Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. І. Сікорського»
Інститут прикладного системного аналізу

Курсова робота
з курсу «Організація баз даних та знань»
з теми «База даних ЗВО»

Виконав: студент 3 курсу
групи КА-81
Галганов Олексій
Прийняла: асистент
Гуськова Віра Геннадіївна

Київ 2021

ЗМІСТ

ВСТУП

ЗВО (заклад вищої освіти) є досить складною системою, в якій непросто описати всі сутності та зв’язки між ними. Основною задачею бази даних ЗВО є забезпечення зручного та швидкого доступу до інформації про факультети та кафедри, викладачів та їх навантаження, студентів, розклад. Необхідно враховувати реальну структуру закладу вищої освіти та зв’язки між наведеними сутностями.

Актуальність. Функціонування такої складної системи, як заклад вищої освіти, неможливе без використання ефективної інформаційної системи, яка забезпечуватиме швидкий та зручний доступ до потрібної інформації.

Мета. Метою роботи є розробка автоматизованої інформаційної системи для закладу вищої освіти, яка дозволить зберігати всю необхідну інформацію, забезпечить виконання всіх видів інформаційних запитів, які необхідні при експлуатації даної системи.

Завдання. Спроектувати базу даних та підготувати усі необхідні запити та процедури для роботи з нею.

Практичне значення. Вдосконалення навичок SQL-програмування, аналізу предметної області, проектування баз даних.

Програмне забезпечення. При виконанні роботи було використано СУБД MySQL 8.0, веб-інтерфейс реалізовано з використанням фреймворку Flask мови програмування Python 3.9. Використовувалися ОС Windows 10, середовища розробки Visual Studio Code і MySQL Workbench та система контролю версій Git.

РОЗДІЛ 1
ПОСТАНОВКА ЗАДАЧІ

Студенти, організовані в групи, які навчаються на одному з факультетів, очолюваному деканатом, в функції якого входить контроль навчального процесу. У навчальному процесі беруть участь викладачі кафедр, адміністративно відносяться до одного з факультетів. Викладачі поділяються на такі категорії: асистенти, викладачі, старші викладачі, доценти, професори. Асистенти і викладачі можуть навчатися в аспірантурі, ст. викладачі, доценти, можуть очолювати наукові теми, професора – наукові напрямки. Викладачі будь-якої категорії свого часу могли захистити кандидатську, а доценти і професори і докторську дисертацію, при цьому викладачі можуть займати посади доцента і професора тільки, якщо вони мають відповідно звання доцента і професора. Навчальний процес регламентується навчальним планом, в якому вказується, які навчальні дисципліни на яких курсах і у яких семестрах читаються для студентів кожного року набору, із зазначенням кількості годин на кожен вид занять з дисципліни (види занять: лекції, семінари, лабораторні роботи, консультації, курсові роботи, і т.д.) і форми контролю (залік, іспит). Перед початком навчального семестру деканати роздають на кафедри навчальні доручення, в яких вказуються будь кафедри (не обов’язково пов’язані з цим факультетом), які дисципліни і для яких груп повинні вести в черговому семестрі. Керуючись ними, на кафедрах здійснюється розподіл навантаження, при цьому по одній дисципліні в одній групі різні види занять можуть вести один або кілька різних викладачів кафедри (з урахуванням категорії викладачів, наприклад, асистент не може читати лекції, а професор ніколи не буде проводити лабораторні роботи). Викладач може вести заняття по одній або декількох дисциплінах для студентів як свого, так і інших факультетів. Відомості про проведені іспитах і заліках збираються деканатом. Після закінчення навчання студент виконує дипломну роботу, керівником якої є викладач кафедри, що відноситься до того ж факультету, де навчається студент, при цьому викладач може керувати кількома студентами.

Види запитів в інформаційний системі:

1. Отримати перелік і загальне число студентів зазначених груп або вказаного курсу (курсів) факультету повністю, за статевою ознакою, року,

віком, ознакою наявності дітей, за ознакою отримання і розміром стипендії.

2. Отримати список і загальне число викладачів зазначених кафедр або зазначеного факультету повністю або зазначених категорій (асистенти, доценти, професори і т.д.) за статевою ознакою, року, віком, ознакою наявності та кількості дітей, розміру заробітної плати, є аспірантами, захистили кандидатські, докторські дисертації в зазначений період.
3. Отримати перелік і загальне число тем кандидатських і докторських дисертацій, які захистили співробітники зазначеної кафедри для зазначеного факультету.
4. Отримати перелік кафедр, які проводять заняття у зазначеній групі або на зазначеному курсі вказаного факультету в зазначеному семестрі, або за вказаний період.
5. Отримати список і загальне число викладачів, які проводили (проводять) заняття по вказаній дисципліні в зазначеній групі або на зазначеному курсі вказаного факультету.
6. Отримати перелік і загальне число викладачів, які проводили (проводять) лекційні, семінарські та інші види занять у зазначеній групі або на зазначеному курсі вказаного факультету в зазначеному семестрі, або за вказаний період.
7. Отримати список і загальне число студентів зазначених груп, які здали залік або іспит з вказаною дисципліни зі встановленою оцінкою.
8. Отримати список і загальне число студентів зазначених груп або вказаного курсу зазначеного факультету, які здали зазначену сесію на відмінно, без трійок, без двійок.
9. Отримати перелік викладачів, які беруть (брали) іспити в зазначених групах, із зазначених дисциплін, в зазначеному семестрі.
10. Отримати список студентів зазначених груп, яким заданий викладач поставив деяку оцінку за іспит з певних дисциплін, в зазначених семестрах, за деякий період.
11. Отримати список студентів і тим дипломних робіт на зазначеній кафедрі або у зазначеного викладача.
12. Отримати список керівників дипломних робіт по заданій кафедрі або

факультету повністю і окремо по деяким категоріям викладачів.

13. Отримати навантаження викладачів (назва дисципліни, кількість годин), її обсяг на окремі види занять і загальне навантаження в зазначеному семестрі для конкретного викладача або для викладачів зазначеної кафедри.

РОЗДІЛ 2
АРХІТЕКТУРА ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ БД

2.1 Аналіз функціонування та організаційні засади підприємства

До основних функціональних завдань ЗВО належить планування занять та формування їх розкладу, облік викладачів та студентів, контроль успішності студентів. Для зберігання необхідних даних доцільно сформувати такі таблиці: «Факультети», «Кафедри», «Групи», «Студенти», «Викладачі», «Дисципліни», «Розклад» та «Сесія». На кожному факультеті є кафедри, за якими закріплені викладачі та навчальні групи студентів. В таблиці «Дисципліни» має бути загальний перелік дисциплін, а вже в таблиці «Розклад» – розподіл дисциплін за групами та викладачами.

2.2 Проектування структури бази даних

Структуру таблиць БД та зв’язки між ними зображено на EER-діаграмі:

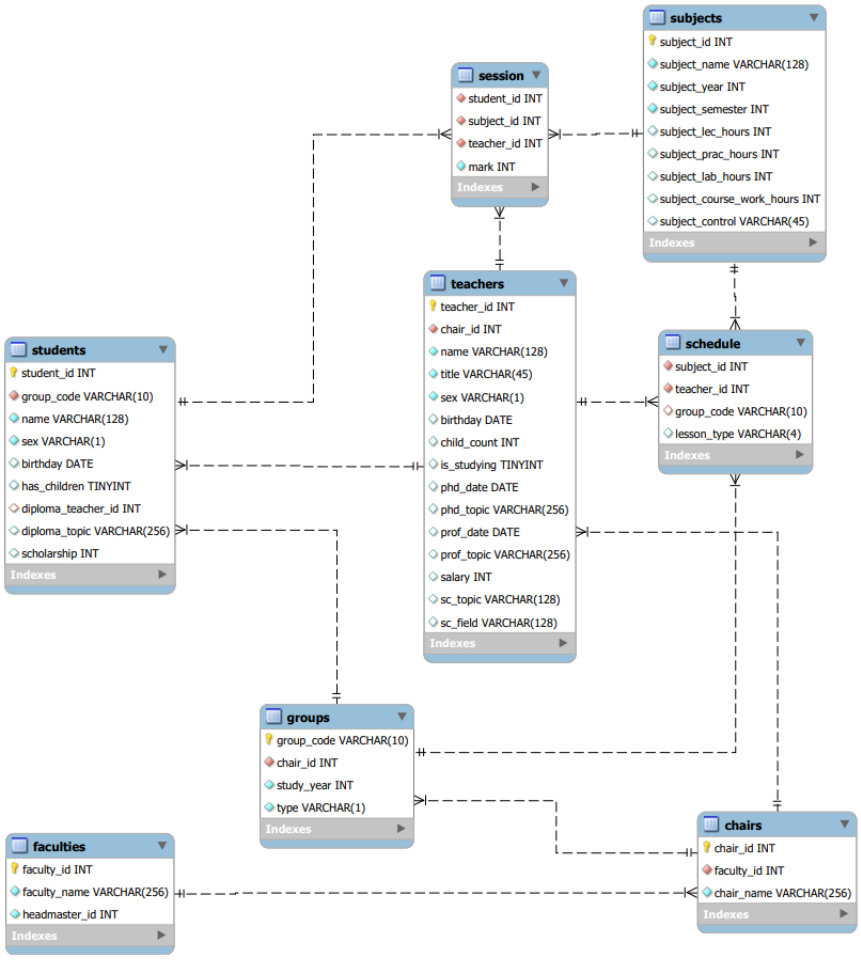

Рис. 2.1 – EER-діаграма

Зв’язки між різними таблицями реалізовано через механізм Foreign Key. Наприклад, в таблиці «Сесія» поля subject_id, student_id, teacher_id пов’язані з однойменними полями, що є Primary Key, в таблицях «Дисципліни», «Студенти» та «Викладачі».

2.3 Життєві цикли бази даних

1. Попереднє планування. На даному етапі було сформовано модель структури бази даних, визначено сутності та проаналізовано зв’язки між ними.
2. Перевірка здійсненності. Для розробки даної системи необхідно мати встановлений сервер MySQL 8.0, інтерпретатор мови Python 3.9 з найновішими версіями фреймворку Flask і бібліотеки PyMySQL та середовище розробки MySQL Workbench.
3. Вимоги до даної інформаційної системи були сформовані в постановці завдання, зокрема, вказана структура організації та види інформаційних заходів, які мають бути реалізовані в системі.
4. Проектування. Для реалізації завдання обрано реляційну модель бази даних. В якості СУБД обрано MySQL. Створення початкової структури бази даних (сутності та зв’язки між ними) буде проведено в середовищі MySQL Workbench, а реалізацію веб-інтерфейсу доступу до БД та зв’язку між веб-інтерфейсом та MySQL-сервером – в середовищі Visual Studio Code з використанням мови Python та фреймворку Flask.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМНОЇ ВЗАЄМОДІЇ З БД

3.1 Посібник користувача

Попередньо на ПК має бути встановлено сервер MySQL 8.0 та інтерпретатор мови програмування Python 3.9. Після запуску SQL-сервера перед першим запуском інформаційної системи необхідно запустити файл setup.py (додаток 2, ст. 15), який виконає створення схеми БД на сервері, створить усі необхідні таблиці, створить усі необхідні для запитів процедури та завантажить тестові дані до таблиць:

```
PS C:\Users\Yalikesi\...\campus> python setup.py
enter connection parameters:
        host (leave empty for default localhost):
        port (leave empty for default 3307):
        user (leave empty for default root):
        password for root:
creating schema and tables...
        done in 1.109 sec
inserting data...
        done in 28.481 sec
creating stored procedures...
        done in 0.227 sec
database is ready!
```

Після цього параметри підключення до БД буде збережено і можна буде запускати веб-інтерфейс за допомогою файлу main.py: буде виведено рядок `Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)`, який означатиме, що веб-інтерфейс доступний за адресою 127.0.0.1:5000. Його можна відкрити за допомогою будь-якого сучасного браузера. Під час тестування коректність його роботи перевірялася в Google Chrome 90. У веб інтерфейсі є можливість перегляду окремих таблиць БД та виконання передбачених запитів. На всіх сторінках, де виводяться таблиці, є можливість сортування за кожним стовпцем та пошуку по виведеній таблиці.

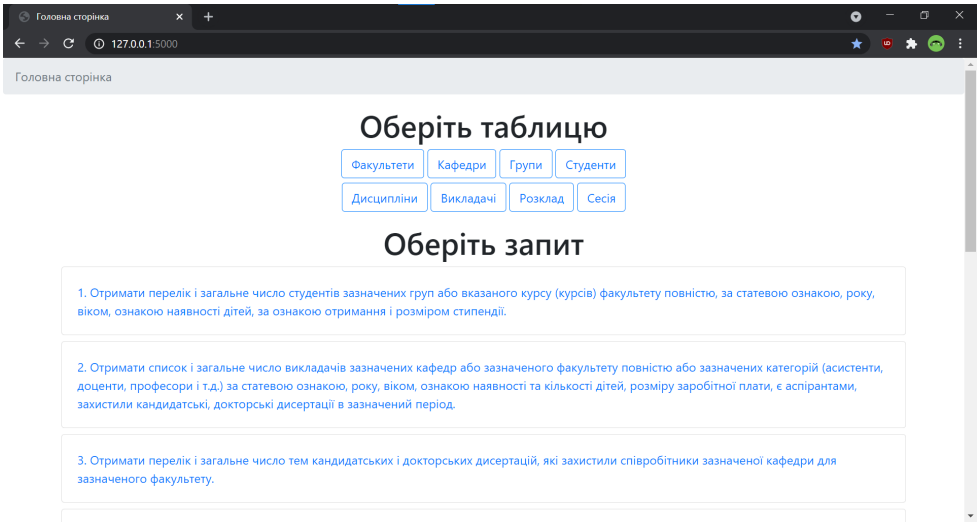

Рис. 3.1 – Головна сторінка веб-інтерфейсу

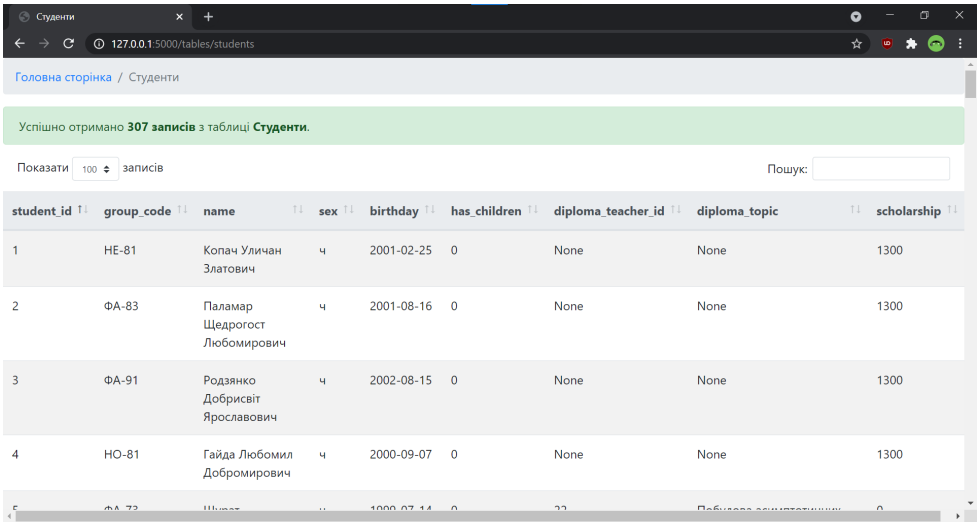

Рис. 3.2 – Перегляд таблиці «Студенти»

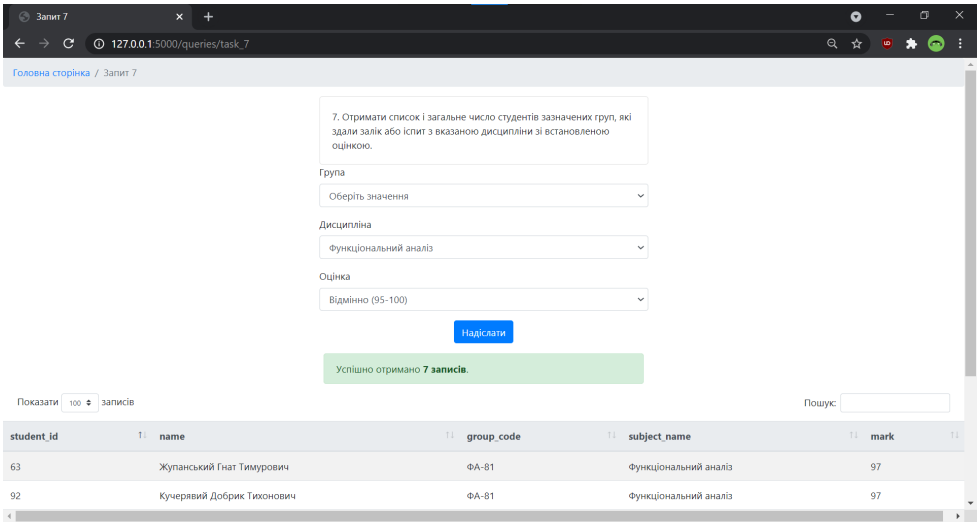

Рис. 3.3 – Форма для запиту та результат виконання

3.2 Реалізація механізмів БД

Реалізовано 13 процедур, що відповідають поставленим вимогам до запитів. SQL-код цих процедур наведено в додатку 3, ст. **??**. Виклик усіх процедур відбувається через веб-інтерфейс і користувачу не потрібно писати назву чи параметри процедури вручну.

ДОДАТКИ

Додаток 1. Створення схеми бази даних.

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES
    ,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,
    NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema univ_db
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `univ_db` ;

-- -----------------------------------------------------
-- Schema univ_db
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `univ_db` DEFAULT CHARACTER SET utf8mb4 COLLATE
    utf8mb4_0900_ai_ci ;
USE `univ_db` ;

-- -----------------------------------------------------
-- Table `univ_db`.`faculties`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `univ_db`.`faculties` ;

CREATE TABLE IF NOT EXISTS `univ_db`.`faculties` (
  `faculty_id` INT NOT NULL,
  `faculty_name` VARCHAR(256) NOT NULL,
  `headmaster_id` INT NOT NULL,
  PRIMARY KEY (`faculty_id`),
  UNIQUE INDEX `faculty_ID_UNIQUE` (`faculty_id` ASC) VISIBLE,
  UNIQUE INDEX `faculty_Name_UNIQUE` (`faculty_name` ASC) VISIBLE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `univ_db`.`chairs`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `univ_db`.`chairs` ;

CREATE TABLE IF NOT EXISTS `univ_db`.`chairs` (
```

```
  `chair_id` INT NOT NULL,
  `faculty_id` INT NOT NULL,
  `chair_name` VARCHAR(256) NOT NULL,
  PRIMARY KEY (`chair_id`),
  UNIQUE INDEX `teacher_id_UNIQUE` (`chair_id` ASC) VISIBLE,
  INDEX `faculty_id_idx` (`faculty_id` ASC) VISIBLE,
  CONSTRAINT `faculty_id`
    FOREIGN KEY (`faculty_id`)
    REFERENCES `univ_db`.`faculties` (`faculty_id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `univ_db`.`groups`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `univ_db`.`groups` ;

CREATE TABLE IF NOT EXISTS `univ_db`.`groups` (
  `group_code` VARCHAR(10) NOT NULL,
  `chair_id` INT NOT NULL,
  `study_year` INT NOT NULL,
  `type` VARCHAR(1) NOT NULL DEFAULT 'b',
  PRIMARY KEY (`group_code`),
  UNIQUE INDEX `group_code_UNIQUE` (`group_code` ASC) VISIBLE,
  INDEX `chair_id_idx` (`chair_id` ASC) VISIBLE,
  CONSTRAINT `chair_id_group`
    FOREIGN KEY (`chair_id`)
    REFERENCES `univ_db`.`chairs` (`chair_id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `univ_db`.`subjects`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `univ_db`.`subjects` ;

CREATE TABLE IF NOT EXISTS `univ_db`.`subjects` (
  `subject_id` INT NOT NULL,
  `subject_name` VARCHAR(128) NOT NULL,
  `subject_year` INT NOT NULL,
  `subject_semester` INT NOT NULL,
  `subject_lec_hours` INT NULL DEFAULT NULL,
  `subject_prac_hours` INT NULL DEFAULT NULL,
```

```
  `subject_lab_hours` INT NULL DEFAULT NULL,
  `subject_course_work_hours` INT NULL DEFAULT NULL,
  `subject_control` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`subject_id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `univ_db`.`teachers`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `univ_db`.`teachers` ;

CREATE TABLE IF NOT EXISTS `univ_db`.`teachers` (
  `teacher_id` INT NOT NULL,
  `chair_id` INT NOT NULL,
  `name` VARCHAR(128) NOT NULL,
  `title` VARCHAR(45) NOT NULL,
  `sex` VARCHAR(1) NOT NULL,
  `birthday` DATE NULL DEFAULT NULL,
  `child_count` INT NULL DEFAULT NULL,
  `is_studying` TINYINT NULL DEFAULT NULL,
  `phd_date` DATE NULL DEFAULT NULL,
  `phd_topic` VARCHAR(256) NULL DEFAULT NULL,
  `prof_date` DATE NULL DEFAULT NULL,
  `prof_topic` VARCHAR(256) NULL DEFAULT NULL,
  `salary` INT NULL DEFAULT NULL,
  `sc_topic` VARCHAR(128) NULL DEFAULT NULL,
  `sc_field` VARCHAR(128) NULL DEFAULT NULL,
  PRIMARY KEY (`teacher_id`),
  UNIQUE INDEX `teacher_id_UNIQUE` (`teacher_id` ASC) VISIBLE,
  INDEX `chair_id_idx` (`chair_id` ASC) VISIBLE,
  CONSTRAINT `chair_id_teacher`
    FOREIGN KEY (`chair_id`)
    REFERENCES `univ_db`.`chairs` (`chair_id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `univ_db`.`schedule`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `univ_db`.`schedule` ;

CREATE TABLE IF NOT EXISTS `univ_db`.`schedule` (
```

```
  `subject_id` INT NOT NULL,
  `teacher_id` INT NOT NULL,
  `group_code` VARCHAR(10) NULL DEFAULT NULL,
  `lesson_type` VARCHAR(4) NULL DEFAULT NULL,
  INDEX `teacher_id_idx` (`teacher_id` ASC) VISIBLE,
  INDEX `subject_id_idx` (`subject_id` ASC) VISIBLE,
  INDEX `group_code_idx` (`group_code` ASC) VISIBLE,
  CONSTRAINT `group_code`
    FOREIGN KEY (`group_code`)
    REFERENCES `univ_db`.`groups` (`group_code`),
  CONSTRAINT `subject_id`
    FOREIGN KEY (`subject_id`)
    REFERENCES `univ_db`.`subjects` (`subject_id`),
  CONSTRAINT `teacher_id`
    FOREIGN KEY (`teacher_id`)
    REFERENCES `univ_db`.`teachers` (`teacher_id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `univ_db`.`students`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `univ_db`.`students` ;

CREATE TABLE IF NOT EXISTS `univ_db`.`students` (
  `student_id` INT NOT NULL,
  `group_code` VARCHAR(10) NOT NULL,
  `name` VARCHAR(128) NOT NULL,
  `sex` VARCHAR(1) NOT NULL,
  `birthday` DATE NULL DEFAULT NULL,
  `has_children` TINYINT NULL DEFAULT NULL,
  `diploma_teacher_id` INT NULL DEFAULT NULL,
  `diploma_topic` VARCHAR(256) NULL DEFAULT NULL,
  `scholarship` INT NULL DEFAULT NULL,
  PRIMARY KEY (`student_id`),
  INDEX `student_group_idx` (`group_code` ASC) VISIBLE,
  INDEX `diploma_teacher_id_idx` (`diploma_teacher_id` ASC) VISIBLE,
  CONSTRAINT `diploma_teacher_id`
    FOREIGN KEY (`diploma_teacher_id`)
    REFERENCES `univ_db`.`teachers` (`teacher_id`),
  CONSTRAINT `student_group`
    FOREIGN KEY (`group_code`)
    REFERENCES `univ_db`.`groups` (`group_code`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
```

```
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `univ_db`.`session`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `univ_db`.`session` ;

CREATE TABLE IF NOT EXISTS `univ_db`.`session` (
  `student_id` INT NOT NULL,
  `subject_id` INT NOT NULL,
  `teacher_id` INT NOT NULL,
  `mark` INT NOT NULL,
  INDEX `student_idx` (`student_id` ASC) VISIBLE,
  INDEX `subject_idx` (`subject_id` ASC) VISIBLE,
  INDEX `teacher_idx` (`teacher_id` ASC) VISIBLE,
  CONSTRAINT `student`
    FOREIGN KEY (`student_id`)
    REFERENCES `univ_db`.`students` (`student_id`),
  CONSTRAINT `subject`
    FOREIGN KEY (`subject_id`)
    REFERENCES `univ_db`.`subjects` (`subject_id`),
  CONSTRAINT `teacher`
    FOREIGN KEY (`teacher_id`)
    REFERENCES `univ_db`.`teachers` (`teacher_id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Додаток 2. Початкове налаштування бази даних, внесення тестових даних.

```
import pymysql
import sys
import getpass
import time

def update_config(host, port, user, password):
    with open('db_connect.py', 'w') as f:
        f.writelines(["import pymysql\n\n",
                      "db = pymysql.connect(\n",
                      f"\thost='{host}',\n",
```

```
                    f"\tport={port},\n",
                    f"\tuser='{user}',\n",
                    f"\tpassword='{password}',\n",
                    f"\tdatabase='univ_db'\n)"])

# https://stackoverflow.com/questions/745538/create-function-through-mysqldb
def execute_script(fname, db):
    delimiter = ';'
    statement = ""
    with open(fname, encoding='utf-8') as f:
        for line in f.readlines():
            line = line.strip()
            if line.startswith('delimiter'):
                delimiter = line[10:]
            else:
                statement += line + '\n'
                if line.endswith(delimiter):
                    statement = statement.strip().strip(delimiter)
                    with db.cursor() as cursor:
                        try:
                            cursor.execute(statement)
                            db.commit()
                            statement = ""
                        except cursor.Error as e:
                            print(f"{fname} - error applying ({str(e)})\
   nstatement:{statement}\nterminating...")
                            sys.exit(1)

if __name__ == '__main__':
    print('enter connection parameters:')
    host = input('\thost (leave empty for default localhost): ') or 'localhost
   '
    port = input('\tport (leave empty for default 3307): ') or 3307
    user = input('\tuser (leave empty for default root): ') or 'root'
    password = getpass.getpass(f'\tpassword for {user}: ')
    update_config(host, port, user, password)
    db = pymysql.connect(host=host, port=port, user=user, password=password)
    print('creating schema and tables...')
    tock = time.time()
    execute_script('./inserts/create_tables.sql', db)
    tick = time.time()
    print(f'\tdone in {(tick-tock):.3f} sec')
    db = pymysql.connect(host=host, port=port, user=user, password=password,
   database='univ_db')
    print('inserting data...')
    tock = time.time()
    execute_script('./inserts/faculties.sql', db)
```

```
execute_script('./inserts/chairs.sql', db)
execute_script('./inserts/teachers.sql', db)
execute_script('./inserts/groups.sql', db)
execute_script('./inserts/students.sql', db)
execute_script('./inserts/subjects.sql', db)
execute_script('./inserts/schedule.sql', db)
execute_script('./inserts/session.sql', db)
tick = time.time()
print(f'\tdone in {(tick-tock):.3f} sec')
print('creating stored procedures...')
tock = time.time()
for i in range(1, 14):
    execute_script(f'./queries/task_{i}.sql', db)
tick = time.time()
print(f'\tdone in {(tick-tock):.3f} sec')
print('database is ready!')
```