

ЛАБОРАТОРНА РОБОТА № 7

Перехід від CSV до бази даних: проектування схеми та репозиторій на PDO

Мета роботи: Спроекувати реляційну базу даних для сутностей, з якими застосунок працював у лабораторних роботах № 3–6, наповнити її тестовими даними, і замінити файлове (CSV) зберігання даних на зберігання в базі даних через PDO — не змінюючи жодного рядка коду в API (api.php) чи на SPA-сторінці (index.html) з лабораторної роботи № 6.

Теоретичні відомості

1. Навіщо база даних замість CSV

CSV-файли (лабораторні роботи № 4–6) чудово підходять для навчальних цілей, але мають суттєві практичні обмеження: під час одночасного запису двома користувачами файл можна пошкодити; складні вибірки («усі рюкзаки дорожчі за 2000 грн певної категорії») доводиться реалізовувати вручну в PHP-кодi, а не одним SQL-запитом; цілісність зв'язків (наприклад, що характеристика справді посилається на наявний рюкзак) ніхто не гарантує. Система керування базами даних (тема 7) вирішує всі ці проблеми на рівні самої СУБД.

Обмеження CSV	Як вирішує СУБД
Пошкодження при паралельному записі	Транзакції та блокування на рівні рядків (тема 7)
Складні вибірки й фільтрація	Мова SQL: WHERE, JOIN, ORDER BY одним запитом
Цілісність зв'язків	Зовнішні ключі (FOREIGN KEY), що не дозволяють «висячих» посилань
Дублювання списку характеристик у полі	Окрема таблиця з власними записами й зв'язком через ID

2. Проектування схеми: нормалізація та зв'язки

У CSV-варіанті (лабораторна робота № 4) характеристики рюкзака доводилося зберігати одним текстовим полем із власним розділювачем («Матеріал:Нейлон|Вага:1.2 кг») — компромісний прийом, згадуваний ще тоді як тимчасовий. У реляційній базі даних для цього передбачена окрема таблиця: кожна характеристика — власний рядок із посиланням на рюкзак через `backpack_id`. Це і є нормалізація — уникнення дублювання й довільної структури всередині одного поля.

```
categories (1) —< (N) backpacks (1) —< (N) characteristics
  id, name           id, name,           id, backpack_id,
                    category_id,
                    price, volume_liters,
                    photo
```

Символи «1» і «N» позначають тип зв'язку: одна категорія може мати багато рюкзаків (1:N), один рюкзак може мати багато характеристик (1:N). Це точнісінько той самий аналіз сутностей і зв'язків, що виконується під час курсової роботи з «Організації баз даних» (лабораторна робота № 3) — лабораторні роботи цього курсу свідомо спираються на вже знайому вам методологію проєктування.

3. DDL: CREATE TABLE і зовнішні ключі

DDL (Data Definition Language) — підмножина SQL для опису структури бази даних. Інструкція FOREIGN KEY (тема 7) оголошує зв'язок між таблицями на рівні самої СУБД: спроба додати характеристику з неіснуючим `backpack_id` чи видалити категорію, на яку ще посилається рюкзак, буде відхилена базою даних автоматично, без додаткової перевірки в PHP-коді.

```
CREATE TABLE backpacks (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(150) NOT NULL,  
    category_id INT NOT NULL,  
    price DECIMAL(10,2) NOT NULL,  
    volume_liters DECIMAL(5,1) NOT NULL,  
    photo VARCHAR(255) NULL,  
    FOREIGN KEY (category_id) REFERENCES categories(id)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

ON DELETE CASCADE у зовнішньому ключі таблиці характеристик (розділ «Приклад виконання») означає: якщо рюкзак видаляється, усі його характеристики видаляються автоматично разом із ним — базі даних не потрібна для цього окрема логіка в PHP, на відміну від CSV-варіанту, де про пов'язані записи довелося б піклуватися вручну.

4. Підключення через PDO (нагадування теми 7)

PDO (PHP Data Objects) — єдиний інтерфейс доступу до різних СУБД, уже використаний у темі 7. Підключення створюється один раз (наприклад, у `bootstrap.php`) і передається в конструктор репозиторію — так само, як у CSV-варіанті туди передавались шляхи до файлів.

```
<?php  
$pdo = new PDO(  
    'mysql:host=localhost;dbname=backpacks_shop;charset=utf8mb4',  
    'root',  
    'secret',  
    [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]  
);
```

5. Репозиторій як межа заміни

Уся мета цієї лабораторної роботи зосереджена в одному класі — `BackpackRepository`. Його публічні методи (`findAll()`, `findById()`, `create()`, `createCategory()` тощо)

лишаються **без жодної зміни в назвах чи параметрах** — змінюється лише те, *як* вони отримують і зберігають дані всередині. Це практичний наслідок патерну Repository, згаданого ще в лабораторній роботі № 3: код, що користується репозиторієм (API з лабораторної роботи № 6), узагалі не знає й не повинен знати, зберігаються дані в CSV-файлі чи в базі даних.

Компонент	Змінюється в цій роботі?
Category, Characteristic, Backpack (сутності)	Ні — лишаються повністю без змін
BackpackRepository (реалізація)	Так — CSV-логіка замінюється на SQL-запити через PDO
bootstrap.php	Так — замість шляхів до CSV передається об'єкт PDO
api.php (маршрутизатор API)	Ні — звертається до репозиторію так само, як і раніше
index.html (SPA-сторінка)	Ні — жодних змін, бо API повертає той самий формат JSON

Завдання на лабораторну роботу

Продовжуйте працювати з предметною областю з лабораторних робіт № 3–6.

1. Спроектуйте реляційну схему бази даних для сутностей вашої предметної області: головна сутність, пов'язана довідникова сутність і сутність-список (за зразком категорія / рюкзак / характеристика), з коректними зв'язками через зовнішні ключі.
2. Створіть SQL-скрипт (DDL) зі створенням усіх таблиць бази даних, включно з первинними й зовнішніми ключами та обмеженнями типів даних.
3. Наповніть базу даних тестовими даними (INSERT) — щонайменше стільки ж записів, скільки використовувалось у прикладах попередніх лабораторних робіт.
4. Налаштуйте підключення до бази даних через PDO у bootstrap.php.
5. Перепишіть клас-репозиторій так, щоб усі його методи виконували підготовлені SQL-запити (тема 7) через PDO замість читання й запису CSV-файлів. Публічні сигнатури методів (назви, параметри, типи, що повертаються) повинні лишитися незмінними.
6. Переконайтеся, що api.php та index.html із лабораторної роботи № 6 продовжують працювати без жодної правки в них самих — весь застосунок повинен поводитися для користувача цілком однаково, попри повну заміну сховища даних.

Вимоги до звіту

Звіт повинен містити: тему й мету роботи; ER-схему (чи текстовий опис зв'язків) спроектованої бази даних; DDL-скрипт створення таблиць; скрипт наповнення тестовими даними; лістинг оновленого коду репозиторію та bootstrap.php; підтвердження (знімок екрана), що API й SPA-сторінка з лабораторної роботи № 6 працюють без змін; висновок про переваги переходу на базу даних, помічені особисто під час виконання роботи.

Приклад виконання: «Спортивні рюкзаки»

Приклад продовжує предметну область із попередніх лабораторних робіт. Файли classes/Category.php, classes/Characteristic.php, classes/Backpack.php, api.php і index.html переносяться з лабораторної роботи № 6 **без жодної зміни** — тут наведено лише те, що справді змінюється: схема бази даних, її наповнення, bootstrap.php і BackpackRepository.php.

Структура проєкту

```
lab7/
├── classes/
│   ├── Category.php           (без змін від лаб. № 6)
│   ├── Characteristic.php     (без змін від лаб. № 6)
│   ├── Backpack.php          (без змін від лаб. № 6)
│   └── BackpackRepository.php (повністю переписано на PDO)
├── database/
│   ├── schema.sql            (DDL: створення таблиць)
│   └── seed.sql              (наповнення тестовими даними)
├── uploads/
├── bootstrap.php             (підключення PDO замість шляхів до CSV)
├── api.php                   (без змін від лаб. № 6)
└── index.html                (без змін від лаб. № 6)
```

1. Схема бази даних (database/schema.sql)

```

CREATE TABLE categories (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE backpacks (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(150) NOT NULL,
    category_id INT NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    volume_liters DECIMAL(5,1) NOT NULL,
    photo VARCHAR(255) NULL,
    FOREIGN KEY (category_id) REFERENCES categories(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE characteristics (
    id INT AUTO_INCREMENT PRIMARY KEY,
    backpack_id INT NOT NULL,
    name VARCHAR(100) NOT NULL,
    value VARCHAR(255) NOT NULL,
    FOREIGN KEY (backpack_id) REFERENCES backpacks(id) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

2. Наповнення тестовими даними (database/seed.sql)

```

INSERT INTO categories (id, name) VALUES
    (1, 'Туристичні'),
    (2, 'Спортивні'),
    (3, 'Тактичні');

INSERT INTO backpacks (id, name, category_id, price, volume_liters, photo)
VALUES
    (1, 'TrailMaster 40', 1, 2400.00, 40.0, NULL),
    (2, 'RunLight 15', 2, 1200.00, 15.0, NULL),
    (3, 'TacticalPro 30', 3, 3200.00, 30.0, NULL);

INSERT INTO characteristics (backpack_id, name, value) VALUES
    (1, 'Матеріал', 'Нейлон Cordura'),
    (1, 'Вага', '1.2 кг'),
    (2, 'Матеріал', 'Поліестер'),
    (2, 'Вага', '0.6 кг'),
    (3, 'Матеріал', 'Кордура 1000D'),
    (3, 'MOLLE-система', 'Так');

```

Явне вказування id в INSERT (замість покладання лише на AUTO_INCREMENT) — свідомий вибір для скрипту наповнення тестовими даними: це гарантує, що зв'язки category_id і backpack_id у наступних INSERT-інструкціях точно відповідають щойно доданим записам, незалежно від того, скільки разів скрипт наповнення запускався раніше.

3. Підключення PDO (bootstrap.php)

```

<?php
require_once __DIR__ . '/classes/Category.php';
require_once __DIR__ . '/classes/Characteristic.php';
require_once __DIR__ . '/classes/Backpack.php';
require_once __DIR__ . '/classes/BackpackRepository.php';

$pdo = new PDO(
    'mysql:host=localhost;dbname=backpacks_shop;charset=utf8mb4',
    'root',
    'secret',
    [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]
);

$repository = new BackpackRepository($pdo);

```

Порівняйте цей файл із `bootstrap.php` лабораторної роботи № 4: замість двох рядків шляху до CSV-файлів (`__DIR__ . '/data/categories.csv'` тощо) тепер передається один об'єкт `$pdo`. Це єдина зміна, яку побачить решта застосунку — сам факт існування змінної `$repository` з тим самим інтерфейсом не змінився.

4. Репозиторій: категорії (`classes/BackpackRepository.php`)

```

<?php
class BackpackRepository {
    public function __construct(private PDO $pdo) {}

    public function getCategories(): array {
        $rows = $this->pdo->query('SELECT * FROM categories ORDER BY id')->fetchAll();

        $categories = [];
        foreach ($rows as $row) {
            $categories[(int) $row['id']] = new Category((int) $row['id'], $row['name']);
        }
        return $categories;
    }

    public function findCategoryById(int $id): ?Category {
        $stmt = $this->pdo->prepare('SELECT * FROM categories WHERE id = :id');
        $stmt->execute(['id' => $id]);
        $row = $stmt->fetch();
        return $row ? new Category((int) $row['id'], $row['name']) : null;
    }

    public function createCategory(string $name): Category {
        $stmt = $this->pdo->prepare('INSERT INTO categories (name) VALUES (:name)');
        $stmt->execute(['name' => $name]);
        return new Category((int) $this->pdo->lastInsertId(), $name);
    }
}

```

\$this->pdo->lastInsertId() (тема 7) повертає ID щойно вставленого рядка, згенерований *AUTO_INCREMENT* самою базою даних — той самий за призначенням механізм, яким у CSV-варіанті керував лічильник *\$nextCategoryId*, лише тепер відповідальність за унікальність ідентифікатора повністю передано СУБД.

5. Репозиторій: рюкзаки (продовження)

```
public function findAll(): array {
    $stmt = $this->pdo->query('SELECT * FROM backpacks ORDER BY id');

    $backpacks = [];
    foreach ($stmt->fetchAll() as $row) {
        $backpacks[(int) $row['id']] = $this->hydrateBackpack($row);
    }
    return $backpacks;
}

public function findById(int $id): ?Backpack {
    $stmt = $this->pdo->prepare('SELECT * FROM backpacks WHERE id =
:id');
    $stmt->execute(['id' => $id]);
    $row = $stmt->fetch();
    return $row ? $this->hydrateBackpack($row) : null;
}

public function create(string $name, Category $category, float $price,
float $volume): Backpack {
    $stmt = $this->pdo->prepare(
        'INSERT INTO backpacks (name, category_id, price, volume_liters)
        VALUES (:name, :category_id, :price, :volume)'
    );
    $stmt->execute([
        'name' => $name,
        'category_id' => $category->getId(),
        'price' => $price,
        'volume' => $volume,
    ]);
    return new Backpack((int) $this->pdo->lastInsertId(), $name,
    $category, $price, $volume);
}
```

6. Репозиторій: фото, характеристики й приватний метод «збирання» об'єкта

```

        public function attachPhoto(int $backpackId, string $photoFilename): bool
        {
            $stmt = $this->pdo->prepare('UPDATE backpacks SET photo = :photo
WHERE id = :id');
            return $stmt->execute(['photo' => $photoFilename, 'id' =>
$backpackId]);
        }

        public function addCharacteristicToBackpack(int $backpackId, string
$name, string $value): bool {
            if ($this->findById($backpackId) === null) {
                return false;
            }
            $stmt = $this->pdo->prepare(
                'INSERT INTO characteristics (backpack_id, name, value)
                VALUES (:backpack_id, :name, :value)'
            );
            return $stmt->execute(['backpack_id' => $backpackId, 'name' => $name,
'value' => $value]);
        }

        private function hydrateBackpack(array $row): Backpack {
            $category = $this->findCategoryById((int) $row['category_id']);
            $backpack = new Backpack(
                (int) $row['id'], $row['name'], $category,
                (float) $row['price'], (float) $row['volume_liters']
            );
            if ($row['photo'] !== null) {
                $backpack->setPhoto($row['photo']);
            }

            $charStmt = $this->pdo->prepare('SELECT * FROM characteristics WHERE
backpack_id = :id');
            $charStmt->execute(['id' => $row['id']]);
            foreach ($charStmt->fetchAll() as $charRow) {
                $backpack->addCharacteristic(new Characteristic($charRow['name'],
$charRow['value']));
            }

            return $backpack;
        }
    }
}

```

hydrateBackpack() — приватний допоміжний метод, що «збирає» повноцінний об'єкт *Backpack* із рядка таблиці *backpacks*, довантажуючи пов'язану категорію та всі характеристики окремими запитамі. Саме тут проявляється головна перевага правильної інкапсуляції: цей метод повністю прихований усередині репозиторію — ані *api.php*, ані *index.html* не підозрюють про його існування, так само як вони не знали про внутрішній формат CSV-рядка в лабораторній роботі № 6.

7. Що лишається незмінним

Жоден рядок у *api.php* чи *index.html* (лабораторна робота № 6) не потребує редагування: маршрут GET *api.php?resource=backpacks* усе так само викликає *\$repository->findAll()* і отримує масив об'єктів *Backpack*, кожен із яких так само вміє

`toArray()` для серіалізації в JSON — той факт, що дані щойно надійшли з бази даних, а не з CSV-файлу, для решти застосунку абсолютно непомітний.

Контрольні запитання

- Чому список характеристик, який у CSV-варіанті зберігався одним текстовим полем, у реляційній базі даних виноситься в окрему таблицю?
- Що саме гарантує СУБД завдяки оголошенню FOREIGN KEY (category_id) REFERENCES categories(id), чого CSV-варіант не гарантував?
- Для чого потрібне ON DELETE CASCADE у зовнішньому ключі таблиці characteristics і що станеться без нього при видаленні рюкзака?
- Чому методи репозиторію не змінили назв, параметрів чи типів результату, попри повну заміну внутрішньої реалізації?
- Яку роль виконує `$this->pdo->lastInsertId()` і чим вона схожа на лічильник `$nextId` із CSV-варіанту репозиторію?
- Для чого потрібен приватний метод `hydrateBackpack()` і чому він не публічний?
- Чому саме відсутність змін у `api.php` та `index.html` є головним доказом правильно спроектованого репозиторію, а не просто приємним бонусом?
- Який із двох варіантів (CSV чи база даних) було б природніше обрати для запису, до якого одночасно звертаються десятки користувачів, і чому?