Кориснички дефинисане функције

Постоје три врсте корисничких дефинисаних функција:

1. Скаларне функције – повратна вредност је скалар, користи се као израз у SELECT, WHERE, ORDER BY, GROUP BY клаузулама упита
2. Inline Table-Valued функција – повратна вредност је сет података кога дефинише SELECT наредба у функцији; слична погледима али може имати параметре у WHERE клаузули (параметаризовани поглед); имају само једну SELECT наредбу; не модификују податке
3. Multistatement Table-Valued функције – повратна вредност је тип табеле са свим својим деловима и подацима у њој; користи се у FROM клаузули; не модификује податке

Скаларне кориснички дефинисане функције

Синтакса за креирање скаларне кориснички дефинисане функције:

```
CREATE FUNCTION sema.ImeFunkcije
(parametar1, parametar2, ..., parametarn)
RETURNS povratni_tip_podataka
WITH (ENCRYPTION | SCHEMABINDING | EXECUTE AS DB_ImeKorisnika)
AS
BEGIN
      telo_funkcije
      RETURN skalarni_izraz
END
```

Са RETURNS се дефинише повратни тип функције и може бити било који од стандардних, алијас или корсинички дефинисан тип који постоји у бази података где се и креира функција.
Клаузула WITH је опциона и може да се корисити као:

- ENCRYPTION – скрипт који креира функцију с ене чува у системским табелама
- SCHEMABINDING – не дозвољава модификацију објеката на које се реферише функција
- EXECUTE AS – увек покреће функцију у безбедоносном контексту задатог корисника
- RETURN – у телу функције враћа вредност из функције и прекида њено даље извршавање; тип повратне вредности мода да буде исти као тип функције декларисане са RETURNS

Пример:

```
CREATE FUNCTION [dbo].[KupacBrojFaktura]
(@MusterijaID int)
RETURNS int
WITH SCHEMABINDING
AS
BEGIN
      DECLARE @Brojac int = 0;
      SELECT @Brojac = COUNT(*)
      FROM dbo.Fakture
      WHERE MusterijaID = @MusterijaID;
      RETURN @Brojac;
END
```

Функција се тестира са SELECT упитом: SELECT dbo.KupacBrojFaktura (10);
И као резултат се добија врендост 15.
Ова функција за задатог купца враћа укупан број реализованих фактура.
Улазни параметар у функцију је вредност примарног кључа (MusterijaID) табеле dbo.Musterije.
У телу функције се помоћу агрегатне функције COUNT(*) и WHERE клаузуле добија укупан број фактура за задатог купца и та вредност је повратна вредност функције.
Са опцијом SCHEMABINDING се спречава брисање и измена структуре табеле dbo.Fakture на коју се реферише ова функција.
Измене структуре табела које не утичу на рад функције су дозвољене.

Окидачи

Окидач (trigger) је именован и снимљен скрипт који се извршава искључиво посредно, када се изврши SQL акција за коју је везан – не може се директно позвати.
У SQL Server бази података постоје следеће врсте окидача:

1. DML окидачи (Data Manipulation Language) су окидачи који се везују за SQL наредбе INSERT, UPDATE, DELETE; за наредбе које додају, апдејтују и бришу податке
2. DDL окидачи (Data Definition Language) се везују за SQL наредбе CREATE, ALTER, DROP; за наредбе које креирају, мењају и бришу податке
3. Logon окидачи се везују за моменат логовања корисника

DML окидачи

Они се везују за неку од акција над подацима (INSERT, UPDATE, DELETE) које се извршавају над одређеним табелом.
На пример може се направити окидач који се извршава сваки пут када се изврши INSERT naredba nad tabelom dbo.Musterije, чиме се аутоматски покреће скрипт који је тело окидача.
Ови окидачи се користе за сложеније валидације приликом уноса, измене или брисања података, али и за синхронизацију и пренос података према другим серверима, логовање активности корисника...
Покретање окидача је везано за једну трансакацију.
Ако је окидач везан за DELETE акцију над табелом и корисник покрене један DELETE упит који брише више записа, овај окидач ће се покренути само једном (исто важи и за INSERT и за UPDATE).
Ако је потребно поништити резултате окидача користи се команда ROLLBACK TRANSACTION.
У телу окидача су доступне две виртуелне табеле: inserted и deleted.
Структурно оне су идентичене са табелом над којом се извршава окидач и зависно од акције за коју се покреће окидач, њихов садржај може бити:

Табеле/Акција	INSERT	UPDATE	DELETE
inserted	Унесени слогови	Слогови после измене	празна
deleted	празна	Слогови пре измене	Обрисани слогови

Нпр ако постоји окидач за UPDATE наредбу дефинисан над табелом и потом се изврши UPDATE наредба која ажурира записе, табела inserted ће имати нове вредности записа, а табела deleted ће имати старе вредности и обе табеле ће имати исти број записа.

Постоје две врсте DML окидача:

1. After trigger – покреће се после акције; подаци су већ измењени и тек онда се покреће одговарајући окидач; али, ако акција није у складу са неким од ограничења (примарни кључ, страни кључ) акција се неће извршити па се окидач неће ни покренути
2. Instead of trigger – покреће се уместо акције; оригиналне акције које се покренуле окидач се не извршавају; у телу окидача се после провере могу извршити акције или се изменити

Синтакса за креирање окидача:

```
CREATE trigger sema.ImeOkidaca
ON {tabela | pogled}
[WITH ENCRYPTION, EXECUTE AS DB_Ime_Korisnika]
{FOR | AFTER | INSTEAD OF}
INSERT, UPDATE, DELETE
AS sql_klauzula;
```

Приликом креирања окидача задавање шеме је опционо.
ON опција дефинише над којом табелом или погледом се креира окидач.
Опције ENCRYPTION и EXECUTE AS имају исту улогу као код ускладиштених функција.
За After окидаче се могу користити речи FOR или AFTER а за Instead окидаче се користи кључна реч INSTEAD OF.
Наводе се акције за које се везује окидач (INSERT, UPDATE, DELETE).
Окидач се може везати за једну, било коју комбинацију две или три акције.
Може се креирати више окидача за исту табелу и исту акцију.

Пример за after окидач:
У табели dbo.Musterije се налази колона KreditLimit.
Нека је уведено правило да се не може брисати купац који има KreditLimit већи од просечног кредитног лимита израчунатог од свих купаца у табели.
Ради поједностављења примера, креираће се нова табела на основу dbo.Musterije и у њу пренети сви купци где је KreditLimit различит од NULL.
Нова табела ће се звати dbo.TestMusterije.

```
--brise se test tabela ako vec postoji
IF OBJECT_ID('TestMusterije') IS NOT NULL
      DROP TABLE dbo.TestMusterije;

--kreira se test tabela
SELECT MusterijaID, MusterijaIme, KleditLimit
INTO dbo.TestMusterije
FROM dbo.Musterije
WHERE KreditLimit IS NOT NULL;

--dodaje se primarni kljuc u test tabelu
ALTER TABLE dbo.TestMusterije
ADD CONSTRAINT PK_TestMusterije
PRIMARY KEY (MusterijaID);
```

Пошто је просечна вредност за колону KlreditLimit промењива величина, мора се увек израчунати у телу окидача и вршити провера у односу на вредност колоне KleditLimit записа који се жели обрисати.
Окидач се креира на табели dbo.TestMusterije и изводи DELETE акцију:

```
CREATE trigger dbo.KreditLimitProvera
ON dbo.TestMusterije
FOR DELETE
AS
DECLARE @Prosek decimal(18, 2);
SELECT @Prosek = AVG (KreditLimit) FROM sbo.TestMusterije;
IF EXISTS (SELECT MusterijaID FROM deleted WHERE KreditLimit > @Prosek)
BEGIN
      ROLLBACK TRANSACTION;
      PRINT 'Brisanje nije dozvoljeno';
END
```

У окидачу се користи EXISTS функција са којом се проверава да ли међу потенцијално више обрисаних купаца постоји бар један чији је KreditLimit већи од просека.
Ако једна DELETE команда брише више слогова, овај окидач ће се покренути само једном.
Окидачи се покрећу једном по трансакцији, а не по обисаном, додатом или измењеном запису у табели.
Ако постоји само један купац који се не сме обрисати, врши се ROLLBACK трансакције, тако да неће бити обрисани ни они који се могу брисати.

Тестирање окидача
Провера колико је просек колоне KreditLimit:
SELECT AVG (KreditLimit) FROM dbo.TestMusterije;
Даје 2614.85.

Ако се покуша обрисати купац са MusterijaID = 801 и који има KreditLimit од 3000.00 са
DELETE FROM dbo.TestMusterije
WHERE MusterijaID = 801;
Добија се порука:
Brisanje nije dozvoljeno

Ако се проба обрисати купци са вредностима примарног кључа 801, 803 и 810 чије су вредности за KreditLimit 3000.00, 2000.00, 1200.00:
DELETE FROM dbo.TestMusterije
WHERE MusterijaID IN (801, 803, 810);

Од ова три купца два (803 и 810) могу да се бришу јер им је KreditLimit мања од просека.
Међутим и сада се добија иста порука из окидача и ниједан од три купца неће бити обрисан.

Ако се проба обрисати само купце које је дозвољено обрисати:
DELETE FROM dbo.TestMusterije
WHERE MusterijaID IN (803, 810);
Трансакција успешно пролази и оба купца су обрисана.
На крају се брише и тест табела чиме се аутоматски бришу сви њени пратећи објекти, укључујући и окидаче:
DROP TABLE dbo.TestMusterije;