OBLIKOVANJE BAZA PODATAKA

Predavanje 09

MI – komentar rezultata

Broj studenata		Udio
121	upisanih studenata	
38	ponavljača	31%
104	prijavljenih na MI	86%
30	nisu se pojavili	29%
74	upisali bodove	61%

Prolaznost na pojedinom ishodu	I1	I2	I3
Broj studenata	44	29	33
Udio	59%	39%	45%

Ukupna prolaznost	Prošli sve ishode	Prošli dva ishoda	Prošli jedan ishod	Pali sve ishode
Broj studenata	22	9	22	21
Udio	30%	12%	30%	28%

Složeni parametri

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?**

Rješenje problema

- Do sada smo procedurama ili funkcijama prosljeđivali parametre jednostavnih tipova podataka (`int`, `nvarchar()`, ...)

- Zanima nas kako možemo procedurama ili funkcijama proslijediti složenije parametre

- Pet metoda:
  1. Ručna izrada upita u aplikaciji
  2. Iterativna metoda
  3. XML
  4. Korisnički definirani tablični tipovi podataka
  5. JSON

Metoda 1:
Ručna izrada upita u aplikaciji

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

Problemi s generiranjem upita u aplikaciji

- **Najlošija metoda, ne treba je koristiti ni 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 - *Object-Relational Mapping library*)
- Svi problemi su riješeni korištenjem procedura

SQL injection demo

```
create table Zaposlenik
(
    IDZaposlenik int primary key identity,
    Ime nvarchar(100),
    Prezime nvarchar(100)
)

insert into Zaposlenik values ('a','a')
insert into Zaposlenik values ('b','b')
insert into Zaposlenik values ('c','c')
insert into Zaposlenik values ('d','d')
select * from zaposlenik
```

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');
SHUTDOWN WITH NOWAIT; SELECT GETDATE(
```

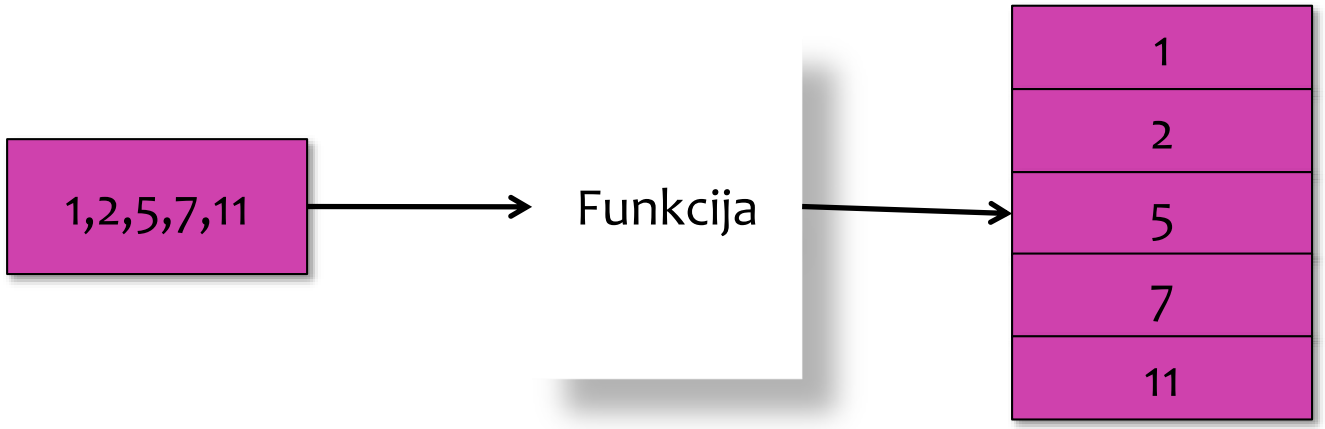
Metoda 2: Iterativna metoda

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

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

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

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

Extensible Markup Language (XML)

- **GML** (*Generalized Markup Language*) – smislio IBM 60-ih godina 20. stoljeća za označavanje tehničke dokumentacije. Nedostatak: nije standardiziran

- **SGML** (*Standard Generalized Markup Language*) – 80-ih godina 20. stoljeća ANSI standard. Nedostatak: opširan

- **HTML** (*HyperText Markup Language*) – iz SGML skupa oznaka odabran manji skup oznaka za oblikovanje dokumenata (CERN). Nedostatak: mali skup zadanih oznaka, s vremenom postaje više orijentiran na formatiranje sadržaja

- **XML** (*Extensible Markup Language*)
  - *World Wide Web Consortium* je 1998. objavio prvu verziju XML preporuka
  - Jezik za označavanje podataka sličan HTML-u, ali bez predefiniranih oznaka (tagova)
  - Vrlo dobro i često rješenje i za prvi i za drugi problem
  - Osnova ove metode je **tip podataka `xml`**

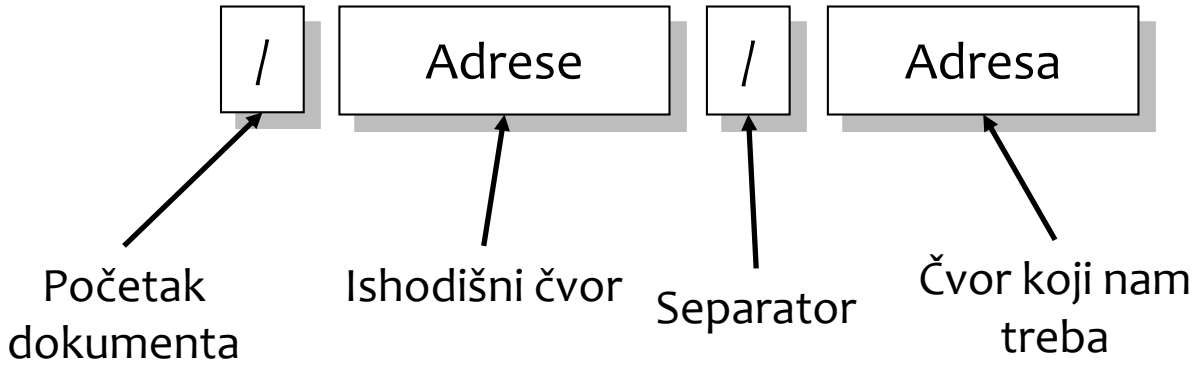
Extensible Markup Language (XML)

- 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` sadrži 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:



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

Metoda nodes()

- Koristimo je u `FROM` dijelu
- Iz XML-a vraća tablicu u čijim retcima 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>

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)')
```

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 gradove koje ćemo brisati možemo koristiti sljedeći XML dokument:

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

Pretvorba u XML oblik

- Iz tabličnog oblika kreira se XML oblik
- Primjer:

```
select k.IDKupac as [@IDKupac], k.Ime, k.Prezime, k.Email as [Emailovi/Email]
from kupac as k
for xml path('Kupac'), root('Kupci')
```

```
<Kupci>
  <Kupac IDKupac="1">
    <Ime>Gustavo</Ime>
    <Prezime>Achong</Prezime>
    <Emailovi>
      <Email>gustavo0@adventure-works.com</Email>
    </Emailovi>
  </Kupac>
</Kupci>
```

Primjeri

6. Zadani XML pretvorite u tablicu koja sadržava stupce Grad, PostanskiBroj i Ulica.

```
<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>
```

Primjeri

7. Napišite proceduru koja u XML dokumentu dobiva ID-eve proizvoda i za te proizvode dohvaća ID, naziv i boju. Pozovite proceduru.

8. Pomoću XML-a riješite problem umetanja države s više gradova.

9. Prikažite vrijednosti iz tablice Proizvod u sljedećem XML obliku:

```
<Proizvodi>
    <Proizvod IDProizvod="318">
        <Naziv>ML Crankarm</naziv>
         <Brojproizvoda>CA-6738</brojproizvoda>
        <Boje>
          <Boja>Crna</Boja>
        </Boje>
    </Proizvod>
</Proizvodi>
```

Metoda 4:
Korisnički definirani tablični tipovi

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
- Tablični tip je prije korištenja potrebno definirati, primjerice:

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

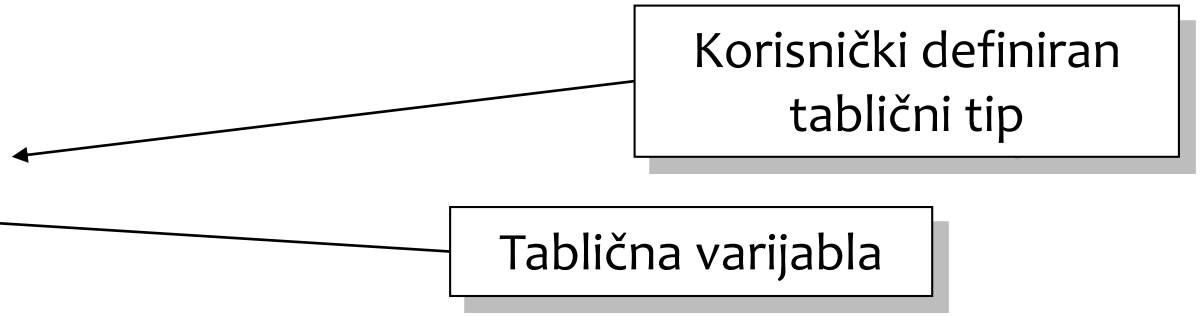
- Tip uklanjamo s `DROP TYPE` naredbom

Tablične varijable

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

- Primjer korištenja:

```
DECLARE @Ljudi Osoba
```



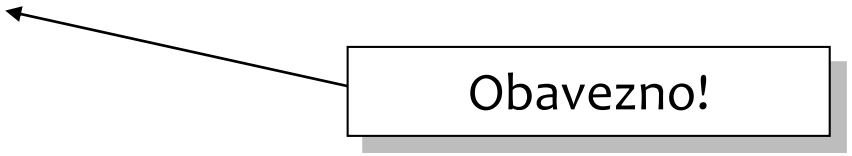

```
INSERT INTO @Ljudi VALUES ('Miro', 'Mirić')
INSERT INTO @Ljudi VALUES ('Ana', 'Anić')

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
```



- Pozivanje procedure:

```
DECLARE @Ljudi Osoba

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

EXEC UmetniLjude @Ljudi
```

Primjeri

10. Definirajte tablični tip za ID-eve proizvoda. Napišite proceduru koja prima tabličnu varijablu tog tipa i koja vraća ID, naziv i boju zadanih proizvoda.

11. Napravite tablicu Osoba. Definirajte tablični tip za upis osoba. Napišite proceduru koja jednim pozivom omogućava unos više osoba.

Metoda 5: JSON

JavaScript Object Notation (JSON)

- Pojavio se početkom stoljeća
- Format zapisivanja podataka, neovisan o razvojnim jezicima
- 2013. godine *European Computer Manufacturers Association (ECMA)* - prvi standard
- Vrlo dobro i često rješenje i za prvi i za drugi problem

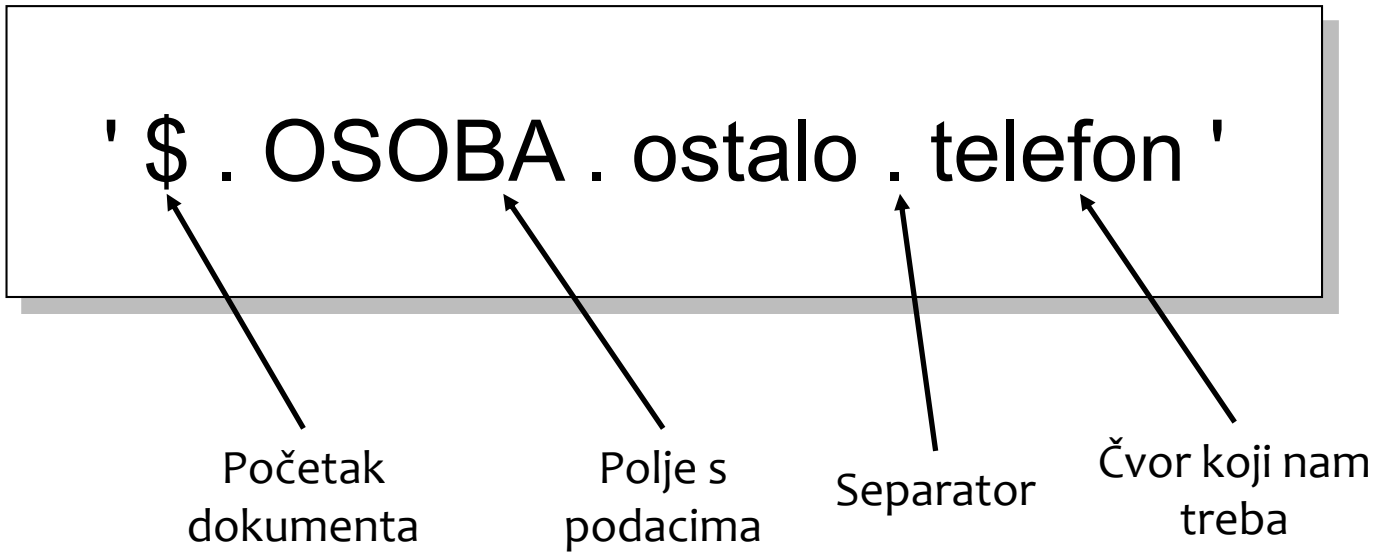
JavaScript Object Notation (JSON)

- Primjer:

```
'{
    "OSOBE":
    [
        {"OSOBA":
            {"idosoba": 1,"ime": "Pero", "prezime": "Perić"}},
        {"OSOBA":
            {"idosoba": 2, "ime": "Iva", "prezime": "Ivić",   "ostalo":
                {"email": "iva.ivic@algebra.hr", "telefon": "091 222 3333"}}}
    ]
}'
```

Operacije nad JSON tipom podataka

- JSON sadrži tekstualnu prezentaciju podataka, a nama je za SQL potrebna relacijska prezentacija, tj. tablica
- Funkcija za dobivanje tablične prezentacije: OPENJSON
- Za dohvaćanje važno znati način pisanja putanje do podatka:



Kako koristiti JSON podatak

- Plan je sljedeći:

1. Procedura će primiti jedan JSON parametar
2. Napisat ćemo SELECT koji će JSON parametar pretvoriti u tablicu (funkcijom **OPENJSON)**
3. Tablicu ćemo dalje koristiti za rješavanje poslovnog problema

Pretvorba u tablični oblik

- Koristimo je u FROM dijelu
- Iz JSON oblika vraća tablicu s podacima iz JSON-a

- Primjer:

```
SELECT ime, prezime, email, telefon
FROM OPENJSON (@json, '$.OSOBE')
WITH
(
    Ime         nvarchar(50)  '$.OSOBA.ime',
    Prezime     nvarchar(50)  '$.OSOBA.prezime',
    Email       nvarchar(50)  '$.OSOBA.ostalo.email',
    Telefon     nvarchar(50)  '$.OSOBA.ostalo.telefon'
)
```

Pretvorba u JSON oblik

- Iz tabličnog oblika kreira se JSON oblik
- Primjer:

```
SELECT k.IDKupac as [OSOBA.idkupac],
    k.Ime as [OSOBA.ime],
    k.prezime as [OSOBA.prezime],
    email as [OSOBA.ostalo.email],
    telefon as [OSOBA.ostalo.telefon]
FROM Kupac as k
FOR JSON PATH, ROOT('OSOBE')
```

```
{"OSOBE":
    [{"OSOBA":
        {"idkupac":1,"ime":"Gustavo","prezime":"Achong",
            "ostalo":{"email":"gustavo0@adventure-works.com","telefon":"398-5550132"}}
        }]}
```

Primjeri

12. Napisati proceduru koja prima JSON dokument oblika:

```
'{"OSOBE":
[
    {"OSOBA":
        {"ime": "Pero", "prezime": "Perić"}},
    {"OSOBA":
        {"ime": "Iva", "prezime": "Ivić"}}
]}'
```

Neka procedura vrati tablicu s imenima i prezimenima osoba.
Pozvati proceduru s gornjim parametrom.

Primjeri

13. Napisati proceduru koja prima JSON dokument oblika:

```
'{ "OSOBE":
  [{ "OSOBA":
      { "idosoba": 2,  "ime": "Iva",  "prezime": "Ivić",  "ostalo":
          {"email": "iva.ivic@algebra.hr", "telefon": "091 222 3333" }}
      }
]}'
```

Neka procedura vrati tablicu s podacima id, ime, prezime i telefonski broj osoba.
Pozvati proceduru s gornjim parametrom.

Primjeri

14. Napišite upit kojim ćete s podacima IDKupac, Ime, Prezime, Email i Telefon iz tablice Kupac kreirati JSON dokument oblika kako je prikazano niže, za sve kupce koji imaju IDKupac<11:

```
{"OSOBE":
    [{"OSOBA":
        {"idkupac":1,"ime":"Gustavo","prezime":"Achong",
            "ostalo":{"email":"gustavo0@adventure-works.com","telefon":"398-550132"}}}
    ]
}
```