

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 ІII курсу

ФПМ групи КВ-83

Кубай Олег

Київ – 2020

**Засоби оптимізації роботи СУБД PostgreSQL**

*Метою роботи* є здобуття практичних навичок використання засобів

оптимізації СУБД PostgreSQL.

*Завдання* роботи полягає у наступному:

1. Перетворити модуль “Модель” з шаблону MVC лабораторної роботи №2

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

1. Створити та проаналізувати різні типи індексів у PostgreSQL.
2. Розробити тригер бази даних PostgreSQL.

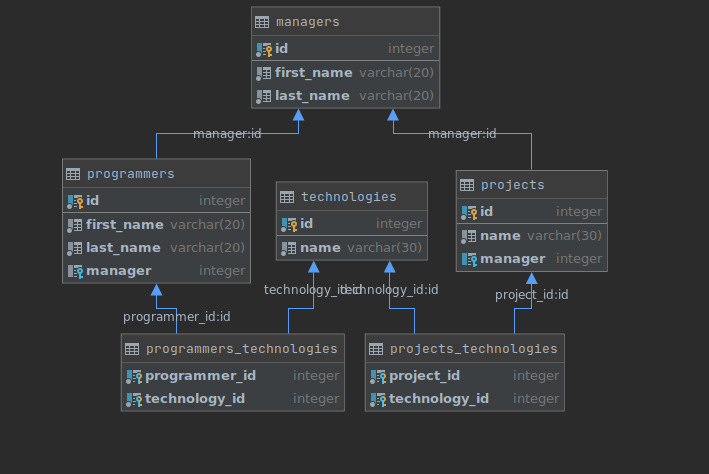
**Варіант 10**

У другому завданні проаналізувати індекси HASH,BRIN.

Умова для тригера – after delete,insert

**Завдання 1**

Логічна схема бази даних



Для перетворення модулю “Model” програми, створеної в 2 лабораторній роботі, у вигляд об’єктно-реляційної моделі використовую бібліотеку Hibernate Core.

Зобразимо сутнісні класи програми:

@Data@AllArgsConstructor@NoArgsConstructor@EqualsAndHashCode(onlyExplicitlyIncluded = true)@ToString@Entity@Table(name = "managers")public class Manager { @EqualsAndHashCode.Include @Id @GeneratedValue(strategy = GenerationType.*IDENTITY*) private Integer id; @Column(name = "first\_name") private String firstName; @Column(name = "last\_name") private String lastName; @OneToMany(mappedBy = "manager", fetch = FetchType.*EAGER*) @ToString.Exclude @JsonIgnore private List<Programmer> programmers; @OneToOne(mappedBy = "manager") @ToString.Exclude @JsonIgnore private Project project;}

@Data@AllArgsConstructor@NoArgsConstructor@EqualsAndHashCode(onlyExplicitlyIncluded = true)@ToString@Entity@Table(name = "programmers")public class Programmer { @EqualsAndHashCode.Include @Id @GeneratedValue(strategy = GenerationType.*IDENTITY*) private Integer id; @Column(name = "first\_name") private String firstName; @Column(name = "last\_name") private String lastName; @ManyToOne @JoinColumn(name = "manager") @JsonIgnore @ToString.Exclude private Manager manager; @ManyToMany @JsonIgnore @ToString.Exclude private List<Technology> technologies;}

@AllArgsConstructor@NoArgsConstructor@EqualsAndHashCode(onlyExplicitlyIncluded = true)@ToString@Entity@Table(name = "projects")public class Project { @EqualsAndHashCode.Include @Id @GeneratedValue(strategy = GenerationType.*IDENTITY*) private Integer id; @Column(name = "name") private String name; @OneToOne @JoinColumn(name = "manager") @JsonIgnore @ToString.Exclude private Manager manager; @ManyToMany @JsonIgnore @ToString.Exclude private List<Technology> technologies;}

@Data@AllArgsConstructor@NoArgsConstructor@EqualsAndHashCode(onlyExplicitlyIncluded = true)@ToString@Entity@Table(name = "technologies")public class Technology { @EqualsAndHashCode.Include @Id @GeneratedValue(strategy = GenerationType.*IDENTITY*) private Integer id; @Column(name = "name") private String name; @ManyToMany @JsonIgnore @ToString.Exclude private List<Project> projects; @ManyToMany @JsonIgnore @ToString.Exclude private List<Programmer> programmers;}

Конфігерація hibernate:

<?xml version='1.0' encoding='utf-8'?>  
<!DOCTYPE hibernate-configuration PUBLIC  
 "-//Hibernate/Hibernate Configuration DTD//EN"  
 "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">  
  
<hibernate-configuration>  
 <session-factory>  
 <property name="hibernate.connection.driver\_class">org.postgresql.Driver</property>  
 <property name="hibernate.connection.url">jdbc:postgresql://localhost:5432/db\_labs</property>  
 <property name="hibernate.connection.username">test</property>  
 <property name="hibernate.connection.password">testpass</property>  
 <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQL82Dialect</property>  
  
 <property name="hibernate.connection.pool\_size">1</property>  
 <property name="show\_sql">true</property>  
 <property name="hibernate.current\_session\_context\_class">thread</property>  
  
 <mapping class="com.naofi.model.entity.Manager"/>  
 <mapping class="com.naofi.model.entity.Programmer"/>  
 <mapping class="com.naofi.model.entity.Project"/>  
 <mapping class="com.naofi.model.entity.Technology"/>  
 </session-factory>  
</hibernate-configuration>

Конфігурація Spring framework:

<?xml version="1.0" encoding="UTF-8"?><beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:c="http://www.springframework.org/schema/c" xmlns:p="http://www.springframework.org/schema/p" xmlns:mvc="http://www.springframework.org/schema/mvc" xmlns:context="http://www.springframework.org/schema/context" xmlns:aop="http://www.springframework.org/schema/aop" xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/mvc https://www.springframework.org/schema/mvc/spring-mvc.xsd http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context.xsd http://www.springframework.org/schema/aop https://www.springframework.org/schema/aop/spring-aop.xsd"> <import resource="db.xml"/> <mvc:annotation-driven/> <context:component-scan base-package="com.naofi.controllers"/> <mvc:resources mapping="/css/\*\*" location="/css/"/> <mvc:resources mapping="/js/\*\*" location="/js/"/> <bean name="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver" p:viewClass="org.springframework.web.servlet.view.JstlView" c:prefix="/jsp/" c:suffix=".jsp"/> <bean name="gen" class="com.naofi.util.IdGenerator" scope="request"/> <!-- Use aop to begin and stop transaction for controller methods --> <bean id="transactionalBean" class="com.naofi.controllers.aop.TransactionAspectBean"/> <aop:config> <aop:aspect id="transactionAspect" ref="transactionalBean"> <aop:pointcut id="transactional" expression="execution(\* com.naofi.controllers.\*EntityController.\*(..))"/> <aop:before method="beginTransaction" pointcut-ref="transactional"/> <aop:after-returning method="commit" pointcut-ref="transactional"/> <aop:after-throwing method="rollback" pointcut-ref="transactional"/> </aop:aspect> </aop:config></beans>

Для бази даних:

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:p="http://www.springframework.org/schema/p"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context.xsd">  
  
 <bean name="hibernateConfiguration" class="org.hibernate.cfg.Configuration" init-method="configure" autowire-candidate="false"/>  
  
 <bean name="factory" class="org.hibernate.SessionFactory" factory-bean="hibernateConfiguration"  
 factory-method="buildSessionFactory"/>  
  
 <bean name="postgreDS" class="org.springframework.jdbc.datasource.DriverManagerDataSource"  
 scope="prototype"  
 p:url="jdbc:postgresql://localhost:5432/db\_labs"  
 p:username="test"  
 p:password="testpass"  
 p:driverClassName="org.postgresql.Driver"/>  
  
 <bean name="generator" class="com.naofi.model.DataGenerator"/>  
</beans>

**Завдання 2**

**Створення та аналіз індекса BTREE**

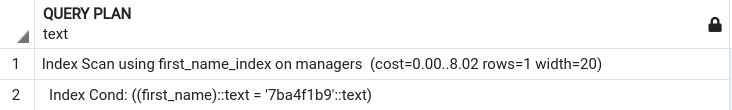
Для дослідження індексу була використана таблиця managers та поле first\_name. Вони проіндексовані як HASH. У таблицю було занесено 100000 записів. Виконаний запит.

select \* from managers where first\_name = '7ba4f1b9';

До iндексування.

****

Пiсля iндексування.

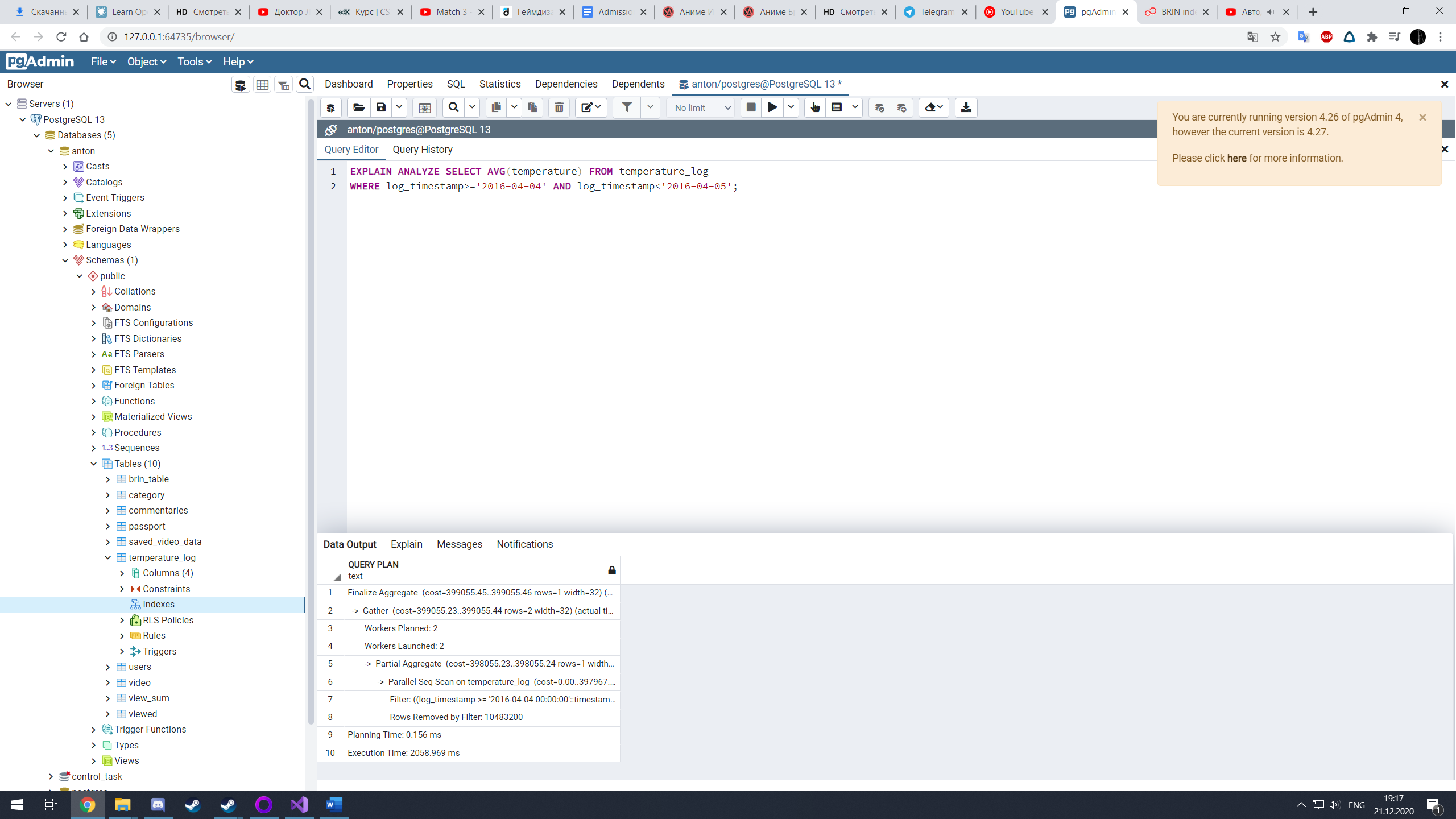


**Створення та аналіз індекса BRIN**

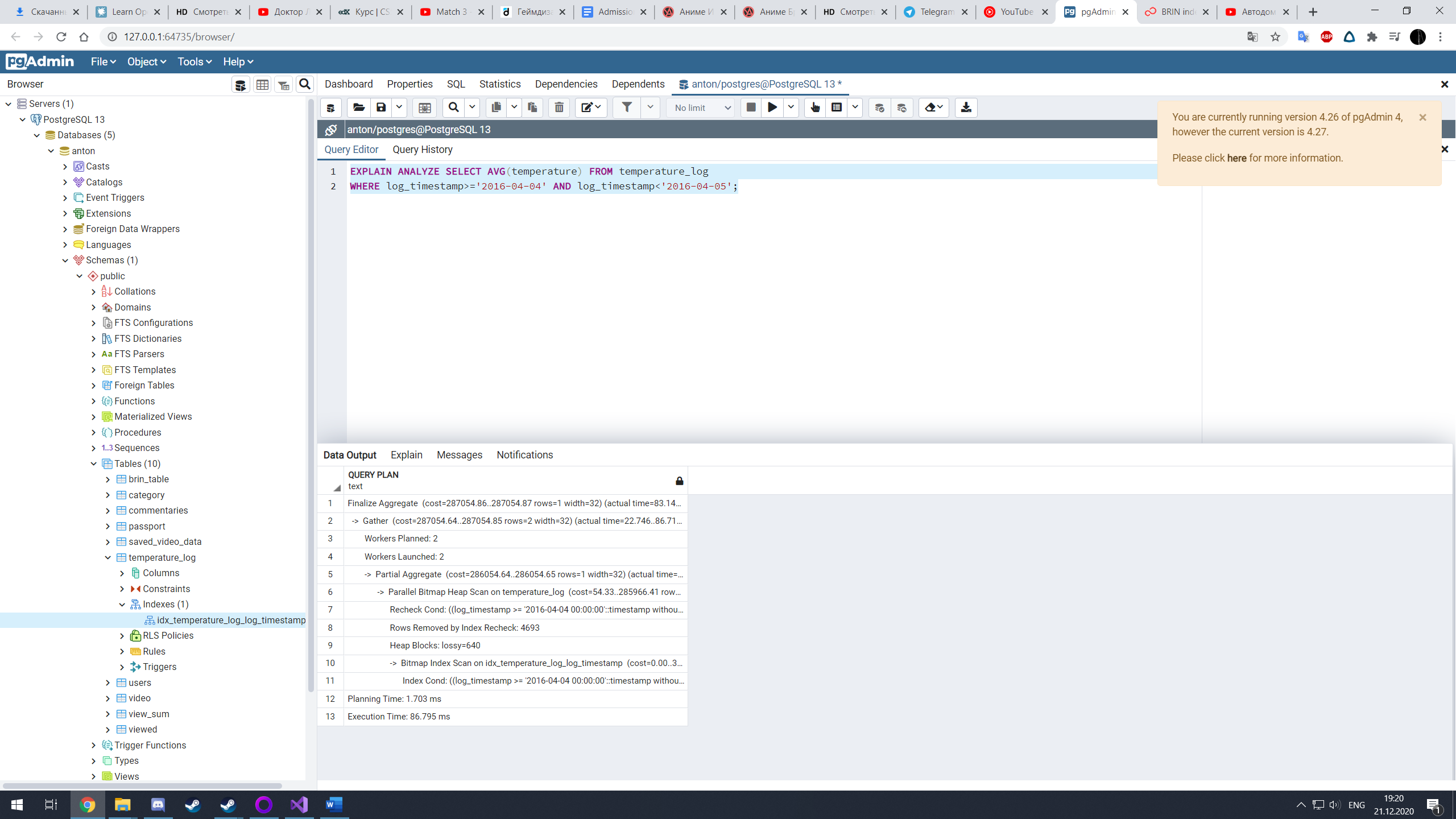
Iндекс BRIN використовують для данних значення яких вiдповiдає іх розташуванню в пам'яті. Була використана таблиця що має поле log\_timestamp що зберігає час отримання данних. В таблицю занесено 31536001 кортежей. Виконаний запит:

SELECT AVG(temperature) FROM temperature\_log  
WHERE log\_timestamp>='2016-04-04' AND log\_timestamp<'2016-04-05';

До індексування



Після індексування.



**Завдання 3**

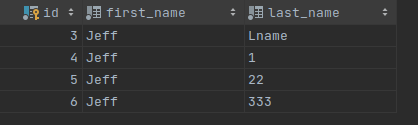
Тригер:

DROP TABLE IF EXISTS insert\_log\_table;  
DROP TRIGGER IF EXISTS after\_delete\_trigger ON managers;  
DROP FUNCTION IF EXISTS *trigger\_function*;  
  
  
CREATE TABLE insert\_log\_table  
(  
 id serial primary key,  
 size INTEGER,  
 insert\_date DATE default NOW()  
);  
  
create or replace function *trigger\_function*()  
 returns trigger  
 language plpgsql  
as $$  
 declare  
 len integer := 0;  
 manager record;  
 begin  
 for manager in select \* from managers  
 loop  
 if manager.first\_name = OLD.first\_name then  
 len = len + char\_length(manager.last\_name);  
 end if;  
 end loop;  
 insert into insert\_log\_table (size) VALUES (len);  
 return OLD;  
 end;  
 $$;  
  
CREATE TRIGGER after\_delete\_trigger  
 after delete  
 on managers  
 for each row  
execute function *trigger\_function*();

Принцип роботи:

Тригер спрацьовує після видалення з таблиці managers та зберігає суму довжин усіх last\_name менеджерів у яких first\_name сіпвпадає з тим що видаляється.

Таблиця перед видаленням:



Видаляємо:

delete from managers where id = 3;

insert\_log\_table:



Перевіримо суму:

select sum(char\_length(last\_name)) from managers  
where first\_name = 'Jeff';

