Kurzorok, kurzorváltozók

Kurzorok

- Környezeti terület: memóriaterület
  - információ a kezelt sorokról
    - lekérdezés esetén *aktív halmaz*
  - mutató az utasítás belső reprezentációjára
  - stb.
- Kurzorral nevesítjük
  - explicit
  - implicit
  - (rejtett)

Explicit kurzor

- Életciklus:
  - deklaráció

    ```
    CURSOR név[(paraméter[,paraméter]…)] [RETURN sortípus] IS select_utasítás;
    ```

  - megnyitás

    ```
    OPEN kurzornév[(aktuális_paraméterlista)];
    ```

  - sorok betöltése

    ```
    FETCH {kurzornév|kurzorváltozónév}
      {INTO {rekordnév|változónév[,változónév]…} |
       BULK COLLECT INTO kollekciónév[,kollekciónév]…
         LIMIT sorok};
    ```

  - lezárás

    ```
    CLOSE {kurzornév|kurzorváltozónév};
    ```

```
 CURSOR cur_ugyfelek RETURN ugyfel%ROWTYPE IS
    SELECT * FROM ugyfel
      ORDER BY UPPER(nev);

v_Uid    ugyfel.id%TYPE;
CURSOR cur_ugyfel1 IS
    SELECT nev, tel_szam FROM ugyfel
      WHERE id = v_Uid;

CURSOR cur_ugyfel2(p_Uid ugyfel.id%TYPE DEFAULT v_Uid)
  IS
    SELECT * FROM ugyfel
      WHERE id = p_Uid;
```

```
DECLARE
CURSOR c2 IS
    SELECT * FROM employees
    WHERE job_id LIKE '%MGR' OR job_id LIKE '%MAN'
    ORDER BY job_id;
  v_employees employees%ROWTYPE;
BEGIN
  OPEN c2;
  LOOP
    FETCH c2 INTO v_employees;
    EXIT WHEN c2%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE( v_employees.last_name ||
                          v_employees.job_id );
  END LOOP;
  CLOSE c2;
END;
/
```

```
DECLARE
  CURSOR c IS SELECT e.job_id, j.job_title
    FROM employees e, jobs j
    WHERE e.job_id = j.job_id AND e.manager_id = 100
    ORDER BY last_name;
  job1 c%ROWTYPE;
  job2 c%ROWTYPE;
  job3 c%ROWTYPE;
BEGIN
  OPEN c;
  FETCH c INTO job1;  -- fetches first row
  FETCH c INTO job2;  -- fetches second row
  FETCH c INTO job3;  -- fetches third row
  CLOSE c;
DBMS_OUTPUT.PUT_LINE(job1.job_title||' '||job1.job_id);
DBMS_OUTPUT.PUT_LINE(job1.job_title||' '||job2.job_id);
DBMS_OUTPUT.PUT_LINE(job1.job_title||' '||job3.job_id);
END;
/
```

Kurzorattribútumok

- %FOUND
  - megnyitás után, de első betöltés előtt NULL,
  - sikeres betöltés esetén TRUE,
  - egyébként FALSE.
- %ISOPEN
  - ha meg van nyitva, TRUE,
  - egyébként FALSE.
- %NOTFOUND
  - %FOUND negáltja.
- %ROWCOUNT
  - megnyitás után, de első betöltés előtt 0,
  - minden sikeres betöltés esetén eggyel nő.

```
DECLARE
  CURSOR c1 IS
    SELECT last_name, salary FROM employees
    WHERE ROWNUM < 11;
  the_name employees.last_name%TYPE;
  the_salary employees.salary%TYPE;
BEGIN
  IF NOT c1%ISOPEN THEN OPEN c1; END IF;

  FETCH c1 INTO the_name, the_salary;

  IF c1%ISOPEN THEN
    CLOSE c1;
  END IF;
END;
/
```

```
DECLARE
  CURSOR c1 IS
    SELECT last_name, salary FROM employees
    WHERE ROWNUM < 11 ORDER BY last_name;
  my_ename   employees.last_name%TYPE;
  my_salary  employees.salary%TYPE;
BEGIN
 OPEN c1;
 LOOP
  FETCH c1 INTO my_ename, my_salary;
  IF c1%FOUND THEN  -- fetch succeeded
   DBMS_OUTPUT.PUT_LINE(my_ename||', '||my_salary);
   ELSE  EXIT; -- fetch failed
  END IF;
 END LOOP;
END;
/
```

```
DECLARE
  CURSOR c1 IS
    SELECT last_name FROM employees
    WHERE ROWNUM < 11 ORDER BY last_name;
  name  employees.last_name%TYPE;
BEGIN
 OPEN c1;
 LOOP
  FETCH c1 INTO name;
  EXIT WHEN c1%NOTFOUND OR c1%NOTFOUND IS NULL;
  DBMS_OUTPUT.PUT_LINE(c1%ROWCOUNT || '. ' || name);
  IF c1%ROWCOUNT = 5 THEN
     DBMS_OUTPUT.PUT_LINE('--- Fetched 5th row ---');
  END IF;
 END LOOP;
 CLOSE c1;
END;
```

Kivételek

- `CURSOR_ALREADY_OPEN`: megnyitott kurzor újra megnyitásakor
- `INVALID_CURSOR`: nem megnyitott kurzoron végzett `FETCH`/`CLOSE` vagy kurzorattribútum-hivatkozás (kivéve az `ISOPEN`-t) esetén

```
DECLARE
  CURSOR c IS SELECT * FROM HR.EMPLOYEES;
  v c%ROWTYPE;
BEGIN
  --FETCH c INTO v;
  OPEN c;
  --OPEN c;
  LOOP
    FETCH c INTO v;
    EXIT WHEN c%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Név: ' || v.FIRST_NAME ||
      v.LAST_NAME || '; Fizetés: ' || v.SALARY);
  END LOOP;
  CLOSE c;
  --CLOSE c;
EXCEPTION
  WHEN INVALID_CURSOR THEN
    DBMS_OUTPUT.PUT_LINE('Nem volt megnyitva!');
  WHEN CURSOR_ALREADY_OPEN THEN
    DBMS_OUTPUT.PUT_LINE('Már nyitva volt!');
END;

/
```

```
DECLARE
  CURSOR cur_ugyfel2(p_Uid ugyfel.id%TYPE
  DEFAULT v_Uid) IS
    SELECT * FROM ugyfel
      WHERE id = p_Uid;
  v_Ugyfel      ugyfel%ROWTYPE;
BEGIN
 v_Uid := 15;
 OPEN cur_ugyfel2(15);
 LOOP
    FETCH cur_ugyfel2 INTO v_Ugyfel;
    EXIT WHEN cur_ugyfel2%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE(v_Ugyfel.nev);
 END LOOP;
 CLOSE cur_ugyfel2;
END;
/
```

SELECT … FOR UPDATE

- A lekérdezés által érintett rekordok explicit zárolása egy későbbi UPDATE vagy DELETE miatt
  - biztosítjuk, hogy más ne módosíthasson az adatokon, amíg a tranzakciónk be nem fejeződik
- Ilyenkor az UPDATE ill. DELETE tartalmazhat egy WHERE CURRENT OF utasításrészt, amellyel a legutóbbi FETCH által betöltött sor módosítható, ill. törölhető

```
DECLARE
   CURSOR c1 IS SELECT empno, job, sal
                FROM emp FOR UPDATE;
   v_c1 c1%rowtype;
   c_job emp.job%type:='MANAGER';

BEGIN
   OPEN c1;
   LOOP
      FETCH c1 INTO v_c1;
      exit when c1%notfound;
      if v_c1.job=c_job and v_c1.sal>5000
         then UPDATE emp SET sal = new_sal
              WHERE CURRENT OF c1;
      end if;
   END LOOP;
   CLOSE c1;
END;
```

Implicit kurzor

- Minden DML utasításhoz, amelyhez nem tartozik explicit kurzor, felépül egy implicit
- Neve: `SQL`
- Attribútumai:
  - `%FOUND`: `TRUE`, ha az utasítás legalább egy sort érintett, különben `FALSE`;
  - `%ISOPEN`: mindig `FALSE`;
  - `%NOTFOUND`: a `FOUND` negáltja;
  - `%ROWCOUNT`: kezelt sorok darabszáma (kivéve `SELECT INTO`);
  - `%BULK_ROWCOUNT`: asszociatív tömb, `FORALL` használata esetén *i* indexű eleme megadja, hogy az erre lefuttatott DML utasítás hány sort kezelt;
  - `%BULK_EXCEPTIONS`: `<ERROR_INDEX, ERROR_CODE>` felépítésű rekordok asszociatív tömbje a `FORALL` működése közben bekövetkezett kivételek kezelésére.

```
BEGIN
UPDATE employee
    SET salary = 100
    WHERE id = '01';

IF SQL%ROWCOUNT = 0 THEN
    INSERT INTO employee (id, salary)
         VALUES ('99', 100);
    DBMS_OUTPUT.put_line('SQL%ROWCOUNT = 0');
END IF;
END;
/
```

Kurzor FOR ciklus

```
FOR ciklusváltozó IN
   {kurzornév[(aktuális_paraméterlista)] |
   (select_utasítás)} LOOP
     utasítás [,utasítás]…
END LOOP;
```

- Implicit módon *kurzornév*%ROWTYPE típusú lesz a ciklusváltozó
- Megnyitja a kurzort
- Betölti az aktív halmaz összes sorát
- Lezárja a kurzort

- Ha explicit kurzor helyett egy SELECT utasítást adunk meg, akkor egy *rejtett kurzor* jön létre

```
DECLARE
  CURSOR c(p HR.EMPLOYEES.FIRST_NAME%TYPE) IS
    SELECT department_name FROM HR.DEPARTMENTS
     WHERE department_id IN
          (SELECT department_id FROM HR.EMPLOYEES
            WHERE first_name=p);
BEGIN
  FOR nevek IN (SELECT DISTINCT first_name
                  FROM hr.employees) LOOP
    DBMS_OUTPUT.PUT_LINE(nevek.first_name || ':');
    FOR i IN c(nevek.first_name) LOOP
      DBMS_OUTPUT.PUT_LINE('  ' || i.department_name);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE;
  END LOOP;
END;

/
```

```
DECLARE
  CURSOR c1 (job VARCHAR2, max_wage NUMBER) IS
    SELECT * FROM employees
    WHERE job_id = job
    AND salary > max_wage;
BEGIN
 FOR person IN c1('ST_CLERK', 3000)
 LOOP
     -- process data record
   DBMS_OUTPUT.PUT_LINE (
    'Name = ' || person.last_name || ', salary = ' ||
    person.salary || ', Job Id = ' || person.job_id);
  END LOOP;
END;
/
```

```
BEGIN
 FOR item IN (
  SELECT first_name|| ' ' ||last_name AS full_name,
           salary * 10 AS dream_salary
    FROM employees
    WHERE ROWNUM <= 5
    ORDER BY dream_salary DESC, last_name ASC)
  LOOP
   DBMS_OUTPUT.PUT_LINE
    (item.full_name||' dreams of making `
     || item.dream_salary);
  END LOOP;
END;
/
```

Kurzorváltozók – 1

- Nem kell fordítási időben ismerni a kurzorhoz kapcsolt SELECT-et
- Referencia típusú változó, amely mindig a hivatkozott sor címét tartalmazza
- Létrehozása két lépésben történik
  - REF CURSOR típus létrehozása

```
TYPE név IS REF CURSOR [RETURN
  {{táblanév|kurzornév|kurzorváltozónév}%ROWTYPE |
  rekordnév%TYPE | rekordtípusnév |
  kurzorreferenciatípus_név}];
```

  - kurzorváltozó deklarálása

```
v_refcursor t_refcursor;
```

Kurzorváltozók – 2

- Kurzorreferencia típus lehet
  - erős (ha szerepel a `RETURN` rész): a fordító ellenőrzi a kapcsolt lekérdezés típuskompatibilitását
  - gyenge (különben): bármely lekérdezés hozzákapcsolható ilyen típusú kurzorváltozóhoz
    - előre definiált típus: `SYS_REFCURSOR`
- Életciklus ugyanaz, mint az explicit kurzoroknál, de
  - megnyitás OPEN-FOR utasítással
    - akárhányszor megnyitható lezárás nélkül!
    - OPEN *kurzorváltozó_név* FOR *select_utasítás*;
- Kurzorreferencia típus lehet alprogram formális paramétere is

```
DECLARE
  TYPE t_eros_dept IS REF CURSOR
    RETURN hr.departments%ROWTYPE;
  v_refc SYS_REFCURSOR;
  v_eros t_eros_dept;
  v_emp_rec hr.employees%ROWTYPE;
  v_datum DATE;
  v_felh_nev VARCHAR2(20);
BEGIN
  OPEN v_refc FOR SELECT * FROM hr.employees WHERE salary>15000;
--  fordítási hiba:
--  OPEN v_eros FOR SELECT * FROM hr.employees;
  LOOP
    FETCH v_refc INTO v_emp_rec;
    EXIT WHEN v_refc%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_refc.employee_id);
  END LOOP;
  OPEN v_refc FOR SELECT SYSDATE, USER FROM DUAL;
  FETCH v_refc INTO v_datum, v_felh_nev;
  DBMS_OUTPUT.PUT_LINE(v_felh_nev || ' ' || v_datum);
  CLOSE v_refc;
END;
/
```

```
CREATE FUNCTION f(cur SYS_REFCURSOR, mgr_hiredate
   DATE) RETURN NUMBER IS
emp_hiredate DATE; before number :=0; after number:=0;
begin
   loop
       fetch cur into emp_hiredate;
       exit when cur%NOTFOUND;
       if emp_hiredate > mgr_hiredate then
             after:=after+1;
       else
             before:=before+1;
       end if;
   end loop;
   close cur;
   if before > after then return 1;
   else return 0;
   end if;
end;
/
```

```
DECLARE
 TYPE t_egyed IS RECORD (id NUMBER,
                         leiras     VARCHAR2(100));
  TYPE t_ref_egyed IS REF CURSOR RETURN t_egyed;

  v_Refcursor       SYS_REFCURSOR;
  v_Egyedek1        t_ref_egyed;
  v_Egyedek2        t_ref_egyed;
  v_Egyed           t_egyed;

  PROCEDURE megnyit_konyv
      (p_cursor IN OUT SYS_REFCURSOR) IS
  BEGIN
    OPEN p_cursor FOR
      SELECT id, cim FROM konyv;
  END;

  …
```

```
…
FUNCTION megnyit_ugyfel RETURN t_ref_egyed IS
    rv        t_ref_egyed;
  BEGIN
    OPEN rv FOR SELECT id, nev FROM ugyfel;
    RETURN rv;
  END;

FUNCTION betolt(p_cursor IN t_ref_egyed)
    RETURN t_egyed IS
    rv        t_egyed;
  BEGIN
    FETCH p_cursor INTO rv;
    RETURN rv;
  END;
…
```

```
…
PROCEDURE bezar(p_cursor IN SYS_REFCURSOR) IS
  BEGIN
    IF p_cursor%ISOPEN THEN CLOSE p_cursor;
    END IF;
  END;

BEGIN
  megnyit_konyv(v_Egyedek1);
  v_Refcursor := megnyit_ugyfel;
  v_Egyedek2 := v_Refcursor;
  v_Egyed := betolt(v_Egyedek2);
  DBMS_OUTPUT.PUT_LINE(v_Egyed.id||', `
  ||v_Egyed.leiras);
  v_Egyed := betolt(v_Refcursor);
  DBMS_OUTPUT.PUT_LINE(v_Egyed.id || ', ' ||
  v_Egyed.leiras);
 …
```

```
…
  v_Refcursor := v_Egyedek1;
  v_Egyed := betolt(v_Egyedek1);
  DBMS_OUTPUT.PUT_LINE(v_Egyed.id || ', ' ||
  v_Egyed.leiras);
  v_Egyed := betolt(v_Refcursor);
  DBMS_OUTPUT.PUT_LINE(v_Egyed.id || ', ' ||
  v_Egyed.leiras);
  bezar(v_Egyedek1);
  bezar(v_Egyedek2);

  BEGIN
    v_Egyed := betolt(v_Refcursor);
    DBMS_OUTPUT.PUT_LINE(v_Egyed.id || ', ' ||
                         v_Egyed.leiras);
  EXCEPTION WHEN INVALID_CURSOR THEN
   DBMS_OUTPUT.PUT_LINE('Tényleg be volt zárva!');
  END;
…
```

```
…
OPEN v_Refcursor FOR SELECT 'alma',2,3,4 FROM DUAL;
v_Egyedek2 := v_Refcursor;

EXCEPTION
  WHEN OTHERS THEN
    CLOSE v_Refcursor;
    RAISE;
END;
/
```

CURSOR kifejezések

- Beágyazott kérdés sorait kezeljük kurzorral
- Az aktív halmaz soraiban értékek és kurzorok vannak
- A beágyazott kurzor implicit nyílik meg
  - a szülő kurzor sorának betöltésekor
- Lezáródik, ha
  - explicit módon bezárjuk
  - új sort tölt be vagy lezáródik a szülő
  - hiba történik a szülőbe betöltéskor (clean-up)

CURSOR kifejezésekre vonatkozó megszorítások

- Ha a tartalmazó utasítás SELECT, akkor beágyazott kurzorok megjelenhetnek a legkülső SELECT-listán, vagy egy másik beágyazott kurzor legkülső SELECT-listáján is
- Beágyazott kurzorok nem szerepelhetnek nézetekben

```
declare
  begin
  for dep in (select department_id, department_name
                      from hr.departments
                      where department_name in
                      ('Executive‘,'Marketing' )
                      order by department_name)
  loop
    Dbms_Output.Put_Line ( department.department_name );
    for employee in (select last_name
                      from hr.employees
                      where department_id = dep.department_id
                      order by last_name)
    loop
      Dbms_Output.Put_Line ( employee.last_name );
    end loop;
  end loop;
end;
/
```

```
declare
  cursor the_departments is
    select department_name, cursor (select last_name
                                    from hr.employees e
                                    where e.department_id = d.department_id
                                    order by last_name)
      from hr.departments d
      where department_name in ( 'Executive', 'Marketing' )
      order by department_name;

  v_department_name hr.departments.department_name%type;
  the_employees sys_refcursor;

  v_employee_last_name hr.employees.last_name%type;
begin
  open the_departments;
  loop
    fetch the_departments into v_department_name, the_employees;
    exit when the_departments%notfound;
    Dbms_Output.Put_Line ( v_department_name );

    loop
      fetch the_employees into v_employee_last_name;
      exit when the_employees%notfound;
      Dbms_Output.Put_Line ( v_employee_last_name );
    end loop;
  end loop;
  close the_departments;
end;
/
```

```
select a.table_name,
  cursor(select column_name
         from user_tab_columns b
         where b.table_name =
               a.table_name) col_list
from user_tables a
```

```
DECLARE
  sal           employees.salary%TYPE;
  sal_multiple  employees.salary%TYPE;
  factor        INTEGER := 2;
  CURSOR c1 IS SELECT salary, salary*factor
    FROM employees WHERE job_id LIKE 'AD_%';
BEGIN
 OPEN c1;  -- PL/SQL evaluates factor
 LOOP
  FETCH c1 INTO sal, sal_multiple;
  EXIT WHEN c1%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE('factor=' || factor);
  DBMS_OUTPUT.PUT_LINE('sal=' || sal);
  DBMS_OUTPUT.PUT_LINE('sal_multiple=' || sal_multiple);
  factor := factor + 1; -- Does not affect sal_multiple
  END LOOP;
  CLOSE c1;
END;
/
```

```
DECLARE
CURSOR c(job VARCHAR2,max_sal NUMBER) IS
SELECT last_name,first_name,(salary - max_sal) overpayment
 FROM employees WHERE job_id = job AND salary > max_sal
 ORDER BY salary;

PROCEDURE print_overpaid IS
 last_name_   employees.last_name%TYPE;
 first_name_  employees.first_name%TYPE;
 overpayment_ employees.salary%TYPE;
 BEGIN
 LOOP
  FETCH c INTO last_name_, first_name_, overpayment_;
  EXIT WHEN c%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE(last_name_ || ', ' || first_name_
  || ' (by ' || overpayment_ || ')');
 END LOOP;
 END print_overpaid;
…
```

```
…
BEGIN
  DBMS_OUTPUT.PUT_LINE('Overpaid Stock Clerks:');
  OPEN c('ST_CLERK', 5000);
  print_overpaid;
  CLOSE c;

  DBMS_OUTPUT.PUT_LINE('Overpaid Sales Representatives:');
  OPEN c('SA_REP', 10000);
  print_overpaid;
  CLOSE c;
END;
/
```