

Triggerek

Triggerek

- Olyan névvel ellátott adatbázisobjektumok, amelyek eseményorientált feldolgozást tesznek lehetővé
 - egy eseményre „válaszul” futnak le
 - a triggernevek külön névteret alkotnak
- Esemény bekövetkezésének helye szerint
 - alkalmazástriggerek
 - adatbázistriggerek

Triggerek osztályozása

- Kiváltó esemény alapján
 - rendszeresemény (adatbázis indul/leáll, szerverhiba):
STARTUP, SHUTDOWN, SERVERERROR
 - felhasználói esemény
 - be-/kijelentkezés: LOGON, LOGOFF
 - DDL utasítás: CREATE, ALTER, DROP
 - DML utasítás: DELETE, INSERT, UPDATE
 - sor szintű
 - utasítás szintű
- Összetettség
 - egyszerű
 - összetett
- Időzítés
 - BEFORE
 - AFTER
 - INSTEAD OF

Egyszerű trigger

- Pontosán egy időzítési ponton aktivizálódhat
 - a trigger kiváltó utasítás végrehajtása előtt
 - a trigger kiváltó utasítás végrehajtása után
 - a trigger kiváltó utasítás által érintett minden sor előtt
 - a trigger kiváltó utasítás által érintett minden sor után
- A kiváltó utasítás csak DML lehet

Egyszerű DML triggererek létrehozása

```
CREATE [OR REPLACE] TRIGGER név
  {BEFORE|AFTER|INSTEAD OF}
  {INSERT|DELETE|UPDATE [OF oszlop[,oszlop]...}
  [OR {INSERT|DELETE|UPDATE [OF oszlop[,oszlop]...}]...
ON {táblanév|[NESTED TABLE bát_oszlop OF] nézet}
[REFERENCING {OLD [AS] régi | NEW [AS] új |
  PARENT [AS] szülő}
[{OLD [AS] régi | NEW [AS] új | PARENT [AS]
  szülő}]...]
[FOR EACH ROW]
[{FOLLOWS|PRECEDES} [séma.]név[, [séma.]név]...
{ENABLE | DISABLE}
[WHEN (condition)]
{plsql_blokk|eljáráshívás}
```

```
CREATE OR REPLACE TRIGGER tr_insert_kolcsonzes
  AFTER INSERT ON kolcsonzes
  FOR EACH ROW
DECLARE
  v_Ugyfel    ugyfel%ROWTYPE;
BEGIN
  SELECT * INTO v_Ugyfel
    FROM ugyfel WHERE id = :NEW.kolcsonzo;
  IF v_Ugyfel.max_konyv = 8 THEN
    RAISE_APPLICATION_ERROR(-20010,
      v_Ugyfel.nev || ' ügyfél nem kölcsönözhet több
      könyvet. ');
  END IF;

END tr_insert_kolcsonzes;
/
show errors
```

```
CREATE OR REPLACE TRIGGER tr_kolcsonzes_naploz
  AFTER DELETE ON kolcsonzes
  REFERENCING OLD AS kolcsonzes
  FOR EACH ROW
DECLARE
  v_Konyv      konyv%ROWTYPE;
  v_Ugyfel     ugyfel%ROWTYPE;
BEGIN
  SELECT * INTO v_Ugyfel
    FROM ugyfel WHERE id = :kolcsonzes.kolcsonzo;
  SELECT * INTO v_Konyv
    FROM konyv WHERE id = :kolcsonzes.konyv;
  INSERT INTO kolcsonzes_naplo VALUES (
    v_Konyv.ISBN, v_Konyv.cim, v_Ugyfel.nev,
    v_Ugyfel.anyja_neve, :kolcsonzes.datum,
    SYSDATE, :kolcsonzes.megjegyzes);
END tr_kolcsonzes_naploz;
/
show errors
```

```
CREATE OR REPLACE TRIGGER tr_kolcsonzes_hosszabbit
  BEFORE INSERT OR UPDATE ON kolcsonzes
  FOR EACH ROW
  WHEN (NEW.hosszabbitva > 2 OR NEW.hosszabbitva < 0)
BEGIN
  RAISE_APPLICATION_ERROR(-20005, 'Nem megengedett a
    hosszabbítások száma');
END tr_kolcsonzes_hosszabbit;
/
show errors

UPDATE kolcsonzes SET hosszabbitva = 10;
```



```
CREATE OR REPLACE TRIGGER
  tr_kolcsonzes_naplo_del_upd
  BEFORE DELETE OR UPDATE ON kolcsonzes_naplo
BEGIN
  RAISE_APPLICATION_ERROR(-20100,
    'Nem megengedett művelet a kolcsonzes_naplo
    táblán');
END tr_kolcsonzes_naplo_del_upd;
```

```
CREATE TABLE szam_tabla(a NUMBER);

CREATE OR REPLACE TRIGGER tr_duplaz
  BEFORE INSERT ON szam_tabla
  FOR EACH ROW
BEGIN
  :NEW.a := :NEW.a * 2;
END tr_duplaz;
/

INSERT INTO szam_tabla VALUES(5);

SELECT * FROM szam_tabla;
```

```
CREATE OR REPLACE TRIGGER tr_utasitas_before
    BEFORE INSERT OR UPDATE OR DELETE ON tabla
BEGIN
    tabla_insert('UTASITAS BEFORE '
        || CASE
            WHEN INSERTING THEN 'INSERT'
            WHEN UPDATING THEN 'UPDATE'
            WHEN DELETING THEN 'DELETE'
        END);
END tr_utasitas_before;
/
```

Összetett trigger (11g óta)

- Egynél több időzítési ponton aktivizálódhat
- A kiváltó utasítás csak DML lehet
- Részei:
 - egy opcionális *közös* deklarációs rész
 - az egyes időzítési pontokhoz tartozó szakaszok (legalább egy)
 - egyazon időzítési ponthoz nem tartozhat több szakasz
 - ezeknek is van deklarációs része (privát)
- Ezek a szakaszok egy közös PL/SQL állapotban osztoznak

```
CREATE OR REPLACE TRIGGER osszetett_trigger
FOR UPDATE OF salary ON employees
COMPOUND TRIGGER

-- Deklarációs rész (opcionális)
kuszob CONSTANT SIMPLE_INTEGER := 200;

BEFORE STATEMENT IS
BEGIN
    NULL;
END BEFORE STATEMENT;

BEFORE EACH ROW IS
BEGIN
    NULL;
END BEFORE EACH ROW;

AFTER EACH ROW IS
BEGIN
    NULL;
END AFTER EACH ROW;

AFTER STATEMENT IS
BEGIN
    NULL;
END AFTER STATEMENT;
END osszetett_trigger;
/
```

- A deklarációs rész
 - változódeklarációkat és alprogramdefiníciókat tartalmazhat
 - ezek élettartama a trigger kiváltó utasítás!
- Az időzítési pontokhoz tartozó szakaszok
 - nézet esetében csak INSTEAD OF EACH ROW lehet
 - tábla esetén kötött sorrendben (bármelyik elmaradhat)
 - BEFORE STATEMENT
 - BEFORE EACH ROW
 - AFTER EACH ROW
 - AFTER STATEMENT
 - tartalmazhatnak INSERTING, DELETING, UPDATING függvényeket

Összetett trigger (11g óta)

- Hátrányok:
 - nem tartalmazhat WHEN részt
 - nem definiálhat autonóm tranzakciót
- Előnyök:
 - együttes hozzárendeléssel végezhető beszúrás
 - a változás alatt álló táblák problémájának elkerülése

Triggerek tárolása

- P-kód
- **USER_TRIGGERS (ALL, DBA)**
adatszótárnézetek
- `ALTER TRIGGER triggernév`
`{ENABLE | DISABLE | RENAME TO új_név |`
`COMPILE [DEBUG] [REUSE SETTINGS]};`
- `DROP TRIGGER triggernév;`

Triggerekre vonatkozó megszorítások

- Maximális triggerméret
 - a törzs max. 32760 bájt lehet
 - kerülőút: tárolt eljárások vagy csomagok használata
- SQL-utasítások
 - nem tartalmazhatnak tranzakcióvezérlő utasítást (COMMIT, ROLLBACK, SAVEPOINT)
 - kivéve az autonóm tranzakciókat
 - ez a triggerből hívott programegységekre is igaz!
 - a nem-rendszertriggerek nem tartalmazhatnak DDL-utasítást

Triggerek működése

- Triggerek állapotai
 - engedélyezett
 - letiltott
- 1. Lefuttatja a triggeret (ha több azonos típusú hallgat az adott eseményre, ezek sorrendje határozatlan)
- 2. Ellenőrzi az integritási megszorításokat
- 3. Olvasási konzisztenciát biztosít
- 4. Kezeli a trigger és a sémaobjektumok közötti függőségeket
- 5. Osztott adatbázis esetén, ha kell, kétfázisú véglegesítést végez

DML triggerek működése

1. Utasítás szintű BEFORE triggerek
2. A DML utasítás által érintett minden sorra
 - a) sorszintű BEFORE triggerek
 - b) zárolja és megváltoztatja a sort, ellenőrzi az integritási megszorításokat (a zárok a tranzakció végéig fent maradnak)
 - c) sorszintű AFTER triggerek
3. Ellenőrzi a késleltetett integritási megszorításokat
4. Utasítás szintű AFTER triggerek

Le nem kezelt kivétel esetén az SQL utasítás és a trigger hatása is visszagörgetésre kerül!

Megszorítások

- A trigger törzsében nem lehetnek tranzakcióvezérlő utasítások
 - lehet viszont autonóm tranzakció!
- *Módosítás alatt álló tábla*, amit egy DML művelet éppen változtat (+ DELETE CASCADE)
 - azok, amiket megszorítás ellenőrzéséhez olvasni kell, *megszorítással kapcsolt táblák*
- A trigger törzsében lévő SQL utasítás nem olvashatja az aktivizáló utasítás által módosítás alatt álló táblákat!!!

```
DROP TABLE log;  
CREATE TABLE log ( emp_id NUMBER(6), l_name VARCHAR2(25),  
f_name VARCHAR2(20) );
```

```
CREATE OR REPLACE TRIGGER log_deletions  
AFTER DELETE ON employees  
FOR EACH ROW  
DECLARE  
    n INTEGER;  
BEGIN  
INSERT INTO log  
VALUES ( :OLD.employee_id, :OLD.last_name, :OLD.first_name );  
SELECT COUNT(*) INTO n FROM employees;  
DBMS_OUTPUT.PUT_LINE('There are now ' || n || ' employees.');
```

```
END;  
/
```

```
DELETE FROM employees WHERE employee_id = 197;
```

Eredménye:

```
ERROR at line 1: ORA-04091: table HR.EMPLOYEES is mutating,  
trigger/function might not see it
```

```
ORA-06512: at "HR.LOG_DELETIONS", line 10
```

```
ORA-04088: error during execution of trigger 'HR.LOG_DELETIONS'
```

```
CREATE TABLE test (testcol VARCHAR2(15));
INSERT INTO test VALUES ('dummy');
CREATE OR REPLACE TRIGGER follows_a
  AFTER UPDATE ON test FOR EACH ROW
  BEGIN dbms_output.put_line('A');
  END follows_a;
/
CREATE OR REPLACE TRIGGER follows_b
  AFTER UPDATE ON test FOR EACH ROW
  BEGIN dbms_output.put_line('B');
  END follows_b;
/
set serveroutput on
UPDATE test SET testcol = 'a';
CREATE OR REPLACE TRIGGER follows_b
  AFTER UPDATE ON test FOR EACH ROW
  FOLLOWS follows_a
  BEGIN dbms_output.put_line('B');
  END follows_b;
/
UPDATE test SET testcol = 'a';
```

```
CREATE TABLE Company_holidays (Day DATE);
```

```
CREATE OR REPLACE TRIGGER Emp_permit_changes  
BEFORE INSERT OR DELETE OR UPDATE ON Emp
```

```
DECLARE
```

```
    Dummy INTEGER;
```

```
    Not_on_weekends EXCEPTION;
```

```
    Not_on_holidays EXCEPTION;
```

```
    Non_working_hours EXCEPTION;
```

```
BEGIN
```

```
/* Check for weekends: */
```

```
IF (TO_CHAR(Sysdate, 'DY') = 'SAT' OR
```

```
    TO_CHAR(Sysdate, 'DY') = 'SUN')
```

```
    THEN RAISE Not_on_weekends;
```

```
END IF;
```

```
...
```



```
/* Check for company holidays: */
SELECT COUNT(*) INTO Dummy
FROM Company_holidays WHERE TRUNC(Day) = TRUNC(Sysdate);
-- Discard time parts of dates
IF dummy > 0 THEN RAISE Not_on_holidays; END IF;
/* Check for work hours (8am to 6pm): */
IF (TO_CHAR(Sysdate, 'HH24') < 8 OR TO_CHAR(Sysdate, 'HH24') > 18)
THEN RAISE Non_working_hours; END IF;
EXCEPTION
WHEN Not_on_weekends
THEN Raise_application_error(-20324,'Might not change ' ||'employee table
during the weekend');
WHEN Not_on_holidays
THEN Raise_application_error(-20325,'Might not change ' ||'employee table
during a holiday');
WHEN Non_working_hours
THEN Raise_application_error(-20326,'Might not change ' ||'emp table during
nonworking hours');
END;
```

```
CREATE OR REPLACE TRIGGER Derived
BEFORE INSERT OR UPDATE OF Ename ON Emp
/* Before updating the ENAME field, derive the values for the
   UPPERNAME and SOUNDEXNAME fields. Restrict users
   from updating these fields directly: */
FOR EACH ROW
BEGIN
:NEW.Uppername := UPPER(:NEW.Ename);
:NEW.Soundexname := SOUNDEX(:NEW.Ename);
END;
/
```

CREATE OR REPLACE TRIGGER

Check_Employee_Salary_Raise

FOR UPDATE OF Salary ON Employees

COMPOUND TRIGGER

Ten_Percent CONSTANT NUMBER := 0.1;

TYPE Salaries_t IS TABLE OF Employees.Salary%TYPE;

Avg_Salaries Salaries_t;

TYPE Department_IDs_t IS TABLE OF
Employees.Department_ID%TYPE;

Department_IDs Department_IDs_t;

TYPE Department_Salaries_t IS TABLE OF
Employees.Salary%TYPE INDEX BY VARCHAR2(80);

Department_Avg_Salaries Department_Salaries_t;

...

...

BEFORE STATEMENT IS

BEGIN

SELECT AVG(e.Salary), NVL(e.Department_ID, -1)

BULK COLLECT INTO Avg_Salaries, Department_IDs

FROM Employees e

GROUP BY e.Department_ID;

FOR j IN 1..Department_IDs.COUNT()

LOOP

Department_Avg_Salaries(Department_IDs(j)) := Avg_Salaries(j);

END LOOP;

END BEFORE STATEMENT;

AFTER EACH ROW IS

BEGIN

IF :NEW.Salary - :Old.Salary >

Ten_Percent*Department_Avg_Salaries(:NEW.Department_ID)

THEN Raise_Application_Error(-20000, 'Raise too big');

END IF;

END AFTER EACH ROW;

END Check_Employee_Salary_Raise;