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

DELIMITER //

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.
     2. 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.
     3. 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
   2. 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
  2. Створити таблицю-журнал, в якій вести логи зі штампом часу при видаленні даних для певної таблиці
  3. Створити таблицю-журнал, в якій вести логи зі штампом часу при модифікації даних для таблиці
  4. Для певного стовпця забезпечити формат вводу:    
     2 довільні букви, окрім M і R + '-' + 3 цифри + '-' + 2цифри
  5. Для певного стовпця забезпечити формат вводу:  
     1 буква: A, M чи Z  + 5 цифр + 2 довільні букви
  6. Для певного стовпця допускається ввід лише таких імен: 'Svitlana', 'Petro', 'Olha', 'Taras'.

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

# SQL-скріпт

-- -----------------------------------------------------

-- Schema eurosport

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `eurosport` DEFAULT CHARACTER SET utf8 ;

USE `eurosport` ;

-- -----------------------------------------------------

-- Table `eurosport`.`time`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `eurosport`.`time` (

`id` INT NOT NULL AUTO\_INCREMENT,

`day` VARCHAR(15) NULL,

`time\_start` VARCHAR(45) NOT NULL,

`time\_end` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `id\_UNIQUE` (`id` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `eurosport`.`service`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `eurosport`.`service` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`time\_id` INT NOT NULL,

PRIMARY KEY (`id`, `time\_id`),

INDEX `fk\_service\_time1\_idx` (`time\_id` ASC) VISIBLE,

CONSTRAINT `fk\_service\_time1`

FOREIGN KEY (`time\_id`)

REFERENCES `eurosport`.`time` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `eurosport`.`trainer`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `eurosport`.`trainer` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`surname` VARCHAR(45) NOT NULL,

`phone` VARCHAR(12) NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `eurosport`.`membership\_card\_type`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `eurosport`.`membership\_card\_type` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `id\_UNIQUE` (`id` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `eurosport`.`client`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `eurosport`.`client` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`surname` VARCHAR(45) NOT NULL,

`phone` VARCHAR(12) NULL,

`membership\_card\_type\_id` INT NOT NULL,

PRIMARY KEY (`id`, `membership\_card\_type\_id`),

INDEX `fk\_client\_membership\_card\_type1\_idx` (`membership\_card\_type\_id` ASC) VISIBLE,

CONSTRAINT `fk\_client\_membership\_card\_type1`

FOREIGN KEY (`membership\_card\_type\_id`)

REFERENCES `eurosport`.`membership\_card\_type` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `eurosport`.`room`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `eurosport`.`room` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NULL,

`area` FLOAT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `id\_UNIQUE` (`id` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `eurosport`.`equipment`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `eurosport`.`equipment` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(70) NULL,

`serial\_number` VARCHAR(45) NULL,

`room\_id` INT NOT NULL,

PRIMARY KEY (`id`, `room\_id`),

UNIQUE INDEX `id\_UNIQUE` (`id` ASC) VISIBLE,

INDEX `fk\_equipment\_room1\_idx` (`room\_id` ASC) VISIBLE,

CONSTRAINT `fk\_equipment\_room1`

FOREIGN KEY (`room\_id`)

REFERENCES `eurosport`.`room` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `eurosport`.`group\_training`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `eurosport`.`group\_training` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NULL,

`trainer\_id` INT NOT NULL,

`time\_id` INT NOT NULL,

PRIMARY KEY (`id`, `trainer\_id`, `time\_id`),

UNIQUE INDEX `id\_UNIQUE` (`id` ASC) VISIBLE,

INDEX `fk\_group\_training\_trainer1\_idx` (`trainer\_id` ASC) VISIBLE,

INDEX `fk\_group\_training\_working\_time1\_idx` (`time\_id` ASC) VISIBLE,

CONSTRAINT `fk\_group\_training\_trainer1`

FOREIGN KEY (`trainer\_id`)

REFERENCES `eurosport`.`trainer` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_group\_training\_working\_time1`

FOREIGN KEY (`time\_id`)

REFERENCES `eurosport`.`time` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `eurosport`.`trainer\_service`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `eurosport`.`trainer\_service` (

`trainer\_id` INT NOT NULL,

`service\_id` INT NOT NULL,

PRIMARY KEY (`trainer\_id`, `service\_id`),

INDEX `fk\_trainer\_has\_service\_service1\_idx` (`service\_id` ASC) VISIBLE,

INDEX `fk\_trainer\_has\_service\_trainer\_idx` (`trainer\_id` ASC) VISIBLE,

CONSTRAINT `fk\_trainer\_has\_service\_trainer`

FOREIGN KEY (`trainer\_id`)

REFERENCES `eurosport`.`trainer` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_trainer\_has\_service\_service1`

FOREIGN KEY (`service\_id`)

REFERENCES `eurosport`.`service` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `eurosport`.`client\_training`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `eurosport`.`client\_training` (

`client\_id` INT NOT NULL,

`group\_training\_id` INT NOT NULL,

PRIMARY KEY (`client\_id`, `group\_training\_id`),

INDEX `fk\_client\_has\_group\_training\_group\_training1\_idx` (`group\_training\_id` ASC) VISIBLE,

INDEX `fk\_client\_has\_group\_training\_client1\_idx` (`client\_id` ASC) VISIBLE,

CONSTRAINT `fk\_client\_has\_group\_training\_client1`

FOREIGN KEY (`client\_id`)

REFERENCES `eurosport`.`client` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_client\_has\_group\_training\_group\_training1`

FOREIGN KEY (`group\_training\_id`)

REFERENCES `eurosport`.`group\_training` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `eurosport`.`personal\_training`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `eurosport`.`personal\_training` (

`id` INT NOT NULL AUTO\_INCREMENT,

`trainer\_id` INT NOT NULL,

`client\_id` INT NOT NULL,

`name` VARCHAR(45) NULL,

PRIMARY KEY (`id`, `trainer\_id`, `client\_id`),

UNIQUE INDEX `id\_UNIQUE` (`id` ASC) VISIBLE,

INDEX `fk\_personal\_training\_trainer1\_idx` (`trainer\_id` ASC) VISIBLE,

INDEX `fk\_personal\_training\_client1\_idx` (`client\_id` ASC) VISIBLE,

CONSTRAINT `fk\_personal\_training\_trainer1`

FOREIGN KEY (`trainer\_id`)

REFERENCES `eurosport`.`trainer` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_personal\_training\_client1`

FOREIGN KEY (`client\_id`)

REFERENCES `eurosport`.`client` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `eurosport`.`personal\_training\_equipment`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `eurosport`.`personal\_training\_equipment` (

`personal\_training\_id` INT NOT NULL,

`equipment\_id` INT NOT NULL,

PRIMARY KEY (`personal\_training\_id`, `equipment\_id`),

INDEX `fk\_personal\_training\_has\_equipment\_equipment1\_idx` (`equipment\_id` ASC) VISIBLE,

INDEX `fk\_personal\_training\_has\_equipment\_personal\_training1\_idx` (`personal\_training\_id` ASC) VISIBLE,

CONSTRAINT `fk\_personal\_training\_has\_equipment\_personal\_training1`

FOREIGN KEY (`personal\_training\_id`)

REFERENCES `eurosport`.`personal\_training` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_personal\_training\_has\_equipment\_equipment1`

FOREIGN KEY (`equipment\_id`)

REFERENCES `eurosport`.`equipment` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- Several additional indexes according to business rules

CREATE INDEX idx\_client\_surname ON `eurosport`.`client` (`surname`);

CREATE INDEX idx\_client\_phone ON `eurosport`.`client` (`phone`);

CREATE INDEX idx\_equipment\_name ON `eurosport`.`equipment` (`name`);

CREATE INDEX idx\_equipment\_serial\_number ON `eurosport`.`equipment` (`serial\_number`);

-- Seed data for room table

INSERT INTO `room` (`name`, `area`) VALUES

('swimming pool', 150),

('main', 200),

('crossfit', 50),

('cardio', 100);

-- Seed data for membership\_card\_type table

INSERT INTO `membership\_card\_type` (`name`) VALUES

('DAILY plus'),

('SILVER plus'),

('GOLD');

-- Seed data for time table

INSERT INTO `time` (`day`, `time\_start`, `time\_end`) VALUES

('Monday', '08:00', '10:00'),

('Tuesday', '10:00', '12:00'),

('Wednesday', '12:00', '14:00'),

('Thursday', '14:00', '16:00'),

('Friday', '16:00', '18:00');

-- Seed data for service table

INSERT INTO `service` (`name`, `time\_id`) VALUES

('Yoga', 1),

('Zumba', 2),

('Boxing', 3),

('Dance', 4),

('Pilates', 5);

-- Seed data for trainer table

INSERT INTO `trainer` (`name`, `surname`, `phone`) VALUES

('John', 'Doe', '0981234567'),

('Jane', 'Smith', '0982345678'),

('Robert', 'Brown', '0983456789'),

('Emily', 'Johnson', '0984567890'),

('Michael', 'White', '0985678901');

-- Seed data for equipment table

INSERT INTO `equipment` (`name`, `serial\_number`, `room\_id`) VALUES

('Treadmill A', 'SN001', 4),

('Dumbbell Set', 'SN002', 2),

('Pool Ladder', 'SN003', 1),

('Crossfit Rig', 'SN004', 3),

('Rowing Machine', 'SN005', 3),

('Bench Press', 'SN006', 2),

('Elliptical', 'SN007', 4),

('Boxing Gloves', 'SN008', 2),

('Yoga Mats', 'SN009', 2),

('Swim Floats', 'SN010', 1);

-- Seed data for client table

INSERT INTO `client` (`name`, `surname`, `phone`, `membership\_card\_type\_id`) VALUES

('Alice', 'Martin', '0986789012', 1),

('David', 'Jones', '0987890123', 2),

('Angela', 'Clark', '0988901234', 3),

('Richard', 'Hall', '0989012345', 1),

('Patricia', 'Lee', '0980123456', 2);

-- Seed data for group\_training table

INSERT INTO `group\_training` (`name`, `trainer\_id`, `time\_id`) VALUES

('Morning Yoga', 1, 1),

('Evening Zumba', 2, 2),

('Midday Boxing', 3, 3),

('Afternoon Dance', 4, 4),

('Pilates Session', 5, 5);

-- Seed data for trainer\_service table

INSERT INTO `trainer\_service` (`trainer\_id`, `service\_id`) VALUES

(1, 1),

(2, 2),

(3, 3),

(4, 4),

(5, 5);

-- Seed data for client\_training table

INSERT INTO `client\_training` (`client\_id`, `group\_training\_id`) VALUES

(1, 1),

(2, 2),

(3, 3),

(4, 4),

(5, 5);

-- Seed data for personal\_training table

INSERT INTO `personal\_training` (`trainer\_id`, `client\_id`, `name`) VALUES

(1, 1, 'PT Session 1'),

(2, 2, 'PT Session 2'),

(3, 3, 'PT Session 3'),

(4, 4, 'PT Session 4'),

(5, 5, 'PT Session 5');

-- Seed data for personal\_training\_equipment table

INSERT INTO `personal\_training\_equipment` (`personal\_training\_id`, `equipment\_id`) VALUES

(1, 1),

(2, 2),

(3, 3),

(4, 4),

(5, 5);