Εργαστήριο Βάσεων Δεδομένων

Triggers

CREATE TRIGGER

Δήλωση δημιουργίας Trigger:

```
CREATE [DEFINER = { user | CURRENT_USER }]
TRIGGER trigger_name trigger_time
  trigger_event
ON tbl_name FOR EACH ROW trigger_stmt
```

- Μία σκανδάλη (trigger) είναι ένα αντικείμενο βάσης δεδομένων το οποίο σχετίζεται με ένα συγκεκριμένο πίνακα και ενεργοποιείται όταν συμβεί συγκεκριμένο γεγονός στον συγκεκριμένο πίνακα.
- Η δήλωση CREATE TRIGGER προστέθηκε στην έκδοση 5.0.2. της MySQL.
- Για την χρήση το απαιτούνται δικαιώματα SUPER χρήστη.

trigger_name

- Η σκανδάλη συσχετίζεται με τον πίνακα με το όνομα *tbl_name*
- *Το tbl_name* πρέπει να αναφέρεται σε έναν μόνιμο πίνακα.
  - Μια σκανδάλη δεν μπορεί να συσχετιστεί με έναν προσωρινό πίνακα ή όψη (view).
- Τα ονόματα των Triggers ορίζονται σε συγκεκριμένο χώρο ονομάτων(namespace) της βάσης δεδομένων
  - Οι σκανδάλες που ορίζονται στον ίδιο χώρο ονομάτων πρέπει να έχουν ξεχωριστό όνομα.
  - Οι σκανδάλες που ορίζονται σε ξεχωριστό χώρο ονομάτων μπορούν να έχουν το ίδιο όνομα

trigger_time

- *trigger_time* ορίζει τον χρόνο δράσης της σκανδάλης.
- Οι διαφορετικές τιμές που μπορεί να λάβει είναι :
  - BEFORE
    - Η σκανδάλη ενεργοποιείται πριν συμβεί το γεγονός που την πυροδοτεί
  - AFTER
    - Η σκανδάλη ενεργοποιείται μετά το γεγονός που την πυροδοτεί

- Προσοχή με τη χρήση του BEFORE. Μπορεί να εμφανιστούν περιορισμοί (ειδικά στη InnoDB), όπου ένα INSERT μπορεί να αποτύχει αλλά το trigger θα εκτελεστεί επιτυχώς.
  - Χρησιμοποιείστε BEFORE triggers κυρίως για περιορισμούς και κανόνες, αλλά όχι για δοσολοψίες.
  - Χρησιμοποιείστε AFTER triggers για τις όλες σχεδόν τις υπόλοιπες λειτουργίες.

trigger_event 1/2

- *trigger_event* καταδεικνύει το είδος του γεγονότος που ενεργοποιεί την σκανδάλη
- Το *trigger_event* μπορεί να είναι ένα από τα ακόλουθα:
  - **INSERT**: Η σκανδάλη ενεργοποιείται όταν εισάγονται νέες εγγραφές στον πίνακα με τον οποίο σχετίζεται η σκανδάλη
    - Κάτι τέτοιο μπορεί να γίνει μέσω δηλώσεων INSERT, LOAD DATA και REPLACE .
  - **UPDATE**: Η σκανδάλη ενεργοποιείται όταν κάποια εγγραφή του πίνακα με τον οποία σχετίζεται η σκανδάλη τροποποιηθεί
    - Πχ μέσω δηλώσεων UPDATE .
  - **DELETE**: Η σκανδάλη ενεργοποιείται όταν κάποια εγγραφή του πίνακα με τον οποία σχετίζεται η σκανδάλη διαγραφή.
    - κάτι τέτοιο μπορεί να γίνει μέσω δηλώσεων DELETE και REPLACE
    - Οι δηλώσεις DROP TABLE και TRUNCATE στον αντίστοιχο πίνακα δεν ενεργοποιούν την σκανδάλη διότι δεν διαγράφουν απλά εγγραφές

trigger_event 2/2

- Οι σκανδάλες που σχετίζονται με συγκεκριμένο πίνακα δεν πρέπει να έχουν το ίδιο
  - trigger_time
  - trigger_event
- Για παράδειγμα:
  - Δεν επιτρέπεται δύο σκανδάλες να περιέχουν ταυτόχρονα την δήλωση BEFORE UPDATE και να σχετίζονται με τον ίδιο πίνακα.
  - Επιτρέπεται όμως να υπάρχουν ταυτόχρονα σκανδάλες που θα περιέχουν δηλώσεις:
    - BEFORE UPDATE η πρώτη και BEFORE INSERT η δεύτερη ,
    - Ή BEFORE UPDATE η πρώτη και AFTER UPDATE η δεύτερη

trigger_stmt

- ***Trigger_stmt*** : είναι η ενέργεια που εκτελείται όταν πυροδοτείται η σκανδάλη.
- Για την εκτέλεση πολλαπλών ενεργειών χρησιμοποιείται η δομή BEGIN ... END η οποία περικλείει την λίστα των προς εκτέλεση ενεργειών
- Αυτό μας επιτρέπει να χρησιμοποιήσουμε τις ίδιες δηλώσεις που χρησιμοποιούνται εσωτερικά στις αποθηκευμένες διαδικασίες (stored procedures) όπως συνθήκες (conditionals) και βρόγχους (loops).

Ψευδώνυμα New -Old

- Μπορούμε να αναφερθούμε στις στήλες του πίνακα με τον οποίο σχετίζεται μία σκανδάλη χρησιμοποιώντας τα ψευδώνυμα OLD και NEW.
  - **OLD.*col_name*** αναφέρεται στην στήλη μίας αποθηκευμένης εγγραφής πριν αυτή αλλαχθεί ή διαγραφεί.
  - ***NEW.col_name*** αναφέρεται στην στήλη μίας καινούργιας εγγραφής που πρόκειται να εισαχθεί σε δεδομένο πίνακα ή σε μια αποθηκευμένη εγγραφή μετά την τροποποίηση της.

- Σε μία εντολή εισαγωγής (INSERT), μπορούμε να χρησιμοποιήσουμε μόνο τιμές στηλών NEW.
- Σε μία εντολή διαγραφής (DELETE), πρέπει να χρησιμοποιήσουμε το ψευδώνυμο OLD.

- Τα ψευδώνυμα OLD και NEW επιτρέπουν την προσπέλαση σε στήλες των εγγραφών που επηρεάζονται από μία σκανδάλη
- Τα ψευδώνυμα OLD και NEW δεν είναι case sensitive.

Παράδειγμα 1 1/4

- Δημιουργία πινάκων

```
CREATE TABLE test1(a1 INT);

CREATE TABLE test2(a2 INT);

CREATE TABLE test3(a3 INT NOT NULL AUTO_INCREMENT PRIMARY
KEY);

CREATE TABLE test4( a4 INT NOT NULL AUTO_INCREMENT
PRIMARY KEY, b4 INT DEFAULT 0 );
```

Παράδειγμα 1 2/4

- Δημιουργία Trigger

```
•DELIMITER |

•CREATE TRIGGER testref BEFORE INSERT ON test1
FOR EACH ROW
BEGIN

    INSERT INTO test2 SET a2 = NEW.a1;
    DELETE FROM test3 WHERE a3 = NEW.a1;
    UPDATE test4 SET b4 = b4 + 1 WHERE a4 = NEW.a1;

END;

|

•DELIMITER ;
```

Παράδειγμα 1 3/4

- Εισαγωγή δεδομένων

```
•INSERT INTO
test3 (a3) VALUES (NULL), (NULL), (NULL), (NULL), (NULL),
(NULL), (NULL), (NULL), (NULL), (NULL);

•INSERT INTO
test4 (a4) VALUES (0), (0), (0), (0), (0), (0), (0), (0),
(0), (0);

•INSERT INTO
test1 VALUES -> (1), (3), (1), (7), (1), (8), (4), (4);
```

Παράδειγμα 1 4/4

- Εξαγωγή αποτελεσμάτων

```
•mysql> SELECT * FROM test1;
+------+
|  a1  |
+------+
|  1   |
|  3   |
|  1   |
|  7   |
|  1   |
|  8   |
|  4   |
|  4   |
+------+
 8 rows in set (0.00 sec)
```

```
•mysql> SELECT * FROM test2;
+------+
|  a2  |
+------+
|  1   |
|  3   |
|  1   |
|  7   |
|  1   |
|  8   |
|  4   |
|  4   |
+------+
 8 rows in set (0.00 sec)
```

```
•mysql> SELECT * FROM test3;
+------+
|  a3  |
+------+
|  2   |
|  5   |
|  6   |
|  9   |
|  10  |
+------+
5 rows in set (0.00 sec)
```

```
•mysql> SELECT * FROM test4;
+----+------+
| a4 |  b4  |
+----+------+
| 1  |  3   |
| 2  |  0   |
| 3  |  1   |
| 4  |  2   |
| 5  |  0   |
| 6  |  0   |
| 7  |  1   |
| 8  |  1   |
| 9  |  0   |
| 10 |  0   |
+----+------+
10 rows in set (0.00 sec)
```

Διαγραφή Triggers

- `DROP TRIGGER [IF EXISTS] [schema_name.]trigger_name`

Πχ:

```
•Drop trigger AddCost;
```

•Για την χρήση του απαιτούνται δικαιώματα SUPER χρήστη.

•Χρησιμοποιούμε την δήλωση IF EXISTS για να αποτρέψουμε την πρόκληση λάθους λόγο της μη ύπαρξης ενός trigger.

•Η δήλωση IF EXISTS προστέθηκε στην έκδοση 5.1.14 της MySQL

Προβολή Triggers

- `SHOW TRIGGERS [FROM db_name] [LIKE expr]`

- Πχ:
  - Show Triggers;
  - Show Triggers like '%_%';

Using Triggers

Χρήσεις

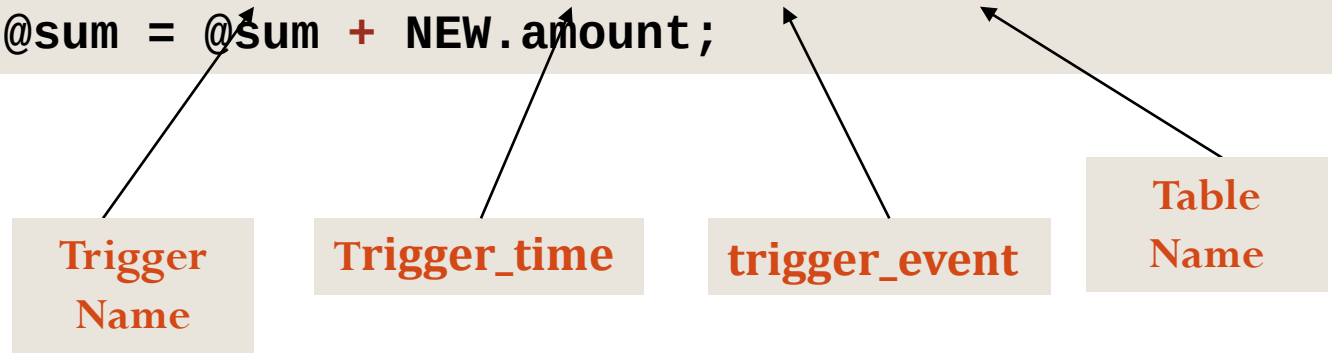
- Για την εκτέλεση ελέγχων σε τιμές που πρόκειται να εισαχθούν σε κάποιο πίνακα
- Για την εκτέλεση των υπολογισμών σε τιμές που συμμετέχουν σε μία τροποποίηση των τιμών ενός πίνακα.

Παράδειγμα 2 1/3

- Στην μεταβλητή @sum αποθηκεύεται το άθροισμα των τιμών της στήλης amount των εγγραφών που είναι αποθηκευμένες στον πίνακα account

```
•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;
```



Παράδειγμα 2 2/3

- Για να χρησιμοποιήσουμε την σκανδάλη
  - Θέτουμε την τιμή της μεταβλητής@sum ίση με μηδέν.
  - Εκτελούμε μία εντολή εισαγωγής στον πίνακα account ,
  - Προβάλουμε το περιεχόμενο της μεταβλητής και βλέπουμε πως την επηρέασε η πυροδότηση της σκανδάλης

```
•SET @sum = 0;

•INSERT INTO account VALUES (137,14.98), (141,1937.50),
(97,-100.00);

•SELECT @sum AS 'Total amount inserted';
```

@sum = 14.98 + 1937.50 – 100= 1852.48

Παράδειγμα 2 3/3

- Η δήλωση **BEFORE** καθορίζει τον χρόνο ενεργοποίησης της σκανδάλης.
  - Η σκανδάλη θα ενεργοποιηθεί πριν από την εισαγωγή εγγραφών στον αντίστοιχο πίνακα.
  - Η εναλλακτική δήλωση που μπορεί να χρησιμοποιηθεί είναι η δήλωση AFTER.
- Η δήλωση **INSERT** καταδεικνύει το γεγονός που πυροδοτεί την σκανδάλη.
  - Μια σκανδάλη μπορεί να ενεργοποιηθεί επίσης από ενέργειες DELETE και UPDATE.
- Η δήλωση που ακολουθεί την εντολή **FOR EACH ROW** καθορίζει τις εντολές που θα εκτελεστούν κάθε φορά που θα πυροδοτηθεί η σκανδάλη, η οποία συμβαίνει μία φορά για κάθε εγγραφή που επηρεάζεται από την **`trigger_stmt`**
- Στο προηγούμενο παράδειγμα το **`trigger_stmt`** είναι μια απλή ανάθεση τιμών (**SET**) σε μία μεταβλητή στην οποία αποθηκεύεται το συνολικό άθροισμα των τιμών που αποθηκεύονται σε μία στήλη συγκεκριμένου πίνακα.
- Αναφερόμαστε στην τιμή της στήλης που θα προσθέσουμε στην μεταβλητή με το ψευδώνυμο NEW.amount το οποίο σημαίνει "την τιμή της στήλης amount της τελευταίας εγγραφής που πρόκειται να αποθηκευθεί στον πίνακα **account.**

Περιορισμοί 1

- Σε μία INSERT σκανδάλη,
  - Μπορεί να χρησιμοποιηθεί μόνο το **NEW**.*col_name*.
  - Δεν υπάρχει εγγραφή **OLD**.
- Σε μία DELETE σκανδάλη,
  - Μπορεί να χρησιμοποιηθεί μόνο το **OLD**.*col_name*.
  - Δεν υπάρχει εγγραφή **NEW**.
- Σε μία UPDATE σκανδάλη, μπορούμε να χρησιμοποιήσουμε
  - OLD.*col_name* αναφέρεται στα γνωρίσματα μιας εγγραφής πριν αυτά τροποποιηθούν.
  - NEW.*col_name* αναφέρεται στα γνωρίσματα μιας εγγραφής αφού αυτά τροποποιηθούν.

Περιορισμοί 2

- Μία στήλη με όνομα OLD είναι μόνο-ανάγνωσης.
  - Μπορείτε να αναφερθείτε σε αυτό (έχοντας επαρκή δικαιώματα) αλλά δε μπορείτε να το τροποποιήσετε.
- Μία στήλη με όνομα NEW μπορεί να αναφερθεί μόνο αν έχετε επαρκή δικαιώματα.
- Σε μια BEFORE trigger, μπορείτε επίσης να αλλάξετε την τιμή NEW, εκτελώντας SET NEW.*col_name* = *value.*
- Σε μια BEFORE trigger, η τιμή NEW για μια AUTO_INCREMENT στήλη είναι 0 και όχι ο αριθμός που θα δημιουργηθεί αυτόματα όταν γίνει η εισαγωγή.

Παράδειγμα 3 1/2

- Ορίζουμε μια σκανδάλη τροποποίησης (UPDATE) η οποία ελέγχει τις τιμές που πρόκειται να τροποποιήσουν τιμές εγγραφών δεδομένου πίνακα
- Η σκανδάλη περιορίζει τις νέες τιμές στο διάστημα 0 έως 100. Αν μια τιμή είναι μεγαλύτερη από το 100 τίθεται ίση με 100, ενώ αν είναι μικρότερη από 0 τίθεται ίση με 0.
- Η συγκεκριμένη σκανδάλη έχει **BEFORE** `trigger_time` καθώς η νέα τιμή πρέπει πρώτα να ελεγχθεί, πριν χρησιμοποιηθεί για να τροποποιήσει κάποια εγγραφή.

Παράδειγμα 3 2/2

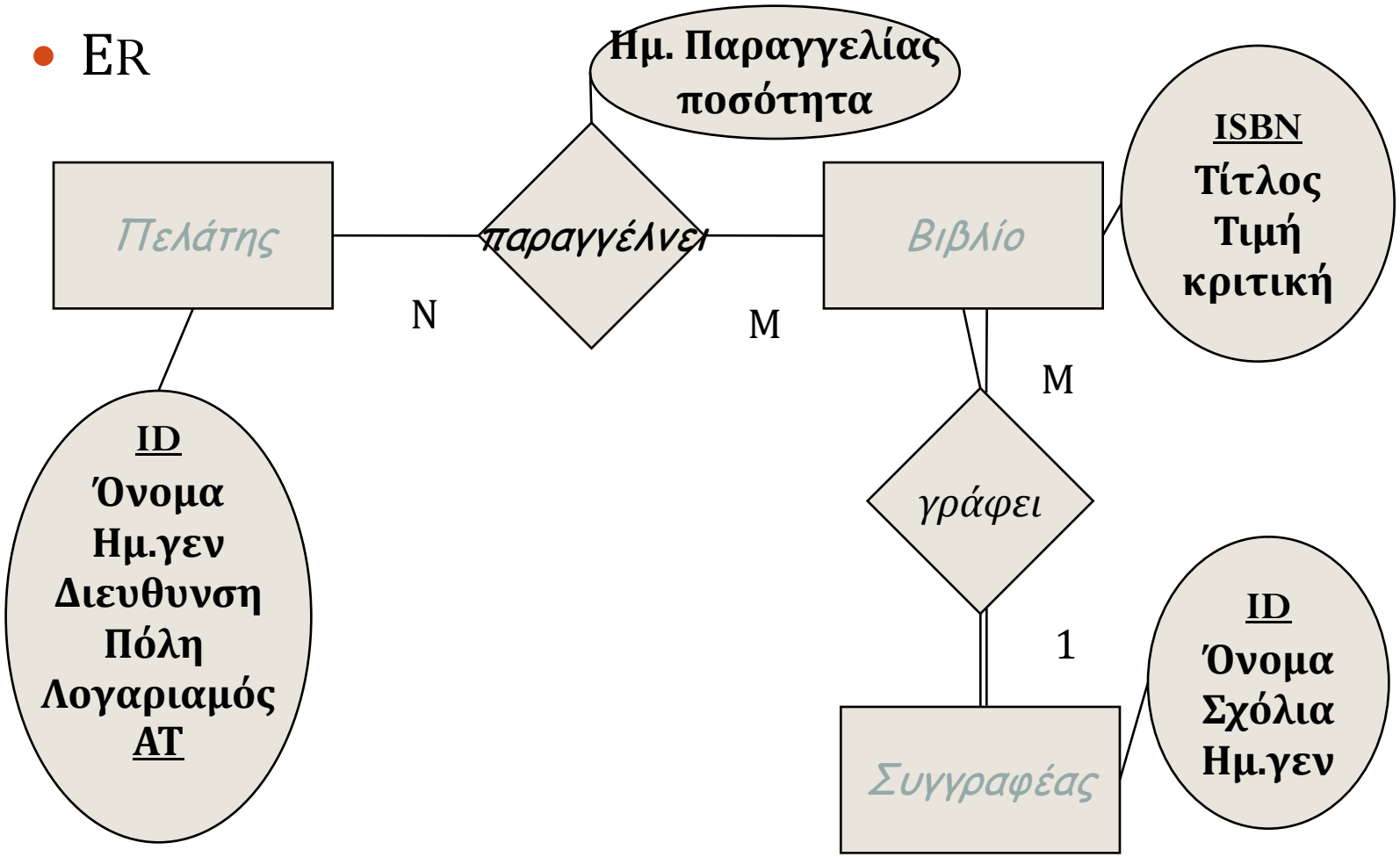
```
delimiter $
CREATE TRIGGER upd_check BEFORE UPDATE ON account
FOR EACH ROW
  BEGIN
  IF NEW.amount < 0 THEN
      SET NEW.amount = 0;
  ELSEIF NEW.amount > 100 THEN
      SET NEW.amount = 100;
  END IF;
END;$
delimiter ;
```

Χρήση Store Procedures Μέσω Triggers

- Υπάρχει η δυνατότητα να γίνει η κλήση μιας αποθηκευμένης διαδικασίας (stored procedure) κατά την πυροδότηση μιας σκανδάλης.
- Για να το επιτύχουμε πρέπει:
  - Να ορίσουμε την αποθηκευμένη διαδικασία
  - Να γίνει κλήση της αποθηκευμένης διαδικασίας με χρήση της δήλωσης CALL
- Με την συγκεκριμένη μέθοδο μπορούμε να χρησιμοποιήσουμε την ίδια αποθηκευμένη διαδικασία σε παραπάνω από μία σκανδάλη.
  - Οι αποθηκευμένες διαδικασίες μπορούν να επιστρέψουν μία τιμή στον χρήστη με χρήση OUT ή INOUT παραμέτρων

Παράδειγμα 4 1/5 E-R

- ER



Παράδειγμα 4 1/5 Create

```
CREATE TABLE `author` (
  `name` varchar(20) NOT NULL default '',
  `birthday` date NOT NULL default '0000-00-00',
  `authorID` int(11) NOT NULL auto_increment,
  `comments` longtext NOT NULL,
  PRIMARY KEY  (`authorID`)
)
ENGINE=InnoDB DEFAULT CHARSET=latin1
 AUTO_INCREMENT=1 ;


CREATE TABLE `book` (
  `ISBN` varchar(13) NOT NULL,
  `Title` varchar(30) NOT NULL default '',
  `Price` float(5,2) NOT NULL default '0.00',
  `review` longtext NOT NULL ,
  `author_ID` int(11) NOT NULL default '0',
  PRIMARY KEY  (`ISBN`)
)
ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Παράδειγμα 4 2/5 Create

```
CREATE TABLE `customer` (
  `custID` int(11) unsigned NOT NULL auto_increment,
  `name` varchar(30) NOT NULL default 'unknown',
  `Birthday` date NOT NULL default '0000-00-00',
  `address` varchar(20) NOT NULL default '',
  `city` varchar(30) NOT NULL default '',
  `AT` varchar(7) NOT NULL default '',
  `logariasmos` float(6,2) NOT NULL ,
  PRIMARY KEY  (`custID`),
  UNIQUE KEY `idcard` (`AT`)
)
ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

CREATE TABLE `request` (
  `orderdate` TIMESTAMP  NOT NULL,
  `customerID` int(11) unsigned NOT NULL ,
  `book_ISBN` varchar(13) NOT NULL ,
  `quantity` int NOT NULL NOT NULL default 1,
  primary KEY  (`book_ISBN`,`customerID`,`orderdate`)
) ;
```

Παράδειγμα 4 4/5 Trigger

```
delimiter $
 CREATE TRIGGER AddCost AFTER INSERT ON request
    FOR EACH ROW
     BEGIN
        DECLARE Pr float(6,2);
        select Price into Pr from book where ISBN=New.book_ISBN;
        UPDATE customer SET logariasmos = logariasmos + Pr*NEW.quantity WHERE customer.custID = NEW.customerID;
    END$
```

Παράδειγμα 4 5/5 Επεξηγήσεις

- **trigger_name :** AddCost
- **trigger_time :** AFTER
  - To trigger_stmt εκτελείται μετά την εισαγωγή των δεδομένων στον πίνακα request
- **trigger_event:** INSERT
  - Πυροδοτείται όταν εισάγεται μια νέα εγγραφή στον πίνακα request.
- **trigger_stmt:** Εκτελούνται πολλαπλές ενέργειες (χρήση Begin – End;)
  - Δήλωση μεταβλητής Pr τύπου float
  - Εύρεση της τιμής του βιβλίου που αγόρασε ο πελάτης και αποθήκευσή της στην μεταβλητή Pr.
  - Ανανέωση του λογαριασμού του πελάτη, προσθέτοντας το γινόμενο της τιμής του βιβλίου επί το πλήθος των βιβλίων που αγοράστηκαν