a FOGLALKOZAS táblában található.) Helyezzük el a hiba naplózását is a triggerben!

```
CREATE OR REPLACE TRIGGER trg_alkalmazott_bir
   BEFORE INSERT ON alkalmazott
   FOR EACH ROW
BEGIN
 -- HA NINCS MEGADVA FIZETÉS, AZ ALSÓ HATÁRT KAPJA 1000 Ft-ban
  IF :NEW.fizetes IS NULL THEN
     SELECT ROUND(f.fizetes * f.szazalek / 100, -3)
       INTO :NEW.fizetes
       FROM foglalkozas f
      WHERE f.fogl_azon = :NEW.fogl_azon;
  END IF;
  EXCEPTION
     WHEN OTHERS THEN
        naplo.naplo_ir('TRG_ALKALMAZOTT_BIR');
END;
/
```

Teszteljük le a triggert!

```
-- TRIGGER TESZT
INSERT into alkalmazott
 (alkalmazott_azon,nev,fogl_azon,fonok,belepes_dat,iroda_azon)
VALUES
 (8000,'BARABÁS',20,7839,to_date('81-05-01','RR-MM-DD'),30);
COMMIT;
```

A FIZETES mező kitöltődik a 30-as foglalkozás szerint számolt értékkel. (252 000)

Teszteljük a hibakezelést is!

```
INSERT INTO alkalmazott (alkalmazott_azon) VALUES (10);
```

```
DATUM      FELH PROGRAM                    HIBAUZENET
2012.06.08 TANF TRG_ALKALMAZOTT_BIR        ORA-01403: no data found
```

Ha elemezzük az INSERT utasítást és a triggert, akkor azt látjuk, hogy azért „no data found” a hibaüzenet, mert nem adtunk meg foglalkozás kódot, ezért nem talált fizetés és százalék adatot sem.

Másik próba:

```
INSERT INTO alkalmazott (alkalmazott_azon,fogl_azon)
                 VALUES (10,30);
```

A képernyőre hibaüzenetet kapunk, de a hibatáblában nem jelenik meg újabb sor.

SQL Error: ORA-01400: cannot insert NULL into ("TANF"."ALKALMAZOTT"."NEV")

Ennek az az oka, hogy nem a trigger végrehajtásakor keletkezett a hiba, hanem már előtte, ezért nem hívódott meg a naplózó eljárás.

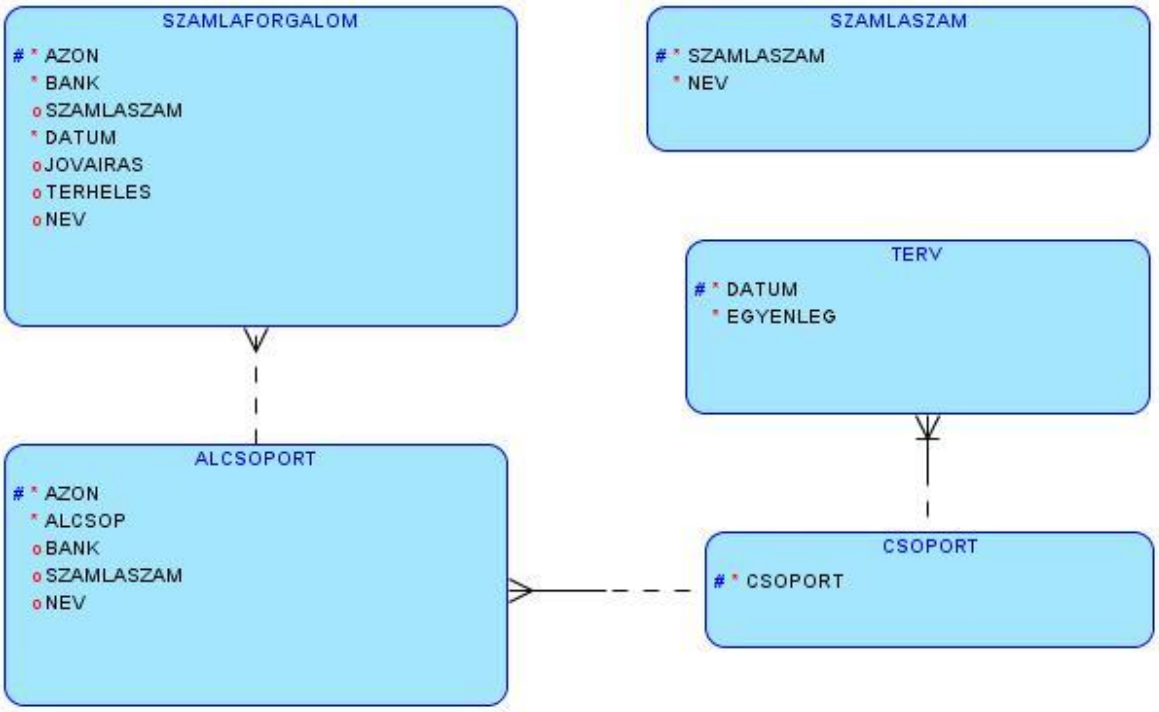
II. Számlák kezelése, összesítések készítése

II. 1. Adatmodell

Készítsünk adatmodellt különböző számlák figyelésére. Több banknál van folyószámlánk. Ezek adatait gyűjtjük. A pénzforgalmat kategorizáljuk, éves tervet készítünk a havi kiadásokra, bevételekre kategóriánként.

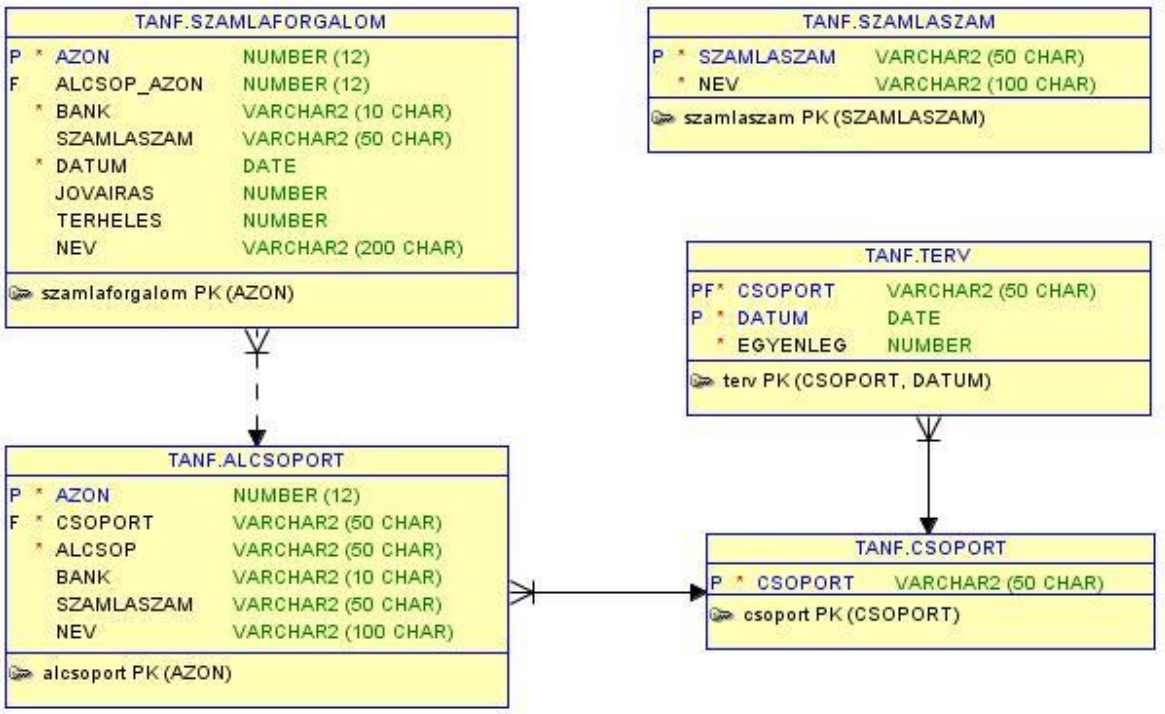
Az alábbi adatokat tartjuk nyilván.
Bank neve, számlaszámok és nevük, ahova utalunk, vagy ahonnét pénz érkezik, dátum, jóváírás, terhelés összege, a pénzforgalom adott tételének neve (kísérő szövege). Kategória csoport, azon belüli bontás neve. Terv kategória csoportja, hónapja, tervezett összeg.

Az alábbi logikai modell alakul ki a fenti adatokból:



A SZAMLASZAM-ot nem kapcsoljuk direktben a SZAMLAFORGALOM-hoz, mivel nem mindig számlához kapcsolódik a pénzforgalom, lehet készpénzes be- vagy kifizetés is. Nem tartunk nyilván minden számlaszámot, csak a rendszeresen használtakat. Az eseti számlaszámokat nem kell felvinni a táblába.
Az ALCSOPORT a SZAMLAFORGALOMHOZ nem kötelezően kapcsolódik, mivel a felvitt adatokat utólag fogjuk kategóriákba sorolni. Az alcsoportnak viszont kötelezően csoporthoz kell tartoznia, és terv is a csoportok alapján készül, ezért ezek a kapcsolatok ebben az irányban kötelezőek. Az alcsoportnál adunk meg olyan adatokat, ami alapján tudja kategorizálni a számlaforgalmat. A BANK, SZAMLASZAM és NEV mező tartalma alapján kerülnek egy-egy kategóriába a rekordok.

Az adattípusok meghatározásával az alábbi fizikai modellt alakítjuk ki:



A táblák létrehozó utasításai és kezdeti feltöltésük a III. 2 mellékletben találhatók.

II. 2. Csomag készítés

Készítsünk egy BANK nevű csomagot, amelyben készítünk egy függvényt, amely kategorizálja a SZAMLAFORGALOM rekordjait az ALCSOPORT adatai alapján.
Input paraméterrel szabályozzuk, hogy csak a kategorizálatlanokat sorolja be, vagy az összeset újra kell kategorizálni.
Mivel az egyes mezők kitöltöttségétől függ, hogy kell-e az adott mezőre szűrni, ezért készítsünk dinamikus select-et!

```
CREATE OR REPLACE PACKAGE bank AS

  -- Author  : Ender János
  -- Created : 2012.02.27.
  -- Purpose : Bankszámlák karbantartása

  -- Public function and procedure declarations
  PROCEDURE csoport_update(p_teljes IN VARCHAR2 DEFAULT NULL);

END bank;
/
```

```
CREATE OR REPLACE PACKAGE BODY bank AS

  /*  csoport_update
      Feladata: Kategóriákba sorolja a számla forgalmi rekordokat

      Input paraméter:  p_teljes  Ha NULL, akkor csak az eddig nem
                        kategorizáltakat kategorizálja, egyébként az
                        összeset
  */
  PROCEDURE csoport_update(p_teljes IN VARCHAR2 DEFAULT NULL) AS
      v_fix VARCHAR2(100);
      v_felt VARCHAR2(4000);
      v_comm VARCHAR2(4000);
  BEGIN
      v_fix := 'alcsop_azon IS NULL';
      IF p_teljes IS NOT NULL THEN
         UPDATE szamlaforgalom
            SET alcsop_azon = NULL;
            COMMIT;
      END IF;

      FOR cr IN (SELECT * FROM alcsoport ORDER BY azon, alcsop) LOOP
         v_felt := '';

         IF cr.bank IS NOT NULL THEN
            v_felt := v_felt || ' AND bank=''' || cr.bank || '''';
         END IF;

         IF cr.szamlaszam IS NOT NULL THEN
            v_felt := v_felt || ' AND szamlaszam LIKE ''' || cr.szamlaszam
                             || '%''';
         END IF;

         IF cr.nev IS NOT NULL THEN
            v_felt := v_felt || ' AND nev LIKE ''%' || cr.nev || '%''';
         END IF;

         v_comm := 'UPDATE szamlaforgalom SET alcsop_azon = ' || cr.azon ||
                   ' WHERE ' || v_fix || v_felt;

         EXECUTE IMMEDIATE v_comm;

      END LOOP;
      COMMIT;

   EXCEPTION
      WHEN OTHERS THEN
         naplo.naplo_ir('BANK.csoport_update');
         RAISE_APPLICATION_ERROR(-20000, 'Hiba: ' || v_comm || chr(10) ||
                                 SQLERRM);
  END csoport_update;
END bank;
/
```

Futtassuk le a karbantartást!

```
BEGIN
   bank.csoport_update();
END;
/
```

Havonkénti átlag számolása kategóriánként

Számítsuk ki csoportonként a havi átlag egyenleget! Legyen egy függvény a csomagban, ami a csoport neve alapján visszaadja az átlagot!

```
CREATE OR REPLACE PACKAGE bank AS
   PROCEDURE CSOPORT_UPDATE(p_teljes IN VARCHAR2 DEFAULT NULL);
   FUNCTION CSOPORT_ATLAG(p_csoport IN VARCHAR2) RETURN NUMBER;
END bank;
/
CREATE OR REPLACE PACKAGE BODY bank AS
…
…
…
  /*  csoport_atlag
      Feladata:         Egy kategóriákba átlagát számolja ki

      Input paraméter:    p_csoport  a csoport
      Visszatérési érték: a kiszámolt átlag
  */
  FUNCTION csoport_atlag(p_csoport IN VARCHAR2)
    RETURN NUMBER IS
     ret_val NUMBER;
  BEGIN
     SELECT ROUND(AVG(egyenleg))  INTO ret_val
      FROM (SELECT TO_CHAR(datum,'yyyymm'),
                SUM(NVL(jovairas,0)-NVL(terheles,0)) egyenleg
              FROM szamlaforgalom sz, alcsoport cs
             WHERE cs.csoport = p_csoport
               AND sz.alcsop_azon = cs.azon
             GROUP BY TO_CHAR(datum,'yyyymm'));
     RETURN ret_val;
  END csoport_atlag;
END bank;
/
```

Egy select utasítással írjuk ki az összes csoport átlagát!

```
SELECT csoport, bank.csoport_atlag(csoport)
  FROM csoport
  ORDER BY csoport;
```

Eredmény:

```
CSOPORT                        BANK.CSOPORT_ATLAG(CSOPORT)
------------------------------ ---------------------------
Átutalás                                                 0
Bankköltség                                            -34
Biztosítás                                           -9600
Egyéb                                                 9003
 …
```

Összegző nézet készítése

Készítsünk olyan nézetet, amelyben szerepelnek a csoportonkénti havi összesítések, a teljes összesítés, a terv, és a tervtől való eltérés!

```
CREATE OR REPLACE VIEW osszesito_v AS
SELECT fsz.honap,fsz.csoport,fsz.jovairas, fsz.terheles,fsz.egyenleg,
       terv.egyenleg tervezett, fsz.egyenleg-terv.egyenleg elteres
FROM
(SELECT  '* Összesen' AS honap, '* Összesen' AS csoport,
         ROUND(SUM(jovairas)) jovairas,
         ROUND(SUM(terheles)) terheles,
         ROUND(SUM(NVL(jovairas, 0))) -
                     ROUND(SUM(NVL(terheles, 0))) egyenleg
     FROM szamlaforgalom
UNION ALL
SELECT   nvl(TO_CHAR(datum, 'yyyy.mm'),'* Összesen') AS honap,
         nvl(ac.csoport,'* Összesen') AS csoport,
         ROUND(SUM(jovairas)) jovairas,
         ROUND(SUM(terheles)) terheles,
         ROUND(SUM(NVL(jovairas, 0))) -
                     ROUND(SUM(NVL(terheles, 0))) egyenleg
     FROM szamlaforgalom b, alcsoport ac
     WHERE ac.azon = alcsop_azon
     GROUP BY GROUPING SETS ((TO_CHAR(datum, 'yyyy.mm'), ac.csoport),
                             ac.csoport, TO_CHAR(datum, 'yyyy.mm'))
  ) fsz,
(SELECT '* Összesen' AS honap, '* Összesen' AS csoport,
         ROUND(SUM(egyenleg)) AS egyenleg
     FROM terv b
UNION ALL
SELECT  NVL(TO_CHAR(datum, 'yyyy.mm'),'* Összesen') AS honap,
        NVL(csoport,'* Összesen') AS csoport,
        ROUND(SUM(egyenleg)) AS egyenleg
     FROM terv
     GROUP BY GROUPING SETS ((TO_CHAR(datum, 'yyyy.mm'), csoport),
                              TO_CHAR(datum, 'yyyy.mm'), csoport)
   ) terv
WHERE nvl(terv.honap,'* Összesen') = nvl(fsz.honap,'* Összesen')
  AND nvl(terv.csoport,'* Összesen') = nvl(fsz.csoport,'* Összesen')
ORDER BY fsz.honap, fsz.csoport;
```

Kurzort visszaadó eljárások

Egyéb csoport részletezése

Készítsünk olyan eljárást a BANK csomagba, amely megadott időszakra kihozza az „Egyéb” csoportba eső számla információkat (bank, számlaszám, név). Az eredményt egy kurzorban adja vissza egy OUT paraméterben!
A kurzor változót REF CURSOR–ként lehet létrehozni, de ehhez először egy típust kell létrehozni a csomag specifikációs részében, majd ez a típus használható a paraméter típusaként.

```
CREATE OR REPLACE PACKAGE bank AS
   TYPE my_cur IS REF CURSOR;
   PROCEDURE CSOPORT_UPDATE(p_teljes IN VARCHAR2 DEFAULT NULL);
   FUNCTION CSOPORT_ATLAG(p_csoport IN VARCHAR2) RETURN NUMBER;
   PROCEDURE EGYEB_LIST(p_cursor OUT my_cur,
                        p_idoszak IN VARCHAR2 DEFAULT '*');
END bank;
/
CREATE OR REPLACE PACKAGE BODY bank AS
…
…
  /*  egyeb_list
      Feladata:         kurzor viszadása az Egyéb kategóriába esőkről

      Input paraméter:  p_idoszak  Időszak 'ÉÉÉÉ.HH' formában vagy *
      Output paraméter: p_cursor
  */
  PROCEDURE egyeb_list(p_cursor OUT my_cur,
                       p_idoszak IN VARCHAR2 DEFAULT '*')
  IS
  BEGIN
    IF p_idoszak IS NULL OR p_idoszak = '*' THEN
      OPEN p_cursor FOR
       SELECT s.bank,s.szamlaszam,s.nev FROM szamlaforgalom s, alcsoport cs
        WHERE cs.csoport = 'Egyéb'
          AND s.alcsop_azon = cs.azon;
    ELSE
      OPEN p_cursor FOR
       SELECT s.bank,s.szamlaszam,s.nev FROM szamlaforgalom s, alcsoport cs
        WHERE cs.csoport = 'Egyéb'
          AND s.alcsop_azon = cs.azon
          AND to_char(s.datum,'yyyy.mm') = p_idoszak;
      END IF;
  END egyeb_list;
END bank;
/
```

Használata: (PL/SQL Developer nem támogatja a REFCURSOR használatát, sqlplus alatt működik)

```
VARIABLE v_return REFCURSOR
EXECUTE  BANK.egyeb_list(:v_return);
PRINT v_return
```

Összesítő lekérdezés

Készítsünk a BANK csomagba egy függvényt, amely az OSSZESITETT_V nézet alapján visszaad egy kurzort! A függvény két input paramétert kap, a hónapot és a csoportot, amire szűri a view-t. Amelyik érték helyett NULL-t kap, vagy nem kap értéket, ott az összesített adatokra szűr. '*'-ot megadva az adott paraméterben nem szűr arra a mezőre.

```
CREATE OR REPLACE PACKAGE bank AS

   TYPE my_cur IS REF CURSOR;

   -- Public function and procedure declarations
   PROCEDURE CSOPORT_UPDATE(p_teljes IN VARCHAR2 DEFAULT NULL);
   FUNCTION CSOPORT_ATLAG(p_csoport IN VARCHAR2) RETURN NUMBER;
   PROCEDURE EGYEB_LIST(p_cursor OUT my_cur,
                        p_idoszak IN VARCHAR2 DEFAULT '*');
   FUNCTION OSSZESITO_LEK(p_idoszak IN VARCHAR2 DEFAULT '* Összesen',
                          p_csoport IN VARCHAR2 DEFAULT '* Összesen')
            return my_cur;

END bank;
/
CREATE OR REPLACE PACKAGE BODY bank AS
…
…
…
  /*  osszesito_lek
      Feladata:         kurzor viszadása az OSSZESITO_V alapján

      Input paraméter:  p_idoszak  Időszak 'ÉÉÉÉ.HH' formában
                        p_csoport  Csoport neve
            Amelyik üres, annál összesített adat jelenik meg
            * -nál minden adat megjelenik
            Visszatérési érték a kurzor.
            A hívónak kell lezárni a kurzort.
  */
   FUNCTION OSSZESITO_LEK(p_idoszak IN VARCHAR2 DEFAULT '* Összesen',
                          p_csoport IN VARCHAR2 DEFAULT '* Összesen')
            RETURN my_cur IS
     c_lek my_cur;
  BEGIN
     IF p_idoszak = '*' THEN
        IF p_csoport = '*' THEN
           OPEN c_lek FOR SELECT * FROM osszesito_v;
        ELSE
           OPEN c_lek FOR SELECT * FROM osszesito_v
               WHERE UPPER(csoport) = UPPER(NVL(p_csoport,'* Összesen'));
        END IF;
     ELSE
        IF p_csoport = '*' THEN
           OPEN c_lek FOR SELECT * FROM osszesito_v
                           WHERE honap = NVL(p_idoszak,'* Összesen');
        ELSE
           OPEN c_lek FOR
              SELECT * FROM osszesito_v
               WHERE honap = NVL(p_idoszak,'* Összesen')
                 AND UPPER(csoport) = UPPER(NVL(p_csoport,'* Összesen'));
        END IF;
     END IF;
```

```
     RETURN c_lek;


   EXCEPTION
      WHEN OTHERS THEN
         naplo.naplo_ir('BANK.osszesito_lek');
         RAISE_APPLICATION_ERROR(-20001, 'Hiba: p_idoszak=' || p_idoszak ||
                                            'p_csoport=' || p_csoport ||
                                             CHR(10) || SQLERRM);
         RETURN NULL;
  END osszesito_lek;
END bank;
/
```

A függvény hívása

```
VARIABLE v_return REFCURSOR

DECLARE
  p_idoszak VARCHAR2(200);
  p_csoport VARCHAR2(200);
  V_RETURN TANF.bank.my_cur;
BEGIN
  p_idoszak := '2012.03';
  p_csoport := '*';

  v_Return := bank.osszesito_lek(
    p_idoszak => p_idoszak,
    p_csoport => p_csoport
  );
  :v_return := v_return;
END;
/
PRINT v_return
```

Ha csak összesítést akarunk látni, akkor ez a legegyszerűbb forma:

```
VARIABLE v_return REFCURSOR
EXECUTE  :v_return := bank.osszesito_lek;
PRINT    v_return
```

SQL scripttel történő lekérdezés

Készítsünk egy SQL szkriptet, amelyet lefuttatva kilistázza a csoportokat, bekéri, melyik csoportra vagyunk kíváncsiak, majd kiírja havi növekvő sorrendben az adott kategória forgalmát, eltérését a tervezett egyenlegtől!

```
SET PAGESIZE 300
SET LINESIZE 200
CLEAR COLUMN
SELECT csoport FROM csoport;
COLUMN honap FORMAT A10
COLUMN csoport FORMAT A13
COLUMN jovairas FORMAT 99,999,999
COLUMN terheles FORMAT 99,999,999
COLUMN egyenleg FORMAT 99,999,999
COLUMN terv FORMAT 99,999,999
COLUMN elteres FORMAT 99,999,999
SELECT honap, SUBSTR(csoport,1,13) csoport, jovairas,
       terheles, egyenleg, elteres
  FROM osszesito_v
 WHERE LOWER(csoport) LIKE LOWER('%&Csoport%');
```

Eredmény (rezsi-t adtam meg a Csoport bekérésekor):

```
CSOPORT
-------------------------------------------------
Bankköltség
Biztosítás
Egyéb
Fizetés
Gyerekek
Hitel
Iskola
Készpénz felvétel
Rezsi
Vásárlás
Átutalás

 11 rows selected
```

-- Ez csak sqlplus alatt jelenik meg:

```
old:SELECT honap,SUBSTR(csoport,1,13)
csoport,jovairas,terheles,egyenleg,elteres
  FROM osszesito_v
 WHERE LOWER(csoport) LIKE LOWER('%&Csoport%')
new:SELECT honap,SUBSTR(csoport,1,13)
csoport,jovairas,terheles,egyenleg,elteres
  FROM osszesito_v
 where LOWER(csoport) LIKE LOWER('%rezsi%')
--
```

```
HONAP      CSOPORT          JOVAIRAS    TERHELES    EGYENLEG     ELTERES
---------- ------------- ----------- ----------- ----------- -----------
* Összesen Rezsi                       154,827    -154,827     265,173
2012.01    Rezsi                        21,306     -21,306      13,694
2012.02    Rezsi                        77,583     -77,583     -42,583
2012.03    Rezsi                        32,345     -32,345       2,655
2012.04    Rezsi                        23,593     -23,593      11,407
```

III. MELLÉKLETEK

III. 1. Dolgozók nyilvántartása

Adattáblák létrehozása

```
CREATE TABLE MUNKA
    (
     M_AZON  NUMBER(2)           NOT NULL,
     M_FOGL  VARCHAR2 (20 CHAR) NOT NULL,
     M_FIZ   NUMBER (10)         NOT NULL,
     M_SZAZ  NUMBER (3)          DEFAULT 80
    ) ;

CREATE TABLE IRODA
    (
     IRODA_AZON NUMBER(2)          NOT NULL ,
     NEV        VARCHAR2(15 CHAR) NOT NULL ,
     HELY       VARCHAR2(30 CHAR) NOT NULL
    ) ;

ALTER TABLE IRODA
    ADD CONSTRAINT
        PK_IRODA PRIMARY KEY ( IRODA_AZON );

CREATE TABLE FOGLALKOZAS
    (
      FOGL_AZON  NUMBER(2)           NOT NULL,
     FOGLALKOZAS VARCHAR2 (20 CHAR) NOT NULL,
     FIZETES     NUMBER (10)         NOT NULL,
     SZAZALEK    NUMBER (3)          DEFAULT 80
    ) ;

ALTER TABLE FOGLALKOZAS
    ADD CONSTRAINT PK_FOGLALKOZAS
        PRIMARY KEY ( FOGL_AZON );
```

```
CREATE TABLE ALKALMAZOTT
    (
     ALKALMAZOTT_AZON NUMBER(4)          NOT NULL,
     NEV              VARCHAR2(20 CHAR)  NOT NULL,
     FOGL_AZON        NUMBER(2)          NOT NULL,
     FONOK            NUMBER(4),
     BELEPES_DAT      DATE,
     FIZETES          NUMBER(10),
     JUTALEK          NUMBER(10),
     IRODA_AZON       NUMBER(2)
    ) ;

ALTER TABLE ALKALMAZOTT
    ADD CONSTRAINT PK_ALKALMAZOTT
        PRIMARY KEY ( ALKALMAZOTT_AZON ) ;

ALTER TABLE ALKALMAZOTT
    ADD CONSTRAINT FK_IRODA_AZON
        FOREIGN KEY  (  IRODA_AZON  )
        REFERENCES IRODA  ( IRODA_AZON );

ALTER TABLE ALKALMAZOTT
    ADD CONSTRAINT FK_FOGL_AZON
        FOREIGN KEY  (  FOGL_AZON  )
        REFERENCES FOGLALKOZAS  ( FOGL_AZON );
```

Feltöltés adatokkal

```
Insert into IRODA (IRODA_AZON,NEV,HELY)
 values (10,'Központ','Budapest');
Insert into IRODA (IRODA_AZON,NEV,HELY)
 values (20,'Kutatás','Veszprém');
Insert into IRODA (IRODA_AZON,NEV,HELY)
 values (30,'Raktár','Eger');
Insert into IRODA (IRODA_AZON,NEV,HELY)
 values (40,'Gyártás','Szolnok');

Insert into FOGLALKOZAS (FOGL_AZON,FOGLALKOZAS,FIZETES)
 values (10,'Elnök',500000);
Insert into FOGLALKOZAS
(FOGL_AZON,FOGLALKOZAS,FIZETES,SZAZALEK)
 values (20,'Igazgató',280000,90);
Insert into FOGLALKOZAS
(FOGL_AZON,FOGLALKOZAS,FIZETES,SZAZALEK)
 values (30,'Elemző',300000,100);
Insert into FOGLALKOZAS
(FOGL_AZON,FOGLALKOZAS,FIZETES,SZAZALEK)
 values (40,'Eladó',130000,70);
Insert into FOGLALKOZAS
(FOGL_AZON,FOGLALKOZAS,FIZETES,SZAZALEK)
```

```
 values (50,'Ügyintéző',80000,90);

Insert into ALKALMAZOTT
(ALKALMAZOTT_AZON,NEV,FOGL_AZON,FONOK,BELEPES_DAT,FIZETES,
JUTALEK,IRODA_AZON)
 values (7499,'ALBERT',40,7698,
         to_date('81-02-20','RR-MM-DD'),160000,30000,30);
Insert into ALKALMAZOTT
(ALKALMAZOTT_AZON,NEV,FOGL_AZON,FONOK,BELEPES_DAT,FIZETES,
JUTALEK,IRODA_AZON)
 values (7369,'SMIDT',50,7902,
         to_date('80-12-17','RR-MM-DD'),80000,null,20);
Insert into ALKALMAZOTT
(ALKALMAZOTT_AZON,NEV,FOGL_AZON,FONOK,BELEPES_DAT,FIZETES,
JUTALEK,IRODA_AZON)
 values (7521,'KISS',40,7698,
         to_date('81-02-22','RR-MM-DD'),125000,50000,30);
Insert into ALKALMAZOTT
(ALKALMAZOTT_AZON,NEV,FOGL_AZON,FONOK,BELEPES_DAT,FIZETES,
JUTALEK,IRODA_AZON)
 values (7566,'KOVÁCS',20,7839,
         to_date('81-04-02','RR-MM-DD'),297500,null,20);
Insert into ALKALMAZOTT
(ALKALMAZOTT_AZON,NEV,FOGL_AZON,FONOK,BELEPES_DAT,FIZETES,
JUTALEK,IRODA_AZON)
 values (7654,'MAJOR',40,7698,
         to_date('81-09-28','RR-MM-DD'),125000,140000,30);
Insert into ALKALMAZOTT
(ALKALMAZOTT_AZON,NEV,FOGL_AZON,FONOK,BELEPES_DAT,FIZETES,
JUTALEK,IRODA_AZON)
 values (7698,'BUZÁS',20,7839,
         to_date('81-05-01','RR-MM-DD'),285000,null,30);
Insert into ALKALMAZOTT
(ALKALMAZOTT_AZON,NEV,FOGL_AZON,FONOK,BELEPES_DAT,FIZETES,
JUTALEK,IRODA_AZON)
 values (7782,'CZINEGE',20,7839,
         to_date('81-01-09','RR-MM-DD'),245000,null,10);
Insert into ALKALMAZOTT
(ALKALMAZOTT_AZON,NEV,FOGL_AZON,FONOK,BELEPES_DAT,FIZETES,
JUTALEK,IRODA_AZON)
 values (7788,'SÓBRY',30,7566,
         to_date('87-04-19','RR-MM-DD'),300000,null,20);
Insert into ALKALMAZOTT
(ALKALMAZOTT_AZON,NEV,FOGL_AZON,FONOK,BELEPES_DAT,FIZETES,
JUTALEK,IRODA_AZON)
 values (7839,'KÁRMÁN',10,null,
         to_date('81-11-17','RR-MM-DD'),500000,null,10);
Insert into ALKALMAZOTT
(ALKALMAZOTT_AZON,NEV,FOGL_AZON,FONOK,BELEPES_DAT,FIZETES,
JUTALEK,IRODA_AZON)
 values (7844,'TÓTH',40,7698,
```

```
        to_date('81-09-08','RR-MM-DD'),150000,0,30);
Insert into ALKALMAZOTT
(ALKALMAZOTT_AZON,NEV,FOGL_AZON,FONOK,BELEPES_DAT,FIZETES,
JUTALEK,IRODA_AZON)
 values (7876,'ADAMIS',50,7788,
        to_date('87-05-23','RR-MM-DD'),110000,null,20);
Insert into ALKALMAZOTT
(ALKALMAZOTT_AZON,NEV,FOGL_AZON,FONOK,BELEPES_DAT,FIZETES,
JUTALEK,IRODA_AZON)
 values (7900,'JÓZSEF',50,7698,
        to_date('81-12-03','RR-MM-DD'),95000,null,30);
Insert into ALKALMAZOTT
(ALKALMAZOTT_AZON,NEV,FOGL_AZON,FONOK,BELEPES_DAT,FIZETES,
JUTALEK,IRODA_AZON)
 values (7902,'FELEKY',30,7566,
        to_date('81-12-03','RR-MM-DD'),300000,null,20);
Insert into ALKALMAZOTT
(ALKALMAZOTT_AZON,NEV,FOGL_AZON,FONOK,BELEPES_DAT,FIZETES,
JUTALEK,IRODA_AZON)
 values (7934,'MÜLLER',50,7782,
        to_date('82-01-23','RR-MM-DD'),130000,null,10);
commit;


Insert into MUNKA (M_AZON,M_FOGL,M_FIZ)
          values (15,'Alelnök',450000);
Insert into MUNKA (M_AZON,M_FOGL,M_FIZ,M_SZAZ)
          values (20,'Igazgató',400000,90);
Insert into MUNKA (M_AZON,M_FOGL,M_FIZ,M_SZAZ)
          values (25,'Főelemző',330000,100);
Insert into MUNKA (M_AZON,M_FOGL,M_FIZ,M_SZAZ)
          values (40,'Eladó',130000,80);
Insert into MUNKA (M_AZON,M_FOGL,M_FIZ,M_SZAZ)
          values (60,'Beszerző',110000,70);
commit;
```

III. 2. Számlák kezelése, összesítések készítése

Adattáblák létrehozása

```
CREATE TABLE alcsoport
    (
     azon       NUMBER(12)          NOT NULL ,
     csoport    VARCHAR2 (50 CHAR)  NOT NULL ,
     alcsop     VARCHAR2 (50 CHAR)  NOT NULL ,
     bank       VARCHAR2 (10 CHAR) ,
     szamlaszam VARCHAR2 (50 CHAR) ,
     nev        VARCHAR2 (100 CHAR)
    ) ;

ALTER TABLE alcsoport
    ADD CONSTRAINT "alcsoport PK" PRIMARY KEY ( azon ) ;

CREATE TABLE szamlaforgalom
    (
     azon        NUMBER(12)          NOT NULL ,
     alcsop_azon NUMBER(12) ,
     bank        VARCHAR2 (10 CHAR)  NOT NULL ,
     szamlaszam  VARCHAR2 (50 CHAR),
     datum       DATE                NOT NULL ,
     jovairas    NUMBER ,
     terheles    NUMBER ,
     nev         VARCHAR2 (200 CHAR)
    ) ;

ALTER TABLE szamlaforgalom
    ADD CONSTRAINT "szamlaforgalom PK" PRIMARY KEY ( azon ) ;

CREATE TABLE csoport
    (
     csoport VARCHAR2 (50 CHAR)  NOT NULL
    ) ;

ALTER TABLE csoport
    ADD CONSTRAINT "csoport PK" PRIMARY KEY ( csoport ) ;
```

```
CREATE TABLE szamlaszam
    (
     szamlaszam VARCHAR2 (50 CHAR)   NOT NULL ,
     nev        VARCHAR2 (100 CHAR)  NOT NULL
    ) ;

ALTER TABLE szamlaszam
    ADD CONSTRAINT "szamlaszam PK" PRIMARY KEY ( szamlaszam );

CREATE TABLE terv
    (
     csoport  VARCHAR2 (50 CHAR)  NOT NULL ,
     datum    DATE                NOT NULL ,
     egyenleg NUMBER              NOT NULL
    ) ;

ALTER TABLE terv
    ADD CONSTRAINT "terv PK" PRIMARY KEY ( csoport, datum ) ;


ALTER TABLE  szamlaforgalom
    ADD CONSTRAINT szamlaforgalom_alcsop_fk FOREIGN KEY
(alcsop_azon)
    REFERENCES alcsoport (azon);

ALTER TABLE alcsoport
    ADD CONSTRAINT alcsop_csop_fk FOREIGN KEY ( csoport )
    REFERENCES csoport ( csoport);

ALTER TABLE terv
    ADD CONSTRAINT terv_csop_fk FOREIGN KEY ( csoport)
    REFERENCES csoport ( csoport);
```

Feltöltés adatokkal

Csak néhány rekordnyit mellékelek, a teljes feltöltés állományai a tanfolyami táblákban megtalálhatók.

```
REM INSERTING into TANF.CSOPORT
Insert into TANF.CSOPORT (CSOPORT) values ('Bankköltség');
Insert into TANF.CSOPORT (CSOPORT) values ('Biztosítás');
Insert into TANF.CSOPORT (CSOPORT) values ('Egyéb');
Insert into TANF.CSOPORT (CSOPORT) values ('Fizetés');
Insert into TANF.CSOPORT (CSOPORT) values ('Gyerekek');
Insert into TANF.CSOPORT (CSOPORT) values ('Hitel');
Insert into TANF.CSOPORT (CSOPORT) values ('Iskola');
Insert into TANF.CSOPORT (CSOPORT) values ('Készpénz
felvétel');
Insert into TANF.CSOPORT (CSOPORT) values ('Rezsi');
Insert into TANF.CSOPORT (CSOPORT) values ('Vásárlás');
Insert into TANF.CSOPORT (CSOPORT) values ('Átutalás');


REM INSERTING into TANF.ALCSOPORT
Insert into TANF.ALCSOPORT
(AZON,CSOPORT,ALCSOP,BANK,SZAMLASZAM,NEV) values
('1','Fizetés','Én','OTP','10023002-00299389',null);
Insert into TANF.ALCSOPORT
(AZON,CSOPORT,ALCSOP,BANK,SZAMLASZAM,NEV) values
('2','Fizetés','Te','OTP','10300002-20347563-00003285',null);
Insert into TANF.ALCSOPORT
(AZON,CSOPORT,ALCSOP,BANK,SZAMLASZAM,NEV) values
('20','Készpénz felvétel','-','OTP',null,'ATM-');
Insert into TANF.ALCSOPORT
(AZON,CSOPORT,ALCSOP,BANK,SZAMLASZAM,NEV) values
('30','Vásárlás','-','OTP',null,'VÁSÁRLÁS K');
Insert into TANF.ALCSOPORT
(AZON,CSOPORT,ALCSOP,BANK,SZAMLASZAM,NEV) values
('50','Rezsi','Gáz','OTP','10918001-00000003-51200061',null);
Insert into TANF.ALCSOPORT
(AZON,CSOPORT,ALCSOP,BANK,SZAMLASZAM,NEV) values
('51','Rezsi','Villany','OTP','13700016-02287016',null);
Insert into TANF.ALCSOPORT
(AZON,CSOPORT,ALCSOP,BANK,SZAMLASZAM,NEV) values
('90','Egyéb','-',null,null,null);
Insert into TANF.ALCSOPORT
(AZON,CSOPORT,ALCSOP,BANK,SZAMLASZAM,NEV) values
('40','Gyerekek','-','OTP',null,'Gyerek');
Insert into TANF.ALCSOPORT
(AZON,CSOPORT,ALCSOP,BANK,SZAMLASZAM,NEV) values
('58','Bankköltség','OTP','OTP',null,'KP.FELVÉT/-BEFIZ.
DÍJA ');
…
```

```
REM INSERTING into TANF.SZAMLASZAM
Insert into TANF.SZAMLASZAM (SZAMLASZAM,NEV) values
('11773487-11223344','BANK 1');
Insert into TANF.SZAMLASZAM (SZAMLASZAM,NEV) values
('19191919-19925199','BANK 2');
Insert into TANF.SZAMLASZAM (SZAMLASZAM,NEV) values
('10700079-22222222-51111115','BANK 3');
Insert into TANF.SZAMLASZAM (SZAMLASZAM,NEV) values
('16720013-11119999','BANK kölcsön');
Insert into TANF.SZAMLASZAM (SZAMLASZAM,NEV) values
('10700024-04040404-51111115','BANK lakáskölcsön');
Insert into TANF.SZAMLASZAM (SZAMLASZAM,NEV) values
('10023002-00299389','Tiéd munkahely');
Insert into TANF.SZAMLASZAM (SZAMLASZAM,NEV) values
('10300002-20347563-00003285','Enyém munkahely');
…

REM INSERTING into TANF.TERV
Insert into TANF.TERV (CSOPORT,DATUM,EGYENLEG) values
('Átutalás',to_date('12-JAN-01','RR-MON-DD'),'0');
Insert into TANF.TERV (CSOPORT,DATUM,EGYENLEG) values
('Bankköltség',to_date('12-JAN-01','RR-MON-DD'),'0');
Insert into TANF.TERV (CSOPORT,DATUM,EGYENLEG) values
('Biztosítás',to_date('12-JAN-01','RR-MON-DD'),'-2672');
Insert into TANF.TERV (CSOPORT,DATUM,EGYENLEG) values
('Egyéb',to_date('12-JAN-01','RR-MON-DD'),'-7542');
Insert into TANF.TERV (CSOPORT,DATUM,EGYENLEG) values
('Fizetés',to_date('12-JAN-01','RR-MON-DD'),'584394');
Insert into TANF.TERV (CSOPORT,DATUM,EGYENLEG) values
('Gyerekek',to_date('12-JAN-01','RR-MON-DD'),'-66400');
Insert into TANF.TERV (CSOPORT,DATUM,EGYENLEG) values
('Hitel',to_date('12-JAN-01','RR-MON-DD'),'-105030');
Insert into TANF.TERV (CSOPORT,DATUM,EGYENLEG) values
('Iskola',to_date('12-JAN-01','RR-MON-DD'),'-5000');
Insert into TANF.TERV (CSOPORT,DATUM,EGYENLEG) values
('Készpénz felvétel',to_date('12-JAN-01','RR-MON-DD'),'-
90000');
Insert into TANF.TERV (CSOPORT,DATUM,EGYENLEG) values
('Rezsi',to_date('12-JAN-01','RR-MON-DD'),'-35000');
Insert into TANF.TERV (CSOPORT,DATUM,EGYENLEG) values
('Vásárlás',to_date('12-JAN-01','RR-MON-DD'),'-272750');
…

REM INSERTING into TANF.SZAMLAFORGALOM
Insert into TANF.SZAMLAFORGALOM
(AZON,ALCSOP_AZON,BANK,SZAMLASZAM,DATUM,JOVAIRAS,TERHELES,NEV)
values ('1','82','AXA','11773487-11223344',to_date('12-JAN-
06','RR-MON-DD'),'300000',null,'Te ÉS Én LAFO, OTP Bank,
11773487-11223344 EB');
…
```

```
Insert into TANF.SZAMLAFORGALOM
(AZON,ALCSOP_AZON,BANK,SZAMLASZAM,DATUM,JOVAIRAS,TERHELES,NEV)
values ('12','60','CIB',null,to_date('12-JAN-02','RR-MON-
DD'),null,'313','JUTALÉK   HS-Havi számlavezetési díj
Értéknap: 2012.01.02');
…
Insert into TANF.SZAMLAFORGALOM
(AZON,ALCSOP_AZON,BANK,SZAMLASZAM,DATUM,JOVAIRAS,TERHELES,NEV)
values ('22','62','OTP',null,to_date('12-JAN-01','RR-MON-
DD'),null,'141','IDŐSZAKOS KÖLTSÉGEK');
…
Insert into TANF.SZAMLAFORGALOM
(AZON,ALCSOP_AZON,BANK,SZAMLASZAM,DATUM,JOVAIRAS,TERHELES,NEV)
values ('332','59','CIB',null,to_date('12-APR-27','RR-MON-
DD'),null,'4.55','KAMAT Értéknap: 2012.04.27');
```