# Aufgabe Trigger

Mario Fentler 5CHIT
20.02.2019

## *Deployment*

Einfach das mitgegebene File ausführen. (**WICHTIG** -> davor mysql -p restaurant, um in die Datenbank zu kommen)
Dieses File erstellt am Anfang die Tabellen neu und danach den Trigger + Testabfragen.

## *Praxis*

### a)

Wenn bei einer zu speichernden rechnung die Spalte datum den Wert NULL hat, soll sie jeweils durch den aktuellen Wert CURRENT_DATE ersetzt werden.

**->** Hier muss mit einer IF-Abfrage gearbeitet werden.

```
-- Aufgabe AU09a
-- Mario Fentler

DELIMITER //
DROP TRIGGER IF EXISTS trigger_a//

CREATE TRIGGER trigger_a
    BEFORE INSERT
    ON rechnung
    FOR EACH ROW
    BEGIN
    IF NEW.datum IS NULL THEN
        SET NEW.datum = CURRENT_DATE();
    END IF;
END; //
DELIMITER ;

SELECT * FROM rechnung;
INSERT INTO rechnung(rnr,datum,tisch,status,knr) VALUES (7,NULL,2,'offen',1);
SELECT * FROM rechnung;
```

### b)

Wenn in der Tabelle speise ein preis geändert wird, soll ein zusätzlicher Datensatz in der Tabelle preisaenderung eingefügt werden. In der Spalte datum soll das aktuelle Datum gespeichert werden und in der Spalte aenderung soll die Preisänderung gespeichert werden. PK (snr, datum).

**->** Hier arbeite ich mit einer neuen Tabelle, diese hat als PK eine id mit AUTO-INCREMENT Constraint, was bedeutet, dass ich einfach INSERTEN kann ohne mich um doppelte Daten kümmern zu müssen.

```
-- Aufgabe AU09b
-- Mario Fentler

DROP TABLE IF EXISTS preisaenderung;
CREATE TABLE preisaenderung(
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    datum DATE,
    snr INTEGER,
    bezeichnung VARCHAR(255),
    preisVorher DECIMAL(6,2),
    preisNachher DECIMAL(6,2)
);

DROP TRIGGER IF EXISTS trigger_b;
DELIMITER //
CREATE TRIGGER trigger_b
    AFTER UPDATE
    ON speise
    FOR EACH ROW
BEGIN
    INSERT INTO preisaenderung(datum,snr,bezeichnung,preisVorher,preisNachher)
VALUES(CURRENT_DATE,OLD.snr,OLD.bezeichnung,OLD.preis,NEW.preis);
END; //
DELIMITER ;

-- update preis pute natur
UPDATE speise SET preis = 4.5 WHERE snr = 4;
-- show in table
SELECT * FROM preisaenderung;
```

### c)

Wenn in der Tabelle bestellung ein Datensatz gelöscht wird, soll ein zusätzlicher Datensatz in der Tabelle bestellstorno gespeichert werden.

**->** Bevor der Datensatz gelöscht wird speichere ich ihn noch in einer Tabelle mit gleicher Struktur. (BEFORE DELETE)

```
-- Aufgabe AU09c
-- Mario Fentler

DROP TABLE IF EXISTS bestellstorno;
CREATE TABLE bestellstorno(
    datum DATE,
    anzahl SMALLINT,
    rnr         INTEGER,
    snr         INTEGER,
```

```
    PRIMARY KEY (rnr, snr),
    FOREIGN KEY (rnr) REFERENCES rechnung (rnr)
                    ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (snr) REFERENCES speise (snr)
                    ON UPDATE CASCADE ON DELETE CASCADE
);

DROP TRIGGER IF EXISTS trigger_c;
DELIMITER //
CREATE TRIGGER trigger_c
    BEFORE DELETE
    ON bestellung
    FOR EACH ROW
BEGIN
    INSERT INTO bestellstorno (datum,anzahl,rnr,snr)
    VALUES (CURRENT_DATE(),OLD.anzahl,OLD.rnr,OLD.snr);
END; //
DELIMITER ;

-- Testing
SELECT * FROM bestellung;
DELETE FROM bestellung WHERE rnr = 4;
-- Datensatz wurde geloescht
SELECT * FROM bestellung;
-- Neuer Datensatz in der Stornotabelle
SELECT * FROM bestellstorno;
```

### d)

In der Tabelle statistik soll datumsabhängig die Anzahl aller im Restaurant angebotenen Speisen dokumentiert werden. Erstelle alle erforderlichen Definitionen!

**->** Hier braucht man zwei Trigger. Einen für INSERT und einen für DELETE, da sich bei beiden Events die Anzahl an Speisen ändern kann (Update daher irrelevant).
Hier wird auch statt INSERT und UPDATE in der Tabelle ein REPLACE verwendet. REPLACE fügt einen neuen Wert in die Tabelle ein wenn es den noch nicht gibt. Sonst ersetzt es den Wert. Das **wird gebraucht, weil es an einem Tag mehrere Änderungen geben kann** und die Daten in der Tabelle aber nur nach dem Datum gespeichert werden sollen.

```
-- Aufgabe AU09d
-- Mario Fentler

DROP TABLE IF EXISTS statistik;
CREATE TABLE statistik(
    datum DATE PRIMARY KEY,
    anzahl INTEGER
);

DROP TRIGGER IF EXISTS trigger_d_1;
DELIMITER //
CREATE TRIGGER trigger_d_1
```

```
    AFTER INSERT
    ON speise
    FOR EACH ROW
    BEGIN
        REPLACE INTO statistik (datum,anzahl)
        SELECT
            CURRENT_DATE(),count(*)
        FROM speise;
    END; //

DROP TRIGGER IF EXISTS trigger_d_2//
CREATE TRIGGER trigger_d_2
    AFTER DELETE
    ON speise
    FOR EACH ROW
    BEGIN
        REPLACE INTO statistik (datum,anzahl)
        SELECT
            CURRENT_DATE(),count(*)
        FROM speise;
    END; //
DELIMITER ;

-- Testing
INSERT INTO speise(snr,bezeichnung,preis) VALUES (9, 'Test menue', 14.5);
SELECT * FROM statistik;
INSERT INTO speise(snr,bezeichnung,preis) VALUES (10, 'Pizza', 3.99);
SELECT * FROM statistik;

-- Now deleting one speise
DELETE FROM speise WHERE snr = 10;
SELECT * FROM statistik;
```

e)

Die ENGINE=MYISAM gestattet die Verwendung von Triggern. Realisiere eine 1:N Beziehung mit der ENGINE=MYISAM, d.h. das FK-Constraint und die CASCADE Verarbeitung sollen mittels Triggern nachgebildet werden.

# *Theorie*

## Frage 1)

***Sind mehrere Trigger für dasselbe Aktivierungs-Event (z.B. BEFORE INSERT ON xxx) zulässig?***

Nein, man kann nicht zwei Trigger auf die selbe Tabelle mit dem selben Event haben. Eine Außnahme ist, wenn der eine Trigger "tiefer" geht. Und sich beispielsweise auf eine Spalte der Tabelle bezieht.

---

## Frage 2)

***Können innerhalb BEGIN ... END mehrere SQL-Anweisungen definiert werden?***

Ja, das funktioniert wie mit den Stored Procedures/Functions.

---

## Frage 3)

***Können Trigger die Anweisungen START TRANSACTION, COMMIT oder ROLLBACK enthalten?***

Nein, "*The trigger cannot use statements that explicitly or implicitly begin or end a transaction such as START TRANSACTION, COMMIT, or ROLLBACK.*" - Quelle

---

## Frage 4)

***Was bewirkt das Constraint NOT NULL in Kombination mit einem BEFORE-Trigger? IF NEW.xxx IS NULL THEN SET NEW.xxx = ... END IF;***

Das NOT NULL constraint würde einen Fehler werfen wenn ein Datensatz mit einem NULL eingefügt werden würde. Da wir allerdings einen Trigger verwenden, der die Werte davor überprüft und auf einen default Wert setzt wenn sie NULL sind, ist das **Constraint redundant**.

---

## Frage 5)

***Kann in einem Trigger auf Datensätze einer anderen Tabelle zugegriffen werden?***

Ja, siehe Seite 3 im PDF.

---

## Frage 6)

| | | BEFORE | | | AFTER | | |
|---|---|---|---|---|---|---|---|
| | | INSERT | DELETE | UPDATE | INSERT | DELETE | UPDATE |
| NEW.xxx | lesen | ja | nein | ja | ja | nein | ja |
| | ändenr | ja | nein | ja | nein | nein | nein |
| OLD.xxx | lesen | nein | ja | ja | nein | ja | ja |
| | ändern | nein | nein | nein | nein | nein | nein |

GRÜN: There is no add row in on INSERT Trigger
ROT: Updating of OLD row is not allowed in Trigger
YELLOW: Updating of NEW row is not allowed in after Trigger
BLUE: There is no NEW row in an DELETE Trigger