

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

групи КВ-02

Лунгов О.В.

Перевірив: Павловський В. І.

Київ – 2022

**Варіант 12**

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.
4. Навести приклади та проаналізувати рівні ізоляції транзакцій у PostgreSQL.

**Логічна модель бази даних**

Нижче (Рисунок 1) наведено логічну модель бази даних:

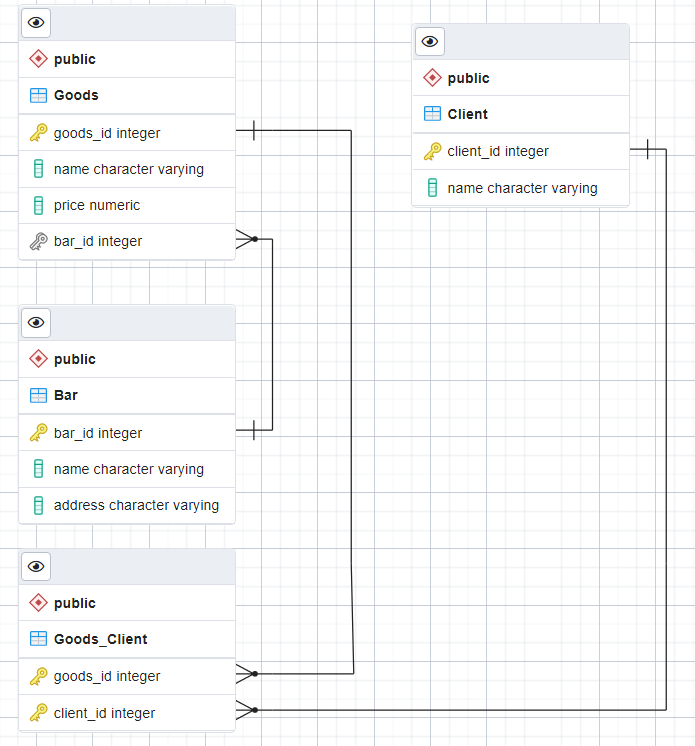


Рисунок 1 – Логічна модель бази даних

Зміни у порівнянні з першою лабораторною роботою відсутні

**Класи ORM**

import sqlalchemy  
from sqlalchemy import Column, ForeignKey, Integer, String  
from sqlalchemy.orm import relationship  
from sqlalchemy.ext.declarative import declarative\_base  
  
Base = declarative\_base()  
  
  
class Bar(Base):  
 \_\_tablename\_\_ = 'Bar'  
 bar\_id = Column(Integer, primary\_key=True)  
 name = Column(String)  
 address = Column(String)  
  
 def \_\_init\_\_(self, name, address):  
 self.name = name  
 self.address = address  
 super(Bar, self).\_\_init\_\_()  
  
  
class Goods(Base):  
 \_\_tablename\_\_ = 'Goods'  
 goods\_id = Column(Integer, primary\_key=True)  
 name = Column(String)  
 price = Column(Integer)  
 bar\_id = Column(Integer, ForeignKey('Bar.bar\_id'))  
  
 plant = relationship('Bar', foreign\_keys=[bar\_id])  
  
 def \_\_init\_\_(self, name, price, bar\_id):  
 self.name = name  
 self.price = price  
 self.bar\_id = bar\_id  
 super(Goods, self).\_\_init\_\_()  
  
  
class Client(Base):  
 \_\_tablename\_\_ = 'Client'  
 client\_id = Column(Integer, primary\_key=True)  
 name = Column(String)  
  
 def \_\_init\_\_(self, name):  
 self.name = name  
 super(Client, self).\_\_init\_\_()  
  
  
class GoodsClient(Base):  
 \_\_tablename\_\_ = 'Goods\_Client'  
 goods\_id = Column(Integer, ForeignKey('Goods.goods\_id'), primary\_key=True)  
 client\_id = Column(Integer, ForeignKey('Client.client\_id'), primary\_key=True)  
  
 goods = relationship("Goods", foreign\_keys=[goods\_id])  
 client = relationship("Client", foreign\_keys=[client\_id])  
  
 def \_\_init\_\_(self, goods\_id, client\_id):  
 self.goods\_id = goods\_id  
 self.client\_id = client\_id

BTree

Для дослідження індексу була створена таблиця, яка має дві колонки:

числову і текстову. Вони проіндексовані як BTree. У таблицю було занесено1000000 записів.

Створення таблиці та її заповнення:

**CREATE** **TABLE** "test\_btree"(

"id" bigserial **PRIMARY** **KEY**,

"test\_text" **varchar**(255));

**INSERT** **INTO** "test\_btree"("test\_text")

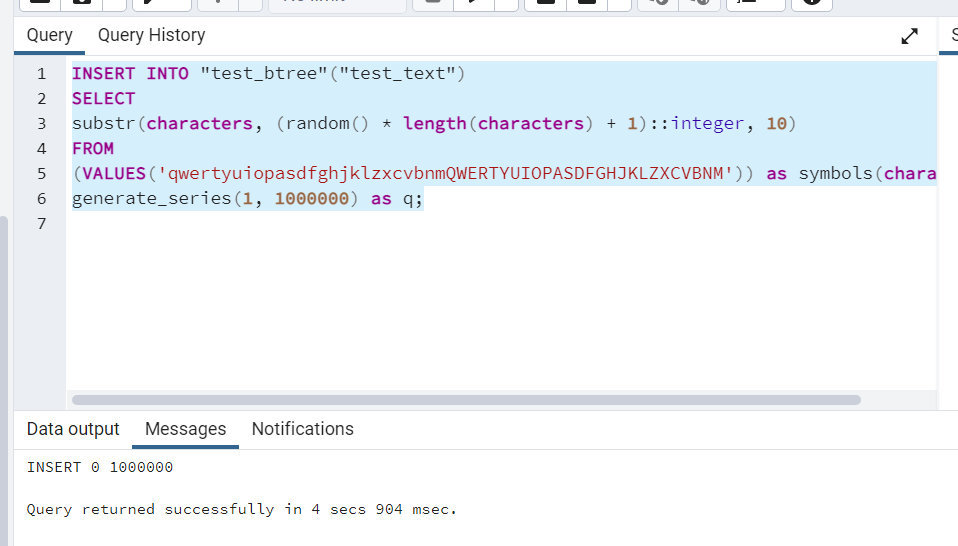
**SELECT**

**substr**(characters, (**random**() \* **length**(characters) + 1)::integer, 10)

**FROM**

(**VALUES**('qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM')) **as** symbols(characters),

generate\_series(1, 1000000) **as** q;

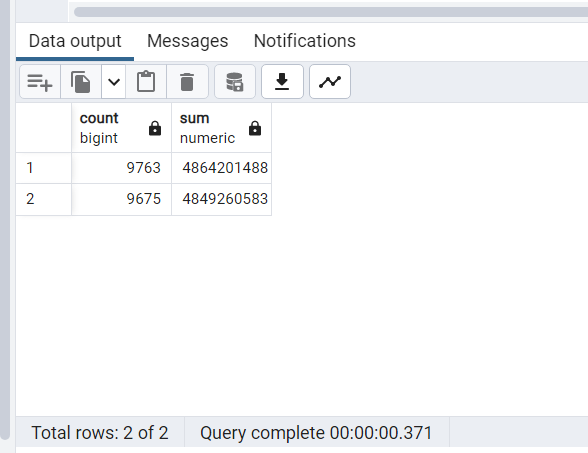


Вибір даних без індексу:

**SELECT** **COUNT**(\*) **FROM** "test\_btree" **WHERE** "id" % 2 = 0;

**SELECT** **COUNT**(\*) **FROM** "test\_btree" **WHERE** "id" % 2 = 0 **OR** "test\_text" **LIKE** 'b%';

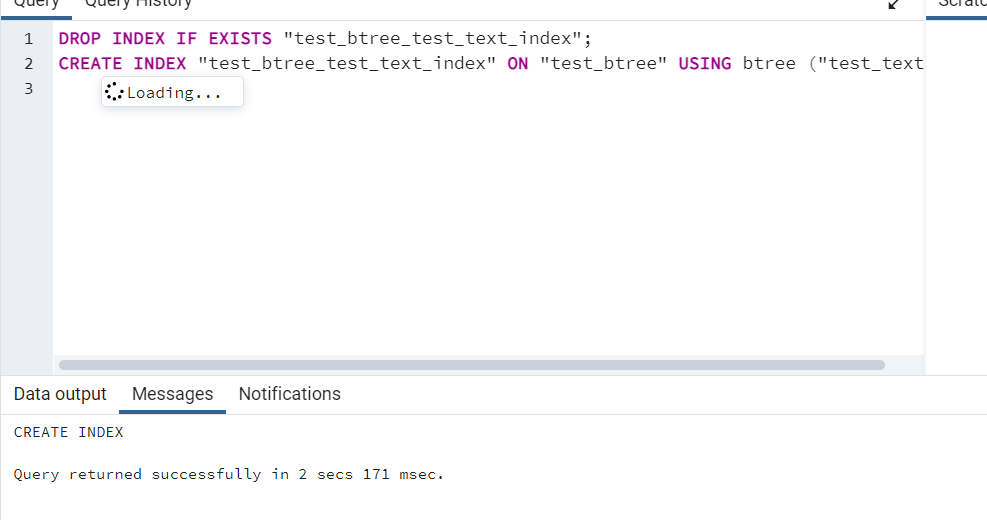
**SELECT** **COUNT**(\*), **SUM**("id") **FROM** "test\_btree" **WHERE** "test\_text" **LIKE** 'b%' **GROUP** **BY** "id" % 2;



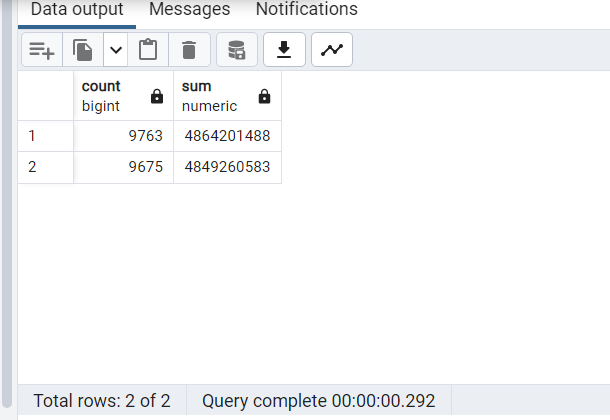
Сворюємо індекс:

**DROP** **INDEX** **IF** **EXISTS** "test\_btree\_test\_text\_index";

**CREATE** **INDEX** "test\_btree\_test\_text\_index" **ON** "test\_btree" **USING** btree ("test\_text");



Вибір даних з створеним індексом:



GIN

Для дослідження індексу була створена таблиця, яка має дві колонки: числову і текстову. Вони проіндексовані як GIN. У таблицю було занесено 1000000 записів.

Створення таблиці та її заповнення:

**DROP** **TABLE** **IF** **EXISTS** "test\_gin";

**CREATE** **TABLE** "test\_gin"(

"id" bigserial **PRIMARY** **KEY**,

"test\_text" **varchar**(255)

);

**INSERT** **INTO** "test\_gin"("test\_text")

**SELECT**

**substr**(characters, (**random**() \* **length**(characters) + 1)::integer, 10)

**FROM**

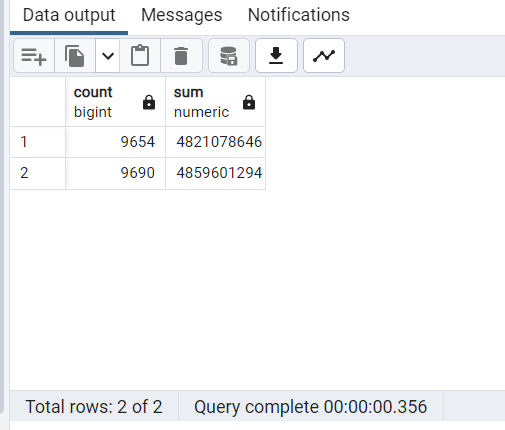
(**VALUES**('qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM')) **as** symbols(characters), generate\_series(1, 1000000) **as** q;

Вибір даних без індексу:

**SELECT** **COUNT**(\*) **FROM** "test\_gin" **WHERE** "id" % 2 = 0;

**SELECT** **COUNT**(\*) **FROM** "test\_gin" **WHERE** "id" % 2 = 0 **OR** "test\_text" **LIKE** 'b%';

**SELECT** **COUNT**(\*), **SUM**("id") **FROM** "test\_gin" **WHERE** "test\_text" **LIKE** 'b%' **GROUP** **BY** "id" % 2;



Сворюємо індекс:

**DROP** **INDEX** **IF** **EXISTS** "test\_gin\_test\_text\_index";

**CREATE** **INDEX** "test\_gin\_test\_text\_index" **ON** "test\_gin" **USING** gin ("test\_text");

Вибір даних з створеним індексом:

