ADATBÁZIS ALAPÚ RENDSZEREK

PL/SQL I.

NYELVI ALAPOK

- ***PL/SQL = Procedural Language/SQL***

- utasítást `;` zárja le
- PL/SQL *blokk* lezárása: `/`
- kis- és nagybetű egyenértékű (az utasításokban a kulcsszavakat szoktuk nagybatűvel írni, de nem kötelző)
- comment `REM` vagy `--`, többsoros `/* */`
- használat előtt deklarálni kell a változókat, azonosítókat, eljárásokat
- `**` hatványozás, `!=` nem egyenlő, `||` karakterlánc összefűzés
- egy PL/SQL program egy vagy több *blokkból* áll, a blokkok egymásba ágyazhatók

PL/SQL BLOKK FELÉPÍTÉSE

```
[DECLARE
      [változó deklarációk] ]
BEGIN
      [szekvenciális utasításblokk]
[EXCEPTION
      [kivételkezelés] ]
END;
```

1. PÉLDA

```
VAR X NUMBER
DECLARE
      a NUMBER;
BEGIN
      a := 3;
      :X := a + 3;
END;
.

PRINT X
```

VÁLTOZÓK DEKLARÁCIÓJA

```
DECLARE
        változónév [CONSTANT] adattípus [NOT NULL] [DEFAULT érték];
BEGIN
        […]
END;
```

2. példa

```
DECLARE
        a CONSTANT NUMBER := 3;
        b NUMBER NOT NULL := 5;
BEGIN
        :X := :X + a + b + 3;
END;
.

PRINT X
```

DBMS_OUTPUT (PL/SQL CSOMAG)

- használatát a **SET SERVEROUTPUT ON** SQL*Plus paranccsal engedélyezni kell
- PUT: ugyanabba a sorba ír több outputot
- NEW_LINE: sor végét jelzi
- PUT_LINE: minden outputot külön sorba ír

3. példa

```
SET SERVEROUTPUT ON
ACCEPT nev PROMPT 'Kerem adja meg a nevét: '
DECLARE
        szoveg varchar2(50);
BEGIN
        szoveg := CONCAT('&nev',' sikeresen végrehajtotta a
programot!');
        DBMS_OUTPUT.PUT_LINE (szoveg);
END;
```

SELECT … INTO … - ÉRTÉKADÁS

- a “klasszikus” SELECT utasítás PL/SQL blokkon belüli használatakor az INTO alparanccsal változó(k)nak adhatunk értéket

4. példa

```
CREATE TABLE T1(
        e INTEGER,
        f INTEGER
);

INSERT INTO T1 VALUES(1, 3);
INSERT INTO T1 VALUES(2, 4);

DECLARE
        a NUMBER;
        b NUMBER;
BEGIN
        SELECT e,f INTO a,b FROM T1 WHERE e>1;
        INSERT INTO T1 VALUES(b,a);
END;
```

REKORDOK, ADATTÁBLÁBÓL TÍPUS KINYERÉSE

- ***Rekord***: n db adott típusú adatból álló rendezett n-es (például: egy tábla egy sora)
- először mindig megadjuk, mint új típus, majd "példányok" létrehozása
- `TYPE rekordtípusnév IS RECORD (mezőnév típus, ..., mezőnév típus)`

5. példa

```
DECLARE
        TYPE dolgozo_rekord IS RECORD (
                adoszam INTEGER,
                nev CHAR(30),
                lakcim VARCHAR(50));
        egy_dolgozo dolgozo_rekord;
BEGIN
        egy_dolgozo.nev = 'Kovacs';
END;
```

REKORDOK, ADATTÁBLÁBÓL TÍPUS KINYERÉSE

- Lehetőség van egy létező adattábla alapján valamely attribútum típusának, vagy a tábla egy sorával ekvivalens rekordtípusnak a kinyerésére: %TYPE, %ROWTYPE

6. példa

```
DECLARE
        v_veznev DEMO.MUNKATARS.VEZETEKNEV%TYPE;
        v_kernev DEMO.MUNKATARS.KERESZTNEV%TYPE;
BEGIN
        SELECT vezeteknev, keresztnev
        INTO v_veznev, v_kernev
        FROM DEMO.munkatars
        WHERE vezeteknev LIKE 'Nagy'
        AND keresztnev LIKE 'Elek';
        DBMS_OUTPUT.PUT_LINE('Teljes neve: ' || v_veznev || ' ' || v_kernev);
END;
```

%ROWTYPE PÉLDA

```
DECLARE
      v_sor DEMO.vevo%ROWTYPE;
BEGIN
      SELECT * INTO v_sor
      FROM DEMO.vevo
      WHERE partner_id = 21;

DBMS_OUTPUT.PUT_LINE(v_sor.MEGNEVEZES);
END;
```

VEZÉRLÉSI SZERKEZETEK

```
DECLARE
        […]
BEGIN
    .
    .
    .
END;
```

- **Feltételes elágazás:**
  `IF … ELSIF … ELSE;`
- **Esetkiválasztás:**
  `CASE … END CASE;`
- **Ismétlés:** `LOOP … END LOOP;`
  - **előfeltételes:**
    `WHILE … LOOP … END LOOP;`
  - **utófeltételes :**
    `LOOP … EXIT WHEN … END LOOP;`
  - **számlálásos/diszkrét:**
    `FOR … LOOP … END LOOP;`

`LOOP … END LOOP;`
Használható önmagában is, ekkor tkp. végtelen ciklus.
Megszakítható: `EXIT [WHEN feltétel];` utasítás

VEZÉRLÉSI SZERKEZETEK
IF … ELSIF … ELSE

```
DECLARE
        v_avgber        DEMO.munkatars.ber%TYPE;
        szoveg          VARCHAR2(50);
begin
        SELECT AVG(ber)
        INTO v_avgber
        FROM DEMO.munkatars;
        IF v_avgber < 100000 THEN
                szoveg:='kevesebb mint szazezer';
        ELSIF (v_avgber > 100000) AND (v_avgber <= 200000) THEN
                szoveg:='szazezer es ketszazezer kozt';
        ELSE
                szoveg:='ketszazezer folott';
        END IF;
        DBMS_OUTPUT.PUT_LINE(szoveg);
END;
```

VEZÉRLÉSI SZERKEZETEK
CASE … END CASE

```
CASE num
      WHEN 1 THEN dbms_output.put_line('Egy');
             .
             .
             .
      WHEN 9 THEN dbms_output.put_line('Kilenc');
      ELSE dbms_output.put_line('Nem egyjegyu');
END CASE;
```

VEZÉRLÉSI SZERKEZETEK

LOOP … END LOOP + EXIT WHEN …

```
DECLARE
      v_sorsz     DEMO.vevo.partner_id%TYPE := 21;
      v_megnev    DEMO.vevo.megnevezes%TYPE;
BEGIN
      LOOP
            SELECT megnevezes
            INTO v_megnev
            FROM DEMO.vevo
            WHERE partner_id = v_sorsz;
            DBMS_OUTPUT.PUT_LINE(v_megnev);
            v_sorsz := v_sorsz +1;
      EXIT WHEN v_sorsz > 28;
      END LOOP;
END;
```

VEZÉRLÉSI SZERKEZETEK
WHILE … LOOP … END LOOP;

```
DECLARE
      v_sorsz     DEMO.vevo.partner_id%TYPE := 21;
      v_megnev    DEMO.vevo.megnevezes%TYPE;
BEGIN
      WHILE v_sorsz < 29
      LOOP
            SELECT megnevezes
            INTO v_megnev
            FROM DEMO.vevo
            WHERE partner_id = v_sorsz;
            DBMS_OUTPUT.PUT_LINE(v_megnev);
            v_sorsz := v_sorsz +1;
      END LOOP;
END;
```

VEZÉRLÉSI SZERKEZETEK
FOR … LOOP … END LOOP

- FOR ciklusváltozó IN [REVERSE] alsóhatár .. felsőhatár LOOP ... END LOOP; **(számlálás)**
- FOR rekord IN (SELECT …) LOOP … END LOOP; **(kurzorok – később)**

```
FOR num IN 1..500 LOOP
          INSERT INTO roots VALUES (num, SQRT(num));
END LOOP;
```

GYŰJTŐTÁBLÁK

- részben listaszerűen, részben tömbszerűen kezelhető adatstruktúra
- oszlopai: adat (oszlop vagy rekord típusú lehet), index (`BINARY INTEGER`)
- mérete nem korlátozott
- tömbként indexelhető, ahol az index NEGATÍV is lehet: adat(index)
- új sorokkal bővíthető és törölhető → hézagok lehetnek benne

- **létrehozás:** `TYPE táblatípusnév IS TABLE OF {oszloptípus | rekordtípus} INDEX BY BINARY_INTEGER`
- az `INSERT, UPDATE, FETCH, SELECT` utasításokkal kezelhető
- gyűjtőtábla metódusok

`EXISTS(n)`	- igaz, ha létezik az n-edik elem
`COUNT`	- táblában lévő elemek száma
`FIRST/LAST`	- tábla első/utolsó indexértéke
`NEXT`	- a táblában a következő index értéke
`DELETE(n)`	- az n-edik elem törlése

GYŰJTŐTÁBLÁK / PÉLDA

```
DECLARE
        TYPE tipus IS TABLE OF DEMO.vevo%ROWTYPE
        INDEX BY BINARY_INTEGER;
        valtozo tipus;
        ind BINARY_INTEGER := 1;
BEGIN
        LOOP
                SELECT *
                INTO valtozo(ind)
                FROM DEMO.vevo
                WHERE partner_id = 20 + ind;
                ind := ind + 1;
        EXIT WHEN ind > 8;
        END LOOP;
        ind := valtozo.FIRST;
        LOOP

DBMS_OUTPUT.PUT_LINE(valtozo(ind).megnevezes);
                ind := ind + 1;
        EXIT WHEN ind > valtozo.LAST;
        END LOOP;
END;
```

KIVÉTELKEZELÉS

- A más nyelveken megismert kivételkezelés itt is működik:

```
DECLARE
        [...]
BEGIN
        [...]
EXCEPTION
        WHEN [...]
END;
```

- különböző hibákra más-más kivétel (akár sajátok is felvehetők, `RAISE` utasítás)
- `EXCEPTION WHEN kivétel [OR kivétel ...] utasítások, [WHEN OTHERS utasítások]`
- `NO_DATA_FOUND`: SELECT utasítás nem ad vissza sort
- `TOO_MANY_ROWS`: egy sort kellett volna visszaadnia egy SELECT-nek, de többet kaptunk
- `INVALID_NUMBER`: karakterlánc sikertelen számmá konvertálása
- `DUP_VAL_ON_INDEX`: kulcsfeltétel megsértése

KIVÉTELKEZELÉS / PÉLDA

```
DECLARE
        v_ber DEMO.munkatars.ber%TYPE;
        v_veznev DEMO.munkatars.vezeteknev%TYPE;
        v_kernev DEMO.munkatars.keresztnev%TYPE;
BEGIN
        v_ber := 200000;
        SELECT vezeteknev, keresztnev
        INTO v_veznev, v_kernev
        FROM DEMO.munkatars
        WHERE ber = v_ber;
        DBMS_OUTPUT.PUT_LINE(v_veznev || ' ' || v_kernev);
EXCEPTION
        WHEN NO_DATA_FOUND THEN
                DBMS_OUTPUT.PUT_LINE('Nincs ilyen fizetés');
        WHEN TOO_MANY_ROWS THEN
                DBMS_OUTPUT.PUT_LINE('Több embernek is ez a
fizetése');
        WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('Egyéb hiba');
END;
```

SAJÁT KIVÉTELEK

- a `DECLARE` szakaszban: `kivételnév EXCEPTION`
- végrehajtható szegmensben `RAISE` utasítással váltható ki

```
DECLARE
        nincs_vevo EXCEPTION;
BEGIN
        DBMS_OUTPUT.PUT_LINE('Ez vegrehajtodik');
        RAISE nincs_vevo;
        DBMS_OUTPUT.PUT_LINE('Ez nem hajtodik vegre');
EXCEPTION
        WHEN nincs_vevo THEN
                DBMS_OUTPUT.PUT_LINE('Nincs ilyen vevo');
        WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('Egyéb hiba');
END;
```

FELADATOK

1. Kérj be két egész számot, és döntsd el, hogy az összegük páros vagy páratlan!
2. Írj egy programot, ami kiírja az EMP tábla sorainak számát, és a tárolt dolgozók átlagfizetését.
3. Adj meg egy programot, ami létrehozza a FIBON(n, ertek) táblát, és feltölti azt n=100-ig a Fibonacci-számokkal!
4. Adj egy programot, ami kiszámolja a felhasználó által megadott telephelyen dolgozók bérösszegét (EMP).