МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ «ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Навчально-науковий інститут комп’ютерного моделювання, прикладної фізики та математики

Звіт

З лабораторної роботи №3

З дисципліни:“ Ефективність та якість архітектурних рішень інформаційних систем ”

На тему:

«Патерн проєктування Будівельник»

Виконав:

студент групи ІКМ-М225В

Загорулько Віталій Олегович

Харків 2025

Мета роботи

Метою даної лабораторної роботи є ґрунтовне вивчення та практичне засвоєння породжувального патерна проєктування Будівельник (Builder), а також набуття навичок його застосування для створення гнучких та складних об'єктів (SQL-запитів), відокремлюючи процес їх конструювання від їх кінцевого представлення.

Завдання

У межах виконання лабораторної роботи необхідно виконати такі завдання:

- Ознайомитися з теоретичними засадами та основними ідеями патерна проєктування Будівельник.
- Спроєктувати та реалізувати власний конструктор SQL-запитів (QueryBuilder), що підтримує дві різні СУБД: PostgreSQL та MySQL.
- Забезпечити спільний інтерфейс для обох будівельників запитів.
- Реалізувати методи select, where, limit та getSQL з використанням ланцюгового виклику (Fluent Interface).
- Навести приклад клієнтського коду, що демонструє використання патерна з обома СУБД.

Теоретичні відомості

Будівельник (Builder) — це породжувальний патерн проєктування, який дозволяє створювати складні об'єкти поетапно, відокремлюючи логіку конструювання від його представлення. Цей принцип критично важливий, оскільки він дозволяє використовувати ідентичний алгоритм побудови для створення об'єктів із різними характеристиками та структурою.

Структура та ролі компонентів

Патерн "Будівельник" складається з чотирьох основних компонентів:

- Product (Продукт): Складний об'єкт, який потрібно створити. У нашому випадку — кінцевий SQL-запит (рядок). Продукти одного процесу конструювання можуть мати різну внутрішню структуру.
- Builder (Інтерфейс Будівельника): Визначає уніфікований інтерфейс для конструювання частин продукту. Усі методи конструювання повинні повертати сам об'єкт будівельника (this), що дозволяє використовувати ланцюговий виклик (Fluent Interface).
- Concrete Builder (Конкретний Будівельник): Реалізує інтерфейс Будівельника. Він зберігає внутрішній стан частин продукту та містить специфічну логіку для складання цих частин у кінцевий Продукт.
- Director (Директор): (Опціональний, але рекомендований компонент). Керує процесом конструювання. Його завдання — викликати методи будівельника у певній послідовності для отримання типового або стандартного продукту. Клієнт може використовувати як Директора, так і викликати методи будівельника безпосередньо.

Переваги та коли застосовувати

Патерн "Будівельник" є особливо корисним у таких випадках:

- Складні об'єкти з багатьма опціональними параметрами: Коли конструктор об'єкта вимагає великої кількості параметрів, і більшість із них є необов'язковими. Це дозволяє уникнути проблеми "телескопічного конструктора" (коли доводиться створювати десятки перевантажених конструкторів).

- Різні представлення одного і того ж процесу: Як у нашому випадку, ми маємо один процес (побудова SQL-запиту: SELECT, WHERE, LIMIT), але два різних кінцевих продукти (синтаксис PostgreSQL vs. MySQL).
- Поетапне конструювання: Коли об'єкт має бути створений лише після того, як були виконані всі необхідні кроки, інакше він буде невалідним (наприклад, не можна викликати LIMIT, якщо не було викликано SELECT).

Програмний код реалізації на Java розміщено у Додатку А.

Висновок

Під час виконання лабораторної роботи було успішно вивчено та практично застосовано породжувальний патерн проєктування Будівельник (Builder), що є важливим елементом при проєктуванні архітектурних рішень інформаційних систем. Була спроєктована та реалізована структура класів, яка демонструє патерн "Будівельник" для створення конструкторів SQL-запитів (QueryBuilder), що підтримує дві різні СУБД: PostgreSQL та MySQL. Завдяки реалізації єдиного інтерфейсу QueryBuilder клієнтський код отримав можливість створювати складні запити, використовуючи послідовність простих та зрозумілих методів (select, where, limit) через ланцюговий виклик. Застосування цього патерна дозволило досягти ключової мети — відокремлення процесу конструювання запиту від його конкретного представлення (синтаксису СУБД), оскільки логіка формування специфічного SQL-коду повністю інкапсульована у відповідних конкретних будівельниках (PostgreSQLQueryBuilder та MySQLQueryBuilder). Це робить код гнучким, легко розширюваним для підтримки нових СУБД та незалежним від деталей реалізації, що є ознакою високої якості архітектурного рішення.

Додаток А

Lab3.java