12. gyakorlat

I. PLSQL

1. Blokk

A PL/SQL alapvető egysége a blokk. Lehet nevet adni neki, ekkor az adatbázisban tárolódik, és többször meghívható újra. Maradhat név nélkül is, ekkor nem tárolódik az adatbázisban, és nem használható fel újra, hanem azonnal végrehajtható.

Egy blokk általános kinézete:

```
DECLARE
     Variable declaration
BEGIN
     Program Execution
EXCEPTION
     Exception handling
END;
```

Declare: típus, nevesített konstans, változó, kivétel, kurzor

Szintaxisa:

```
identifier [CONSTANT] datatype [NOT NULL] [:= | DEFAULT expr];
```

Például:

```
konstansom CONSTANT NUMBER NOT NULL:=42;
```

konstans érték megadása

vagy:

```
valtozom VARCHAR2(40) := 'ezakezdetiértek';
```

kezdeti érték megadása

2. A % típusú attribútumok

Ez nagyon praktikus, ha a blokkunk egy meglevő adatbázison szeretne majd dolgozni, jó, ha nem kell a dokumentációt végigböngészni, hogy milyen típusú egy-egy attribútum. Ám azt tudjuk, hogy nagyon fontos ez az információ, hiszen adott adatot csak megfelelő típusú helyre lehet beilleszteni, stb.. egyszóval mindig tudni kell, milyen típusú adatokkal dolgozunk. Erre jó a %TYPE.

Ezzel tehát a % előtt meghatározott tábla meghatározott attribútumának típusát „vesszük át”.

Például:

```
termname go13.term.name%TYPE;
```

a „termname” nevű váltózónk típsua az lesz, ami az „term” táblában a „TYPE” attribútumé.

A%ROWTYPE egy egész sor típusát megában foglalja, vagyis tulajdonképpen a fejlécet.

Például:

```
term_egysor go13.term%ROWTYPE;
```

Az „egysor” nevű változónk elbír annyi, és olyan típusú adatot, amilyen sorok a „**term**” táblánkban szerepelnek.

Rekord: a programozó által definiálható változó (struct)

Kurzor: SELECT állítás a deklarásiós részben definiálva és elnevezve, azaz:

```
CURSOR kurzor_neve IS SELECT ... FROM... WHERE...;
```

Program E. : amit a blokk végrehajt (pl. feltétel, ciklus, SQL parancs)

Lekérdezés bokkban:

pl.

```
DECLARE
     v_ename VARCHAR2(10);
     v_id NUMBER(3);
BEGIN
     SELECT ename, id
     INTO v_ename, v_id
     FROM emp
     WHERE id = '112';
END;
```

3. Elágazás

Ha a kiválasztott sorra valami igaz vagy nem igaz, ez vagy az történjen.

Szintaxisa:

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

Case utasítás egy elágazó utasítás, ahol az egymást követő kölcsönösen kizáró tevékenységek közül egy kifejezés értékei, vagy feltételek teljesülése szerint lehet választani.

Szintaxisa:

```
CASE [kifejezés] --kifejezés: amit a feltételekkel
összehasonlítunk
     WHEN {kifejezés | feltétel} THEN utasítás
[utasítás]...
     [WHEN {kifejezés | feltétel} THEN utasítás
[utasítás]...]...
     [ELSE utasítás [utasítás]...]
END CASE;
```

4. Ciklus

1. egyszerű ciklus:
LOOP mitörténjen
EXIT WHEN kilépési feltétel
END LOOP;

2. for ciklus:
FOR i IN 1..10 LOOP
mitörténjen
END LOOP;

3. while ciklus:
WHILE i<|>|==valami LOOP
mitörténjen
END LOOP;

5. Függvények, eljárások létrehozása

```
CREATE OR REPLACE FUNCTION fgvneve
(paraméterlista)
RETURN visszatérési érték típusa
IS
     begin
     end
```

Eljárás:

```
CREATE OR REPLACE PROCEDURE procneve
(paraméterlista)
IS
innentől egy rendes blokk…
```

Később a nevükkel és a paramétereikkel hívhatók.

6. Kiírás:

set serveroutput on;

dbms_output.put_line();

7. Kurzor:

Cursor általános alakja:

```
DECLARE
    variables;
    records;
    create a cursor;
BEGIN
  OPEN cursor;
  FETCH cursor;
    process the records;
  CLOSE cursor;
END;
```

II. Gyakorló feladatok

1. Készítsük egy hello world nevű programot, ami egy változó segítségével kiírja azt, hogy hello world! (dbms_output.put_line();)
2. Bővítsük a programot, úgy hogy egy ciklus segítségével 10x írja ki azt!
3. Írassuk ki a páros számok négyzetét 1 és 15 között.
4. Iteráljunk végig a Signalink.interaction tábláján egy kurzor segítségével és írjuk ki az interakció id-jait és source attribútumáit, csak azokat az attribútumokat írjuk ki, ahol az **is_direct**=2.
5. Iteráljunk végig a Signalink interaction tábláján egy kurzor segítségével és írjuk ki az interakciók id-jait és source attribútumait, csak azokat az attribútumokat írjuk ki, ahol az **is_direct**=2. Használjuk *for változó in cursor loop… end loop;* konstrukciót
6. Hozzunk létre egy függvényt, mely kiszámolja egy r sugarú kör kerületét!
7. Hívjuk meg ezt a függvényt, a visszatérési értékét írassuk ki!
8. Készítsen egy eljárást, ami egy adott típusú (bemenő paraméter) súly minimum és maximum értékét kiírja. A szükséges adatokat az SL séma weigth_interaction nevű táblájában vannak.
9. Készítsünk egy eljárást, ami egy normalizálja a fehérjék közötti interakciós súlyokat 0 és 1 közé. A szükséges adatok az SL séma INTERACTION, WEIGHT_INTERACTION nevű tábláiban vannak. A weigth_id legyen egy bemenő paraméter. A normalizált súly: (weight – min_weight) / (max_weigth – min_weight). A kimenet alakja legyen a következő: interaction id normalizált súly eredeti súly