Министерство науки и высшего образования Российской Федерации

**Муромский институт (филиал)**

федерального государственного бюджетного образовательного учреждения высшего образования

**«Владимирский государственный университет**

**Имени Александра Григорьевича и Николая Григорьевича Столетовых»**

**(МИВлГУ)**

Факультет ИТ

Кафедра ПИн

*Отчет*

*по практике*

Научно-исследовательская работа .

(наименование практики)

Руководитель

(фамилия, инициалы)

(подпись) (дата)

Студент ПИн-119

(группа)

Лямина И.А.

(фамилия, инициалы)

(подпись) (дата)

Муром 2021

Содержание

Введение 3

1. Отдел автоматизированных систем управления производством 4

1.1 Общие положения 4

1.2 Руководящие документы 4

1.3 Основные задачи 5

1.4 Организационная структура отдела АСУП 6

1.5 Функции 6

1.5.1 Общие функции отдела АСУП 6

1.5.2 Функции структурных подразделений отдела АСУП 8

2. Индивидуальное задание 13

2.1 PostgreSQL 13

2.2 Описание таблиц 14

2.3 Изменяемые представления 21

2.4 Функции 24

2.5 Индексы 28

Заключение 30

Введение

...

1 Отдел автоматизированных систем управления производством

…

2 Индивидуальное задание

Согласно полученному от руководителя практики заданию необходимо:

* ознакомиться с сервером баз данных PostgreSQL;
* разработать БД телефонного справочника предприятия.

2.1 PostgreSQL

PostgreSQL — это популярная свободная объектно-реляционная система управления базами данных. PostgreSQL поддерживает большую часть стандарта SQL и предлагает множество современных функций:

* сложные запросы;
* внешние ключи;
* триггеры;
* изменяемые представления;
* транзакционная целостность;
* многоверсионность.

Кроме того, пользователи могут всячески расширять возможности PostgreSQL, например создавая свои

* типы данных,
* функции,
* операторы,
* агрегатные функции,
* методы индексирования,
* процедурные язык.

Для упрощения администрирования на сервере PostgreSQL в базовый комплект установки входит такой инструмент как pgAdmin. Он представляет графический клиент для работы с сервером, через который в удобном виде можно создавать, удалять, изменять базы данных и управлять ими.

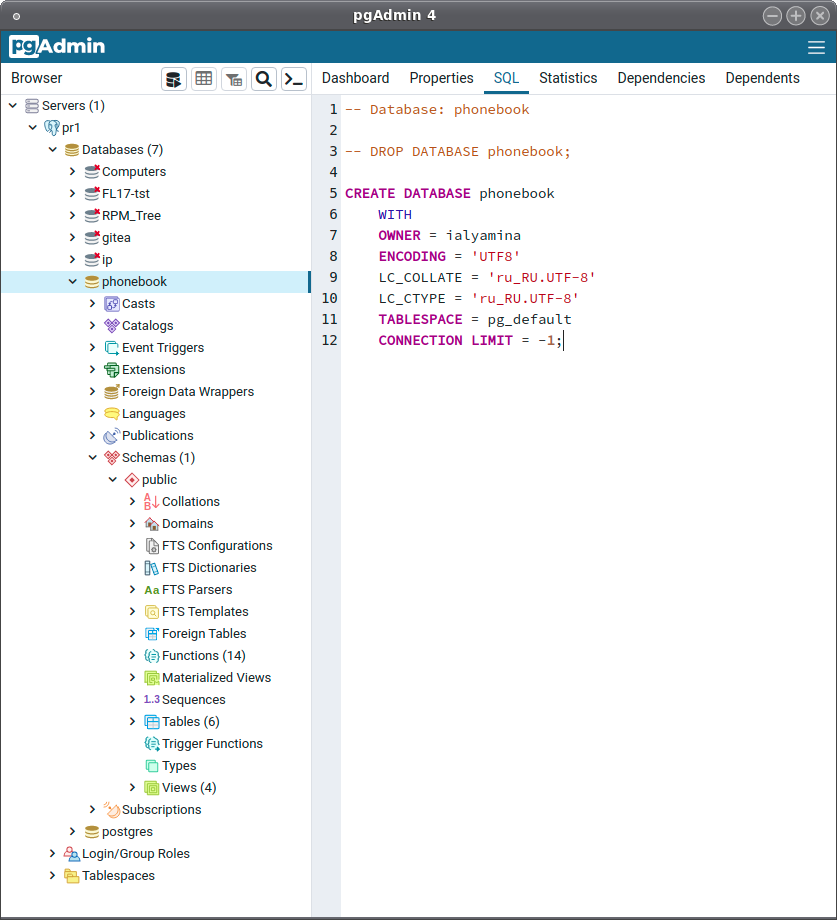


Рисунок 1 – скриншот pgAdmin

2.2 Описание таблиц

В ходе практики нужно было разработать базу данных телефонного справочника. Для данной БД были построены следующие таблицы: divisions — таблица с отделами, subdivisions — таблица подотделов, а также phone\_numbers — таблица с номерами телефонов. По заданию руководителя нужно было хранить в бд комментарии пользователей, для этого была разработана таблица с комментариями — comments. Также у каждого пользователя есть права на изменения информации в БД, но при этом нужно было хранить информацию о соответствии пользователя и отдела, который он имеет право изменять. Для этого была создана таблица с пользователями — users и таблица прав— relation\_user\_division.

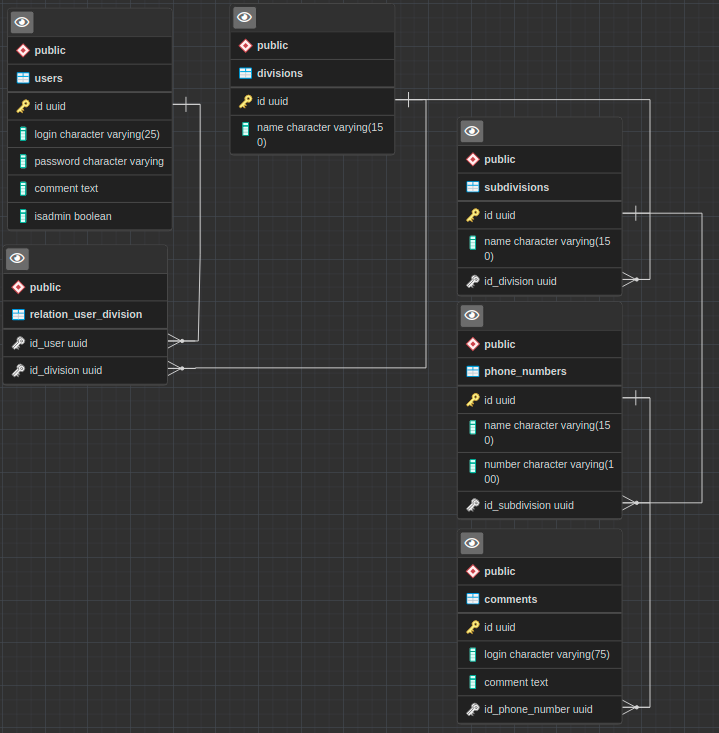


Рисунок 2 — модель БД

2.2.1 Таблица отделов - divisions

В данной таблице хранится id и name - название отдела. Также в таблице присутствуют следующие ограничения:

* id является первичным ключом;
* name должно быть уникальным.

Листинг создания таблицы:

CREATE TABLE IF NOT EXISTS public.divisions

(

id uuid NOT NULL DEFAULT uuid\_generate\_v1(),

name character varying(150) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT divisions\_pkey PRIMARY KEY (id),

CONSTRAINT name\_unique UNIQUE (name)

)



Рисунок 3 — таблица отделов

2.2.2 Таблица подотделов - subdivisions

В данной таблице хранится id, name - название подотдела и id\_division - внешний ключ. Также в таблице присутствуют следующие ограничения:

* id является первичным ключом;
* связка name и id\_division должна быть уникальной, так как в каждом отделе должны быть подотделы с уникальными названиями;
* id\_division является внешним ключом, который хранит id отдела, которому принадлежит данный подотдел. Также при удалении отдела удалятся все строки подотделов, которые ссылаются на данный отдел.

Листинг создания таблицы:

CREATE TABLE IF NOT EXISTS public.subdivisions

(

id uuid NOT NULL DEFAULT uuid\_generate\_v1(),

name character varying(150) COLLATE pg\_catalog."default" NOT NULL,

id\_division uuid NOT NULL,

CONSTRAINT subdivisions\_pkey PRIMARY KEY (id),

CONSTRAINT name\_fk\_unique UNIQUE (name, id\_division),

CONSTRAINT fk\_division FOREIGN KEY (id\_division)

REFERENCES public.divisions (id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE

)

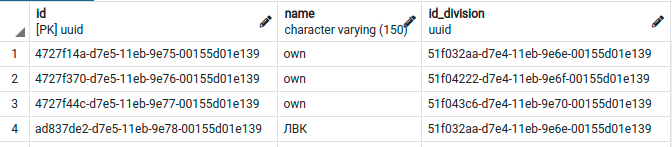


Рисунок 4 — таблица подотделов

2.2.3 Таблица с номерами телефонов — phone\_numbers

В данной таблице хранится id, name – название бюро, number – номер телефона и id\_subdivision - внешний ключ. Также в таблице присутствуют следующие ограничения:

* id является первичным ключом;
* связка name и id\_subdivision должна быть уникальной, так как в каждом подотделе должны быть бюро с уникальными названиями;
* связка name, number и id\_subdivision должна быть уникальной,
* id\_subdivision является внешним ключом, который хранит id подотдела, которому принадлежит данное бюро. Также при удалении подотдела удалятся все строки бюро, которые ссылаются на данный подотдел.

Листинг создания таблицы:

CREATE TABLE IF NOT EXISTS public.phone\_numbers

(

id uuid NOT NULL DEFAULT uuid\_generate\_v1(),

name character varying(150) COLLATE pg\_catalog."default" NOT NULL,

"number" character varying(100) COLLATE pg\_catalog."default" NOT NULL,

id\_subdivision uuid NOT NULL,

CONSTRAINT phone\_numbers\_pkey PRIMARY KEY (id),

CONSTRAINT pn\_name\_fk\_unique UNIQUE (name, id\_subdivision),

CONSTRAINT pn\_name\_number\_unique UNIQUE (name, "number", id\_subdivision),

CONSTRAINT fk\_subdivison FOREIGN KEY (id\_subdivision)

REFERENCES public.subdivisions (id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE

)

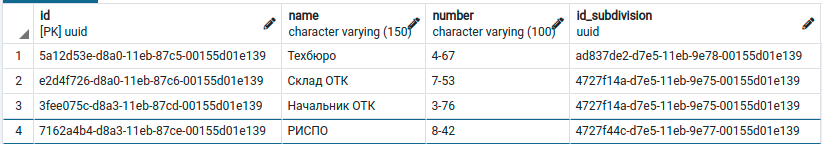


Рисунок 5 - таблица с номерами телефонов

2.2.4 Таблица пользователей— users

В данной таблице хранится id, login и password – соответственно логин и пароль пользователя, comment – комментарий к пользователю (для администратора), isadmin – указывает на то, является ли пользователь администратором. Также в таблице присутствуют следующие ограничения:

* id является первичным ключом;
* login должен быть уникальным.

Листинг создания таблицы:

CREATE TABLE IF NOT EXISTS public.users

(

id uuid NOT NULL DEFAULT uuid\_generate\_v1(),

login character varying(25) COLLATE pg\_catalog."default" NOT NULL,

password character varying COLLATE pg\_catalog."default" NOT NULL,

comment text COLLATE pg\_catalog."default",

isadmin boolean NOT NULL DEFAULT false,

CONSTRAINT users\_pkey PRIMARY KEY (id),

CONSTRAINT unique\_login UNIQUE (login)

)

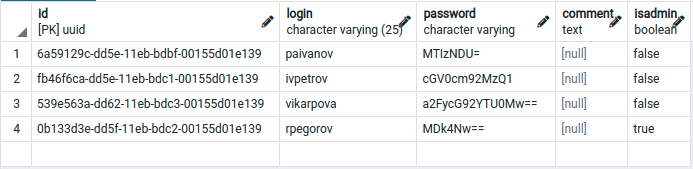


Рисунок 6 - таблица пользователей

2.2.5 Таблица комментариев— comments

В данной таблице хранится id, login –логин пользователя, который написал данный комментарий, comment – комментарий пользователя, id\_phone\_number – внешний ключ. Также в таблице присутствуют следующие ограничения:

* id является первичным ключом;
* связка login и id\_phone\_number должна быть уникальной, так как пользователь может оставить только один комментарий для каждого номера телефона;
* id\_phone\_number является внешним ключом, который хранит id телефонного номера, которому принадлежит данный комментарий. Также при удалении номера телефона удаляются все строки комментариев, которые ссылаются на данный номер телефона.

Листинг создания таблицы:

CREATE TABLE IF NOT EXISTS public.comments

(

id uuid NOT NULL DEFAULT uuid\_generate\_v1(),

login character varying(75) COLLATE pg\_catalog."default" NOT NULL,

comment text COLLATE pg\_catalog."default" NOT NULL,

id\_phone\_number uuid NOT NULL,

CONSTRAINT comments\_pkey PRIMARY KEY (id),

CONSTRAINT cm\_login\_fk\_unique UNIQUE (login, id\_phone\_number),

CONSTRAINT fk\_phone\_number FOREIGN KEY (id\_phone\_number)

REFERENCES public.phone\_numbers (id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE

)

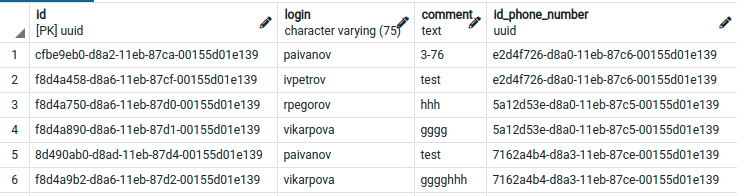


Рисунок 7 - таблица комментариев

2.2.6 Таблица прав— relation\_user\_division

В данной таблице хранятся два внешних ключа - id\_user и id\_division. Также в таблице присутствуют следующие ограничения:

* id\_user является внешним ключом, который хранит id пользователя. Также при удалении пользователя удаляются все строки прав, которые были у данного пользователя;
* id\_division является внешним ключом, который хранит id отдела. Также при удалении отдела у пользователя не будет права на изменение информации в данном отделе.

Листинг создания таблицы:

CREATE TABLE IF NOT EXISTS public.relation\_user\_division

(

id\_user uuid NOT NULL,

id\_division uuid NOT NULL,

CONSTRAINT fk\_divisions FOREIGN KEY (id\_division)

REFERENCES public.divisions (id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE,

CONSTRAINT fk\_users FOREIGN KEY (id\_user)

REFERENCES public.users (id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE

)

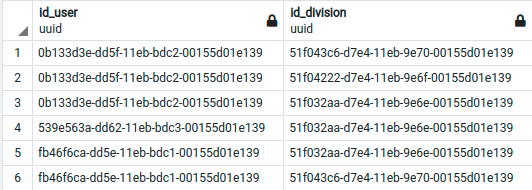


Рисунок 8 - таблица прав

2.3 Представления

Представление создает виртуальную таблицу, содержимое которой (столбцы и строки) определяется запросом. Используется для создания представления данных, содержащихся в одной или более таблицах базы данных. Например, представление можно использовать в следующих целях:

* для направления, упрощения и настройки восприятия информации в базе данных каждым пользователе;
* в качестве механизма безопасности, позволяющего пользователям обращаться к данным через представления, но не предоставляя им разрешений на непосредственный доступ к базовым таблицам;
* для предоставления интерфейса обратной совместимости, моделирующего таблицу, схема которой изменилась.

2.3.1 Представление vsubdivisions

Представление предназначено для упрощения восприятия информации. Благодаря ему можно сразу понять какому отделу принадлежит данный подотдел, так как вместо id отдела показывается его имя.

Листинг:

СREATE OR REPLACE VIEW public.vsubdivisions

AS

SELECT d.name AS division,

s.name AS subdivision

FROM divisions d

LEFT JOIN subdivisions s ON s.id\_division = d.id;

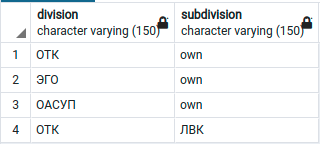


Рисунок 9 – скриншот работы vsubdivisions

2.3.2 Представление vphones

Представление предназначено для упрощения восприятия информации. Благодаря ему можно сразу понять к какому отделу и подотделу относится данное бюро и номер телефона, вместо того чтобы видеть просто id подотдела, к которому относится данное бюро и номер телефона.

Листинг:

CREATE OR REPLACE VIEW public.vphones

AS

SELECT d.name AS division,

s.name AS subdivision,

pn.name,

pn.number

FROM divisions d

JOIN subdivisions s ON s.id\_division = d.id

JOIN phone\_numbers pn ON pn.id\_subdivision = s.id;

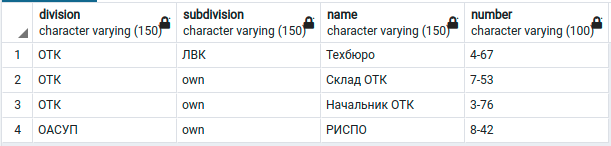


Рисунок 10 – скриншот работы vphones

2.3.3 Представление vcomments

Представление предназначено для упрощения восприятия информации. Благодаря ему можно сразу понять к какому номеру телефона привязан данный комментарий, вместо того чтобы видеть id данного номера телефона.

Листинг:

CREATE OR REPLACE VIEW public.vcomments

AS

SELECT d.name AS division,

s.name AS subdivision,

pn.name,

pn.number,

cm.login,

cm.comment

FROM divisions d

JOIN subdivisions s ON s.id\_division = d.id

JOIN phone\_numbers pn ON pn.id\_subdivision = s.id

LEFT JOIN comments cm ON cm.id\_phone\_number = pn.id;

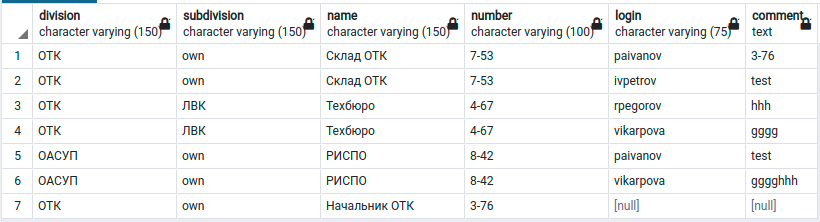


Рисунок 11 – скриншот работы vcomments

2.3.4 Представление vrelation\_user\_division

Представление предназначено для упрощения восприятия информации. Благодаря ему можно узнать логин пользователя и название отдела, который данный пользователь может изменять, вместо того чтобы видеть два id.

Листинг:

CREATE OR REPLACE VIEW public.vrelation\_user\_division

AS

SELECT u.login,

d.name

FROM users u

JOIN relation\_user\_division r ON u.id = r.id\_user

JOIN divisions d ON d.id = r.id\_division;

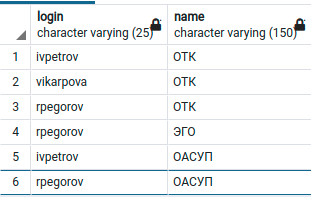


Рисунок 12 – скриншот работы vcomments

2.4 Функции

2.4.1 Функция add\_relation\_user\_division

Функция предназначена для добавления строк в таблицу прав relation\_user\_division. Благодаря этой функции добавление происходит не по id, а просто передав название отдела и логин пользователя.

Листинг:

CREATE OR REPLACE FUNCTION public.add\_relation\_user\_division(

username character varying,

name\_division character varying)

RETURNS void

LANGUAGE 'plpgsql'

AS $BODY$

begin

insert into relation\_user\_division (id\_user, id\_division)

values (

(select u.id from users as u where u.login = username),

(select d.id from divisions as d where d.name = name\_division)

);

end

$BODY$;

2.4.2 Функция getnumbersbylogin

Функция предназначена для показа информации конкретному пользователю, так как каждый пользователь может видеть только свои комментарии. На вход в функцию приходит логин пользователя. Функция возвращает таблицу со всеми отделами, подотделами, бюро, номерами телефона и комментарием данного пользователя к конкретному номеру, если он есть.

Листинг:

CREATE OR REPLACE FUNCTION public.getnumbersbylogin(

username character varying)

RETURNS TABLE(division character varying, subdivision character varying, name character varying, number character varying, login character varying, comment text)

LANGUAGE 'plpgsql'

AS $BODY$

begin

return query

select

d.name as division,

s.name as subdivision,

pn.name,

pn.number,

cm.login,

cm.comment

from divisions as d

inner join subdivisions as s on s.id\_division = d.id

inner join phone\_numbers as pn on pn.id\_subdivision = s.id

left join comments as cm on cm.id\_phone\_number = pn.id AND cm.login=userName;

end;

$BODY$;

Вызов функции:

select \* from getnumbersbylogin('paivanov')

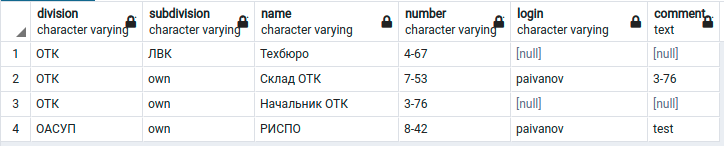


Рисунок 13 – скриншот работы функции getnumbersbylogin

2.4.3 Функция searchdata

Функция предназначена для поиска данных по БД. На вход в функцию приходят данные для поиска и логин, так как можно показать лишь те комментарии, которые принадлежат пользователю, который будет осуществлять поиск. Поиск происходит по отделам, подотделам, бюро, номерам телефонов и комментариям.

Листинг:

CREATE OR REPLACE FUNCTION public.searchdata(

search\_data character varying,

username character varying)

RETURNS TABLE(division character varying, subdivision character varying, name character varying, number character varying, login character varying, comment text)

LANGUAGE 'plpgsql'

AS $BODY$

begin

return query

select \* from getnumbersbylogin(username) as gn

where gn.division = search\_data or

gn.subdivision = search\_data or

gn."name" = search\_data or

gn."number" = search\_data or

gn."comment" LIKE Concat('%', search\_data, '%');

end

$BODY$;

Вызов функции:

select \* from searchdata('ggg', 'vikarpova')

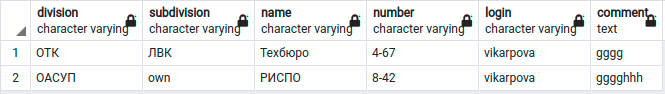


Рисунок 14 – скриншот работы функции searchdata

select \* from searchdata('own', 'vikarpova')

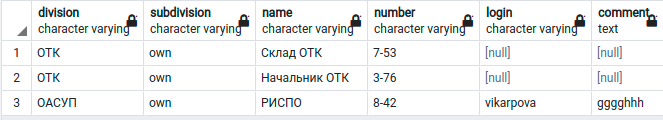


Рисунок 15 – скриншот работы функции searchdata

2.4.4 Функция statistics

Функция предназначена для получения статистики в БД. Можно узнать количество отделов, подотделов, пользователей, номеров телефона и среднее количество номеров телефона в одном отделе.

Листинг:

CREATE OR REPLACE FUNCTION public.statistics(

)

RETURNS TABLE("Count of divisions" bigint, "Count of subdivisions" bigint, "Count of users" bigint, "Count of phone numbers" bigint, "Avg of phone numbers" numeric)

LANGUAGE 'plpgsql'

AS $BODY$

declare

count\_div bigint;

count\_sub bigint;

count\_users bigint;

count\_pn bigint;

avg\_pn numeric;

begin

select count(\*) into count\_div from divisions;

select count(\*) into count\_sub from subdivisions;

select count(\*) into count\_users from users;

select count(\*) into count\_pn from phone\_numbers;

select avg(foo.count) from (

select count(\*)

from phone\_numbers as pn

join subdivisions as sub on sub.id = pn.id\_subdivision

join divisions as d on d.id = sub.id\_division

group by d.id

) as foo into avg\_pn;

return query

select count\_div, count\_sub, count\_users, count\_pn, avg\_pn;

end;

$BODY$;

Вызов функции:

select \* from statistics()

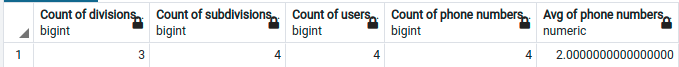
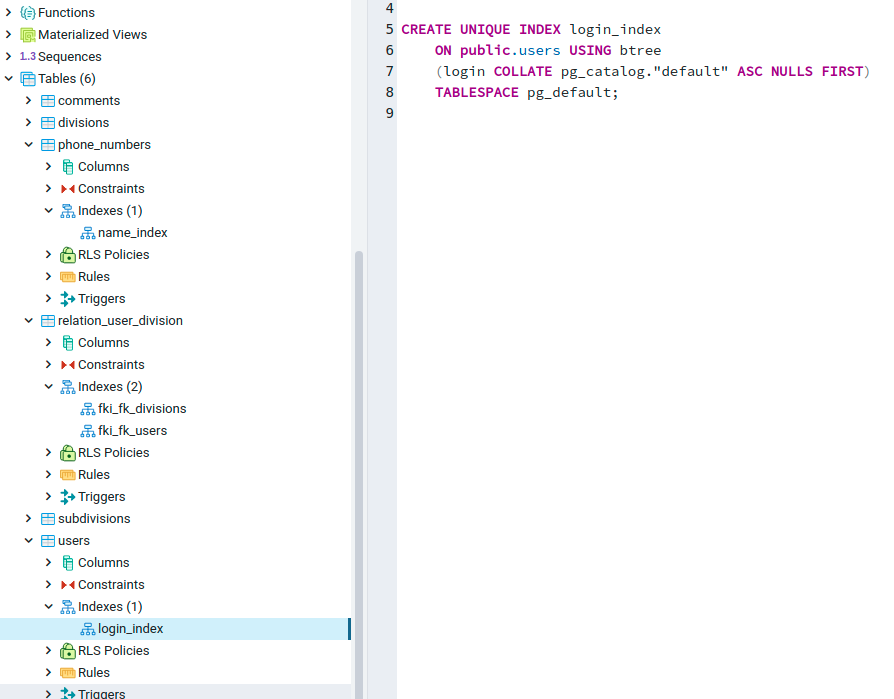


Рисунок 16 – скриншот работы функции statistics

2.5 Индексы

Индекс — объект [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), создаваемый с целью повышения производительности [поиска данных](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B8%D1%81%D0%BA_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному критерию путём последовательного просмотра таблицы строка за строкой может занимать много времени. Индекс формируется из значений одного или нескольких столбцов таблицы и указателей на соответствующие строки таблицы и, таким образом, позволяет искать строки, удовлетворяющие критерию поиска. Ускорение работы с использованием индексов достигается в первую очередь за счёт того, что индекс имеет структуру, оптимизированную под поиск — например, [сбалансированного дерева](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%B0%D0%BB%D0%B0%D0%BD%D1%81%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE).

Индекс помогает ускорить запросы на получение данных (SELECT [WHERE]), но замедляет процесс добавления и изменения записей (INSERT, UPDATE). Индексы могут быть добавлены или удалены без влияния на сами данные.

Рисунок 17 – скриншот созданных индексов

Заключение

В ходе производственной практики я ознакомилась с организационной структурой предприятия, задачами и функциями отдела автоматизированных систем управления производством, функциями структурных подразделений отдела АСУП.

Согласно заданию руководителя практики, были изучены и решены следующие задачи:

* ознакомление с сервером баз данных PostgreSQL;
* разработка БД телефонного справочника предприятия.