**Triggerek**

A triggert SQL utasítással hozhatjuk létre (CREATE TRIGGER), de a trigger végrehajtható részét PL/SQL nyelven kell megírnunk. A trigger valamilyen esemény hatására automatikusan elindul és végrehajtja a PL/SQL blokkban megadott utasításokat. Ezen utasítások végrehajtásához a trigger tulajdonosának kell, hogy joga legyen, méghozzá közvetlenül és nem role-okon keresztül. Az esemény lehet DML utasítás (pl. insert, update), DDL utasítás (pl. create, drop), vagy adatbázis esemény (pl. startup, login).

**DML triggerek**

CREATE [OR REPLACE] TRIGGER [schema.]trigger

 {BEFORE | AFTER | INSTEAD OF}

 {DELETE | INSERT | UPDATE [OF column [, column] ...]}

[OR {DELETE | INSERT | UPDATE [OF column [, column] ...]}] ...

 ON [schema.]table

 [ [REFERENCING { OLD [AS] old [NEW [AS] new]

 | NEW [AS] new [OLD [AS] old] } ]

 [ FOR EACH ROW

 [WHEN (condition)] ]

 pl/sql\_block

A triggerhez tartozik egy **kiváltó (elsütő) művelet** (INSERT, DELETE, UPDATE).

A trigger egy **objektumhoz** (tábla vagy esetleg nézet) kötődik.

**Időzítés**: A trigger egy módosító művelet előtt vagy után fut le.

Trigger **típusa**: Ha megadjuk a FOR EACH ROW opciót akkor a trigger minden sorra egyszer végrehajtódik. Az ilyen triggereket **sor-trigger**nek hívjuk. Ellenkező esetben csak utasításonként egyszer hajtódik végre a trigger. Ekkor a neve **utasítás-trigger**.

**When feltétel** csak sortriggerre adható meg. Ilyenkor a trigger csak azokra a sorokra fut le, amelyek kielégítik a feltételt.

Triggerek **engedélyezhetők** vagy letilthatók (ALTER TRIGGER)

Ha egy művelet több triggert is aktivizál akkor azok futási sorrendje nem garantált.

Triggeren belül nem adható ki tranzakciókezelő utasítás. COMMIT, ROLLBACK, SAVEPOINT

Az oszlopok régi és új értékére a PL/SQL blokkban úgy hivatkozhatunk mint host változókra. (kettőspont a változó előtt: **:NEW.oszlop**, **:OLD.oszlop**)

BEFORE triggerben az új értéket meg is változtathatjuk és ekkor ez kerül be majd az oszlopba. AFTER trigger esetén ezt nem tehetjük meg.

Egy AFTER trigger viszont már használhatja a ROWID-jét a sornak.

**A triggerek aktivizálódási sorrendje:**

1. BEFORE utasítás szintű triggerek

2. Minden egyes érintett sorra

 a) a BEFORE sor szintű triggerek

 b) maga a DML utasítás és az integritási feltételek ellenőrzése (részben)

 c) az AFTER sor szintű triggerek

3. integritási feltételek ellenőrzésének befejezése

4. AFTER utasítás szintű triggerek

A trigger futása alatt a rendszer egy READ konzisztens állapotát garantálja minden hivatkozott táblának, így a trigger nem látja a futása alatt történt változásokat.

Mire vigyázzunk triggerek megadásakor?

Amit deklaratív módon is meg tud oldani az Oracle arra ne írjunk triggert.

Ne hozzunk létre rekurzív triggereket. (Pl. egy AFTER UPDATE trigger ne adjon ki update utasítást.)

INSERT esetén csak a NEW értékeknek van értelme, a régiek NULL-ok.

DELETE esetén csak az OLD értékeknek van értelme, az újak NULL-ok.

A WHEN után még nem kell kettőspont az OLD és NEW elé, csak a blokkban.

Ha több művelet elsütheti a triggert akkor így dönthetjük el melyik volt a tényleges:

 IF INSERTING ... IF UPDATING [(‘oszlop’)]... IF DELETING ...

Update esetén még az oszlopot is megtudhatjuk.

Ha a trigger közben kivétel lép fel, amit nem kezeltek le akkor a trigger és az elsütő művelet is ROLLBACK-elve lesz. Így lehet pl. triggerből visszacsévélni az eredeti műveletet. Gyakran erre a célra a RAISE\_APPLICATION\_ERROR(hibakód, hibaüzenet) procedúrát használják, mert ekkor a kiváltó műveletet kiadó program kultúráltan lekezelheti a hibát.

DDL és tranzakció-kezelő utasítás nem lehet a triggerben.

Információk a triggerekről: DBA\_TRIGGERS

**Megszorítások a triggerek használatával kapcsolatban:**

Hivatkozó tábla elsődleges és idegen kulcs oszlopát nem módosíthatja a trigger. (Hivatkozó tábla az, amelyik idegen kulcs hivatkozásban van a módosított táblával, amire épp sor szintű trigger fut.) Így az alábbi csak akkor működik ha nincs idegen kulcs definíció a két tábla között.

CREATE OR REPLACE TRIGGER cascade\_upd

AFTER UPDATE OF deptno ON dept

# FOR EACH ROW

BEGIN

 UPDATE emp SET emp.deptno = :new.deptno

 WHERE emp.deptno = :old.deptno;

END;

Változás alatt lévő táblát nem olvashat a trigger. (Változás alatti az a tábla, amelynek módosítása alatt épp sor szintű trigger fut.) Vagyis egy sor szintű trigger nem olvashat egy éppen módosítás alatt levő táblából.

**DDL és Adatbázis triggerek**

Kiváltó eseményük lehet csak egy adott sémára, vagy az egész adatbázisra vonatkozó művelet.

Lehetséges DDL események: CREATE, DROP, RENAME, DDL …

Lehetséges adatbázis események: SERVERERROR, LOGON, STARTUP …

DBA\_TRIGGERS.base\_object\_type -> database, schema, table, view

További infók: 11g dokumentációban PL/SQL Langueage Reference.