Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут» імені Ігоря Сікорського
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота № 8

із дисципліни: «Технології розроблення програмного забезпечення»

на тему: ШАБЛОНИ «COMPOSITE», «FLYWEIGHT», «INTERPRETER», «VISITOR»

Виконав:
студент групи ІА-34
Чунаков А.Р.
Перевірив:
Мягкий Михайло Юрійович

Київ-2025

Мета: Вивчити структуру шаблонів «COMPOSITE», «FLYWEIGHT», «INTERPRETER», «VISITOR» та навчитися застосовувати їх в реалізації програмної системи.

Короткі теоретичні відомості:

Шаблони доступу до даних (Data Access Patterns)

Шаблон «Active Record» (Активний запис) — це шаблон, у якому об'єкт є безпосередньою обгорткою над рядком у таблиці бази даних або представленням (view). Він інкапсулює доступ до даних і додає до них логіку домену, тобто методи на кшталт Save(), Update() або Delete() знаходяться прямо в цьому об'єкті. Цей підхід ідеальний для простих додатків, де бізнес-логіка нескладна і тісно пов'язана зі структурою БД, але стає проблематичним у великих системах через порушення принципу єдиної відповідальності.

Шаблон «Table Data Gateway» (Шлюз до даних таблиці) — це об'єкт, який діє як шлюз до певної таблиці бази даних, інкапсулюючи всі SQL-запити (SELECT, INSERT, UPDATE, DELETE) до неї. На відміну від Active Record, один екземпляр шлюзу обробляє всі рядки таблиці, а методи зазвичай повертають не об'єкти доменної області, а структури даних типу RecordSet, DataTable або масиви. Цей патерн часто використовується як прошарок між базою даних та бізнес-логікою, коли бізнес-об'єкти не повинні містити SQL-код.

Шаблон «Data Mapper» (Перетворювач даних) — це шаблон, який виступає посередником між об'єктами в пам'яті та схемою бази даних, переміщуючи дані між ними та ізолюючи їх одне від одного. Завдяки цьому ні об'єкти доменної області не знають про існування бази даних, ні база даних не залежить від структури об'єктів. Це дозволяє змінювати структуру БД або бізнес-логіку незалежно, що є стандартом для складних корпоративних систем та DDD (Domain-Driven Design).

Структурні та поведінкові шаблони (GoF)

Шаблон «Composite» (Компонувальник — це патерн, який дозволяє згрупувати об'єкти у деревоподібні структури для представлення ієрархій «частина-ціле». Головна ідея полягає в тому, щоб дозволити клієнтському коду працювати з окремими об'єктами та їх групами (композиціями) однаково, через спільний інтерфейс. Класичним прикладом є файлова система, де папка може містити файли та інші папки, і операція «отримати розмір» працює ідентично для обох типів елементів.

Шаблон «Flyweight» (Пристосуванець) — це патерн, призначений для ефективної підтримки великої кількості дрібних об'єктів шляхом розділення їхнього стану на внутрішній (спільний для багатьох об'єктів, незмінний) та зовнішній (унікальний, контекстний). Замість зберігання однакових даних у кожному об'єкті, вони виносяться в окремі спільні екземпляри, що дозволяє

суттєво економити оперативну пам'ять, наприклад, при малюванні мільйонів символів у текстовому редакторі або дерев у грі.

Шаблон «Interpreter» (Інтерпретатор) — це патерн, який використовується для визначення граматики простої мови та інтерпретації речень цієї мови. Він будує абстрактне синтаксичне дерево, де кожен вузол представляє правило граматики, і дозволяє виконувати над цим деревом операції. Патерн часто застосовується для розробки парсерів SQL, математичних виразів або регулярних виразів, хоча для складних мов він може бути неефективним.

Шаблон «Visitor» (Відвідувач) — це патерн, який дозволяє додавати нові операції до об'єктів стабільної структури класів, не змінюючи самі ці класи (це саме те, що ми використали для аналізу та порівняння схем БД). Суть полягає у створенні зовнішнього класу-відвідувача, який реалізує потрібну операцію для кожного типу елемента структури, а елементи лише надають метод Accept, який "впускає" відвідувача до себе, реалізуючи механізм подвійної диспетчеризації.

Обрана тема лабораторної роботи: **SQL IDE**

9. **Sql IDE** (singleton, command, observer, bridge, visitor, SOA)

Інтегроване середовище розробки скриптів SQL повинна вміти працювати з більшістю сучасних баз даних (Microsoft SQL server, Oracle, IBM DB2, MySQL, PostgreSQL), виділяти синтаксичні одиниці конструкцій, виконувати і показувати результат їх виконання, отримувати схеми даних з джерела (список таблиць, стовпців, ключів і т.д.), порівнювати схеми даних з різних джерел.

Діаграма класів реазілації шаблону Visitor

Хід роботи:

1. Реалізація частини функціоналу робочої програми у вигляді класів та їх взаємодій

Для досягнення функціональної можливості виконання різнорідних операцій над структурою бази даних (відображення у вигляді тексту та порівняння схем) без зміни класів самої структури, було розроблено та налаштовано взаємодію наступних класів:

ISchemaElement: Це базовий інтерфейс для всіх частин структури бази даних. Він містить єдиний метод Accept(ISchemaVisitor visitor). Це двері, через які елемент впускає до себе алгоритм-відвідувач.

Посилання на код: https://github.com/1chuny/SQL-IDE/blob/main/Models/Schema/ISchemaElement.cs

SchemaRoot, TableElement, ColumnElement (Конкретні Елементи): Це класи даних, що описують ієрархічну модель:

- SchemaRoot: Кореневий об'єкт, містить список таблиць.
- TableElement: Описує таблицю, містить список стовпців.
- ColumnElement: Описує окремий стовпець. Усі вони реалізують метод Accept. У цьому методі вони викликають відповідний метод відвідувача , передаючи посилання на себе.

Посилання на код: папка https://github.com/1chuny/SQL-IDE/tree/main/Models/Schema

ISchemaVisitor (Інтерфейс Відвідувача): Це контракт, який визначає набір методів Visit(...) для кожного конкретного типу елемента структури. Це дозволяє додавати нові операції, не змінюючи класи елементів.

Посилання на код: https://github.com/1chuny/SQL-IDE/blob/main/Patterns/Visitor/ISchemaVisitor.cs

SchemaTextRendererVisitor, SchemaComparisonVisitor (Конкретні Відвідувачі): Це класи, які містять реальну логіку обробки:

- SchemaTextRendererVisitor: Обходить дерево об'єктів і формує текстовий рядок для звіту.
- SchemaComparisonVisitor: Обходить одну схему, паралельно перевіряючи іншу, і формує список відмінностей.

Посилання на код: https://github.com/1chuny/SQL-IDE/tree/main/Patterns/Visitor

Опис взаємодії:

1. Драйвер бази даних будує об'єктну модель схеми (SchemaRoot), що містить список таблиць та стовпців.
2. Клієнтський код (UI) створює потрібного відвідувача.
3. Клієнт викликає метод Accept(visitor) у кореневого об'єкта схеми.
4. SchemaRoot не знає, що робить відвідувач, він просто каже йому: "Відвідай мене" (visitor.Visit(this)).
5. Відвідувач виконує свою логіку для кореня, а потім сам ініціює обхід вкладених елементів.
6. Процес повторюється рекурсивно до найнижчого рівня.
7. Відвідувач накопичує результат, який потім повертається клієнту.

2. Застосування шаблону «Відвідувач» (Visitor) при реалізації програми

У проєкті шаблон **Visitor** було використано для розділення структури даних (схема БД) від алгоритмів її обробки.

Роль Елемента (Element): Інтерфейс ISchemaElement. Усі класи моделі (SchemaRoot, TableElement, ColumnElement) реалізують цей інтерфейс, надаючи метод Accept(ISchemaVisitor visitor). Цей метод використовує механізм подвійної диспетчеризації, передаючи посилання на себе (this) назад у відповідний метод відвідувача.

Роль Відвідувача (Visitor): Інтерфейс ISchemaVisitor. Він гарантує, що будь-який алгоритм вмітиме працювати з усіма частинами схеми.

Роль Конкретних Відвідувачів (Concrete Visitors):

- SchemaTextRendererVisitor: Інкапсулює логіку форматування тексту.
- SchemaComparisonVisitor: Інкапсулює логіку порівняння об'єктів.

Обґрунтування використання: Використання шаблону Visitor дозволило дотримуватися принципу Open/Closed:

1. Класи моделі даних залишаються простими та стабільними. Ми не змінюємо їх кожного разу, коли хочемо додати нову функцію.
2. Додавання нової функціональності (наприклад, генерація SQL-скрипту CREATE TABLE на основі схеми) реалізується шляхом створення всього одного нового класу-відвідувача, не зачіпаючи існуючий код.
3. Логіка пов'язаних операцій зібрана в одному місці, а не розпорошена по десятках класів моделі.

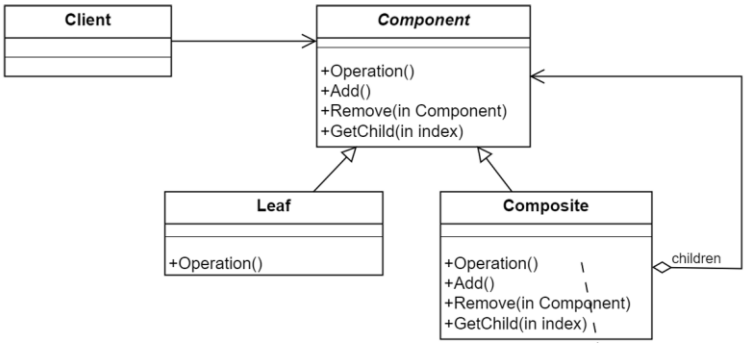
Висновок

У ході виконання лабораторної роботи було проведено детальний аналіз та реалізацію ряду структурних і поведінкових шаблонів проєктування в контексті розробки SQL IDE. Особливу увагу було приділено шаблону «Відвідувач», що надав можливість розширювати функціонал системи новими операціями (аналіз, порівняння) без модифікації класів структури даних. Отримані знання дозволяють проєктувати гнучкі архітектурні рішення, які легко адаптуються до змін вимог та забезпечують високу якість програмного коду.

Контрольні питання:

1. Яке призначення шаблону «Композит» (Composite)? Призначення шаблону «Композит» — дозволити клієнтам працювати з ієрархічними деревоподібними структурами об'єктів (де є прості елементи та контейнери, що містять інші елементи) так, ніби це все — один тип об'єкта. Він дозволяє групувати об'єкти в деревоподібні структури і працювати з цими деревами через спільний інтерфейс, ігноруючи різницю між складними та простими елементами.

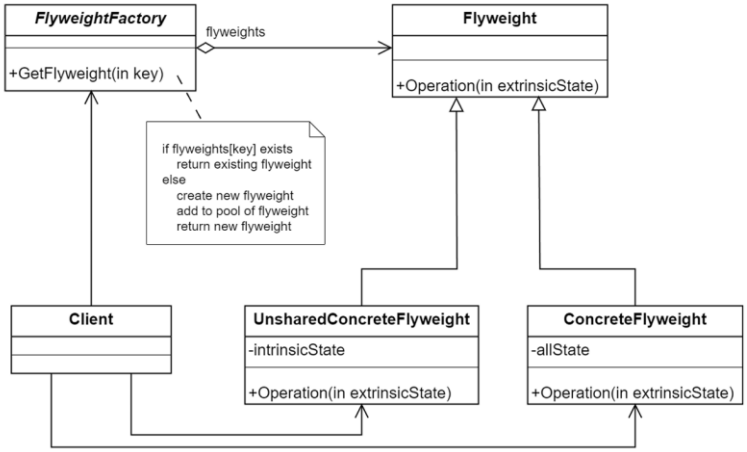
2. Нарисуйте структуру шаблону «Композит».



3. Які класи входять в шаблон «Композит», та яка між ними взаємодія? До шаблону входять: Component (спільний інтерфейс для всіх елементів), Leaf (простий елемент, "листок" дерева, що не має підлеглих) та Composite (складний елемент, "гілка" або контейнер, що містить колекцію дочірніх компонентів). Клієнт взаємодіє з усіма об'єктами через інтерфейс Component. Коли клієнт викликає метод у Composite, той делегує виклик усім своїм дочірнім елементам (Leaf або іншим Composite), а Leaf просто виконує свою роботу.

4. Яке призначення шаблону «Легковаговик» (Flyweight)? Шаблон «Легковаговик» призначений для ефективної підтримки величезної кількості дрібних об'єктів шляхом економії оперативної пам'яті. Він досягає цього через розділення стану об'єкта на внутрішній (спільний, незмінний) та зовнішній (контекстний, який передається об'єкту ззовні під час виклику методу), що дозволяє використовувати один спільний екземпляр замість створення тисяч однакових копій.

5. Нарисуйте структуру шаблону «Легковаговик».



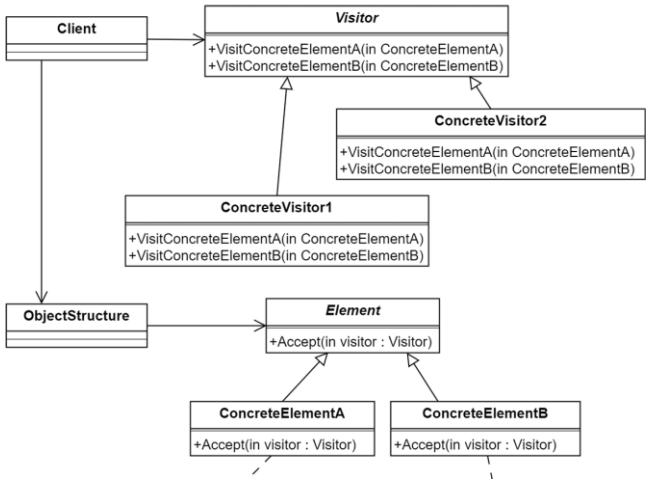
6. Які класи входять в шаблон «Легковаговик», та яка між ними взаємодія? До шаблону входять: Flyweight (інтерфейс або абстрактний клас, що приймає

зовнішній стан як аргумент), ConcreteFlyweight (реалізує інтерфейс і зберігає внутрішній, спільний стан), FlyweightFactory (створює та керує пулом легковаговиків) та Client (обчислює або зберігає зовнішній стан і передає його легковаговику). Клієнт звертається до Фабрики за легковаговиком; якщо такий об'єкт вже існує — повертається він, якщо ні — створюється новий.

7. Яке призначення шаблону «Інтерпретатор» (Interpreter)? Призначення шаблону «Інтерпретатор» — вирішувати часто повторювані задачі, які можна описати за допомогою простої мови або граматики. Він визначає представлення граматики цієї мови та надає інтерпретатор, який використовує це представлення для аналізу та виконання речень мови, будуючи абстрактне синтаксичне дерево (AST), де кожен клас відповідає правилу граматики.

8. Яке призначення шаблону «Відвідувач» (Visitor)? Призначення шаблону «Відвідувач» — відділити алгоритм від структури об'єктів, над якими він оперує. Це дозволяє додавати нові операції (функції) до існуючої ієрархії класів без необхідності змінювати код цих класів, дотримуючись принципу відкритості/закритості.

9. Нарисуйте структуру шаблону «Відвідувач».



10. Які класи входять в шаблон «Відвідувач», та яка між ними взаємодія? До шаблону входять: Visitor (інтерфейс, що оголошує методи Visit для кожного типу елемента), ConcreteVisitor (реалізує конкретну операцію), Element (інтерфейс елемента структури з методом Accept), ConcreteElement (реалізує метод Accept) та ObjectStructure (колекція елементів). Взаємодія відбувається через подвійну диспетчеризацію: клієнт викликає Accept(visitor) у елемента, а елемент всередині

цього методу викликає visitor.Visit(this), передаючи себе, що дозволяє відвідувачу виконати потрібну дію для конкретного типу об'єкта.