SQL Procedure

Trigeri, Funkcije i Procedure
- Sličnosti -

- To su objekti baze podataka koji sadrže kod koji se izvršava kao zasebna celina
- Sličnii su metodama, subrutinama ili modulima u drugim programskim jezicima
- Obično se pišu u T-SQL ali mogu da se napišu u bilo kom NET jeziku(VBA ili C#)

- Razlike -

	Trigeri	Funkcije	Store Procedure
Promena podataka	da	ne	da
Vraćanje vrednosti	nikad	uvek	ponekad
Način pozivanja	događaj	u iskazu	exec

Trigeri, Funkcije i Procedure
- Razlike -

	Trigeri	Funkcije	Store Procedure
Promena podataka	da	ne	da
Vraćanje vrednosti	nikad	uvek	ponekad
Način pozivanja	događaj	u iskazu	exec

- Trigerima i Procedurama je dozvoljeno da menjaju podatke u bazi što obično i rade dok funkcijama to nije dozvoljeno
- Suština funkcija je da vraćaju vrednost koja može biti skalarna(jedna vrednost) ili tabelu
- Trigeri nikad ne vraćaju vrednost jer oni manipulišu sa podacima onog trenutka kada se desi događaj.
- Procedure obično imaju zadatak da promene vrednost u tabeli ali mogu i da vrate vrednost obično u formi 1 ili 0 u zavisnosti da li je zadatak uređen ili nije

Trigeri, Funkcije i Procedure
- Razlike -

	Trigeri	Funkcije	Store Procedure
Promena podataka	da	ne	da
Vraćanje vrednosti	nikad	uvek	ponekad
Način pozivanja	događaj	u iskazu	exec

- Najveća razlika je u načinu pozivanja.
  - Procedure se pozivaju komandom exec
  - Funkcija je deo Select, Delete ili Update iskaza
    - Ukoliko vraća skalarnu vrednost poziva se u Select, Where ili Order segmentu
    - Ukoliko vraća tabelu poziva se u From segmentu SQL iskaza
  - Ne postoji komanda kojom se triger pokreće, već se on pokreće kao reakcija na neki događaj(Update, Insert ili Delete)

Stored Procedure
- Karakteristike -

- **Osnovna prednost je što pojednostavljuju administraciju**
  - Tipično za velike projekte gde više developera ima pristup bazi
  - Smanjuje se verovatnoća greške i konflikt prilikom unosa podataka i problemi prilikom izmene šeme baze podataka
- **Kod u Store proceduri se čuva sa podacima**
  - Efikasnije rešenje od korišćenja skripti
- **Poboljšana bezbednost**
  - Možemo da obezbedimo različit pristup različitim korisnicima preko procedura
  - Korisniku koji nema mogućnost brisanja zabranićemo pristup delete proceduri
- **Poboljšanje performansi**

Stored Procedure
- Koraci u izvršavanju SQL upita -

- Kada pokrenemo klasičan SQL upit DBMS izvršava nekoliko koraka dok ne izvrši upit
  - Parsiranje (proverava se ispravnost sintakse)
  - Promena imena objekata u konzistentan format (jedan objekat(tabela) može da se pozove na više načina dok mašina mora da standardizuje ime)
  - Optimizacija – DBMS odlučuje koju data strukturu da koristi, numeraciju tabela, kojim redosledom da se čitaju tabele i podaci, indeksiranje i druge tehnike optimizacije.
  - Plan izvršenja (execution plan) se kreira i upit se izvršava
- Kada ponovo pokrenemo isti upit, DBMS prolazi kroz sva četri koraka kao da upit pre toga nikad nije bio izvršen

Stored Procedure
- Poboljšanje performansi -

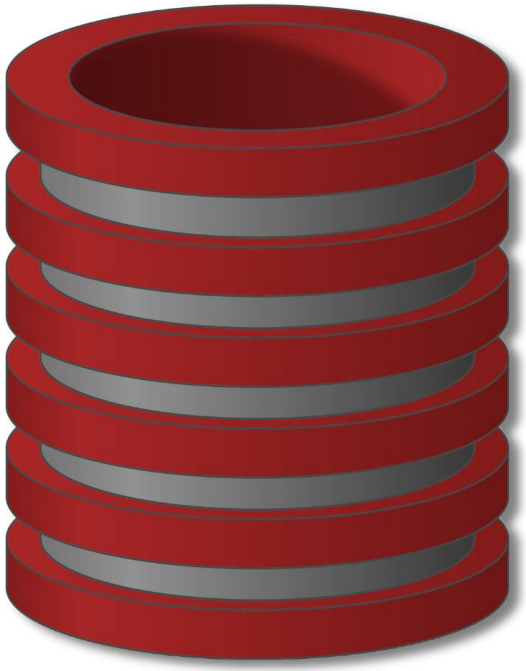
- **Store Procedure zaobilaze određene korake prilikom izvršenja klasičnih upita.**
  - Prvo izvršenje prolazi kroz sve korake
  - Nakon toga DBMS snimi plan izvršenja procedure
  - Za svako naredno pokretanje zaobilaze se prva tri koraka i odmah se prelazi na plan izvršenja koji je od ranije sačuvan
  - **Najviše vremena DBMS troši na optimizaciji (veličina tabela, indeksiranje,...)**

Funkcije
- Karakteristike -

- Većina benefita slična procedurama
  - Pojednostavljuju administraciju
  - Kod se čuva sa podacima
  - Poboljšanje performansi
  - Mogu da uključe složene i kompleksne zadatke (formatiranje podataka za svaku zemlju posebno)

Trigeri
- Karakteristike -

- Koriste se da prate promene
  - Možemo da beležimo svaku promenu u tabeli (istorija tabele)
  - Mogu da se koriste da obezbede integritet podataka
    - Kontrolišu dozvoljene podatke
    - Sprečavaju greške

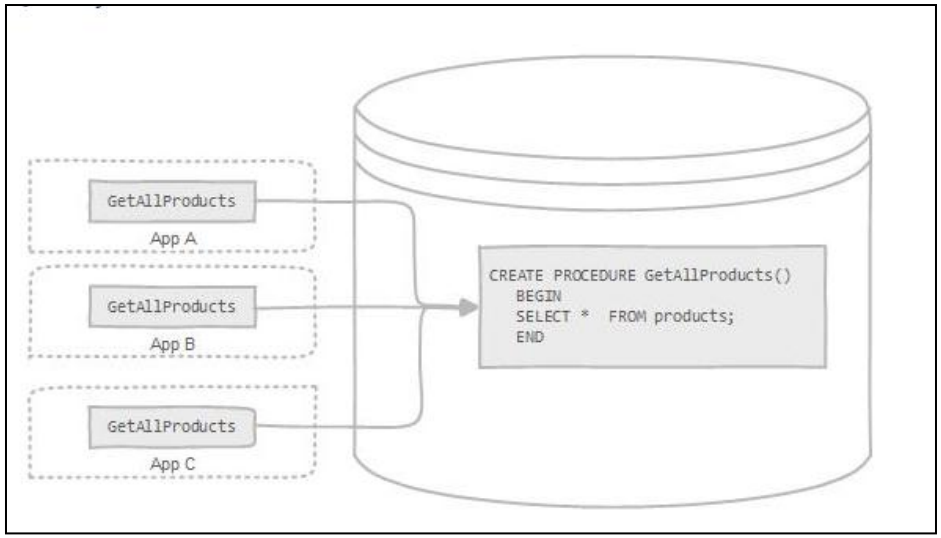


MySQL

Procedure

Definicija Store Procedure

- Store procedura se sastoji iz SQL iskaza koji se čuvaju unutar kataloga baze podataka
- Store procedura se može pozvati od strane trigera, drugih store procedura ili aplikacija kao što su Java, PHP itd.



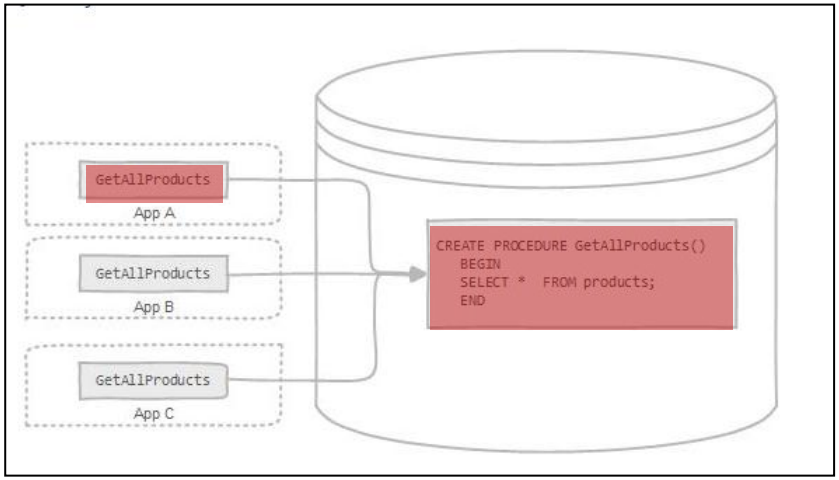
- Store procedura koja poziva samu sebe zove se rekurzivna store procedura
- Većina DBMS podržava rekurzivne strore procedure, međutim MySql ih ne podržava tj. treba proveriti dokumentaciju za verziju MySql koja se koristi.

MySQL Store Procedure

- MySQL je trenutno najpopularnija *open source* RDBMS .
- U početku Store procedure, trigeri i funkcije nisu bile podržane od strane MySQL-a.
- Od MySql verzije 5.0, ovi objekti su dodati.

MySQL Store Procedure Prednosti

- Store Procedure pomažu u poboljšanju performansi aplikacije
- Kada se jednom kreira ,Store Procedura je kompajlirana i čuva se u bazi
- Međutim, MySql implementira store procedure neznatno drugačije
  - MySql store procedure se kompajliraju na zahtev
  - Nakom kompajliranja se smeštaju u keš i MySql održava keš store procedure za svaku konekciju pojedinačno.
  - Ukoliko aplikacija koristi proceduru više puta unutar iste konekcije, kompajlirana verzija procedure se koristi u suprotnom store procedura se ponaša kao klasičan SQL upit.



MySQL Store Procedure Prednosti

- Procedure pomažu u smanjenju saobraćaja između aplikacija i servera baze podataka jer umesto da se šalju više dugačkih SQL iskaza , aplikacija šalje samo ime i parametre store procedure.
- Store procedure su transparetne i dostupne bilo kojoj aplikaciji tako da developeri ne moraju da razvijaju funkcije koje su već podržane u store proceduri
- Store procedure su secure jer db administrator može da dodeli odgovarajuće permisije da bi aplikacija pristupila proceduri bez dodatnih permisija za pristup tabelama.

MySQL Store Procedure Nedostaci

- Ukoliko se koriste dosta, iskorišćenost memorije postaje prilična jer se za svaku konekciju koja poziva tu proceduru čuva kompajlirana store procedura u lokalnom kešu.
- Posebno, ukoliko se u proceduri pretera sa korišćenjem logičkih operatora, iskorišćenost CPU-a će značajno porasti jer serveri baze podataka nisu dobro dizajnirani za logičke operacije.
- Store procedure se prilično teško debug-uju, samo nekoliko dbms dozvoljava debug store procedure(MySql nema alat za debug store procedure)
- Kreiranje i održavanje store procedura zahteva određenu veštinu koju ne poseduju svi developeri.

Jednostavna MySQL Store Procedura

```
DELIMITER //
USE ImeBaze //
CREATE PROCEDURE SviProizvodi()
  BEGIN
  SELECT *  FROM proizvodi;
  END //
DELIMITER ;
```

- DELIMITER // komanda menja standardni delimiter a to je (;)
- Promenili smo delimiter jer želimo da proceduru prosledimo serveru kao jedan segment a ne da MySql svaki iskaz tretira zasebno jer se svaki select izkaz završava (;)
- Delimiter // koristili smo posle END komande da bi smo ukazali na kraj procedure a zatim smo Delimiter vratili na default (;)
- Oblast između BEGIN i END zove se telo store procedure i tu se nalaze SQL iskazi tj. logika procedure.

Pozivanje MySQL Store Procedura

```
CALL SviProizvodi();
```

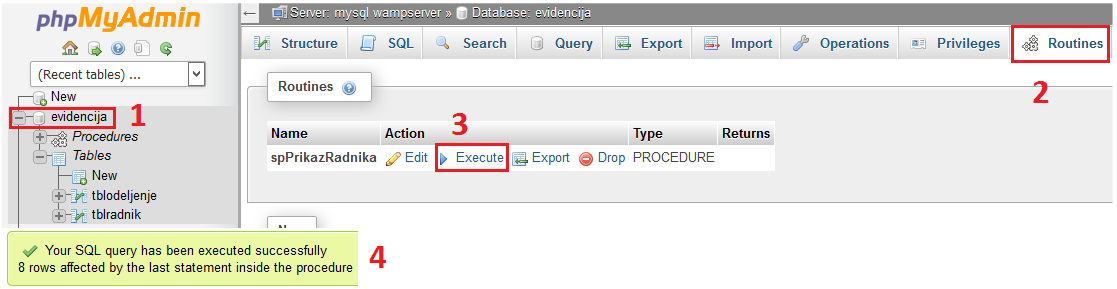
Kreiranje procedure u MySQL-u

- Sledeći kod predstavlja postupak kreiranja procedure u MySQL-u:

Run SQL query/queries on database evidencija:

```
1  DELIMITER //
2   CREATE PROCEDURE spPrikazRadnika()
3     BEGIN
4     SELECT ime, pol, plata  FROM `tblradnik`;
5     END //
6   DELIMITER ;
```

- Kod SQL-a nisu potrebni **DELIMITERI**, dok kod MySQL-a su obavezni.
- **spPrikazRadnika()** označava naziv korisničke procedure.



1. Selektujemo bazu **Evidencija**
2. Izaberemo **Routines**
3. U odseku **Routines** nalaze se sve procedure i funkcije, u našoj listi vidimo našu proceduru **spPrikazRadnika**, biramo **Execute**
4. Prikaz izvršene procedure kao i SQL sintaksa iz komandnog interfejsa.

MySQL Store Procedure Promenjive

- Promenjiva je imenovan objekat čija vrednost može da se promeni tokom izvršenja store procedure.
- Promenjive u store procedurama se obično koriste da pamte neposredne rezlutate.
- Ove promenjive su lokalne za proceduru
- Promenjiva mora prvo da se deklariše da bi mogla da se koristi.

```
DECLARE ime_promenjive tip_podataka(veličina) DEFAULT vrednost;
```

- Kada se promenjiva deklariše njena inicijalna vrednost je NULL
- Promenjivoj može da se zada Default vrednost preko DEFAULT komande

```
DECLARE ukupna_prodaja INT DEFAULT 0;
```

```
DECLARE x, y INT DEFAULT 0;
```

Dodela Vrednosti Promenjivoj

```
DECLARE ukupna_prodaja INT DEFAULT 0;
SET ukupna_prodaja = 10;
```

- Osim SET iskaza, može da se koristi SELECT INTO iskaz da se rezlutat upita koji vraća skalarnu vrednost upiše u promenjivu.

```
DECLARE ukupna_prodaja INT DEFAULT 0;

SELECT COUNT(*)
FROM proizvodi;
```

Opseg Promenjive

- Svaka promenjiva ima svoj opseg delovanja(lifetime).
- Ukoliko je promenjiva deklarisana unutar store procedure, ona više neće biti dostupna kada se napusti END iskaz u proceduri
- Ukoliko proceduru definišemo u BEGIN END bloku, ona više neće biti dostupna kada se izađe iz tog bloka.
- Promenjiva koja počinje sa @ postaje promenjiva sesije.
- Ona je dostupna sve dok se sesija ne završi