

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** 3

з дисципліни “Бази даних”

# тема “Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL”

|  |  |
| --- | --- |
| Виконав  Студент II курсу  групи КП-03  Павлюшин Максим Юрійович  варіант №12 |  |

**Київ 2021**

**Мета**

Здобуття практичних навичок використання засобів оптимізації СУБД PostgreSQL.

**Постановка завдання**

1. Перетворити модуль “Модель” з шаблону MVC лабораторної роботи No2

у вигляд об’єктно-реляційної проекції (ORM).

2. Створити та проаналізувати різні типи індексів у PostgreSQL.

3. Розробити тригер бази даних PostgreSQL.

**Вимоги до інструментарію**

1. Бібліотека для реалізації ORM - SQLAlchemy для Python або інша з

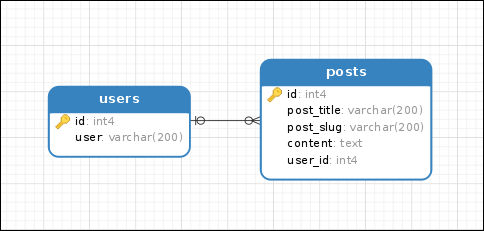
подібною функціональністю.

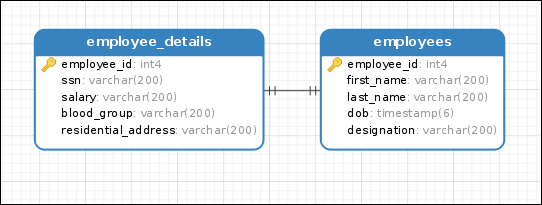
2. Середовище для відлагодження SQL-запитів до бази даних – pgAdmin 4.

3. СУБД - PostgreSQL 11-12.

**Результати роботи**

1.





|  |
| --- |
| Program.py |
| from sqlalchemy import MetaData, Table, String, Integer, Column, Text, DateTime, Boolean  from datetime import datetime  metadata = MetaData()  blog = Table('blog', metadata,  Column('id', Integer(), primary\_key=True),  Column('post\_title', String(200), nullable=False),  Column('post\_slug', String(200), nullable=False),  Column('content', Text(), nullable=False),  Column('published', Boolean(), default=False),  Column('created\_on', DateTime(), default=datetime.now),  Column('updated\_on', DateTime(), default=datetime.now, onupdate=datetime.now)  from sqlalchemy import MetaData, Table, Column, Integer, ARRAY  metadata = MetaData()  employee = Table('employees', metadata,  Column('id', Integer(), primary\_key=True),  Column('workday', ARRAY(Integer)),  )  from sqlalchemy import MetaData, Table, Column, Integer  from sqlalchemy.dialects import postgresql  metadata = MetaData()  comments = Table('comments', metadata,  Column('id', Integer(), primary\_key=True),  Column('ipaddress', postgresql.INET),  )  from sqlalchemy import MetaData, Table, Column, Integer, String, Text, ForeignKey  metadata = MetaData()  user = Table('users', metadata,  Column('id', Integer(), primary\_key=True),  Column('user', String(200), nullable=False),  )  posts = Table('posts', metadata,  Column('id', Integer(), primary\_key=True),  Column('post\_title', String(200), nullable=False),  Column('post\_slug', String(200), nullable=False),  Column('content', Text(), nullable=False),  Column('user\_id', ForeignKey("users.id")),  )  from sqlalchemy import MetaData, Table, Column, Integer, String, DateTime, ForeignKey  metadata = MetaData()  employees = Table('employees', metadata,  Column('employee\_id', Integer(), primary\_key=True),  Column('first\_name', String(200), nullable=False),  Column('last\_name', String(200), nullable=False),  Column('dob', DateTime(), nullable=False),  Column('designation', String(200), nullable=False),  )  employee\_details = Table('employee\_details', metadata,  Column('employee\_id', ForeignKey('employees.employee\_id'), primary\_key=True),  Column('ssn', String(200), nullable=False),  Column('salary', String(200), nullable=False),  Column('blood\_group', String(200), nullable=False),  Column('residential\_address', String(200), nullable=False),  ) |

Таблиці в SQLAlchemy представлені у вигляді екземплярів класу Table. Його конструктор приймає назву таблиці, метадані та одну або кілька колонок.

Тип визначає, які дані колонка зможе зберігати. SQLAlchemy надає абстракцію для великої кількості типів. Проте є три категорії:

Загальні типи

Стандартні SQL-типи

Типи окремих розробників

Дві таблиці пов'язані ставленням одним, якщо запис у першій таблиці пов'язана з одним або декількома записами другий.

Дві таблиці мають зв'язок один до одного, якщо запис в одній таблиці пов'язана тільки з одним записом в інший.

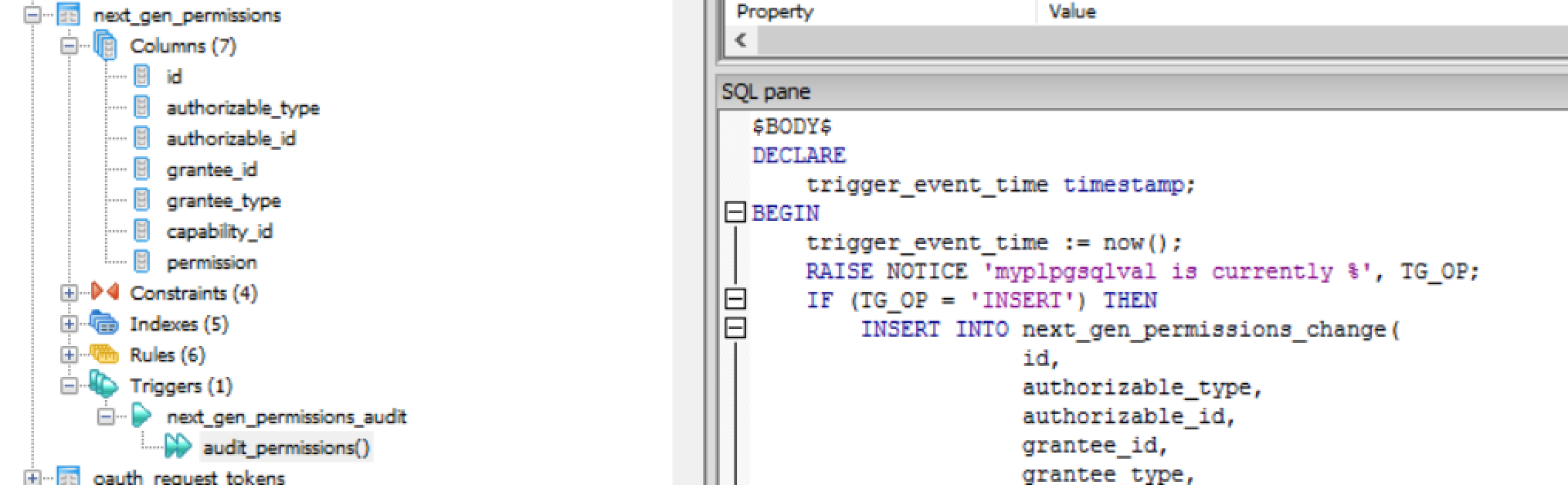
2.

PostgreSQL підтримує кілька типів індексів: B-дерево, хеш, GiST, SP-GiST, GIN та BRIN. Для різних типів індексів застосовуються різні алгоритми, орієнтовані певні типи запитів. За замовчуванням команда CREATE INDEX створює індекси типу B-дерево, найефективніші у більшості випадків.

Використовуючи індексування свого варіанту (BTree, Gin) підвищив швидкодії використання запитів SQL SELECT.

3.

|  |
| --- |
| Python.py |
| DROP TRIGGER [IF EXISTS] trigger\_name  ON table\_name [ CASCADE | RESTRICT ];  CREATE FUNCTION check\_staff\_user()  RETURNS TRIGGER  AS $$  BEGIN  IF length(NEW.username) < 8 OR NEW.username IS NULL THEN  RAISE EXCEPTION 'The username cannot be less than 8 characters';  END IF;  IF NEW.NAME IS NULL THEN  RAISE EXCEPTION 'Username cannot be NULL';  END IF;  RETURN NEW;  END;  $$  LANGUAGE plpgsql;  CREATE TRIGGER username\_check  BEFORE INSERT OR UPDATE  ON staff  FOR EACH ROW  EXECUTE PROCEDURE check\_staff\_user(); |



Виконуючи третє завдання створив тригер бази даних PostgreSQL відповідно до варіанта (after, update, insert). Тригерна функція включає обробку запису, що модифікується (вставляється або вилучається), умовні оператори, курсорні цикли та обробку виключних ситуацій.

**Висновки**

Під час виконання лабораторної я здобув практичні навички використання засобів оптимізації СУБД PostgreSQL.

Перетворив модуль «Модель» з шаблону MVC ЛР№2, створив та проаналізував різні типи індексів у PostgreSQL та розробив тригер бази даних PostgreSQL.