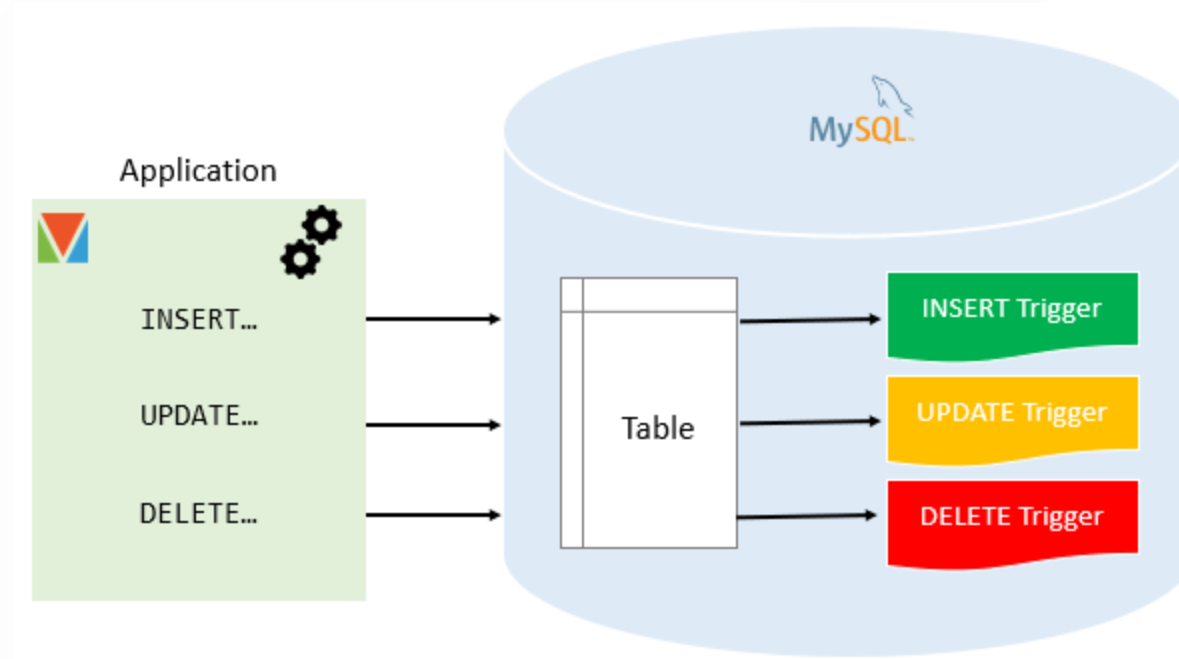


Lab6 Triggers, Stored Procedures / Functions

Lab6 Agenda

- Triggers
- Stored Procedures / Functions
- Εργαστηριακές Ασκήσεις
- Εξαμηνιαία Εργασία

Triggers



- MySQL supports only row-level triggers
- check your RDBMS capabilities

what is a Trigger

- if some specific action occurs, you want that action to cause another action(s)
- a trigger is an action or event that causes another event to occur
- stored procedure invoked automatically when a event occurs
- e.g.
 - after inserting, editing, or deleting a table row
 - log the action
 - after placing an order
 - change product status in the inventory table to reserved
 - execute a command/ shell script

Sakila DB Triggers

```
>SHOW TRIGGERS where `Table`='film'\G
***** 1. row *****
      Trigger: ins_film
      Event: INSERT
      Table: film
      Statement: BEGIN
      INSERT INTO film_text (film_id, title, description)
      VALUES (new.film_id, new.title, new.description);
      END
      Timing: AFTER
      Created: 2023-02-24 22:03:57.83
      sql_mode: STRICT_TRANS_TABLES,STRICT_ALL_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,TRADITIONAL,NO_ENGINE_SUBSTITUTION
      Definer: root@localhost
      character_set_client: utf8mb4
      collation_connection: utf8mb4_0900_ai_ci
      Database Collation: utf8mb4_0900_ai_ci
***** 2. row *****
      Trigger: upd_film
      Event: UPDATE
```

Sakila DB Triggers

The screenshot displays a database management interface with the following components:

- Navigator Panel (Left):** Shows the database structure. Under the 'sakila' schema, the 'film' table is selected. The 'Triggers' folder under 'film' is expanded, showing 'ins_film', 'upd_film', and 'del_film'.
- Table Properties Panel (Top Right):** Shows details for the 'film' table, including 'Table Name: film', 'Charset/Collation: utf8', and 'Engine: InnoDB'.
- Trigger List (Middle Left):** A list of triggers for the 'film' table: 'BEFORE INSERT', 'AFTER INSERT' (selected), 'BEFORE UPDATE', 'AFTER UPDATE', 'BEFORE DELETE', and 'AFTER DELETE'.
- SQL Editor (Bottom Right):** Contains the SQL code for creating the 'ins_film' trigger:

```
1 CREATE DEFINER='root'@'localhost' TRIGGER `ins_film` AFTER INSERT ON `film` FOR EACH ROW BEGIN
2     INSERT INTO film_text (film_id, title, description)
3     VALUES (new.film_id, new.title, new.description);
4 END
```

a closer look

```
TRIGGER `ins_film` AFTER INSERT ON `film` FOR EACH ROW
BEGIN
    INSERT INTO film_text (film_id, title, description)
        VALUES (new.film_id, new.title, new.description);
END
```

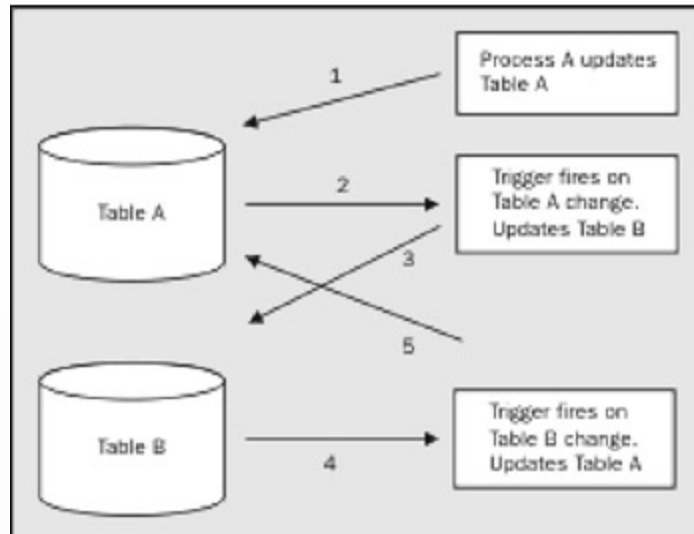
```
TRIGGER `upd_film` AFTER UPDATE ON `film` FOR EACH ROW
BEGIN
    IF (old.title != new.title) or (old.description != new.description)
    THEN
        UPDATE film_text
            SET title=new.title, description=new.description, film_id=new.film_id
            WHERE film_id=old.film_id;
    END IF;
END
```

Good Practice

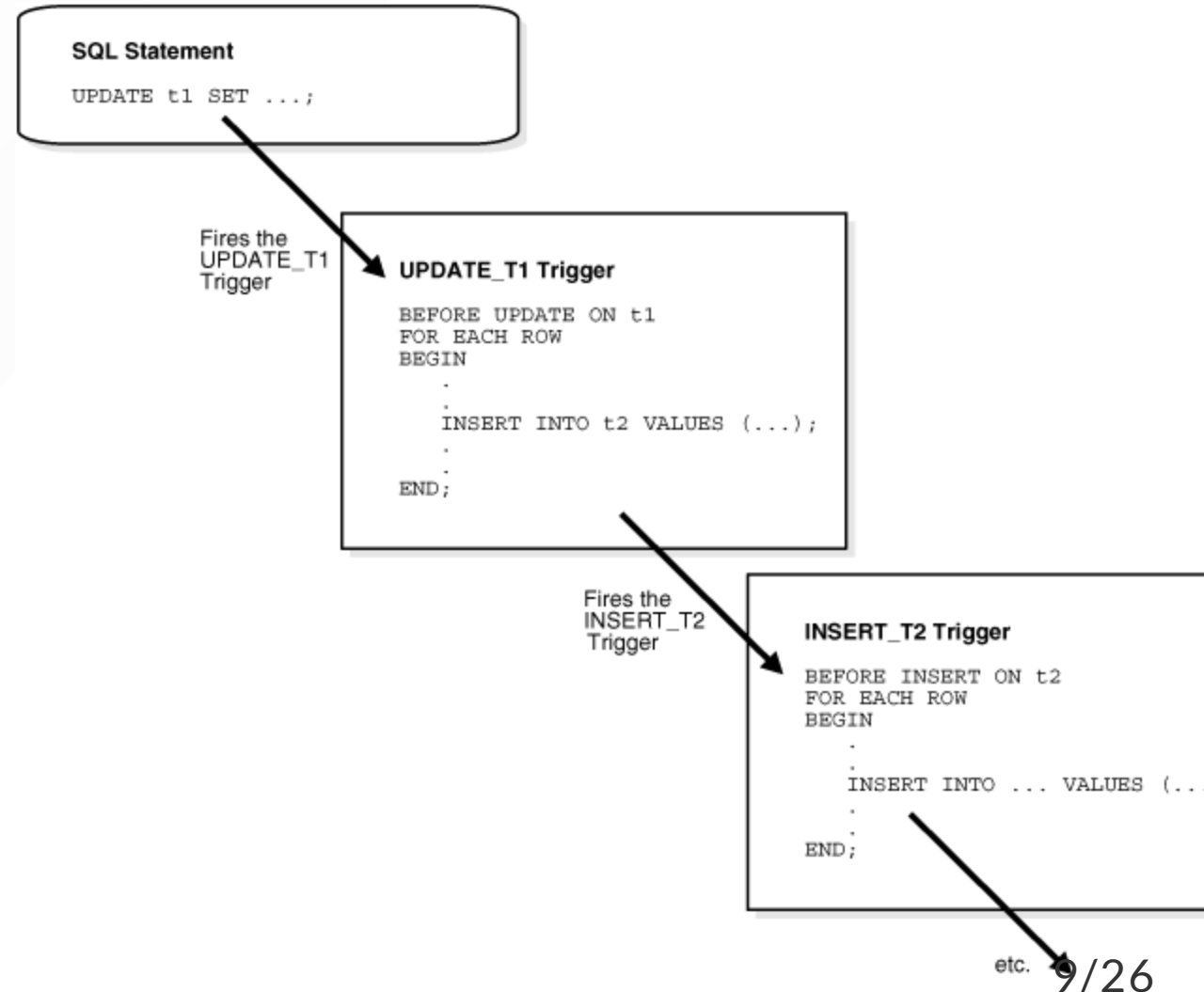
- keep the trigger as short as possible to execute quickly
 - locks held on the underlying tables
- use triggers to enforce business rules

don't overcomplicate things

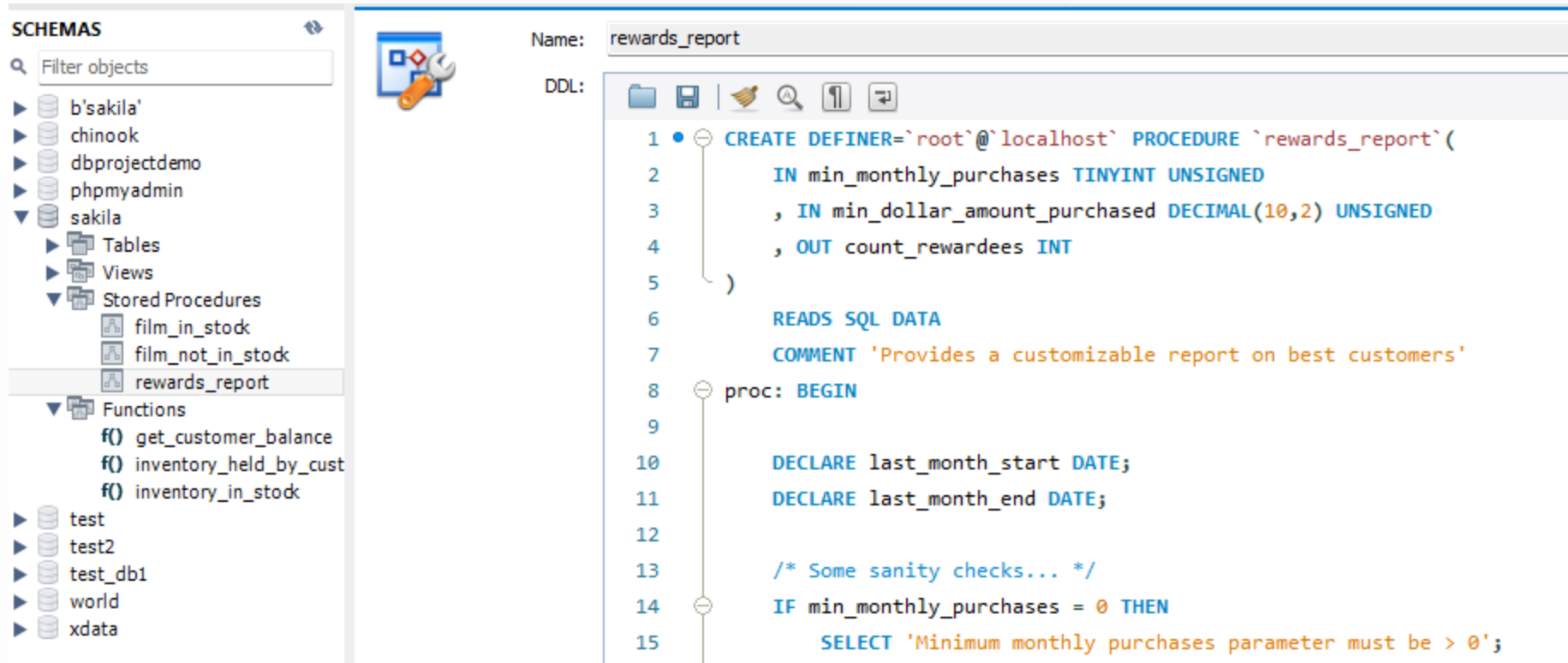
endless loop



Cascading Triggers



Stored Procedures / Functions



The screenshot displays a database management interface. On the left, a 'SCHEMAS' tree shows a hierarchy of databases including 'sakila', which is expanded to show 'Stored Procedures'. The 'rewards_report' procedure is selected. The main editor on the right shows the DDL for this procedure. The code defines a procedure that takes two input parameters: 'min_monthly_purchases' (TINYINT UNSIGNED) and 'min_dollar_amount_purchased' (DECIMAL(10,2) UNSIGNED), and returns an output parameter 'count_rewardees' (INT). It includes a comment describing its purpose and a sanity check for the first parameter.

```
1 CREATE DEFINER='root'@'localhost' PROCEDURE `rewards_report`(  
2     IN min_monthly_purchases TINYINT UNSIGNED  
3     , IN min_dollar_amount_purchased DECIMAL(10,2) UNSIGNED  
4     , OUT count_rewardees INT  
5 )  
6 READS SQL DATA  
7 COMMENT 'Provides a customizable report on best customers'  
8 proc: BEGIN  
9  
10    DECLARE last_month_start DATE;  
11    DECLARE last_month_end DATE;  
12  
13    /* Some sanity checks... */  
14    IF min_monthly_purchases = 0 THEN  
15        SELECT 'Minimum monthly purchases parameter must be > 0';
```

Stored Procedures

- `SHOW PROCEDURE STATUS`
- `SHOW CREATE PROCEDURE yourProcName`
- reside in the database
- invoked with a **CALL** statement
- can have arguments
- use procedures to implement bussiness logic

Sakila film_in_stock Stored Procedure

- The film_in_stock stored procedure is used to determine whether any copies of a given film are in stock at a given store
[<https://dev.mysql.com/doc/sakila/en/sakila-structure-procedures.html>]
- Parameters: p_film_id, p_store_id, p_film_count (OUT Parameter)

Stored Procedure Call

```
call film_in_stock(1,1,@out);
```

```
+-----+
```

```
| inventory_id |
```

```
+-----+
```

```
|          1 |
```

```
|          2 |
```

```
|          3 |
```

```
|          4 |
```

```
+-----+
```

```
MariaDB [sakila]> select @out;
```

```
+-----+
```

```
| @out |
```

```
+-----+
```

```
|     4 |
```

```
+-----+
```

Stored Functions

- `SHOW FUNCTION STATUS\G;`
- `SHOW CREATE FUNCTION yourFunctionName`
- invoked with a function call
- can replace an argument of an SQL statement

```
○ MariaDB [sakila]> SELECT inventory_held_by_customer(9);  
+-----+  
| inventory_held_by_customer(9) |  
+-----+  
|                366          |  
+-----+
```

1. Add a trigger that will make sure that foreign films are inserted with the original_language_id.
2. Create a stored procedure that returns title, description, release_year, rating for each film
3. Create a stored procedure that returns title, description, release_year, rating for each film based on rating (input parameter)
4. Create a stored procedure that returns the count of films that have a PG-13 rating (output parameter)
5. Create a stored procedure that returns the count of films based on rating (input/output parameter).
6. Study MYSQL [built in functions](#)
7. Write a query to return the month portion of the current date.
8. Study sakila inventory_in_stock, inventory_held_by_customer and get_customer_balance functions

Εξαμηνιαία Εργασία

- Database Schema Design
 1. Start thinking about the entities you need
 - Identify entities, attributes and relationships from the problem description
 - identify cardinality ratios of the relationships found
 2. Design an E/R diagram for your database
 - Look for any issues that are apparent in the E/R diagram

Εξαμηνιαία Εργασία

- Materialize Schema: DDL statements
 1. Create your tables
 - create a table for each entity
 - a table (representing an entity) should have:
 - a column for each attribute, with appropriate data type
 - a primary key and possibly some candidate keys
 - include a foreign key (one-to-many relationships)
 - add indexes & constraints to your tables
 2.
 - Create views as needed
 - Create triggers for your tables
 - Create Stored Procedures & Functions for your application

Εξαμηνιαία Εργασία

- Add Information to the Database: DML script
 - Populate the database with data

“ hint: start running your SQL commands from a separate file. This makes it much easier to alter and change your SQL code ”

Wrap Up

1. [x] Triggers
2. [x] Stored Procedures / Functions
3. [x] Εργαστηριακές Ασκήσεις
4. [x] Εξαμηνιαία Εργασία

Wrap Up

 Απορίες <https://discord.gg/g3fFxWVPfD>

Εργαστηριακές Ασκήσεις / Απαντήσεις

1. Add a trigger that will make sure that foreign films are inserted with the original_language_id.

```
CREATE DEFINER='root'@'localhost' TRIGGER `ins_validate_language` BEFORE INSERT ON `film` FOR EACH ROW
BEGIN
  IF NEW.language_id != 1 AND NEW.original_language_id IS NULL
  THEN
    SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Original language id is required for Foreign language films.';
  END IF;
END
```

```
MariaDB [sakila]> INSERT INTO film (title, language_id) values('my cool movie',2);
ERROR 1644 (45000): Original language id is required for Foreign language films.
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

2. Create a stored procedure that returns title, description, release_year, rating for each film

```
CREATE PROCEDURE `filmDetails`()  
BEGIN  
    select title,description,release_year,rating from film;  
END
```

```
MariaDB [sakila]> call filmDetails();
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

3. Create a stored procedure that returns title, description, release_year, rating for each film based on rating (input parameter)

```
CREATE PROCEDURE `filmDetailsRating` (IN myrating varchar(50))  
BEGIN  
    select title,description,release_year,rating from film where rating=myrating;  
END
```

```
MariaDB [sakila]> call filmDetailsRating('f');  
Empty set (0.00 sec)
```

```
Query OK, 0 rows affected (0.00 sec)
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

4. Create a stored procedure that returns the count of the films that have a PG-13 rating (output parameter)

```
CREATE PROCEDURE `filmsCountonRating` (OUT filmCount int)
BEGIN
    select count(title) INTO filmCount from film where rating='PG-13';
END
```

```
MariaDB [sakila]>CALL pg13films(@myCount);
Query OK, 1 row affected (0.00 sec)
MariaDB [sakila]>select @myCount;
+-----+
| @myCount |
+-----+
|      223 |
+-----+
1 row in set (0.00 sec)
```


Εργαστηριακές Ασκήσεις / Απαντήσεις

5. Create a stored procedure that returns the count of films based on rating

```
CREATE PROCEDURE `filmsCountonRating` (OUT filmCount int, IN myrating varchar(50))
BEGIN
    select count(title) INTO filmCount from film where rating=myrating ;
END
```

```
MariaDB [sakila]>CALL filmsCountonRating(@myCount, 'PG-13');
Query OK, 1 row affected (0.00 sec)
MariaDB [sakila]>select @myCount;
+-----+
| @myCount |
+-----+
|      223 |
+-----+
1 row in set (0.00 sec)
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

7. Write a query to return the month portion of the current date.

```
SELECT EXTRACT(MONTH FROM CURRENT_DATE());
```