8.előadás: Adatbázisok-1

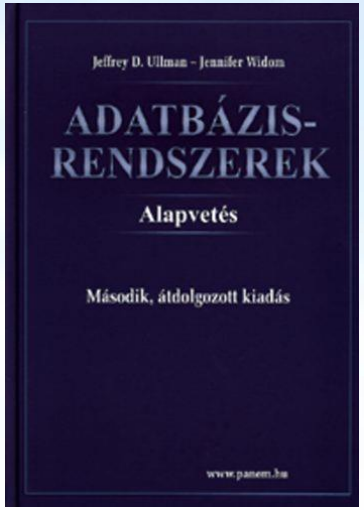
dr. Hajas Csilla (ELTE IK)

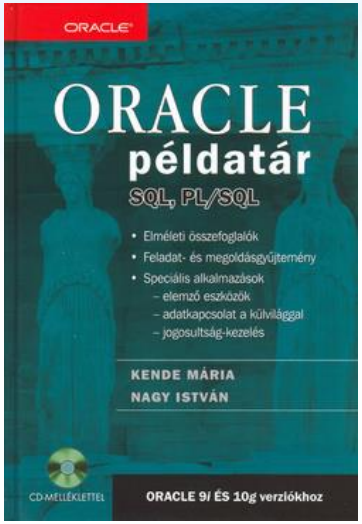
https://sila.hajas.elte.hu/

PL/SQL 1.rész [Tk-ben: SQL/PSM]

9.3. Az SQL és a befogadó nyelv közötti felület

9.4. SQL/PSM Sémában tárolt alprogramok





Gyakorlaton: Oracle PL/SQL

>> Kende Mária-Nagy István:

Oracle példatár (8-10.fejezet)

>> Gábor András-Juhász István:

PL/SQL programozás

Emlékeztető: SQL fő komponensei

- Az SQL elsődlegesen lekérdező nyelv (Query Language)
  SELECT utasítás (az adatbázisból információhoz jussunk)
- Adatkezelő nyelv, DML (Data Manipulation Language)
  INSERT, UPDATE, DELETE, SELECT
- Sémaleíró nyelv, DDL (Data Definition Language)
  CREATE, ALTER, DROP
- Tranzakció-kezelés
  COMMIT, ROLLBACK, SAVEPOINT
- Adatvezérlő nyelv, DCL (Data Control Language)
  GRANT, REVOKE
- Procedurális kiterjesztések
  SQL/PSM és a gyakorlatban Oracle PL/SQL

Emlékeztető: Az Eljut-feladat (Tk.10.2) Ezt most PL/SQL-ben is megírjuk!

Tankönyv 10.2. fejezet példája (az ELJUT feladat)

- Jaratok(legitarsasag, honnan, hova, koltseg, indulas, erkezes) táblában repülőjáratok adatait tároljuk.
- A járatok táblát létrehozó script:

  http://people.inf.elte.hu/sila/eduAB/jaratok_tabla.txt
- Mely (x,y) párokra lehet eljutni x városból y városba?
- Ezt egy relációs algebrai kifejezésként nem tudjuk megadni zárt alakban, klasszikus SQL SELECT utasítással sem tudjuk kifejezni, csak azt tudjuk, hogy átszállás nélkül, egy, két, stb… átszállással:

Az Eljut-feladatnak nincs algebrai megoldása

```
select distinct honnan, hova
    from jaratok
union
select j1.honnan, j2.hova
    from jaratok j1, jaratok j2
    where j1.hova=j2.honnan
union
select j1.honnan, j3.hova
    from jaratok j1, jaratok j2, jaratok j3
    where j1.hova=j2.honnan
    and j2.hova=j3.honnan
--- union stb... Ezt így nem lehet felírni…
```

Az Eljut-feladat Datalogban

Tankönyv 10.2. fejezet példája (az ELJUT feladat)

- Jaratok(legitarsasag, honnan, hova, koltseg, indulas, erkezes) EDB-táblában repülőjáratok adatait tároljuk.

  Mely (x,y) párokra lehet eljutni x városból y városba?

- Datalogban felírva (lineáris rekurzió)

  Eljut(x, y) <- Jaratok(l, x, y, k, i, e)
  Eljut(x, y) <- Eljut(x, z) AND Jaratok(l, z, y, k, i, e)

- Vagy másképp felírva Datalogban (mi a különbség?)

  Eljut(x, y) <- Jaratok(_, x, y, _, _, _) --- anonimus változók
  Eljut(x, y) <- Eljut(x, z) AND Eljut(z, y) --- nem lineáris rek.

Az Eljut feladat SQL-99 szabványban

- Datalog LINEÁRIS, MONOTON rekurzió átírható:
  Eljut(x, y) <- Jaratok(l, x, y, k, i, e)
  Eljut(x, y) <- Eljut(x, z) AND Jaratok(l, z, y, k, i, e)

- Hova, mely városokba tudunk eljutni Budapestről?

```
WITH RECURSIVE Eljut(honnan, hova) AS
        (SELECT honnan, hova FROM Jaratok
      UNION
          SELECT Eljut.honnan, Jaratok.hova
          FROM Eljut, Jaratok
          WHERE Eljut.hova = Jaratok.honnan)
SELECT hova FROM Eljut WHERE honnan='Bp';
```

SQL-99 szabvány: Rekurzív lekérdezés

- A WITH utasítás több ideiglenes relációra vonatkozó definíciója:

WITH [RECURSIVE] R_1 AS <R_1 definíciója>
[RECURSIVE] R_2 AS <R_2 definíciója>
...
[RECURSIVE] R_n AS <R_n definíciója>
< R_1,R_2,...,R_n relációkat tartalmazó lekérdezés>

Oracle SQL megoldások: with utasítással

- Az Oracle SQL a WITH RECURSIVE utasítást azt úgy támogatja, hogy WITH szerepel RECURSIVE nélkül, és UNION helyett UNION ALL-t lehet csak beírni
- WITH utasítással (Oracle 11gR2 verziótól használható)

```
WITH eljut (honnan, hova) as
   (select honnan, hova from jaratok
        UNION ALL
    select jaratok.honnan, eljut.hova
    from jaratok, eljut
    where jaratok.hova=eljut.honnan)
SEARCH DEPTH FIRST BY honnan SET SORTING
CYCLE honnan SET is_cycle TO 1 DEFAULT 0
select distinct honnan, hova from eljut order by honnan;
```

Oracle SQL megoldások: connect by

- SELECT DISTINCT hova FROM jaratok
  WHERE HOVA <> 'DAL'
  START WITH honnan = 'DAL'
  CONNECT BY NOCYCLE PRIOR hova = honnan;

- SELECT LPAD(' ', 4*level) || honnan, hova,
  level-1 Atszallasok,
  sys_connect_by_path(honnan||'->'||hova, '/'),
  connect_by_isleaf, connect_by_iscycle
  FROM jaratok
  START WITH honnan = 'SF'
  CONNECT BY NOCYCLE PRIOR hova = honnan;

Háromféle programozási megközelítés

1.) SQL kiterjesztése procedurális eszközökkel, az adatbázis séma részeként tárolt kódrészekkel, tárolt modulokkal (pl. PSM = Persistent Stored Modules, Oracle PL/SQL).

2.) Beágyazott SQL (sajátos előzetes beágyazás EXEC SQL. - Előfordító alakítja át a befogadó gazdanyelvre/host language, pl. C)

3.) Hívásszintű felület: hagyományos nyelvben programozunk, függvénykönyvtárat használunk az adatbázishoz való hozzáféréshez (pl. CLI = call-level interface, JDBC, PHP/DB)

SQL programnyelvi környezetben

- Milyen problémák merülnek fel, amikor egy alkalmazás részeként, programban használjuk az SQL utasításokat?

1.) Osztott változók használata: közös változók a nyelv és az SQL utasítás között (ott használható SQL utasításban, ahol kifejezés használható).

2.) A típuseltérés problémája: Az SQL magját a relációs adatmodell képezi. Reláció: gyűjtemény, sorok multihalmaza, mint adattípus nem fordul elő a magasszintű nyelvekben. A lekérdezés eredménye hogyan használható fel? Megoldás:

Lekérdezések használata a PSM-ben

- A típuseltérés problémája: Az SQL multihalmaz szemlélete hogyan egyeztethető össze a magas-szintű programnyelvekkel? A lekérdezés eredménye hogyan használható fel?
- Három esetet különböztetünk meg attól függően, hogy a SELECT FROM [WHERE stb] lekérdezés eredménye skalárértékkel, egyetlen sorral vagy egy listával (multihalmazzal) tér-e vissza.

Lekérdezések használata a PSM-ben

- SELECT eredményének használata:
  1. SELECT eredménye egy skalárértékkel tér vissza, elemi kifejezésként használhatjuk.
  2. SELECT egyetlen sorral tér vissza
     SELECT e_1, …, e_n INTO vált$_1$, … vált$_n$
     -- A végrehajtásnál visszatérő üzenethez az
     -- SQL STATE változóban férhetünk hozzá.
  3. SELECT eredménye több sorból álló tábla, akkor az eredményt soronként bejárhatóvá tesszük, kurzor használatával.

PL/SQL

Oracle PL/SQL 9-11.gyakorlatok témaköre:

- Blokkos szerkezet
- Változók, Típusok
- Vezérlési szerkezetek
- Kurzorok, kurzorváltozók
- Alprogramok, Tárolt eljárások és függvények
- Hiba-és kivételkezelés
- Triggerek
- Monoton rekurzív lekérdezések (Eljut-feladat)

PL/SQL – I.rész az alapok

- ELTE Adatbázisok gyakorlaton: Oracle PL/SQL
- **Oracle® Database PL/SQL Language Reference**
- PL/SQL
- Procedurális nyelv
- Az SQL DML-t kombinálja a procedurális nyelvi feldolgozással

(adatbázis + programozás)

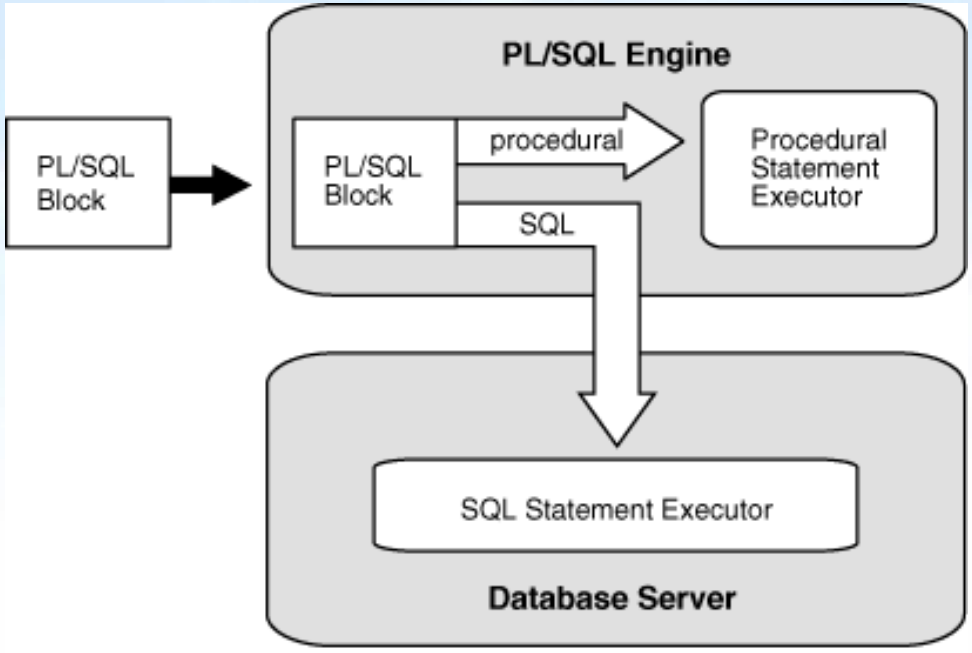


Figure 1-1 PL/SQL Engine

PL/SQL

- Egy PL/SQL blokk szerkezete:

```
[címke]
[DECLARE
    deklarációs utasítások ]
BEGIN
    végrehajtandó utasítások
    [ EXCEPTION
        kivételkezelés ]
END [név];
```

PL/SQL

- Eljárás deklaráció

```
PROCEDURE eljárás_neve [(formális
paraméterlista)]
IS
[deklarációs utasítások]
BEGIN
   végrehajtandó utasítások
   [EXCEPTION kivételkezelő]
END [név];
```

PL/SQL

- Függvény deklaráció

```
FUNCTION függvény_neve [(formális
paraméterlista)]
RETURN típus IS
[deklarációs utasítások]
BEGIN
   végrehajtandó utasítások
   [EXCEPTION kivételkezelő]
END [név];
```

PL/SQL

- A különbség az eljárás és a függvény között
  - Eljárás: direkt módon nem adnak vissza értéket, általában utasítások lefuttatása a cél (illetve logikailag egy egységbe tartozó utasítások egy helyen kezelése)
  - Függvény: visszaad egy értéket, általában arra használják, hogy kiszámítsanak valamit és azt visszaadják.

PL/SQL – Deklarációs rész ---1

- Tartalma lehet
  - Típus definíció
  - Változó deklaráció

  Név típus [[NOT NULL] {:= | DEFAULT} kifejezés];

  Példák: belépési idő változó, illetve dolgozók száma változó és az alapértelmezett értéke 0.

  belepesi_ido DATE;
  dolg_szama NUMBER NOT NULL DEFAULT 0;
  dolg_fizetes NUMBER NOT NULL := 1000;

PL/SQL – Deklarációs rész ---2

- Tartalma lehet
  - Nevesített konstans deklaráció
    - Név CONSTANT típus [NOT NULL] {:= | DEFAULT} kifejezés;

      Példa: fizetés konstans, melynek értéke 1000.

      fizetes CONSTANT NUMBER := 1000;
  - Kivétel deklaráció
  - Kurzor definíció
  - Alprogram definíció

PL/SQL – Adattípusok ---1

- Logikai (ez új, nem volt a create table esetén)
  - `BOOLEAN` --- 3-értékű logika

- Numerikus
  - `NUMBER` – ez így lebegőpontos
  - `NUMBER(3)` – ez így fixpontos
  - `FLOAT` – nem korlátozott lebegőpontos
  - `INT, INTEGER, SMALLINT` – korlátozott fixpontos
  - stb …

PL/SQL – Adattípusok---2

- Karakteres
  - CHAR
  - VARCHAR2
  - NVARCHAR2
  - stb…
- Dátum
  - DATE
  - TIMESTAMP(p)
  - INTERVAL

PL/SQL – Adattípusok---3

- A deklarációban a típus lehet
  - Skalár adattípus
  - Hivatkozási típus: %TYPE, %ROWTYPE
    - Változónév / rekordnév / adatbázis_tábla_név.oszlopnév / kurzorváltozó_név / kollekciónév / objektumnév%TYPE
    - Adatbázis_táblanév / kurzornév%ROWTYPE

PL/SQL – Adattípusok---4

- PL/SQL Ref.: Example 2-24 Assigning Values

```
DECLARE  -- You can assign initial values here
  counter        NUMBER := 0;
  done           BOOLEAN;
  emp_rec        employees%ROWTYPE;
BEGIN  -- You can assign values here too
  done := (counter > 100);
  emp_rec.first_name := 'Antonio';
  emp_rec.last_name := 'Ortiz';
END;
/
```

PL/SQL – Adattípusok---5

- Rekord típus deklaráció
  - TYPE név IS RECORD (
    mezőnév típus [[NOT NULL] {:= | DEFAULT} kifejezés], ...);
  - Példa: telefonkönyv rekord
    TYPE telkonyv IS RECORD (
    szam NUMBER,
    nev VARCHAR2(20));
- Rekord deklaráció
  telefonkonyv telkonyv;
- Rekord mezőjének elérése
  telefonkonyv.nev;

PL/SQL – Adattípusok---6

- Altípusok
  - `SUBTYPE név IS alaptípus_név[(korlát)] [NOT NULL];`
  - Példa: beépített altípus az INTEGER
    `SUBTYPE INTEGER IS NUMBER(38,0);`
- Tömbök
  - `TYPE típusnév IS VARRAY(n) OF <elemek típusa>;`
  - Példa: `TYPE szamtomb IS VARRAY(10) OF NUMBER;`

PL/SQL - Típuskonverzió

- Implicit a skalártípusok között
- Explicit a beépített függvények használatával
  - TO_DATE
  - TO_NUMBER
  - TO_CHAR

PL/SQL – Kiírás a konzolra

- A PL/SQL nem tartalmaz I/O utasításokat.
- A DBMS_OUTPUT csomag segítségével üzenetet helyezhetünk el egy belső pufferbe.
- PUT_LINE eljárás üzenetet ír a pufferbe
- A puffer tartalmát a SET SERVEROUTPUT ON utasítással jeleníthetjük meg a képernyőn

- Példa: Hello World!

```
SET SERVEROUTPUT ON
BEGIN
  DBMS_OUTPUT.PUT_LINE('Hello World!');
END;
/
```

PL/SQL – Utasítások

- Üres
  - NULL;
- Értékadó
  - X := 0;
- Ugró
  - GOTO címke;

PL/SQL – Utasítások

- Elágazás
  - IF
  - CASE
- Ciklusok
  - Végtelen
  - WHILE
  - FOR
  - Kurzor FOR (később)
- SQL utasítások

PL/SQL – IF utasítás

- Szintaxis:

```
IF (feltétel)
  THEN utasítás [utasítás] ...
  [ELSIF (feltétel)
    THEN utasítás [utasítás] ... ] ...
  [ELSE utasítás [utasítás] ... ]
END IF;
```

PL/SQL – IF utasítás

```
SET SERVEROUTPUT ON
DECLARE
   a number(3) := 100;
BEGIN
   IF ( a = 10 ) THEN
      dbms_output.put_line('Value of a is 10' );
   ELSIF ( a = 20 ) THEN
      dbms_output.put_line('Value of a is 20' );
   ELSIF ( a = 30 ) THEN
      dbms_output.put_line('Value of a is 30' );
   ELSE
       dbms_output.put_line('None of the values is matching');
   END IF;
   dbms_output.put_line('Exact value of a is: '|| a );
END;
/
```

PL/SQL – CASE utasítás

- Szintaxis:

```
CASE kifejezés
  WHEN érték1 THEN utasítás1;
  ...
  ELSE utasítás
END CASE;
```

PL/SQL – CASE utasítás

```
SET SERVEROUTPUT ON
DECLARE
   grade char(1) := 'A';
BEGIN
   CASE grade
      when 'A' then dbms_output.put_line('Excellent');
      when 'B' then dbms_output.put_line('Very good');
      when 'C' then dbms_output.put_line('Well done');
      when 'D' then dbms_output.put_line('You passed');
      when 'F' then dbms_output.put_line('Better try
again');
      else dbms_output.put_line('No such grade');
   END CASE;
END;
/
```

PL/SQL – LOOP utasítás

- Végtelen ciklus
- Szintaxis:

```
LOOP
utasítás(ok);
END LOOP;
```

- EXIT-re lép ki
  - Ehelyett használható EXIT WHEN (feltétel) is

PL/SQL – LOOP utasítás

```
SET SERVEROUTPUT ON
DECLARE
   x number := 10;
BEGIN
   LOOP
      dbms_output.put_line(x);
      x := x + 10;
      IF x > 50 THEN
         exit; -- itt lep majd ki
      END IF;
   END LOOP;
   dbms_output.put_line('After Exit x is: ' || x);
END;
/
```

PL/SQL – WHILE utasítás

- Előltesztelős ciklus
- Szintaxis:

```
WHILE feltétel LOOP
utasítás(ok);
END LOOP;
```

PL/SQL – WHILE utasítás

```
SET SERVEROUTPUT ON
DECLARE
   a number(2) := 10;
BEGIN
   WHILE a < 20 LOOP
      dbms_output.put_line('value of a: ' || a);
      a := a + 1;
   END LOOP;
END;
```

PL/SQL – FOR utasítás

- Számlálós ciklus
- Szintaxis:

FOR számláló IN [REVERSE] kezdőérték .. Végérték LOOP
utasítás(ok);
END LOOP;

PL/SQL – FOR utasítás

```
SET SERVEROUTPUT ON
DECLARE
   a number(2);
BEGIN
   FOR a in 10 .. 20 LOOP
      dbms_output.put_line('value of a: ' || a);
  END LOOP;
END;
```

SQL utasítások PL/SQL-ben

- Nem használható SELECT, csak spec.esetben
  - amikor egy sort ad vissza kiegészül egy INTO (ill. ált. BULK COLLECT INTO) utasításrésszel
- DML utasítások: INSERT, DELETE, UPDATE
  - kiegészülnek egy RETURNING utasításrésszel, segítségével az érintett sorok alapján számított értéket kaphatunk meg
- MERGE
  - „UPSERT” funkcionalitás
- Tranz.kez.: COMMIT, ROLLBACK, SAVEPOINT

SQL utasítások PL/SQL-ben

- SELECT értékének kiválasztása egy változóba
  - SELECT select_kifejezés INTO változónév
    FROM táblanév;
- Példa: King adatainak tárolása a dolg változóban:
  - DECLARE
    dolg dolgozo%ROWTYPE;
    BEGIN
    SELECT * INTO dolg
    FROM dolgozo
    WHERE dnev='KING';
    END;

SQL utasítások PL/SQL-ben

- PL/SQL Ref: Example 2-25 SELECT INTO

```
DECLARE
  bonus   NUMBER(8,2);
BEGIN
  SELECT salary * 0.10 INTO bonus
  FROM employees
  WHERE employee_id = 100;
DBMS_OUTPUT.PUT_LINE('bonus = ' || TO_CHAR(bonus));
END;
/
```

SQL utasítások PL/SQL-ben

- Törlés egy táblából

```
DELETE [FROM] táblahivatkozás
[WHERE feltétel]
[returning utasításrész];
```

- A RETURNING formája

```
RETURNING
 egysoros select kifejezés[, ...]
 INTO {változó[, ...] | rekord};
```

SQL utasítások PL/SQL-ben

- Beszúrás egy táblába

```
INSERT INTO táblahivatkozás
 [(oszlop, ...)]
VALUES
 {(sql_kifejezés, [...]) | rekord}
[returning utasításrész];
```

SQL utasítások PL/SQL-ben

- Táblában érték módosítása

```
UPDATE táblahivatkozás
SET oszlop=sql_kifejezés [, ...]
 [WHERE feltétel]
[returning utasításrész];
```

SQL utasítások PL/SQL-ben

```
DECLARE        -- PL/SQL Ref.: Example 6-1 Static SQL Statements
  emp_id     employees.employee_id%TYPE := 299;
  emp_first_name  employees.first_name%TYPE  := 'Bob';
  emp_last_name   employees.last_name%TYPE   := 'Henry';
BEGIN
   INSERT INTO employees (employee_id, first_name, last_name)
            VALUES (emp_id, emp_first_name, emp_last_name);
   UPDATE employees
            SET first_name = 'Robert'
            WHERE employee_id = emp_id;
   DELETE FROM employees
            WHERE employee_id = emp_id
            RETURNING first_name, last_name
            INTO emp_first_name, emp_last_name;
   COMMIT;
   DBMS_OUTPUT.PUT_LINE (emp_first_name || ' ' || emp_last_name);
END;
/
```

SQL utasítások PL/SQL-ben

- PL/SQL Ref.: Example 6-4 SQL%ROWCOUNT

```
DROP TABLE emp_temp;
CREATE TABLE emp_temp AS
      SELECT * FROM employees;
DECLARE
      mno NUMBER(6) := 122;
BEGIN
  DELETE FROM emp_temp WHERE manager_id = mno;
  DBMS_OUTPUT.PUT_LINE ('Number of employees
            deleted: ' || TO_CHAR(SQL%ROWCOUNT));
END;
/
```

PL/SQL - II.rész folyt.jön 9.előadáson: Kurzorok, Alprogramok, Triggerek, Kivétel- és hibakezelés

- A következő 9.előadáson folytatjuk a PL/SQL-t
- Most a 8.előadás hátralevő részében az eddigi tananyaghoz példaként nézzük meg, hogy az Eljut feladatot hogyan oldjuk meg PL/SQL-ben!

Eljut-feladat (monoton rekurzió) megoldása PL/SQL programmal

Tankönyv 10.2. fejezet példája (az ELJUT feladat)

- Jaratok(legitarsasag, honnan, hova, koltseg, indulas, erkezes) táblában repülőjáratok adatait tároljuk.
- A járatok táblát létrehozó script:

  http://people.inf.elte.hu/sila/eduAB/jaratok_tabla.txt
- Mely (x,y) párokra lehet eljutni x városból y városba?
- Ezt egy relációs algebrai kifejezésként nem tudjuk megadni zárt alakban, klasszikus SQL SELECT utasítással sem tudjuk kifejezni, csak azt tudjuk, hogy átszállás nélkül, egy, két, stb… átszállással:

Az Eljut-feladatnak nincs algebrai megoldása

```
select distinct honnan, hova
    from jaratok
union
select j1.honnan, j2.hova
    from jaratok j1, jaratok j2
    where j1.hova=j2.honnan
union
select j1.honnan, j3.hova
    from jaratok j1, jaratok j2, jaratok j3
    where j1.hova=j2.honnan
    and j2.hova=j3.honnan
--- union stb... Ezt így nem lehet felírni…
```

Az Eljut feladat SQL-99 szabványban

- Datalog LINEÁRIS, MONOTON rekurzió átírható:
  Eljut(x, y) <- Jaratok(l, x, y, k, i, e)
  Eljut(x, y) <- Eljut(x, z) AND Jaratok(l, z, y, k, i, e)

- Hova, mely városokba tudunk eljutni Budapestről?

```
WITH RECURSIVE Eljut AS
        (SELECT honnan, hova FROM Jaratok
      UNION
          SELECT Eljut.honnan, Jaratok.hova
          FROM Eljut, Jaratok
          WHERE Eljut.hova = Jaratok.honnan)
SELECT hova FROM Eljut WHERE honnan='Bp';
```

Rekurzív lekérdezések

- Datalog rekurzió segít megérteni az SQL-99 szabványban bevezetett rekurzív lekérdezések WITH RECURSIVE záradékát.
- A BSc-n csak MONOTON rekurziót vesszük, vagyis nem használjuk nem-monoton különbség műveletet, nincs csoportosítás-aggregálás (ugyanis az olyan lekérdezések, amelyek nem-monotonok, megengedik a negációt és aggregálást az olyan különös hatással van a rekurzióra, ezt csak MSc kurzusokon vesszük).
- Nézzük meg a monoton rekurzív Eljut-feladatnak a megvalósítását Oracle PL/SQL programmal

Eljut feladat PL/SQL-ben ---1

- **Rek1.feladat:** Mely (x, y) várospárokra lehet egy vagy több átszállással eljutni x városból y városba?
- Ehhez hozzuk létre eljut(honnan,hova) táblát,

```
DROP TABLE eljut;
CREATE TABLE eljut(
        honnan VARCHAR2(10),
        hova VARCHAR2(10));
```

- Írjunk egy olyan PL/SQL programot, ami feltölti az ELJUT táblát a sorait a járatok tábla alapján (ehhez ciklust szervezni, az insert több sor felvitele 2.alakja alkérdéssel járatok és eljut táblák alapján)

Eljut feladat PL/SQL-ben ---2

- Az ELJUT feladat megoldása Oracle PL/SQL-ben
- A ciklus során ellenőrizni kell, hogy addig hajtsuk végre a ciklust, amíg növekszik az eredmény (Számláló)
- DECLARE RegiSzamlalo Integer;
  UjSzamlalo Integer;
- Deklarációs rész után BEGIN … END; között az utasítások, először az eljut táblának kezdeti értéket adunk (a megvalósításnál az INSERT-nél figyelni, hogy ne legyenek ismétlődő sorok: select distinct)

  delete from eljut;

  insert into eljut (SELECT distinct honnan, hova
  FROM jaratok);

Eljut feladat PL/SQL-ben ---3

- Szamlalo változóknak adunk kiindulási értéket:

```
RegiSzamlalo := 0;
select count(*) into UjSzamlalo from eljut;
```

- A ciklust addig kell végrehajtani, amíg növekszik az eredmény (Szamlalo) duplikátumokra figyelni!

```
LOOP
   insert into eljut (lásd a köv.oldalon…)
   select count(*) into UjSzamlalo from eljut;
   EXIT WHEN UjSzamlalo = RegiSzamlalo;
   RegiSzamlalo := UjSzamlalo;
END LOOP;
commit;
```

Eljut feladat PL/SQL-ben ---4

- Az eljut tábla növelése a ciklusban, figyelni kell a duplikátumokra, csak olyan várospárokat vegyünk az eredményhez, ami még nem volt!

```
insert into eljut
    (select distinct eljut.honnan, jaratok.hova
     from eljut, jaratok   --- *from (lineáris rekurzió)
     where eljut.hova = jaratok.honnan
     and (eljut.honnan,jaratok.hova)
          NOT IN (select * from eljut));
```

- Megjegyzés: PSM-ben a nem-lineáris rekurzió is megengedett: from eljut e1, eljut e2 ---*from-ban

Eljut feladat PL/SQL-ben ---5

- **Rek2.feladat:** Mely (x,y) város párokra hány darab átszállással és milyen költségekkel lehetséges egy vagy több átszállással eljutni x városból y városba?
- Ehhez készítsünk Eljut2(honnan, hova, atszallas, koltseg) táblát. Írjunk egy olyan PL/SQL programot, ami feltölti az ELJUT táblát.

- **Rek3.feladat:** Tegyük fel, hogy nemcsak az érdekel, hogy el tudunk-e jutni az egyik városból a másikba, hanem az is, hogy utazásunk során az átszállások is ésszerűek legyenek, ami azt jelenti, hogy ha több járattal utazunk, akkor nézni kell átszálláskor az érkező járatnak legalább egy órával a rákövetkező indulás előtt meg kell érkeznie, és 6 óránál ne kelljen többet várnia.

Kérdés/Válasz

- Köszönöm a figyelmet!
- Kérdés/Válasz?

- Gyakorlás az Oracle Példatár feladatai:
  - 8.fejezet PL/SQL példái és feladatai