SQL Server PRIMERI - nastavak

STORNE PROCEDURE i FUNKCIJE

Promenljive
Promenljive su simboli koji mogu da uzmu vrednost bilo kog tipa. U okviru složenih naredbi promenljive imaju lokalni karakter.

Promenljive sadrže vrednosti u koje se prenose učitane vrednosti iz baze podataka, ili su u njima vrednosti koje se predaju bazi podataka. Posle deklaracija parametara u procedurama i funkcijama se mogu navesti lokalne promenljive prema sledećoj sintaksi:

```
DECLARE <ime promenljive > <tip podataka > ;
```

Dodeljivanje vrednosti promenljivama se vrši prema sledećoj sintaksi:

```
SET <ime promenljive> = <value expression> ;
```

Upravljanje greškama

Izvršavanje SQL naredbi može da dovede do uspešnih rezultata, sumnjivih rezultata ili potpuno pogrešnih rezultata. Svaka od ovih grešaka korespondira sa nekim uslovom. Kada god se izvede SQL naredba server baze podataka smešta vrednost u parametar koji se naziva SQLSTATE. SQLSTATE je opisan kao niz znakova dužine pet. Prva dva znaka označavaju uspešnost izvršavanja naredbe i to:
00 - Uspešno izvršena naredba
01 – Upozorenje (nekada se naredba završava uspešno, a ponekad neuspešno; potrebno je proveriti naredbu;)
02 – Rezultat nije vraćen. Generiše tip uslova NOT FOUND.
Sve druge vrednosti koje se pojave znače grešku. Bliže objašnjenje greške je sadržano u preostala tri znaka. Upravljanje greškama znači predvideti koje vrste grešaka mogu da se pojave i kakva će akcija u tom slučaju da usledi. Neke greške mogu da se obrade, tj. da se odredi dalji tok. Druge greške su fatalne i dovode do prekida aplikacije. Generišu tip uslova SQLEXCEPTION.
U zavisnosti koji uslov je ispunjen se određuje akcija. Moguće su sledeće akcije:

`CONTINUE` – nastavlja izvršenje odmah posle naredbe koja je prouzrokovala izuzetak.
`EXIT` – nastavlja izvršavanje posle složene naredbe koja sadrži rukovanje izuzetkom.
`UNDO` – poništava rad prethodnih naredbi u složenoj naredbi i nastavlja izvršavanje posle naredbe koja sadrži rukovanje izuzetkom.

Rukovanje izuzetkom može da se postavi u okviru složene naredbe. Sadrži najpre deklaraciju uslova kojim će da se upravlja. Uslov koji se deklariše može da bude izuzetak ili provera istinitosti.

U primeru koji sledi je prikazano upravljanje greškom. Radi se o proširenju prethodnog primera u kome je sada dodato upravljanje greškom. Ako bilo koja naredba *INSERT* uzrokuje narušavanje ograničenja, kao što je dodavanje vrsta sa istim primarnim ključem, SQLSTATE će dobiti vrednost 23000 i postaviće *narušavanje_ograničenja* na TRUE. To govori rukovaocu izuzecima da poništi sve promene u bilo kojoj tabeli koje je učinila naredba *INSERT*. Naredba *RESIGNAL* vraća kontrolu proceduri koja je pozvala proceduru koja se izvršava. Ako su obe naredbe *INSERT* izvršene uspešno, izvršavanje se nastavlja s naredbom koja je iza ključne reči *END*. Ključna reč ATOMIC postaje aktivna kad god je rezultat poništavanje akcija u složenoj naredbi (UNDO). Ovo nije slučaj ako su rezultat provere akcije *CONTINUE* ili *EXIT*. Postavlja se pitanje, šta će da se dogodi ako vrednost SQLSTATE nije 23000. u tom slučaju kontrola toka programa će da se

prenese na viši nivo od koga zavisi da li će izvršavanje da se nastavi ili će doći do prekida aplikacije. Preporuka je da se predvide akcije i za sve druge moguće vrednosti SQLSTATE.

```
BEGIN ATOMIC
DECLARE narušavanje_ograničenja CONDITION
FOR SQLSTATE VALUE ‘23000’ ;
DECLARE UNDO HANDLER
FOR narušavanje_ograničenja
RESIGNAL ;
INSERT INTO Nastavnici (Snast, Imen)
VALUES (:Snast, :Imen) ;
INSERT INTO Angažovanje (ClassID, Class, StudentID)
VALUES (:Snast, :Spred, :Ssmer) ;
END
```

Naredbe za kontrolu toka

Uvođenjem naredbi za kontrolu toka SQL je dobio svojstva proceduralnih programskih jezika. To znači da u slučaju potrebe uključenja naredbi za kontrolu toka nije potrebno da se kontrola predaje jeziku domaćinu, pa ponovo vraća na SQL.

```
IF    ... THEN
           BEGIN
           ...
           END
          ELSE
           BEGIN
           ...
           END

CASE a
      WHEN 1 THEN ...
      WHEN 2 THEN ...
      ...
      ELSE ...
      END
ili
CASE
      WHEN uslov1 THEN ...
      WHEN uslov2 THEN ..
      ...
      ELSE ...
      END

WHILE... BEGIN ... END

WHILE Boolean_expression
     { sql_statement | statement_block | BREAK | CONTINUE }

WAITFOR { DELAY 'time' | TIME 'time' }
```

```
CREATE FUNCTION formiranje_imena
      (@ime VARCHAR(30),
      @prezime VARCHAR(30) )
RETURNS VARCHAR(60)
AS
BEGIN
   DECLARE @puno_ime VARCHAR(60)
   SET @puno_ime = @ime + ' ' + @prezime
   RETURN @puno_ime
END

SELECT formiranje_imena (Ime, Prezime) AS Ime, id AS Šifra
FROM Imenik
```

```
CREATE PROCEDURE azuriranje_ocena
      (@Indeks SMALLINT,
       @Upisan SMALLINT,
       @Spred SMALLINT,
       @Snast SMALLINT,
      @azur_ocena SMALLINT)
AS
BEGIN
  UPDATE Prijave SET Ocena = @azur_ocena
WHERE Indeks = @Indeks AND Upisan = @Upisan AND Spred = @Spred AND Snast =
@Snast;
END;
```

izvršavanje

```
EXECUTE azuriranje_ocena_2 @indeks=1, @upisan=2002, @snast=7, @azur_ocena=8

CREATE FUNCTION [dbo].plan_na_smeru(@smer SMALLINT)
RETURNS TABLE
AS
      RETURN (SELECT * FROM Planst WHERE Ssmer = @smer)
```

Upotreba funkcije u naredbi SELECT:

```
SELECT * FROM plan_na_smeru(3)
```

```
CREATE FUNCTION [dbo].[provera_usl_predmet] (@INDEKS INT, @UPISAN INT, @Spred
INT)
RETURNS VARCHAR(3)
AS BEGIN
DECLARE @Uslovni_predmet INT, @P VARCHAR(3), @Polozen INT
SELECT @Uslovni_predmet = UslPredmet FROM Uslovni WHERE USLOVNI.Spred = @Spred
SELECT @Polozen = Ocena FROM PRIJAVE WHERE PRIJAVE.Spred = @Uslovni_predmet AND
Indeks = @INDEKS AND Upisan = @UPISAN AND Ocena > 5
IF @Uslovni_predmet IS NULL
   SET @P = 'OK'
ELSE
IF (@Uslovni_predmet > 0 AND @Polozen > 5)
   SET @P = 'OK'
  ELSE
```

```
    SET @P = 'NOT';
RETURN @P
END
```

Primeri izvršavanja ove funkcije:

```
SELECT dbo.provera_usl_predmet(2,2002,7)
```

```
CREATE PROCEDURE novo_angazovanje (@nastavnik SMALLINT, @predmet SMALLINT,
@smer SMALLINT)
AS
DECLARE @NAST VARCHAR(7)
SELECT @NAST =  Snast FROM Angazovanje WHERE Spred = @predmet AND ssmer = @smer
IF @NAST IS NULL
        INSERT INTO Angazovanje  VALUES(@nastavnik, @predmet, @smer)
     ELSE
        SELECT 'Angazovanje vec postoji';

EXECUTE novo_angazovanje 1,19,2
```

TRIGERI

SQL podržava objekte koji izvršavaju akcije automatski. Ovi objekti se nazivaju trigeri i reaguju na modifikacije podataka u okviru tabele. Ove modifikacije se odnose na upis, izmene i brisanje podataka. U vezi sa tim postoje tri vrste trigera: INSERT, UPDATE i DELETE. Kao objekat baze podataka jedan triger je povezan samo sa jednom tabelom.
Kod trigera razlikujemo tri komponente: **događaj**, koji predstavlja upravo određenu modifikaciju, **uslov** koji predstavlja proveru određenog uslova da bi triger bio pokrenut. Rezultat provere uslova je TRUE ili FALSE. Ako je uslov ispunjen pokreće se automatski **akcija**. Funkcionalnost koju trigeri daju bazi podataka iskazuje se preko posebnog naziva - *aktivna baza podataka*.
Triger se izvršava u svom izvođačkom kontekstu. Ovaj kontekst čini memorijski prostor koji sadrži procese naredbi koje triger pokreće. Izvođački kontekst trigera se kreira uvek kada se triger pozove. Ako je pozvano više trigera izvođački kontekst se kreira za svakog od njih. U jednom trenutku je aktivan samo jedan izvođački kontekst.
Izvođački kontekst uključuje i jednu ili dve *tabele izmena* (transition table) koje se često nazivaju i *pseudo tabele*. Tabela izmena je virtuelna tabela koja sadrži podatke koji su ažurirani, upisani ili obrisani iz tabele. Ako se vrši ažuriranje, kreiraju se dve tabele izmena, jedna za stare podatke, druga za nove podatke. Ako se vrši upisivanje, kreira se jedna tabela izmena za nove podatke, a ako se brišu podaci kreira se jedna tabela izmena za stare podatke.

SQL Server omogućava više trigera za datu operaciju manipulacije podacima na tabeli ili pogledu. Tako su moguća tri trigera *UPDATE* na jednoj tabeli Njihov redosled nije posebno definisan, mada prvi i poslednji trigeri mogu eksplicitno da se deklarišu korišćenjem sistemske uskladištene procedure `sp_settriggerorder`.

primer iz MSDN-a

```
IF EXISTS (SELECT name FROM sysobjects
      WHERE name = 'reminder' AND type = 'TR')
   DROP TRIGGER reminder
GO
CREATE TRIGGER reminder
ON titles
FOR INSERT, UPDATE, DELETE
```

```
AS
   EXEC master..xp_sendmail 'MaryM',
      'Don''t forget to print a report for the distributors.'
GO
```

Triger koristi specijalnu tabelu **inserted**. SQL Server 2005 automatski kreira dve privremene tabele **inserted** i **deleted** za testiranje efekata promena podataka i skupa uslova za akcije DML trigera. **Inserted** tabele memorišu kopije vrsta koje su nastale naredbama *INSERT* i *UPDATE*. Za vreme transakcija upisivanja ili ažuriranja vrsta, u isto vreme su dodate nove vrste, i u tabeli na koju se triger odnosi, i u tabeli **inserted**. Vrste u tabeli **inserted** su kopije novih vrsta u tabeli na koju se trigger odnosi. Prilikom izvršavanja transakcija ažuriranja, stare vrste se prvo kopiraju u tabeli **deleted**, a onda nove vrste upisuju u tabeli **inserted** i u tabeli na koji se trigger odnosi. Tabela **deleted**, prema tome, sadrži kopije vrsta nastalih izvršavanjem naredbi *DELETE* i *UPDATE*.

```
CREATE TRIGGER novi_studenti
INSTEAD OF INSERT ON Kandidati
BEGIN
  INSERT INTO Studenti
  SELECT * FROM inserted WHERE Status =  'položio'
  INSERT INTO Kandidati
  SELECT * FROM inserted WHERE status = 'nije položio'
 END
```

Sledeći triger prikazuje kontrolu upisa u novu tabelu koja sadrži uslovne predmete. U odnosu na tabelu Predmeti ova tabela je rekurzivni tip. Kreiranjem spoljašnjeg ključa koji povezuje tabelu Uslovni sa tabelom Predmeti, obezbeđena je kontrola ispravnosti šifre za koju se upisuju uslovni predmeti. Međutim, kontrola ispravnosti upisivanja uslovnog premeta nije postignuta. To se postiže kreiranjem odgovarajućeg trigera koji reaguje na događaj upisa nove vrste u tabeli Uslovni, zatim kontroliše ispravnost šifre uslovnog predmeta i ako je šifra neispravna, odnosno ne postoji u tabeli Predmeti, pokreće akciju koja se sastoji u izdavnju odgovarajuće poruke i prekidanju transakcije.

```
CREATE TRIGGER upis_u_uslovni
ON Uslovni
FOR INSERT
AS
print 'pokrenuo triger'
DECLARE @PREDMET SMALLINT, @P VARCHAR(2)
SELECT @PREDMET = Predmeti.Spred FROM Predmeti, Inserted WHERE Predmeti.Spred =
Inserted.UslPredmet
SET @P = CAST((@PREDMET) AS VARCHAR(2))
print @p
IF  @P IS NULL
BEGIN
  RAISERROR ('Uslovni predmet ne postoji u šifarniku predmeta!', 16, 1)
ROLLBACK TRANSACTION
END

INSERT INTO USLOVNI VALUES(4,58)

Msg 50000, Level 16, State 1, Procedure upis_u_uslovni, Line 10
Uslovni predmet ne postoji u šifarniku predmeta!
Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.

INSERT INTO Uslovni VALUES(4,3)
(1 row(s) affected)
```

KURSORI

Kao podrška ugrađenom SQL-u, SQL standard omogućava deklarisanje kursora koji radi kao pokazivač na određene vrste u rezultatu upita. Kursor je ugrađen u programske jezike slično kao i sve druge SQL naredbe.
Kursor vraća skup podataka, ali omogućava programskim jezicima da pozovu samo jednu vrstu iz tog skupa u jednom trenutku. Puna funkcionalnost kursora je obezbeđena preko četiri SQL naredbe: *DECLARE CURSOR*, *OPEN*, *FETCH* i *CLOSE*. Ove naredbe se primenjuju u navedenom redosledu u kursoru.

- Naredba *DECLARE CURSOR* deklariše kursor definisanjem njegovog imena, njegovih karakteristika i upita koji se poziva kada se kursor otvori. Znači, u osnovi kursora je validna *SELECT* naredba.
- Naredba *OPEN* otvara kursor i poziva naredbu *SELECT* i omogućava da rezultat bude raspoloživ za naredbu *FETCH*.
- Naredba *FETCH* vadi podatke iz tabela baze podataka i smešta ih u promenljive koje predaju podatke jeziku domaćinu ili drugim ugrađenim SQL naredbama. Višestruki pristup vrstama (višestruko korišćenje naredbe *FETCH)* je omogućeno preko struktura jezika domaćina, koje omogućavaju rad u petlji, ali i u okviru samog SQL preko naredbi koje omogućavaju ponavljanje (videti poglavlje 13).
- Naredba *CLOSE* zatvara kursor i tada se ne može pristupiti podacima iz deklaracije kursora.

Sve četiri naredbe se pozivaju iz jezika domaćina.

```
DECLARE generacija_2000 CURSOR FOR
  SELECT Indeks, Upisan
  FROM dbo.Studenti
  WHERE Upisan = 2000
OPEN generacija_2000
FETCH NEXT FROM generacija_2000
WHILE @@FETCH_STATUS = 0
BEGIN
  FETCH NEXT FROM generacija_2000
END
CLOSE generacija_2000
DEALLOCATE generacija_2000
```

Kreiranje ugnježdenog kursora sa ciljem da se ispita da li ima studenata koji su položili sve ispite po nastavnom planu(diplomirani) i da upiše određene podatke u tabelu Diplomirani. Spoljašnji kursor (spoljašnja petlja) ima zadatak da pročita potrebne podatke za sve studente, a unutrašnji da za svakog studenta pronađe plan po kome određeni student studira, a zatim da proveri da li je položio sve ispite po tom planu.
Sistemska promenljiva @@FETCH_STATUS, koja vraća status poslednje tekuće *FETCH* naredbe (0 označava da je naredba uspešno izvršena, -1 da je naredba neuspešno završena ili da je vrsta izvan rezultujućeg skupa, -2 da je očitana vrsta izgubljena)
DEALLOCATE briše veze između kursora i imena kursora ili promenljivih kursora (suprotno naredbi DECLARE CURSOR).

```
DECLARE diplomirani_cursor CURSOR
FOR SELECT Indeks, Upisan, Ssmer FROM Studenti
DECLARE @Indeks INT, @Upisan INT, @Ssmer INT, @Polozen INT, @Spred INT
OPEN diplomirani_cursor
FETCH NEXT FROM diplomirani_cursor INTO @Indeks, @Upisan, @Ssmer
WHILE @@FETCH_STATUS = 0
BEGIN
DECLARE predmeti_cursor CURSOR
FOR SELECT Spred FROM Planst WHERE SSmer = @Ssmer

OPEN predmeti_cursor
FETCH NEXT FROM predmeti_cursor INTO @Spred
WHILE @@FETCH_STATUS = 0
BEGIN
SET @Polozen = 0
SELECT @Polozen = Ocena FROM Prijave WHERE Spred = @Spred AND Indeks = @Indeks
AND Upisan = @Upisan AND Ocena > 5
IF @Polozen = 0
   BREAK
ELSE
   FETCH NEXT FROM predmeti_cursor INTO @Spred
   CONTINUE
END
BEGIN
IF @Polozen > 0
   INSERT INTO Diplomirani VALUES(@Indeks, @Upisan, @Ssmer)
END

CLOSE predmeti_cursor
DEALLOCATE predmeti_cursor

FETCH NEXT FROM diplomirani_cursor INTO @Indeks, @Upisan, @Ssmer
END

CLOSE diplomirani_cursor
DEALLOCATE diplomirani_cursor
```

Rezultat izvršavanja ovog kursora je:

```
SELECT * FROM Diplomirani
```

Indeks	Upisan	Ssmer
2	2001	3
3	2002	4
4	2000	3

TRANSAKCIJE