

Napredne baze podataka

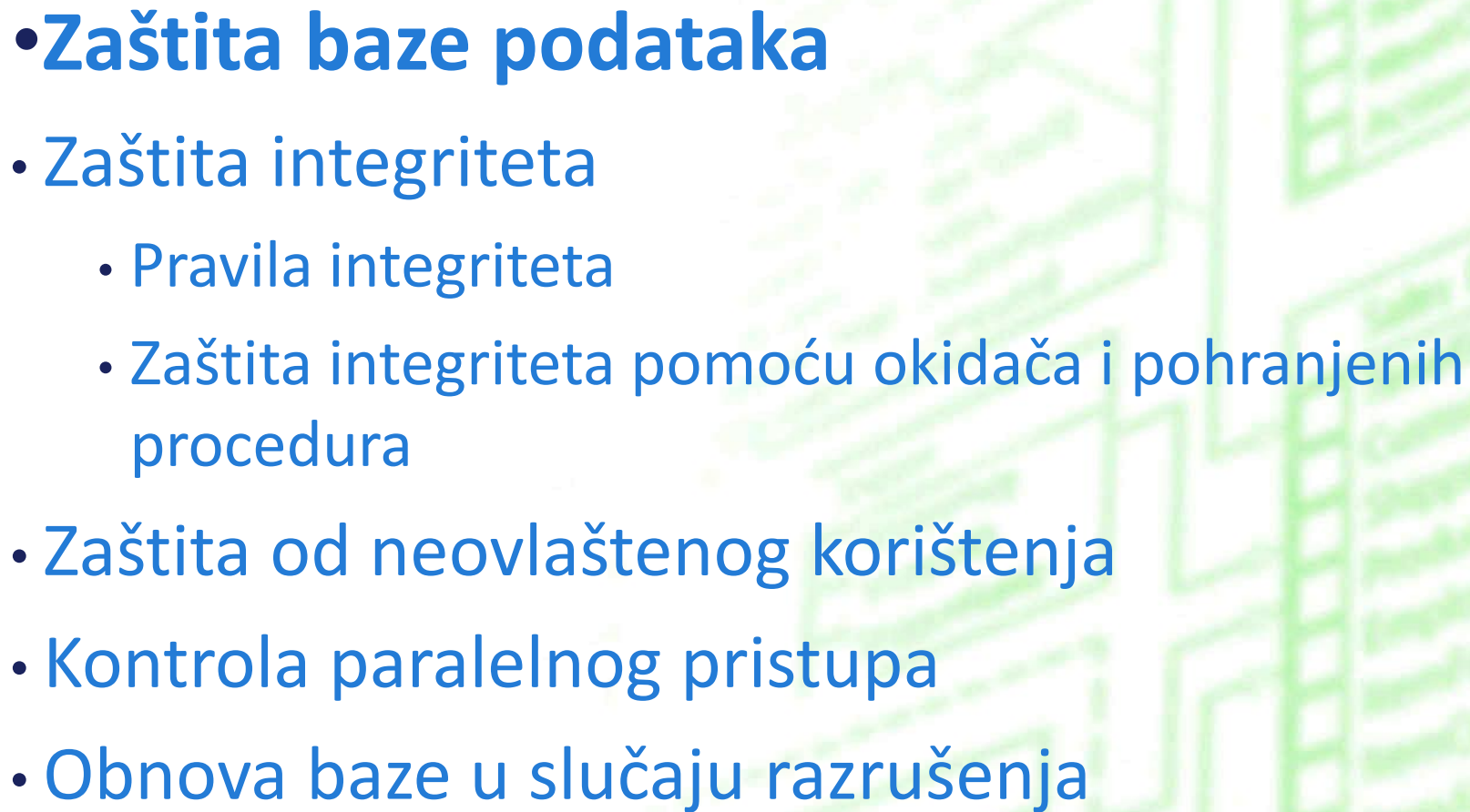
Okidači



• Sadržaj predavanja

- Zaštita baze podataka
- Integritet baze podataka
- Pravila integriteta
- Okidači

- **Pohranjeni zadaci**
- Automatizirane radnje pohranjene u samu bazu podataka
- Dijelev se na:
 - Pohranjene rutine (stored routines)
 - Pohranjene funkcije (stored functions)
 - Ubacuju se u SQL naredbe
 - Pohranjene procedure (stored subroutines)
 - Pozivaju se naredbom CALL,
 - Za razliku od funkcija, mogu vratiti više rezultata
 - **Okidači (trigger)**
 - Događaji (events)

- 
- **Zaštita baze podataka**
 - Zaštita integriteta
 - Pravila integriteta
 - Zaštita integriteta pomoću okidača i pohranjenih procedura
 - Zaštita od neovlaštenog korištenja
 - Kontrola paralelnog pristupa
 - Obnova baze u slučaju razrušenja

• Pravila integriteta

- **Pojam integriteta baze podataka** odnosi se na ispravnost i istinitost podataka sadržanih u bazi
- Neispravni ili netočni podaci mogu biti posljedica:
 - slučajne pogreške kod unosa ili ažuriranja
 - pogreške programera
 - pogreške sustava
- Integritet baze podataka može biti narušen i zbog posljedica diverzije ili sabotaze, međutim o tome brine poseban dio DBMS (SUBP) koji je zadužen za sigurnost baze podataka
- Opća pravila integriteta
 - Pravilo entitetskog integriteta
 - Pravilo referencijskog integriteta
- Korisnička pravila integriteta
 - Pravilo domenskog integriteta
 - Pravilo odnosnog integriteta

•Entitetski integritet

- *(Codd, 1970) - Vrijednost primarnog ključa kao cjeline ne smije biti jednaka NULL vrijednosti.*
- *Ako je primarni ključ relacije složen, ni jedna njegova komponenta ne smije poprimiti NULL vrijednost.*

Primjer:

- Matbr, jmbg, sifra ni u jednom zapisu ne smiju poprimiti NULL vrijednost

•Referencijski integritet

- *Ako u relacijskoj shemi R postoji strani ključ koji odgovara primarnom ključu rel. sheme S, tada svaka vrijednost stranog ključa u relaciji $r(R)$ mora biti:*
 - *ili jednaka vrijednosti primarnog ključa neke n-torke iz relacije $s(S)$*
 - *ili jednaka NULL vrijednosti*

relacija r		
Sifra	prezime	pbr
323	Čupić	10000
13232	Mikić	NULL
133	Dragunja	20000

relacija s	
pbr	grad
10000	Zagreb
21000	Split

- U relaciji r nalazi se upisan poštanski broj 20000 za koji ne postoji zapis u relaciji s -> **narušen referencijski integritet!**
- Postoje slučajevi kada strani ključ iz $r(R)$ ne smije biti jednak NULL vrijednosti. To vrijedi za slučaj kad se pravila referencijskog integriteta sukobe s pravilom entitetskog integriteta

•Integritet (korisnička pravila)

- Domenski integritet
 - *Definira domenu atributa - specificira skup vrijednosti koje atribut smije poprimiti*
- Odnosni integritet
 - *Određuju se dozvoljeni odnosi među pojedinim atributima*

Primjer:

- DJELATNIK = {Sifra, Prezime, Starost, Staz } može se definirati:
 - Domenski integritet za atribut Starost - domena je skup cijelih brojeva iz intervala 16 do 80
 - Odnosni integritet između atributa Staz i Starost, npr. $\text{Starost} \geq \text{Staz} + 16$ (Općenito staž i starost nisu dobri atributi jer su vremenski promjenjivi)

• Implementacija pravila integriteta

- Potrebno je definirati:

1. pod kojim se uvjetima definitivno odbija obavljanje operacije koja bi narušila pravila integriteta
2. pod kojim se uvjetima obavlja operacija uz obavljanje nekih kompenzacijskih operacija
3. pravila entitetskog integriteta nužno moraju biti zadovoljena - ne smije biti nikakvog odstupanja

- *Referencijski* integritet za kritične operacije, npr. operaciju *brisanja*, dozvoljava sljedeće strategije:

- i. ciljna n-torka ne može se obrisati ako u bazi postoje odgovarajuće pozivajuće n-torke
- ii. uz brisanje ciljne n-torke treba izvesti brisanje svih pozivajućih n-torki kojima je vrijednost stranog ključa jednaka vrijednosti primarnog ključa ciljne n-torke
- iii. kao dio operacije brisanja ciljne n-torke, vrijednosti stranih ključeva u pozivajućim n-torkama postavljaju se na NULL

• Implementacija pravila integriteta

- Entitetski integritet
 - osigurava se definiranjem primarnog ključa pomoću naznake PRIMARY KEY
 - time DBMS osigurava:
 - ključni atributi relacije ne smiju imati NULL vrijednost
 - jedinstvenost ključa
- Domenski integritet
 - djelomično je definiran samom definicijom tipa podatka
 - npr. definiranjem podatka tipa SMALLINT određena je njegova domena kao skup cijelih brojeva u intervalu -32767 do 32767
- Referencijski integritet:
 - Pravila referencijskog integriteta također se definiraju prilikom kreiranja tablica pomoću naznake FOREIGN KEY

• Okidač

- Okidač (trigger) je objekt baze podataka koji se asocira s **tablicom** i aktivira kada se dogodi neki događaj nad tablicom (insert, update, delete)
- Omogućava manipuliranje podacima prije nego što stvarno budu **uneseni, izmijenjeni ili obrisani**
- Okidač je uskladištena procedura koja se ne poziva naredbom CALL nego se automatski aktivira prilikom izvršavanja odrađenih akcija od strane korisnika

• Okidač omogućava

- eksplicitno definiranje vrste događaja koji aktiviraju okidač
- **događaj** koji aktivira okidač nazivat će se “aktivirajuća operacija” (*triggering event*)
- eksplicitno specificiranje **liste naredbi** koje se obavljaju kad se dogodi aktivirajuća operacija
- skup tih naredbi nazivat će se "**aktivirana operacija**" (*triggered action list*)
- eksplicitno specificiranje **uvjeta** (*condition*) pod kojima se aktivirana operacija obavlja

• Podjela

- prema događaju (aktivirajućoj akciji) na koju se pokreće (3 kategorije):
 - INSERT
 - UPDATE
 - DELETE
- prema vremenu – 2 podkategorije
 - BEFORE
 - AFTER
- ne mogu postojati dva ista okidača za istu tablicu, npr:
 - BEFORE INSERT i BEFORE INSERT -> **ne može**
 - BEFORE INSERT i AFTER INSERT -> **može**
 - BEFORE INSERT i BEFORE UPDATE -> **može**
- okidač mora biti vezan uz pravu tablicu
- ne može se postaviti okidač na privremenu tablicu

•Sintaksa

- **CREATE TRIGGER** ime vrijeme događaj
ON tablica
FOR EACH ROW izjava
- FOR EACH ROW
 - blok naredbi izvršava se sve dok ima redaka u tablici

• Primjer – insert okidač

- INSERT OKIDAČI
 - promjena sadržaja unutar baze
 - sprječavanje umetanja novog zapisa
 - lančano ažuriranje tablica u bazi – konzistencija baze
- Primjer: Kod unosa u tablicu *account*, trigger automatski sumira novi iznos u globalnoj varijabli.

```
CREATE TABLE account  
    (acct_num INT, amount (DECIMAL(10,2)));
```

```
CREATE TRIGGER ins_sum  
    BEFORE INSERT ON account  
    FOR EACH ROW  
    SET @sum=@sum+NEW.amount;
```

•Before okidači

- Događaju se prije nego podatak dođe do tablice
- MySQL u trenutku unosa kreira dodatnu tablicu u memoriji – **NEW**
 - Struktura identična strukturi originalne tablice
 - Sadrži samo podatke koje unosimo ili mijenjamo
 - Podaci se prije unosa u originalnu tablicu unose u privremenu tablicu **NEW**, te se unutar nje može korisnički manipulirati podacima na potreban način
 - Po završetku okidača, DBMS će podatke iz tablice **NEW** automatski prenijeti u originalnu tablicu

Primjer:

```
CREATE TRIGGER provjeraImena  
  BEFORE INSERT ON klient  
  FOR EACH ROW  
    SET new.imeklijent=  
      CONCAT(new.imeKlijent, 'test');
```


•Okidači - primjer

Primjer:

Prilikom unosa novog imena u tablicu klijent, provjeriti da li je ime klijenta kraće od 5 slova. Ako jest, pridijeliti mu sufiks *test*.

- Paziti na privremenu izmjenu delimitera (kada se u tijelu okidača nalazi više od jedne naredbe)

```
DELIMITER //
```

```
CREATE TRIGGER provjeraImena BEFORE INSERT ON klijent FOR EACH ROW
```

```
  BEGIN
```

```
    IF LENGTH(new.imeklijent)<5 THEN
```

```
      SET NEW.imeklijent=concat(NEW.imeklijent,'test');
```

```
    END IF;
```

```
  END//
```

```
DELIMITER ;
```

Primjer naredbe koja će aktivirati okidač:

```
INSERT INTO klijent (sifKlijent, imeKlijent) VALUES (11, 'Pero');
```

•Before okidači

- Tablice u memoriji kod aktivirajućih operacija
 - INSERT – tablica NEW
 - UPDATE – tablice NEW i OLD
 - DELETE – tablica OLD

Primjer (BEFORE UPDATE, zadatak kao i prethodni, ali za ažuriranje umjesto za unos podataka – sufiks *test* se dodaje ako je staro ime kraće od 5 znakova):

```
DROP TRIGGER provjeraImena;  
DELIMITER //  
CREATE TRIGGER provjeraImena BEFORE UPDATE ON klijent  
FOR EACH ROW  
BEGIN  
    IF LENGTH(OLD.imeklijent)<5 THEN  
        SET NEW.imeklijent=CONCAT(NEW.imeklijent,'test');  
    END IF;  
END//  
DELIMITER ;
```

Pogrešno je ovdje koristiti *klijent.imeKlijent* jer se na taj način ne možemo referencirati na jedan zapis već najčešće na set zapisa (n-torki)

• **After okidači**

- Aktiviraju se nakon unosa podataka
- U kodu okidača imamo pristup već unesenim podacima
- Npr: ako želimo da paralelno s glavnom tablicom ažuriramo i neku arhivsku, log ili backup tablicu

•After okidači

Primjer:

Za svaki unos novog imena u tablicu, unijeti isti podatak i u *backup* tablicu naziva *backupTablica*.

```
DROP TRIGGER provjeraImena;  
DELIMITER $$  
CREATE TRIGGER provjeraImena  
  AFTER INSERT ON mojaTablica  
  FOR EACH ROW  
  BEGIN  
      INSERT INTO backupTablica  
        VALUES (NEW.ime);  
  END $$  
DELIMITER ;
```


• Ograničenja okidača – MySQL 5,6

- Trigger može pozvati proceduru
- Trigger ne smije koristiti naredbe koje eksplicitno označavaju početak i kraj transakcije
- Što ako trigger ima grešku?
 - Ako je to before ili after trigger operacija koja ga je pozvala neće se izvesti
- Triggeri se mogu pronaći u bazi - INFORMATION_SCHEMA

• Primjer okidača

- Napisati okidač koji će prilikom unosa zapisa u tablicu mjesto provjeriti je li dobro unesena županija. Ako je šifra županije neispravna, postaviti će je na nula.

```
DELIMITER $$
CREATE TRIGGER zupanija BEFORE INSERT ON mjesto
FOR EACH ROW
BEGIN
    DECLARE br INT;
    SELECT COUNT(*) INTO br FROM zupanija WHERE
        zupanija.sifZupanija=NEW.sifZupanija;
    IF br=0 THEN
        SET NEW.sifZupanija=0;
    END IF;
END;
$$
DELIMITER ;
```

- **Primjer**
- Izvršavanje sljedeće naredbe:
 - INSERT INTO mjesto VALUES(10001, 'Test', 22);
- Rezultat će s unesenom n-torkom:
 - 10001, 'Test', 0

•Primjer

- Riješimo isti problem preko procedure

```
DELIMITER $$  
CREATE PROCEDURE zupanija(INOUT sifrazup INT)  
BEGIN  
    DECLARE br INT;  
    SELECT COUNT(*) INTO br FROM zupanija WHERE  
        zupanija.sifZupanija=sifrazup;  
    IF br =0 THEN  
        SET sifrazup=0;  
    END IF;  
END;  
$$  
DELIMITER ;
```


• Primjer

- Stvorimo okidač

```
CREATE TRIGGER zupanija  
  BEFORE INSERT ON mjesto  
  FOR EACH ROW  
  CALL zupanija(NEW.sifZupanija);
```

- Pozovimo INSERT

```
INSERT INTO mjesto VALUES(10001, 'Test', 22);
```

•Primjer

- Ne želimo dopustiti brisanje iz tablice *zupanija*, a nemamo mogućnosti zabraniti naredbu *delete* naredbom *grant*

```
DELIMITER $$  
CREATE TRIGGER zupanija BEFORE DELETE ON zupanija  
FOR EACH ROW  
BEGIN  
    CALL nepostojeca_procedura();  
END;  
$$  
DELIMITER ;
```

- Problem je u tome što mysql za sada nije razvio mogućnost izazivanja pogreške (raise exception) kako bi obavijestio aplikaciju pod kojom radi. Rješenje je da pokušamo pozvati proceduru koja ne postoji i tako izazovemo grešku.

•Primjer

- Za svako brisanje zapisa u tablici *mjesto*, postaviti *pbrKlijenta* klijenata koji stanuju u mjestu koje se briše na NULL.

```
DELIMITER $$  
CREATE TRIGGER mjesto BEFORE DELETE ON mjesto  
FOR EACH ROW  
BEGIN  
    UPDATE klijent SET pbrKlijent=NULL WHERE  
        klijent.pbrKlijent=OLD.pbrMjesto;  
END;  
$$  
DELIMITER ;
```

•Primjer

- Napravite „socijalni“ okidač. Ako se ažuriraju podaci u tablici radnik na način da se unosi koeficijent plaće manji od 1 potrebno ga je odmah korigirati na 1.

```
DROP TRIGGER koef1;  
DELIMITER $$  
CREATE TRIGGER koef1 BEFORE UPDATE ON radnik  
FOR EACH ROW  
BEGIN  
    IF NEW.KoefPlaca<1 THEN  
        SET NEW.KoefPlaca=1;  
    END IF;  
END;  
$$  
DELIMITER ;
```


•Primjer

- Napravite okidač za „mogućnosti napredovanja“. Ako se ažurira tablica radnik na način da se unosi koeficijent plaće povećan za više od 2, potrebno ga je odmah korigirati da je uvećan za točno 2.

```
DROP TRIGGER koef2;  
DELIMITER $$  
CREATE TRIGGER koef2 BEFORE UPDATE ON radnik  
FOR EACH ROW  
BEGIN  
IF NEW.KoefPlaca-OLD.KoefPlaca>2 THEN  
    SET NEW.KoefPlaca=OLD.KoefPlaca+2;  
END IF;  
END;  
$$  
DELIMITER ;
```