

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»**

**Тема:** «Засоби оптимізації роботи СУБД PostgreSQL»

Виконав:  
студент групи КВ-84  
Тупало К.С  
Перевірив:  
Петрашенко А.В.

Київ 2020

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

## **Завдання**

*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:M, M:M 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.

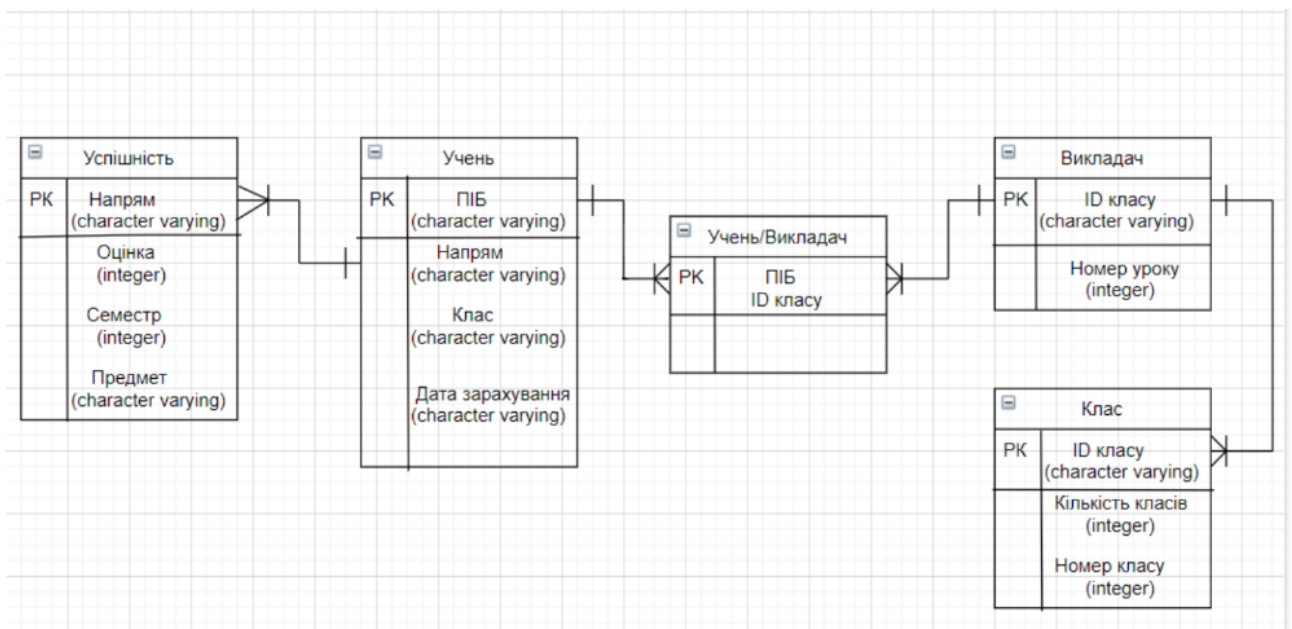
### *Вимоги до пункту завдання №2*

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).

### 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.

27	GIN, BRIN	after update, insert
----	-----------	----------------------



## Завдання 1

```
import psycopg2
import sqlalchemy
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy import Column,String,Integer,ForeignKey
from sqlalchemy.orm import relationship ,sessionmaker

DATABASE = 'postgres+psycopg2://postgres:@localhost:2341/MyData'
engine = create_engine(DATABASE)
Session = sessionmaker(bind=engine)

Base = declarative_base()
```

### Класи сутності

```
class Student(Base):
    __tablename__ = 'student'
    student_data = Column(String,primary_key = True)
    direction=Column(String,ForeignKey('success.direction'))
    m_class = Column(String)
    receipt_date = Column(String)
    teachers = relationship("Teacher",secondary = student_teacher)
    def __init__(self,student_data,direction,m_class,receipt_date):
        self.student_data = student_data
        self.direction = direction
        self.m_class = m_class
        self.receipt_date = receipt_date
```

```
class Teacher(Base):
    __tablename__ = 'teacher'
    class_id = Column(String,primary_key = True)
    lesson_count = Count(Integer)
    def __init__(self,class_id,lesson_count):
        self.class_id=class_id
        self.lesson_count = lesson_count
```

```
class Success(Base):
    __tablename__ = 'success'
    direction = Column(String)
    score = Column(Integer)
    semester = Column(Integer)
    subject = Column(String)
    def __init__(self,direction,score,semester,subject):
        self.direction=direction
        self.score=score
        self.semester=semester
        self.subject=subject
```

```
class School_class(Base):
    __tablename__ = 'school_class'
    class_id = Column(String,primary_key = True)
    class_count = Column(Integer)
    class_number =Column(Integer)
    def __init__(self,class_id,class_count,class_number):
        self.class_id = class_id
        self.class_count = class_count
        self.class_number=class_number
```

## Функції Insert,Delete,Update

```
def insert(self,table,values):
    try:
        keys = [x.lstrip("!") if x.startswith("!") else "{}".format(x) for x in values]
        return table.insert(self.session,values = keys)
    except Exception as err:
        print(err)
```

```
def update(self,table,relay,values):
    try:
        value = [x.lstrip("!") if x.startswith("!") else "{}".format(x) for x in values]
        return table.update(self.sesion).where(relay).values(value)
    except Exception as err:
        print(err)
```

```
def delete(self,table,key):
    try:
        return table.delete(self.session).where(key)
    except Exception as err:
        print(err)
```

## Завдання 2

Виконаємо пошук за значенням

Без індексу

1	<code>explain analyze select * from test_index where "tst_gin" = 'effekerdcw'</code>
<div><div>Data Output</div><div>Explain</div><div>Messages</div><div>Notifications</div></div>	
	<div><div>QUERY PLAN</div><div>text</div><div>1 Seq Scan on test_index (cost=0.00..25.00 rows=6 width=40) (actual time=0.017..0.017 rows=1 ...</div><div>2 Filter: ((tst_gin)::text = 'effekerdcw'::text)</div><div>3 Rows Removed by Filter: 10</div><div>4 Planning Time: 0.201 ms</div><div>5 Execution Time: 0.032 ms</div></div>

GIN

<div><div>Query Editor</div><div>Query History</div></div>	
1	<code>create index indx_gin on test_index using brin("tst_gin");</code>
2	<code>explain analyze select * from test_index where "tst_gin" = 'effekerdcw'</code>
<div><div>Data Output</div><div>Explain</div><div>Messages</div><div>Notifications</div></div>	
	<div><div>QUERY PLAN</div><div>text</div><div>1 Seq Scan on test_index (cost=0.00..1.14 rows=1 width=40) (actual time=0.012..0.013 rows=1 loo...</div><div>2 Filter: ((tst_gin)::text = 'effekerdcw'::text)</div><div>3 Rows Removed by Filter: 10</div><div>4 Planning Time: 0.329 ms</div><div>5 Execution Time: 0.022 ms</div></div>

Індекс GIN покращив пошук, його основною задачею є прискорення повнотекстового пошуку. GIN добре підходить для даних, які не часто оновлюються. Для таблиць де зберігаються часто змінні дані не рекомендовано використовувати цей індекс, бо переіндексація може займати багато часу.

## BRIN

### Без індексу

Query Editor		Query History		
1	<code>explain analyze select * from test_index where "tst_brin" = '20:28:31.271692'</code>			
Data Output		Explain	Messages	Notifications
	QUERY PLAN			
	text			
1	Seq Scan on test_index (cost=0.00..1.14 rows=1 width=40) (actual time=0.024..0.024 rows=1 loo...			
2	Filter: (tst_brin = '20:28:31.271692'::time without time zone)			
3	Rows Removed by Filter: 10			
4	Planning Time: 293.809 ms			
5	Execution Time: 0.039 ms			

### З індексом

Query Editor

Query History

1

create index brin\_index on test\_index using brin("tst\_brin")

2

explain analyze select \* from test\_index where "tst\_brin" = '20:28:31.271692'

Data Output

Explain

Messages

Notifications

▲

QUERY PLAN

text

🔒

1

Seq Scan on test\_index (cost=0.00..1.14 rows=1 width=40) (actual time=0.013..0.013 rows=1 loo...

2

Filter: (tst\_brin = '20:28:31.271692'::time without time zone)

3

Rows Removed by Filter: 10

4

Planning Time: 0.084 ms

5

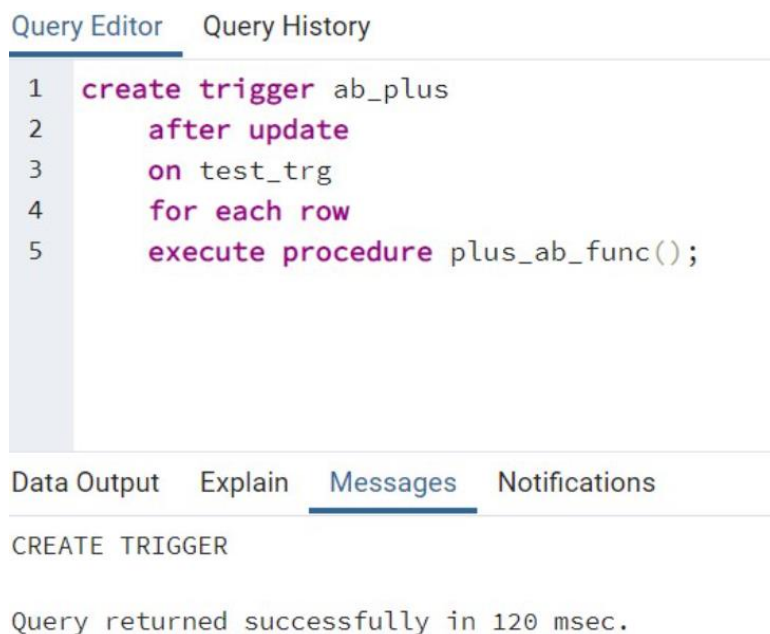
Execution Time: 0.026 ms

Спостерігаємо те, що застосування індексу BRIN пришвидшило пошук заданого значення.

BRIN відмінно працює для стовпців, значення яких корелюють з їх фізичним розташуванням в таблиці.

BRIN в основному призначений для таблиць великих і навіть величезних розмірів, які або взагалі не оновлюються, або оновлюються дуже незначно.

### Завдання 3



The screenshot shows a database query editor interface. At the top, there are two tabs: 'Query Editor' (selected) and 'Query History'. The 'Query Editor' tab contains a SQL script with five lines of code, numbered 1 to 5 on the left margin. The code is: 1 create trigger ab\_plus, 2 after update, 3 on test\_trg, 4 for each row, 5 execute procedure plus\_ab\_func();. Below the code editor, there are four tabs: 'Data Output', 'Explain', 'Messages' (selected), and 'Notifications'. The 'Messages' tab displays the output of the query: 'CREATE TRIGGER' followed by a blank line, and then 'Query returned successfully in 120 msec.'

```
1 create trigger ab_plus
2   after update
3   on test_trg
4   for each row
5   execute procedure plus_ab_func();
```

CREATE TRIGGER

Query returned successfully in 120 msec.

### *Створення тригерної функції*



plus\_ab\_func()

×

General

Definition

Code

Options

Parameters

Security

SQL

```

1 declare
2 ab cursor for select m_a,m_b from test_trg
3 where m_group = old.m_group;
4 var_a numeric;
5 var_b numeric;
6 ab_plus_var numeric;
7 begin
8     open ab;
9     loop
10    fetch ab into var_a,var_b;
11    if not found then exit;end if;
12    ab_plus_var = var_a + var_b;
13    insert into test_trg (m_sum,m_group)
14    values (ab_plus_var,old.m_group);
15    end loop;
16    return new;
17 end;
```







## Перевірка роботи

```

1 SELECT * FROM public.test_trg
2
```

	m_a integer	m_b integer	m_sum integer	m_id integer	m_group integer
1	5	4	[null]	12	2
2	4	4	[null]	13	3
3	1	2	[null]	14	8
4	7	2	[null]	15	9
5	5	3	[null]	16	1
6	2	6	[null]	17	3
7	1	4	[null]	18	3

```
1  update test_trg set m_a = 25 where m_id = 12;
2  select * from test_trg
```

	 m_a integer		m_b integer		m_sum integer		m_id integer		m_group integer	
1		4		4	[null]		13		3	
2		1		2	[null]		14		8	
3		7		2	[null]		15		9	
4		5		3	[null]		16		1	
5		2		6	[null]		17		3	
6		1		4	[null]		18		3	
7		25		4	[null]		12		2	
8		[null]		[null]	29		[null]		2	

У рядку 8 в стовбці m\_sum записана сума m\_a та m\_b.