Triggerek

A trigger olyan PL/SQL nyelven írt tárolt program, amelyet valamilyen esemény indít el (aktivizál, tüzel). A triggerek a hozzájuk rendelt esemény hatására automatikusan aktivizálódnak, a felhasználó azt befolyásolni nem tudja. (Emiatt a triggerek és az általuk hívott eljárások nem tartalmazhatnak `COMMIT`, `ROLLBACK` és `SAVEPOINT` utasításokat.) Egy trigger működése más triggereket is elindíthat. Az ilyen láncolt kapcsolatokat a triggerek tervezésekor figyelembe kell venni. (A triggerekről bővebben *lásd* [15].)

Triggerindító események

A triggert indító események lehetnek:

- a `DELETE`, `INSERT`, `UPDATE` DML-utasítások,
- a `CREATE`, `ALTER`, `DROP` DDL-utasítások,
- rendszeresemények, mint például az adatbázis indítása, leállítása és hibaüzenetek,
- felhasználói események, mint például egy felhasználó be- vagy kilépése.

A továbbiakban mi a DML-utasításokhoz (tehát törléshez, beszúráshoz és módosításhoz) kapcsolódó triggerekkel fogunk foglalkozni.

A triggerek alkalmazása

A triggerek alkalmazásának elsődleges célja az adatbázis integritásának (a benne tárolt adattáblák hitelességének és összhangjának) védelme, az adatbázisban történt változások feljegyzése. Ennek értelmében triggereket használhatunk:

- felhasználói előírások érvényesítésére (például többtáblás függőségek megvalósítására – *lásd* 4. melléklet),
- érvénytelen tranzakciók megelőzésére,
- származtatott (kiszámított) oszlopértékek automatikus generálására,
- adattáblák szinkronizálására (például elsődleges kulcs–idegen kulcs kapcsolathoz),
- események (törlés, beszúrás, módosítás) naplózására.

A triggerek felépítése és lekérdezése

A trigger felépítésének általános alakja:

```
A trigger feje
A trigger törzse
```

ahol a `trigger törzse` egy névtelen PL/SQL blokk, a `trigger feje` (description) pedig tartalmazza a trigger jellemzőit. Az adatbázisban tárolt triggerek fejrészei a

```
SELECT description
  FROM user_triggers;
```

utasítással, míg a trigger teljes forráskódja a fent már bemutatott módon a

```
SELECT line, text
  FROM user_source
  WHERE UPPER(type)='TRIGGER' AND
        UPPER(name)='TriggerNeve';
```

utasítással kérdezhetők le, ahol *TriggerNeve* a lekérdezendő trigger neve.

A tárolt alprogramok és triggerek lekérdezése

Mivel a tárolt alprogramok és triggerek (hasonlóan az adat- és nézettáblákhoz) olyan objektumok, melyeket az Oracle-rendszer tárol (és így az SQL*Plus-környezetből való kilépés után is megőrződnek), ezért célszerű időnként lekérdezni őket. Ez a

```
SELECT object_type, object_name
  FROM user_objects
  WHERE object_type IN ('PROCEDURE','FUNCTION','TRIGGER');
```

utasítással tehető meg.

A triggerek jellemzői

- A trigger időzítése (BEFORE, AFTER, INSTEAD OF).
- A trigger által figyelt esemény (DELETE, INSERT, UPDATE).
- A trigger típusa (utasításszintű, sorszintű).
- A trigger által felügyelt (a triggerhez rendelt) objektum (adattábla, nézettábla).

A triggerek szintaktikus alakja

```
CREATE [OR REPLACE] TRIGGER triggernév
  [BEFORE | AFTER | INSTEAD OF]
  triggerelő_esemény [OR triggerelő_esemény]...
  ON {adattábla | nézettábla}
[FOR EACH ROW
  [WHEN feltétel] ]
[DECLARE
  lokális változó deklarációja]
BEGIN
  utasítások
  [kivételkezelés]
END;
```

ahol:

- a REPLACE opciót akkor használjuk, ha a trigger már létezik,
- a *triggerelő_esemény* egy DML-utasításhoz kapcsolódik, szintaktikus alakja: {DELETE | INSERT | UPDATE [OF *oszlop*] },
- az OF kulcsszóval megadhatjuk, hogy melyik oszlop módosítására legyen érzékeny a trigger (ha ez hiányzik, akkor bármely oszlop módosítására érzékeny),
- a *trigger törzse* tartalmazhat kivételkezelést,
- a FOR EACH ROW opció teszi a triggert sorszintűvé,
- a sorszintű trigger minden olyan soron végrehajtódik, melyet a triggert kiváltó DML-utasítás érint,
- az INSTEAD OF triggerek (az Oracle 8-as verziójától léteznek) csak sorszintűek lehetnek, ezért esetükben a FOR EACH ROW opció el is hagyható,
- a BEFORE és az AFTER triggerek csak adattáblára, míg az INSTEAD OF triggerek csak nézettáblára definiálhatók (melyet ezért előzetesen létre kell hozni a módosítandó adattáblára vonatkozóan),
- az INSTEAD OF trigger mindig az őt elindító DML-művelet *helyett* aktivizálódik, ezért (ellentétben a BEFORE és AFTER triggerekkel) a triggerben elő kell írni a megfelelő DML-műveletet (e megoldás járulékos előnye, hogy egy esetlegesen előírt törlési művelet helyett végrehajthatunk például egy módosítást),
- az utasításszintű (tehát FOR EACH ROW opciót nem tartalmazó BEFORE és AFTER) triggerek az általuk felügyelt táblára vonatkozó triggerelő esemény hatására mindig aktivizálódnak (lefutnak), akkor is, ha az esemény egyetlen sort sem kezel,
- a sorszintű triggerekben, valamint az e triggerekből hívott tárolt alprogramokban hivatkozhatunk a módosított sorok eredeti és új értékeire az OLD és a NEW korrelációs nevek segítségével (*lásd* alább),
- a WHEN utáni feltétel a sorszintű triggerek működését az e feltétel által kijelölt sorokra korlátozza, ahol a feltételben az OLD és a NEW korrelációs nevekkel minősített adatokra kell hivatkozni.

A triggerek működésének jellegzetességei

- Ha egy triggerben, vagy egy benne meghívott tárolt alprogramban szerepel a RAISE_APPLICATION_ERROR eljárás (és ez meg is kapja a vezérlést), akkor a triggert kiváltó utasítás nem kerül végrehajtásra, illetve automatikusan visszagördül (mintha egy ROLLBACK utasítást adtunk volna ki, *lásd* még a 10.4. példa megjegyzését).
- A RAISE_APPLICATION_ERROR eljárás további hatása, hogy felfüggeszti a DBMS_OUTPUT.PUT_LINE kiíratő eljárás hatását mind a triggerben, mind a belőle hívott tárolt eljárásokban. A DBMS_OUTPUT.PUT_LINE eljárással történő kiíratások zárolását, vagyis e zárolt memóriaterületről a szöveg képernyőn való megjelenését csak egy névtelen blokkban történő kiíratás szabadítja fel. (*Lásd* még a *Tárolt alprogramok hibakezelése* c. pontot jelen fejezet korábbi részében, valamint a 10.1. feladat megoldásánál tett megjegyzést).
- Egy adattábla elsődleges- és idegenkulcs-megszorításokkal ellátott oszlopainak értékeit nem lehet triggerből megváltoztatni, illetve ilyen megszorításokat tartalmazó táblán csak

olyan sorok törölhetők triggerrel, melyeket nem védenek e megszorítások. (Például az emp táblában nem törölhetők a főnökök, *lásd* 10.5. példa.)

- Egy (DML-utasítás által) módosulásra kijelölt tábla BEFORE vagy AFTER triggerből nem kérdezhető le, és nem módosítható, kivéve, ha a triggerelő esemény egyetlen sor beszúrása. (Az INSTEAD OF triggerre ilyen korlátozás nincs.)

A korrelációs nevek használata

Gyakran előfordul, hogy egy sorszintű triggerben szükségünk volna a triggerelő esemény (utasítás) valamelyik adatparaméterére, illetve a triggerhez rendelt (a trigger által felügyelt) táblának a triggerelő esemény által kijelölt sora valamelyik adatmezőjére. Az ilyen igények kielégítésére szolgálnak az OLD és NEW korrelációs nevek, melyeket minősítő nevekként kell használni a triggerhez rendelt adattábla, vagy nézettábla oszlopaira például :OLD.ename, illetve :NEW.ename módon. A korrelációs nevek jelentése az egyes DML-utasításoknál az alábbi:

- DELETE műveletnél az utasításbeli adatra :OLD értékként kell hivatkozni, a :NEW értéke NULL,
- INSERT műveletnél az utasításbeli adatra :NEW értékként kell hivatkozni, az :OLD értéke NULL,
- UPDATE műveletnél az utasításbeli adatra :NEW értékként, a tárolt adatra pedig :OLD értékként kell hivatkozni.

Trigger törlése

A triggert a CREATE OR REPLACE előírás ellenére célszerű a létrehozása előtt törölni, mivel nem történik meg a régi változat felülírása, ha az például másik táblára vonatkozott.
A trigger törlése a

```
DROP TRIGGER TriggerNeve;
```

utasítással tehető meg.

Ügyeljünk arra, hogy az adattáblára definiált BEFORE és AFTER triggerek az adattáblájuk törlésekor *automatikusan* törlődnek. Ezzel ellentétben az INSTEAD OF triggerek a hozzájuk rendelt nézettábla, illetve az abban hivatkozott adattábla törlésekor *nem* törlődnek.

Azonos eseményre tüzelő triggerek

Az azonos eseményre tüzelő triggerek közül a legrégebbi aktivizálódik, ezért ügyelni kell arra, hogy az azonos fejrészű (description) triggerek közül a régebbieket töröljük. (Ehhez persze ismernünk kell, hogy milyen triggerek léteznek, amihez pedig le kell kérdezni a triggereket, *lásd* korábban *Triggerek felépítése és lekérdezése* és *A tárolt alprogramok és triggerek lekérdezése* c. pontok.)

A triggerek letiltása és engedélyezése

A már tárolt (létrehozott) triggereket letilthatjuk és engedélyezhetjük. Az engedélyezett trigger a hozzá rendelt esemény hatására aktivizálódik, a letiltott pedig nem. Létrehozásukkor a triggerek engedélyezett állapotban vannak, esetenként azonban szükség lehet (például bizonyos triggerekkel megvalósított megszorítások átmeneti felfüggesztése érdekében) a triggerek letiltására (*lásd* 10.5. példa).

Trigger letiltása, illetve engedélyezése

```
ALTER TRIGGER TriggerNeve [DISABLE | ENABLE];
```

Egy tábla összes triggerének letiltása, illetve engedélyezése

```
ALTER TRIGGER TriggerNeve [DISABLE | ENABLE] ALL TRIGGERS;
```

ahol a `DISABLE` jelenti a letiltást és az `ENABLE` az engedélyezést.

10.4. példa

Hozzunk létre egy olyan utasításszintű `BEFORE` triggert, amely megakadályozza a munkaidőn kívüli adatmanipulációkat az `emp` táblán.

Megoldás

```
CREATE OR REPLACE TRIGGER NeNyúljHozzá
BEFORE DELETE OR INSERT OR UPDATE ON emp
BEGIN
  IF TO_CHAR(sysdate, 'HH24:MI') NOT BETWEEN '08:00' AND '16:30'
  THEN
    RAISE_APPLICATION_ERROR(-20111,
                            'Csak munkaidőben szabad az adatbázis!');
  END IF;
END;
/
SHOW ERRORS
```

Eredmény

```
A trigger létrejött.
Nincsenek hibák.
```

Ellenőrzés

```
INSERT INTO emp
  VALUES (1234,'KISS','CLERK',1111,'99-MÁJ-20',1200,NULL,10);
```

Az ellenőrzés eredménye

```
INSERT INTO emp
            *
Hiba a(z) 1. sorban:
ORA-20111: Csak munkaidőben szabad az adatbázis!
ORA-06512: a(z) "SCOTT.NENYÚLJHOZZÁ", helyen a(z) 4. sornál
ORA-04088: hiba a(z) 'SCOTT.NENYÚLJHOZZÁ' trigger futása közben
```

A triggerelő esemény meghatározása

Ha egy trigger több műveletet is elindíthat, akkor a feldolgozásnál szükség lehet az indító művelet meghatározására. Erre a célra szolgálnak a trigger törzsében, vagy az onnan hívott tárolt alprogramban egyaránt használható `DELETING`, `INSERTING` és `UPDATING` függvények.

10.5. példa

Oldjuk meg az előző feladatot oly módon, hogy azt is kiíratjuk, milyen műveletet kíséreltek meg végrehajtani munkaidőn kívül.

Megoldás

```
CREATE OR REPLACE TRIGGER NeNyúljHozzá
BEFORE DELETE OR INSERT OR UPDATE ON emp
BEGIN
  IF TO_CHAR(sysdate, 'HH24:MI') NOT BETWEEN '08:00' AND '16:30'
  THEN
    IF DELETING THEN
      RAISE_APPLICATION_ERROR(-20211,
        'Csak munkaidőben szabad adatot törölni!');
    ELSIF INSERTING THEN
      RAISE_APPLICATION_ERROR(-20212,
        'Csak munkaidőben szabad adatot bevinni!');
    ELSE
      RAISE_APPLICATION_ERROR(-20213,
        'Csak munkaidőben szabad adatot módosítani!');
    END IF;
  END IF;
END;
/
SHOW ERRORS
```

Eredmény

```
A trigger létrejött.
Nincsenek hibák.
```

Ellenőrzés

```
DELETE FROM emp
  WHERE UPPER(ename) = UPPER('Smith');
```

Az ellenőrzés eredménye

```
DELETE FROM emp
            *
Hiba a(z) 1. sorban:
ORA-20211: Csak munkaidőben szabad adatot törölni!
ORA-06512: a(z) "SCOTT.NENYÚLJHOZZÁ", helyen a(z) 5. sornál
ORA-04088: hiba a(z) 'SCOTT.NENYÚLJHOZZÁ' trigger futása közben
```

10.6. példa

Fogalmazzuk át a 10.4. példát a következőképpen:
Hozzunk létre olyan triggereket, melyek segítségével megakadályozzuk (letiltjuk) a munkaidőn kívüli beavatkozásokat az `emp` táblán, továbbá egy `Feljegyzés` nevű adattáblába feljegyezzük a munkaidőn kívül megkísérelt, és a munkaidőben sikeresen elvégzett (engedélyezett) beavatkozások időpontját, jellegét ("Törlés", "Beszúrás", "Módosítás") és minősítését ("Letiltott", "Engedélyezett").

Megoldás

1. lépés (A `Feljegyzés` adattábla létrehozása)

```
DROP TABLE Feljegyzés;
CREATE TABLE Feljegyzés
  (DátumIdő     VARCHAR2(22),
   Beavatkozás  VARCHAR2(12),
   Jelleg       VARCHAR2(13));
```

Eredmény

```
DROP TABLE Feljegyzés
           *
Hiba a(z) 1. sorban:
ORA-00942: a tábla vagy a nézet nem létezik
A tábla létrejött.
```

Megjegyzés
A fenti "Hiba..." valójában nem hiba, egyszerűen csak azt jelzi, hogy a tábla még nem létezett.

2. lépés (Segéd-nézettábla létrehozása)

```
CREATE OR REPLACE VIEW DolgozóNézet
AS  SELECT * FROM emp;
```

Eredmény

```
A nézet létrejött.
```

3. lépés (A tiltott beavatkozást figyelő és feljegyző trigger törlése és létrehozása)

```
DROP TRIGGER NeNyúljHozzá;

CREATE OR REPLACE TRIGGER NeNyúljHozzá
INSTEAD OF DELETE OR INSERT OR UPDATE ON DolgozóNézet
DECLARE
  TörlésPróba     EXCEPTION;
  BeszúrásPróba   EXCEPTION;
  MódosításPróba  EXCEPTION;
  xempno          emp.empno%TYPE;
  xename          emp.ename%TYPE;
  xjob            emp.job%TYPE;
  xmgr            emp.mgr%TYPE;
  xhiredate       emp.hiredate%TYPE;
  xsal            emp.sal%TYPE;
  xcomm           emp.comm%TYPE;
  xdeptno         emp.deptno%TYPE;
BEGIN
  DBMS_OUTPUT.PUT_LINE('==> A NeNyúljHozzá trigger aktív...');
  IF TO_CHAR(sysdate, 'HH24:MI') NOT BETWEEN '08:00' AND '16:30'
  THEN
    IF DELETING THEN
      INSERT INTO Feljegyzés
        VALUES (TO_CHAR(sysdate,'YYYY.MM.DD, HH24:MI:SS'),
                'Törlés', 'Letiltott');
      RAISE TörlésPróba;
    ELSIF INSERTING THEN
      INSERT INTO Feljegyzés
        VALUES (TO_CHAR(sysdate,'YYYY.MM.DD, HH24:MI:SS'),
                'Beszúrás', 'Letiltott');
      RAISE BeszúrásPróba;
    ELSE
      INSERT INTO Feljegyzés
        VALUES (TO_CHAR(sysdate,'YYYY.MM.DD, HH24:MI:SS'),
                'Módosítás', 'Letiltott');
      RAISE MódosításPróba;
    END IF;
  ELSE
    IF DELETING THEN
      DELETE FROM emp
        WHERE UPPER(ename) = UPPER(:OLD.ename);  -- VIGYÁZAT: NEM :NEW!!!
```

```
    ELSIF INSERTING THEN
      INSERT INTO emp
        VALUES (:NEW.empno,    :NEW.ename, :NEW.job,  :NEW.mgr,
                :NEW.hiredate, :NEW.sal,   :NEW.comm, :NEW.deptno);
    ELSIF UPDATING THEN
      IF :NEW.empno    IS NULL
        THEN xempno    := :OLD.empno;
        ELSE xempno    := :NEW.empno;    END IF;
      IF :NEW.ename    IS NULL
        THEN xename    := :OLD.ename;
        ELSE xename    := :NEW.ename;    END IF;
      IF :NEW.job      IS NULL
        THEN xjob      := :OLD.job;
        ELSE xjob      := :NEW.job;      END IF;
      IF :NEW.mgr      IS NULL
        THEN xmgr      := :OLD.mgr;
        ELSE xmgr      := :NEW.mgr;      END IF;
      IF :NEW.hiredate IS NULL
        THEN xhiredate := :OLD.hiredate;
        ELSE xhiredate := :NEW.hiredate; END IF;
      IF :NEW.sal      IS NULL
        THEN xsal      := :OLD.sal;
        ELSE xsal      := :NEW.sal;      END IF;
      IF :NEW.comm     IS NULL
        THEN xcomm     := :OLD.comm;
        ELSE xcomm     := :NEW.comm;     END IF;
      IF :NEW.deptno   IS NULL
        THEN xdeptno   := :OLD.deptno;
        ELSE xdeptno   := :NEW.deptno;   END IF;
      UPDATE emp
        SET empno    = xempno,
--          ename    = xename,  -- Mivel névvel azonosítunk, ez nem aktív!
            job      = xjob,
            mgr      = xmgr,
            hiredate = xhiredate,
            sal      = xsal,
            comm     = xcomm,
            deptno   = xdeptno
        WHERE UPPER(ename) = UPPER(xename); -- Névvel azonosítunk!
    END IF;
  END IF;
EXCEPTION
  WHEN TörlésPróba THEN
    DBMS_OUTPUT.PUT_LINE('>> Csak munkaidőben szabad adatot törölni!');
```

```
  WHEN BeszúrásPróba THEN
    DBMS_OUTPUT.PUT_LINE('>> Csak munkaidőben szabad adatot bevinni!');
  WHEN MódosításPróba THEN
    DBMS_OUTPUT.PUT_LINE('>> Csak munkaidőben szabad adatot módosítani!');
END;
/
SHOW ERRORS
```

Eredmény

```
DROP TRIGGER NeNyúljHozzá
*
Hiba a(z) 1. sorban:
ORA-04080: 'NENYÚLJHOZZÁ' trigger nem létezik
A trigger létrejött.
Nincsenek hibák.
```

Megjegyzés

- A fenti „Hiba..." ezúttal sem hiba (*lásd* az előző megjegyzést).
- Mivel `INSTEAD OF` triggert használtunk, ezért gondoskodni kellett a triggert indító DML-utasítások helyettesítéséről a triggerben. Szintén a trigger `INSTEAD OF` jellegéből következik, hogy egyrészt magát a triggert, valamint az egyes DML-utasításokat is a munkatáblán értelmezett nézettáblára kell definiálni.
- Ezúttal a hibaüzenetet nem a `RAISE_APPLICATION_ERROR` függvénnyel, hanem a `DBMS_OUTPUT.PUT_LINE` függvénnyel írattuk ki. Ennek az az oka, hogy a `RAISE_APPLICATION_ERROR` függvény minden, a trigger tüzelését követő DML-műveletet visszagördít, és nem csupán azt, amelyikre a trigger érzékeny. Ennek az lett volna az eredménye, hogy a `Feljegyzés` táblába történő feljegyzéseket is kitörölte volna, márpedig e megoldás célja éppen a feljegyzések archiválása volt. Megjegyezzük, hogy látszólag a `DBMS_OUTPUT.PUT_LINE` függvény alkalmas volt a hibaüzenet kiíratására, ám ez nem akadályozza meg az `INSTEAD OF` trigger által kiváltott eredeti DML-utasítás üzenetének kiíratását (melyet viszont a `RAISE_APPLICATION_ERROR` függvény megakadályoz). Például a trigger által megakadályozott törlés esetén kiírta volna az „`1 sor törölve.`" üzenetet, amelyik persze hamis, hiszen a trigger helyes működése megakadályozta a törlést. Annak érdekében, hogy elkerüljük e hamis hibaüzenet kiírását, kiadtuk a `SET feedback OFF` utasítást.

4. lépés (Az engedélyezett beavatkozást utólag feljegyző trigger törlése és létrehozása)

```
DROP TRIGGER Feljegyez;

CREATE OR REPLACE TRIGGER Feljegyez
AFTER DELETE OR INSERT OR UPDATE ON emp
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('==> A Feljegyez trigger aktív...');
```

```
  IF DELETING THEN
    INSERT INTO Feljegyzés
      VALUES (TO_CHAR(sysdate,'YYYY.MM.DD, HH24:MI:SS'),
              'Törlés', 'Engedélyezett');
    DBMS_OUTPUT.PUT_LINE('>> Sikeres adattörlés!');
  ELSIF INSERTING THEN
    INSERT INTO Feljegyzés
      VALUES (TO_CHAR(sysdate,'YYYY.MM.DD, HH24:MI:SS'),
              'Beszúrás', 'Engedélyezett');
    DBMS_OUTPUT.PUT_LINE('>> Sikeres adatbeszúrás!');
  ELSE  -- (ekvivalens a ELSIF UPDATING THEN utasításrésszel)
    INSERT INTO Feljegyzés
      VALUES (TO_CHAR(sysdate,'YYYY.MM.DD, HH24:MI:SS'),
              'Módosítás', 'Engedélyezett');
    DBMS_OUTPUT.PUT_LINE('>> Sikeres adatmódosítás!');
  END IF;
END;
/
SHOW ERRORS
```

Eredmény

```
DROP TRIGGER Feljegyez
*
Hiba a(z) 1. sorban:
ORA-04080: 'FELJEGYEZ' trigger nem létezik
A trigger létrejött.
Nincsenek hibák.
```

5. lépés (Ellenőrzés)

```
--tesztelő program eleje
SET serveroutput ON
SET feedback OFF

PROMPT Törlési próba:
DELETE FROM DolgozóNézet
  WHERE UPPER(ename) = UPPER('Smith');

PROMPT Beszúrási próba:
INSERT INTO DolgozóNézet
  VALUES (1234,'KISS','CLERK',7499,'99-MÁJ-20',1200,NULL,10);

PROMPT Módosítási próba:
UPDATE DolgozóNézet
  SET sal = sal + 1000
  WHERE UPPER(ename) = UPPER('Blake');
```

```
SET feedback 5
PROMPT A beavatkozások listázása:
SELECT * FROM Feljegyzés;

PROMPT A módosított emp tábla listázása:
SET numwidth 5
SELECT * FROM emp;
SET numwidth 10
--tesztelő program vége
```

Az ellenőrzés eredménye (az engedélyezett időszakban)

```
Törlési próba:
==> A NeNyúljHozzá trigger aktív...
==> A Feljegyez trigger aktív...
>> Sikeres adattörlés!
Beszúrási próba:
==> A NeNyúljHozzá trigger aktív...
==> A Feljegyez trigger aktív...
>> Sikeres adatbeszúrás!
Módosítási próba:
==> A NeNyúljHozzá trigger aktív...
==> A Feljegyez trigger aktív...
>> Sikeres adatmódosítás!
```

Az ellenőrzés eredménye (a tiltott időszakban)

```
Törlési próba:
==> A NeNyúljHozzá trigger aktív...
>> Csak munkaidőben szabad adatot törölni!
Beszúrási próba:
==> A NeNyúljHozzá trigger aktív...
>> Csak munkaidőben szabad adatot bevinni!
Módosítási próba:
==> A NeNyúljHozzá trigger aktív...
>> Csak munkaidőben szabad adatot módosítani!

A beavatkozások listázása:
DÁTUMIDŐ              BEAVATKOZÁS  JELLEG
--------------------- ------------ -------------
2004.12.24. 14:05:07  Törlés       Engedélyezett
2004.12.24. 14:05:07  Beszúrás     Engedélyezett
2004.12.24. 14:05:08  Módosítás    Engedélyezett
2004.12.24. 18:09:51  Törlés       Letiltott
2004.12.24. 18:09:51  Beszúrás     Letiltott
2004.12.24. 18:09:51  Módosítás    Letiltott
```

```
6 sor kijelölve.

A módosított emp tábla listázása:
EMPNO ENAME      JOB          MGR HIREDATE    SAL  COMM DEPTNO
----- ---------- --------- ----- --------- ----- ----- ------
 7839 KING       PRESIDENT       81-NOV-17  5000            10
 7698 BLAKE      MANAGER    7839 81-MÁJ-01  3850            30
 7782 CLARK      MANAGER    7839 81-JÚN-09  2450            10
 7566 JONES      MANAGER    7839 81-ÁPR-02  2975            20
 7654 MARTIN     SALESMAN   7698 81-SZE-28  1250  1400      30
 7499 ALLEN      SALESMAN   7698 81-FEB-20  1600   300      30
 7844 TURNER     SALESMAN   7698 81-SZE-08  1500     0      30
 7900 JAMES      CLERK      7698 81-DEC-03   950            30
 7521 WARD       SALESMAN   7698 81-FEB-22  1250   500      30
 7902 FORD       ANALYST    7566 81-DEC-03  3000            20
 7788 SCOTT      ANALYST    7566 82-DEC-09  3000            20
 7876 ADAMS      CLERK      7788 83-JAN-12  1100            20
 7934 MILLER     CLERK      7782 82-JAN-23  1300            10
 1234 KISS       CLERK      7499 99-MÁJ-20  1200            10

14 sor kijelölve.
```

Megjegyzés

- Szegény `Smith` törölve...
- Figyeljünk fel arra, hogy a fenti `Feljegyez` nevű, az `emp` táblára definiált `AFTER` trigger korrekt módon működött a hozzá rendelt DML-utasításokra, bár az azokat tartalmazó tesztelőutasítások nem közvetlenül az `emp` táblán, hanem közvetett módon, a `DolgozóNézet` nevű nézettáblán keresztül végezték a beavatkozásokat. Tehát a `Feljegyez` triggert tulajdonképpen nem a tesztelőutasítások, hanem a megfelelő (az `emp` táblára ható) DML-utasításokat tartalmazó `NeNyúljHozzá` trigger indította el!
- A fentiek tesztelésénél ügyeljünk arra, hogy az `emp` tábla újragenerálásakor (törlésekor) a `Feljegyez` trigger törlődik, ezért azt szintén újra kell generálni. (Nem mondhatunk futtatást, hiszen az csak a triggerelő esemény hatására következik be.)
- Hasonlóképpen, ha a tesztelés során töröljük a `DolgozóNézet` nézettáblát, akkor az erre hivatkozó `NeNyúljHozzá` trigger törlődik.

6. lépés (Takarítás: adat-visszaállítás, létrehozott triggerek, adattáblák és nézetek törlése)

```
ROLLBACK;
DROP TRIGGER NeNyúljHozzá;
DROP TRIGGER Feljegyez;
DROP TABLE Feljegyzés;
DROP VIEW DolgozóNézet;
```

```
SET numwidth 5
SELECT * FROM emp;
SET numwidth 10
```

Eredmény

```
A visszaállítás befejeződött.
A trigger eldobva.
A trigger eldobva.
A tábla eldobva.
A nézet eldobva.

EMPNO ENAME      JOB         MGR HIREDATE    SAL  COMM DEPTNO
----- ---------- --------- ----- --------- ----- ----- ------
 7839 KING       PRESIDENT       81-NOV-17  5000            10
 7698 BLAKE      MANAGER    7839 81-MÁJ-01  2850            30
 7782 CLARK      MANAGER    7839 81-JÚN-09  2450            10
 7566 JONES      MANAGER    7839 81-ÁPR-02  2975            20
 7654 MARTIN     SALESMAN   7698 81-SZE-28  1250  1400      30
 7499 ALLEN      SALESMAN   7698 81-FEB-20  1600   300      30
 7844 TURNER     SALESMAN   7698 81-SZE-08  1500     0      30
 7900 JAMES      CLERK      7698 81-DEC-03   950            30
 7521 WARD       SALESMAN   7698 81-FEB-22  1250   500      30
 7902 FORD       ANALYST    7566 81-DEC-03  3000            20
 7369 SMITH      CLERK      7902 80-DEC-17   800            20
 7788 SCOTT      ANALYST    7566 82-DEC-09  3000            20
 7876 ADAMS      CLERK      7788 83-JAN-12  1100            20
 7934 MILLER     CLERK      7782 82-JAN-23  1300            10

14 sor kijelölve.
```

Megjegyzés

Fontos a fenti törlési sorrend betartása. Ha a `NeNyúljHozzá` trigger törlése előtt töröljük a `DolgozóNézet` nézettáblát, akkor az erre hivatkozó trigger automatikus törlődése miatt azt már nem tudjuk törölni. Ez még nem olyan nagy baj. Ennél súlyosabb az, ha az adat-visszaállítás (`ROLLBACK`) előtt töröljük azt a nézettáblát (`DolgozóNézet`), melyen keresztül a beavatkozások történtek. Ekkor tudniillik nem lehet az adat-visszaállítást elvégezni.

10.7. példa

Írjon triggert, mely megakadályozza az elnökre (`president`) vonatkozó törlő, beszúró és adatmódosító utasítások működését.

Megoldás (BEFORE-WHEN trigger alkalmazása)

```
CREATE OR REPLACE TRIGGER KivéveAzElnök
BEFORE DELETE OR INSERT OR UPDATE ON emp
FOR EACH ROW
WHEN (UPPER(OLD.job) = 'PRESIDENT'  OR
      UPPER(NEW.job) = 'PRESIDENT')
BEGIN
  IF DELETING THEN
    RAISE_APPLICATION_ERROR(-20001,'>> Az elnököt nem lehet törölni...');
  ELSIF INSERTING THEN
    RAISE_APPLICATION_ERROR(-20002,'>> Elnököt nem lehet beszúrni...');
  ELSIF UPDATING THEN
    RAISE_APPLICATION_ERROR(-20003,'>> Elnök adatok nem
módosíthatók...');
  END IF;
END;
/
SHOW ERRORS
```

Eredmény

```
A trigger létrejött.
Nincsenek hibák.
```

Ellenőrzés

```
-- A tesztelő szkript program eleje
SET serveroutput ON
PROMPT Hivatkozási megszorítás felfüggesztése:
ALTER TABLE emp
  DISABLE CONSTRAINT EMP_SELF_KEY;

UPDATE emp
  SET sal = sal - 555
  WHERE deptno = 10;

INSERT INTO emp
  VALUES (1234,'KISS','president',NULL,sysdate,6000,NULL,10);

DELETE FROM emp
  WHERE deptno = 10;

PROMPT Hivatkozási megszorítás engedélyezése:
ALTER TABLE emp
  ENABLE CONSTRAINT EMP_SELF_KEY;
SET numwidth 5
```

```
SELECT * FROM emp;
SET numwidth 10
-- A tesztelő szkript program vége
```

Eredmény

```
Hivatkozási megszorítás felfüggesztése:
A tábla módosítva.

  WHERE deptno = 10
                 *
Hiba a(z) 3. sorban:
ORA-20003: >> Elnök adatok nem módosíthatók...
ORA-06512: a(z) "SCOTT.KIVÉVEAZELNÖK", helyen a(z) 7. sornál
ORA-04088: hiba a(z) 'SCOTT.KIVÉVEAZELNÖK' trigger futása közben

  VALUES (1234,'KISS','president',NULL,sysdate,6000,NULL,10)
                                  *
Hiba a(z) 2. sorban:
ORA-20002: >> Elnököt nem lehet beszúrni...
ORA-06512: a(z) "SCOTT.KIVÉVEAZELNÖK", helyen a(z) 5. sornál
ORA-04088: hiba a(z) 'SCOTT.KIVÉVEAZELNÖK' trigger futása közben

  WHERE deptno = 10
                 *
Hiba a(z) 2. sorban:
ORA-20001: >> Az elnököt nem lehet törölni...
ORA-06512: a(z) "SCOTT.KIVÉVEAZELNÖK", helyen a(z) 3. sornál
ORA-04088: hiba a(z) 'SCOTT.KIVÉVEAZELNÖK' trigger futása közben

Hivatkozási megszorítás engedélyezése:
A tábla módosítva.

EMPNO ENAME      JOB         MGR HIREDATE    SAL  COMM DEPTNO
----- ---------- --------- ----- --------- ----- ----- ------
 7839 KING       PRESIDENT       81-NOV-17  5000            10
 7698 BLAKE      MANAGER    7839 81-MÁJ-01  2850            30
 7782 CLARK      MANAGER    7839 81-JÚN-09  2450            10
 7566 JONES      MANAGER    7839 81-ÁPR-02  2975            20
 7654 MARTIN     SALESMAN   7698 81-SZE-28  1250  1400      30
 7499 ALLEN      SALESMAN   7698 81-FEB-20  1600   300      30
 7844 TURNER     SALESMAN   7698 81-SZE-08  1500     0      30
 7900 JAMES      CLERK      7698 81-DEC-03   950            30
 7521 WARD       SALESMAN   7698 81-FEB-22  1250   500      30
 7902 FORD       ANALYST    7566 81-DEC-03  3000            20
```

```
7369 SMITH      CLERK      7902 80-DEC-17   800          20
7788 SCOTT      ANALYST    7566 82-DEC-09  3000          20
7876 ADAMS      CLERK      7788 83-JAN-12  1100          20
7934 MILLER     CLERK      7782 82-JAN-23  1300          10

14 sor kijelölve.
```

Megjegyzés

- Ahhoz, hogy a fenti törlési utasítás esetén a trigger hatását be tudjuk mutatni, előtte törölnünk kellett az `mgr` oszlop fölötti hivatkozási megszorítást a következő utasítással:

```
ALTER TABLE emp
  DISABLE CONSTRAINT EMP_SELF_KEY;
```

  A tesztelés után természetesen vissza kellett állítanunk e megszorítást az alábbi utasítással:

```
ALTER TABLE emp
  ENABLE CONSTRAINT EMP_SELF_KEY;
```

- Feltehető a kérdés, hogyan lehetne a `KivéveAzElnök` trigger tárolása esetén mégis csak módosítani az elnök adatait. Ez a trigger működésének felfüggesztésével végezhető el, az

```
ALTER TRIGGER KivéveAzElnök DISABLE;
```

  utasítással azok révén, akiknek erre jogosultságuk van, majd a megfelelő beavatkozás után az

```
ALTER TRIGGER KivéveAzElnök ENABLE;
```

  utasítással újra zárolhatók az elnök adatai (*lásd Triggerek letiltása és engedélyezése* c. pont).

Feladatok

10.1. feladat

Írjon triggert (és ellenőrizze is működését), amely az alábbi feltételek teljesülése esetén engedélyezi az adatbevitelt, míg e feltételek nem teljesülése esetén az adatbevitelt megtagadja. Minden feltétel ellenőrzését külön alprogrammal valósítsa meg. A trigger tartalmazzon kivételkezelést, és a hibaüzenetek is ebben szerepeljenek. A feltételek az alábbiak:

- egy dolgozóhoz csak már tárolt dolgozó azonosítója adható meg főnök-kódként,
- csak olyan munkakör adható meg, amely már szerepel,
- új dolgozó az azonos munkakörű társainál csak alacsonyabb fizetést kaphat.

10.2. feladat

Írjon triggert (és ellenőrizze is működését), amely az alábbi feltételek teljesülése esetén engedélyezi az adatbevitelt, míg e feltételek nem teljesülése esetén az adatbevitelt megtagadja. Min-

den feltétel ellenőrzését külön alprogrammal valósítsa meg, és ezek tartalmazzák a kivételkezelést, valamint a hibaüzenet kiírását is. A feltételek az alábbiak:

- egy dolgozóhoz csak már tárolt dolgozó azonosítója adható meg főnök-kódként,
- csak olyan részlegazonosító adható meg, amely már szerepel,
- új dolgozó az azonos részlegbeli társainál csak alacsonyabb fizetést kaphat.

10.3. feladat

Írjon triggert (és ellenőrizze is működését), amely megakadályozza, hogy a felhasználó olyan dolgozót

- töröljön,
- beszúrjon, vagy
- lásson el fizetésemeléssel,

akinek havi bére eléri vagy meghaladja részlegének átlagfizetését.

10.4. feladat

Írjon triggert (és ellenőrizze is működését), amely az alábbi feltételek teljesülése esetén engedélyezi az adatbevitelt, míg e feltételek nem teljesülése esetén az adatbevitelt megtagadja. Minden feltétel ellenőrzését külön alprogrammal valósítsa meg. A trigger tartalmazzon kivételkezelést, és a hibaüzenet ebben szerepeljen. A fizetésemelésre vonatkozó feltételek az alábbiak:

- az elnök (`president`) soha nem kaphat fizetésemelést,
- akinek fizetése eléri a 3000 USD-t, az csak a kijelölt fizetésemelés felét kapja,
- akinek egynél több közvetlen beosztottja van, az a kijelölt fizetésemelés kétszeresét kapja, feltéve, hogy fizetése nem éri el a 3000 USD-t.

10.5. feladat

Készítsen triggert (és ellenőrizze is működését), amely *megakadályozza* olyan új dolgozó felvételét, akinek

- *nincs* kijelölt főnöke (`mgr`), vagy a kijelölt főnöke *még nem* főnök,
- a kijelölt főnöke *már* legalább két közvetlen beosztottal rendelkezik,
- akár a kijelölt munkaköre, akár a kijelölt részlege *még nem* létezik,
- a megígért fizetése a leendő részlege átlagfizetésénél *nagyobb*,
- a megígért fizetése a leendő munkaköre legkisebb fizetésénél *kisebb*.

A fenti feltételek mindegyikét különálló tárolt alprogrammal ellenőrizze.