OBLIKOVANJE BAZA PODATAKA

Predavanje 10

Blic

- URL: goo.gl/kyJbqc

SLOŽENI PARAMETRI

ALGEBRA

Uvod

- Postoje dva česta problema koji se javljaju u radu s procedurama i funkcijama u bazama podataka
- **Problem 1:**
  - Ako imamo dvije tablice koje su u odnosu **1:N**, kako možemo jednim pozivom procedure upisati jedan zapis u jednu, te više zapisa u drugu tablicu?
- **Problem 2:**
  - Kako možemo proceduri ili funkciji proslijediti vrijednosti ključeva za retke koje želimo dohvatiti ili obrisati?
- Oba problema mogu se riješiti **višestrukim pozivanjem** jedne ili više procedura/funkcija
  - Možemo li **riješiti problem pomoću jednog poziva?**

ALGEBRA

Rješenje problema u C#

- Kako bismo u programskom jeziku C# riješili te probleme:
  - Problem 1:

```
class Grad {
    public string Naziv;
}
…
Drzava d = new Drzava();
d.Gradovi.Add(new …)
d.Gradovi.Add(new …)
…
Upisi(d);
```

```
class Drzava {
    public string Naziv;
    public List<Grad> Gradovi;
}
```

  - Problem 2:

```
int[] obrisati = new int[] { 21, 24, 36, 97 };
Obrisi(obrisati);
```


ALGEBRA

Rješenje problema u SQL-u

- Do sada smo procedurama ili funkcijama prosljeđivali parametre jednostavnih tipova podataka (`int`, `float`, …)
- Zanima nas kako možemo procedurama ili funkcijama proslijediti složenije parametre
- Četiri metode:
  - Ručna izrada upita u aplikaciji
  - Iterativna metoda
  - XML
  - Korisnički definirani tablični tipovi podataka

ALGEBRA

METODA 1: RUČNA IZRADA UPITA U APLIKACIJI

ALGEBRA

Ručna izrada upita u aplikaciji

- U programskom kôdu se generira SQL upit i prosljeđuje bazi
  - Privlačno, jer je u kôdu lako generirati WHERE dio spajanjem stringova, primjerice:

```
string upit = "DELETE FROM Racun "
upit += "WHERE IDRac = 1 ";
upit += "OR IDRac = 2 ";
upit += "OR IDRac = 3 ";
SqlDataReader podaci = cmd.ExecuteReader(upit);
```

  - Općenito, generiranje SQL upita u kôdu koji bi riješili navedene probleme je relativno jednostavno

ALGEBRA

Problemi s generiranje upita u aplikaciji

- **Najlošija metoda, ne treba je koristiti niti u kakvim uvjetima!**
- Problemi:
  1. Mogućnost ***SQL injection*** napada – u generiranom SQL upitu se mogu naći neočekivane SQL naredbe
     - Moguće ako se pri izradi upita koriste nevalidirane korisnički upisane vrijednosti (primjerice, u TextBox)
     - Rezultati mogu biti katastrofalni po sigurnost i integritet podataka
  2. Limitirana mogućnost ponovnog korištenja plana izvršavanja
     - Samo identični upiti će koristiti generirani plan
  3. Povećan mrežni promet – SQL naredbe se prenose svaki put
- Problemi 1 i 2 se mogu riješiti parametriziranjem upita (ORM)
- Svi problemi su riješeni korištenjem procedura

ALGEBRA

SQL injection demo

- Primjer SQL injection napada

```
1); DROP TABLE  Zaposlenik; CREATE TABLE
MiroWasHere (Poruka nvarchar(150)); INSERT
INTO MiroWasHere VALUES ('Drugi put koristite
procedure, bwahahaha'); SELECT GETDATE(
```


ALGEBRA

METODA 2: ITERATIVNA METODA

ALGEBRA

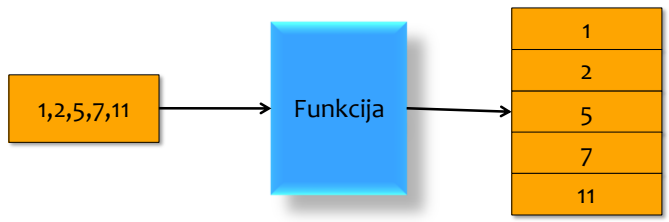
Uvod

- Dobro rješenje za drugi problem – prosljeđivanje više vrijednosti u proceduru
- Ideja:
  - Procedura će primiti jedan parametar, najčešće veliki string
  - U tom stringu ćemo proslijediti više vrijednosti odvojenih nekim separatorom
    - Separator treba pametno odabrati
  - Unutar procedure ćemo proslijeđene vrijednosti izvaditi iz stringa i staviti u tablični oblik
    - Svaki redak u tablici će biti jedna vrijednost iz stringa
    - Obično se za ovu svrhu radi nova funkcija

ALGEBRA

Vađenje vrijednosti iz stringa

- Vrijednosti stižu u proceduru kao string
- Sve operacije u SQL-u radimo na tablicama
- Treba nam **tablična funkcija** koja će uzeti vrijednosti iz stringa i vratiti ih kao retke unutar tablice



- Kod odabira separatora paziti da se odabrani separator ne može pojaviti kao vrijednost

ALGEBRA

Sistemske funkcije za rad sa stringovima

- Najvažnije funkcije smo već upoznali:
  - `LEN(`*`string`*`)`
    - Vraća broj znakova u zadanom stringu
    - `LEN('Ana')` vraća 3
  - `SUBSTRING(`*`string, početak, duljina`*`)`
    - Vraća dio zadanog stringa
    - `SUBSTRING('Miroslav', 1, 4)` vraća 'Miro'
  - `CHARINDEX(`*`što, gdje [, početak`*`])`
    - Vraća prvu poziciju stringa *što* u stringu *gdje* veću od *početak*
    - `CHARINDEX('a', 'Katarina')` vraća 2
    - `CHARINDEX('a', 'Katarina', 5)` vraća 8
    - `CHARINDEX('f', 'Katarina', 1)` vraća 0

ALGEBRA

Prednosti i nedostaci ove metode

- Prednost:
  - Odlazak iz aplikacije u bazu je relativno skupa operacija
    - Bolje je u jednom odlasku obaviti što više posla
    - Ova metoda omogućava dohvaćanje ili brisanje više vrijednosti u samo jednom odlasku u bazu – dobro za performanse
  - Dostupna na svim RDBMS-ovima
- Nedostaci:
  - SQL je optimiziran za rad s tablicama
  - Proces "rezanja" ulaznog stringa i njegovo pretvaranje u tablicu nije najoptimalniji – potencijalno loše po performanse
- Zbog svoje jednostavnosti ova metoda je dobro rješenje dok god je duljina ulaznog stringa relativno mala

Primjeri

1. Analizirajte tabličnu funkciju za razdvajanje cijelih brojeva odvojenih zarezima.
2. Napišite proceduru koja dohvaća ID, naziv i boju proizvoda prema proslijeđenim ID-evima. Koristite iterativnu metodu.
3. Može li funkcija iz 1 raditi sa string vrijednostima? Ako ne, promijenite ju tako da može.
4. Promijenite proceduru iz 2 tako da dohvaća prema proslijeđenim nazivima proizvoda.
5. Koji problem postoji s funkcijom? Promijenite ju tako da uvedete separator kao dodatni parametar.

METODA 3: XML

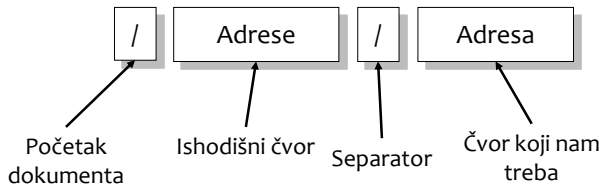
Uvod

- Vrlo dobro i često rješenje i za prvi i za drugi problem
- Osnova ove metode je **tip podataka `xml`**
  - Čuva u sebi ispravni XML dokument ili fragment
- Primjer:

```
DECLARE @Adrese xml
SET @Adrese = '
  <Adrese>
    <Adresa Grad="Zagreb" Pbr="10000">Sunčana 17</Adresa>
    <Adresa Grad="Varaždin" Pbr="42000">Široka 25</Adresa>
    <Adresa Grad="Zabok" Pbr="49210">Trg slobode 4</Adresa>
  </Adrese>'
```


Operacije nad xml tipom podataka

- xml sadržava tekstualnu prezentaciju podataka, a nama je za SQL potrebna relacijska prezentacija, tj. tablica
- SQL Server sadrži metode definirane na tipu podataka xml koje služe za dobivanje tablične prezentacije: nodes i value
- Za dohvaćanje važno znati osnovnu XQuery sintaksu:



ALGEBRA

Plan akcije

- Plan je sljedeći:
  1. Procedura će primiti jedan XML parametar
  2. Napisat ćemo SELECT koji će XML parametar pretvoriti u tablicu
     a. Metoda nodes() će XML izrezati u retke
     b. Metode value() će odabrati stupce
  3. Tablicu ćemo dalje koristiti za rješavanje poslovnog problema

ALGEBRA

Metoda nodes()

- Metoda nodes()
  - Koristimo je u FROM dijelu
  - Iz XML-a vraća tablicu u čijim recima se nalaze dijelovi XML-a
    - Tablica **uvijek sadrži jedan stupac**
    - Moramo toj tablici i stupcu dati nazive pomoću aliasa
  - Primjerice:

```
FROM @Adrese.nodes('/Adrese/Adresa') AS tbl(stupac)
```

  - Dobivamo tablicu **tbl** sa stupcem **stupac** i 3 retka:

stupac
<Adresa Grad="Zagreb" Pbr="10000">Sunčana 17</Adresa>
<Adresa Grad="Varaždin" Pbr="42000">Široka 25</Adresa>
<Adresa Grad="Zabok" Pbr="49210">Trg slobode 4</Adresa>

ALGEBRA

Metoda value()

- Metoda value()
  - Služi **uzimanju SQL vrijednosti iz XML-a** i koristimo je u SELECT listi za definiranje stupaca
  - Metoda prima dva string parametra:
    - Prvi parametar govori koji dio XML-a želimo uzeti
      - Točka '.' označava vrijednost trenutnog elementa
      - Dvije točke '..' označavaju roditeljski element
      - Atribut dohvaćamo navođenjem '@' i njegovog naziva
    - Drugi parametar je tip podataka u koji želimo pretvoriti vrijednost
  - Primjerice:

```
SELECT
    tbl.stupac.value('@Grad', 'nvarchar(50)')
    tbl.stupac.value('.', 'nvarchar(50)')
```


ALGEBRA

Rješenje problema

- **Problem 1:** da bismo unijeli državu i njene gradove možemo koristiti sljedeći XML dokument:

```
<Drzava>
  <Naziv>Velika Britanija</Naziv>
  <Gradovi>
      <Grad Naziv="Manchester" />
      <Grad Naziv="Liverpool" />
  </Gradovi>
</Drzava>
```

- **Problem 2:** da bismo proslijedili gravode koje ćemo brisati možemo koristiti sljedeći XML dokument:

```
<Brisati>
  <ID>1</ID>
  <ID>5</ID>
</Brisati>
```


Primjeri

6. Zadani XML pretvorite u tablicu koja sadržava stupce Grad, PostanskiBroj i Ulica.
7. Napišite proceduru koja u XML dokumentu dobiva ID-eve proizvoda i za te proizvode dohvaća ID, naziv i boju. Pozovite proceduru.
8. Napravite tablice Racun i Stavka i pomoću XML-a riješite problem umetanja računa s više stavki.

METODA 4: KORISNIČKI DEFINIRANI TABLIČNI TIPOVI

ALGEBRA

Definiranje korisničkih tabličnih tipova

- SQL Server 2008 (i noviji) omogućava korištenje **korisnički definiranih tabličnih tipova** (engl. *user-defined table types*)
- Predstavljaju nacrt za izradu tabličnih varijabli
- Tablična varijabla predstavlja tablicu u memoriji
  - Ekvivalent kolekcijama u C#-u
- Tablični tip je prije korištenja potrebno definirati, primjerice:

```
CREATE TYPE Osoba AS TABLE
(
  Ime nvarchar(50),
  Prezime nvarchar(50)
)
```

- Tip uklanjamo s DROP TYPE naredbom

ALGEBRA

Tablične varijable

- Nakon definiranja tabličnog tipa, na osnovu njega radimo **tablične varijable** koje koristimo slično kao i tablice
- Primjer korištenja:

```
DECLARE @Ljudi Osoba
```

Korisnički definirani tablični tip

Tablična varijabla

```
INSERT INTO @Ljudi VALUES ('Miro', 'Mirić')
INSERT INTO @Ljudi VALUES ('Ana', 'Anić')

SELECT * FROM @Ljudi
```


Tablične varijable i procedure

- Tablične varijable se mogu prosljeđivati kao parametri procedurama, što rješava probleme 1 i 2
- Definicija procedure:

```
CREATE PROC dbo.UmetniLjude
     @Ljudi Osoba READONLY
AS
```

Obavezno!

- Pozivanje procedure:

```
DECLARE @Ljudi Osoba

INSERT INTO @Ljudi VALUES ('Miro', 'Mirić')
INSERT INTO @Ljudi VALUES ('Ana', 'Anić')

EXECUTE dbo.UmetniLjude @Ljudi
```


Primjeri

9. Definirajte tablični tip za ID-eve proizvoda. Napišite proceduru koja prima tabličnu varijablu tog tipa i koja vraća ID, naziv i boju zadanih proizvoda.
10. Napravite tablicu Osoba. Definirajte tablični tip za upis osoba. Napišite proceduru koja jednim pozivom omogućava unos više osoba.

ALGEBRA