Міністерство освіти і науки

Національний університет „Львівська політехніка”



**Звіт**

з лабораторної роботи №5

з дисципліни: “ Організація баз даних та знань”

Виконав:

Ст. гр. ІР-25

Баланик Б. В.

**Львів**

**2023**

**Порядок виконання роботи**

На базі попередньої роботи (back-end with Flask) слід написати для існуючої БД ряд програмних конструкцій (тригери, процедури, функції, курсори). Для збережуваних процедур забезпечити їхній виклик за допомогою контролерів бекенду.

1. Додати до БД 1 додаткову довільну таблицю і зв’язати з іншою існуючою таблицею зв’язком  1:M. Однак для забезпечення цілісності значень використати **тригери** замість фізичного зовнішнього ключа.

-- Additional Table: `eurosport`.`activity\_log`

CREATE TABLE IF NOT EXISTS `eurosport`.`activity\_log` (

  `log\_id` INT NOT NULL AUTO\_INCREMENT,

  `activity\_name` VARCHAR(100) NULL,

  `client\_id` INT NOT NULL,

  `timestamp` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

  PRIMARY KEY (`log\_id`),

  INDEX `fk\_activity\_log\_client1\_idx` (`client\_id` ASC) VISIBLE)

ENGINE = InnoDB;

-- Trigger to maintain data integrity without a physical foreign key

CREATE TRIGGER before\_insert\_activity\_log

BEFORE INSERT ON `eurosport`.`activity\_log`

FOR EACH ROW

BEGIN

   DECLARE client\_exists INT;

   SELECT COUNT(\*) INTO client\_exists FROM `eurosport`.`client` WHERE id = NEW.client\_id;

   IF client\_exists = 0 THEN

     SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Referenced client does not exist';

   END IF;

END;

**Збережені процедури**:

* 1. Забезпечити параметризовану вставку нових значень у довільну таблицю.

CREATE PROCEDURE InsertIntoClient (IN \_name VARCHAR(45), IN \_surname VARCHAR(45), IN \_phone VARCHAR(12), IN \_membership\_card\_type\_id INT)

BEGIN

  INSERT INTO `eurosport`.`client` (`name`, `surname`, `phone`, `membership\_card\_type\_id`) VALUES (\_name, \_surname, \_phone, \_membership\_card\_type\_id);

END;

* 1. Забезпечити реалізацію зв’язку М:М між 2ма таблицями, тобто вставити в стикувальну таблицю відповідну стрічку за реально-існуючими значеннями (напр. surname, name) в цих основних таблицях.

CREATE PROCEDURE InsertTrainerService(IN \_trainer\_name VARCHAR(45), IN \_service\_name VARCHAR(45))

BEGIN

  DECLARE \_trainer\_id INT;

  DECLARE \_service\_id INT;

  SELECT id INTO \_trainer\_id FROM `eurosport`.`trainer` WHERE `name` = \_trainer\_name;

  SELECT id INTO \_service\_id FROM `eurosport`.`service` WHERE `name` = \_service\_name;

  IF \_trainer\_id IS NOT NULL AND \_service\_id IS NOT NULL THEN

    INSERT INTO `eurosport`.`trainer\_service` (`trainer\_id`, `service\_id`) VALUES (\_trainer\_id, \_service\_id);

  ELSE

    SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Trainer or Service does not exist';

  END IF;

END;

* 1. Створити пакет, який вставляє 10 стрічок у довільну таблицю БД у форматі < Noname+№> , наприклад:Noname5, Noname6, Noname7 і т.д.

CREATE PROCEDURE InsertClientsWithPattern()

BEGIN

  DECLARE i INT DEFAULT 1;

  WHILE i <= 10 DO

    INSERT INTO `eurosport`.`client` (`name`, `surname`, `phone`, `membership\_card\_type\_id`) VALUES (CONCAT('Noname', i), 'Surname', '0000000000', 1);

    SET i = i + 1;

  END WHILE;

END;

* 1. Написати **користувацьку функцію**, яка буде шукати Max, Min, Sum чи Avg для стовпця довільної таблиці у БД. Написати процедуру, яка буде у SELECT викликати цю функцію.

CREATE FUNCTION AggregateOperation(tableName VARCHAR(100), columnName VARCHAR(100), operationType VARCHAR(10))

RETURNS DOUBLE

BEGIN

  SET @s = CONCAT('SELECT ', operationType, '(', columnName, ') FROM ', tableName);

  PREPARE stmt FROM @s;

  EXECUTE stmt;

  DEALLOCATE PREPARE stmt;

  RETURN @result;

END;

CREATE PROCEDURE GetAggregateResult(IN tableName VARCHAR(100), IN columnName VARCHAR(100), IN operationType VARCHAR(10))

BEGIN

  SELECT AggregateOperation(tableName, columnName, operationType) AS Result;

END;

* 1. Написати 1 процедуру із **курсором** для виконання однієї із наступних задач:
     1. Використовуючи курсор, забезпечити динамічне створення 2х таблиць з іменами що містять штамп часу, структура таблиць ідентична будь-якій структурі таблиці БД. Після чого випадковим чином пострічково скопіювати стрічки із батьківської таблиці або в одну, або в іншу додаткові таблиці. Повторний запуск процедури знову створює нові аналогічні таблиці, в яких випадковим чином знову будуть розкинуті дані з батьківської таблиці.

CREATE PROCEDURE DynamicTableCreationFromClient()

BEGIN

  DECLARE finished INT DEFAULT 0;

  DECLARE \_id INT;

  DECLARE \_name VARCHAR(45);

  DECLARE \_surname VARCHAR(45);

  DECLARE \_phone VARCHAR(12);

  DECLARE \_membership\_card\_type\_id INT;

  DECLARE cur CURSOR FOR SELECT `id`, `name`, `surname`, `phone`, `membership\_card\_type\_id` FROM `eurosport`.`client`;

  DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

  SET @table1 = CONCAT('client\_copy1\_', UNIX\_TIMESTAMP());

  SET @table2 = CONCAT('client\_copy2\_', UNIX\_TIMESTAMP());

  SET @s1 = CONCAT('CREATE TABLE ', @table1, ' LIKE `eurosport`.`client`');

  PREPARE stmt1 FROM @s1;

  EXECUTE stmt1;

  DEALLOCATE PREPARE stmt1;

  SET @s2 = CONCAT('CREATE TABLE ', @table2, ' LIKE `eurosport`.`client`');

  PREPARE stmt2 FROM @s2;

  EXECUTE stmt2;

  DEALLOCATE PREPARE stmt2;

  OPEN cur;

  read\_loop: LOOP

    FETCH cur INTO \_id, \_name, \_surname, \_phone, \_membership\_card\_type\_id;

    IF finished = 1 THEN

      LEAVE read\_loop;

    END IF;

    IF RAND() < 0.5 THEN

      SET @s = CONCAT('INSERT INTO ', @table1, ' VALUES (', \_id, ', "', \_name, '", "', \_surname, '", "', \_phone, '", ', \_membership\_card\_type\_id, ')');

    ELSE

      SET @s = CONCAT('INSERT INTO ', @table2, ' VALUES (', \_id, ', "', \_name, '", "', \_surname, '", "', \_phone, '", ', \_membership\_card\_type\_id, ')');

    END IF;

    PREPARE stmt FROM @s;

    EXECUTE stmt;

    DEALLOCATE PREPARE stmt;

  END LOOP;

  CLOSE cur;

END;

* + 1. Використовуючи курсор, забезпечити динамічне створення таблиць з назвами+штамп часу, взятими зі стовпця з довільної таблиці БД, з випадковою кількістю стовпців (від 1 до 9). Імена та тип стовпців довільні.
    2. Використовуючи курсор, забезпечити динамічне створення баз даних з іменами, взятими зі стовпця з довільної таблиці поточної БД, з випадковою кількістю таблиць для кожної БД (від 1 до 9). Структура таблиць довільна. Імена таблиць відповідають імені БД з порядковим номером від 1 до 9.

CREATE PROCEDURE DynamicTableCreation()

BEGIN

  DECLARE done INT DEFAULT FALSE;

  DECLARE \_name VARCHAR(45);

  DECLARE \_surname VARCHAR(45);

  DECLARE \_phone VARCHAR(12);

  DECLARE \_membership\_card\_type\_id INT;

  DECLARE cur CURSOR FOR SELECT `name`, `surname`, `phone`, `membership\_card\_type\_id` FROM `eurosport`.`client`;

  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

  SET @table1 = CONCAT('client\_', UNIX\_TIMESTAMP());

  SET @table2 = CONCAT('client\_', UNIX\_TIMESTAMP() + 1);

  SET @s1 = CONCAT('CREATE TABLE ', @table1, ' LIKE `eurosport`.`client`');

  PREPARE stmt1 FROM @s1;

  EXECUTE stmt1;

  DEALLOCATE PREPARE stmt1;

  SET @s2 = CONCAT('CREATE TABLE ', @table2, ' LIKE `eurosport`.`client`');

  PREPARE stmt2 FROM @s2;

  EXECUTE stmt2;

  DEALLOCATE PREPARE stmt2;

  OPEN cur;

  read\_loop: LOOP

    FETCH cur INTO \_name, \_surname, \_phone, \_membership\_card\_type\_id;

    IF done THEN

      LEAVE read\_loop;

    END IF;

    IF RAND() < 0.5 THEN

      SET @s = CONCAT('INSERT INTO ', @table1, ' VALUES (NULL, "', \_name, '", "', \_surname, '", "', \_phone, '", ', \_membership\_card\_type\_id, ')');

    ELSE

      SET @s = CONCAT('INSERT INTO ', @table2, ' VALUES (NULL, "', \_name, '", "', \_surname, '", "', \_phone, '", ', \_membership\_card\_type\_id, ')');

    END IF;

    PREPARE stmt FROM @s;

    EXECUTE stmt;

    DEALLOCATE PREPARE stmt;

  END LOOP;

  CLOSE cur;

END;

1. Написати 3 довільні **тригери** для таблиць поточної БД, як приклад можна взяти наступні:
   1. Значення певного стовпця не може закінчувати двома нулями

CREATE TRIGGER prevent\_two\_zeros\_in\_phone

BEFORE INSERT OR UPDATE ON `eurosport`.`client`

FOR EACH ROW

BEGIN

  IF RIGHT(NEW.phone, 2) = '00' THEN

    SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Phone number cannot end with two zeros';

  END IF;

END;

* 1. Заборонити будь-яку модифікацію даних в таблиці

CREATE TRIGGER prohibit\_data\_modification

BEFORE UPDATE ON `eurosport`.`client`

FOR EACH ROW

BEGIN

  SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Modification of data in the client table is prohibited';

END;

* 1. Заборонити видалення стрічок з таблиці

CREATE TRIGGER prohibit\_row\_deletion

BEFORE DELETE ON `eurosport`.`equipment`

FOR EACH ROW

BEGIN

  SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Deletion of rows from the equipment table is prohibited';

END;

* 1. Забезпечити мінімальну кардинальність 6 стрічок для певної  таблиці БД

CREATE TRIGGER ensure\_min\_cardinality

BEFORE DELETE ON `eurosport`.`service`

FOR EACH ROW

BEGIN

  DECLARE row\_count INT;

  SELECT COUNT(\*) INTO row\_count FROM `eurosport`.`service`;

  IF row\_count <= 6 THEN

    SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'The service table must contain at least 6 rows';

  END IF;

END;

* 1. Забезпечити кардинальність (min=2, max=6) стрічок для певної  таблиці БД

CREATE TRIGGER ensure\_min\_service\_rows

BEFORE DELETE ON `eurosport`.`service`

FOR EACH ROW

BEGIN

  DECLARE row\_count INT;

  SELECT COUNT(\*) INTO row\_count FROM `eurosport`.`service`;

  IF row\_count <= 2 THEN

    SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Minimum 2 rows required in service table';

  END IF;

END;

CREATE TRIGGER ensure\_max\_service\_rows

BEFORE INSERT ON `eurosport`.`service`

FOR EACH ROW

BEGIN

  DECLARE row\_count INT;

  SELECT COUNT(\*) INTO row\_count FROM `eurosport`.`service`;

  IF row\_count >= 6 THEN

    SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Maximum 6 rows allowed in service table';

  END IF;

END;

* 1. Створити таблицю-журнал, в якій вести логи зі штампом часу при видаленні даних для певної таблиці

CREATE TABLE `eurosport`.`trainer\_deletion\_log` (

  `log\_id` INT NOT NULL AUTO\_INCREMENT,

  `trainer\_id` INT NULL,

  `deletion\_time` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

  PRIMARY KEY (`log\_id`))

ENGINE = InnoDB;

CREATE TRIGGER log\_trainer\_deletion

AFTER DELETE ON `eurosport`.`trainer`

FOR EACH ROW

BEGIN

  INSERT INTO `eurosport`.`trainer\_deletion\_log` (`trainer\_id`) VALUES (OLD.id);

END;

* 1. Створити таблицю-журнал, в якій вести логи зі штампом часу при модифікації даних для таблиці

CREATE TABLE `eurosport`.`equipment\_modification\_log` (

  `log\_id` INT NOT NULL AUTO\_INCREMENT,

  `equipment\_id` INT NULL,

  `modification\_time` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

  PRIMARY KEY (`log\_id`))

ENGINE = InnoDB;

CREATE TRIGGER log\_equipment\_modification

AFTER UPDATE ON `eurosport`.`equipment`

FOR EACH ROW

BEGIN

  INSERT INTO `eurosport`.`equipment\_modification\_log` (`equipment\_id`) VALUES (NEW.id);

END;

* 1. Для певного стовпця забезпечити формат вводу:    
     2 довільні букви, окрім M і R + '-' + 3 цифри + '-' + 2цифри

CREATE TRIGGER check\_serial\_format

BEFORE INSERT OR UPDATE ON `eurosport`.`equipment`

FOR EACH ROW

BEGIN

  IF NOT NEW.serial\_number REGEXP '^[^MR][^MR]-[0-9]{3}-[0-9]{2}$' THEN

    SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Serial number format is not valid';

  END IF;

END;

* 1. Для певного стовпця забезпечити формат вводу:  
     1 буква: A, M чи Z  + 5 цифр + 2 довільні букви

CREATE TRIGGER check\_trainer\_phone\_format

BEFORE INSERT ON `eurosport`.`trainer`

FOR EACH ROW

BEGIN

  IF NOT NEW.phone REGEXP '^[AMZ][0-9]{5}[A-Za-z]{2}$' THEN

    SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Phone format is not valid';

  END IF;

END;

* 1. Для певного стовпця допускається ввід лише таких імен: 'Svitlana', 'Petro', 'Olha', 'Taras'.

CREATE TRIGGER check\_client\_name

BEFORE INSERT ON `eurosport`.`client`

FOR EACH ROW

BEGIN

  IF NEW.name NOT IN ('Svitlana', 'Petro', 'Olha', 'Taras') THEN

    SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Invalid name. Only Svitlana, Petro, Olha, or Taras are allowed';

  END IF;

END;

1. Для певного стовпця забезпечити формат вводу:  
   перша літера у значенні повинна відповідати першій літері значення сусіднього поля у рядку

CREATE TRIGGER match\_first\_letters

BEFORE INSERT OR UPDATE ON `eurosport`.`client`

FOR EACH ROW

BEGIN

  IF LEFT(NEW.name, 1) <> LEFT(NEW.surname, 1) THEN

    SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'First letters of name and surname must match';

  END IF;

END;