OBLIKOVANJE BAZA PODATAKA

Predavanje 03

Blic

- URL: https://bit.ly/3Sw0yfY

Okidači

Uvod u okidače

- Okidači (engl. ***triggers***) su posebna vrsta procedura
- Svaki okidač je vezan uz točno jednu tablicu
- **Ne poziva ih korisnik već RDBMS** kao reakciju na događaje
  - Događaj (akcija) uzrokuje pozivanje okidača (reakcija)
  - Događaj i svi time pokrenuti okidači su unutar iste transakcije

- Postoje dvije vrste okidača
  - DDL okidači (engl. *Data Definition Language triggers*)
    - Reakcija na izmjenu strukture baze – rjeđe se koriste
  - DML okidači (engl. *Data Modification Language triggers*)
    - Reakcija na promjenu podataka – češće se koriste

DML okidači

- Elementi koji definiraju DML okidač:
  - Točno jedna tablica (ili pogled) uz koju je vezan
  - Događaji uz koje je vezan i koji uzrokuju njegovo pozivanje:
    - Umetanje podataka (`INSERT` naredba)
    - Izmjena podataka (`UPDATE` naredba)
    - Brisanje podataka (`DELETE` naredba)
  - Svaki okidač može biti vezan uz bilo koju kombinaciju događaja
  - Naredbe koje čine tijelo okidača
- Tri podvrste:
  - BEFORE – izvršavaju se **prije** pokretačke naredbe
  - **AFTER** – izvršavaju se **nakon** pokretačke naredbe
  - INSTEAD OF – izvršavaju se **umjesto** pokretačke naredbe

DML AFTER okidači

- U ovom kolegiju nas zanimaju samo **DML AFTER okidači**
  - Pod pojmom **okidač** ćemo podrazumijevati DML AFTER okidač
- Specifičnosti okidača:
  - Ne možemo ih pozivati direktno, već to za nas radi RDBMS kao odgovor na određene događaje
    - Okidač vezan uz INSERT događaj ćemo pozvati umetanjem redaka
    - Okidač vezan uz DELETE događaj ćemo pozvati brisanje redaka
    - Okidač vezan uz UPDATE događaj ćemo pozvati izmjenom redaka
  - Ne mogu primati parametre i **ne mogu vraćati podatke**
  - Mogu umetati, mijenjati i brisati podatke iz bilo koje tablice
  - Mogu odrađivati većinu operacija kao i procedure (npr. poslati e-mail kad se umetne novi zapisu u tablicu Narudzba)

Kada koristiti okidače

- Prekomjerna upotreba okidača čini bazu složenom i teškom za održavanje zbog **kaskadnog efekta**:
  1. Umetnemo redak u tbl1. To izaziva pokretanje okidača o1.
  2. o1 umeće 5 redaka u tablicu tbl2, što izaziva okidač o2.
  3. Okidač o2 briše 10 redaka u tablici t3.
  4. …

- Okidače treba koristiti **samo ako se problem ne može drukčije riješiti ili je to previše posla**
  - Ako ne smijemo/možemo mijenjati objekte u bazi podataka

- Svaki okidač u bazi treba **dokumentirati**

Primjeri kada koristiti okidače

- Primjer kada koristiti okidače:
  - Kupili smo informacijski sustav koji se sastoji od aplikacije i baze. Sve procedure u bazi su zaštićene. Nama je bitno da umetanjem nove narudžbe prodaja dobije e-mail. Firma od koje smo kupili sustav tu funkcionalnost planira omogućiti u verziji 2019. Stavit ćemo okidač na tablicu Narudzba, vezati ga uz umetanje i iz njega poslati e-mail.

- Primjer kada **NE KORISTITI** okidače:
  - **POGREŠNO**: Imamo tablice Drzava (IDDrzava, Naziv) i Grad (IDGrad, Naziv, DrzavaID). Nemamo nikakvih ključeva u bazi. Na tablicu Grad ćemo staviti okidač vezan uz INSERT. Ako korisnik umetne grad koji referencira nepostojeću državu, okidač neće dozvoliti umetanje.
  - **ISPRAVNO**: povezati tablice stranim ključem.

T-SQL za rad s okidačima

- Kreiranje/izmjena okidača:

```
CREATE/ALTER TRIGGER shema_tablice.naziv_okidača
ON shema_tablice.tablica
AFTER INSERT, UPDATE, DELETE
AS
  niz_naredbi
```

- Možemo navesti bilo koju kombinaciju `INSERT`, `UPDATE` i `DELETE` događaja odvojenu zarezima
- Shema okidača uvijek **mora odgovarati shemi tablice**
- Umjesto `AFTER` možemo koristiti `FOR` (sinonimi)
- Uklanjanje okidača:

```
DROP TRIGGER naziv_okidača
```

Specijalne tablice

- Unutar okidača postoje dvije specijalne tablice:
  - Tablica **inserted**
  - Tablica **deleted**
  - **Ne postoji tablica updated**

- Tablice uvijek postoje
  - Mogu biti prazne ili mogu sadržavati retke
- Struktura obje tablice je jednaka strukturi tablice na kojoj je okidač

- Tablice i njihove podatke nije moguće mijenjati, samo čitati
- Tablice postoje samo unutar okidača, završetkom okidača nestaju i tablice iz memorije

Retci u specijalnim tablicama

- Postojanje redaka u specijalnim tablicama je određeno događajem za koji je okidač pozvan
  - Za `INSERT`
    - Tablica **inserted** sadržava sve upravo umetnute retke
    - Tablica **deleted** je prazna
  - Za `DELETE`
    - Tablica **inserted** je prazna
    - Tablica **deleted** sadržava sve upravo obrisane retke
  - Za `UPDATE`
    - Tablica **inserted** sadrži nove verzije svih upravo ažuriranih redaka
    - Tablica **deleted** sadrži stare verzije svih upravo ažuriranih redaka

Broj redaka u specijalnim tablicama

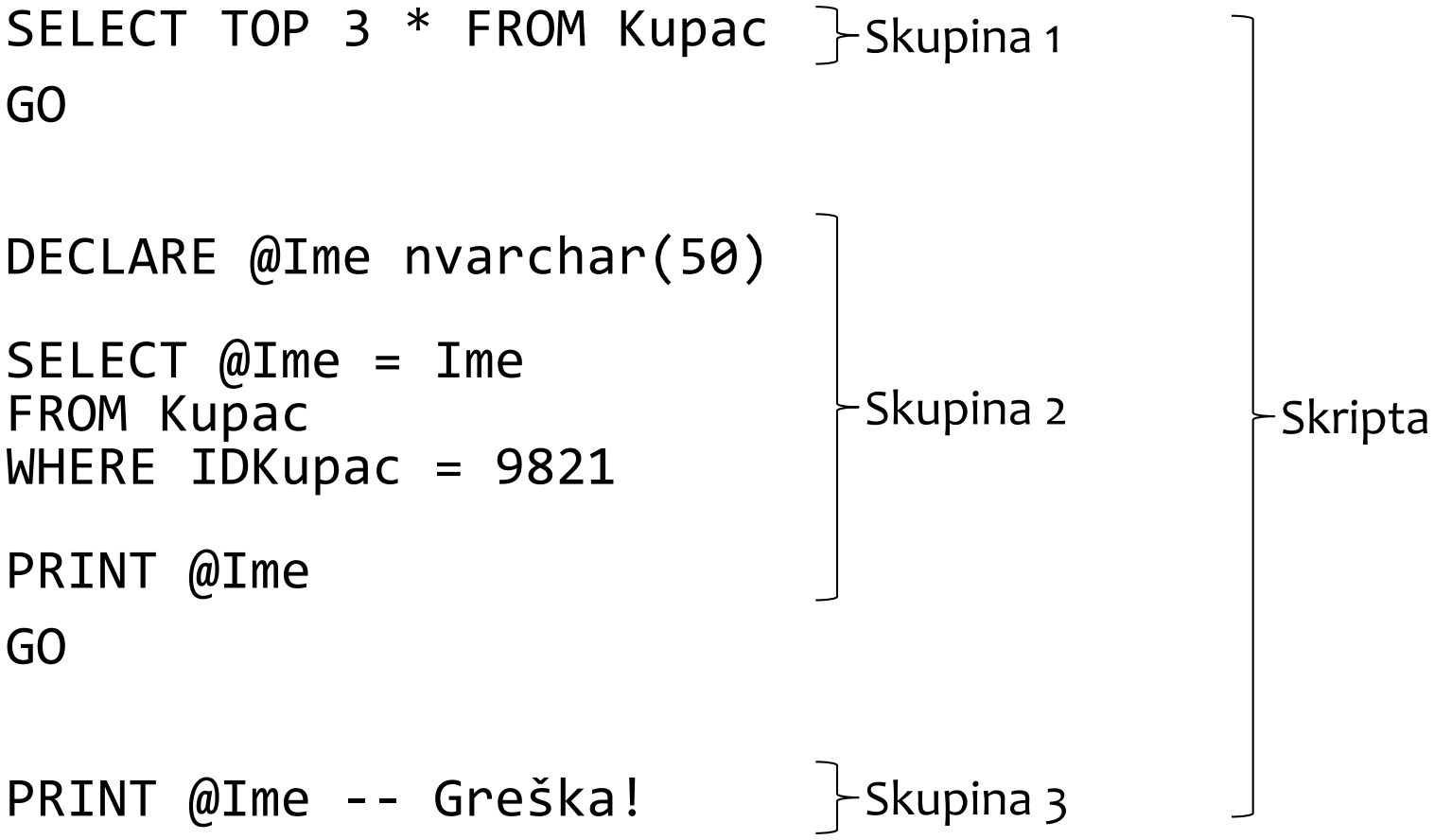
- Specijalne tablice sadržavaju onoliko redaka koliko je bilo pogođenom originalnom SQL naredbom
  - `INSERT INTO Osoba (Ime) VALUES ('Miro')`
    - inserted sadržava 1 redak
    - deleted sadržava 0 redaka
  - `DELETE FROM Osoba WHERE Ime = 'Iva'`
    - inserted sadržava 0 redaka
    - deleted sadržava onoliko redaka koliko ima Iva u tablici
  - `UPDATE Osoba SET Placa = Placa * 1.1`
    `WHERE Placa < 5000.0`
    - inserted i deleted sadržavaju onoliko redaka koliko ima zapisa s plaćom manjom od 5000 kuna

Skripte i skupine naredbi

- T-SQL skripta je **datoteka** s nastavkom **.sql** i sastoji se od proizvoljnog broja naredbi

- Skriptu možemo podijeliti u manje cjeline:
  - **Skupina naredbi** (engl. *batch*) je niz naredbi s karakteristikama:
    - Prije izvršavanja se sve naredbe iz te skupine kompajliraju
    - Izrađuje se plan izvršavanja (engl. *execution plan*)
    - Naredbe se izvršavaju jedna po jedna
  - Varijable iz jedne skupine nisu vidljive drugoj skupini
  - Skupine naredbi se kreiraju korištenjem naredbe `GO`
  - Određene T-SQL naredbe se **moraju** nalaziti u vlastitoj skupini: `CREATE PROC`, `CREATE VIEW`, ...

Primjer skripte sa skupinama naredbi



Drugi primjer skripte sa skupinama naredbi

```
SELECT TOP 3 * FROM Kupac
SELECT TOP 3 * FROM Racun
GO

SELECT TOP 3 * FROM Kupac
SELECT TOP 3 Nepostojece FROM Kupac
GO

DELETE FROM Racun
SELECT TOP 3 * FROM Racun
```

- Koliko skupina ovdje vidimo i što će se desiti izvršavanjem cijele skripte?

Varijable u SQL-u (1/3)

- Deklariranje varijable
  - Varijabla je vidljiva samo u onoj skupini naredbi (*batchu*) u kojoj je deklarirana

```
DECLARE @naziv tip -- Bilo koji SQL tip
```

- Tri načina dodjeljivanja vrijednosti:
  1. Direktno:

```
DECLARE @ime nvarchar(50)
SET @ime = 'Ana'
```

  2. Pomoću podupita koji vraća skalar:

```
DECLARE @prodano int
SET @prodano = (SELECT SUM(Kolicina) FROM Stavka)
```

Varijable u SQL-u (2/3)

3. Pomoću SELECT upita koji bi trebao vratiti **jedan redak**:

```
DECLARE @NazivProizvoda nvarchar(50)
DECLARE @BojaProizvoda nvarchar(50)
SELECT
    @NazivProizvoda = p.Naziv,
    @BojaProizvoda = p.Boja
FROM Proizvod AS p WHERE p.IDProizvod = 741
```

- Ako SELECT vrati **nula redaka**, obje varijable će sadržavati NULL vrijednosti
- Ako SELECT vrati **više redaka**, varijable će poprimiti vrijednosti **iz zadnjeg vraćenog retka**

Varijable u SQL-u (3/3)

- Prema onome što vraćaju korisniku, razlikujemo dvije vrste SELECT naredbi
  - SELECT koji korisniku vraća skup stupaca i skup redaka
  - SELECT koji varijablama dodjeljuje vrijednosti
    - Ovakav SELECT korisniku ne vraća nikakve podatke već ih stavlja u varijable

Korištenje varijabli

- Varijable možemo koristiti na onim mjestima u upitu gdje se **očekuje jedna vrijednost**, primjerice:

```
DECLARE @ProsjecnaCijena money
SELECT @ProsjecnaCijena = AVG(CijenaBezPDV)
    FROM Proizvod AS p
SELECT * FROM Proizvod
    WHERE CijenaBezPDV >= @ProsjecnaCijena
GO;
```

- Vrijednost varijable možemo ispisati pomoću naredbi SELECT i PRINT:

```
PRINT @ProsjecnaCijena -- Korisno samo za debuggiranje
SELECT @ProsjecnaCijena
```

Primjeri

1. Deklarirajte varijable @Ime i @Prezime i dodijelite im neke vrijednosti. Prikažite dodijeljene vrijednosti.

2. Deklarirajte varijable @Ime i @Prezime i dodijelite im vrijednosti iz tablice Kupac za IDKupac jednak 16. Prikažite dodijeljene vrijednosti.

3. Deklarirajte varijable @Ime i @Prezime i dodijelite im vrijednosti iz tablice Kupac tako da odaberete sve retke iz tablice. Prikažite dodijeljene vrijednosti.

Primjeri

Napravite tablicu Student i u nju umetnite nekoliko redaka. Napravite tablicu Zapisnik.

1. Napravite okidač kojim ćete svako umetanje retka u tablicu Student zapisati u tablicu Zapisnik.

2. Promijenite okidač tako da zapiše ime, prezime i JMBAG umetnutog studenta u Zapisnik.

3. Promijenite okidač tako da se veže uz sve događaje i u Zapisnik zapisuje broj redaka u inserted i deleted tablicama. Umetnite novog studenta, promijenite svim studentima prezime i na kraju obrišite sve studente.

4. Promijenite okidač tako da upisuje staru i novu vrijednost promijenjenog prezimena u Zapisnik.

Uključivanje i isključivanje okidača

- Uključeni (engl. *enabled*) okidači se pozivaju od strane RDBMS-a kao reakcija na događaje
- Isključeni (engl. *disabled*) okidači se ne pozivaju – dobije se efekt kao da nisu niti kreirani
  - Isključivanje okidača se obično radi pri umetanju većih količina podataka u tablicu

- T-SQL sintaksa:

```
DISABLE TRIGGER naziv_okidača ON tablica
ENABLE TRIGGER naziv_okidača ON tablica
```

Okidač vezan uz više događaja

- Jedan okidač može biti vezan uz samo jedan, dva ili sva tri događaja
- U okidaču vezanom za dva ili tri događaja moramo moći razlikovati koji se događaj desio
- Jednostavna pravila:
  - Ako postoje reci u inserted, a ne postoje u deleted => INSERT
  - Ako ne postoje reci u inserted, a postoje u deleted => DELETE
  - Ako postoje i u inserted i u deleted => UPDATE
- Za provjeru postojanja najbolje koristiti operatore `EXISTS` i `NOT EXISTS`

Promijenjeni stupac

- Kod `INSERT` i `DELETE` događaja se mijenjaju kompletni reci
- Kod `UPDATE` naredbe `SET` dio može promijeniti jedan ili više stupaca
- Treba nam način kako saznati koji stupci su pogođeni `SET` dijelom naredbe
- Sistemska funkcija `UPDATE()` kao parametar prima naziv stupca te vraća **TRUE** ili **FALSE**
  - TRUE označava da je stupac naveden u SET dijelu naredbe
  - FALSE označava da stupac nije naveden u SET dijelu naredbe

Naredba `IF - ELSE IF - ELSE` (1/2)

- U T-SQL je moguće uvjetno izvršavati dijelove kôda:

```
DECLARE @Broj float
SET @Broj = RAND()

IF @Broj >= 0 AND @Broj < 0.33 BEGIN
    SELECT 'Prva trećina'
END
ELSE IF @Broj >= 0.33 AND @Broj < 0.67 BEGIN
    SELECT 'Druga trećina'
END
ELSE BEGIN
    SELECT 'Treća trećina'
END
```

Naredba `IF - ELSE IF - ELSE` (2/2)

- Ponaša se kao i ekvivalentna naredba u C++
  - Pronalaskom prvog uvjeta izvršava njegov blok i nastavlja iza cijele naredbe
- Uvjet uz `IF` i `ELSE IF` se mora izračunati u istinu ili laž
  - Ne moramo ga staviti u zagrade
- Ključne riječi `BEGIN` i `END` su ekvivalentne { i } u C++
  - Ne moramo ih pisati ako se blok sastoji od samo jedne naredbe
  - Dobra praksa je da ih pišemo

Primjeri

5. Onemogućite okidač iz prethodnih primjera i promijenite redak u tablici. Ponovno ga omogućite.

6. Dodajte novi okidač na tablicu Student i vežite ga uz sva tri događaja. U okidaču saznajte koji događaj ga je pozvao i tu informaciju upišite u Zapisnik. Napravite umetanje, izmjenu i brisanje.

7. Isključite sve okidače na tablici Student. Dodajte novi okidač i vežite ga uz `UPDATE` događaj. Neka okidač zapiše da se desio događaj samo ako je promijenjen stupac Prezime.

Više okidača na istoj tablici

- Na istoj tablici može biti više okidača vezanih uz isti događaj
- Kad se desi neki događaj, RDBMS će pozvati sve okidače vezane uz njega

- Redoslijed pozivanja:
  - Predefinirano ponašanje RDBMS-a je da poziva okidače redoslijedom kako su kreirani
  - Možemo utjecati na izvođenje tako da definiramo koji okidač želimo da se izvrši **prvi**, a koji **posljednji (na ostale ne možemo utjecati)**:

```
EXEC sp_settriggerorder
 @triggername = 'naziv_okidača',
 @order = 'FIRST | LAST | NONE'
 @stmttype = 'INSERT | UPDATE | DELETE'
```

Sistemska procedura

Napredne postavke okidača

- **Ugniježdeni** (engl. *nested*) okidač je onaj koji je pozvan zbog događaja uzrokovanog aktivnošću drugog okidača
- **Rekurzivn**i (engl. *recursive*) okidač je onaj okidač koji akcijom nad svojom tablicom poziva sam sebe
- Kod obje vrste okidača postoji opasnost beskonačne petlje
  - RDBMS-ovi na razne načine rješavaju taj problem
    - SQL Server dopušta najviše 32 poziva ugniježdenih okidača
    - SQL Server predefinirano ima isključene rekurzivne okidače - to znači da **okidač ne može pozvati sam sebe**
    - Uključivanje i isključivanje rekurzivnih okidača:

```
ALTER DATABASE naziv_baze
SET RECURSIVE_TRIGGERS ON/OFF
```

Primjeri

8. Uklonite sve okidače na tablici Student. Dodajte 4 nova okidača koji u zapisnik ispisuju "Pozdrav iz broja *n*" nakon umetanja retka. Umetnite redak. Posložite redoslijed okidača tako da bude 4, 2, 3, 1. Umetnite redak. Vratite originalni redoslijed. Umetnite redak. Uklonite okidače.

9. Napravite tablice Tbl1 i Tbl2 s proizvoljnim stupcima. Na Tbl1 napravite okidač vezan uz INSERT koji umeće redak u Tbl2 i u Zapisnik. Na Tbl2 napravite okidač vezan uz INSERT koji umeće redak u Tbl1 i u Zapisnik. Što se dogodilo?

10. Na tablicu Student dodajte okidač koji je vezan uz INSERT i koji zapisuju novi redak u istu tablicu i u Zapisnik. Umetnite redak. Uključite rekurzivne okidače. Umetnite redak. Isključite rekurzivne okidače.

INSTEAD OF okidači

```
CREATE/ALTER TRIGGER shema_tablice.naziv_okidača
ON shema_tablice.tablica
INSTEAD OF INSERT, UPDATE, DELETE
AS
niz_naredbi
```

- Izvršavaju niz naredbi UMJESTO pokretačke naredbe
- Prilikom izvršavanja također koriste specijalne tablice *inserted* i *deleted*
- Može biti samo po jedan INSTEAD OF okidač na objektu
- Mogu biti na istom objektu zajedno s AFTER okidačima – AFTER okidač može biti izazvan radnjama iz INSTEAD OF okidača

INSTEAD OF okidači

Primjer 1: Imamo pogled koji pokazuje podatke iz više tablica. Što učiniti ako korisnik pokuša kroz njega upisati nove podatke?

Primjer 2: Imamo pogled koji pokazuje podatke iz više tablica. Što učiniti ako korisnik pokuša kroz njega brisati podatke?

Ograničenja:

- Ako je definiran INSTEAD OF DELETE okidač, nije više moguće koristiti Cascade Delete, isto vrijedi i za INSERT

- Rekurzija nije moguća – INSTEAD OF okidači se izvršavaju samo jednom

DDL okidači

```
CREATE/ALTER TRIGGER naziv_okidača
ON database | all server
FOR događaj
AS
niz_naredbi
```

- *događaj* može biti:
  - Na razini servera:
    CREATE_DATABASE, ALTER_DATABASE, CREATE_LOGIN, ALTER_LOGIN,...
  - Na razini baze:
    CREATE_TABLE, CREATE_PROCEDURE, CREATE_FUNCTION, ALTER_TABLE, ALTER_PROCEDURE, ALTER_FUNCTION,...

DDL okidači

- Za nadgledanje i kontrolu promjena objekata:
  - Npr. tko je i kad pokrenuo naredbu za brisanje ili promjenu objekata
- Za prevenciju mijenjanja strukture baze

Primjer 1: Sprečavanje brisanja i promjene postojećih tablica ili kreiranja novih tablica.