MS SQL Triggery (spouště)

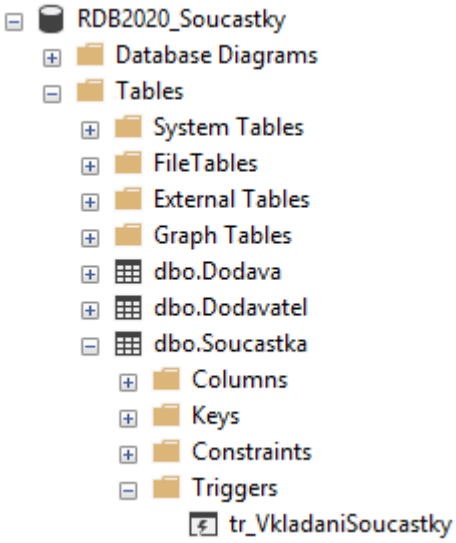
Úvod

Trigger je databázový objekt, který si můžete představit jako spoušť určité činnosti. Jednoduše si nadefinujete, při jaké akci (UPDATE, DELETE, INSERT) se mají spustit určité SQL příkazy. Obecně trigger definuje událost, která zautomatizuje jednotlivé procesy.

Základní rozdělení triggerů

Jsou tři základní typy triggerů:

- **DML** triggery – jsou prováděny automaticky v odpovědi na DML události (INSERT, UPDATE, DELETE)
  - AFTER triggers – zavolá se po úvodní akci. Občas se volá jako (FOR triggery).
  - INSTEAD OF triggery – zavolá se na místo původní akce.
    (V MySQL i jinde se používá klíčové slovo BEFORE, avšak chování je trochu jiné, příkaz se ale vždy po skončení činnosti vykoná. V MS SQL činnost triggeru nahrazuje původní příkaz INSERT, UPDATE či DELETE, který musíme případně zavolat z triggeru znovu.)
  - je svázán s konkrétní DB tabulkou:



- **DDL** triggery – spouští se při CREATE, DROP a ALTER operacích.
- **LOGON** triggery – spouští se při vytvoření uživatelské session.

V rámci jednoho triggeru existují na MS SQL Serveru dvě pseudotabulky. Ty zpřístupňují nová a stará data (**Inserted** – pro INSERT a UPDATE, **Deleted** – pro DELETE a UPDATE). V rámci triggeru lze pomocí ROLLBACK zrušit operaci, která trigger spustila. Jeden trigger lze použít i pro více akcí najednou (například UPDATE a DELETE).

Zaměříme se na DML triggery.

Syntaxe

```
CREATE TRIGGER tr_name ON [table||view]
[FOR||AFTER||INSTEAD OF]
[[INSERT],[UPDATE],[DELETE]] AS ...
```

Příklad

Po vytvoření nového uživatele se zapíše událost do tabulky (například report). Zapíšeme si id daného uživatele a kdy byl vytvořen:

```
CREATE TRIGGER tr_user_forInsert
ON user
FOR INSERT AS
BEGIN
  DECLARE @id int
  SELECT @id FROM Inserted
  INSERT INTO myLogTable VALUES ('new user with id = '
                                 + CAST(@id AS nvarchar(5)) +
                                 ' is added at ' + CAST(GETDATE() AS
nvarchar(20)))
END
```

Tento trigger se spustí při zápisu nového uživatele do tabulky user. Zapisuje novou událost do tabulky LogTable.

Ten samý postup bychom zvolili například při vymazání uživatele. Pak bychom trigger pouze trochu pozměnili. Místo původního FROM Inserted bychom zapsali FROM Deleted.

```
SELECT * FROM Inserted
SELECT * FROM Deleted
```

Takto by se trigger zavolat při příkazu DELETE nad tabulkou user.

Příklad 2 – vymezení sloupců

Máme například tabulku produktů, která má následující strukturu:

id	sekce_id	cenova_hladina_id	nazev	url	cena
3	2	5	prod1	/prod1	1000

Pokud budeme chtít upravit název, url nebo cenu, tak se příkaz UPDATE provede. Důležité je však zamezit úpravu sloupce id, sekce_id a cenova_hladina_id. Proto musíme při aktualizaci zjistit, které sloupce se mají přepsat a podle toho UPDATE dovolit nebo naopak zrušit.

```
CREATE TRIGGER tr_user_forInsert
ON user
FOR INSERT AS
BEGIN
  IF (update(id))
    BEGIN raiserror('Nemůžete upravovat sloupec id!', 16, 1)
      return
    END

  IF (update(sekce_id))
    BEGIN raiserror('Nemůžete na přímo upravovat sloupec sekce_id', 16, 1)
      return
    END

    IF (update(cenova_hladina_id))
      BEGIN raiserror('Nemůžete na přímo upravovat sloupec
cenova_hladina_id', 16,1)
        return
      END

      -- zde bude update
END
```

AFTER UPDATE trigger

Tento typ triggeru umožňuje použít dvě pseudotabulky (INSERTED, DELETED). Inserted tabulka obsahuje změněná data a Deleted tabulka obsahuje stará data. Pro demonstraci použijeme příklad s eventy:

```
CREATE TRIGGER tr_user_afterUpdate
ON user
AFTER UPDATE AS
BEGIN
  DECLARE @old_name varchar, @new_name varchar, @odl_age int, @new_age int,
@TEXT VARCHAR(1000), @id
  SELECT * INTO #TempTable;
  WHILE (EXIST(SELECT id FROM #TempTable))
    BEGIN
      SET @TEXT = '';
      SELECT TOP 1 @new_name = name, @new_age = age, @id = id FROM
#TempTable
      SELECT @old_name = name, @odl_age = age FROM Deleted WHERE id = 1
      SET @TEXT = 'uživatel s id = '+ @id + 'Provedl změny: '
      IF (@old_name <> @new_name)
        SET @TEXT = @TEXT + ' Přejmenoval se z ' + @old_name +  ' na ' +
@new_name + '.'
      IF (@odl_age <> @new_age)
        SET @TEXT = @TEXT + ' Změna roku z ' + @old_age + ' na ' + @new_age
+ '.'
```

```
      INSERT INTO myReportTable VALUES (@TEXT);
    END
END
```

INSTEAD OF trigger

Tento druh triggeru se spouští místo dané akce. Například INSTEAD OF UPDATE se spustí místo původního UPDATE. Takže **musíme aktualizaci zavolat z triggeru**.

Příklad

Trigger se bude spouštět nad tabulkou tblDepartment. Pokud se do proměnné @DetId zapíše nějaké id (identifikátor daného oddělení) z tabulky tblDepartment, vypíše se error přes funkci Raiserror a příkaz INSERT se neprovede.

Pokud budeme chtít do tabulky uložit oddělení, které již existuje (poznáme tak, že hodnota Inserted.DeptName se přes JOIN spojí s nějakým záznamem tblDepartment.DeptName), tak se příkaz zruší.

V opačném případě můžeme zapsat nový záznam do tabulky tblDepartment.

```
CREATE TRIGGER tr_user_afterUpdate
ON user
AFTER UPDATE AS
BEGIN
  DECLARE @DetId int
  SELECT @DetId = DetId FROM tblDepartment
    JOIN Inserted ON Inserted.DeptName = tblDepartment.DeptName

  IF (@DetId IS NULL)
  BEGIN
    Raiserror('bla bla bla',16,1)
    return
  END

  INSERT INTO tblEmployee (id, name, gender, departmentId)
  SELECT id, name, gender, @DetId FROM Inserted
END
```