



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»

Виконав: студент III курсу

ФПМ групи КВ-83

Трофімцов Дмитро

Лабораторна робота № 3.

Засоби оптимізації роботи СУБД PostgreSQL

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

Завдання роботи полягає у наступному:

Перетворити модуль “Модель” з шаблону MVC лабораторної роботи №2 у вигляд об’єктно-реляційної проекції (ORM).

Створити та проаналізувати різні типи індексів у PostgreSQL.

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.

Вимоги до інструментарію

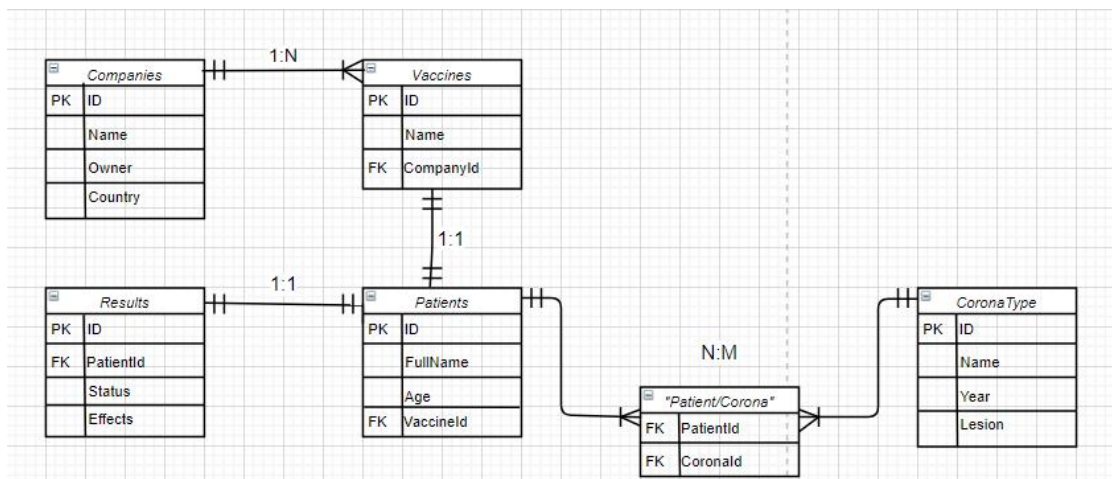
1. Бібліотека для реалізації ORM - [SQLAlchemy для Python](#) або інша з подібною функціональністю.
2. Середовище для відлагодження SQL-запитів до бази даних – pgAdmin 4.
3. СУБД - PostgreSQL 11-12.

Варіант 26 (номер залікової книжки 8126)

24	GIN, BRIN	after update, insert
----	-----------	----------------------

Завдання 1

Логічна схема бази даних “Лікарня”



Моделі ORM

Приклади запитів у вигляді ORM

Ссылка: 4

```
public class Company
```

```
{
```

Ссылка: 2

```
public int id { get; set; }
```

Ссылка: 3

```
public string name { get; set; }
```

Ссылка: 3

```
public string owner { get; set; }
```

Ссылка: 3

```
public string country { get; set; }
```

```
}
```

```
public class CoronaType
```

```
{
```

```
public int id { get; set; }
```

Ссылка: 3

```
public string name { get; set; }
```

Ссылка: 3

```
public int year { get; set; }
```

Ссылка: 3

```
public string lesion { get; set; }
```

```
}
```

```
public class Patient
```

```
{
```

```
public int id { get; set; }
```

Ссылка: 3

```
public string fullname { get; set; }
```

Ссылка: 3

```
public int age { get; set; }
```

```
}
```

```
public class PatientCorona
{
    public int id { get; set; }

    public int coronaId { get; set; }

    public int patientId { get; set; }

    public CoronaType corona { get; set; }

    public Patient patient { get; set; }
}

public class PatientVaccine
{
    public int id { get; set; }

    public int patientId { get; set; }

    public Patient patient { get; set; }
    Ссылка: 0
    public Vaccine vaccine { get; set; }
    Ссылка: 3
    public int vaccineId { get; set; }
}

public class Vaccine
{
    public int id { get; set; }

    public string name { get; set; }

    public int companyId { get; set; }

    public Company company { get; set; }
}
```

```
public class Context : DbContext
```

2

20	
21	<code>explain analyze select * from companies order by name</code>
Data Output Explain Messages Notifications	
	<div>QUERY PLAN</div> <div>text</div> <div></div>
1	Sort (cost=136537.84..139037.84 rows=1000000 width=24) (actual time=5111.535..6467.940 rows=1000000 lo...
2	Sort Key: name
3	Sort Method: external merge Disk: 33288kB
4	-> Seq Scan on companies (cost=0.00..16370.00 rows=1000000 width=24) (actual time=0.026..68.079 rows=1...
5	Planning Time: 1.381 ms
6	Execution Time: 6493.494 ms

Сортування без індексу відбувається дуже повільно.

BRIN індекс

28	<code>create index on companies using BRIN (owner);</code>
29	<code>explain analyze select * from companies order by owner</code>
Data Output Explain Messages Notifications	
	<div>QUERY PLAN</div> <div>text</div> <div></div>
1	Gather Merge (cost=76194.00..173423.09 rows=833334 width=100) (actual time=1901...
2	Workers Planned: 2
3	Workers Launched: 2
4	-> Sort (cost=75193.98..76235.65 rows=416667 width=100) (actual time=1803.174.....
5	Sort Key: owner
6	Sort Method: external merge Disk: 17696kB
7	Worker 0: Sort Method: external merge Disk: 19264kB
8	Worker 1: Sort Method: external merge Disk: 19896kB
9	-> Parallel Seq Scan on companies (cost=0.00..13512.67 rows=416667 width=100...
10	Planning Time: 0.801 ms
11	Execution Time: 3228.517 ms

Як видно, з BRIN індексом execution time значно зменшився (вдвічі). Це працює тому, що даний індекс поділяє дані на секції для того, щоб при пошуку не проходити по уже пройдених секціях.

GIN індекс

31	
32	<code>create index on companies using GIN (name);</code>
33	<code>explain analyze select * from companies order by name</code>
<div> Data Output Explain Messages Notifications </div>	
	<div> <div> <div>QUERY PLAN</div> <div>text</div> </div> <div> <div>1</div> <div>Gather Merge (cost=66226.00..163455.09 rows=833334 width=47) (actual time=609.3...</div> </div> </div> <div> <div>2</div> <div>Workers Planned: 2</div> </div> <div> <div>3</div> <div>Workers Launched: 2</div> </div> <div> <div>4</div> <div>-> Sort (cost=65225.98..66267.65 rows=416667 width=47) (actual time=561.548..69...</div> </div> <div> <div>5</div> <div>Sort Key: name</div> </div> <div> <div>6</div> <div>Sort Method: external merge Disk: 20440kB</div> </div> <div> <div>7</div> <div>Worker 0: Sort Method: external merge Disk: 17696kB</div> </div> <div> <div>8</div> <div>Worker 1: Sort Method: external merge Disk: 18712kB</div> </div> <div> <div>9</div> <div>-> Parallel Seq Scan on companies (cost=0.00..13512.67 rows=416667 width=47) ...</div> </div> <div> <div>10</div> <div>Planning Time: 0.855 ms</div> </div> <div> <div>11</div> <div>Execution Time: 1077.700 ms</div> </div>

GIN індекс зменшив execution time у 6 разів, що дуже хороший результат. Даний індекс створений для повнотекстового пошуку, тому дуже добре оптимізований для великих рядків.

Завдання 3

Створюю тригер при видаленні

```
37
38 create table deleted_companies
39 (
40     id integer not null,
41     name tsvector,
42     owner text,
43     country text
44 );
45
46 create function tr1()
47 returns trigger as $$
48     begin
49         if (TG_OP = 'DELETE') then
50             if (old.id % 2 = 1) then
51                 insert into deleted_companies
52                     (id, name, owner, country)
53                     select old.id, old.name, old.owner, old.country;
54             end if;
55             return old;
56         end if;
57     end;
58 $$ language plpgsql;
59
60 create trigger tr1
61 after delete on companies
62 for each row execute procedure tr1();
63
```

Data Output Explain Messages Notifications

CREATE TRIGGER

Query returned successfully in 86 msec.






Перевірю його роботу:

```

63
64 delete from companies where id < 100;
65 select * from deleted_companies order by id;

```

Data Output Explain Messages Notifications

	 id integer	 name tsvector	 owner text	 country text	
1	1	'gomw':1 'ky...	VOCC `RCN	SXYU	
2	3	'cfsk':2 'gakf'...	aDMO RB...	\FLV	
3	5	'bxhp':2 'ligr':1	JANJ cYUQ	fRVU	
4	7	'qko':1 'rbi':2	MKUQ nB...	iQDU	
5	9	'phtm':2 'wko...	CBPS TM...	FJNT	
6	11	'eybu':1 'qxs...	_XNE `RTF	ITWD	
7	13	'cgns':1 'rlts':2	ZVLG UHBY	gJJK	
8	15	'eqc':1 'tgaf':2	PWPC YO...	kVWP	
9	17	'frnm':2 'moh...	LLYA iNMP	_JUO	
10	19	'fcu':1 'xtgt':2	CPPD IPGA	ZQQF	
11	21	'qwid':1 'sxtp...	YKLD iFCS	CAXS	
12	23	'ajxp':2 'ovdq...	pFPC KQKD	INNw	
13	25	'fjys':1 'krsg':2	KJOT YD...	HYWW	
14	27	'ecca':1 'kko...	jITA QVAE	AIUH	
15	29	'cymg':2 'pfv...	piUM MV...	JENG	
16	31	'epft':2 'pnu':1	mCKA QI...	AVKE	
17	33	'oiia':2 'zsme...	BKMA jKXD	nFJP	
18	35	'rjpb':2 'roru':1	CNBH oC...	HTNI	
19	37	'jfnf':1 'jkkt':2	PSRA qV...	TQNN	

Тригер працює правильно.

Створю тригер при вставці і перевірці:

```

67 create or replace function tr()
68 returns trigger as $$
69     begin
70         if(TG_OP = 'INSERT') then
71             if (new.id < 10) then
72                 raise exception 'you cannot insert these records';
73             end if;
74             return new;
75
76         end if;
77     end;
78 $$ language plpgsql;
79
80
81 create trigger tr_insert
82 before insert on deleted_companies
83 for each row execute procedure tr();
84
85 insert into deleted_companies
86 (id, name, owner, country)
87 values
88 (3, 'erwfd', 'erwfd', 'erwfd')
89

```

Data Output Explain Messages Notifications

ERROR: ОШИБКА: you cannot insert these records
CONTEXT: функция PL/pgSQL tr(), строка 5, оператор RAISE

SQL state: P0001

Все працює правильно.