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

МЕТОДИЧНА РОЗРОБКА

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

Лекція № 2. Тема 1.2: «Основи баз даних та SQL. Доступ до баз даних з Java (JDBC)»

Мета: сформувати у здобувачів освіти розуміння ролі реляційної бази даних у вебзастосунку, навички проєктування таблиць і зв'язків, написання SQL-запитів та безпечного доступу до бази даних з Java-застосунку засобами JDBC.

Час: 2 години.

Місце заняття: лекційна аудиторія (або онлайн у MS Teams).

Матеріальне забезпечення: мультимедійний проєктор, ПК з доступом до мережі Internet, PostgreSQL, IntelliJ IDEA, презентація до лекції, електронний курс у «Віртуальному університеті».

Навчальні питання:

1. Роль баз даних у вебзастосунках. Реляційні СУБД.
2. Проєктування таблиць: ключі, зв'язки, обмеження цілісності, індекси.
3. Мова SQL: вибірка, з'єднання, групування, зміна даних.
4. Доступ до бази даних з Java: JDBC, PreparedStatement.
5. Транзакції в JDBC.
6. SQL-ін'єкції та захист від них.

Після лекційного заняття здобувачі освіти повинні:

знати: поняття таблиці, первинного і зовнішнього ключа, типи зв'язків; синтаксис основних команд SQL; архітектуру JDBC (DriverManager, Connection, PreparedStatement, ResultSet); властивості транзакцій ACID; механізм SQL-ін'єкції;

вміти: проєктувати схему БД предметної області, писати запити з JOIN і GROUP BY, виконувати запити з Java через PreparedStatement, керувати транзакцією.

Література:

1. Віртуальне навчальне середовище ЛДУ БЖД «Віртуальний університет» [Електронний ресурс]. – Доступний з: https://virt.ldubgd.edu.ua/course/view.php?id=2111
2. PostgreSQL Documentation [Електронний ресурс]. – Доступний з: https://www.postgresql.org/docs/
3. Schildt H. Java: The Complete Reference. 13th ed. – McGraw Hill, 2024. – 1280 p.
4. Хорстманн К. С. Java. Бібліотека професіонала. Т. 1. Основи. 11-е вид. – Київ : Наука і техніка, 2020. – 864 с.
5. OWASP. SQL Injection Prevention Cheat Sheet [Електронний ресурс]. – Доступний з: https://cheatsheetseries.owasp.org

І. Підготовка до заняття

Перед початком заняття викладач перевіряє роботу проєктора, доступ до PostgreSQL і демо-проєкту (тег `lab-02`) для демонстрації запитів і коду.

ІІ. Зміст лекції

Вступ

Статична сторінка з лабораторної роботи № 1 містить дані, «зашиті» в HTML. Щойно потрібно додати укриття, змінити його стан чи знайти всі укриття району, дані треба зберігати окремо від сторінки – у **базі даних**. Майже кожен вебзастосунок – це «інтерфейс до бази даних»: сервер приймає HTTP-запит, виконує SQL-запит до БД і повертає результат клієнтові.

1. Роль баз даних у вебзастосунках. Реляційні СУБД

> **База даних** – упорядкована сукупність даних предметної області. **СУБД** (система управління базами даних) – програмне забезпечення для створення БД, зберігання, пошуку і зміни даних, контролю цілісності та одночасного доступу багатьох користувачів.

Реляційна модель (E. Кодд, 1970) подає дані у вигляді **таблиць** (відношень): рядок – запис про один об’єкт, стовпець – атрибут певного типу. Найпоширеніші реляційні СУБД: **PostgreSQL** (відкрита, потужна, стандарт для нових проєктів), **MySQL/MariaDB** (популярна у вебхостингу), Oracle, Microsoft SQL Server, вбудовані **H2** і SQLite (для розроблення і тестів). Крім реляційних, існують NoSQL-СУБД (MongoDB, Redis), однак для облікових систем із чіткою структурою даних реляційна модель залишається основною.

2. Проєктування таблиць

Проєктування починається з виділення **сутностей** предметної області та зв’язків між ними. Для навчального проєкту «Реєстр укриттів»: район, укриття, перевірка укриття, користувач (інспектор).

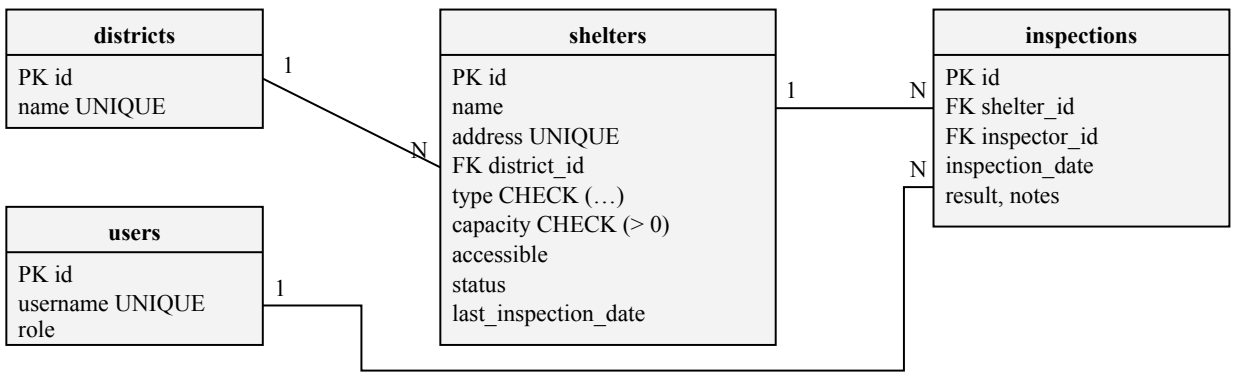

Рисунок 1 – Схема бази даних «Реєстр укриттів» (1 – «один», N – «багато»)

Первинний ключ (PRIMARY KEY) однозначно ідентифікує рядок; зазвичай це штучний цілочисловий `id`, який генерує СУБД (`SERIAL` / `IDENTITY`). **Зовнішній ключ** (FOREIGN KEY) посилається на первинний ключ іншої таблиці і реалізує зв'язок: «один район – багато укриттів» (1:N). Зв'язок «багато до багатьох» (M:N) реалізують окремою проміжною таблицею. **Обмеження цілісності** не дають записати некоректні дані: `NOT NULL`, `UNIQUE`, `CHECK`, `REFERENCES … ON DELETE CASCADE`. **Індекс** прискорює пошук за стовпцем (як алфавітний покажчик у книзі), але сповільнює вставку, тому його створюють для стовпців, за якими часто шукають або з'єднують таблиці.

```
CREATE TABLE shelters (
    id          SERIAL PRIMARY KEY,
    name        VARCHAR(150) NOT NULL,
    address     VARCHAR(255) NOT NULL UNIQUE,
    district_id INT NOT NULL REFERENCES districts (id),
    capacity    INT NOT NULL CHECK (capacity > 0),
    status      VARCHAR(20) NOT NULL DEFAULT 'NOT_READY'
);
CREATE INDEX idx_shelters_district ON shelters (district_id);
```

Лістинг 1 – Створення таблиці з ключами, обмеженнями та індексом

3. Мова SQL

SQL (Structured Query Language) – декларативна мова: ми описуємо, *які* дані потрібні, а СУБД сама обирає, *як* їх знайти. Команди поділяють на групи: **DDL** – визначення структури (`CREATE`, `ALTER`, `DROP`); **DML** – робота з даними (`SELECT`, `INSERT`, `UPDATE`, `DELETE`); **DCL** – права доступу (`GRANT`, `REVOKE`); **TCL** – транзакції (`BEGIN`, `COMMIT`, `ROLLBACK`).

```
-- вибірка з умовою і сортуванням
SELECT name, capacity FROM shelters
WHERE status = 'READY' ORDER BY capacity DESC;

-- з'єднання таблиць: назва району замість district_id
SELECT s.name, d.name AS district
FROM shelters s JOIN districts d ON d.id = s.district_id;

-- групування: місткість за районами, лише райони понад 200 місць
SELECT d.name, COUNT(s.id) AS shelters, SUM(s.capacity) AS total
FROM districts d LEFT JOIN shelters s ON s.district_id = d.id
GROUP BY d.name HAVING SUM(s.capacity) > 200;

-- зміна даних
INSERT INTO shelters (name, address, district_id, capacity)
VALUES ('Укриття № 7', 'вул. Зелена, 1', 4, 70);
UPDATE shelters SET status = 'READY' WHERE id = 7;
DELETE FROM shelters WHERE id = 7;
```

Лістинг 2 – Основні SQL-запити

`JOIN` (INNER JOIN) повертає лише рядки, для яких знайшлася пара; `LEFT JOIN` – усі рядки лівої таблиці, навіть без пари (райони без укриттів отримають `NULL`). `WHERE` фільтрує рядки до групування, `HAVING` – групи після нього. Агрегатні функції: `COUNT`, `SUM`, `AVG`, `MIN`, `MAX`. Підзапит – запит усередині іншого, наприклад укриття з місткістю, більшою за середню: `WHERE capacity > (SELECT AVG(capacity) FROM shelters)`.

4. Доступ до бази даних з Java: JDBC

JDBC (Java Database Connectivity) – стандартний API Java для роботи з реляційними БД. Застосунок працює з інтерфейсами з пакета `java.sql`, а конкретну СУБД підтримує **драйвер** – бібліотека від виробника (для PostgreSQL – `org.postgresql:postgresql`, підключається через Maven). Тому, змінивши драйвер і рядок підключення, можна перейти з PostgreSQL на MySQL без зміни решти коду.

Інтерфейс / клас	**Призначення**
`DriverManager`	створює з'єднання за URL, напр. `jdbc:postgresql://localhost:5432/shelter_registry`
`Connection`	з'єднання з БД; керує транзакціями
`PreparedStatement`	попередньо скомпільований запит з параметрами `?`
`ResultSet`	результат вибірки: курсор, що рухається рядками (`next()`)
`SQLException`	помилка роботи з БД

Таблиця 1 – Основні елементи JDBC

```
public List<ShelterRow> findByDistrict(String district)
        throws SQLException {
    String sql = """
        SELECT s.id, s.name, d.name AS district
        FROM shelters s JOIN districts d ON d.id = s.district_id
        WHERE d.name = ?""";
    try (Connection con = Database.getConnection();
         PreparedStatement ps = con.prepareStatement(sql)) {
        ps.setString(1, district);            // значення параметра
        try (ResultSet rs = ps.executeQuery()) {
            List<ShelterRow> result = new ArrayList<>();
            while (rs.next()) {
                result.add(new ShelterRow(rs.getInt("id"),
                        rs.getString("name"),
                        rs.getString("district")));
            }
            return result;
        }
    }                          // ресурси закриваються автоматично
}
```

Лістинг 3 – Вибірка через PreparedStatement (демо-проєкт, тег lab-02)

Конструкція **try-with-resources** гарантує закриття `ResultSet`, `PreparedStatement` і `Connection` навіть у разі помилки – незакриті з'єднання швидко вичерпують ліміт СУБД. Для `INSERT`, `UPDATE`, `DELETE` викликають `executeUpdate()`, який повертає кількість змінених рядків; згенерований ключ нового запису отримують через `Statement.RETURN_GENERATED_KEYS` і `getGeneratedKeys()`. Параметри підключення (URL, користувач, пароль) не записують у код, а беруть зі змінних оточення чи конфігураційного файлу.

5. Транзакції

> **Транзакція** – група операцій, яка виконується як одне ціле: або всі зміни фіксуються, або жодна. Властивості **ACID**: атомарність (Atomicity), узгодженість (Consistency), ізольованість (Isolation), довговічність (Durability).

Реєстрація перевірки укриття складається з двох дій: вставити запис у `inspections` і змінити стан у `shelters`. Якщо друга дія не вдасться, перша теж має бути скасована. За замовчуванням JDBC фіксує кожен запит окремо (*autocommit*), тому транзакцію керують вручну:

```
try (Connection con = Database.getConnection()) {
    con.setAutoCommit(false);              // початок транзакції
    try {
        insertInspection(con, shelterId, result);
        updateShelterStatus(con, shelterId, result);
        con.commit();                      // зафіксувати обидві зміни
    } catch (SQLException e) {
        con.rollback();                    // скасувати обидві зміни
        throw e;
    }
}
```

Лістинг 4 – Керування транзакцією в JDBC

У Spring Boot (тема 2.1) цей шаблонний код замінить одна анотація `@Transactional`.

6. SQL-ін’єкції та захист від них

SQL-ін’єкція – атака, під час якої дані користувача потрапляють у текст SQL-запиту і змінюють його зміст. Вона входить до переліку найнебезпечніших вразливостей вебзастосунків (OWASP Top 10, категорія Injection). Причина – складання запиту конкатенацією рядків:

```
// НЕБЕЗПЕЧНО
String sql = "SELECT * FROM shelters s JOIN districts d "
           + "ON d.id = s.district_id "
           + "WHERE d.name = '" + district + "'";
// district = "' OR '1'='1"
// →  WHERE d.name = '' OR '1'='1'  →  повертаються ВСІ записи
```

Лістинг 5 – Уразливий код

Якщо замість `' OR '1'='1` зловмисник передасть `'; DROP TABLE shelters; --`, наслідки будуть ще серйознішими. **Захист:** завжди використовувати `PreparedStatement` з параметрами `?` – значення передається окремо від тексту запиту і не може змінити його структуру (у демо-проєкті той самий рядок атаки повертає 0 записів). Додатково: перевіряти вхідні дані, надавати обліковому запису БД мінімальні права, не показувати користувачеві тексти помилок СУБД.

Висновки

Реляційна БД зберігає дані вебзастосунку в таблицях, пов’язаних первинними і зовнішніми ключами; обмеження цілісності не дають записати некоректні дані. SQL дає змогу вибирати, з’єднувати, групувати і змінювати дані. JDBC – стандартний спосіб доступу до БД з Java: `Connection`, `PreparedStatement`, `ResultSet`, транзакції. Параметризовані запити –

обов’язковий захист від SQL-ін’єкцій. Ці самі механізми працюватимуть «під капотом» Spring Data JPA у темі 2.4.

Контрольні запитання

1. Що таке первинний і зовнішній ключ? Як реалізують зв’язки 1:N і M:N?
2. Для чого призначені обмеження NOT NULL, UNIQUE, CHECK?
3. Коли варто створювати індекс і чому не на всіх стовпцях?
4. Чим відрізняються INNER JOIN і LEFT JOIN?
5. Чим відрізняються WHERE і HAVING?
6. Яку роль виконує драйвер JDBC?
7. Навіщо використовувати try-with-resources під час роботи з JDBC?
8. Поясніть властивості ACID. Як керувати транзакцією в JDBC?
9. Як виникає SQL-ін’єкція і як від неї захиститися?

Завдання для самостійної роботи

Опрацювати матеріал лекції. Виконати вправи на складні SQL-запити (з’єднання, групування, підзапити) для власної предметної області. Завершити та оформити лабораторну роботу № 2. Пройти тест № 2 у «Віртуальному університеті».

ІІІ. Підведення підсумків заняття

Узагальнити основні положення лекції, відповісти на запитання здобувачів освіти. Наголосити на зв’язку матеріалу з лабораторною роботою № 2. Видати завдання для самостійної роботи.

Методичну розробку розробив:

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

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