ADATBÁZIS ALAPÚ RENDSZEREK

PL/SQL folytatás

Kurzorok

Alprogramok

Tárolt eljárások

KURZOR

- Adattábla soronkénti feldolgozására szolgál
- A memóriában egy munkaterületen tárolódik a kurzorhoz tartozó tábla
- **Explicit kurzor**: a kurzorhoz tartozó tábla SELECT utasítással definiált
- **Implicit kurzor**: minden INSERT, DELETE, UPDATE és explicit kurzorral nem rendelkezõ SELECT utasításhoz automatikusan jön létre (hivatkozás: SQL)
- Kurzorfüggvények:
  - `SQL%FOUND`: a legutóbbi SQL utasítás legalább egy sort feldolgozott
  - `SQL%NOTFOUND`: a legutóbbi SQL utasítás nem dolgozott fel sort
  - `SQL%ROWCOUNT`: a kurzorral összesen feldolgozott sorok száma
  - `SQL%ISOPEN`: igaz, ha a kurzor meg van nyitva
  - Explicit kurzor esetén `kurzornev%...` alakúak

1. PÉLDA: IMPLICIT KURZOR

```
DECLARE
      v_sor DEMO.vevo%ROWTYPE;
BEGIN
      SELECT *
      INTO v_sor
      FROM DEMO.vevo
      WHERE partner_id = 21;
      DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT);
END;
```

EXPLICIT KURZOR HASZNÁLATA

- **Deklaráció:** `CURSOR kurzornév IS lekérdezés`
- **Megnyitás:** `OPEN kurzornév` **(megnyitáskor hajtódik végre a lekérdezés, a létrejövõ eredménytábla nem frissítõdik!!!)**
- **Léptetés:** `FETCH kurzornév INTO változók` **(az aktuális sor adatai a változókba kerülnek és a kurzor eggyel elõre lép, ellenõrizni kell, hogy a kurzorhoz tartozó eredménytáblának van-e sora!!!)**
- **Lezárás:** `CLOSE kurzornév`

2. PÉLDA: EXPLICIT KURZOR

```
DECLARE
        v_veznev DEMO.munkatars.vezeteknev%TYPE;
        v_kernev DEMO.munkatars.keresztnev%TYPE;
        v_tel DEMO.munkatars.telefon%TYPE;
        CURSOR nev_es_tel IS
                SELECT vezeteknev, keresztnev, telefon
                FROM DEMO.munkatars
                ORDER BY vezeteknev, keresztnev;
BEGIN
        OPEN nev_es_tel;
        LOOP
                FETCH nev_es_tel
                INTO v_veznev, v_kernev, v_tel;
                EXIT WHEN nev_es_tel%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE(v_veznev || ' ' ||
                        v_kernev || ': ' || v_tel);
        END LOOP;
        CLOSE nev_es_tel;
END;
```

3. PÉLDA: EXPLICIT KURZOR FOR CIKLUSBAN

```
DECLARE
      CURSOR nev_es_tel IS
            SELECT vezeteknev, keresztnev, telefon
            FROM DEMO.munkatars
            ORDER BY vezeteknev, keresztnev;
BEGIN
      /* a rekordnevet nem kell külön deklarálni */
      FOR m_rek IN nev_es_tel
      LOOP
            DBMS_OUTPUT.PUT_LINE(m_rek.vezeteknev
                  || ' ' || m_rek.keresztnev || ': ' ||
                  m_rek.telefon);
      END LOOP;
END;
```

PARAMÉTEREZETT KURZOR

```
CURSOR kurzornév (paraméternév
adattípus, ..., paraméternév
adattípus) IS alkérdés;
```

- Lehetőségünk van így a kurzort különböző paraméterekkel megnyitni (pl. más és más kódú dolgozó adatainak lekérése, stb.)

TÁBLA MÓDOSÍTÁSA KURZORRAL / ROWID

```
ACCEPT partner_azon PROMPT 'Kérem adja meg a
partner azonosítót: '
      DECLARE
            row_id ROWID;
            id DEMO.vevo.partner_id%TYPE;
      BEGIN
            SELECT ROWID INTO row_id
            FROM DEMO.vevo
            WHERE partner_id = '&partner_azon';
            UPDATE DEMO.vevo
            SET kiallt_szamlak_db =
                  kiallt_szamlak_db + 1
            WHERE ROWID = row_id;
            DBMS_OUTPUT.PUT_LINE(row_id);
      END;
```

TÁBLA MÓDOSÍTÁSA KURZORRAL / FOR UPDATE, CURRENT OF

```
DECLARE
        CURSOR partner_update(p_azon CHAR) IS
        SELECT kiallt_szamlak_db
        FROM DEMO.vevo
        WHERE partner_id = p_azon
        FOR UPDATE OF kiallt_szamlak_db NOWAIT;
        darab DEMO.vevo.kiallt_szamlak_db%TYPE;
BEGIN
        OPEN partner_update('&partner_azon');
        FETCH partner_update
              INTO darab;
        UPDATE DEMO.vevo
        SET kiallt_szamlak_db = darab + 1
        WHERE CURRENT OF partner_update;
        CLOSE partner_update;
END;
```

paraméteres kurzor

a pillanatnyilag érintett rekordot módosítja

NOWAIT: Ha egy másik tranzakció zárolta a kérédéses sort, akkor hibajelentéssel tovább fut

ALPROGRAMOK

- Névvel ellátott és paraméterezhetõ blokk (eljárás / függvény)
- Deklarációjuk a fõprogram DECLARE szegmensének végén
- Eljárások:

```
PROCEDURE név [(paraméterek)] IS
      [lokális deklarációk]
BEGIN
      utasítások
END név;
```

- paramétereik lehetnek IN, OUT, INOUT módúak (bemeneti változó, kimeneti változó, B/K változó)

```
egy_szam IN NUMBER,
egy_szam_negyzete OUT NUMBER
```

4. PÉLDA: EGY ELJÁRÁS

```
DECLARE
        v_megnev DEMO.vevo.megnevezes%TYPE;
        PROCEDURE nyomtat(szoveg IN VARCHAR2)
        IS
        BEGIN
                DBMS_OUTPUT.PUT_LINE(szoveg);
        END;
BEGIN
        SELECT megnevezes
        INTO v_megnev
        FROM DEMO.vevo
        WHERE partner_id = 22;
        nyomtat(v_megnev);
END;
```

eljáráshívás főprogramon belül

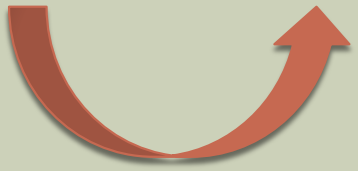
ALPROGRAMOK / FÜGGVÉNYEK

- azonos szabályok érvényesek, mint az eljárásoknál, de van visszatérési értékük

```
FUNCTION név [(paraméterek)]
RETURN adattípus IS
      [lokális deklarációk]
BEGIN
      utasítások
END név;
```

5. PÉLDA: EGY FÜGGVÉNY

```
DECLARE
  v_partnerid NUMBER;
  v_ber DEMO.munkatars.ber%TYPE;
  FUNCTION min_ber (ber IN NUMBER)
  RETURN BOOLEAN
  IS
    tmp_ber DEMO.munkatars.ber%TYPE;
  BEGIN
    SELECT MIN(ber)
    INTO tmp_ber
    FROM DEMO.munkatars;
    RETURN (tmp_ber = ber);
  END min_ber;
```



```
BEGIN
  SELECT MIN(partner_id)
  INTO v_partnerid
  FROM DEMO.munkatars;
  LOOP
    SELECT ber
    INTO v_ber
    FROM DEMO.munkatars
    WHERE partner_id =
       v_partnerid;
    v_partnerid := v_partnerid
+      1;
  EXIT WHEN min_ber(v_ber);
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('A
    minimal ber: ' || v_ber);
END;
```

ADATBÁZIS-OBJEKTUMKÉNT TÁROLT ALPROGRAM

- lehetőség van az adatbázisban eltárolni alprogramokat
- ezek később tetszőleges, az adatbázison futtatott PL/SQL blokkból, SQL lekérdezésből, az EXEC utasítással, vagy SQL*Plus környezetben az EXECUTE paranccsal hívhatók
- deklaráció elejére CREATE, IS helyett AS
- alapértelmezés IN típusu paraméterekhez: paraméternév típus DEFAULT érték, paramétert csak a lista végérõl lehet elhagyni

```
PROCEDURE nyomtat(szoveg IN VARCHAR2 DEFAULT 'empty')
FUNCTION min_ber (ber IN NUMBER DEFAULT 0) RETURN BOOLEAN
```

```
CREATE OR REPLACE PROCEDURE kiir AS
BEGIN
      DBMS_OUTPUT.PUT_LINE('Szoveg');
END kiir;
```

- Később: `exec kiir();`

FELADATOK / 1.

1. Adj meg egy **tárolt eljárást**, ami kiírja a képernyőre az EMP tábla sorainak számát és a dolgozók átlagfizetését. A megoldáshoz **ne használd az SQL összesítő függvényeit**, hanem **kurzorral** és ciklussal számold ki az értékeket. Adj példát az eljárás meghívására.

2. Adj meg egy **tárolt függvényt**, amely egy adott osztálykódra meghatározza, hogy az osztályon dolgozók összesen hány dollárt keresnek. **Kivételkezeléssel** biztosítsd, hogy ne lehessen érvénytelen osztálykódot megadni; ha ez történne, -1 legyen a visszaadott érték. Adj meg egy olyan SQL lekérdezést, ami megjeleníti az összes dolgozó kódját, nevét, fizetését, és a vele egy osztályon dolgozók bérösszegét!

FELADATOK / 2.

3. Adj meg egy **olyan PL/SQL programot**, ami FOR UPDATE OF … kurzorral a 2000 dollárnál kevesebbet kereső dolgozóknak az alapbérük 10%-át jutalomként adja! A jutalmat a COMM oszlophoz kell hozzáadni. Futás közben írjuk ki a jutalmat kapó dolgozók nevét a képernyőre, majd végül jelenjen meg a ‘Kész.’ üzenet!
   - Szükséges lehet az NVL() függvény használata.
   - A `SET AUTOCOMMIT ON` SQL*Plus utasítással automatikusan végrehajtódik a változtatások COMMIT-ja.

4. Adj meg egy olyan **PL/SQL programot**, ami **alprogram segítségével** ellenőrzi egy adott megnevezésű munkakör bérköltségét. Az alprogram egy **eljárás legyen**, és **OUT paraméterrel adja át a kiszámolt értéket.** A munkakört a program futása előtt **a felhasználótól kérd be.** Az alprogramban **kivételkezeléssel** ellenőrizd, hogy valódi-e a megadott munkakör; ha nem, -1 legyen a visszaadott bérösszeg.