Προηγμένα Θέματα Βάσεων Δεδομένων

Διδάσκων: Άγγελος Μιχάλας

Περιεχόμενα

- Triggers
- Events

Εισαγωγή

- *Τα SQL trigger είναι ένα σύνολο από SQL statements αποθηκευμένα στον κατάλογο της βάσης δεδομένων (ένα αποθηκευμένο πρόγραμμα) το οποίο εκτελείται αυτόματα όταν εμφανιστεί ένα συγκεκριμένο συμβάν το οποίο σχετίζεται με ένα πίνακα π.χ.,* ***insert, update ή delete.***
- Χρήσιμο για:
  - Προστασία της ακεραιότητας των δεδομένων.
  - Αυτοματοποίηση διαδικασιών στη βάση δεδομένων όπως logging, auditing, κ.α.

Εισαγωγή

- Το SQL trigger είναι ένας ειδικός τύπος stored procedure.
- Είναι ειδικός γιατί δεν καλείται όπως μία stored procedure. Η κύρια διαφορά ενός trigger και μιας stored procedure είναι ότι:
  - Ένα trigger καλείται αυτόματα όταν ένα συμβάν δημιουργείται σε ένα πίνακα.
  - Μία stored procedure πρέπει να καλεστεί ρητά.

Πλεονεκτήματα

1. Παρέχουν ένα εναλλακτικό τρόπο ελέγχου της ακεραιότητας των δεδομένων.
2. Παρέχουν ένα εναλλακτικό τρόπο εκτέλεσης προγραμματισμένων ενεργειών (scheduled tasks). Τα triggers καλούνται αυτόματα πριν ή μετά από μία αλλαγή των δεδομένων σε έναν πίνακα.
3. Μπορεί να πιάσει σφάλματα στην επιχειρισιακή λογική στο επίπεδο της ΒΔ.
4. Μπορεί να ελέγχει τις αλλαγές των δεδομένων στους πίνακες.

Μειονεκτήματα

1. Τα SQL triggers μπορεί να αυξήσουν την επιβάρυνση (overhead) στον εξυπηρετητή της βάσης δεδομένων.
2. Τα SQL triggers καλούνται και εκτελούνται χωρίς να είναι ορατά από τις εφαρμογές. Είναι δύσκολο να καταλάβουν οι εφαρμογές τι γίνεται στο επίπεδο της ΒΔ.

Εάν δεν υπάρχει τρόπος να γίνει η δουλειά με stored procedure, τότε σκέψου τα SQL trigger

Μεταβολές Δεδομένων

- Ένα trigger είναι ένα σύνολο από SQL statements τα οποία καλούνται/εκτελούνται αυτόματα όταν συμβεί μια αλλαγή στα δεδομένα ενός πίνακα (ο οποίος σχετίζεται με το trigger).
- Για κάθε πίνακα μπορούν να οριστούν 6 triggers (πριν την έκδοση MySQL 5.7.2)
  - **BEFORE INSERT**: εκτελείται πριν την εισαγωγή των δεδομένων
  - **AFTER INSERT**: εκτελείται μετά την εισαγωγή των δεδομένων
  - **BEFORE UPDATE**: εκτελείται πριν την ενημέρωση των δεδομένων
  - **AFTER UPDATE**: εκτελείται μετά την ενημέρωση των δεδομένων
  - **BEFORE DELETE**: εκτελείται πριν τη διαγραφή των δεδομένων
  - **AFTER DELETE**: εκτελείται μετά τη διαγραφή των δεδομένων

Μεταβολές Δεδομένων

- Όταν δεν χρησιμοποιούνται τα INSERT, DELETE ή UPDATE statements για την μεταβολή των δεδομένων, τα triggers που σχετίζονται με τον πίνακα δε θα καλεστούν.
  - TRUNCATE statement απομακρύνει όλα τα δεδομένα ενός πίνακα αλλά δεν καλεί το σχετιζόμενο με το πίνακα trigger
- Τα triggers ενεργοποιούνται (fired) και για κάποια τα οποία χρησιμοποιούν το INSERT statement (στο background) όπως είναι το REPLACE ή LOAD DATA statement.

Συμβάσεις Ονοματοδοσίας Trigger

1. **(BEFORE | AFTER)_tableName_(INSERT| UPDATE | DELETE)**
   - before_order_update
2. **tablename_(BEFORE | AFTER)_(INSERT| UPDATE | DELETE)**
   - order_before_update

MySQL Triggers – Απόθηκευση

- Τα triggers στη MySQL αποθηκεύονται σε ένα κατάλογο δεδομένων π.χ. /data/customerproduct/ με τα αρχεία να ονομάζονται **tablename.TRG** και **triggername.TRN** :
  - Το αρχείο tablename.TRG συσχετίζει το trigger με το πίνακα.
  - Το αρχείο triggername.TRN περιέχει τον ορισμό του trigger.

MySQL Trigger – Περιορισμοί

- Τα triggers καλύπτουν όλα τα χαρακτηριστικά που ορίζονται στην standard SQL.
- Τα MySQL triggers δεν μπορούν να:
  - Χρησιμοποιήσουν τα SHOW, LOAD DATA, LOAD TABLE, BACKUP DATABASE, RESTORE, FLUSH και RETURN statements.
  - Χρησιμοποιήσουν statements όπως τα COMMIT , ROLLBACK , START TRANSACTION, LOCK/UNLOCK TABLES, ALTER, CREATE, DROP, RENAME.
  - Χρησιμοποιήσουν statements όπως τα PREPARE και EXECUTE.
  - Χρησιμοποιήσουν δυναμικά SQL statements.

Ένα trigger μπορεί να καλέσει μία stored procedure ή stored function

Δημιουργία Trigger

- Σύνταξη

```
CREATE TRIGGER trigger_name
trigger_time  trigger_event  ON  table_name
FOR EACH ROW
BEGIN
 ...
END;
```

Δημιουργία Trigger

- Ένα trigger θα πρέπει να ακολουθεί μία σύμβαση σε ότι αφορά την ονομασία: **[trigger time]_[table name]_[trigger event]**
  - E.g. before_employees_update
- Ο χρόνος ενεργοποίσης του trigger (activation time) μπορεί να είναι **BEFORE** ή **AFTER**
- Το trigger event μπορεί να είναι **INSERT**, **UPDATE** ή **DELETE**
- Ένα trigger πρέπει να σχετίζεται με ένα **πίνακα**. Χωρίς τον πίνακα, το trigger δεν θα υπάρχει. Συνεπώς θα πρέπει να καθοριστεί το όνομα του πίνακα μετά το **ON** keyword.

Δημιουργία Trigger

- Ένα trigger που ορίζεται για ενέργεια **INSERT** μπορεί να χρησιμοποιήσει μόνο το **NEW** keyword.
- Ένα trigger που ορίζεται για ενέργεια **DELETE** μπορεί να χρησιμοποιήσει μόνο το **OLD** καθώς δεν υπάρχει καμία νέα γραμμή.
- Ένα trigger που ορίζεται για ενέργεια **UPDATE** μπορεί να χρησιμοποιήσει το **OLD** (αναφέρεται στη γραμμή πριν γίνει η ενημέρωση) και το **NEW** (αναφέρεται στη γραμμή αφού γίνει η ενημέρωση) keyword.

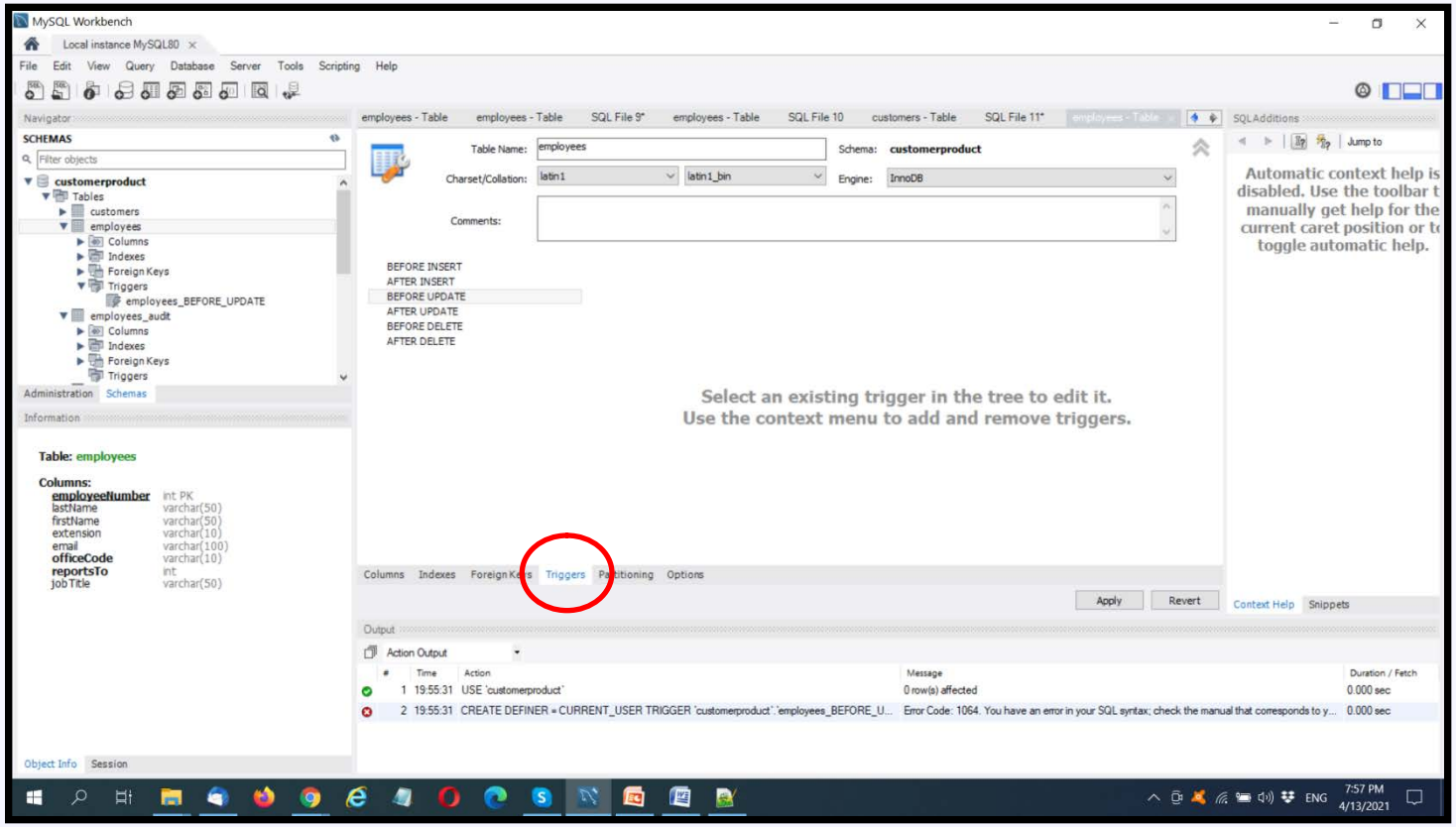
Trigger – Παράδειγμα

- Δημιουργία ενός πίνακα employees_audit που θα αποθηκεύει τις αλλαγές στον πίνακα employee

```
CREATE TABLE employees_audit (
    id INT AUTO_INCREMENT PRIMARY KEY,
    employeeNumber INT NOT NULL,
    lastname VARCHAR(50) NOT NULL,
    changedat DATETIME DEFAULT NULL,
    action VARCHAR(50) DEFAULT NULL
);
```

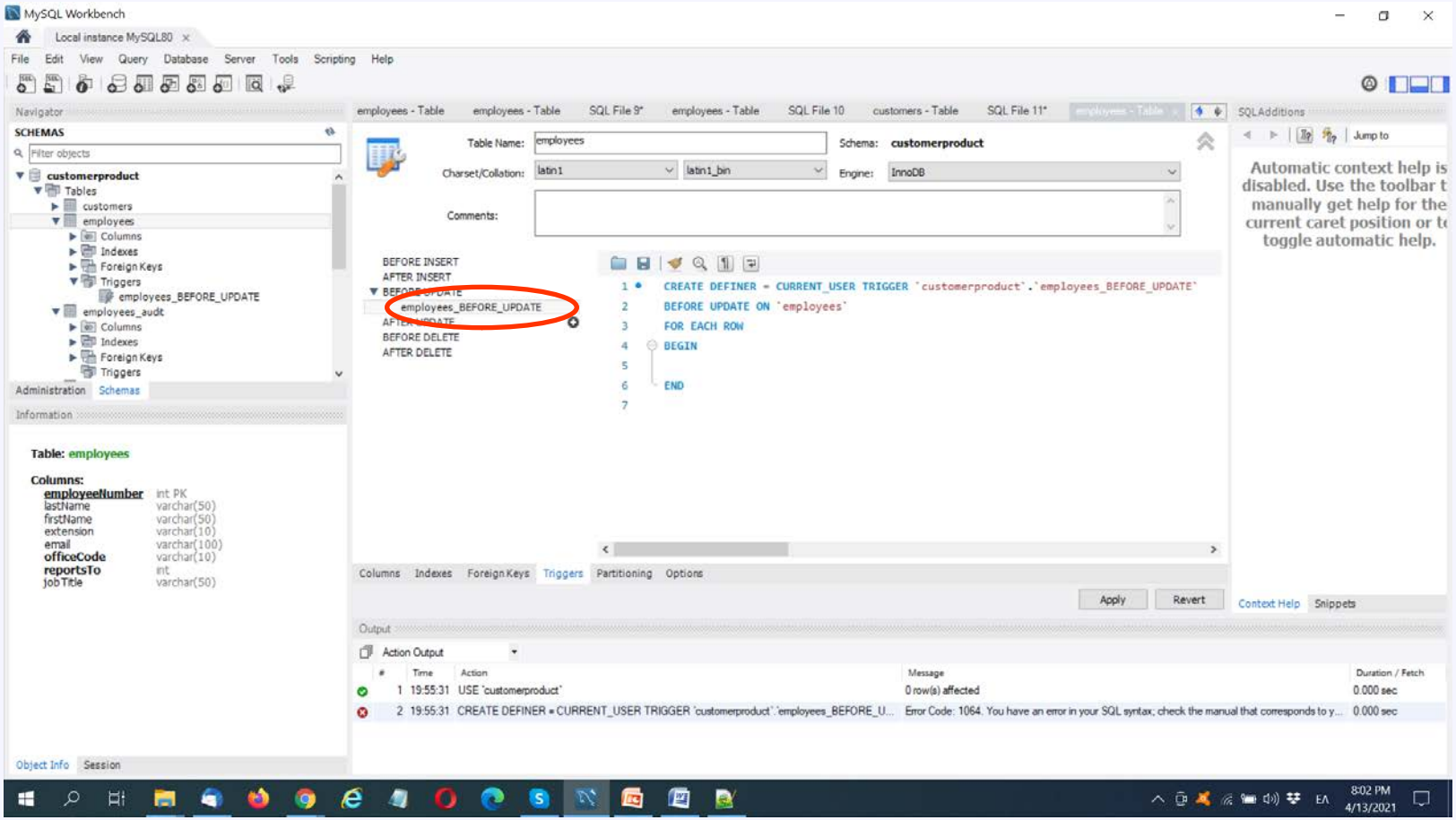
Trigger – Παράδειγμα

- Δημιουργία ενός trigger για καταγραφή των αλλαγών στον πίνακα employee
- Επιλέγω στην καρτέλα του πίνακα employee το Triggers



Trigger – Παράδειγμα

- Προσθέτω ένα νέο Trigger στο BEFORE UPDATE



Trigger – Παράδειγμα

- Συμπληρώνω τον κώδικα μεταξύ των BEGIN-END

```
CREATE TRIGGER before_employee_update
BEFORE UPDATE ON employees
FOR EACH ROW
BEGIN
    INSERT INTO employees_audit
    SET action = 'update',
     employeeNumber = OLD.employeeNumber,
     lastname = OLD.lastName,
     changedat = NOW();
END
```

Trigger – Παράδειγμα

- Update statement

```
UPDATE employees
SET lastName = 'Phan'
WHERE employeeNumber = 1056;

select * from employees_audit;
```

Πολλαπλά Triggers

- **Πολλαπλά triggers για το ίδιο event και action time** σε ένα πίνακα. Τα triggers θα ενεργοποιηθούν **σειριακά** όταν συμβεί το event.
  - MySQL 5.7.2+
- Το συντακτικό για τη δημιουργία κάθε ενός trigger είναι ίδιο.
- Η κλήση των triggers γίνεται με τη σειρά που έχουν δημιουργηθεί.

Πολλαπλά Triggers

- Για να αλλάξει η σειρά κλήσης των triggers θα πρέπει να ορίσετε την επιλογή **FOLLOWS** ή **PRECEDES** ύστερα από τον όρο FOR EACH ROW.
  - Η επιλογή FOLLOWS επιτρέπει στο νέο trigger να ενεργοποιηθεί μετά (έπεται η εκτέλεσή του) το trigger που δηλώνεται (το οποίο πρέπει να υπάρχει).
  - Η επιλογή PRECEDES επιτρέπει στο νέο trigger να ενεργοποιηθεί πριν (προηγείται η εκτέλεσή του) το trigger που δηλώνεται (το οποίο πρέπει να υπάρχει).

Πολλαπλά Triggers

- Σύνταξη

```
CREATE TRIGGER  trigger_name
[BEFORE|AFTER] [INSERT|UPDATE|DELETE] ON table_name
FOR EACH ROW
[FOLLOWS|PRECEDES] existing_trigger_name
BEGIN
…
END
```

Πολλαπλά Triggers – Παράδειγμα

- Όταν αλλάζει η τιμή ενός προιόντος (products) θα πρέπει να αποθηκεύουμε την παλιά τιμή σε ένα χωριστό πίνακα
- Αρχικά, θα δημιουργήσουμε τον πίνακα price_logs

```
CREATE TABLE price_logs (
    id INT(11) NOT NULL AUTO_INCREMENT,
    product_code VARCHAR(15) NOT NULL,
    price DOUBLE NOT NULL,
    updated_at TIMESTAMP NOT NULL,
    PRIMARY KEY (id),
    CONSTRAINT price_logs_ibfk_1 FOREIGN KEY (product_code)
    REFERENCES products (productCode)
    ON DELETE CASCADE  ON UPDATE CASCADE
);
```

Πολλαπλά Triggers – Παράδειγμα

- Θεωρούμε ότι εκτός από την παλιά τιμή θέλουμε να αποθηκεύουμε και ποιος την άλλαξε.
- Θα δημιουργήσουμε τον πίνακα user_change_logs

```
CREATE TABLE user_change_logs (
    id int(11) NOT NULL AUTO_INCREMENT,
    product_code varchar(15) DEFAULT NULL,
    updated_at timestamp NOT NULL,
    updated_by varchar(30) NOT NULL,
    PRIMARY KEY (id),
    CONSTRAINT user_change_logs_ibfk_1 FOREIGN KEY
    (product_code) REFERENCES products (productCode)
    ON DELETE CASCADE ON UPDATE CASCADE
);
```

Πολλαπλά Triggers – Παράδειγμα

- Δημιουργία πρώτου trigger που ενεργοποιείται πριν την ενημέρωση του πίνακα products και θα ενημερώνει το πίνακα price_logs

```
CREATE TRIGGER before_products_update_1
BEFORE UPDATE ON products
FOR EACH ROW
BEGIN
    INSERT INTO price_logs(product_code,price)
    VALUES(old.productCode,old.msrp);
END
```

Πολλαπλά Triggers – Παράδειγμα

- Δημιουργία δεύτερου trigger που ενεργοποιείται πριν την ενημέρωση του πίνακα products και θα ενημερώνει το πίνακα user_change_logs

```
CREATE TRIGGER before_products_update_2
BEFORE UPDATE ON products
FOR EACH ROW FOLLOWS before_products_update_1
BEGIN
  INSERT INTO user_change_logs(product_code,updated_by)
  VALUES(old.productCode,user());
END
```

Πολλαπλά Triggers – Παράδειγμα

- Update statement

```
UPDATE products
SET msrp = 95.3
WHERE productCode = 'S10_1678';
```

Πληροφορίες για Triggers

- Σύνταξη SHOW TRIGGERS

  SHOW TRIGGERS [FROM|IN] database_name [LIKE expr | WHERE expr];

  Π.χ. SHOW TRIGGERS FROM customerproduct;

- Πληροφορίες σχετικά με τη σειρά ενεργοποίησης θα πρέπει να γράψετε επερώτηση στο πίνακα triggers.
- Σύνταξη

  SELECT *

  FROM information_schema.triggers

  WHERE trigger_schema = ‘database name';

Πληροφορίες για Triggers

- Παράδειγμα

```
SELECT trigger_name, action_order
FROM information_schema.triggers
WHERE trigger_schema = 'customerproduct'
ORDER BY event_object_table, action_timing, event_manipulation;
```

- Κατάργηση trigger

```
DROP TRIGGER table_name.trigger_name;
```

Events

- Ένα event είναι μία εργασία (task) που εκτελείται βασισμένη σε ένα προκαθορισμένο πρόγραμμα (schedule)
- Ενεργοποιείται από το χρόνο (time), και όχι από την ενημέρωση κάποιου πίνακα κ.τ.λ όπως γίνεται με τα triggers
- Είναι παρόμοιο με το cron job στο UNIX και το task scheduler στα Windows
- Χρησιμοποιείται για:
  1. Βελτιστοποίηση των πινάκων
  2. cleaning up logs,
  3. archiving data,
  4. generate complex reports

Events

Διαμόρφωση/σχηματισμός (Configuration)

- Το thread χρονοπρογραμματισμού των events δεν είναι ενεργοποιημένο
  - SET GLOBAL event_scheduler = ON;
  - SHOW PROCESSLIST;

Create Event

- Σύνταξη

  CREATE EVENT [IF NOT EXIST] event_name

  ON SCHEDULE schedule

  DO

  event_body

- Για one-time event (στη θέση schedule)
  - **AT timestamp [+ INTERVAL]**
- Για επαναλαμβανόμενο event (στη θέση schedule)
  - **EVERY interval STARTS timestamp [+INTERVAL] ENDS timestamp [+INTERVAL]**

Create Event – Παράδειγμα

- Δημιουργούμε τον πίνακα messages στον οποίο εισάγονται μηνύματα

```
CREATE TABLE IF NOT EXISTS messages (
    id INT PRIMARY KEY AUTO_INCREMENT,
    message VARCHAR(255) NOT NULL,
    created_at DATETIME NOT NULL
);
```

Create Event – Παράδειγμα

- Δημιουργία ενός event που εκτελείται μία φορά

```
CREATE EVENT IF NOT EXISTS event_01
ON SCHEDULE AT CURRENT_TIMESTAMP
ON COMPLETION PRESERVE
DO
  INSERT INTO messages(message,created_at)
  VALUES('Test MySQL Event 1',NOW());
```

Create Event – Παράδειγμα

- Δημιουργία ενός event που εκτελείται πολλές φορές

```
CREATE EVENT event_02
ON SCHEDULE EVERY 1 MINUTE
STARTS CURRENT_TIMESTAMP
ENDS CURRENT_TIMESTAMP + INTERVAL 1 HOUR
ON COMPLETION PRESERVE
DO
  INSERT INTO messages(message,created_at)
  VALUES('Test MySQL recurring Event',NOW());
```

Εμφάνιση/Κατάργηση ενός Event

- Εμφάνιση

SHOW EVENTS FROM database schema;

- Κατάργηση

DROP EVENT [IF EXIST] event_name;