

# 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

3 дисципліни "Бази даних"

Тема: Засоби оптимізації роботи СУБД PostgreSQL

Виконав: Мричко Богдан Тарасович ФПМ, КП-83 **Мета:** здобуття практичних навичок використання засобів оптимізації СУБД PostgreSQL.

## 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.

Види індексів: BRIN, Hash

Умови для тригера: before update, delete

### Завдання 1

Кожний клас об'єктів було переведено у відпровідний клас моделівикористаної ORM (sequelize). Приклади класів наведено нижче

Приклад моделі: employees

# employees.ts

```
const { Sequelize, DataTypes, Model } = require('sequelize');
export default class Employees extends Model {}
Employees.init({
 employee id: {
   type: DataTypes.NUMERIC,
   allowNull: false
 name: {
   type: DataTypes.TEXT,
   allowNull: false
 },
 birth day: {
   type: DataTypes.DATE,
   allowNull: false
 },
 contact number: {
   type: DataTypes.NUMERIC,
   allowNull: false
 },
   type: DataTypes.TEXT,
   allowNull: false
 },
 post: {
   type: DataTypes.TEXT,
 department id: {
   type: DataTypes.NUMERIC,
```

```
allowNull: false
},
finished_cases: {
   type: DataTypes.NUMERIC
}
}, {
   // Other model options go here
   sequelize, // We need to pass the connection instance
   modelName: 'employees' // We need to choose the model name
});
```

Приклад контроллера: carsController

### carsController.ts

```
import Cars from '../models/cars';

export const onGet = async (): Promise<string> => {
    try {
      const cars = await Cars.get();
      return JSON.stringify(cars, null, 2);
    } catch (err) {
      console.log(err);
      return err.message;
    }
};

export const onDelete = async (id: number): Promise<string> => {
      try {
       await Cars.delete(id);
      return 'Successfully deleted';
    } catch (err) {
      return err.message;
    }
};

export const onUpdate = async (id: number, data: any):
```

```
Promise<string> => {
    try {
       await Cars.update(id, data);
       return 'Successfully update';
    } catch (err) {
       return err.message;
    }
};
```

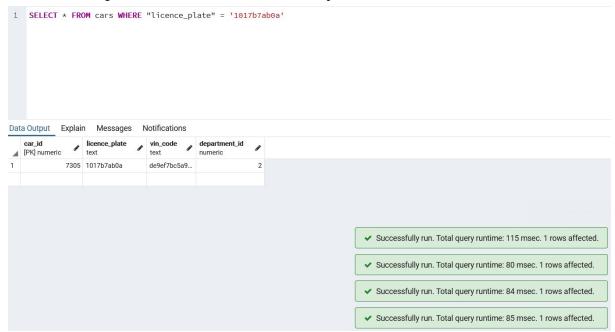
Подібним чином була переписана логіка програми. Весь код  $\varepsilon$  на GitHub

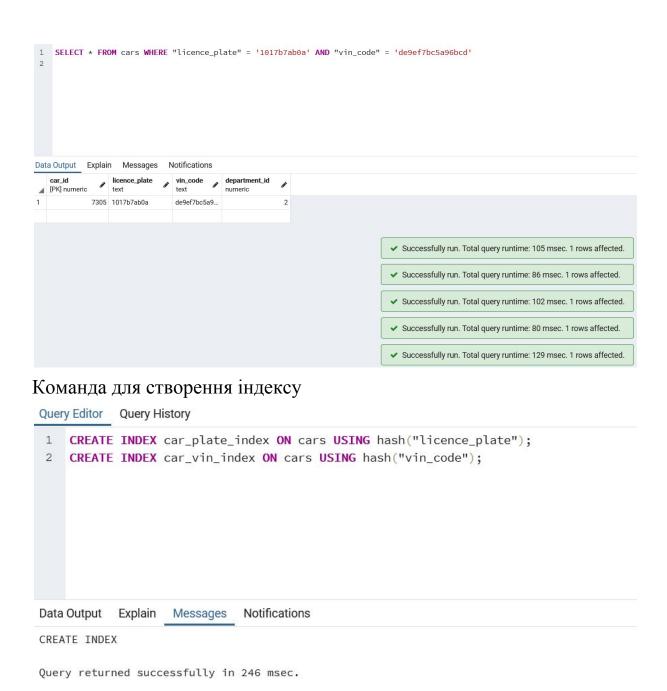
### Завдання 2

Індекс Hash

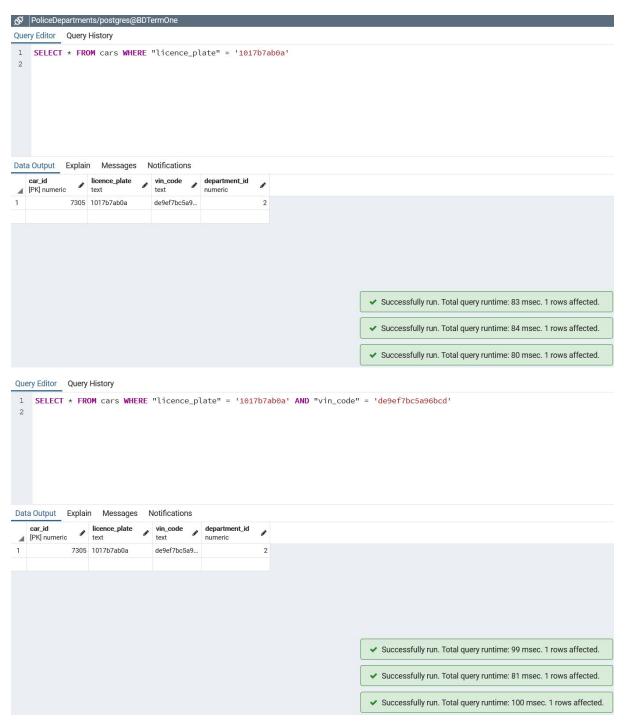
Індексування даного типу доцільно використовувати для рідко змінних атомарних значень, оскільки за основу береться така структура даних як хеш таблиця.

Наведемо приклади запитів до індексування.





# Запити після створення індексу

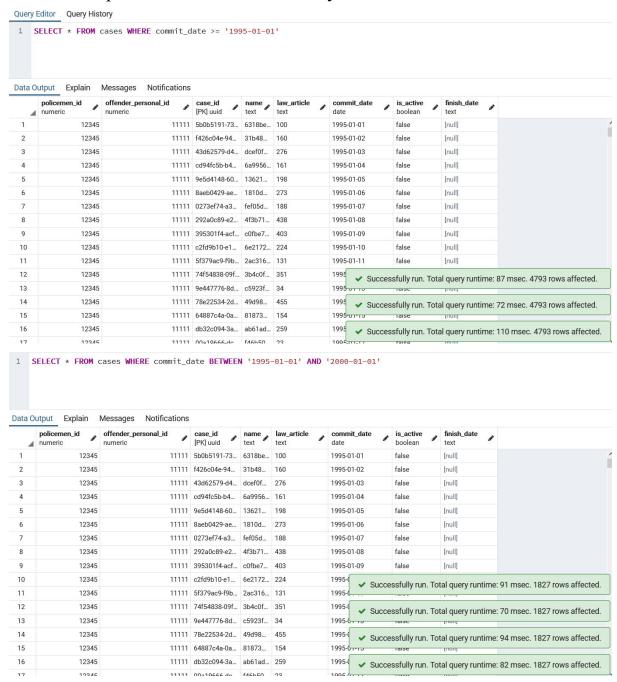


Як бачимо час запитів майже не змінився, можливо для більш помітного результату необхідно перевірити на більшій кількості даних.

### Індекс BRIN

Призначений для обробки дуже великих таблиць, у яких певні стовпці мають певну природну кореляцію з їх фізичним розташуванням у таблиці

# Наведемо приклади запитів до індексування.

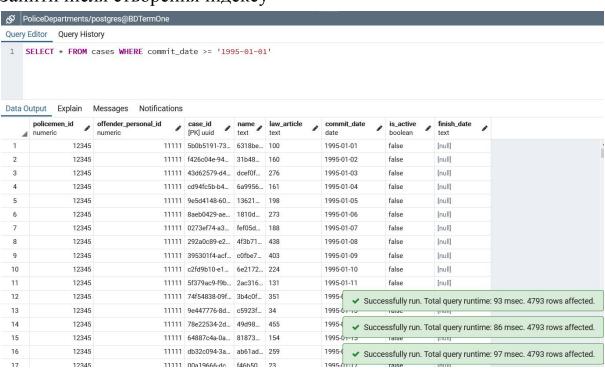


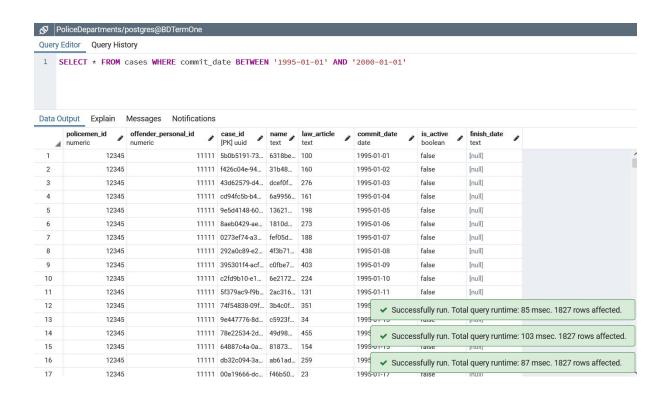
Команда для створення індексу

# Query Editor Query History 1 CREATE INDEX commit\_date\_index ON cases USING brin ("commit\_date"); Data Output Explain Messages Notifications CREATE INDEX

Query returned successfully in 146 msec.

# Запити після створення індексу





Як бачимо різниця в швидкодії доволі незначна, однак в цьому випадку, я припускаю це пов'язано зі штучністю закономірності між датами (кожна нова дата це інкремент попередньої)

### Завдання 3

Before delete trigger

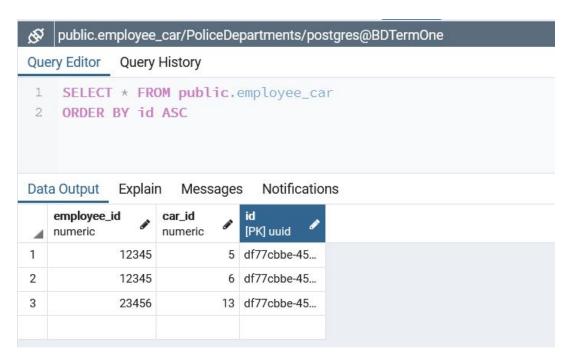
```
CREATE OR REPLACE FUNCTION on_car_delete_function()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN

    DELETE FROM employee_car
    WHERE employee_car.car_id = OLD.car_id;
    RETURN NULL;
END;
$$;

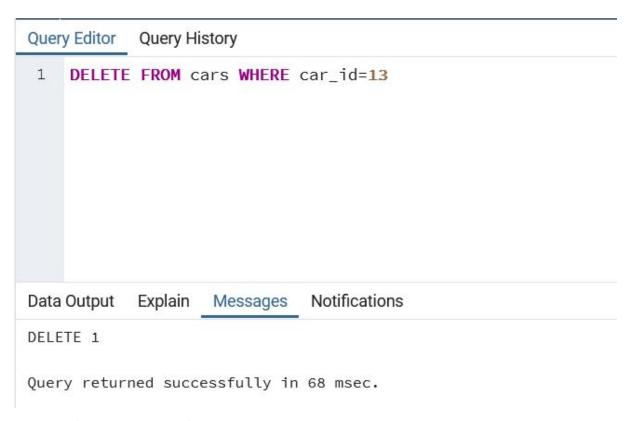
CREATE TRIGGER on_car_delete
BEFORE DELETE
ON cars
EXECUTE PROCEDURE on_car_delete_function()
```

При видалені автомобіля департаменту з бази даних відбувається видалення всіх прив'язок працівників до автомобілів Таблиці сутностей до видалення

ශි	public.cars/Police	Departments/post	gres@BDTerm	One	
Quer	y Editor Query H	listory			
1 2	SELECT * FROM	M public.cars _id ASC			
Data	Output Explain	Messages No	otifications		
4	car_id [PK] numeric	licence_plate text	vin_code text	department_id numeric	•
1	1	newlicenceplate0	f23f3f23rffw		1
2	2	newLicenceplate1	dthstvw3g54		1
3	3	rth34tb4	g34qg434g4		2
4	4	qwd12e1	324g234g234		3
5	5	fwef241f	gwrgregerg3		3
6	6	hnerhrheb	ehrtethrbt		1
7	7	ewfqweggrgv	efbegur4tga		2
8	8	34g3gq34g3	gerg4gw4		3
9	9	ngnyl7itleyj	we45wjqgrg3		1
10	10	4q3t3g34hqjy7	h3574w5tq4		2
11	11	hq45jgrtq35hq	qh5q5j5jfg34h		3
12	12	q34h35j45yf	qg43jn4hgrv		1
13	13	qh5hq45hfebq5h	ye5nen67kru		2



### Виконаємо команду видалення



Таблиці сутностей після видалення

#### Query Editor Query History SELECT \* FROM public.cars ORDER BY car\_id ASC Data Output Explain Messages Notifications car\_id licence\_plate vin\_code department\_id [PK] numeric numeric text text 1 1 newlicenceplate0 f23f3f23rffw... 1 2 2 newLicenceplate1 dthstvw3g54... 1 3 rth34tb4 2 3 g34qg434g4... 4 4 qwd12e1 324g234g234 3 5 5 fwef241f gwrgregerg3... 3 6 6 hnerhrheb ehrtethrbt 1 7 7 ewfqweggrgv efbegur4tga 2 8 34g3gq34g3 3 8 gerg4gw4 9 9 ngnyl7itleyj we45wjqgrg3... 1 10 4q3t3g34hqjy7 h3574w5tq4... 2 10 11 hq45jgrtq35hq qh5q5j5jfg34h 3 11 12 12 q34h35j45yf qg43jn4hgrv... 1 **Query Editor Query History** SELECT \* FROM public.employee\_car 1 ORDER BY id ASC Notifications **Data Output** Explain Messages employee\_id car\_id id numeric numeric [PK] uuid 1 12345 5 df77cbbe-45... 2 12345 6 df77cbbe-45...

# Before insert trigger

```
CREATE OR REPLACE FUNCTION on_case_insert_function()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    IF LENGTH(NEW.name) < 2 THEN
        NEW.name = 'default case name';
    END IF;
        RETURN NEW;
END;
$$;

CREATE TRIGGER on_case_insert
BEFORE DELETE
ON cases
EXECUTE PROCEDURE on_case_insert_function()</pre>
```

При створенні нової справи, якщо довжина назви справи менша двох символів, її назва замінюється на назву по замовчуванні.

Таблиця сутностей до внесення змін (для демострації всі дані видалені)



Виконання функції по створенню запису (з даними які викличуть спрацювання умови в тригері)

### Query Editor Query History

```
INSERT INTO cases VALUES (
2
               12345,
3
               11111,
4
               uuid_in(md5(random()::text || clock_timestamp()::text)::cstring),
5
               (random() * (500 - 1) + 1)::integer,
6
7
               date '1980-09-28',
               false
8
9
               );
```

# Таблиця сутностей після внесення змін



## Висновок

Я здобув практичні навички використання засобів оптимізації СУБД PostgreSQL таких як індексування та створення тригерів, що реагують на деякі події, заміщуючи собою програмну реалізацію у додатках, що пришвидшує роботу та зменшує кількість запитів.