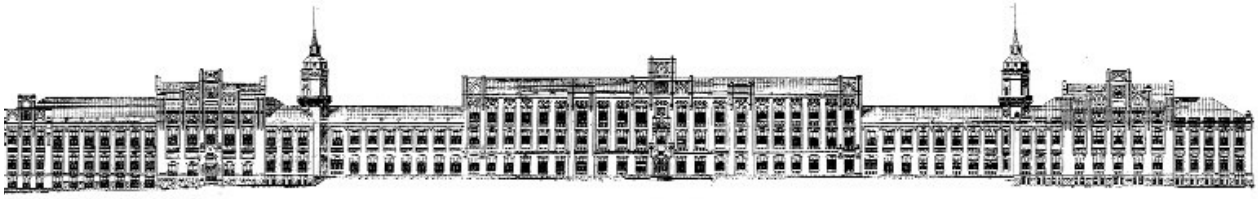

Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №2

з дисципліни «Технології розроблення

програмного забезпечення»

Тема: «Основи проектування»

Виконала
студентка групи ІА-32:
Павлюк В. О.

Зміст

Вступ

Темою роботи є «Основи проєктування». Мета цієї лабораторної це обрати зручний інструмент для побудови UML-діаграм і на його основі навчитися послідовно моделювати систему: від діаграми варіантів використання та коротких сценаріїв до діаграми класів предметної області.

Як об’єкт проєктування обрано тему «Web crawler»: програмний сканер, що розпізнає структуру сторінок сайту, переходить за посиланнями, збирає інформацію за заданим терміном, видаляє несемантичні елементи (рекламу, скрипти тощо), зберігає очищені дані у вигляді структурованого набору HTML-файлів і веде статистику відвідань та метадані.

1 Теоретичні відомості

1.1. Вступ до мови UML

Мова UML є загальноцільовою мовою візуального моделювання, яка розроблена для специфікації, візуалізації, проєктування та документування компонентів програмного забезпечення, бізнес-процесів та інших систем [3]. Мова UML є досить строгим та потужним засобом моделювання, який може бути ефективно використаний для побудови концептуальних, логічних та графічних 18 моделей складних систем різного цільового призначення. Ця мова увібрала в себе найкращі якості та досвід методів програмної інженерії, які з успіхом використовувалися протягом останніх років при моделюванні великих та складних систем.

З погляду методології ООАП (об'єктно-орієнтованого аналізу та проєктування) досить повна модель складної системи є певною кількістю взаємопов'язаних уявлень (views), кожне з яких відображає аспект поведінки або структури системи. У цьому найзагальнішими уявленнями складної системи прийнято вважати статичне і динамічне, які у своє чергу можуть поділятися інші більш приватні.

Принцип ієрархічної побудови моделей складних систем передбачає розгляд процес побудови моделей на різних рівнях абстрагування або деталізації в рамках фіксованих уявлень.

Рівень представлення (layer) – спосіб організації та розгляду моделі на одному рівні абстракції, що представляє горизонтальний зріз архітектури моделі, тоді як розбиття представляє її вертикальний зріз.

При цьому вихідна або початкова модель складної системи має найбільш загальне уявлення та відноситься до концептуального рівня. Така модель, що отримала назву концептуальної, будується на початковому етапі проєктування і може не містити багатьох деталей та аспектів системи, що моделюється. Наступні моделі конкретизують концептуальну модель, доповнюючи її уявленнями логічного та фізичного рівня.

Загалом процес ООАП можна розглядати як послідовний перехід від розробки найбільш загальних моделей та уявлень концептуального рівня до більш приватних і детальних уявлень логічного та фізичного рівня. У цьому кожному етапі ООАП дані моделі послідовно доповнюються дедалі більше

деталей, що дозволяє їм адекватно відбивати різні аспекти конкретної реалізації складної системи. В рамках мови UML уявлення про модель складної системи фіксуються у вигляді спеціальних графічних конструкцій, що отримали назву діаграм.

Діаграма (diagram) – графічне уявлення сукупності елементів моделі у формі зв'язкового графа, вершинам і ребрам (дугам) якого приписується певна семантика. Нотація канонічних діаграм є основним засобом розробки моделей мовою UML.

У нотації мови UML визначено такі види діаграм:

• варіантів використання (use case diagram);

• класів (class diagram); • кооперації (collaboration diagram);

• послідовності (sequence diagram);

• станів (statechart diagram);

• діяльності (activity diagram); • компонентів (component diagram);

• розгортання (deployment diagram).

Перелічені діаграми є невід'ємною частиною графічної нотації мови UML. Понад те, процес ООАП нерозривно пов'язаний з процесом побудови цих діаграм. При цьому сукупність побудованих таким чином діаграм є самодостатньою в тому сенсі, що в них міститься вся інформація, яка потрібна для реалізації проєкту складної системи.

Кожна з цих діаграм деталізує та конкретизує різні уявлення про модель складної системи у термінах мови UML. При цьому діаграма варіантів використання являє собою найбільш загальну концептуальну модель складної системи, яка є вихідною для побудови інших діаграм. Діаграма класів, за своєю суттю, логічна модель, що відбиває статичні аспекти структурної побудови складної системи.

Діаграми кооперації та послідовностей є різновидами логічної моделі, які відображають динамічні аспекти функціонування складної системи. Діаграми станів та діяльності призначені для моделювання поведінки системи. І, нарешті, 20 діаграми компонентів і розгортання служать уявлення фізичних компонентів складної системи і тому представляють її фізичну модель.

1.2. Діаграма варіантів використання (Use-Cases Diagram)

Діаграма варіантів використання (Use-Cases Diagram) – це UML діаграма за допомогою якої у графічному вигляді можна зобразити вимоги до системи, що розробляється [3]. Діаграма варіантів використання – це вихідна концептуальна модель проєктованої системи, вона не описує внутрішню побудову системи.

Діаграми варіантів використання призначені для:

• визначення загальної межі функціональності проєктованої системи;

• формулювання загальних вимоги до функціональної поведінки проєктованої системи;

• подальшої розробка вихідної концептуальної моделі системи (діаграми класів);

• створення основи для виконання аналізу, проєктування, розробки та тестування.

Діаграми варіантів використання є відправною точкою при збиранні вимог до програмного продукту та його реалізації. Дана модель будується на аналітичному етапі побудови програмного продукту (збір та аналіз вимог) і дозволяє бізнес-аналітикам отримати більш повне уявлення про необхідне програмне забезпечення та документувати його. Діаграма варіантів використання складається з низки елементів. Основними елементами є: варіанти використання або прецеденти (use case), актор або дійова особа (actor) та відносини між акторами та варіантами використання (relationship).

1.3. Актори (actor)

Актором називається будь-який об'єкт, суб'єкт чи система, що взаємодіє з модельованою бізнес-системою ззовні для досягнення своїх цілей або вирішення 21 певних завдань [3]. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка служить джерелом впливу на систему, що моделюється. Актора можна розглядати як роль, яка виконується людиною в системі. Зображення акторів на діаграмах представлено на рисунку 2.1.

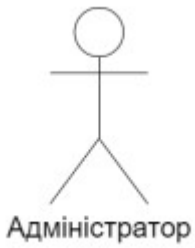

Рисунок 2.1. Зображення акторів на діаграмах UML

Ім'я актора має бути достатньо інформативним з точки зору програмного забезпечення, що розробляється, або предметної області. Для цього підходять найменування посад у компанії (наприклад, продавець, касир, менеджер, президент).

1.4. Варіанти використання (use case)

Варіант використання служить для опису служб, які система надає актору. Інакше кажучи кожен варіант використання визначає набір дій, здійснюваний системою під час діалогу з актором. Кожен варіант використання являє собою послідовність дій, який повинен бути виконаний системою, що

проєктується при взаємодії її з відповідним актором, самі ці дії не відображаються на діаграмі.

Варіант використання відображається еліпсом, всередині якого міститься його коротке ім'я з великої літери у формі іменника або дієслова. Приклад зображення варіанта використання на діаграмі представлено на рисунку 2.2.

Рисунок 2.2. Позначення варіанта використання на діаграмах

При проєктуванні варіантів використання, потрібно приймати до уваги, що з назви варіанта використання має бути зрозуміла тривалість та результат його виконання. Приклади варіантів використання: реєстрація, авторизація, оформлення замовлення, переглянути замовлення, перевірка стану поточного рахунку тощо.

1.5. Відносини на діаграмі варіантів використання

На діаграмі use case відображаються відносини:

- Асоціація – зв’язок між актором і варіантом використання (ненаправлена або спрямована).
- Узагальнення – нащадок успадковує властивості від батька (актори або use case).
- Включення (include) – базовий варіант завжди містить інший.
- Розширення (extend) – базовий варіант може бути доповнений іншим за умови.

1.6. Сценарії використання

Сценарій використання — текстовий покроковий опис взаємодії користувача з системою. Містить: передумови, постумови, сторони, опис, основні кроки, винятки, примітки. Приклад: вхід студента в систему з перевіркою логіну й пароля.

1.7. Діаграми класів

Діаграма класів описує структуру системи: класи, атрибути, методи та відносини між ними.

Види зв’язків:

- асоціація (зв’язок між класами),
- узагальнення (успадкування),
- агрегація (частина–ціле, частини існують окремо),
- композиція (частини не існують без цілого).

1.8. Логічна структура бази даних

Бази даних мають:

- Фізичну модель — файли й структура зберігання на диску.
- Логічну модель — таблиці, індекси, зв’язки у СУБД.

При проєктуванні застосовують нормалізацію:

- 1НФ — атрибути атомарні,
- 2НФ — залежність від усього ключа,
- 3НФ — немає транзитивних залежностей,
- НФ Бойса-Кодда — кожен детермінант є ключем.

2 Хід роботи

2.1. Діаграма варіантів використання

Рисунок 2.1.1 - Діаграма варіантів використання

Актори

Користувач у системі веб-сканера виконує такі дії: налаштовує профіль сканування, запускає процес сканування, переглядає отримані результати, експортує дані у потрібному форматі та переглядає статистику роботи системи.

Адміністратор керує користувачами системи, а також відповідає за налаштування системних обмежень, контролюючи доступ і параметри роботи веб-сканера.

2.2 Варіанти використання (Use Cases)

Для актора Користувач:

1. Налаштувати параметри сканування
    - Вказати стартові URL.
    - Визначити глибину обходу.
    - Задати умови фільтрації (домен, типи ресурсів, мови тощо).
    - Вказати пошуковий термін / ключові слова.
2. Запустити сканування
    - Передати системі збережений профіль налаштувань або поточні параметри.
    - Запустити процес обходу сайтів.
3. Переглянути результати сканування
    - Перегляд списку сторінок, у яких знайдено вказаний термін.
    - Перегляд фрагментів контенту.
    - Перегляд метаданих (дата сканування, час відповіді, розмір сторінки).
4. Експортувати результати
    - Експорт у формат HTML/CSV/JSON.
    - Збереження локального структурованого набору HTML-файлів.
5. Переглянути статистику
    - Кількість відвіданих сторінок.
    - Розподіл за доменами.
    - Частота появи пошукового терміну.

- Середній час відповіді.

Для актора Адміністратор:

6. Керувати користувачами
    - Додавання / блокування облікових записів.
    - Призначення ролей.
7. Налаштувати системні параметри
    - Обмеження на кількість паралельних потоків.
    - Використання проксі-серверів.
    - Інтервали повторних сканувань.

2.3. Написати 3 сценарії використання.

Сценарій 1: «Налаштувати параметри сканування»
Актор: Користувач.
Передумови: Користувач авторизований у системі.
Основний потік: Користувач відкриває форму налаштувань сканування, вводить стартовий URL або кілька URL, задає глибину сканування, вводить термін або набір ключових слів для пошуку, вибирає фільтри (обмеження по домену, виключення файлів певних типів, ігнорування дублікатів) та зберігає налаштування як профіль. Система валідує введені дані, записує профіль у базу даних і повідомляє про успішне збереження.
Помилкові ситуації: Якщо стартовий URL введено у некоректному форматі, система повідомляє про помилку та просить виправити дані. Якщо ключові слова порожні або містять недопустимі символи, система вимагає ввести коректний термін. Якщо глибина сканування перевищує дозволені межі, система підсвічує помилкове поле та блокує збереження. Якщо під час спроби записати профіль у базу даних виникає технічна помилка, система повідомляє про збій та пропонує повторити дію.
Результат: Створено коректний профіль налаштувань сканування або користувач отримує повідомлення про помилку.

Сценарій 2: «Запустити сканування»
Актор: Користувач.
Передумови: У системі існує хоча б один збережений профіль налаштувань.
Основний потік: Користувач відкриває список профілів сканування, обирає потрібний профіль і запускає сканування. Система ініціалізує чергу URL згідно з параметрами профілю, створює необхідні обчислювальні потоки та починає опрацьовувати сторінки. Для кожного URL система виконує HTTP-запит, отримує HTML-сторінку, очищає її від несемантичних елементів, фільтрує контент та зберігає результат у сховищі. Нові посилання додаються до черги відповідно до заданих умов фільтрації. Після завершення сканування система оновлює статистику та зберігає мітку часу останнього виконання профілю.
Помилкові ситуації: Якщо профіль сканування пошкоджений або не містить необхідних параметрів, система повідомляє, що профіль є некоректним, і пропонує його відредагувати. Якщо деякі URL недоступні або сервери не відповідають, система фіксує помилку, пропускає такі адреси та продовжує обробку. Якщо системі не вдається створити необхідну кількість потоків через нестачу ресурсів, користувач отримує повідомлення про перевантаження та рекомендацію знизити параметри сканування.
Результат: Сканування успішно виконано та збережено, або процес завершується з повідомленням про помилку.

Сценарій 3: «Переглянути результати сканування»
Актор: Користувач.
Передумови: У системі є хоча б один завершений сеанс сканування.
Основний потік: Користувач переходить у розділ результатів, обирає потрібний сеанс зі списку, після чого система відображає перелік сторінок, де було знайдено пошукові терміни, разом із короткою інформацією: URL, заголовком сторінки, датою сканування та релевантним фрагментом тексту. Користувач вибирає конкретну сторінку, а система показує її детальний вміст, очищений від несуттєвих елементів, з виділеними входженнями ключових слів та метаданими. За необхідності користувач може експортувати результати у вибраному форматі.
Помилкові ситуації: Якщо результати сканування пошкоджені або недоступні, система повідомляє, що дані не можуть бути завантажені. Якщо в окремої сторінки відсутній контент або він пошкоджений, система показує відповідне повідомлення, але залишає доступними метадані. Якщо під час експорту виникає помилка, система інформує користувача, що файл неможливо сформувати або зберегти.
Результат: Користувач переглянув або експортував результати, або отримав повідомлення про помилку.

2.4. Діаграма класів предметної області.

Основні класи системи web crawler:

- Crawler
    - Атрибути: id, name
    - Операції: startScan(profile), stopScan()
    - Відповідає за загальну координацію процесу сканування.
- CrawlProfile
    - Атрибути: id, name, startUrls, maxDepth, domainFilter, keywords, createdAt
    - Операції: validate()
    - Містить налаштування сканування.
- UrlQueue
    - Атрибути: id, profileId
    - Операції: enqueue(url), dequeue(), isEmpty()
    - Реалізує чергу URL для обходу.
- PageFetcher
    - Операції: fetch(url): HtmlPage
    - Відповідає за отримання HTML-сторінок по HTTP.
- HtmlPage
    - Атрибути: url, rawContent, httpStatus, contentType, fetchedAt.
- ContentFilter
    - Операції: removeAds(page), removeScripts(page), extractText(page)

    - Забезпечує очищення сторінки від несемантичних елементів.

- ParsedPage
    - Атрибути: id, url, plainText, keywordsFound, scanSessionId.
- ScanSession
    - Атрибути: id, profileId, startTime, endTime, status.
    - Операції: markCompleted(), markFailed().
- StatisticsService
    - Операції: calculateDomainStats(session), calculateKeywordStats(session).
- IRepository
    - Операції: save(T), findById(id), findAll(), delete(id).
- Конкретні репозиторії:
    - ProfileRepository : IRepository<CrawlProfile>
    - ScanSessionRepository : IRepository<ScanSession>
    - ParsedPageRepository : IRepository<ParsedPage>

Зв’язки:

- Crawler агрегує UrlQueue, використовує PageFetcher, ContentFilter, ParsedPageRepository, ScanSessionRepository.
- ScanSession асоційований з CrawlProfile.
- ParsedPage асоційований із ScanSession.
- StatisticsService використовує ParsedPageRepository для побудови статистики.

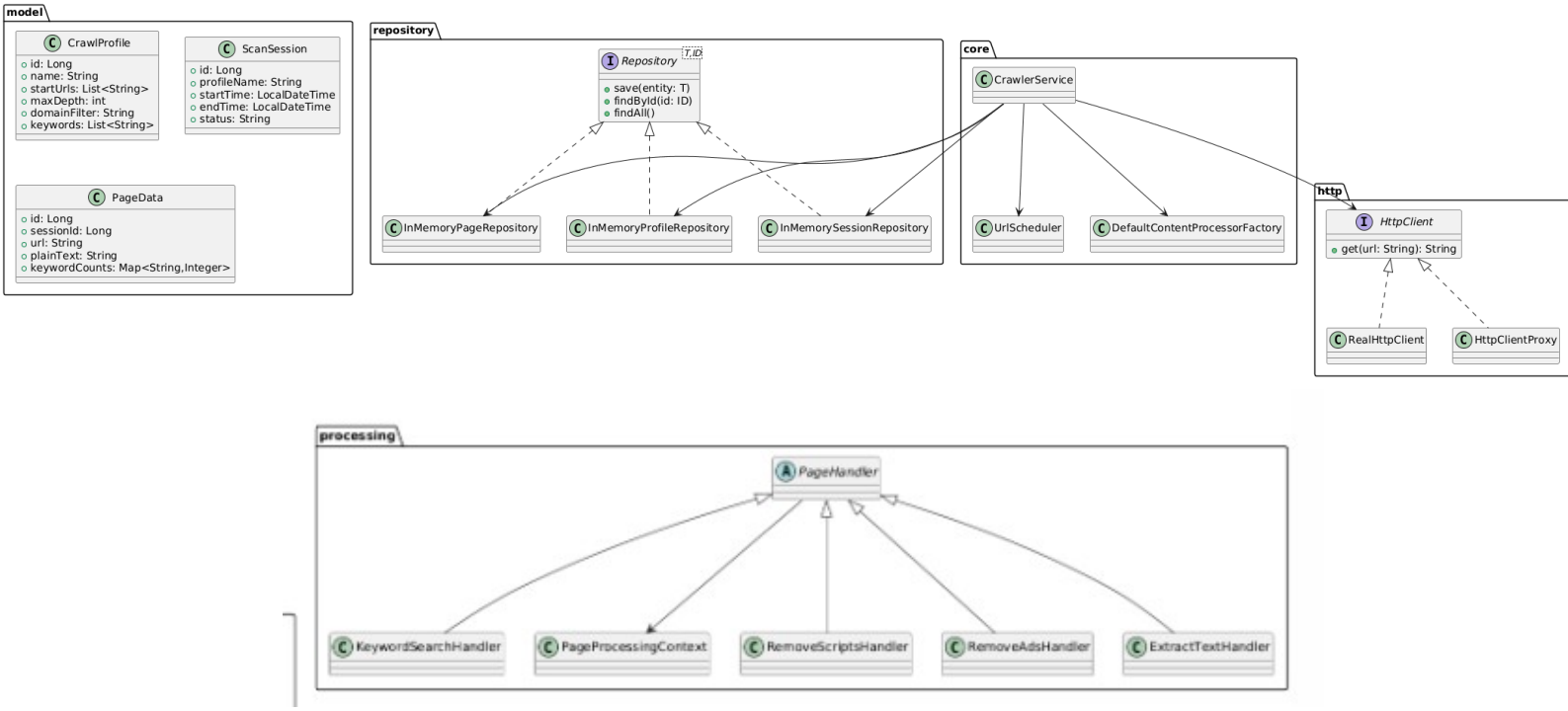

Рисунок 2.4.1 - Комбінована діаграма класів

2.5. Структура бази даних

1. profiles
    - id (PK)
    - name
    - start_urls (текст/JSON)
    - max_depth
    - domain_filter
    - keywords
    - created_at
2. scan_sessions
    - id (PK)
    - profile_id (FK → profiles.id)

- start_time
- end_time
- status

3. pages
    - id (PK)
    - scan_session_id (FK → scan_sessions.id)
    - url
    - http_status
    - content_type
    - fetched_at
    - plain_text
    - keywords_found (наприклад, JSON зі статистикою по ключових словах)
4. statistics
    - id (PK)
    - scan_session_id (FK → scan_sessions.id)
    - metric_name
    - metric_value

2.6. Приклад фрагмента коду (Java, Repository)

```
public interface Repository<T, ID> {
   void save(T entity);
   Optional<T> findById(ID id);
   List<T> findAll();
   void delete(ID id);
}
```

```
public class CrawlProfile {
    private Long id;
    private String name;
    private List<String> startUrls;
    private int maxDepth;
    private String domainFilter;
    private List<String> keywords;

    // getters, setters, validate()
}

public class JdbcProfileRepository implements Repository<CrawlProfile, Long> {
    private final DataSource dataSource;

    public JdbcProfileRepository(DataSource dataSource) {
        this.dataSource = dataSource;
    }

    @Override
    public void save(CrawlProfile profile) {
        // SQL INSERT/UPDATE логіка
    }

    @Override
    public Optional<CrawlProfile> findById(Long id) {
        // SQL SELECT логіка
        return Optional.empty();
    }

    @Override
    public List<CrawlProfile> findAll() {
        // SQL SELECT * FROM profiles
        return List.of();
    }

    @Override
    public void delete(Long id) {
        // SQL DELETE
    }
}
```

Висновки

У ході лабораторної роботи я проаналізувала тему “Web crawler” і спроєктувала модель системи. Побудувала діаграму варіантів використання та діаграму класів для системи. Для ключових варіантів використання були розроблені детальні сценарії, а також визначено структуру бази даних та реалізацію шаблону Repository для взаємодії з даними. Отримані результати формують основу предметної моделі та структури системи, необхідної для подальшого проєктування і реалізації.

Контрольні запитання

1. Що таке UML?

UML (Unified Modeling Language) - універсальна мова візуального моделювання для специфікації, проєктування, візуалізації та документування систем.

2. Що таке діаграма класів UML?

Діаграма класів - статичне представлення структури системи: класи, їх атрибути, операції та зв’язки.

3. Які діаграми UML називають канонічними?

Канонічні діаграми: варіантів використання, класів, кооперації, послідовності, станів, діяльності, компонентів, розгортання.

4. Що таке діаграма варіантів використання?

Діаграма варіантів використання відображає вимоги: актори, use case та їхні відношення.

5. Що таке варіант використання?

Варіант використання - послідовність дій, яку система виконує у взаємодії з актором для досягнення мети.

6. Які відношення можуть бути відображені на діаграмі використання?

Відношення: асоціація, узагальнення, включення (include), розширення (extend).

7. Що таке сценарій?

Сценарій використання - текстовий покроковий опис виконання use case (передумови, кроки, винятки, результат).

8. Що таке діаграма класів?

Діаграма класів - графічна модель класів із їх атрибутами, операціями та зв’язками (асоціації, узагальнення, агрегації, композиції).

9. Які зв’язки між класами ви знаєте?

Зв’язки між класами: асоціація, узагальнення (успадкування), агрегація, композиція.

10. Чим відрізняється композиція від агрегації?

Агрегація - слабкий зв’язок «частина-ціле», частини можуть існувати окремо; композиція - сильний зв’язок, частини не існують без цілого.

11. Чим відрізняється зв'язки типу агрегації від зв'язків композиції на діаграмах класів?

На діаграмі: агрегація - порожній ромб біля цілого; композиція - зафарбований ромб біля цілого.

12. Що являють собою нормальні форми баз даних?

Нормальні форми - правила структурування таблиць для зменшення надмірності: 1НФ (атомарні значення), 2НФ (повна функціональна залежність від усього ключа), 3НФ (без транзитивних залежностей), НФ Бойса-Кодда (кожен детермінант - ключ).

13. Що таке фізична модель бази даних? Логічна?

Фізична модель - файлові/дискові структури зберігання даних; логічна модель - таблиці, зв’язки, індекси в СУБД.

14. Який взаємозв'язок між таблицями БД та програмними класами?

Таблиці відповідають сутностям/класам: найчастіше один клас - одна таблиця; можливі варіанти один клас - кілька таблиць або одна таблиця - кілька класів.