Triggery

1) Typy triggerov
2) Tri príklady

1) Triggery

Trigger (spúšťač) je kód, druh uloženej procedúry, event handler, ktorý sa automaticky vykoná pri výskyte istých DB-vých udalostí.

Rozlišujeme **DML**, **DDL** (aj **CLR DDL** Trigger) a **logon** triggery.
- DML udalosti súvisia s prikazmi INSERT, UPDATE a DELETE na tabuľkách alebo pohľadoch. DML triggery sú často používané na realizáciu obchodných pravidiel a integrity dát.
- DDL udalosti súvisia s príkazmi CREATE, ALTER a DROP
- logon udalosť súvisí so začatím nového session-u

Skrátená syntax:

```
CREATE TRIGGER nazov ON tab
 INSTEAD OF | AFTER [| FOR ]
 INSERT | UPDATE | DELETE
 AS ...
```

INSTEAD OF INSERT / UPDATE / DELETE triggery v súlade ich názvami, nahrádzajú INSERT / UPDATE / DELETE dopyty.
Pred SQL SERVER 2000 existoval iba FOR trigger. AFTER ⇔ FOR.
Pre pohľady AFTER trigger nie je možné definovať – iba INSTEAD OF.

Napr.

```
create trigger tr_1 on T1 instead of update as
      BEGIN … END

create trigger tr_2 on T1 after update as
      BEGIN … END

create trigger tr_3 on T1 after insert, delete as -- logika
      BEGIN … END
```

DB systémy používajú dve špeciálne pomocné tabuľky v triggeroch
- **Inserted** (Oracle, MySQL **new**)
- **Deleted** (Oracle, MySQL **old**)

Inserted Tab je vytvorený pre INSERT and UPDATE udalosti.
Deleted Tab je vytvorený pre DELETE and UPDATE udalosti.

@@rowcount vráti počet riadkov ovplyvnených alebo načítaných posledným príkazom.

Entitná, doménová a referenčná integrita by sa mala zabezpečiť predovšetkým pomocou Primary Key, Unique, Foreign Key a CHECK constraint. V istých prípadoch sa odporúčajú DML triggery namiesto obmedzení. Napr. na rozdiel od CHECK obmedzenia, DML trigger môže sa odvolať na stĺpec inej tabuľky, alebo DML trigger môže použiť ROLLBACK TRANSACTION.

Triggery je možné dočasne zakázať a potom umožniť

```
ALTER TABLE tab1 DISABLE/ENABLE TRIGGER trig1
```

Skalárne **trigger funkcie** pre určenie zmien v údajoch http://msdn.microsoft.com/en-us/library/ff848724.aspx

UPDATE( )
COLUMNS_UPDATED
EVENTDATA
TRIGGER_NESTLEVEL

Pre **poradie** a vykonanie triggerov existujú špeciálne pravidlá, napr.:

INSTEAD OF ⇒ Kontrola obmedzení ⇒ AFTER

- Ak tabuľka má obmedzenia, tie sú kontrolované po INSTEAD OF a pred AFTER triggers. Ak obmedzenia sú narušené, potom akcie v INSTEAD OF sú anulované (a AFTER trigger sa nevykoná). AFTER trigger nemôže vyvolať INSTEAD OF trigger (ale v ALTER DATABASE je možné zvoliť RECURSIVE_TRIGGERS).
- Tabuľka pre každú udalosť INSERT / UPDATE / DELETE môže mať viac AFTER triggerov, ale iba jeden INSTEAD OF trigger
- Cascade Delete / Update má prioritu pred Insted of Delete / Update (systém nedovolí definovať trigger).
- V DML triggeroch nie je povolené použiť CREATE / ALTER / DROP DATABASE.

2) Tri príklady

```
-- MySQL
-- DROP TRIGGER IF EXISTS trig_T;
DELIMITER $$
CREATE TRIGGER trig_T2 BEFORE INSERT ON T
FOR EACH ROW
BEGIN
  IF NEW.x > NEW.y THEN SET NEW.y=0;
  END IF;
END $$
DELIMITER ;
INSERT T VALUES ('Roznava', 5, 1);
SELECT * FROM T;
```

idT	x	y
Kosice	NULL	10
Lucenec	4	100
Poprad	3	101
Roznava	5	0
NULL	NULL	NULL

1) Napíšte trigger, ktorý po smrti človeka ukončí jeho neukončený vzťah.
Teda, ak napr.

```
UPDATE Osoba SET dat_smrti = '2008.3.31' WHERE id = 5;
```

=> trigger:

```
UPDATE Vztah SET do = ( SELECT dat_smrti FROM inserted ) WHERE ...
```

Poznámka: pre návrat k pôvodnému stavu(*) NULL dat_smrti, je lepšie riadky 1/3, 2/3, 3/3 okomentovať, ba dokonca vymazať.

```
USE OsobaVztah;
GO
select * from osoba where id IN(5,6);
select * from vztah;
```

	id	meno	priezvisko	rodne_priezvisko	dat_nar	dat_smrti	pohlavie	vyska	vaha	otec	matka
1	5	Jozef	Urban	NULL	1922-10-19 00:00:00.000	NULL	m	199.5	NULL	NULL	NULL
2	6	Maria	Urbanova	Novakova	1937-12-08 00:00:00.000	NULL	z	172.5	57.5	1	2

	id	id_on	id_ona	od	do
1	1	1	2	1937-06-01 00:00:00.000	1967-11-05 00:00:00.000
2	2	3	2	1967-05-12 00:00:00.000	1988-07-22 00:00:00.000
3	3	3	4	1938-12-02 00:00:00.000	1965-03-11 00:00:00.000
4	4	5	6	1953-11-11 00:00:00.000	NULL
5	5	7	8	1970-07-22 00:00:00.000	1975-09-01 00:00:00.000
6	6	11	12	1980-03-04 00:00:00.000	2000-12-31 00:00:00.000
7	7	16	15	1997-07-31 00:00:00.000	2002-12-04 00:00:00.000

```
IF OBJECT_ID('tr_Osoba', 'TR') IS NOT NULL   DROP TRIGGER tr_Osoba;
GO
CREATE TRIGGER tr_Osoba ON Osoba
AFTER UPDATE
AS
      if UPDATE(dat_smrti)
      begin
            UPDATE Vztah SET do = ( SELECT dat_smrti FROM inserted )
            WHERE ( SELECT id FROM inserted) IN (Vztah.id_on, Vztah.id_ona)
      end;
GO
```

Test trigger `tr_Osoba`

```
UPDATE Osoba SET dat_smrti = '2008.3.31' WHERE id = 5; -- skus aj 6

SELECT * FROM osoba WHERE id IN(5,6)
SELECT * FROM Vztah --where id = 4;
```

Návrat k pôvodnému stavu (*)

```
UPDATE Osoba SET dat_smrti = NULL WHERE id = 5;

SELECT * FROM osoba WHERE id IN(5,6)
SELECT * FROM Vztah --where id = 4;
```

2) Zisti, identifikuj typ triggera pomocou

```
      IF EXISTS(SELECT * FROM inserted)
      IF EXISTS(SELECT * FROM deleted)
```

Nezabudni:

inserted Tab je vytvorený pre INSERT a UPDATE
deleted Tab je vytvorený pre DELETE a UPDATE

```
use tempdb;
GO

-- Create table T1
IF OBJECT_ID('dbo.T1') IS NOT NULL
  DROP TABLE dbo.T1;
GO

CREATE TABLE dbo.T1
(
  id INT         NOT NULL PRIMARY KEY,
  aa VARCHAR(10) NOT NULL
);
GO
```

```
IF OBJECT_ID ('trg_T1', 'TR') IS NOT NULL    DROP TRIGGER trg_T1
GO
CREATE TRIGGER trg_T1 ON dbo.T1 AFTER INSERT, UPDATE, DELETE
AS
      DECLARE @rc AS INT;
      SET @rc = @@rowcount;
      -- 1)
      IF @rc = 0
      BEGIN
        PRINT 'Csaba: Nedoslo k zmene v datach';
        RETURN;
      END
      -- 2-3-4) Nezabudni:
      --            inserted Tab je vytvoreny pre INSERT and UPDATE
      --            deleted  Tab je vytvoreny pre DELETE and UPDATE-!!!
      IF EXISTS(SELECT * FROM inserted)
      BEGIN
        IF EXISTS(SELECT * FROM deleted)
        BEGIN
            PRINT 'Csaba: Ide o UPDATE';
        END
        ELSE
        BEGIN
            PRINT 'Csaba: Ide o INSERT';
        END
      END
      ELSE
      BEGIN
        PRINT 'Csaba: Ide o DELETE';
      END
      GO
```

Test `trg_T1` **trigger**

```
-- 0 Rows
INSERT INTO T1 SELECT 1, 'A' WHERE 1 = 0;

-- INSERT
INSERT INTO T1 SELECT 1, 'A';

-- UPDATE
UPDATE T1 SET aa = 'AA' WHERE id = 1;

select * from T1

-- DELETE
DELETE FROM T1 WHERE id = 1;
GO

-- Upratovanie
IF OBJECT_ID('dbo.T1') IS NOT NULL
  DROP TABLE dbo.T1;
GO
```

3) Napíšme trigger, ktorý nedovolí prepísať atribút *id*.

```
use tempdb;
GO

-- Create table T1
IF OBJECT_ID('dbo.T1') IS NOT NULL
DROP TABLE dbo.T1;
GO
CREATE TABLE dbo.T1
(
id INT NOT NULL PRIMARY KEY,
aa VARCHAR(10) NOT NULL
);
GO
```

AFTER či INSTEAd OF trigger je výhodné použiť pre danú úlohu?

```
IF OBJECT_ID('tr1', 'TR') IS NOT NULL DROP TRIGGER tr1;
GO
CREATE TRIGGER tr1 ON dbo.T1 AFTER UPDATE AS
    IF UPDATE(id)
    BEGIN
        RAISERROR ('Nepovolene - tr1', 11, 1)
        ROLLBACK TRAN
        RETURN
    END
GO

IF OBJECT_ID('tr2', 'TR') IS NOT NULL DROP TRIGGER tr2;
GO
CREATE TRIGGER tr2 ON dbo.T1 INSTEAD OF UPDATE AS
    IF UPDATE(id)
    BEGIN
        RAISERROR ('Nepovolene - tr2', 11, 1)
        ROLLBACK TRAN
        RETURN
    END
GO

INSERT INTO T1 SELECT 1, 'A';
UPDATE T1 SET id = 10 WHERE id = 1; -- nedovoli tr2

ALTER TABLE T1 DISABLE TRIGGER tr2
UPDATE T1 SET id = 10 WHERE id = 1; -- nedovoli tr1

ALTER TABLE T1 DISABLE TRIGGER tr1
UPDATE T1 SET id = 10 WHERE id = 1; -- dovolene

ALTER TABLE T1 ENABLE TRIGGER tr1
ALTER TABLE T1 ENABLE TRIGGER tr2
```