LANDESBERUFSSCHULE 4 SALZBURG

Informatik

Trigger

LBS 4

01.10.2020

.

Inhalt

Lernziele und Kompetenzcheck 3

Einleitung 3

Trigger 3

Syntax Trigger 3

Beispiel: 4

Beispiel 2: 4

Lernziele und Kompetenzcheck

Ich kenne/kann

- die Funktion eines Triggers erläutern
- einen Trigger mit MySQL anlegen

Einleitung

Zur Realisierung der semantischen Integrität einer Datenbank unterstützt SQL neben der Check-Bedingung, die With-Check-Bedingung in Views. Eine weitere Möglichkeit bieten Trigger. Trigger (hinterlegter Code in der Datenbank) werden bei bestimmten Ereignissen aufgerufen. So ist es möglich Informationen vor dem Einfügen/nach dem Einfügen oder Ändern auf Gültigkeit zu prüfen oder Berechnungen etc. durführen zu lassen.

Trigger

Trigger bedeutet Auslöser und ist eine spezifische Funktion in einer Datenbank. Die Funktionen können einerseits zum Überprüfen der Integrität einer Datenbank verwendet werden, andererseits kann ein Trigger auch selbst Informationen in eine Tabelle/Feld schreiben. Trigger können bei der Verarbeitung auch „Stored Procedures“ aufrufen. MySQL unterstützt seit dem Standard SQL2003 die Verwendung von Trigger.

Syntax Trigger

Da verschiedene DBMS eigene prozedurale Sprachen verwenden, kann sich die Syntax unterscheiden. Hier wird nur die Syntax für MySQL verwendet.

```
CREATE TRIGGER Triggername
{BEFORE | AFTER} {DELETE | INSERT | UPDATE}
ON Tabellenname
[ FOR EACH STATEMENT | FOR EACH ROW]
[ WHEN Bedingung]
Anweisungen
```

Ein Trigger kann vor oder nach dem Ändern der Relation ausgeführt werden. Zusätzlich muss noch bestimmt werden, ob der Code beim Löschen, Ändern oder Einfügen neuer Informationen ausgeführt wird.

Beispiel:

Das nächste Beispiel fügt das aktuelle Datum nach dem Einfügen eines Datensatzes in das Feld Datum ein.

Achtung! Dabei wird ein vorher festgelegter Wert überschrieben.

```
delimiter $$
CREATE TRIGGER Auftragsdatum2
AFTER INSERT ON Auftrag
FOR EACH ROW
BEGIN
        Update Auftrag
        SET Datum = now();
END;$$
```

Dieser Trigger ändert das Feld Datum der Tabelle Auftrag auf das aktuelle Datum. Der Befehl FOR EACH ROW bewirkt, dass jeder Eintrag (veränderte Zeile) berücksichtigt wird. Ansonsten würde nur der erste Eintrag von dem Trigger bearbeitet.

Beispiel 2:

Es besteht eine Tabelle `postings` für Blogeinträge.

```
CREATE TABLE `postings` (
    `p_id` INT NOT NULL AUTO_INCREMENT,
    `p_subject` VARCHAR(500) NOT NULL,
    `p_bodytext` MEDIUMTEXT NOT NULL,
    `create_time` TIME NOT NULL,
     PRIMARY KEY (`p_id`)
);
```

Weiters existiert eine Tabelle log, in der das Datum, der Grund des Eintrags sowie der Autor festgehalten werden soll.

```
CREATE TABLE `log` (
    `l_id` INT(20) NOT NULL AUTO_INCREMENT,
    `l_subject` VARCHAR(500) NOT NULL,
    `creator` VARCHAR(30) NOT NULL COMMENT 'Anleger',
    `create_date` DATETIME NOT NULL,
    `posting_id` INT NOT NULL,
     PRIMARY KEY (`l_id`)
);
```

Ein Trigger soll jetzt bei jedem Eintrag in der Tabelle `postings` einen Eintrag in der Tabelle `log` durchführen. Dabei müssen Subject, der aktuelle Benutzer sowie das aktuelle Datum/Uhrzeit und die ID des Postings festgehalten werden.

Der Trigger ist relativ einfach gehalten, erfüllt aber den Zweck.

```
DELIMITER $$
CREATE TRIGGER `posting-log`
AFTER INSERT
ON `postings`
FOR EACH ROW
BEGIN
  INSERT INTO `log` (`l_subject`, `creator`, `create_date`,`posting_id`)
  VALUES(NEW.p_subject, CURRENT_USER, now(), NEW.p_id);
END$$
DELIMITER ;
```

Nach dem Einfügen eines Datensatzes muss in der Tabelle `log` der erforderliche Eintrag vorhanden sein.

```
INSERT INTO postings (`p_subject`, `p_bodytext`, `create_time`) VALUES
('erster Test', 'Das ist der Testtext', '09:10:00' );
```

Allgemein festgehalten werden Trigger zum Einhalten der ‚Business-Rules' sowie zur Sicherung der Datenintegrität verwendet.