Лабораторна робота: Система управління проектами

Цей проект є комплексною реалізацією реляційної бази даних для корпоративної системи управління проектами. Робота охоплює повний цикл проектування: від візуалізації схеми до імплементації розширеної бізнес-логіки та створення рівня доступу до даних мовою C#.

1. Мета лабораторної роботи

Мета роботи – продемонструвати навички проектування, реалізації та супроводу складних реляційних баз даних.

Основні завдання, вирішені в рамках роботи:

- **Проектування Схеми:** Створення логічної моделі даних, що включає понад 15 сутностей.
- **Реалізація в СУБД:** Розгортання схеми у PostgreSQL.
- **Розширена логіка (Soft Delete & Audit):** Реалізація механізмів "м'якого видалення" (is_deleted) та полів аудиту (created_at, updated_at, updated_by_user_id) для ключових сутностей.
- **Бізнес-логіка на рівні БД:** Активне використання збережених процедур, тригерів, функцій та розрізів даних (View) для інкапсуляції логіки.
- **Оптимізація:** Створення різних типів індексів (B-Tree, Partial, GIN) для прискорення запитів.
- **Рівень доступу до даних:** Реалізація патернів **Repository** та **Unit of Work** на C# для безпечної та транзакційної взаємодії з БД.

2. Структура файлів проекту

1. **schema.dbml**:
   - **Опис:** Файл візуалізації схеми для інструменту dbdiagram.io. Дозволяє швидко зрозуміти зв'язки між 17 таблицями.
   - **Використання:** Скопіюйте вміст файлу та вставте його у редактор dbdiagram.io для візуалізації.
2. **database_schema.sql**:
   - **Опис:** Основний SQL-скрипт для створення всіх 17 таблиць, їхніх полів, зв'язків (FOREIGN KEY) та обмежень (CHECK).
   - **Увага:** Цей скрипт має виконуватись **першим**.
3. **database_logic.sql**:
   - **Опис:** SQL-скрипт, що створює всю бізнес-логіку поверх таблиць.
   - **Включає:**

        - **Тригери (5):** Автоматично оновлюють поле updated_at при зміні даних.
        - **Розрізи (Views) (3):** v_active_users, v_active_projects та v_project_task_details. Вони автоматично фільтрують "м'яко видалені" записи (is_deleted = FALSE). **Увесь код має читати дані саме з них, а не з таблиць напряму.**
        - **Збережені процедури (4):** Інкапсулюють CUD-операції (Create, Update, Delete). Наприклад, sp_create_project та sp_archive_project_and_tasks (яка виконує транзакційне видалення проекту та всіх його завдань).
        - **Функції (1):** fn_get_user_active_task_count (приклад скалярної функції).
        - **Індекси (4 типи):** B-Tree, Unique B-Tree, Partial (для is_deleted = FALSE) та GIN (для повнотекстового пошуку).
    - **Увага:** Цей скрипт має виконуватись **другим**, після database_schema.sql.
4. **UnitOfWork.cs**:
    - **Опис:** Приклад реалізації патернів Repository та Unit of Work на C# з використанням Dapper.
    - **Ключова вимога (п. 7):** Цей код суворо дотримується правила:
        - **Читання (Read):** Відбувається **лише** через **Розрізи (Views)** (наприклад, GetAllActiveProjectsAsync робить SELECT * FROM v_active_projects).
        - **Запис (Write/Update/Delete):** Відбувається **лише** через **Збережені процедури (SP)** (наприклад, CreateProjectAsync викликає sp_create_project).
    - **ProjectService** демонструє, як UnitOfWork керує транзакціями, гарантуючи, що або всі операції (наприклад, архівування проекту та всіх його завдань) виконуються успішно, або всі відкочуються.

3. Інструкція з розгортання

Для запуску та тестування системи виконайте наступні кроки:

Вимоги:

- Встановлена СУБД PostgreSQL (та інструмент для керування, наприклад, pgAdmin або DBeaver).
- (Опціонально) .NET SDK та C# IDE (Visual Studio / VS Code) для тестування коду.

Крок 1. Створення Бази Даних

1. Підключіться до вашого сервера PostgreSQL.
2. Створіть нову базу даних (наприклад, project_management_db).
3. Переконайтеся, що ви підключені до цієї нової бази даних.
4. Встановіть розширення uuid-ossp, виконавши:
   CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

Крок 2. Створення Схеми (Таблиць)

1. Відкрийте файл database_schema.sql.
2. Скопіюйте весь його вміст.

3. Виконайте скопійований SQL-скрипт у вашій project_management_db.

Результат: Усі 17 таблиць будуть створені.

Крок 3. Додавання Бізнес-логіки

1. Відкрийте файл database_logic.sql.
2. Скопіюйте весь його вміст.
3. Виконайте скопійований SQL-скрипт у вашій project_management_db.

Результат: Усі тригери, процедури, функції, індекси та розрізи будуть створені.

Крок 4. Тестування (C#)

1. Створіть новий консольний або веб-проект C#.
2. Додайте необхідні NuGet-пакети:
   dotnet add package Dapper
   dotnet add package Npgsql

3. Додайте файл UnitOfWork.cs до вашого проекту.
4. У вашому головному файлі (наприклад, Program.cs) замініть рядок _connectionString на ваш реальний рядок підключення до project_management_db.
5. Ви можете протестувати створення та архівацію проекту, як показано у класі ProjectService.