OKIDAČI

1. Što su okidači

Okidač (eng. trigger) je objekt baze podataka koji se asocira s tablicom i aktivira u trenutku određenog događaja nad tablicom. Čini ga niz definiranih SQL naredbi koje se automatski izvode prilikom unosa, izmjene ili brisanja podataka. Okidači implementiraju funkcionalnost baze podataka koja se mora održati kada dođe do promjene nad podacima u tablici. Obirom da se aktivira automatski, nije ga moguće zaobići putem aplikacije koja koristi dotičnu bazu podataka.

Okidači su korisni kod automatiziranja operacija nad bazom na način da se smanjuje teret zadataka koje bi aplikacija morala izvršiti, kao i kod očuvanja integriteta baze podataka. Neki od primjera primjene okidača su:

- Prilikom održavanja poslovne logike
- Izvođenje automatskih izračuna
- Mijenjanje podataka u jednoj tablici kao odgovor na promjene u drugoj
- Prilikom evidentiranja promjena u podacima (korisnički izrađeni logovi)
- Stavranje snimke podataka prije promjene (za eventualno poništavanje funkcionalnosti)

Okidač se može smatrati uskladištenom procedurom koja se ne poziva naredbom CALL već se automatski aktivira prilikom izvršavanja odrađenih akcija od strane korisnika. Omogućava manipuliranje podacima prije nego što stvarno budu uneseni, izmijenjeni ili obrisani, odnosno aktivira se prilikom naredbe INSERT, UPDATE ili DELETE. Događaj koji aktivira okidač zove se „aktivirajuća operacija“ (eng. triggering event). Potrebno je eksplicitno specificirati listu naredbi koje se izvršavaju kad se dogodi aktivirajuća operacija. Skup tih naredbi naziva se „aktivirana operacija“ (eng. triggered action list). Aktivirane operacije izvršavaju se kod eksplicitno specificiranih „uvjeta“ (eng. condition).

2. Rad s okidačima

Vrste okidača

Prema aktivirajućoj operaciji okidači se dijele na 3 kategorije:

- INSERT
- UPDATE
- DELETE

Prema vremenu izvođenja dijele se na dvije podkategorije:

- BEFORE
- AFTER

Na jednoj tablici nije moguće stvoriti dva okidača koji će biti isti i prema aktivirajućoj operaciji i prema vremenu izvođenja (primjerice ne mogu postojati dva before insert okidača na istoj tablici, ali može jedan before insert i drugi koji je after insert). Okidač je vezan isključivo uz pravu tablicu, i ne može biti vezan uz privremenu tablicu. Svaki okidač je objekt u jednoj bazi podataka i vrijedi samo za tu bazu. Prilikom brisanja baze podataka, bit će obrisani i svi njeni okidači. Da bi korisnik u bazi mogao izraditi okidač, potrebno je da su mu dodjeljene odgovarajuće privilegije (TRIGGER privilege u MySQL 5.1.6+ ili SUPER privilege u MySQL 5.0.x).

Kreiranje okidača

Osnovna sintaksa:

```
CREATE TRIGGER imeOkidaca vrijeme događaj
        ON tablica
FOR EACH ROW izjava
```

Klauzula FOR EACH ROW je obavezna po MySQL sintaksi. Označava da će okidač biti izvršen za svaki redak na kojeg se odnosi DML naredba koja je aktivirala okidač. Prema ANSI standardu moguća je FOR EACH STATEMENT klauzula koje će biti podržana u jednoj od budućih MySQL verzija.

Obzirom da će okidač u većini slučajeva izvršavati više SQL naredbi, one će nakon FOR EACH ROW izjave biti definirane unutar BEGIN … END bloka.

Brisanje okidača

Osnovna sintaksa:

```
DROP TRIGGER imeOkidaca
```

3. Referenciranje na stare / nove vrijednosti

U tijelu okidača moguće je referencirati se na stare vrijednosti koje će biti promijenjene ili obrisane odnosno na nove koje će biti unesene u tablicu. Navedeno je moguće ključnim riječima NEW odnosno OLD nakon kojih se navodi naziv kolone na koju će se podatak odnositi. *NEW.imeKolone* odnosi se na podatak koji će tek biti unesen u bazu, odnosno *OLD.imeKolone* odnosi se na postojeći redak u tablici koji će biti izbrisan ili promijenjen. Navedene oznake postoje i kod BEFORE i kod AFTER okidača. Ključne riječi OLD i NEW nisu osjetljive na mala i velika slova.

Primjerice, ako na određenoj tablici postoji BEFORE UPDATE okidač, vrijednost u koloni koja će tek biti izmjenjena (neka se zove *kolona1*) je OLD.kolona1, a vrijednost nakon izmjene je NEW.kolona1.

Popis tipova okidača i postojećih pripadajućih oznaka:

- ✓ INSERT -> NEW
- ✓ UPDATE -> NEW, OLD
- ✓ DELETE -> OLD

NEW.imeKolone

- Sadrži podatke koji će biti uneseni u tablicu prilikom izvođenja naredbe INSERT ili UPDATE
- Moguće je korisnički manipulirati s ovim podacima prije nego se unesu u originalnu tablicu uz koju je okidač vezan -> sintaksa: `SET NEW.imeKolone=novaVrijednost`
- Navedena mogućnost promjene vrijednosti u retku NEW moguća je isključivo kod BEFORE okidača
- Po završetku rada okidača, podaci koji imaju prefiks NEW, automatski će biti uneseni u originalnu tablicu

Primjer:

Potrebno je napraviti okidač koji će osigurati da se u tablici kvar atribut BrojRadnika ne može postaviti na nulu niti na negativnu vrijednost. U tom je slučaju potrebno postaviti BrojRadnika na 1.

```
DELIMITER $$
DROP TRIGGER IF EXISTS provjeriBrRadnika $$
CREATE TRIGGER provjeriBrRadnika
        BEFORE UPDATE ON kvar FOR EACH ROW
BEGIN
        IF NEW.brojRadnika<=0 THEN
                SET NEW.brojRadnika=1;
        END IF;
END
$$
DELIMITER ;
```

Primjer naredbe koja će aktivirati okidač:

```
UPDATE kvar SET brojRadnika=0 WHERE nazivKvar="Zamjena pragova";
```

Rezultat izvođenja gornje UPDATE naredbe:

sifKvar	nazivKvar	sifOdjel	brojRadnika	satiKvar
36	Zamjena prednjeg fara	9	1	2
22	Popravak radilice	10	8	3
18	Zamjena klipa	10	6	4
7	Izmjena amortizera	6	4	2
4	Zamjena pragova	6	1	4
25	Zamjena rashladne tekućine	11	6	1
3	Zamjena cerade	6	3	3
14	Uštimavanje pokretnog krova	7	7	2

OLD.imeKolone

- Sadrži podatke koji će biti izbrisani u tablici izvođenjem naredbe DELETE ili promijenjeni izvođenjem naredbe UPDATE
- Ove je podatke moguće isključivo čitati, ali ne i mijenjati

Primjer:

Potrebno je napraviti okidač koji će osigurati da se radniku ne može povećati koeficijent plaće za više od 10%.

```
DELIMITER $$
CREATE TRIGGER kontrolirajNapredovanje
        BEFORE UPDATE ON radnik FOR EACH ROW
BEGIN
        DECLARE povisica INT DEFAULT NULL;
        SELECT NEW.koefPlaca-OLD.KoefPlaca INTO povisica;
        IF povisica>(10*OLD.KoefPlaca/100) THEN
                SET NEW.koefPlaca=OLD.KoefPlaca;
        END IF;
END
$$
DELIMITER ;
```

Primjer naredbe koja će aktivirati okidač:

```
UPDATE radnik SET KoefPlaca=3.5 WHERE sifRadnik=130;
```

Rezultat izvođenja gornje UPDATE naredbe:

sifRadnik	imeRadnik	prezimeRadnik	pbrStan	sifOdjel	KoefPlaca	IznosOsnovice
122	Tin	Grabovac	49000	19	1.38	2200.00
126	Zdravko	Budišin	53000	27	0.56	2200.00
130	Antun Ivan	Jonjić	10000	13	2.69	2200.00
134	Želimir	Somrak	33000	8	1.80	2200.00

Primjer naredbe koja NEĆE aktivirati okidač:

```
UPDATE radnik SET KoefPlaca=2.8 WHERE sifRadnik=130;
```

Rezultat izvođenja gornje UPDATE naredbe:

sifRadnik	imeRadnik	prezimeRadnik	pbrStan	sifOdjel	KoefPlaca	IznosOsnovice
122	Tin	Grabovac	49000	19	1.38	2200.00
126	Zdravko	Budišin	53000	27	0.56	2200.00
130	Antun Ivan	Jonjić	10000	13	2.80	2200.00
134	Želimir	Somrak	33000	8	1.80	2200.00

4. BEFORE i AFTER okidači

Kada koristiti BEFORE, a kada AFTER okidač?

BEFORE i AFTER klazule definiraju kada se naredbe navedene u tijelu okidača izvršavaju, da li prije ili poslije DML izjava koje su pokrenule okidač. Glavna razlika između ove dvije vrste okidača je ta da kod AFTER okidača nije moguće mijenjati vrijednosti koje će biti unesene ili ažurirane u tablici (NEW.imeKolone) na koju se okidač odnosi. U tom je slučaju kasno pokušavati mijenjati radnju aktivirajuće operacije. Podaci NEW.imeKolone biti će uneseni bazu nepromijenjeni, a pokušaj njihovog mijenjanja rezultirat će pogreškom. Primjer ovog slučaja dan je u nastavku.

Primjer:
Potrebno je napraviti okidač koji će osigurati da se u tablicu mjesto ne može unijeti novi zapis takav da mu vrijednost u koloni nazivMjesto sadrži manje od 2 slova.

```
DELIMITER $$
CREATE TRIGGER duljinaNazivaMjesta
        AFTER INSERT ON mjesto FOR EACH ROW
BEGIN
        IF LENGTH(NEW.nazivMjesto)<2 THEN
                SET NEW.nazivMjesto=CONCAT(NEW.nazivMjesto,"_test");
        END IF;
END
$$
DELIMITER ;
```

Izvođenje gornjeg koda rezultirat će pogreškom:

```
Error Code: 1362
Updating of NEW row is not allowed in after trigger
```

Ispravno je:

```
DELIMITER $$
CREATE TRIGGER duljinaNazivaMjesta
        BEFORE INSERT ON mjesto FOR EACH ROW
BEGIN
        IF LENGTH(NEW.nazivMjesto)<2 THEN
                SET NEW.nazivMjesto=CONCAT(NEW.nazivMjesto,"_test");
        END IF;
END
$$
DELIMITER ;
```

Primjer naredbe koja će aktivirati okidač:

```
INSERT INTO mjesto VALUES (23500, 'A', 17);
```

Rezultat izvođenja gornje INSERT naredbe:

pbrMjesto	nazivMjesto	sifZup...
31205	Aljmaš	14
23500	A_test	17
20225	Babino Polje	19

Iako se većina potrebnih funkcionalnosti može riješiti sa BEFORE okidačima, AFTER okidače će biti korisno upotrijebiti za radnje koje se logički trebaju izvršiti tek po uspješnom završetku aktivirajuće operacije. Primjerice, revizijske aktivnosti najbolje je izvršiti tek po uspješnom završetku DML naredbe (aktivirajuće operacije). Prilikom izrade sigurnosne kopije (bekapa) također se najčešće koriste AFTER okidači.

Primjer:
Potrebno je napraviti okidač koji će prilikom svakog unosa u tablicu nalog, taj isti podatak unijeti i u backup tablicu naziva nalog_backup.

```
DELIMITER $$
CREATE TRIGGER backupNaloga
AFTER INSERT ON nalog
FOR EACH ROW
BEGIN
INSERT INTO nalog_backup
VALUES (NEW.sifKlijent, NEW.sifKvar, NEW.sifRadnik, NEW.datPrimitkaNalog,
NEW.prioritetNalog, NEW.OstvareniSatiRada);
END $$
DELIMITER ;
```

Primjer naredbe koja će aktivirati okidač:

```
INSERT INTO nalog VALUES (1251, 22, 501, '2013-01-02', 1, 7);
```

Rezultat izvođenja gornje INSERT naredbe:

sifKlijent	sifKvar	sifRadnik	datPrimitkaNalog	prioritetNalog	OstvareniSatiRada
1251	22	501	2013-01-02	1	7
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

5. Ograničenja okidača

Okidač ne smije koristiti naredbe koje eksplicitno označavaju početak i kraj transakcije. Potrebno je izbjeći primjerice izazivanje neželjenog završetka transakcije u trenutku kada je transakcija izvršila određenu DML naredbu koja je aktivirala okidač.

Ako postoji pogreška u okidaču, aktivirajuća operacija neće se izvesti. U slučaju da na tablici postoje i BEFORE i AFTER okidač te ako dođe do pogreške prilikom izvođenja BEFORE okidača, tada se niti AFTER okidač neće izvesti.

MySQL do sada nije razvio mogućnost izazivanja pogreške (raise exception) kako bi obavijestio aplikaciju pod kojom radi da nešto nije u redu. Ako je potrebno direktno prekinuti izvođenje određene radnje, rješenje je prouzrokovati pogrešku što će rezultirati prekidom rada okidača te prekidom aktivirajuće operacije. To se može napraviti na nekoliko načina:

- Unosom podatka u nepostojeću kolonu
- Unosom NULL vrijednosti u kolonu za koju je definirano da ne smije poprimiti NULL vrijednost
- Pozivom nepostojeće pohranjene procedure

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

Primjer naredbe koja će aktivirati okidač:

```
DELETE FROM odjel WHERE sifOdjel=2;
```

Izvođenje gornje naredbe prouzrokovat će (namjerno izazvanu) pogrešku:

```
Error Code: 1305
PROCEDURE autoradionica.nepostojeca_procedura does not exist
```

6. Okidači i procedure

Jedno od obilježja okidača je da osim izvođenja seta naredbi može pozvati i proceduru. Primjer je danu nastavku.

Primjer:

Potrebno je napraviti okidač koji će se pobrinuti za ***osiguravanje referencijskog integriteta*** vezanog uz tablicu mjesto.

Prilikom brisanja zapisa u tablici mjesto, potrebno je provjeriti da li je to mjesto referencirano u tablici klijent ili radnik. Ako je referencirano, potrebno je postaviti vrijednosti odgovarajućih atributa u tablicama klijent i radnik na NULL.

```
DROP TRIGGER mjesto;
DELIMITER $$
CREATE TRIGGER mjesto AFTER DELETE ON mjesto
FOR EACH ROW
BEGIN
        UPDATE klijent SET pbrreg=NULL
                WHERE pbrreg=OLD.pbrmjesto;
        UPDATE klijent SET pbrKlijent=NULL
                WHERE pbrKlijent=OLD.pbrmjesto;
        UPDATE radnik SET pbrStan=NULL
                WHERE pbrStan=OLD.pbrmjesto;
END;
$$
DELIMITER ;
```

Primjer naredbe koja će aktivirati okidač:

```
DELETE FROM mjesto WHERE pbrMjesto=49000;
```

Provjerom zapisa u tablici klijent i radnik možemo se uvjeriti da ne postoji niti jedan radnik odnosno klijent sa vrijednosti pbrStan, pbrReg ili pbrKlijent jednakima 49000.

Isti je okidač moguće realizirati i pozivom procedure:

U tom se slučaju procedura brine za osiguravanje integriteta, a okidač će pozvati proceduru i predati joj odgovarajući argument.

```
DROP PROCEDURE IF EXISTS radi_mjesto;
DELIMITER $$
CREATE PROCEDURE radi_mjesto(IN pbr INT)
BEGIN
        UPDATE klijent SET pbrreg=NULL
                WHERE pbrreg=pbr;
        UPDATE klijent SET pbrKlijent=NULL
                WHERE pbrKlijent=pbr;
```

```
        UPDATE radnik SET pbrStan=NULL
                WHERE pbrStan=pbr;
END;
$$
DELIMITER ;
```

```
DROP TRIGGER mjesto;
DELIMITER $$
CREATE TRIGGER mjesto BEFORE DELETE ON mjesto
FOR EACH ROW
BEGIN
        CALL radi_mjesto(OLD.pbrMjesto);
END;
$$
DELIMITER ;
```

Primjer naredbe koja će aktivirati okidač:

```
DELETE FROM mjesto WHERE pbrMjesto=22000;
```

7. Okidači i evidentiranja promjena u podacima

Kroz prethodna poglavlja već su dani primjeri primjene okidača. Jedna od primjena je snimanje informacija o izmjenama u bazi na korisnički definiran način.

Primjer:

Prilikom dodjeljivanja postojećeg naloga nekom drugom radniku, potrebno je podatke o zamjeni pohraniti u tablicu transaction_log. Tablicu transacion_log kreirati u istoj bazi podataka.

```
CREATE TABLE `transaction_log` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` text COLLATE cp1250_croatian_ci,
  `description` text COLLATE cp1250_croatian_ci,
  `time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=cp1250 COLLATE=cp1250_croatian_ci
```

Okidač:

```
DELIMITER %%
CREATE TRIGGER nalog_ai
AFTER UPDATE ON nalog FOR EACH ROW
BEGIN
        INSERT INTO transaction_log
                (user_id, description)
                VALUES (CURRENT_USER(),
                        CONCAT('Stari radnik -',
                        OLD.sifRadnik,'
                        promijenjen u: novi radnik - ',
                        NEW.sifRadnik));
END;
%%
DELIMITER ;
```

Primjer naredbe koja će aktivirati okidač:

```
UPDATE nalog SET sifRadnik=30
        WHERE sifKlijent=1369 AND datPrimitkaNalog='2005-08-22';
```

Tablica transaction_log:

ID	user_id		description		time
6	root@localhost	14B	Stari radnik - 30 promijenjen u: novi radnik - 162	50B	2013-01-19 16:06:15
(Auto)	(NULL)	0K	(NULL)	0K	CURRENT_TIMESTAMP