9. FEJEZET

Hivatkozási és összetett adattípusok, kurzor, ROWID

Elméleti összefoglaló

A hivatkozási (más néven relatív, vagy kapcsolt) adattípusok valamely korábban már deklarált változóra, vagy valamely adattábla valamely oszlopára, vagy valamely adattáblára (ezáltal annak összes oszlopára) hivatkoznak. A hivatkozott objektum jellegétől függően a hivatkozási adattípus lehet egyszerű vagy rekord típusú. (Például adattáblára való hivatkozás esetén nyilvánvalóan rekord típust kapunk.)

A hivatkozási adattípusok révén olyan feldolgozóprogramok írhatók, melyek az adattáblák szerkezetének kisebb változásaival szemben érzéketlenek, és amelyekben nagyobb változások (például numerikusról karakteresre való áttérés) esetén is jól kézbentarthatók a szükséges módosítások.

A PL/SQL (jelen példatárban használt) összetett adattípusai a rekord, a gyűjtőtábla (PL/SQL-tábla) és a kurzor. Az összetett típusok (bár lehetnek közvetlen típusmegadásúak is) általában hivatkozási típusok. Használatukkal igen tömör és hatékony programok írhatók. A kurzort (mely tulajdonképpen egy lekérdezés) felhasználhatjuk (egyéb utasítások használata mellett) lekérdezések egymást követő sorainak egyenkénti feldolgozására, sőt egy FOR-ciklus adathalmazaként is. Ennek a feladatkörnek talán legösszetettebb esetei az úgynevezett párosítási feladatok (*lásd* jelen fejezet: *Összetett feladatok*). Ezzel a témakörrel részletesen a [15] és [16] foglalkozik.

Hivatkozási adattípus

A hivatkozási típusú változók deklarálásának módja:

```
DECLARE
  változó1    táblanév.oszlopnév%TYPE;
  változó2    változó1%TYPE;
  változó3    táblanév%ROWTYPE;
```

ahol az elemi típusú *változó1* típusa azonos a *táblanév* nevű tábla *oszlopnév* nevű oszlopának típusával, a *változó2* típusa azonos a *változó1* típusával, és a rekord típusú *változó3* egyes me-

zőire minősített nevekkel lehet hivatkozni. (Ha például a *táblanév* nevű táblában volt egy *kor* nevű oszlop, akkor a *változó3* nevű változónak lesz egy *kor* nevű mezője, melyre *változó3.kor* néven hivatkozhatunk.)

Rekord adattípus

Rekord adattípus és ilyen típusú változó deklarálása:

```
DECLARE
  TYPE  rekordtípusnév IS RECORD
          (oszlopnév1    típus1
          [, oszlopnév2  típus2]...);
  változónév  {rekordtípusnév | tábla%ROWTYPE};
```

ahol:

- *oszlopnévn* a rekord egy mezőjének neve,
- *típusn* {*típusnév* | *változó*%TYPE | *tábla.oszlop*%TYPE}.

Gyűjtőtábla típus

A gyűjtőtábla (vagy más néven PL/SQL-tábla) típusú változók adat- vagy nézettáblák lekérdezett sorainak a memóriában való listaszerű tárolására alkalmasak.

```
DECLARE
  TYPE típusnév IS TABLE OF
     {oszloptípus | változó%TYPE | tábla.oszlop%TYPE | tábla%ROWTYPE}
     [NOT NULL]
     INDEX BY BINARY_INTEGER;
  változónév  típusnév;
```

ahol:

- *típusnév* a gyűjtőtábla típus neve,
- *oszloptípus* standard PL/SQL-adattípus, de lehet rekord is,
- *változó*%TYPE hivatkozás korábban deklarált változóra,
- *tábla.oszlop*%TYPE hivatkozás egy tábla oszlopának típusára,
- *tábla*%ROWTYPE hivatkozás egy táblára,
- NOT NULL megszorítás megadása,
- INDEX BY BINARY_INTEGER az utasításrész egyetlen funkciója a felhasználó emlékeztetése arra, hogy az INDEX egész típusú.

Kurzorok

Az eddigi eszközeinkkel nem volt lehetőségünk arra, hogy egy adattábla (nézettábla) megadott feltételeknek eleget tevő sorait egyenként feldolgozzuk. Az SQL UPDATE utasítása segítségével ugyanis a benne szereplő WHERE feltételnek megfelelő sorokon csak ugyanazt a módosítást lehet végrehajtani. Gyakran azonban arra van szükség, hogy a megadott feltételnek eleget tevő sorok feldolgozása különböző lehessen, mivel az egymást követő elemi feldolgozások között kölcsönhatás van (*lásd* például a párosítási feladatokat a jelen fejezet összetett feladatai között).

A PL/SQL már megismert SELECT utasítása ugyan már csak egyetlen sort ad eredményként (SELECT ... INTO ...), ám erről nekünk kellett gondoskodni kiválasztó feltételek (jellemzően a WHERE feltétel) megfelelő beállításával. Azt a célt tehát, hogy ugyanazt a feltételt kielégítő sorokat egyenként dolgozzuk fel, ez sem elégíti ki.

Az PL/SQL azonban rendelkezik egy igen sokoldalúan használható eszközzel is, a kurzorral, melynek segítségével lehetővé válik adattáblák (nézettáblák) soronkénti feldolgozása. Ennek során az Oracle egy munkaterületet hoz létre, egy memóriaterületet foglal le, ahol a feldolgozandó (a kurzorhoz tartozó SELECT utasítással lekérdezett) adatokat, vagyis e SELECT eredménytábláját átmenetileg tárolja. (Ez a SELECT utasítás azonban már egy hagyományos SQL SELECT, amely természetesen általában több sort ad vissza.) Ehhez a lefoglalt munkaterülethez (a kurzorterülethez, vagy más néven az aktív halmazhoz) azonosítót rendelhetünk (ez a kurzor neve), és a későbbiek során e névvel hivatkozhatunk rá. (Ezt nevezzük *explicit* kurzornak). A kurzor neve tehát egy szimbolikus mutató, amely ennek az átmeneti tárolóhelynek (eredménytáblának) a kezdetére mutat. Ha ezt a mutatót (egy ciklus segítségével) végigmozgatjuk az átmeneti eredménytábla minden egyes során, akkor e sorokat egyenként feldolgozhatjuk. Nagyon fontos, hogy egy kurzorterület megnyitása után a kurzorterületen lévő adatok nem frissülnek (egészen a következő megnyitásig), és azok csak feldolgozásra használhatók, azon keresztül az adattábla adatai nem változtathatók meg. Az Oracle-ben kétféle kurzor létezik:

- az implicit és
- az explicit kurzor.

A PL/SQL nyelv minden, PL/SQL blokkban kiadott DML (INSERT, DELETE, UPDATE) és explicit kurzorral nem rendelkező DQL- (SELECT) utasításhoz létrehoz egy úgynevezett *implicit kurzort*. Az ekkor keletkező átmeneti eredménytáblának (kurzorterületnek) nincs neve, azt az Oracle-rendszer automatikusan kezeli. A létrehozott implicit kurzorra SQL előnevű függvényekkel (úgynevezett kurzorattribútumokkal, *lásd* alább) hivatkozhatunk.

A programozó saját maga is deklarálhat kurzort, ezt *explicit kurzornak* nevezzük, és az OPEN megnyitó, a FETCH léptető, és a CLOSE lezáró utasításokkal vezérelhetjük (*lásd* alább).

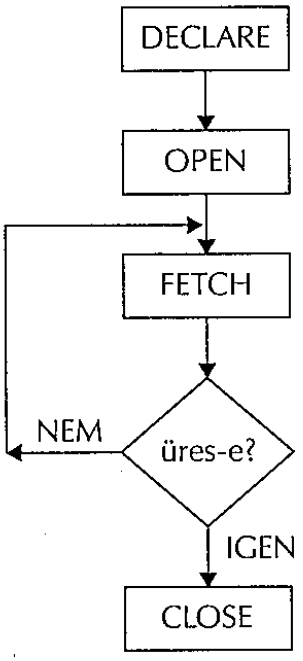
Explicit kurzor

A PL/SQL lehetővé teszi adattáblák soronkénti feldolgozását az úgynevezett kurzor adattípus segítségével. A kurzor használatakor (mint feljebb már említettük) az Oracle egy munkaterületet hoz létre, ahol a feldolgozandó (a kurzorhoz tartozó SELECT utasítással lekérdezett) adatokat átmenetileg tárolja. Az explicit kurzor deklarálásának módja:

```
DECLARE
  CURSOR kurzornév IS
    SELECT-utasítás;
```

ahol a *SELECT-utasítás* SQL szintaktikájú (így természetesen nem tartalmaz `INTO` utasításrészt!).

Explicit kurzor használata esetén először meg kell nyitni a kurzort, ezután történik a kurzor sorainak feldolgozása egyenként, majd a kurzorterületet be kell zárni. Ennek vázlatát mutatja be az alábbi ábra:



Létrehozunk egy elnevezett területet.

Meghatározzuk az aktív halmazt.

Betöltjük az aktuális sor tartalmát a változóba.

Ellenőrizzük, hogy van-e még feldolgozandó sor. Ha van, nem üres a kurzor területe, akkor visszatér a FETCH utasításhoz.

Ha üres az aktív halmaz, nincs több feldolgozandó sor, akkor felszabadítjuk a kurzort.

Kurzorhasználat `FOR`-ciklusban (rejtett kurzorok)

A kurzorok használatát jelentősen leegyszerűsítik a kurzort használó `FOR`-ciklusok. Lehetőség van ugyanis arra (miként ezt korábban már jeleztük), hogy a `FOR`-ciklus rendezett adathalmazaként egy kurzort adjunk meg. Ekkor a ciklusváltozó típusa (hivatkozási típusként) automatikusan felveszi a kurzor által meghatározott rekordszerkezetet, és értékeként minden ciklusban felveszi a kurzor egymást követő sorait. Ebben az esetben a kurzor megnyitása, sorainak betöltése, egyesével való léptetése, és az összes sor feldolgozása utáni bezárása *implicit* módon, tehát felhasználói `OPEN`, `FETCH` és `CLOSE` utasítás kiadása nélkül történik. Ezt a feldolgozási módszert az *explicit kurzor rejtett használatának* is nevezzük.

Tovább is egyszerűsíthetjük azonban a programot azzal, hogy egyáltalán nem deklarálunk kurzort, hanem a kurzorbeli lekérdezést közvetlenül adjuk meg a `FOR`-ciklus rendezett adathalmazaként. Ekkor *rejtett kurzorról* beszélhetünk. (A rejtett kurzor tulajdonképpen egy speciális implicit kurzor.)

A kurzorhasználat mindkét fenti esetben:

```
FOR ciklusváltozó IN kurzor
LOOP
  utasítás;
  [utasítás;]
  ...
END LOOP;
```

Ekkor a *kurzor* helyén állhat egy deklarált (explicit) kurzor neve, de állhat egy lekérdezés (azaz rejtett kurzor) is. Emlékeztetünk arra, hogy a *ciklusváltozót* nem kell külön deklarálni, mert az minden esetben implicit módon történik, és érvényességi köre kizárólag a ciklusra terjed ki.

Kurzorattribútumok

A kurzorattribútumok használata

Az Oracle (mint már jeleztük) minden PL/SQL-ben kiadott SELECT, INSERT, DELETE és UPDATE utasításhoz *implicit kurzort* rendel. Ezeket a név nélküli kurzorokat az Oracle automatikusan kezeli, tulajdonságaiknak lekérdezése az e műveletekhez rendelt memóriaváltozókkal, az SQL előtagú úgynevezett kurzorattribútumokkal

SQL%*Kurzorattribútum*

alakban történhet (például SQL%FOUND).

Explicit kurzorok esetén a fenti kurzorattribútumokat

KurzorNeve%Kurzorattribútum

módon használhatjuk. Segítségükkel ellenőrizhetjük, hogy mi történt a kurzor utolsó használata során. A kurzorattribútumok tipikus alkalmazása a kurzorciklusból való kilépés feltételének figyelése (az EXIT WHEN utasításrészben).

A PL/SQL kurzorattribútumai

- %FOUND visszaadott értéke TRUE, ha a legutóbbi SQL-utasítás legalább egy sort megvizsgált vagy feldolgozott,
- %NOTFOUND ha nincs több feldolgozandó sor, akkor a visszaadott értéke TRUE,
- %ROWCOUNT a megvizsgált, feldolgozott sorok számát adja vissza,
- %ISOPEN a kurzor megnyitását ellenőrzi; ha a kurzor nincs megnyitva, akkor értéke FALSE (implicit kurzor esetén mindig FALSE).

Megjegyzés

A kurzorattribútumok tulajdonképpen úgy viselkednek, mint a paraméter nélküli függvények, ezért rájuk a *kurzorfüggvény* kifejezés is használatos.

ROWID

A ROWID egy 18 karakteres pszeudooszlop, egy fizikai adattáblasor logikai címét tartalmazza. A logikai cím a tábla létrehozásakor jön létre, és adatbázisszinten egyedi. A SELECT utasítás segítségével lekérdezhető. Tipikus alkalmazása a tábla sorainak azonosítása például egy (rejtett) kurzort használó FOR-ciklus esetén.

FOR UPDATE záradék

Szintaktikája:

```
FOR UPDATE [OF oszlop] [NOWAIT]
```

és az SQL UPDATE utasításban

```
CURRENT OF kurzor
```

Ha a PL/SQL-ben sorok értékeit módosítani vagy törölni szeretnénk, akkor szükséges a kurzorban a FOR UPDATE záradékot használni. A FOR UPDATE azonosítja azokat a sorokat, amelyeket módosítani, illetve törölni fogunk, valamint zárolja az eredményhalmaz minden egyes sorát. Különösen akkor hasznos, ha a módosítás a sorok már létező értékein alapul. Biztosnak kell lennünk abban, hogy módosítás előtt más felhasználó nem változtatja meg ezekez az értékeket.

A NOWAIT opcionális kulcsszóval üzenünk a rendszernek, hogy ne várakozzon a zárolt sorokra, hanem végezzen el más feladatot addig, majd próbálkozzon újra.

Amikor a kurzort megnyitjuk, minden sor zárolt lesz, nem csak a FETCH által lehozott sor. A sorok zárolása megszűnik a visszagörgetés és a véglegesítés hatására. Ezért egy FOR UPDATE-tel ellátott kurzorból a COMMIT utasítás kiadása után nem lehet egy következő sort lehívni.

Amikor egy kurzorban többtáblás lekérdezés szerepel, a FOR UPDATE záradék a táblák nem mindegyikére vonatkozik. A zárolandó sorokat itt oszlopmegadással kell elérni, azaz a FOR UPDATE OF *oszlop* utasítással. Ez azonosítja az oszlopnéven keresztül a tábla zárolandó sorát. Tekintsük például az alábbi lekérdezést tartalmazó kurzort:

```
SELECT ename, sal, loc
  FROM emp, dept
  WHERE emp.deptno = dept.deptno
  FOR UPDATE OF sal;
```

Ez azt jelenti, hogy azt a sort (annak a táblának a sorait) zároljuk, amelyikben a sal oszlop szerepel, tehát az emp tábla zárolt lesz, míg a dept tábla nem.

Módosítás esetén a FETCH által utoljára lehívott, azaz aktuális sor azonosítása a

```
CURRENT OF kurzor
```

záradékkal történik, amely viszont csak a FOR UPDATE utasítással együtt használható.