



Okidači (triggers)

Uvod

- Osobine trigera -

- **MySQL trigger** je sačuvan program koji se pokreće automatski uvek kada se desi događaj(event) koji je povezan sa tabelom
- Događaj (event) je povezan sa 3 DML iskaza
 - **INSERT**
 - **UPDATE**
 - **DELETE**
- Jedna od dve vrste trigera se mogu zadati:
 - **BEFORE**
 - **AFTER**

- Osobine trigerera -

Triger je MySQL objekat koji se pokreće(trigger) na određenu **akciju ili kalkulaciju pre (before) ili posle (after) unosa (Insert), brisanja (Delete) ili ažuriranja (Update) iskaza koji se izvršava.**

Triger može da se pokrene **pre nego što se novi zapis(rekord) unese** u tabelu ili **pošto se zapis(record) ažurira.**

Trigeri

“Događaj(event)-Uslov(condition)-Akcija (action rules)”

Kada se *događaj* desi, proverava se uslov; ukoliko je ispunjen, odradi akciju automatski

- 1) Prebaciju logiku nadgledanja baze sa aplikacije na DBMS.
- 2) Imaju primenu na ograničenja integriteta podataka

* Implementacije se značajno razlikuju

trigeri - sintaksa SQL standard

Create Trigger **name**

Before|After|Instead Of **events**

[**referencing-variables**]

[For Each Row]

When (**condition**)

action

Opisuje kada se triger aktivira (**pre, posle ili umesto**) određenog događaja (insert, delete ili updateu tabeli)

Referenciraju podatke koji su modifikovani a koji aktiviraju triger.
old row as ime
new row as ime
old table as ime
new table as ime

Row level je opcioni iskaz i ukazuje da se triger aktivira za svaki modifikovani red.
Ukoliko ova opcija nije prisutna statment level, triger se pokreće samo jedanput za ceo iskaz

Slično Where klauzuli u SQL-u. Ispituje se uslov i ukoliko je ispunjen sprovodi se akcija

SQL iskaz

Primer implementacije Referencijalnog integriteta: R.A se referencira na S.B, kaskadno brisanje

```
Create Trigger Cascade  
After Delete On S  
Referencing Old Row As O  
For Each Row  
[ no condition ]  
Delete From R where A = O.B
```

Triger se aktivira kada obrisemo bilo koji red u tabeli S

Obrisani red označavamo sa O

Triger se aktivira za svaki obrisani red

Nema uslova

Obrisati redove iz tabele R gde je A jednako sa B obrisanog reda(O) iz tabele S

Primer implementacije Referencijalnog integriteta: R.A se referencira na S.B, kaskadno brisanje

```
Create Trigger Cascade  
After Delete On S  
Referencing Old Tables AS OT  
[ For Each Row ]  
[ no condition ]  
Delete From R where A = O.B
```

Promenjivu smo nazvali OT koja će sadržati set obrisanih redova

Ne koristimo ovaj iskaz

Primer implementacije Referencijalnog integriteta:
R.A se referencira na S.B, kaskadno brisanje

Create Trigger **Cascade**

After Delete On **S**

Referencing **Old Table** As **OT**

[For Each Row]

[no condition]

Delete From **R** where **A** in (select **B** from **OT**)

Triki Problemi

- Row-level vs. Statement-level
 - New/Old Row and New/Old Table
 - Before, Instead Of
- Više trigera aktiviranih u isto vreme (Koji se izvršava prvi)
- Aktivacija jednog trigera aktivira drugi triger (chaining)
 - Self trigeri, kružni, ugnježdeni
- Uslov implementiran u **when** segmentu ili **action** segmentu
- * Implementacija se značajno razlikuje (SQLite ima samo row level trigere)

Kompleksnost u
radu sa Triggerima

$T(K,V)$ – K ključ, V vrednost

Create Trigger **IncreaseInserts**

After Insert On **T**

Referencing New Row As **NR**, New Table As **NT**

For Each Row

When (Select Avg(**V**) From **T**) <
(Select Avg(**V**) From **NT**)

Update **T** set **V=V+10** where **K=NR.K**

Postgres >

- Implementiran full standard
- row-level + statement-level
- old/new row & table

SQLite >>

- Podržava samo row-level
- Neposredna aktivacija
- Nema old/new table

MySQL

- Podržava samo row level
- Neposredna
- Nema old/new table
- Samo jedan triger po tipu događaja i Ograničen lanac trigera

SQLite

- Podržava samo row-level
- Neposredna aktivacija
- For each Row se podrazumeva ukoliko se ne navede
- Nema Old Table ili New Table
- Nema iskaza za referenciranje
 - Old i New su predefinisane za Old Row i New Row
- Trigger akcija: SQL iskaz u begin-end bloku

Primer Baze

Primer: Prijava studenata baza podataka

Fakultet(Fime, Grad, BrojMesta)

Student(Sid, Sime, Prosek, Vskole)

Prijava(Sid, Fime, Smer, Odluka)

Student

Sid	Sime	Prosek	Vskole

Prijava

Sid	Fime	Smer	Odluka

Fakultet

Fime	grad	BrojMesata

```
create Trigger R1
after insert on Student
for each row
when New.Prosek > 3.3 and New.Prosek<=3.6
begin
    insert into Prijava values (New.Sid, 'VTS', 'SRT', null);
    insert into Prijava values (New.Sid, 'FON', 'RTI', null);
end;
```

```
insert into Student values ('111', 'Kevin', 3.5, 1000);
insert into Student values ('222', 'Lori', 3.8, 1000);|
```

- Triger presreće ili se aktivira ubacivanjem studenata u tabeli Student
- za studenta, on se automatski upisuje u tabeli Prijava na VTS smer SRT i FON smer RTI.
- Triger **neće sprečiti ubacivanje novog studenta** u tabeli Student čak i ako uslov nije ispunjen jer se triger aktivira nakon (after) ubacivanja novog studenta

```
create Trigger R2
after delete on Student
for each row
begin
    delete from Prijava where Sid = Old.Sid;
end;
```

- Ovaj trigger simulira **cascade delete** kod referencijalnog integriteta
- Trigger presreće ili se aktivira brisanjem studenta u tabeli Student
- Trigger **ne koristi uslov** već samo akciju, briše sve podatke za izbrisanog studenta iz tabele Prijava
- **Old.Sid** se odnosi na SID obrisano studenta

```
create Trigger R3
after update of fime on Fakultet
for each row
begin
    update Prijava
    set fime = New.fime
    where fime = Old.fime;
end;
```

- Koristi se *update of fime* jer želimo da se triger aktivira samo kada se ažurira ime koledža u suprotnom svaki update kolone koledž bi aktivirao triger što nije efikasno
- Ovaj triger simulira cascade update kod referencijalnog integriteta
- Triger presreće ili se aktivira ažuriranjem kolone fime u tabeli koledž
- Akcija zahteva da se u tabeli Prijava za vrednost kolone cName upiše promenjena nova vrednost
- **New.fime** se odnosi sa promenjeni naziv univerziteta
- **Old.fime** se odnosi na stari naziv univerziteta

- Prethodna dva trigera simulirala su ograničenja referencijalnog integriteta
- Sledeća dva trigera simuliraće ograničenja na osnovu ključeva

```
create Trigger R4
before insert on Fakultet
for each row
when exists ( select * from Fakultet where fime =
              New.fime)
begin
    select raise(ignore);
end;
```

- Triger presreće ili se aktivira **pre(before)** unosa novog koledža.
- Uslov proverava da li ime (ključ) novog koledža već postoji u tabeli koledž tj. **sprečava pojavu duplikata**
- Akcija zahteva ignorisanje novog zahteva neće javiti grešku

```
create Trigger R5
before update of fime on Fakultet
for each row
when exists ( select * from Fakultet where fime =
              New.fime)
begin
    select raise(ignore);
end;
```

- Triger presreće ili se aktivira **pre(after)** modifikacije novog koledža.
- Uslov proverava da li ime (ključ) modifikovanog koledža već postoji u tabeli koledž tj. **sprečava pojavu duplikata**
- Akcija zahteva ignorisanje novog zahteva **neće javiti grešku**

```
create trigger R6
before insert on Prijava
for each row
when ( select count(*) from Prijava where fime = New.fime)
begin
    update Fakultet set fime = fime || '-Done'
    where fime = New.fime;
end;
```

- Triger presreće ili se aktivira **pre(after)** ubacivanja nove prijave u tabeli Prijava.
- Uslov proverava da li je **ukupan broj prijava** za uneti koledž **veći od 10**
- Ukoliko jeste **akcija ažurira kolonu fime** u tabeli koledž tako što se **pored imena doda i Done**

- Primer Insert i Update trigger-

```
DELIMITER $$
CREATE TRIGGER before_salaries_insert BEFORE
INSERT ON salaries FOR EACH ROW
BEGIN
    IF NEW.salary < 0
        THEN SET NEW.salary = 0;
    END IF;
END
$$DELIMITER ;
```

```
DELIMITER $$
CREATE TRIGGER trig_upd_salary BEFORE
UPDATE ON salaries FOR EACH ROW
BEGIN
    IF NEW.salary < 0
        THEN SET NEW.salary = OLD.salary;
    END IF;
END
$$DELIMITER ;
```

Ukupno trigera u tabeli u MySQL-u

- Pre verzije MySQL-a 5.7.2 dozvoljeno je bilo kreiranje jednog trigera po događaju(event) u tabeli.
 - Po jedan triger za **BEFORE UPDATE** ili **AFTER UPDATE** događaj
- Od verzije 5.7.2 dozvoljeno je kreiranje više trigera u istoj tabeli koji imaju isti događaj(event) i vreme akcije.
 - Trigeri se aktiviraju sekvencijalno kada se događaj desi
- Follows i Precedes definišu da li će se novi triger pokrenuti pre ili posle postojećeg trigera

```
DELIMITER $$  
  
CREATE TRIGGER trigger_name  
{BEFORE|AFTER}{INSERT|UPDATE|DELETE}  
ON table_name FOR EACH ROW  
{FOLLOWS|PRECEDES} existing_trigger_name  
BEGIN  
    -- statements  
END$$  
  
DELIMITER ;
```


- Evidentiranje događaja uz pomoć trigera -

```
delimiter $$
```

```
create trigger before_update_products before update on products for each row
```

```
begin
```

```
    insert into log_products(name,old_price,new_price,created_at)
```

```
    value (old.name,old.price,new.price, now());
```

```
end $$
```

```
delimiter ;
```

```
update products set price=2.32 where id=4;
```

ŠTA ĆE SE DESITI NAKON PROMENE CENE?

- Validacija podataka uz pomoć trigera -

- U log tabeli za Proizvode *log_products* sačuvati svaku promenu cene iz tabele *products* za proizvod dok za cene koje su veće od 10 ne čuvati unetu cenu već maksimalno dozvoljenu cenu 10.

```
delimiter $$  
create trigger before_update_products before update on products for each row  
begin  
if new.price>10 then  
    insert into log_products (name,old_price,new_price, created_at) value (old.name,old.price,new.price,now());  
    set new.price=10;  
else  
    insert into log_products (name,old_price,new_price, created_at) value (old.name,old.price,new.price,now());  
end if;  
end$$  
delimiter ;
```


- Validacija podataka uz pomoć trigera -

- U log tabeli za Proizvode *log_products* sačuvati svaku promenu cene iz tabele *products* za proizvod dok za cene koje su veće od 10 ne čuvati unetu cenu već maksimalno dozvoljenu cenu 10.

```
delimiter $$
create trigger before_update_products before insert on products for each row
begin
  if new.price>10 then
    set new.price=10;
  end if;
end$$
delimiter ;
```

- Zadatak-

- Kreirati trigger u tabeli *users* za bazu *ig_clone* koji proverava da li je uneti datum veći od trenutnog. Ukoliko jeste potrebno je upisati trenutni datum

```
delimiter $$  
create trigger before_insert_user_date before insert on users for each row  
begin  
    if new.created_at > now()  
        then set new.created_at=now();  
    end if;  
end$$  
delimiter ;
```

```
insert into users (username,created_at) values ('Marko','2021-12-03');
```

- Zadatak1-

- Kreirati triger koji će da se aktivira kada se za radnika unese datum penzionisanja tako što se promeniti vrednost polja info sa 0 na 1.

```
create database kompanija;  
use kompanija;  
create table radnici (id int auto_increment primary key,  
ime varchar(20),  
datum_zaposlenja date,  
datum_penzionisanja date,  
info bool);
```

id	ime	datum_zaposlenja	datum_penzionisanja	info
1	Dusan	2000-10-10	2020-05-12	1
2	Ana	1987-05-23	2020-05-12	1
3	Branko	1978-05-23	NULL	0

- Rešenje Zadatka 1-

- Kreirati trigger koji će da se aktivira kada se za radnika unese datum penzionisanja tako što se promeniti vrednost polja info sa 0 na 1.

```
delimiter $$  
create trigger before_update_radnici before update on radnici for each row  
begin  
    if new.datum_penzionisanja is not null  
        then set new.info=1;  
    end if;  
end $$  
delimiter ;
```

- Zadatak2 -

- Kreirati trigger koji će radnika koji je penzionisan na osnovu polja info da prebaci u tabelu penzioneri

```
create table penzioneri (id int auto_increment primary key,  
                        ime varchar(20),  
                        datum_penzionisanja date);
```

```
delimiter $$  
create trigger after_update_rad after update on radnici for each row  
begin  
    if new.info=1  
        then insert into penzioneri(ime, datum_penzionisanja) values (new.ime,new.datum_penzionisanja);  
    end if;  
end$$  
delimiter ;
```