ЛЕКЦІЯ №1.2. ОСНОВИ БАЗ ДАНИХ ТА SQL

Кафедра інформаційних технологій та систем електронних комунікацій
Спеціальність 122 «Комп'ютерні науки», ступінь вищої освіти «бакалавр»
VI семестр
Дисципліна «Web-технології та web-дизайн»
ЗМІСТОВИЙ МОДУЛЬ 1. ВЕБ-РОЗРОБКА ТА БАЗИ ДАНИХ В JAVA

План лекції

1. Основні поняття баз даних та їх роль у сучасному програмуванні
2. Архітектура інформаційних систем та місце баз даних у веб-додатках
3. Реляційна модель даних та системи управління базами даних (RDBMS)
4. Типи даних у реляційних СУБД
5. Нормалізація баз даних
6. Мова SQL та її класифікація
7. Основні SQL-запити: CREATE, INSERT, SELECT, UPDATE, DELETE
8. Операції JOIN та робота з кількома таблицями
9. Агрегатні функції та групування даних
10. Ключі, індекси та обмеження цілісності
11. Транзакції та ACID-властивості
12. Безпека баз даних
13. Використання SQL у Java Spring Boot
14. Практичні рекомендації щодо проєктування баз даних
15. Висновки

Вступ

Практично кожен сучасний веб-додаток працює з інформацією, яку необхідно зберігати, змінювати, аналізувати та швидко отримувати. Дані можуть стосуватися користувачів, товарів,

замовлень, повідомлень, оцінок, фінансових операцій, логів системи, мультимедійних файлів та інших сутностей.

Якщо у невеликих програмах інформацію ще можна зберігати у файлах, то для великих систем цього вже недостатньо. Уявімо інтернет-магазин із десятками тисяч товарів та тисячами клієнтів. Без централізованої системи керування даними така система швидко стане нестабільною, повільною та ненадійною.

Саме тому у сучасному програмуванні використовуються бази даних та системи управління базами даних (СУБД). Вони забезпечують:

- структуроване зберігання інформації;
- швидкий пошук даних;
- одночасну роботу багатьох користувачів;
- захист інформації;
- резервне копіювання;
- контроль цілісності даних.

Для роботи з реляційними базами даних застосовується SQL (Structured Query Language) — стандартна мова структурованих запитів.

Для Java-розробника знання SQL є обов’язковим, оскільки навіть при використанні Spring Data JPA або Hibernate необхідно розуміти:

- як організовані таблиці;
- як формуються зв’язки;
- як оптимізуються запити;
- чому виникають помилки продуктивності;
- як правильно проєктувати структуру БД.

У межах цієї лекції буде розглянуто базові принципи роботи з реляційними базами даних та мовою SQL.

---

1. Основні поняття баз даних та їх роль у сучасному програмуванні

1.1 Поняття бази даних

База даних (БД) — це організована сукупність взаємопов’язаних даних, що зберігаються відповідно до визначених правил та можуть використовуватися багатьма програмами й користувачами.

База даних повинна забезпечувати:

- збереження інформації;
- швидкий доступ;
- зміну даних;
- захист інформації;

- підтримку цілісності;
- одночасну роботу багатьох користувачів.

Приклади баз даних:

Система	Які дані зберігаються
Інтернет-магазин	Користувачі, товари, замовлення
Соціальна мережа	Профілі, повідомлення, фото
Банківська система	Рахунки, платежі, транзакції
Університетська система	Студенти, оцінки, дисципліни
Система ДСНС	Події, виклики, підрозділи

1.2 Система управління базами даних (СУБД)

Система управління базами даних (СУБД, DBMS) — це програмне забезпечення, яке забезпечує створення, зміну та адміністрування баз даних.

Основні функції СУБД:

1. створення таблиць;
2. збереження інформації;
3. виконання SQL-запитів;
4. забезпечення безпеки;
5. резервне копіювання;
6. контроль цілісності;
7. оптимізація продуктивності.

Популярні СУБД:

СУБД	Особливості
MySQL	популярна у веб-розробці
PostgreSQL	потужна open-source СУБД
Oracle Database	корпоративний рівень
Microsoft SQL Server	інтеграція з Windows
SQLite	вбудована компактна БД

У Java-проєктах дуже часто використовують MySQL або PostgreSQL.

1.3 Переваги використання баз даних

1. Усунення дублювання

Якщо інформація зберігається у файлах, однакові дані можуть повторюватися багато разів.

Наприклад:

- ім’я користувача;
- адреса;

- email.

У реляційній базі даних ці дані зберігаються один раз.

---

2. Цілісність даних

СУБД не дозволяє записати некоректні дані.

Наприклад:

- email повинен бути унікальним;
- ціна товару не може бути від’ємною;
- замовлення не може належати неіснуючому користувачу.

---

3. Безпека

Можна:

- створювати ролі;
- задавати права доступу;
- забороняти зміну даних;
- дозволяти лише читання.

---

4. Масштабованість

Бази даних можуть працювати з:

- тисячами користувачів;
- мільйонами записів;
- великими навантаженнями.

---

5. Швидкість роботи

Завдяки індексам пошук відбувається значно швидше.

---

2. Архітектура інформаційних систем та місце баз даних у веб-додатках

У більшості сучасних веб-додатків використовується трирівнева архітектура:

1. Frontend;
2. Backend;
3. Database.

Frontend

Це клієнтська частина:

- HTML;
- CSS;
- JavaScript;
- React;
- Angular;
- Vue.

Frontend відповідає за інтерфейс користувача.

Backend

Серверна частина:

- Java Spring Boot;
- Node.js;
- PHP;
- Python.

Backend:

- обробляє запити;
- виконує бізнес-логіку;
- працює з базою даних.

Database

База даних забезпечує:

- постійне зберігання інформації;
- доступ до даних;
- пошук;
- зміну даних.

Приклад роботи веб-додатка

1. Користувач вводить логін і пароль.
2. Frontend надсилає HTTP-запит.
3. Backend перевіряє дані.
4. Backend виконує SQL-запит.
5. База даних повертає результат.
6. Backend формує відповідь.
7. Frontend відображає результат.

---

3. Реляційна модель даних та системи управління базами даних

3.1 Основи реляційної моделі

Реляційна модель була запропонована Едгаром Коддом у 1970 році.

Основою є поняття таблиці.

Таблиця складається з:

- рядків;
- стовпців.

Приклад таблиці users:

id	username	email
1	ivan	ivan@gmail.com
2	petro	petro@gmail.com

3.2 Основні елементи реляційної моделі

Таблиця

Зберігає дані про певну сутність.

Наприклад:

- users;
- products;
- orders.

Рядок

Окремий запис.

Стовпець

Поле з певним типом даних.

Сутність

Об’єкт предметної області.

Наприклад:

- користувач;
- товар;
- замовлення.

3.3 Зв’язки між таблицями

Один до одного (1:1)

Наприклад:

- користувач — паспорт.

Один до багатьох (1:N)

Найпоширеніший тип.

Наприклад:

- один користувач має багато замовлень.

Багато до багатьох (M:N)

Наприклад:

- студент — дисципліна.

Реалізується через проміжну таблицю.

---

4. Типи даних у реляційних СУБД

Правильний вибір типу даних впливає на:

- швидкість роботи;
- обсяг пам’яті;
- продуктивність;
- коректність даних.

Основні типи даних

Тип	Призначення
INT	цілі числа
BIGINT	великі числа
DECIMAL	фінансові обчислення
VARCHAR	рядки змінної довжини
TEXT	великі тексти
DATE	дата

Тип	Призначення
DATETIME	дата і час
BOOLEAN	true/false

VARCHAR vs TEXT

VARCHAR:

- обмежена довжина;
- швидший пошук.

TEXT:

- великі обсяги тексту;
- використовується для описів.

DECIMAL vs FLOAT

Для грошей необхідно використовувати DECIMAL.

FLOAT може давати похибку.

Приклад:

```
price DECIMAL(10,2)
```

---

5. Нормалізація баз даних

Нормалізація — це процес організації структури БД для:

- зменшення дублювання;
- усунення аномалій;
- підвищення цілісності.

Перша нормальна форма (1NF)

У таблиці:

- не повинно бути повторюваних груп;
- кожне поле містить одне значення.

Неправильно:

user	phones
Ivan	111,222

Правильно:

user	phone
Ivan	111
Ivan	222

Друга нормальна форма (2NF)

Усі неключові поля повинні залежати від усього ключа.

Третя нормальна форма (3NF)

Неключові поля не повинні залежати одне від одного.

---

6. Мова SQL та її класифікація

SQL — стандартна мова роботи з реляційними базами даних.

Основні групи SQL-команд

Група	Призначення
DDL	створення структури
DML	робота з даними
DCL	права доступу
TCL	транзакції

DDL

CREATE
ALTER
DROP

DML

SELECT
INSERT
UPDATE
DELETE

DCL

GRANT
REVOKE

TCL

COMMIT
ROLLBACK
SAVEPOINT

---

7. Основні SQL-запити

7.1 CREATE TABLE

```
CREATE TABLE users (
   id INT AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(50) NOT NULL,
   email VARCHAR(100) UNIQUE,
   password VARCHAR(255) NOT NULL
);
```

Пояснення:

- AUTO_INCREMENT — автоматична генерація id;
- PRIMARY KEY — первинний ключ;
- UNIQUE — унікальність;
- NOT NULL — поле обов’язкове.

7.2 INSERT

```
INSERT INTO users(username, email, password)
VALUES('ivan', 'ivan@gmail.com', '12345');
```

7.3 SELECT

```
SELECT * FROM users;
```

SELECT із WHERE

```
SELECT * FROM users
WHERE id = 1;
```

SELECT із ORDER BY

```
SELECT * FROM users
ORDER BY username ASC;
```

SELECT із LIMIT

```
SELECT * FROM users
LIMIT 10;
```

7.4 UPDATE

```
UPDATE users
SET email = 'new@gmail.com'
WHERE id = 1;
```

7.5 DELETE

```
DELETE FROM users
WHERE id = 1;
```

---

8. Операції JOIN

JOIN дозволяє об’єднувати таблиці.

INNER JOIN

```
SELECT users.username, orders.total
FROM users
INNER JOIN orders
ON users.id = orders.user_id;
```

Повертає лише пов’язані записи.

LEFT JOIN

```
SELECT users.username, orders.total
FROM users
LEFT JOIN orders
ON users.id = orders.user_id;
```

Повертає всіх користувачів.

RIGHT JOIN

Повертає всі записи правої таблиці.

---

9. Агрегатні функції та групування даних

COUNT

```
SELECT COUNT(*) FROM users;
```

SUM

```
SELECT SUM(total) FROM orders;
```

AVG

```
SELECT AVG(total) FROM orders;
```

MAX та MIN

```
SELECT MAX(total) FROM orders;
```

GROUP BY

```
SELECT user_id, COUNT(*)
FROM orders
GROUP BY user_id;
```

---

10. Ключі, індекси та обмеження цілісності

Первинний ключ

```
PRIMARY KEY
```

Унікально ідентифікує запис.

Зовнішній ключ

```
FOREIGN KEY (user_id)
REFERENCES users(id)
```

Індекси

```
CREATE INDEX idx_email
ON users(email);
```

Індекси:

- прискорюють SELECT;
- уповільнюють INSERT;
- займають пам’ять.

UNIQUE

email VARCHAR(100) **UNIQUE**

NOT NULL

username VARCHAR(50) **NOT NULL**

---

11. Транзакції та ACID-властивості

Транзакція — це набір операцій, які виконуються як єдине ціле.

ACID

Atomicity

Або все виконується, або нічого.

Consistency

База залишається коректною.

Isolation

Транзакції не заважають одна одній.

Durability

Після COMMIT дані не губляться.

Приклад транзакції

```
START TRANSACTION;

UPDATE accounts
SET balance = balance - 100
WHERE id = 1;

UPDATE accounts
```

```
SET balance = balance + 100
WHERE id = 2;

COMMIT;
```

---

12. Безпека баз даних

Основні загрози:

- SQL Injection;
- витік даних;
- несанкціонований доступ.

SQL Injection

Небезпечний SQL-код, який вводить користувач.

Небезпечний приклад:

```
String sql = "SELECT * FROM users WHERE username='" + login + "'";
```

Захист

Використання PreparedStatement:

```
PreparedStatement ps = connection.prepareStatement(
   "SELECT * FROM users WHERE username=?"
);
```

---

13. Використання SQL у Java Spring Boot

У Spring Boot робота з БД часто виконується через:

- JDBC;
- Hibernate;
- Spring Data JPA.

JDBC

Низькорівнева робота з SQL.

Hibernate

ORM-фреймворк.

Дозволяє працювати з Java-об’єктами.

Spring Data JPA

Автоматизує доступ до даних.

Entity-клас

```
@Entity
@Table(name = "users")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String username;

    private String email;
}
```

Repository

```
public interface UserRepository
        extends JpaRepository<User, Long> {
}
```

---

14. Практичні рекомендації щодо проєктування баз даних

1. Використовуйте зрозумілі назви

Правильно:

- users;
- orders;
- products.

Неправильно:

- tbl1;
- data_test.

---

2. Завжди використовуйте PRIMARY KEY

---

3. Уникайте дублювання даних

---

4. Створюйте індекси лише там, де потрібно

---

5. Не використовуйте SELECT * у великих системах

Краще:

```
SELECT username, email
FROM users;
```

---

6. Використовуйте транзакції

Особливо для:

- банківських систем;
- оплат;
- фінансових операцій.

Висновки

У межах лекції було розглянуто:

- основні поняття баз даних;
- архітектуру веб-додатків;
- реляційну модель;
- типи даних;
- нормалізацію;
- SQL-запити;
- JOIN;
- агрегатні функції;
- ключі та індекси;
- транзакції;
- безпеку;
- інтеграцію SQL із Java Spring Boot.

Знання баз даних є фундаментальними для backend-розробника. Навіть при використанні сучасних ORM-фреймворків програміст повинен розуміти:

- як працює SQL;
- як організована структура БД;

- як оптимізуються запити;
- як забезпечується цілісність даних.

Надалі ці знання будуть використані при створенні веб-додатків на Java Spring Boot.

Завдання на самопідготовку

1. Встановити MySQL або PostgreSQL.
2. Створити базу даних university_db.
3. Створити таблиці users, orders, products.
4. Написати:
    - 5 SELECT-запитів;
    - 3 INSERT-запити;
    - 3 UPDATE-запити;
    - 3 DELETE-запити.
5. Реалізувати зв’язок один-до-багатьох.
6. Створити індекс для email.
7. Спробувати виконати JOIN-запит.

Література

1. Herbert Schildt. Java: The Complete Reference.
2. Spring in Action.
3. SQL for Data Science (Coursera).
4. MySQL Documentation.
5. PostgreSQL Documentation.
6. Oracle Database Concepts.
7. Elmasri R., Navathe S. Fundamentals of Database Systems.