

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

# Кафедра системного програмування та спеціалізованих комп’ютерних систем

**Лабораторна робота №3**

з дисципліни

**«Бази даних і засоби управління»**

**На тему** «Засоби оптимізації роботи СУБД PostgreSQL»

Виконав: студент ІII курсу

ФПМ групи КВ-82

Чернухін Владислав

Перевірив: Павловський В. І.

Київ – 2020

*Метою роботи* є здобуття практичних навичок використання засобів оптимізації СУБД PostgreSQL.

*Завдання* роботи полягає у наступному:

1. Перетворити модуль “Модель” з шаблону MVC лабораторної роботи №2 у вигляд об’єктно-реляційної проекції (ORM).
2. Створити та проаналізувати різні типи індексів у PostgreSQL.
3. Розробити тригер бази даних PostgreSQL.

***Вимоги до пункту завдання №1***

Для перетворення функцій, що реалізують запити до об’єктної бази даних, необхідно встановити бібліотеку sqlAlchemy, налаштувати програму на роботу з ORM, розробити класи-сутності для об’єктів-сутностей, представлених відповідними таблицями БД та пов’язаних зв’язками 1:М, М:М та 1:1 виконати опис схеми бази даних. Особливу увагу приділити контролю зовнішніх зв’язків між таблицями засобами ORM.

Замінити виклики запитів мовою SQL на відповідні запити засобами SQLAlchemy по роботі з об’єктами. Обов’язковим є реалізація вставки, вилучення та редагування екземплярів класів-сутностей. Розробка запитів на генерацію даних та пошук екземплярів класів-сутностей вітається, але не є обов’язковою.

Інтерфейси функцій (вхідні та вихідні аргументи функцій модуля “Модель”) мають залишитись без змін.

Корисні посилання: [тут](https://www.learndatasci.com/tutorials/using-databases-python-postgres-sqlalchemy-and-alembic/) і [тут](https://auth0.com/blog/sqlalchemy-orm-tutorial-for-python-developers/).

***Вимоги до пункту завдання №2***

Відповідно до варіанту індексування продемонструвати на прикладах запитів SQL SELECT підвищення швидкодії їх виконання з використанням індексів, а також пояснити чому для деяких випадків індексування використовувати недоцільно. При цьому для наочного представлення слід використати функцію генерування рандомізованих даних з лабораторної роботи №2, створивши необхідну кількість тестових даних. Навести 4-5 прикладів запитів SELECT (із виведенням результуючих даних), що містять фільтрацію, агрегатні функції, групування та сортування (у необхідних комбінаціях).

Корисні посилання: [Hash](https://habrahabr.ru/company/postgrespro/blog/328280/), [B-tree](https://habrahabr.ru/company/postgrespro/blog/330544/), [GIN](https://habrahabr.ru/company/postgrespro/blog/340978/), [BRIN](https://habrahabr.ru/company/postgrespro/blog/346460/).

***Вимоги до пункту завдання №3***

Створити тригер бази даних PostgreSQL відповідно до варіанта. Тригерна функція має включати обробку запису, що модифікується (вставляється або вилучається), умовні оператори, курсорні цикли та обробку виключних ситуацій. Виконати відлагодження тригера при різних вхідних даних, навівши 2-3 приклади його використання.

Корисні посилання: [тут](https://www.enterprisedb.com/postgres-tutorials/everything-you-need-know-about-postgresql-triggers), [тут](https://www.postgresqltutorial.com/postgresql-triggers/).

***Вимоги до інструментарію***

1. Бібліотека для реалізації ORM - [SQLAlchemy для Python](https://www.sqlalchemy.org/) або інша з подібною функціональністю.
2. Середовище для відлагодження SQL-запитів до бази даних – pgAdmin 4.
3. СУБД - PostgreSQL 11-12.



# Завдання 1

**Логічна модель (Структура) БД “** Ігрова платформа **”** (засобами SqlDMB)

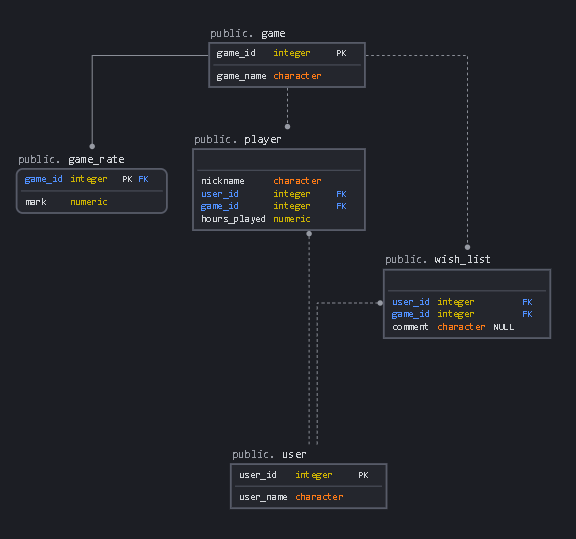


Рисунок 1 - Логічна модель (Структура) БД “ Ігрова платформа ”

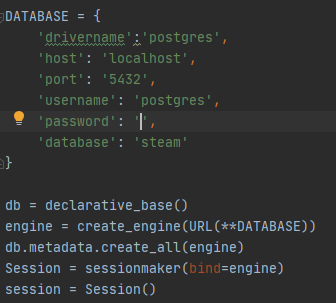
**Нотація:** Модель побудована засобами програми SqlDBM.

Для вирішення задачі перетворення моделі з MVC у вигляд об’єктно-реляційної проекції (ORM) використовується бібліотека SQLAlchemy для мови програмування Python, яка є однією з найпопулярніших для вирішення задач об’єктно-реляційного відображення.

Представлення таблиць у класах полягає у представленні полів класів як колон таблиці. Для цього потрібно врахувати відповідність типів даних між сервером PostgreSQL і відповідною мовою програмування.

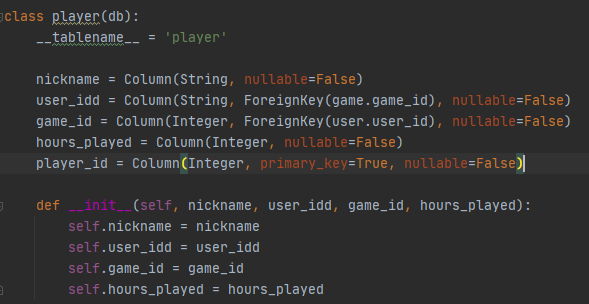
Для реалізації зовнішнього ключа у класі, який відповідає таблиці із зовнішнім ключем з відношенням до колонки таблиці А, потрібно додати поле типу класу, який відповідає таблиці А. Також потрібно написати відповідну команду для визначення цього поля як зовнішній ключ. Програма створена за допомогою мови програмування Python в середовищі розробки PyCharm Community Edition 2020.2.3

Підключення SQLAlchemy до БД:

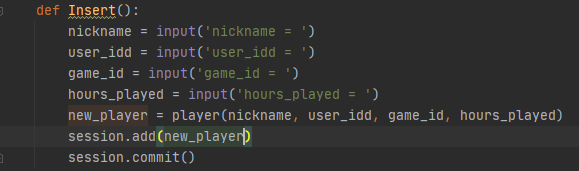


Для прикладу реалізації перетворення моделі з MVC у вигляд об’єктно-реляційної проекції (ORM) розглянемо код для таблиці «player».

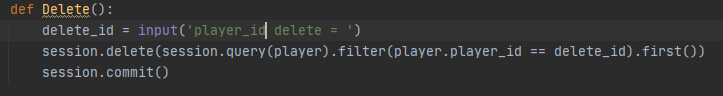
Клас для співвідношення даних класу з таблицею в БД:



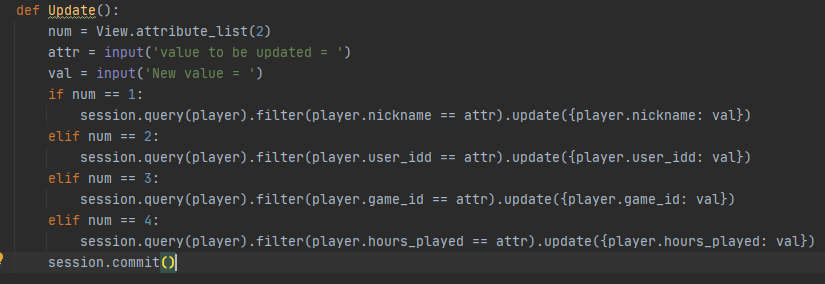
Метод додавання даних до таблиці «player»:



Метод видалення даних з таблиці «player»:



Метод редагування даних таблиці «player»:



На рисунку 2 зображена структура класів програми, що відповідають таблицям БД

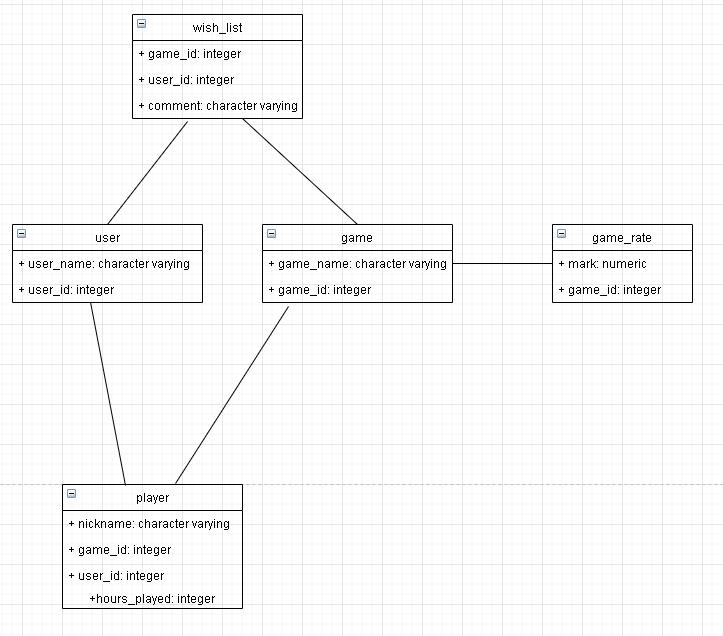


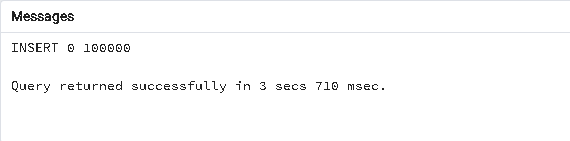
Рисунок 2 – Структура класів програми, що відповідають таблицям БД

# Завдання 2

# Створення та аналіз індексу BTree

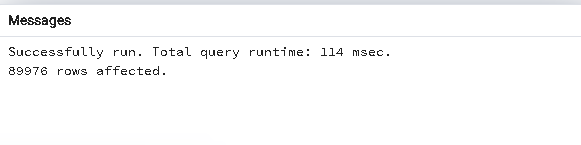
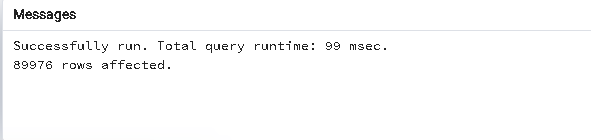
Для дослідження даного індексу створимо таблицю test\_table з одним числовим полем int типу integer та заповнимо її 100000 випадковими значеннями.

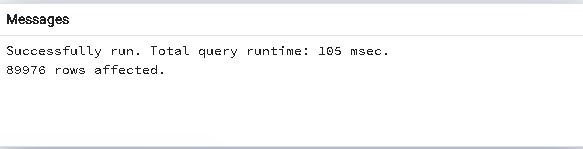




Виконаємо пошук даних для rand >10000 без використання індексу:

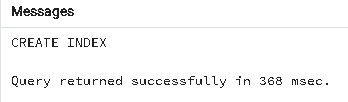




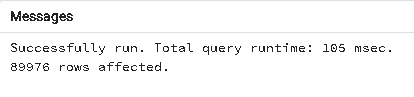
Середній час виконання запиту: 106 msec.

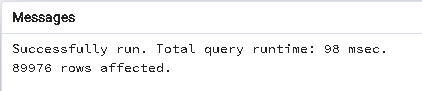
Створимо індекс BTree

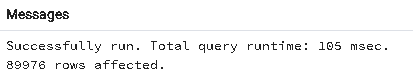
 

Виконаємо пошук даних для rand >10000









Середній час виконання запиту: 103 msec.

Як бачимо, середній час виконання запиту з використанням індексу змінився несуттєво.Це дуже просто пояснюється, PostgreSQL за замовчуванням використовує індекс BTree. Тобто, по суті в обох випадках використовувався індекс BTree.

# Створення та аналіз індексу BRIN

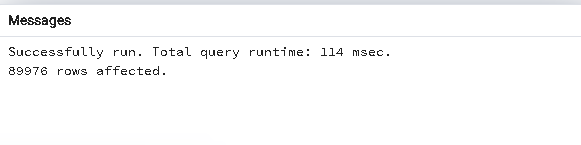
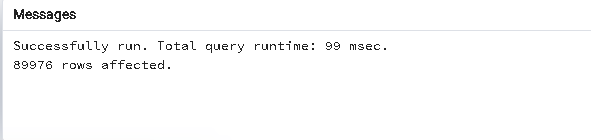
Для дослідження даного індексу будемо використовувати ту ж таблицю test\_table.

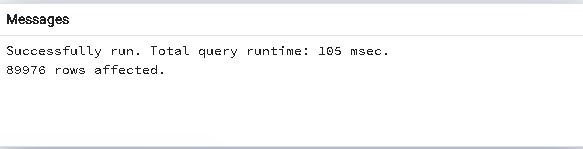


# 

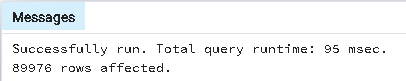
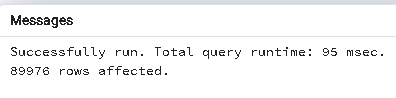
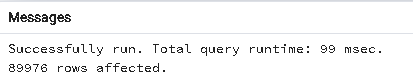
Виконаємо пошук даних для rand >10000 без використання індексу:





Середній час виконання запиту: 106 msec.

Середній час виконання запиту: 96 msec.

Як бачимо, середній час виконання запиту з використанням індексу BRIN став менший на 10 msec. Це означає що даний індекс доцільно використовувати при пошуку даних у великих таблицях.

# Завдання 3

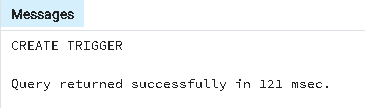
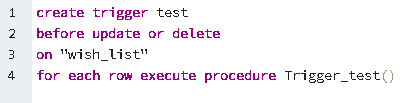
Логіка тригера: Тригер працює перед оновленням або видаленням з таблиці «wish\_list». Якщо була спроба оновити дані, то до нової таблиці «trigger\_test», що має лише один атрибут operation, заноситься «UPDATED». При цьому дані з таблиці «wish\_list» оновлені не будуть.

Якщо була спроба видалити дані з атрибутом game\_id то до таблиці “trigger\_test” буде занесено “DELETED game with id = 1 from wishlist“, при чому дані видалені не будуть

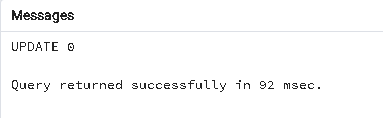
Створення тригеру:

# 

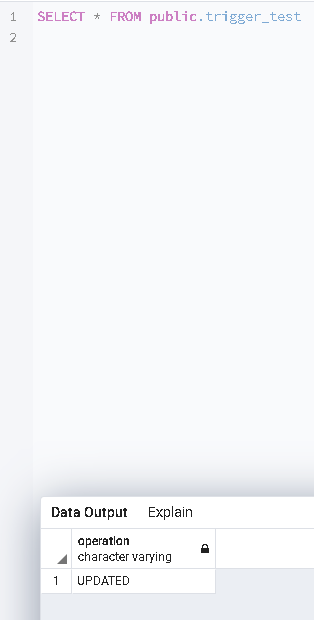
Підключення тригеру до таблиці «wish\_list»:



Спроба оновити дані таблиці «wish\_list», де поле game\_id=1:

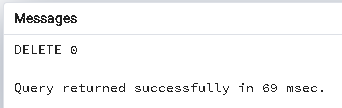


Вміст таблиці «trigger\_test»:

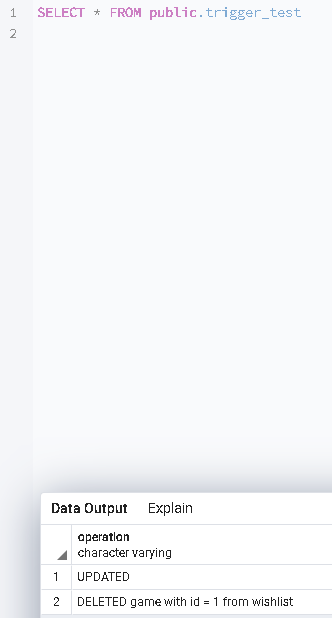


Спроба видалення даних з таблиці «wish\_list»:





Вміст таблиці «trigger\_test»:



Вміст таблиці «trigger\_test» є коректним після проведення дослідження. Отже, тригер працює відповідно до заданої логіки.