ЗАТВЕРДЖУЮ
Начальник кафедри інформаційних технологій та систем електронних комунікацій, к.т.н., доцент
Назарій БУРАК
«____» _____ 20___ р.

МЕТОДИЧНА РОЗРОБКА

для проведення лабораторного заняття із здобувачами освіти першого (бакалаврського) рівня вищої освіти спеціальності F3 «Комп'ютерні науки»
з дисципліни «WEB-технології та WEB-програмування»

Лабораторна робота № 2: «Проєктування бази даних предметної області, створення таблиць і SQL-запитів. Робота з базою даних із Java-застосунку засобами JDBC»

(тема 1.2)

Мета: навчитися проєктувати реляційну схему бази даних предметної області, створювати таблиці з ключами та обмеженнями цілісності, писати SQL-запити з'єднання і групування, виконувати запити і транзакції з Java-застосунку засобами JDBC.

Час: 4 години.

Місце заняття: комп'ютерна лабораторія.

Матеріальне забезпечення: ПК з доступом до мережі Internet; JDK 21; IntelliJ IDEA; Maven; PostgreSQL 15+ (або MySQL 8); Git; демо-проєкт https://github.com/SidiromUA/shelter-registry (тег `lab-02`).

Після лабораторного заняття здобувачі освіти повинні:

вміти: будувати ER-модель і схему БД, створювати таблиці з PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK; писати запити з JOIN, GROUP BY, HAVING і підзапитами; підключати драйвер JDBC через Maven; виконувати запити через PreparedStatement і керувати транзакцією;

знати: типи зв'язків між таблицями, групи команд SQL, основні інтерфейси JDBC, властивості ACID, механізм SQL-ін'єкції.

I. Підготовка до заняття

Викладач перевіряє на ПК лабораторії наявність PostgreSQL (служба запущена), JDK 21, Maven. Здобувачі освіти до заняття опрацьовують лекцію 2 та погоджують із викладачем предметну область власного вебсервісу (обрану в ЛР 1).

II. Структура заняття

Короткі теоретичні відомості

Реляційна БД зберігає дані в таблицях; **первинний ключ** ідентифікує рядок, **зовнішній ключ** посилається на рядок іншої таблиці і реалізує зв'язок «один до багатьох». **JDBC** – стандартний API Java для роботи з БД: `Connection` – з'єднання, `PreparedStatement` – запит з параметрами `?`, `ResultSet` – результат вибірки. Параметризовані запити захищають від SQL-ін'єкцій; транзакція (`setAutoCommit(false)`, `commit()`, `rollback()`) виконує кілька змін як одне ціле.

2.1. Підготовка бази даних (на прикладі демо-проєкту)

```
cd shelter-registry
git checkout lab-02

# користувач і база даних (один раз)
psql -d postgres -c "CREATE ROLE shelter LOGIN PASSWORD 'shelter';"
psql -d postgres -c "CREATE DATABASE shelter_registry OWNER shelter;"

# таблиці і тестові дані
psql -h localhost -U shelter -d shelter_registry \
     -f db/schema.sql -f db/data.sql
```

Ознайомтеся з файлом `db/schema.sql`: таблиці `districts`, `shelters`, `inspections`, `users`; зовнішні ключі, обмеження `CHECK` і `UNIQUE`, індекси. Під'єднайтеся до БД у вікні **Database** IntelliJ IDEA (Ultimate) або через DBeaver/pgAdmin і побудуйте діаграму таблиць.

2.2. SQL-запити

Виконайте запити з файлу `db/queries.sql` і поясніть результат кожного. Приклад запиту з групуванням:

```
SELECT d.name AS district, COUNT(s.id) AS shelters,
       COALESCE(SUM(s.capacity), 0) AS total_capacity
FROM districts d
LEFT JOIN shelters s ON s.district_id = d.id
GROUP BY d.name
ORDER BY total_capacity DESC;
```

2.3. Доступ до БД з Java

1. Перегляньте `pom.xml`: залежність `org.postgresql:postgresql` – драйвер JDBC.
2. Клас `jdbc/Database.java` бере параметри підключення зі змінних оточення `DB_URL`, `DB_USER`, `DB_PASSWORD` (зі значеннями за замовчуванням для локальної роботи).
3. Клас `jdbc/ShelterDao.java` містить методи `findAll`, `findById`, `findByDistrict`, `insert` (з отриманням згенерованого ключа), `updateStatus`, `delete`, `capacityByDistrict`.
4. Клас `jdbc/InspectionDao.java` реєструє перевірку в транзакції: вставляє запис у `inspections` і змінює стан укриття; у разі помилки виконується `rollback()`.
5. Запустіть застосунок і проаналізуйте вивід:

```
mvn compile exec:java
```

```
public void registerInspection(int shelterId, int inspectorId,
        LocalDate date, String result, String notes)
        throws SQLException {
  try (Connection con = Database.getConnection()) {
    con.setAutoCommit(false);
    try (PreparedStatement insert = con.prepareStatement(INSERT_SQL);
         PreparedStatement update = con.prepareStatement(UPDATE_SQL)) {
      // ... встановлення параметрів, insert.executeUpdate()
      if (update.executeUpdate() != 1) {
        throw new SQLException("Укриття не знайдено: " + shelterId);
      }
      con.commit();
    } catch (SQLException e) {
      con.rollback();
      throw e;
    }
  }
}
```

Клас `jdbc/SqlInjectionDemo.java` порівнює небезпечну конкатенацію рядків і `PreparedStatement`: для вхідного рядка `' OR '1'='1` перший метод повертає всі записи, другий – жодного.

2.4. Індивідуальне завдання

Для власної предметної області (погодженої в ЛР 1) у своєму репозиторії:

1. Побудуйте ER-модель: щонайменше **3 сутності** і **2 зв’язки** «один до багатьох»; одна таблиця – «користувачі» (знадобиться в ЛР 8).

2. Створіть файл `db/schema.sql`: первинні та зовнішні ключі, щонайменше по одному обмеженню `NOT NULL`, `UNIQUE`, `CHECK`, один індекс.
3. Створіть файл `db/data.sql`: щонайменше 10 рядків у головній таблиці і по 3–5 у довідникових.
4. Створіть файл `db/queries.sql` з 8 запитами: вибірка з умовою і сортуванням; INNER JOIN двох таблиць; LEFT JOIN; JOIN трьох таблиць; GROUP BY з агрегатною функцією; GROUP BY з HAVING; підзапит; UPDATE і DELETE з умовою.
5. Створіть Maven-проєкт (Java 21) з драйвером JDBC і класами: `Database` (підключення), DAO головної сутності з методами `findAll`, `findById`, пошук за полем, `insert`, `update`, `delete`; метод, що виконує дві зміни в **одній транзакції**; клас `App`, що демонструє роботу всіх методів.
6. Використовуйте лише `PreparedStatement` з параметрами; ресурси закривайте через try-with-resources; пароль БД не записуйте в код.
7. Зафіксуйте зміни, створіть тег `lab-02` і надішліть на GitHub (`git push origin main --tags`).

2.5. Вимоги до звіту

Звіт (PDF або DOC, до 6 сторінок) містить: мету; ER-діаграму (знімок з IDE, DBeaver чи draw.io); текст `schema.sql` з коротким поясненням ключів і обмежень; 8 запитів і знімки результатів; фрагмент коду DAO і транзакції; вивід програми; посилання на репозиторій з тегом `lab-02`; відповіді на 3 контрольні запитання; висновки. У ВУ в завданні «ЛР 2» у поле «Текстова відповідь» вставте посилання на репозиторій, файлом додайте звіт.

2.6. Контрольні запитання

1. Що таке первинний і зовнішній ключ? Як реалізувати зв’язок «багато до багатьох»?
2. Для чого потрібні обмеження CHECK і UNIQUE? Наведіть приклади з вашої схеми.
3. Чим відрізняються INNER JOIN і LEFT JOIN, WHERE і HAVING?
4. Яку роль виконує драйвер JDBC і де його підключено у проєкті?
5. Чому `PreparedStatement` захищає від SQL-ін’єкцій?
6. Що станеться, якщо в транзакції не викликати `commit()`?

7. Як отримати згенерований первинний ключ нового запису?

2.7. Критерії оцінювання (максимально 5 балів)

Бали	Критерій
5	схема коректна (ключі, обмеження, індекс), усі 8 запитів працюють, DAO і транзакція реалізовані через PreparedStatement, тег `lab-02` на GitHub; здобувач вільно пояснює код і запити
4	робота виконана повністю, є незначні недоліки в схемі, запитах чи звіті
3	виконано не менше 60 % (напр. немає транзакції або частини запитів) або є суттєві помилки
0–2	виконано менше 60 % або здобувач не може пояснити поданий код

Робота, подана після встановленого терміну без поважних причин, оцінюється з пониженням на 1 бал.

III. Підведення підсумків заняття

Викладач перевіряє схеми БД і роботу застосунків, відповідає на запитання, звертає увагу на типові помилки (відсутні зовнішні ключі, конкатенація SQL, незакриті з’єднання).

Методичну розробку розробив:

Доцент кафедри ІТтаСЕК, к.т.н. Роман ГОЛОВАТИЙ

Методична розробка обговорена на засіданні кафедри ІТтаСЕК
Протокол № ____ від «____» _____ 20___ р.