

## Оглавление

Оглавление .....	2
Лабораторная работа №1. Проектирование структуры базы данных «Программы» .....	5
1     Анализ предметной области.....	5
2     Разработка концептуальной модели .....	6
3     Разработка логической модели .....	7
4     Выбор СУБД .....	11
5     Физическое проектирование базы данных .....	12
Лабораторная работа №2. Создание базы данных в СУБД PostgreSQL .....	14
1     Начало работы.....	14
2     Создание таблиц .....	14
2.1 Создание таблицы tip_programmy .....	14
2.2 Создание таблицы programmer .....	15
2.3 Создание таблицы polzovatel.....	15
2.4 Создание таблицы programma .....	15
2.5 Создание таблицы rasrabotka.....	16
2.6 Создание таблицы testirovanie.....	16
3     Изменение таблиц.....	16
3.1 Переименование таблицы.....	16
3.2 Добавление столбцов .....	16
3.3 Удаление столбцов.....	16
3.4 Переименование столбцов.....	16
3.5 Изменение типов данных столбцов .....	17
3.5.1 Расширение и сужение столбцов .....	17
3.5.2 Изменение типов данных.....	17
3.6 Работа с ограничениями .....	17
3.6.1 Изменение обязательности.....	17
3.6.2 Добавление первичного ключа .....	17
3.6.3 Добавление внешнего ключа.....	17
3.6.4 Работа с другими ограничениями.....	17
4     Удаление таблиц .....	18
5     Заполнение таблиц .....	18
6     Обновление данных в таблице .....	20
7     Удаление данных из таблиц.....	20

Лабораторная работа №3. Основы выборки данных .....	21
1    Начало работы.....	21
2    Выборка из одной таблицы .....	21
2.1 Выборка данных из одной таблицы.....	21
2.2 Условная выборка из одной таблицы .....	22
2.2.1 Операторы сравнения.....	22
2.2.2 Логические операторы .....	24
2.2.3 Использование предикатов сравнения .....	25
2.2.4 IN.....	27
2.2.5 Поиск по шаблону .....	27
3    Сортировка.....	28
3.1 Сортировка по одному столбцу.....	28
3.2 Сортировка по нескольким столбцам.....	29
4    Агрегатные функции .....	29
5    Группировка .....	31
5.1 Общая информация .....	31
5.2 Группировка с having .....	33
6    Выборка из нескольких таблиц.....	34
6.1 Соединение с помощью FROM и WHERE .....	34
6.2 Соединение с помощью JOIN .....	35
7    Знакомство со встроенными функциями .....	44
8    Дополнительные возможности .....	51
9    Вложенные запросы .....	52
9.1 Общая информация .....	52
9.2 Функция EXISTS .....	55
9.3 Функция ALL .....	56
10   Написание запросов по вариантам .....	57
Лабораторная работа №4. Программируемые объекты базы данных.....	59
1    Начало работы.....	59
2    Дополнительные возможности SELECT-запросов.....	59
2.1 Заполнение таблиц из выборки.....	59
2.2 Объединение запросов .....	59
3    Представления .....	62
4    Общие табличные выражения.....	64

4.1	Общая информация .....	64
4.2	Иерархические запросы.....	65
5	Введение в PL/PGSQL .....	66
5.1	Общая информация .....	66
5.2	Функции и процедуры .....	67
5.3	Триггеры .....	70

## Вариант 11

### Лабораторная работа №1. Проектирование структуры базы данных «Программы»

Базу данных использует для работы коллектив разработчиков программного обеспечения. В таблицы должны быть занесены Ф.И.О., дата рождения, паспортные данные, адрес каждого программиста и его контактные телефоны. Разработанное ПО тестируется пользователями, данные о которых тоже заносятся в базу, как и сведения о разработанном ПО.

#### 1 Анализ предметной области

- База данных должна содержать:
  - ✓ данные о программах;
  - ✓ данные о пользователях;
  - ✓ данные о программистах.
- В соответствии с предметной областью система строится с учётом следующих особенностей:
  - ✓ в разработке отдельного программного продукта может участвовать только один из программистов коллектива;
  - ✓ все разрабатываемые коллективом программы относятся к двум типам: системное программное обеспечение и прикладное программное обеспечение;
  - ✓ каждую программу тестирует отдельный пользователь;
  - ✓ ведется учет системных требований для работы программного обеспечения;
  - ✓ ведется учет системных характеристик компьютеров пользователей;
  - ✓ фиксируются Ф.И.О. программиста, дата его рождения, паспортные данные, домашний адрес и контактные телефоны;
  - ✓ в базу вносятся Ф.И.О. пользователя, дата его рождения, паспортные данные, адрес и контактный телефон.
- Выделим базовые сущности предметной области:
  - **Программы.** Атрибуты программы:
    - ✓ данные разработчика
    - ✓ тип программного обеспечения
    - ✓ системные требования
    - ✓ дата разработки
    - ✓ объем
  - **Пользователи.** Атрибуты пользователей:
    - ✓ Ф.И.О.
    - ✓ дата рождения
    - ✓ паспортные данные

- ✓ адрес
- ✓ телефон
- ✓ данные о системе

○ **Программисты.** Атрибуты программистов:

- ✓ Ф.И.О.
- ✓ дата рождения
- ✓ паспортные данные
- ✓ домашний адрес
- ✓ телефоны

○ **Функции программиста:**

- ✓ разработка программного обеспечения

○ **Функции пользователя:**

- ✓ тестирование программного обеспечения

● **Выводы**

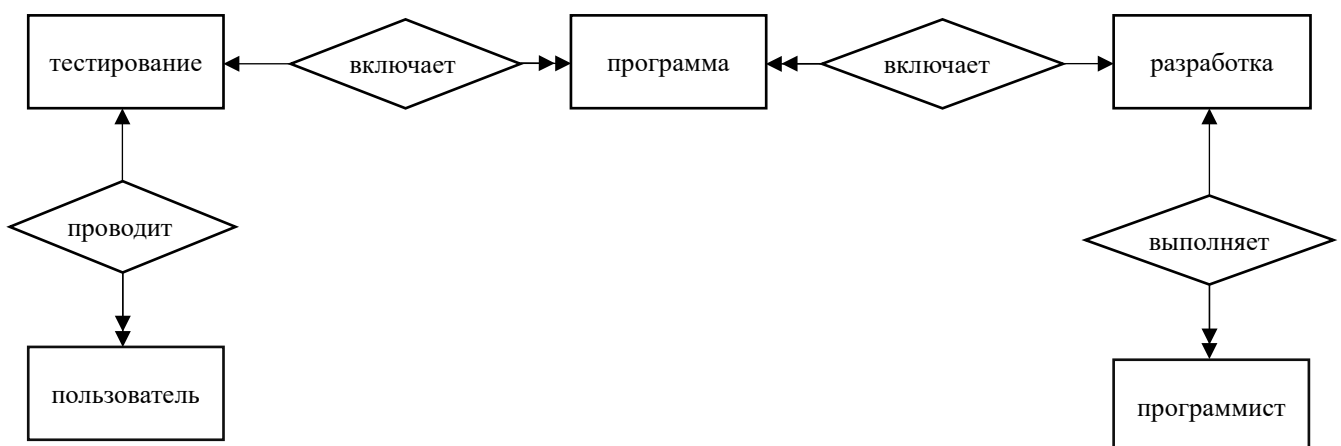
Система должна обеспечивать:

- ✓ поиск программ по любым из их характеристик;
- ✓ поиск программ, разработанных определенным программистом;
- ✓ поиск программ, протестированных определенным пользователем;
- ✓ поиск данных программиста по сведениям о программе;
- ✓ поиск данных пользователя по сведениям о программе;
- ✓ сопоставление данных о программе с характеристиками компьютера пользователя
- ✓ аудит вносимых в базу изменений

## 2 Разработка концептуальной модели

Концептуальное проектирование — начальный этап проектирования, при котором создается черновой вариант продукта, показывающий принцип закладываемой логики для первоначального согласования и проверки ограничений средств разработки.

Проектирование концептуальной модели с построением ER-диаграммы выполним с применением нотации Питера Чена:



Добавлением сущностей «Тестирование» и «Разработка» избавляемся от связей «многое-ко-многим».

Модель спроектирована с учетом построения степеней связи между сущностями по следующим закономерностям:

- ✓ «Программист» – «Разработка» – отношение «один-ко-многим»: в отдельной разработке принимает участие только один программист, разработок же может быть сколь угодно много;
- ✓ «Пользователь» – «Тестирование» – отношение «один-ко-многим»: в отдельном тестировании принимает участие только один пользователь, тестирований же может быть сколь угодно много;
- ✓ «Разработка» – «Программы» – отношение «один-ко-многим»: в процессе разработки производится множество программ, при этом каждая программа является результатом отдельной разработки;
- ✓ «Тестирование» – «Программы» – отношение «один-ко-многим»: процедура тестирования может проводиться со многими программами, но каждая программа тестируется отдельно;

### **3 Разработка логической модели**

Логическое проектирование — создание схемы базы данных на основе конкретной (в нашем случае – реляционной) модели данных.

Цель этапа логического проектирования – преобразование концептуальной модели на основе выбранной модели данных в логическую модель, не зависимую от особенностей используемой в дальнейшем СУБД для физической реализации базы данных.

- Первая нормальная форма

Таблица находится в 1НФ, если она удовлетворяет следующим требованиям:

- ✓ не содержит полей с несколькими значениями;
- ✓ ключевое поле не имеет пустот.

По определению логическая модель является отображением концептуальной модели. В данном случае исходными данными будут являться пять таблиц, выделенных при построении концептуальной модели. Проанализируем их в соответствии с требованиями.

programmist	
	FIO_programmista
	data_rozhdeniya
	pasportnye_dannye_pr
	domashnij_adres
	telefony_pr

polzovatel	
	FIO_polzovatelya
	data_rozhdeniya
	pasportnye_dannye_pol
	adres
	telefon_pol
	sistemnye_dannye

programma	
	programmist
	tip_programmy
	sistemnye_trebovania
	data_rasrabotki
	ob'em

rasrabotka	
	nomer_razrabotki
	data_razrabotki
	spisok_program
	programmist

testirovanie	
	nomer_testirovania
	data_testirovania
	spisok_program
	polzovatel

В составленных таблицах не все поля удовлетворяют требованиям 1НФ.

В частности, поле «pasportnye\_dannye\_pr» в таблице «programmist» содержит несколько значений – данные о серии и номере, дате выдачи документа, а также о наименовании государственного органа, выдавшего документ. С целью поддержания целостности данных о программистах целесообразно разбить данное поле на три поля: «nomer\_pasp\_program», «data\_pasp\_program» и «mest\_pasp\_program». Аналогично поступим и с полем «pasportnye\_dannye\_pol» таблицы «polzovatel»: поле «pasportnye\_dannye\_pol» разобьем на «nomer\_pasp\_polzov», «data\_pasp\_polzov» и «mest\_pasp\_polzov».

Поле «telephon\_pr» в таблице «programmist» содержит информацию о домашнем и контактном телефонах участника коллектива программистов. Поэтому делим его на «telefon\_dom» и «telefon\_con».

По этим же соображениям целесообразно поля «sistemnye\_dannye» и «sistemnye\_trebovania» в таблицах «polzovatel» и «programma» заменить на «chastota\_pr\_pol», «op\_pamyat\_pol» и «mesto\_disk\_pol» для таблицы «polzovatel» и «chastota\_pr\_prog», «op\_pamyat\_prog» и «mesto\_disk\_prog» для таблицы «programma».

В таблице «programma» поле «tip\_programmy» может принимать только два фиксированных значения, поэтому выделяем под него отдельную таблицу.

В результате получаем список атомарных полей, т.е. выполняем первое требование 1НФ.

Для обеспечения второго требования необходимо определить первичные ключи в каждой таблице:

✓ для таблиц «programmist», «polzovatel» и «programma» в качестве первичного

ключа введем семантически незначащие поля «id\_programmista», «id\_polzovatel» и «id\_programmy» соответственно;

- ✓ первичный ключ в таблице «rasrabotka» – поле «nomer\_razrabotki», которое однозначно определяет остальные не ключевые поля данной таблицы;
- ✓ тогда в таблице «testirovanie» первичный ключ – «nomer\_testirovania»;
- ✓ в качестве первичного ключа таблицы «tip\_programmy» введем семантически незначащее поле «id\_tip».

Поскольку ключевые поля всех таблиц не имеют пустот, то процесс приведения таблиц к 1НФ считаем завершенным.

programmist	
	id_programmista
	FIO_programmista
	data_rozhdeniya
	nomer_pasp_program
	data_pasp_program
	mest_pasp_program
	domashnij_adres
	telefon_dom
	telefon_con

polzovatel	
	id_polzovatel
	FIO_polzovatelya
	data_rozhdeniya
	nomer_pasp_polzov
	data_pasp_polzov
	mest_pasp_polzov
	adres
	telefon_pol
	chastota_pr_pol
	op_pamyat_pol
	mesto_disk_pol

programma	
	id_programmy
	id_programmista
	tip_programmy
	chastota_pr_prog
	op_pamyat_prog
	mesto_disk_prog
	data_razrabotki
	ob'em

rasrabotka	
	nomer_razrabotki
	data_razrabotki
	id_programmy
	id_programmista

testirovanie	
	nomer_testirovania
	data_testirovania
	id_programmy
	id_polzovatel

tip_programmy	
	id_tip
	tip_programmy

## • Вторая нормальная форма

Таблица находится во 2НФ, если она удовлетворяет следующим требованиям:

- ✓ таблица должна быть приведена к 1НФ
- ✓ поля, которые зависят только от части первичного ключа должны быть выделены в состав отдельных таблиц
- ✓ все таблицы должны быть связаны между собой

Все таблицы приведены к 1НФ, то есть первое из требований 2НФ выполнено.

Для обеспечения второго требования проанализируем каждую из полученных таблиц на предмет зависимостей не ключевых полей таблицы от части первичного ключа и сделаем заключение, что ключевые поля всех таблиц являются простыми и однозначно определяют детальные не ключевые поля

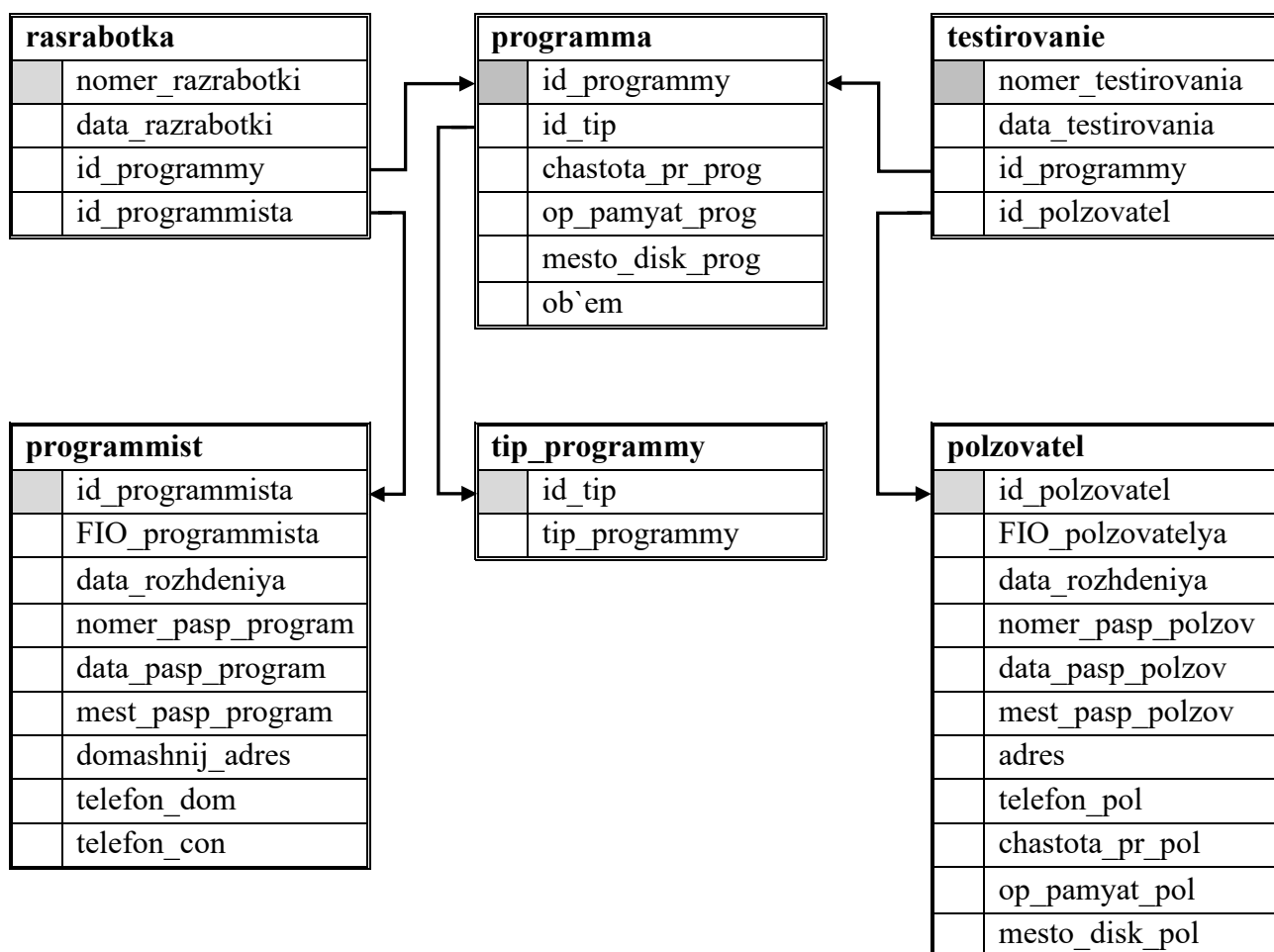


соответствующих таблиц. Второе требование 2НФ выполнено, перейдем к выполнению третьего требования.

Заметим, что таблицы «programma» и «rasrabotka» содержат одни и те же поля «id\_programmista» и «data\_rasrabotki», в результате чего появляется избыточность, а это противоречит основной цели нормализации – устранению избыточности данных. Так как в разработке программного продукта участвует только один программист и это учтено в таблице «rasrabotka», удаляем дублирующие поля из таблицы «programma».

Кроме того, заменяем ссылку «spisok\_program» из таблиц «rasrabotka» и «testirovanie» на ссылку «id\_programmy» и ссылку «tip\_programmy» в таблице «programma» на «id\_tip».

Устанавливаем соответствующие связи «один-ко-многим» и на этом процесс приведения таблиц ко 2НФ считаем завершенным.



- Третья нормальная форма

Таблица находится в 3НФ, если она удовлетворяет следующим требованиям:

- ✓ таблицы должны быть приведены ко 2НФ;

✓ не должно быть транзитивных зависимостей между не ключевыми полями;  
В данном примере таблицы приведены к первой и второй нормальным формам, то есть одно из требований выполнено. Перейдем к выполнению второго требования.

Транзитивная зависимость имеет место, если какое-либо не ключевое поле функционально зависит от другого не ключевого поля, а тот в свою очередь функционально зависит от ключа.

В нашем случае, поскольку в таблицах не находится транзитивных зависимостей, процесс приведения таблиц к 3НФ считаем завершенным.

Таким образом, мы получаем корректную, нормализованную структуру базы данных.

## 4 Выбор СУБД

Сегодня существует большое количество СУБД, которые используются практически везде. Такой выбор объясняется тем, что разные системы управления базами данных применяются в различных областях человеческой деятельности и предназначены для разных задач. Существуют как крупные СУБД, которые предназначены для промышленного использования, так и мелкие, для небольших компаний, которым нет необходимости покупать дорогостоящие СУБД для использования в мелких коммерческих проектах.

PostgreSQL — свободная объектно-реляционная система управления базами данных. Является свободной альтернативой коммерческим СУБД (таким как Oracle Database, Microsoft SQL Server, IBM DB2, Informix и СУБД производства Sybase) вместе с другими свободными СУБД (такими как MySQL и Firebird).

PostgreSQL базируется на языке SQL и поддерживает многие из возможностей стандарта SQL 2003 (ISO/IEC 9075). Соответствует стандартам ANSI SQL-92 и SQL-99. Сильными сторонами PostgreSQL считаются: поддержка БД практически неограниченного размера; надёжные механизмы транзакций и репликации; наследование; легкая расширяемость. Реализация дополнительной логики, например, условных переходов и циклов, выходит за рамки SQL и требует использования некоторых языковых расширений. Функции могут писаться с использованием одного из следующих языков:

- ✓ встроенный процедурный язык PL/pgSQL, во многом аналогичный языку PL/SQL, используемому в СУБД Oracle;
- ✓ скриптовые языки — PL/Lua, PL/LOLCODE, PL/Perl, plPHP, PL/Python, PL/Ruby, PL/sh, PL/Tcl и PL/Scheme;
- ✓ классические языки — C, C++, Java (через модуль PL/Java);
- ✓ статистический язык R (через модуль PL/R).

PostgreSQL допускает использование функций, возвращающих набор записей, который далее можно использовать так же, как и результат выполнения обычного запроса.

Исходя из комплекса поставленных задач целесообразно остановить выбор на СУБД PostgreSQL.

## 5 Физическое проектирование базы данных

**Таблица «programmist»**

Столбец (поле)	Тип данных	Обязательность	Прочие ограничения
id_programmista	INT		PK
FIO_programmista	varchar(50)	NOT NULL	
data_rozhdeniya	DATE	NOT NULL	
nomer_pasp_program	char(11)	NOT NULL	
data_pasp_program	DATE	NOT NULL	
mest_pasp_program	varchar(50)	NOT NULL	
domashnij_adres	varchar(50)	NOT NULL	
telefon_dom	smallint	NOT NULL	
telefon_con	smallint	NOT NULL	

**Таблица «polzovatel»**

Столбец (поле)	Тип данных	Обязательность	Прочие ограничения
id_polzovatel	INT		PK
FIO_polzovatelya	varchar(50)	NOT NULL	
data_rozhdeniya	DATE	NOT NULL	
nomer_pasp_polzov	char(11)	NOT NULL	
data_pasp_polzov	DATE	NOT NULL	
mest_pasp_polzov	varchar(50)	NOT NULL	
adres	varchar(50)	NOT NULL	
telefon_pol	smallint	NOT NULL	
chastota_pr_pol	smallint	NOT NULL	
op_pamyat_pol	smallint	NOT NULL	
mesto_disk_pol	smallint	NOT NULL	

**Таблица «programma»**

Столбец (поле)	Тип данных	Обязательность	Прочие ограничения
id_programmy	INT		PK
id_tip	smallint	NOT NULL	FK
chastota_pr_prog	smallint	NOT NULL	
op_pamyat_prog	smallint	NOT NULL	
mesto_disk_prog	smallint	NOT NULL	
ob'em	smallint	NOT NULL	

**Таблица «rasrabotka»**

Столбец (поле)	Тип данных	Обязательность	Прочие ограничения
nomer razrabotki	INT		PK
data razrabotki	DATE	NOT NULL	
id_programmy	smallint	NOT NULL	FK
id_programmista	smallint	NOT NULL	FK

**Таблица «testirovanie»**

Столбец (поле)	Тип данных	Обязательность	Прочие ограничения
nomer testirovania	INT		PK
data testirovania	DATE	NOT NULL	
id_programmy	smallint	NOT NULL	FK
id_polzovatel	smallint	NOT NULL	FK

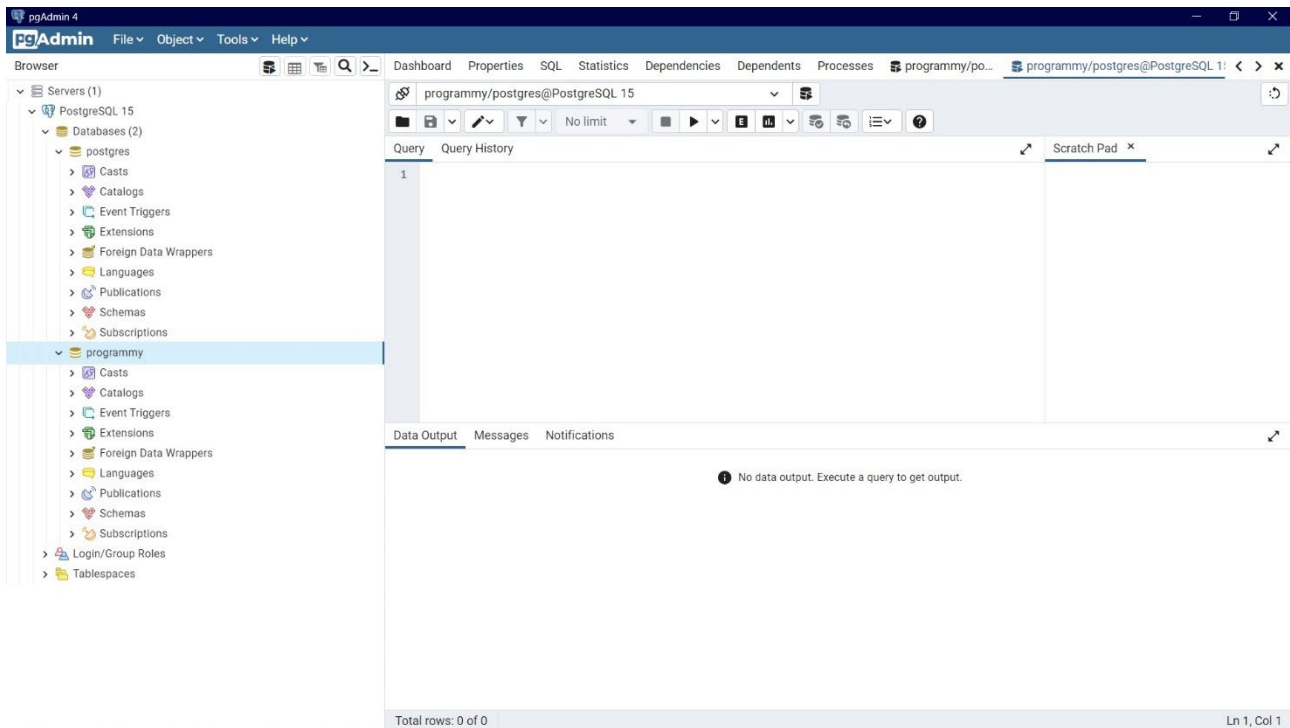
**Таблица «tip\_programmy»**

Столбец (поле)	Тип данных	Обязательность	Прочие ограничения
id_tip	INT		PK
tip_programmy	varchar(20)	NOT NULL	FK

# Лабораторная работа №2. Создание базы данных в СУБД PostgreSQL

## 1 Начало работы

Для создания базы данных установлен и настроен PostgreSQL Version 6.15. Работа ведется в клиенте pgAdmin. Создаем базу данных programmy:



## 2 Создание таблиц

Создавать таблицы можно двумя способами:

### ✓ Способ 1:

- сначала создать независимые таблицы. То есть те, в которых нет внешних ключей;
- создать таблицы, в которых есть внешние ключи для связи с уже созданными таблицами;
- создать оставшиеся таблицы.

### ✓ Способ 2:

- создать все таблицы, но не указывать, какие столбцы являются внешними ключами;
- указать внешние ключи.

### 2.1 Создание таблицы tip\_programmy

```
CREATE TABLE tip_programmy(  
id_tip INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
```

```
tip_programmy varchar(20) NOT NULL  
);
```

## **2.2 Создание таблицы programmer**

```
CREATE TABLE programmer(  
id_programmista INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,  
FIO_programmista VARCHAR(50) NOT NULL,  
data_rozhdeniya DATE NOT NULL,  
nomer_pasp_program CHAR(11) NOT NULL,  
data_pasp_program DATE,  
mest_pasp_program VARCHAR(50) NOT NULL,  
domashnij_adres VARCHAR(50) NOT NULL,  
telefon_dom SMALLINT,  
telefon_con SMALLINT  
);
```

## **2.3 Создание таблицы polzovatel**

```
CREATE TABLE polzovatel(  
id_polzovatel INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,  
FIO_polzovatelya varchar(50) NOT NULL,  
data_rozhdeniya DATE NOT NULL,  
nomer_pasp_polzov char(11) NOT NULL,  
data_pasp_polzov DATE NOT NULL,  
mest_pasp_polzov varchar(50) NOT NULL,  
adres varchar(50) NOT NULL,  
telefon_pol smallint NOT NULL,  
chastota_pr_pol smallint NOT NULL,  
op_pamyat_pol smallint NOT NULL,  
mesto_disk_pol smallint NOT NULL  
);
```

## **2.4 Создание таблицы programma**

```
CREATE TABLE programma(  
id_programmy INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,  
id_tip smallint NOT NULL,  
chastota_pr_prog smallint NOT NULL,  
op_pamyat_prog smallint NOT NULL,  
mesto_disk_prog smallint NOT NULL,  
obem smallint NOT NULL,  
FOREIGN KEY (id_tip) REFERENCES tip_programmy(id_tip)  
);
```

## **2.5 Создание таблицы rasrabotka**

```
CREATE TABLE rasrabotka(  
nomer_razrabotki INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,  
data_razrabotki DATE NOT NULL,  
id_programmy smallint NOT NULL,  
id_programmista smallint NOT NULL,  
FOREIGN KEY (id_programmy) REFERENCES programma(id_programmy),  
FOREIGN KEY (id_programmista) REFERENCES programmist(id_programmista)  
);
```

## **2.6 Создание таблицы testirovanie**

```
CREATE TABLE testirovanie(  
nomer_testirovania INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,  
  
data_testirovania DATE NOT NULL,  
id_programmy smallint NOT NULL,  
id_polzovatel smallint NOT NULL,  
FOREIGN KEY (id_programmy) REFERENCES programma(id_programmy),  
FOREIGN KEY (id_polzovatel) REFERENCES polzovatel(id_polzovatel)  
);
```

# **3 Изменение таблиц**

## **3.1 Переименование таблицы**

```
ALTER TABLE programmist RENAME TO programmers
```

## **3.2 Добавление столбцов**

```
ALTER TABLE testirovanie ADD COLUMN adres_test VARCHAR (50);
```

## **3.3 Удаление столбцов**

```
ALTER TABLE testirovanie DROP COLUMN adres_test;
```

## **3.4 Переименование столбцов**

```
ALTER TABLE polzovatel RENAME COLUMN adres TO adress_pol;
```

### **3.5 Изменение типов данных столбцов**

#### **3.5.1 Расширение и сужение столбцов**

```
ALTER TABLE polzovatel ALTER COLUMN adress_pol TYPE VARCHAR (255);
```

#### **3.5.2 Изменение типов данных**

```
ALTER TABLE polzovatel ALTER COLUMN adress_pol TYPE CHAR (255);  
ALTER TABLE polzovatel ALTER COLUMN adress_pol TYPE VARCHAR (255);
```

### **3.6 Работа с ограничениями**

```
CREATE TABLE test2(  
test_id INT,  
test_name VARCHAR(30),  
test_description VARCHAR(255) NOT NULL,  
author_id INT NOT NULL,  
raiting FLOAT NOT NULL);
```

```
CREATE TABLE author(  
autor_id INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,  
autor_name VARCHAR(30) NOT NULL);
```

#### **3.6.1 Изменение обязательности**

```
ALTER TABLE test2 ALTER COLUMN test_name SET NOT NULL;
```

```
ALTER TABLE test2 ALTER COLUMN test_description DROP NOT NULL;
```

#### **3.6.2 Добавление первичного ключа**

```
ALTER TABLE test2 ADD PRIMARY KEY (test_id);
```

#### **3.6.3 Добавление внешнего ключа**

```
ALTER TABLE test2 ADD CONSTRAINT autor_id_fk FOREIGN KEY (author_id)  
REFERENCES author(author_id);
```

#### **3.6.4 Работа с другими ограничениями**

```
ALTER TABLE test2 ALTER COLUMN raiting SET DEFAULT 0;  
ALTER TABLE test2 ADD CONSTRAINT raiting_check CHECK (raiting >=0);  
ALTER TABLE test2 ALTER COLUMN raiting DROP DEFAULT;
```



## 4 Удаление таблиц

Порядок удаления таблиц:

- ✓ удаляем таблицы с внешними ключами, от которых не зависят никакие другие таблицы;
- ✓ удаляем таблицы без внешних ключей.

```
DROP TABLE test2;  
DROP TABLE author;
```

## 5 Заполнение таблиц

Порядок заполнения таблиц:

- ✓ заполняем независимые таблицы;
- ✓ заполняем остальные таблицы.

```
SELECT * FROM tip_programmy;  
ALTER TABLE tip_programmy ALTER COLUMN tip_programmy TYPE  
VARCHAR (35);  
INSERT INTO tip_programmy(tip_programmy) VALUES('Системное  
программное обеспечение');  
INSERT INTO tip_programmy(tip_programmy) VALUES('Прикладное  
программное обеспечение');
```

```
SELECT * FROM programmist;  
ALTER TABLE programmists RENAME TO programmist  
ALTER TABLE programmist ALTER COLUMN telefon_dom TYPE CHAR (11);  
ALTER TABLE programmist ALTER COLUMN telefon_con TYPE CHAR (11);  
DELETE FROM programmist;  
INSERT INTO programmist(FIO_programmista, data_rozhdeniya,  
nomer_pasp_program, data_pasp_program, mest_pasp_program, domashnij_adres,  
telefon_dom, telefon_con) VALUES('Иванов Иван Иванович', '01.01.2001', '1234  
123456', '01.05.2015', 'РОВД г. Москва', 'г. Москва, ул. Центральная, д. 1, кв. 11',  
'79101112233', '79202221133');  
INSERT INTO programmist(FIO_programmista, data_rozhdeniya,  
nomer_pasp_program, data_pasp_program, mest_pasp_program, domashnij_adres,  
telefon_dom, telefon_con) VALUES('Петров Иван Семенович', '11.11.2003', '1234  
123456', '01.07.2017', 'РОВД г. Химки', 'г. Химки, ул. Луговая, д. 12, кв. 22',  
'79151114433', '79201231133');
```

```
SELECT * FROM polzovatel;  
ALTER TABLE polzovatel ALTER COLUMN telefon_pol TYPE CHAR (11);  
ALTER TABLE polzovatel ALTER COLUMN chastota_pr_pol TYPE CHAR (3);  
ALTER TABLE polzovatel ALTER COLUMN op_pamyat_pol TYPE CHAR (2);  
ALTER TABLE polzovatel ALTER COLUMN mesto_disk_pol TYPE CHAR (4);
```

```
INSERT INTO polzovatel(FIO_polzovatelya, data_rozhdeniya, nomer_pasp_polzov,
data_pasp_polzov, mest_pasp_polzov, adress_pol, telefon_pol, chastota_pr_pol,
op_pamyat_pol, mesto_disk_pol) VALUES ('Семенов Сергей Борисович',
'12.05.1995', '4512 785612', '05.12.2010', 'РОВД г. Ярославль', 'г. Ярославль, пр-т
Толбухина, д. 18, кв. 14', '79994567799', '2,5', '8', '512');
INSERT INTO polzovatel(FIO_polzovatelya, data_rozhdeniya, nomer_pasp_polzov,
data_pasp_polzov, mest_pasp_polzov, adress_pol, telefon_pol, chastota_pr_pol,
op_pamyat_pol, mesto_disk_pol) VALUES ('Сидоров Алексей Андреевич',
'05.12.1990', '8912 212165', '20.01.2006', 'РОВД г. Ярославль', 'г. Ярославль, пр-т
Ленина, д. 5, кв. 20', '79994441188', '2,3', '16', '1024');
```

```
SELECT * FROM programma;
ALTER TABLE programma ADD CONSTRAINT id_tip_fk FOREIGN KEY (id_tip)
REFERENCES tip_programmy(id_tip);
ALTER TABLE programma ALTER COLUMN chastota_pr_prog TYPE CHAR (3);
ALTER TABLE programma ALTER COLUMN op_pamyat_prog TYPE CHAR (2);
ALTER TABLE programma ALTER COLUMN mesto_disk_prog TYPE CHAR (4);
ALTER TABLE programma ALTER COLUMN obem TYPE CHAR (4);
ALTER TABLE programma ADD COLUMN imya VARCHAR (20);
DELETE FROM programma;
INSERT INTO programma(id_tip, chastota_pr_prog, op_pamyat_prog,
mesto_disk_prog, obem, imya)
VALUES
(1, '2,1', '4', '2048', '256', 'Система'),
(2, '1,4', '2', '512', '128', 'Редактор'),
(1, '1,8', '8', '1024', '512', 'Оболочка'),
(2, '2,2', '16', '512', '128', 'Игра'),
(2, '1,4', '2', '1024', '64', 'Поиск'),
(2, '1,7', '2', '512', '128', 'Органайзер');
```

```
SELECT * FROM rasrabotka;
INSERT INTO rasrabotka(data_razrabotki, id_programmy, id_programmista)
VALUES('01.02.2023', 3, 6);
INSERT INTO rasrabotka(data_razrabotki, id_programmy, id_programmista)
VALUES('14.05.2021', 4, 5);
INSERT INTO rasrabotka(data_razrabotki, id_programmy, id_programmista)
VALUES('01.02.2020', 5, 5);
INSERT INTO rasrabotka(data_razrabotki, id_programmy, id_programmista)
VALUES('14.05.2021', 6, 6);
INSERT INTO rasrabotka(data_razrabotki, id_programmy, id_programmista)
VALUES('24.02.2019', 7, 6);
INSERT INTO rasrabotka(data_razrabotki, id_programmy, id_programmista)
VALUES('08.11.2020', 8, 5);
```

```
SELECT * FROM testirovanie;  
INSERT INTO testirovanie(data_testirovania , id_programmy, id_polzovatel)  
VALUES('01.02.2023', 3, 6);  
INSERT INTO testirovanie(data_testirovania , id_programmy, id_polzovatel)  
VALUES('15.03.2022', 8, 5);
```

## **6 Обновление данных в таблице**

```
UPDATE testirovanie SET id_programmy = 4 WHERE nomer_testirovania = 1;
```

## **7 Удаление данных из таблиц**

```
DELETE FROM product_types WHERE type_id = 4;
```

# Лабораторная работа №3. Основы выборки данных

## 1 Начало работы

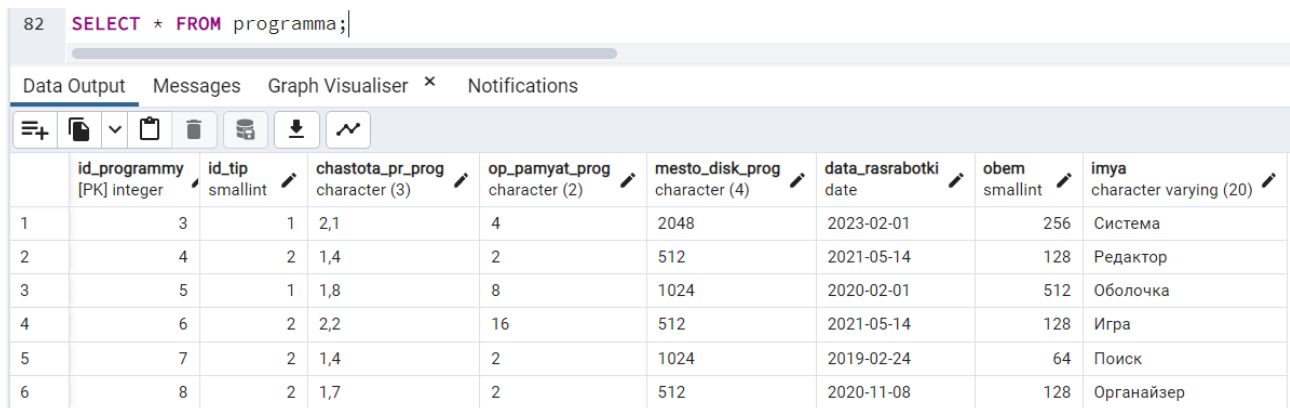
Проведено знакомство со схемой БД, для которой написаны запросы-примеры.

## 2 Выборка из одной таблицы

### 2.1 Выборка данных из одной таблицы

✓ (1) Выборка всех данных из таблицы «programma»:

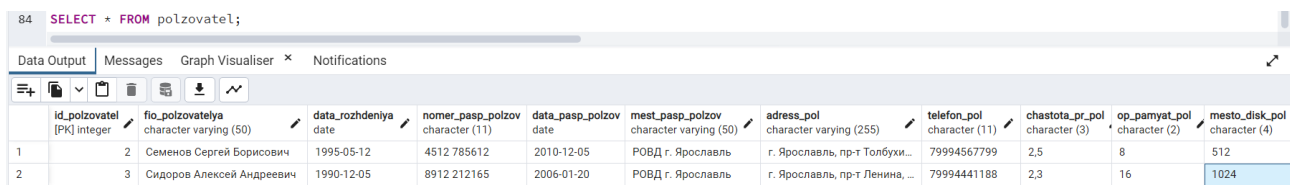
```
SELECT * FROM programma;
```



	id_programmy [PK] integer	id_tip smallint	chastota_pr_prog character (3)	op_pamyat_prog character (2)	mesto_disk_prog character (4)	data_rasrabotki date	obem smallint	imya character varying (20)
1	3	1	2,1	4	2048	2023-02-01	256	Система
2	4	2	1,4	2	512	2021-05-14	128	Редактор
3	5	1	1,8	8	1024	2020-02-01	512	Оболочка
4	6	2	2,2	16	512	2021-05-14	128	Игра
5	7	2	1,4	2	1024	2019-02-24	64	Поиск
6	8	2	1,7	2	512	2020-11-08	128	Органайзер

✓ (2) Выборка всех данных из таблицы «polzovatel»:

```
SELECT * FROM polzovatel;
```



	id_polzovatel [PK] integer	fio_polzovatelya character varying (50)	data_rozhdeniya date	nomer_pasp_polzov character (11)	data_pasp_polzov date	mest_pasp_polzov character varying (50)	adres_pol character varying (255)	telefon_pol character (11)	chastota_pr_pol character (3)	op_pamyat_pol character (2)	mesto_disk_pol character (4)
1	2	Семенов Сергей Борисович	1995-05-12	4512 785612	2010-12-05	РОВД г. Ярославль	г. Ярославль, пр-т Толбухи...	79994567799	2,5	8	512
2	3	Сидоров Алексей Андреевич	1990-12-05	8912 212165	2006-01-20	РОВД г. Ярославль	г. Ярославль, пр-т Ленина, ...	79994441188	2,3	16	1024

✓ (3) Выборка данных столбцов «id\_programmy» и «data\_rasrabotki» из таблицы «rasrabotka»:

```
SELECT id_programmy, data_rasrabotki FROM rasrabotka;
```

```
381 SELECT id_programmy, data_razrabotki FROM rasrabotka;
```

Data Output Messages Graph Visualiser × Notifications

	id_programmy smallint	data_razrabotki date
1	6	[null]
2	3	2023-02-01
3	4	2021-05-14
4	5	2020-02-01
5	7	2019-02-24
6	8	2020-11-08

- ✓ (4) Выборка данных столбцов «fio\_polzovatelya» и «nomer\_pasp\_polzov» из таблицы «polzovatel»:

```
SELECT fio_polzovatelya, nomer_pasp_polzov FROM polzovatel;
```

```
85 SELECT fio_polzovatelya, nomer_pasp_polzov FROM polzovatel;
```

Data Output Messages Graph Visualiser × Notifications

	fio_polzovatelya character varying (50)	nomer_pasp_polzov character (11)
1	Семенов Сергей Борисович	4512 785612
2	Сидоров Алексей Андреевич	8912 212165

- ✓ (5) Выборка с заменой названий столбцов на псевдонимы:

```
SELECT fio_polzovatelya "ФИО пользователя", nomer_pasp_polzov "Номер  
паспорта" FROM polzovatel;
```

```
86 SELECT fio_polzovatelya "ФИО пользователя", nomer_pasp_polzov "Номер паспорта" FROM polzovatel;
```

Data Output Messages Graph Visualiser × Notifications

	ФИО пользователя character varying (50)	Номер паспорта character (11)
1	Семенов Сергей Борисович	4512 785612
2	Сидоров Алексей Андреевич	8912 212165

## 2.2 Условная выборка из одной таблицы

### 2.2.1 Операторы сравнения

- ✓ (6) Выборка из таблицы «programma» программ с требуемым местом на диске равным 512 мегабайт:

```
SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem
FROM programma
WHERE mesto_disk_prog = 512;
```

```
381 SELECT id_programmy, data_razrabotki FROM rasrabotka;
382
383 SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem
384 FROM programma
385 WHERE mesto_disk_prog = 512;
```

	imya character varying (20)	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint	obem smallint
1	Игра	2,2	16	512	128
2	Органайзер	1,7	5	512	128
3	Редактор	1,4	3	512	128

- ✓ (7) Выборка из таблицы «programma» программ с требуемым местом на диске больше 512 мегабайт:

```
SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem
FROM programma
WHERE mesto_disk_prog > 512;
```

```
381 SELECT id_programmy, data_razrabotki FROM rasrabotka;
382
383 SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem
384 FROM programma
385 WHERE mesto_disk_prog > 512;
```

	imya character varying (20)	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint	obem smallint
1	Система	2,1	4	2048	256
2	Навигатор	1,9	8	1024	128
3	Поиск	1,4	2	1024	64
4	Оператор	2,7	32	4096	256

- ✓ (8) Выборка из таблицы «programma» программ с требуемой оперативной памятью не равной 2 гигабайта:

```
SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem
FROM programma
WHERE op_pamyat_prog != 2;
```

```

71 SELECT * FROM programma;
72 SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, data_rasrabotki, obem
73 FROM programma
74 WHERE op_pamyat_prog != 2;

```

Data Output Messages Notifications

	imya character varying (20)	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint	data_rasrabotki date	obem smallint
1	Система	2,1	4	2048	2023-02-01	256
2	Оболочка	1,8	8	1024	2020-02-01	512
3	Игра	2,2	16	512	2021-05-14	128

## 2.2.2 Логические операторы

- ✓ (9) Выборка из таблицы «programma» по двум заданным параметрам и условию «И»:

```

SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem
FROM programma
WHERE mesto_disk_prog > 512 AND op_pamyat_prog > 4;

```

```

383 SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem
384 FROM programma
385 WHERE mesto_disk_prog > 512 AND op_pamyat_prog > 4;
386

```

Data Output Messages Graph Visualiser × Notifications

	imya character varying (20)	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint	obem smallint
1	Навигатор	1,9	8	1024	128
2	Оператор	2,7	32	4096	256

- ✓ (10) Выборка из таблицы «programma» по двум заданным параметрам и условию «И»:

```

SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem
FROM programma
WHERE obem < 512 OR mesto_disk_prog < 128;

```

```

383 SELECT id_programmy, data_razrabotki FROM rasrabotka;
384
385 SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem
386 FROM programma
387 WHERE obem < 512 OR mesto_disk_prog < 128;
388

```

Data Output Messages Graph Visualiser × Notifications

	imya character varying (20)	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint	obem smallint
1	Игра	2,2	16	512	128
2	Система	2,1	4	2048	256
3	Навигатор	1,9	8	1024	128
4	Поиск	1,4	2	1024	64
5	Оператор	2,7	32	4096	256
6	Организер	1,7	5	512	128
7	Редактор	1,4	3	512	128

### 2.2.3 Использование предикатов сравнения

✓ (11) Выборка с предикатом BETWEEN:

```

SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem
FROM programma
WHERE op_pamyat_prog BETWEEN 4 AND 16 AND mesto_disk_prog > 512;

```

```

383 SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem
384 FROM programma
385 WHERE op_pamyat_prog BETWEEN 4 AND 16 AND mesto_disk_prog > 512;

```

Data Output Messages Graph Visualiser × Notifications

	imya character varying (20)	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint	obem smallint
1	Система	2,1	4	2048	256
2	Навигатор	1,9	8	1024	128

✓ (12) Выборка с предикатом NOT BETWEEN:

```

SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem
FROM programma
WHERE op_pamyat_prog NOT BETWEEN 4 AND 8 AND mesto_disk_prog <=
1024;

```



```

383 SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem
384 FROM programma
385 WHERE op_pamyat_prog NOT BETWEEN 4 AND 8 AND mesto_disk_prog <= 1024;

```

	imya character varying (20)	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint	obem smallint
1	Игра	2,2	16	512	128
2	Поиск	1,4	2	1024	64
3	Редактор	1,4	3	512	128

✓ (13) Выборка с предикатом IS NULL:

```

SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem
FROM programma
WHERE mesto_disk_prog IS NULL;

```

```

383 SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem
384 FROM programma
385 WHERE mesto_disk_prog IS NULL;

```

	imya character varying (20)	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint	obem smallint
1	Оболочка	1,8	8	[null]	512

✓ (14) Выборка с предикатом IS TRUE:

```

SELECT id_tip, tip_programmy, test
FROM tip_programmy
WHERE test IS TRUE;

```

```

87 SELECT id_tip, tip_programmy, test
88 FROM tip_programmy
89 WHERE test IS TRUE;

```

	id_tip [PK] integer	tip_programmy character varying (35)	test boolean
1	1	Системное программное обеспечение	true

## 2.2.4 IN

✓ (15) Выборка с использованием IN:

```
SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem,
status
FROM programma
WHERE op_pamyat_prog > 4 AND status IN('Checked');
```

```
383 SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem, status
384 FROM programma
385 WHERE op_pamyat_prog > 4 AND status IN('Checked');
```

	imya character varying (20)	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint	obem smallint	status character varying (10)
1	Игра	2,2	16	512	128	Checked
2	Навигатор	1,9	8	1024	128	Checked
3	Оболочка	1,8	8	[null]	512	Checked
4	Органайзер	1,7	5	512	128	Checked

✓ (16) Выборка с использованием NOT IN:

```
SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem, status
FROM programma
WHERE op_pamyat_prog < 16 AND status NOT IN('Checked');
```

```
383 SELECT imya, chastota_pr_prog, op_pamyat_prog, mesto_disk_prog, obem, status
384 FROM programma
385 WHERE op_pamyat_prog < 16 AND status NOT IN('Checked');
```

	imya character varying (20)	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint	obem smallint	status character varying (10)
1	Поиск	1,4	2	1024	64	Testing
2	Редактор	1,4	3	512	128	Testing
3	Структура	1,2	8	256	512	Testing

## 2.2.5 Поиск по шаблону

✓ (17) Поиск программы с именем, начинающимся на «С»:

```
SELECT * FROM programma WHERE imya LIKE 'C%';
```

96 `SELECT * FROM programma WHERE imya LIKE 'C%';`

	id_programmy [PK] integer	id_tip smallint	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint	data_rasrabotki date	obem smallint	imya character varying (20)	status character varying (10)
1	3	1	2,1	4	2048	2023-02-01	256	Система	Checked
2	11	1	1,2	2	256	2018-02-13	512	Структура	Testing

✓ (18) Поиск программы с именем, заканчивающимся на «р»:

`SELECT * FROM programma WHERE imya LIKE '%p';`

96 `SELECT * FROM programma WHERE imya LIKE '%p';`

	id_programmy [PK] integer	id_tip smallint	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint	data_rasrabotki date	obem smallint	imya character varying (20)	status character varying (10)
1	8	2	1,7	[null]	512	2020-11-08	128	Организер	Checked
2	10	2	1,9	8	1024	2021-05-14	128	Навигатор	Checked
3	9	1	2,7	32	4096	2023-07-01	256	Оператор	Testing
4	4	2	1,4	[null]	512	2021-05-14	128	Редактор	Testing

## 3 Сортировка

### 3.1 Сортировка по одному столбцу

✓ (19) Сортировка всех столбцов таблицы программа по названию программы в порядке возрастания:

`SELECT * FROM programma ORDER BY imya ASC;`

383 `SELECT * FROM programma ORDER BY imya ASC;`

	id_programmy [PK] integer	id_tip smallint	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint	obem smallint	imya character varying (20)	status character varying (10)
1	6	2	2,2	16	512	128	Игра	Checked
2	10	2	1,9	8	1024	128	Навигатор	Checked
3	5	1	1,8	8	[null]	512	Оболочка	Checked
4	9	1	2,7	32	4096	256	Оператор	Testing
5	8	2	1,7	5	512	128	Организер	Checked
6	7	2	1,4	2	1024	64	Поиск	Testing
7	4	2	1,4	3	512	128	Редактор	Testing
8	3	1	2,1	4	2048	256	Система	Checked
9	11	1	1,2	8	256	512	Структура	Testing

✓ (20) Сортировка всех столбцов таблицы программа по требуемому месту на диске по убыванию:

`SELECT * FROM programma ORDER BY mesto_disk_prog DESC;`

```
97 SELECT * FROM programma ORDER BY mesto_disk_prog DESC;
```

Data Output Messages Notifications										
	id_programmy [PK] integer	id_tip smallint	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint	data_rasrabotki date	obem smallint	imya character varying (20)	status character varying (10)	
1	9	1	2,7	32	4096	2023-07-01	256	Оператор	Testing	
2	3	1	2,1	4	2048	2023-02-01	256	Система	Checked	
3	7	2	1,4	2	1024	2019-02-24	64	Поиск	Testing	
4	5	1	1,8	8	1024	2020-02-01	512	Оболочка	Checked	
5	10	2	1,9	8	1024	2021-05-14	128	Навигатор	Checked	
6	4	2	1,4	[null]	512	2021-05-14	128	Редактор	Testing	
7	6	2	2,2	16	512	2021-05-14	128	Игра	Checked	
8	8	2	1,7	[null]	512	2020-11-08	128	Организер	Checked	
9	11	1	1,2	2	256	2018-02-13	512	Структура	Testing	

## 3.2 Сортировка по нескольким столбцам

✓ (21) Группировка по двум столбцам в порядке убывания:

```
SELECT * FROM programma ORDER BY mesto_disk_prog, obem ASC;
```

383 SELECT * FROM programma ORDER BY mesto_disk_prog, obem ASC;									
Data Output Messages Graph Visualiser × Notifications									
	id_programm [PK] integer	id_tip smallint	chastota_pr_ character (3)	op_pamyat_p smallint	mesto_disk_ smallint	obem smallint	imya character varyir	status character varying	
1	11	1	1,2	8	256	512	Структура	Testing	
2	6	2	2,2	16	512	128	Игра	Checked	
3	8	2	1,7	5	512	128	Организер	Checked	
4	4	2	1,4	3	512	128	Редактор	Testing	
5	7	2	1,4	2	1024	64	Поиск	Testing	
6	10	2	1,9	8	1024	128	Навигатор	Checked	
7	3	1	2,1	4	2048	256	Система	Checked	
8	9	1	2,7	32	4096	256	Оператор	Testing	
9	5	1	1,8	8	[null]	512	Оболочка	Checked	

## 4 Агрегатные функции

✓ (22) Подсчет количества разработанных программ:

```
SELECT COUNT (*) FROM programma;
```









98


SELECT COUNT (\*) FROM programma;

Data Output

Messages

Notifications



	count bigint	
1	9	

- ✓ (23) Определение минимального требуемого для установки программы места на диске:

SELECT MIN (mesto\_disk\_prog) FROM programma;

98


SELECT MIN (mesto\_disk\_prog) FROM programma;

Data Output


Messages


Notifications


≡+





▼












	min smallint 	
1	256	

- ✓ (24) Определение максимального требуемого для установки программы места на диске:

SELECT MAX (mesto\_disk\_prog) FROM programma;









98


SELECT MAX (mesto\_disk\_prog) FROM programma;

Data Output

Messages

Notifications



	max smallint 	
1	4096	

- ✓ (25) Определение среднего требуемого для установки программы места на диске:

SELECT AVG (mesto\_disk\_prog) FROM programma WHERE status  
IN('Checked');

98

SELECT AVG (mesto\_disk\_prog) FROM programma WHERE status IN('Checked');

Data Output

Messages

Notifications

≡+

▼

	avg	
	numeric	
1	1024.0000000000000000	

- ✓ (26) Определение суммарного требуемого для установки программы места на диске:

**SELECT SUM (obem) FROM programma WHERE status NOT IN('Checked');**

98


SELECT SUM (obem) FROM programma WHERE status NOT IN('Checked');

Data Output


Messages


Notifications


≡+





▼












	sum	
	bigint	
1	960	

## 5 Группировка

### 5.1 Общая информация

- ✓ (27) Суммирование данных столбца mesto\_disk\_prog таблицы programma с группировкой по содержанию столбца status:

**SELECT status, SUM (mesto\_disk\_prog) AS mesto\_sum**  
**FROM programma**  
**GROUP BY status**  
**ORDER BY mesto\_sum;**

```

100 SELECT status, SUM (mesto_disk_prog) AS mesto_sum
101 FROM programma
102 GROUP BY status
103 ORDER BY mesto_sum;

```

Data Output Messages Notifications

	status character varying (10)	mesto_sum bigint
1	Checked	5120
2	Testing	5888

✓ (28) Группировка по одному столбцу с условием WHERE:

```

SELECT status, SUM (mesto_disk_prog) AS mesto_sum
FROM programma
WHERE op_pamyat_prog <16
GROUP BY status
ORDER BY mesto_sum;

```

```

100 SELECT status, SUM (mesto_disk_prog) AS mesto_sum
101 FROM programma
102 WHERE op_pamyat_prog <16
103 GROUP BY status
104 ORDER BY mesto_sum;

```

Data Output Messages Notifications

	status character varying (10)	mesto_sum bigint
1	Testing	1280
2	Checked	4096

✓ (29) Группировка по двум столбцам:

```

SELECT id_tip, status, SUM (mesto_disk_prog) AS mesto_sum
FROM programma
GROUP BY id_tip, status;

```

```

101 SELECT id_tip, status, SUM (mesto_disk_prog) AS mesto_sum
102 FROM programma
103 GROUP BY id_tip, status;
104

```

Data Output Messages Notifications

	id_tip smallint	status character varying (10)	mesto_sum bigint
1	2	Checked	2048
2	1	Testing	4352
3	2	Testing	1536
4	1	Checked	3072

## 5.2 Группировка с having

- ✓ (30) Группировка с HAVING и агрегатной функцией SUM:

```

SELECT status, SUM (mesto_disk_prog) AS mesto_sum
FROM programma
GROUP BY status
HAVING SUM (mesto_disk_prog) < 5500
ORDER BY mesto_sum;

```

```

105 SELECT status, SUM (mesto_disk_prog) AS mesto_sum
106 FROM programma
107 GROUP BY status
108 HAVING SUM (mesto_disk_prog) < 5500
109 ORDER BY mesto_sum;
110
111

```

Data Output Messages Notifications

	status character varying (10)	mesto_sum bigint
1	Checked	5120

- ✓ (31) Группировка с HAVING и агрегатной функцией MAX:

```

SELECT status, MAX (mesto_disk_prog) AS mesto_max
FROM programma
GROUP BY status
HAVING MAX (mesto_disk_prog) < 3000
ORDER BY mesto_max;

```



```

105 SELECT status, MAX (mesto_disk_prog) AS mesto_max
106 FROM programma
107 GROUP BY status
108 HAVING MAX (mesto_disk_prog) < 3000
109 ORDER BY mesto_max;
110
111

```

Data Output   Messages   Notifications



	status character varying (10)	mesto_max smallint
1	Checked	2048

## 6 Выборка из нескольких таблиц

### 6.1 Соединение с помощью FROM и WHERE

✓ (32) Соединение двух таблиц с помощью FROM и WHERE:

```

SELECT
pr.imya,
pr.chastota_pr_prog,
pr.op_pamyat_prog,
pr.mesto_disk_prog,
tip.tip_programmy
FROM
programma AS pr,
tip_programmy AS tip
WHERE
pr.id_tip = tip.id_tip;

```

```

383 SELECT
384 pr.imya,
385 pr.chastota_pr_prog,
386 pr.op_pamyat_prog,
387 pr.mesto_disk_prog,
388 tip.tip_programmy
389 FROM
390 programma AS pr,
391 tip_programmy AS tip
392 WHERE
393 pr.id_tip = tip.id_tip;

```

Data Output Messages Graph Visualiser × Notifications

	imya character varying (20)	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint	tip_programmy character varying (35)
1	Редактор	1,4	3	512	Прикладное программное обеспечение
2	Организатор	1,7	5	512	Прикладное программное обеспечение
3	Поиск	1,4	2	1024	Прикладное программное обеспечение
4	Навигатор	1,9	8	1024	Прикладное программное обеспечение
5	Игра	2,2	16	512	Прикладное программное обеспечение
6	Структура	1,2	8	256	Системное программное обеспечение
7	Оболочка	1,8	8	[null]	Системное программное обеспечение
8	Оператор	2,7	32	4096	Системное программное обеспечение
9	Система	2,1	4	2048	Системное программное обеспечение

## 6.2 Соединение с помощью JOIN

- ✓ (33) Соединение двух таблиц с помощью INNER JOIN с последующим выводом данных:

```

SELECT
pr.imya,
pr.chastota_pr_prog,
pr.op_pamyat_prog,
pr.mesto_disk_prog,
tip.tip_programmy
FROM
programma AS pr
JOIN tip_programmy AS tip ON tip.id_tip = pr.id_tip;

```

```

383 SELECT
384 pr.imya,
385 pr.chastota_pr_prog,
386 pr.op_pamyat_prog,
387 pr.mesto_disk_prog,
388 tip.tip_programmy
389 FROM
390 programma AS pr
391 JOIN tip_programmy AS tip ON tip.id_tip = pr.id_tip;

```

	imya character varying (35)	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint	tip_programmy character varying (35)
1	Редактор	1,4	3	512	Прикладное программное обеспечение
2	Организатор	1,7	5	512	Прикладное программное обеспечение
3	Поиск	1,4	2	1024	Прикладное программное обеспечение
4	Навигатор	1,9	8	1024	Прикладное программное обеспечение
5	Игра	2,2	16	512	Прикладное программное обеспечение
6	Структура	1,2	8	256	Системное программное обеспечение
7	Оболочка	1,8	8	[null]	Системное программное обеспечение
8	Оператор	2,7	32	4096	Системное программное обеспечение
9	Система	2,1	4	2048	Системное программное обеспечение

- ✓ (34) Соединение двух таблиц с помощью INNER JOIN с последующим выводом данных. Тестирования с не заполненным id\_programmy не попали в выборку:

```

SELECT
test.nomer_testirovania,
pr.*
FROM
testirovanie AS test
JOIN programma AS pr ON pr.id_programmy = test.id_programmy;

```

```

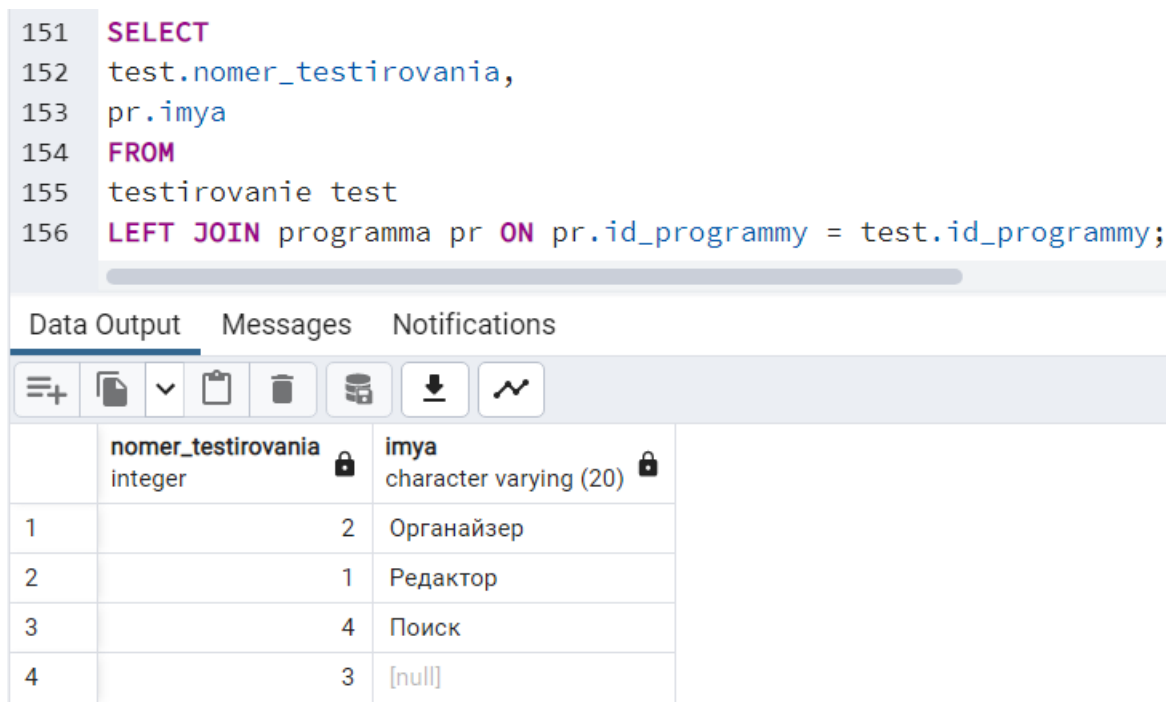
152 SELECT
153 test.nomer_testirovania,
154 pr.*
155 FROM
156 testirovanie AS test
157 JOIN programma AS pr ON pr.id_programmy = test.id_programmy;
158

```

	nomer_testirovania integer	id_programmy integer	id_tip smallint	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint	data_rasrabotka date	obem smallint	imya character varying (35)	status character varying (35)
1	2	8	2	1,7	4	512	2020-11-08	128	Организатор	Checked
2	1	4	2	1,4	2	512	2021-05-14	128	Редактор	Testing
3	4	7	2	1,4	2	1024	2019-02-24	64	Поиск	Testing

- ✓ (35) Соединение двух таблиц с помощью LEFT JOIN с выводом данных. Тестирования с не заполненным test.id\_programmy попали в выборку:

```
SELECT
test.nomer_testirovania,
pr.imya
FROM
testirovanie test
LEFT JOIN programma pr ON pr.id_programmy = test.id_programmy;
```



The screenshot shows a database query editor with a SQL query and its results. The query is a LEFT JOIN between the 'testirovanie' table (aliased as 'test') and the 'programma' table (aliased as 'pr'). The query selects 'test.nomer\_testirovania' and 'pr.imya'. The results are displayed in a table with two columns: 'nomer\_testirovania' (integer) and 'imya' (character varying (20)).

nomer_testirovania integer	imya character varying (20)
1	Организер
2	Редактор
3	Поиск
4	[null]

- ✓ (36) Соединение двух таблиц с помощью RIGHT JOIN с выводом данных. В выборку попали программы, не прошедшие ни одного тестирования, а тестирования с не заполненным test.id\_programmy в выборку не попали:

```
SELECT
test.nomer_testirovania,
pr.imya
FROM
testirovanie test
RIGHT JOIN programma pr ON pr.id_programmy = test.id_programmy;
```

```

158 SELECT
159 test.nomer_testirovania,
160 pr.imya
161 FROM
162 testirovanie test
163 RIGHT JOIN programma pr ON pr.id_programmy = test.id_programmy;

```

Data Output Messages Notifications

	nomer_testirovania integer	imya character varying (20)
1	2	Организер
2	1	Редактор
3	4	Поиск
4	[null]	Структура
5	[null]	Навигатор
6	[null]	Оболочка
7	[null]	Игра
8	[null]	Система
9	[null]	Оператор

- ✓ (37) Соединение двух таблиц с помощью RIGHT JOIN с выводом данных. В выборку попали как программы, не прошедшие ни одного тестирования, так и тестирования с не заполненным test.id\_programmy:

```

SELECT
test.nomer_testirovania,
pr.imya
FROM
testirovanie test
FULL JOIN programma pr ON pr.id_programmy = test.id_programmy;

```

```

158 SELECT
159 test.nomer_testirovania,
160 pr.imya
161 FROM
162 testirovanie test
163 FULL JOIN programma pr ON pr.id_programmy = test.id_programmy;

```

Data Output Messages Notifications



	nomer_testirovania integer	imya character varying (20)
1	2	Организатор
2	1	Редактор
3	4	Поиск
4	3	[null]
5	[null]	Структура
6	[null]	Навигатор
7	[null]	Оболочка
8	[null]	Игра
9	[null]	Система
10	[null]	Оператор

- ✓ (38) Каждой строке столбца nomer\_testirovania таблицы testirovanie сопоставляется каждая строка imya таблицы programma:

```

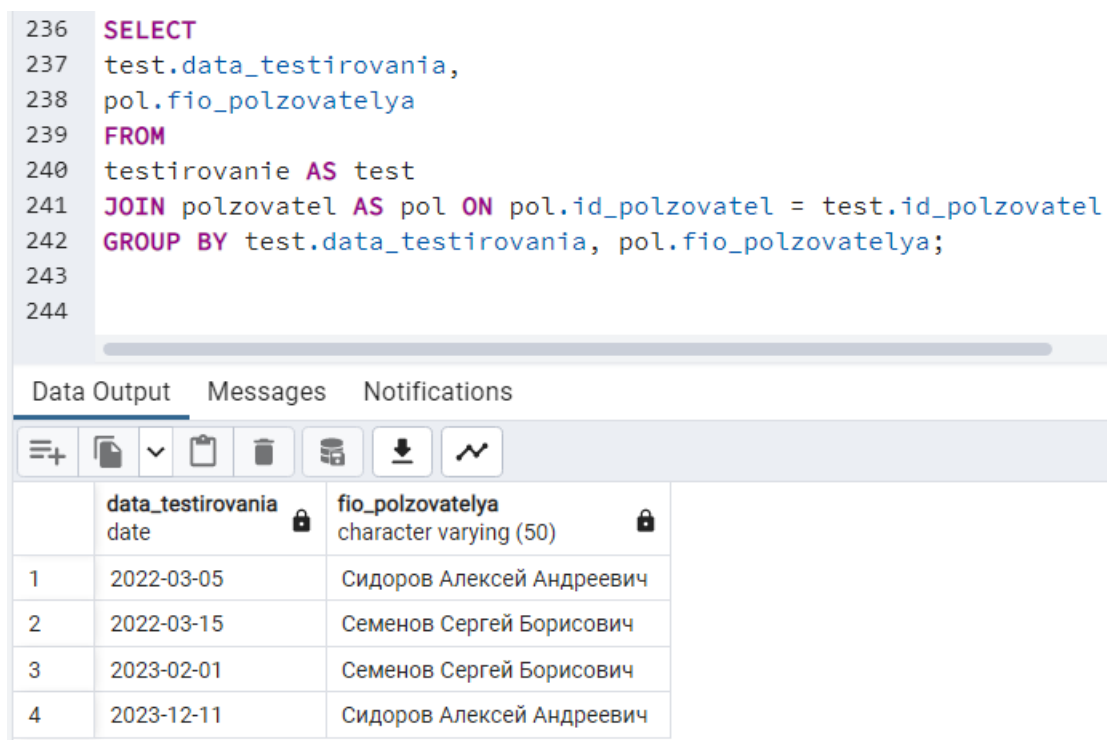
SELECT
test.nomer_testirovania,
pr.imya
FROM
testirovanie test
CROSS JOIN programma pr

```

	<b>nomer_testirovania</b> integer 	<b>imya</b> character varying (20) 
1	2	Оболочка
2	2	Игра
3	2	Система
4	2	Навигатор
5	2	Поиск
6	2	Оператор
7	2	Структура
8	2	Органайзер
9	2	Редактор
10	1	Оболочка
11	1	Игра
12	1	Система
13	1	Навигатор
14	1	Поиск
15	1	Оператор
16	1	Структура
17	1	Органайзер
18	1	Редактор
19	4	Оболочка
20	4	Игра
21	4	Система
22	4	Навигатор
23	4	Поиск
24	4	Оператор
25	4	Структура
26	4	Органайзер
27	4	Редактор
28	3	Оболочка
29	3	Игра
30	3	Система
31	3	Навигатор
32	3	Поиск
33	3	Оператор
34	3	Структура
35	3	Органайзер
36	3	Редактор

- ✓ (39) Объединение таблиц testirovanie и polzovatel с группировкой по дате тестирования:

```
SELECT
test.data_testirovania,
pol.fio_polzovatelya
FROM
AS test
JOIN polzovatel AS pol ON pol.id_polzovatel = test.id_polzovatel
GROUP BY test.data_testirovania, pol.fio_polzovatelya;
```



The screenshot shows a database query editor with a SQL query and its results. The query is as follows:

```
236 SELECT
237 test.data_testirovania,
238 pol.fio_polzovatelya
239 FROM
240 testirovanie AS test
241 JOIN polzovatel AS pol ON pol.id_polzovatel = test.id_polzovatel
242 GROUP BY test.data_testirovania, pol.fio_polzovatelya;
243
244
```

Below the query editor, there is a tabbed interface with 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with the following data:

	data_testirovania date	fio_polzovatelya character varying (50)
1	2022-03-05	Сидоров Алексей Андреевич
2	2022-03-15	Семенов Сергей Борисович
3	2023-02-01	Семенов Сергей Борисович
4	2023-12-11	Сидоров Алексей Андреевич

- ✓ (40) Объединение таблиц с помощью RIGHT JOIN и группировкой по данным столбца imya таблицы programma:

```
SELECT
tes.nomer_testirovania,
tes.data_testirovania,
pr.imya,
pr.mesto_disk_prog
FROM testirovanie tes
RIGHT JOIN programma pr ON pr.id_programmy = tes.id_programmy
GROUP BY pr.imya, tes.nomer_testirovania, pr.mesto_disk_prog;
```



```

333 SELECT
334 tes.nomer_testirovania,
335 tes.data_testirovania,
336 pr.imya,
337 pr.mesto_disk_prog
338 FROM testirovanie tes
339 RIGHT JOIN programma pr ON pr.id_programmy = tes.id_programmy
340 GROUP BY pr.imya, tes.nomer_testirovania, pr.mesto_disk_prog;

```

Data Output Messages Graph Visualiser × Notifications

	nomer_testirovania integer	data_testirovania date	imya character varying (20)	mesto_disk_prog smallint
1	1	2021-06-12	Редактор	512
2	[null]	[null]	Система	2048
3	[null]	[null]	Игра	512
4	4	2022-05-15	Поиск	1024
5	[null]	[null]	Структура	256
6	[null]	[null]	Оператор	4096
7	2	2022-12-02	Организер	512
8	[null]	[null]	Навигатор	1024
9	3	2020-05-01	Оболочка	[null]

- ✓ (41) Объединение таблиц programma и tip\_programmy с помощью JOIN и группировка по столбцу id\_tip таблицы programma с подсчетом среднего значения данных столбца mesto\_disk\_prog таблицы programma:

```

SELECT
pr.id_tip,
tip.tip_programmy,
AVG (pr.mesto_disk_prog) AS mesto_avg
FROM
programma AS pr
JOIN tip_programmy AS tip ON tip.id_tip = pr.id_tip
GROUP BY pr.id_tip, tip.tip_programmy
ORDER BY mesto_avg;

```

```

232 SELECT
233 pr.id_tip,
234 tip.tip_programmy,
235 AVG (pr.mesto_disk_prog) AS mesto_avg
236 FROM
237 programma AS pr
238 JOIN tip_programmy AS tip ON tip.id_tip = pr.id_tip
239 GROUP BY pr.id_tip, tip.tip_programmy
240 ORDER BY mesto_avg;

```

Data Output Messages Notifications

	id_tip smallint	tip_programmy character varying (35)	mesto_avg numeric
1	2	Прикладное программное обеспечение	716.8000000000000000
2	1	Системное программное обеспечение	2133.3333333333333333

- ✓ (42) Объединение таблиц `programma` и `tip_programmy` с помощью `LEFT JOIN` и группировка по столбцу `mesto_disk_prog` таблицы `programma` с суммированием данных столбца `op_pamyat_prog` таблицы `programma`. В выборку попали программы с не заполненными полями столбца `mesto_disk_prog`:

```

SELECT
pr.mesto_disk_prog,
tip.tip_programmy,
SUM (op_pamyat_prog) AS pamyat_sum
FROM programma AS pr
LEFT JOIN tip_programmy AS tip ON tip.id_tip = pr.id_tip
GROUP BY pr.mesto_disk_prog, tip.tip_programmy
ORDER BY pamyat_sum;

```

```

223 SELECT
224 pr.mesto_disk_prog,
225 tip.tip_programmy,
226 SUM (op_pamyat_prog) AS pamyat_sum
227 FROM programma AS pr
228 LEFT JOIN tip_programmy AS tip ON tip.id_tip = pr.id_tip
229 GROUP BY pr.mesto_disk_prog, tip.tip_programmy
230 ORDER BY pamyat_sum;
231

```

Data Output Messages Notifications

	mesto_disk_prog smallint	tip_programmy character varying (35)	pamyat_sum bigint
1	256	Системное программное обеспечение	2
2	2048	Системное программное обеспечение	4
3	[null]	Системное программное обеспечение	8
4	1024	Прикладное программное обеспечение	10
5	512	Прикладное программное обеспечение	22
6	4096	Системное программное обеспечение	32

## 7 Знакомство со встроенными функциями

- ✓ (43) Вычисление разницы суммы свободных мест на дисках пользователей и суммы требуемых мест на диске для разработанных программ:

```

SELECT
SUM (pol.mesto_disk_pol) - SUM(pr.mesto_disk_prog)
FROM programma pr
JOIN testirovanie tes ON pr.id_programmy = tes.id_programmy
JOIN polzovatel pol ON tes.id_polzovatel = pol.id_polzovatel;

```

```

303 SELECT
304 SUM (pol.mesto_disk_pol) - SUM(pr.mesto_disk_prog)
305 FROM programma pr
306 JOIN testirovanie tes ON pr.id_programmy = tes.id_programmy
307 JOIN polzovatel pol ON tes.id_polzovatel = pol.id_polzovatel;
308

```

Data Output Messages Graph Visualiser × Notifications



	?column? bigint
1	1024

- ✓ (44) Вычисление разницы между максимальным размером оперативной памяти компьютера пользователя и минимальным требуемым объемом оперативной памяти программы:

```

SELECT
MAX (pol.op_pamyat_pol) - MIN(pr.op_pamyat_prog)
FROM programma pr
JOIN testirovanie tes ON pr.id_programmy = tes.id_programmy
JOIN polzovatel pol ON tes.id_polzovatel = pol.id_polzovatel;

```

```

303 SELECT
304 MAX (pol.op_pamyat_pol) - MIN(pr.op_pamyat_prog)
305 FROM programma pr
306 JOIN testirovanie tes ON pr.id_programmy = tes.id_programmy
307 JOIN polzovatel pol ON tes.id_polzovatel = pol.id_polzovatel;
308

```

Data Output Messages Graph Visualiser × Notifications



	?column? smallint
1	14

- ✓ (45) Объединение данных двух таблиц с выводом значений в верхнем регистре:

```

SELECT
UPPER(programma.imya),
UPPER(tip_programmy.tip_programmy)
FROM
programma
JOIN tip_programmy ON tip_programmy.id_tip = programma.id_tip;

```

```

252 SELECT
253 UPPER(programma.imya),
254 UPPER(tip_programmy.tip_programmy)
255 FROM
256 programma
257 JOIN tip_programmy ON tip_programmy.id_tip = programma.id_tip;

```

Data Output Messages Notifications



	upper text	upper text
1	РЕДАКТОР	ПРИКЛАДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ
2	ОРГАНАЙЗЕР	ПРИКЛАДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ
3	ПОИСК	ПРИКЛАДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ
4	НАВИГАТОР	ПРИКЛАДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ
5	ИГРА	ПРИКЛАДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ
6	ОБОЛОЧКА	СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ
7	СТРУКТУРА	СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ
8	ОПЕРАТОР	СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ
9	СИСТЕМА	СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

✓ (46) Объединение строк из разных таблиц:

```

SELECT
programma.imya || ' ' || '[' || tip_programmy.tip_programmy || ']'
FROM
programma
JOIN tip_programmy ON tip_programmy.id_tip = программа.id_tip;

```

252	SELECT
253	programma.imya    ' '    '['    tip_programmy.tip_programmy    '']
254	FROM
255	programma
256	JOIN tip_programmy ON tip_programmy.id_tip = programma.id_tip;

Data Output	Messages	Notifications
-------------	----------	---------------

?	column?	text	lock
1	Редактор	[Прикладное программное обеспечение]	
2	Организатор	[Прикладное программное обеспечение]	
3	Поиск	[Прикладное программное обеспечение]	
4	Навигатор	[Прикладное программное обеспечение]	
5	Игра	[Прикладное программное обеспечение]	
6	Оболочка	[Системное программное обеспечение]	
7	Структура	[Системное программное обеспечение]	
8	Оператор	[Системное программное обеспечение]	
9	Система	[Системное программное обеспечение]	

✓ (47) Извлечение года из даты:

SELECT \*, EXTRACT(YEAR FROM data\_razrabotki)  
FROM rasrabotka;

389	SELECT *, EXTRACT(YEAR FROM data_razrabotki)
390	FROM rasrabotka;

Data Output	Messages	Graph Visualiser	×	Notifications
-------------	----------	------------------	---	---------------

?	+	↓	📋	🗑️	🗄️	⬇️	📈
	nomer_razrab	data_razrabotki	id_programm	id_programm	extract		
	[PK] integer	date	smallint	smallint	numeric		lock
1	4	2023-02-01	3	6	2023		
2	2	2021-05-14	4	5	2021		
3	6	2020-02-01	5	5	2020		
4	7	2019-02-24	7	6	2019		
5	5	2020-11-08	8	5	2020		
6	3	2021-05-14	6	6	2021		
7	8	2023-07-01	9	6	2023		
8	9	2021-05-14	10	5	2021		
9	10	2018-02-13	11	6	2018		

✓ (48) Расчет разницы в годах между текущей датой и датой разработки программы с id\_programmy = 6:

```
SELECT date_part('year', now()) - date_part('year', data_razrabotki)
FROM rasrabotka WHERE id_programmy = 6;
```

389	SELECT	date_part('year', now()) - date_part('year', data_razrabotki)
390	FROM	rasrabotka WHERE id_programmy = 6;

Data Output	Messages	Graph Visualiser	×	Notifications
<div> <div>≡</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>				
<div> <div>?column?</div> <div>double precision</div> <div>🔒</div> </div>				
1		2		

- ✓ (49) Расчет разницы в месяцах между текущей датой и датой разработки программы с id\_programmy = 6:

```
SELECT (date_part('year', now()) - date_part('year', data_razrabotki))*12 -
(date_part('month', now()) - date_part('month', data_razrabotki))
FROM rasrabotka WHERE id_programmy = 6;
```

389	SELECT	(date_part('year', now()) - date_part('year', data_razrabotki))*12 -
390		(date_part('month', now()) - date_part('month', data_razrabotki))
391	FROM	rasrabotka WHERE id_programmy = 6;

Data Output	Messages	Graph Visualiser	×	Notifications
<div> <div>≡</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>				
<div> <div>?column?</div> <div>double precision</div> <div>🔒</div> </div>				
1		27		

- ✓ (50) Расчет разницы в днях между текущей датой и датой разработки программы с id\_programmy = 6:

```
SELECT now() - data_razrabotki FROM rasrabotka WHERE id_programmy = 6;
```

389	SELECT	now() - data_razrabotki	FROM	rasrabotka	WHERE	id_programmy = 6;
-----	--------	-------------------------	------	------------	-------	-------------------

Data Output	Messages	Graph Visualiser	×	Notifications
<div> <div>≡</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>				
<div> <div>?column?</div> <div>interval</div> <div>🔒</div> </div>				
1		630 days 18:50:43.460707		

- ✓ (51) Вывод данных программ, которые тестировались более двух лет назад:

```
SELECT
tes.data_testirovania,
```

```

pr.imya,
pr.chastota_pr_prog,
pr.op_pamyat_prog,
pr.mesto_disk_prog,
pr.obem
FROM programma pr
JOIN testirovanie tes ON pr.id_programmy = tes.id_programmy
WHERE tes.data_testirovania <= (now() - interval '2 year');

```

```

390 SELECT
391 tes.data_testirovania,
392 pr.imya,
393 pr.chastota_pr_prog,
394 pr.op_pamyat_prog,
395 pr.mesto_disk_prog,
396 pr.obem
397 FROM programma pr
398 JOIN testirovanie tes ON pr.id_programmy = tes.id_programmy
399 WHERE tes.data_testirovania <= (now() - interval '2 year');

```

	data_testirovania date	imya character varying (20)	chastota_pr_p character (3)	op_pamyat_p smallint	mesto_disk_p smallint	obem smallint
1	2020-05-01	Оболочка	1,8	8	[null]	512

- ✓ (52) Вывод Ф.И.О. пользователей, тестировавших программы и течение последних шести месяцев:

```

SELECT
tes.data_testirovania,
pol.fio_polzovatelya
FROM polzovatel pol
JOIN testirovanie tes ON tes.id_polzovatel = pol.id_polzovatel
WHERE tes.data_testirovania > (now() - interval '6 month');

```



317	SELECT
318	tes.data_testirovania,
319	pol.fio_polzovatelya
320	FROM polzovatel pol
321	JOIN testirovanie tes ON tes.id_polzovatel = pol.id_polzovatel
322	WHERE tes.data_testirovania > (now() - interval '6 month');

Data Output	Messages	Graph Visualiser	×	Notifications
-------------	----------	------------------	---	---------------

	data_testirovania date	fio_polzovatelya character varying (50)
1	2022-12-02	Петров Семен Андреевич

- ✓ (53) Вывод времени, прошедшего с момента выпуска первой программы в формате  
«Дни:Часы:Минусы:Секунды»:

SELECT now() - (SELECT MIN (data\_razrabotki) FROM rasrabotka);

390	SELECT now() - (SELECT MIN (data_razrabotki) FROM rasrabotka);
-----	----------------------------------------------------------------

Data Output	Messages	Graph Visualiser	×	Notifications
-------------	----------	------------------	---	---------------

	?column? interval
1	1816 days 19:06:51.963777

- ✓ (54) Вывод числа символов в строке, содержащей Ф.И.О. пользователя:

```
SELECT
fio_polzovatelya,
char_length(fio_polzovatelya)
FROM
polzovatel;
```

```

327 SELECT
328 fio_polzovatelya,
329 char_length(fio_polzovatelya)
330 FROM
331 polzovatel;

```

	fio_polzovatelya character varying (50)	char_length integer
1	Семенов Сергей Борисович	24
2	Сидоров Алексей Андреевич	25
3	Петров Семен Андреевич	22
4	Иванов Семен Петрович	21

✓ (54) Вывод позиции буквы «о» в Ф.И.О. пользователей:

```

SELECT
fio_polzovatelya,
position('o' in fio_polzovatelya)
FROM
polzovatel;

```

```

327 SELECT
328 fio_polzovatelya,
329 position('o' in fio_polzovatelya)
330 FROM
331 polzovatel;

```

	fio_polzovatelya character varying (50)	position integer
1	Семенов Сергей Борисович	6
2	Сидоров Алексей Андреевич	4
3	Петров Семен Андреевич	5
4	Иванов Семен Петрович	5

## 8 Дополнительные возможности

✓ (55) Исключение из результата выборки повторяющихся строк столбца op\_pamyat\_prog таблицы programma:

```

SELECT DISTINCT op_pamyat_prog FROM programma ORDER BY
op_pamyat_prog;

```

273 `SELECT DISTINCT op_pamyat_prog FROM programma ORDER BY op_pamyat_prog;`

Data Output Messages Notifications

	op_pamyat_prog smallint
1	2
2	4
3	8
4	16
5	32

- ✓ (56) Ограничение максимального числа строк в выборке с указанием строки, с которой будет начинаться выборка:

```
SELECT * FROM programma pro WHERE pro.id_tip = 2 ORDER BY
id_programmy LIMIT 10 OFFSET 2;
```

275 `SELECT * FROM programma pro WHERE pro.id_tip = 2 ORDER BY id_programmy LIMIT 10 OFFSET 2;`

Data Output Messages Notifications

	id_programm [PK] integer	id_tip smallint	chastota_pr_ character (3)	op_pamyat_p smallint	mesto_disk_p smallint	data_rasrabo date	obem smallint	imya character var	status character var
1	7	2	1,4	2	1024	2019-02-24	64	Поиск	Testing
2	8	2	1,7	4	512	2020-11-08	128	Органаиз...	Checked
3	10	2	1,9	8	1024	2021-05-14	128	Навигатор	Checked

## 9 Вложенные запросы

### 9.1 Общая информация

- ✓ (57) Выборка программ из таблицы программа, принадлежащих к типу прикладного ПО и при условии, что требуемый объем оперативной памяти программы не превышает минимального размера оперативной памяти среди компьютеров пользователей.

```
SELECT
pr.imya,
pr.chastota_pr_prog,
pr.op_pamyat_prog
FROM programma pr
JOIN tip_programmy tip ON pr.id_tip = tip.id_tip
WHERE tip.id_tip = 2
```

AND op\_pamyat\_prog <= (SELECT MIN(op\_pamyat\_pol) FROM polzovatel);

```

390 SELECT
391 pr.imya,
392 pr.chastota_pr_prog,
393 pr.op_pamyat_prog
394 FROM programma pr
395 JOIN tip_programmy tip ON pr.id_tip = tip.id_tip
396 WHERE tip.id_tip = 2
397 AND op_pamyat_prog <= (SELECT MIN(op_pamyat_pol) FROM polzovatel);

```

	imya character varying (20)	chastota_pr_prog character (3)	op_pamyat_prog smallint
1	Поиск	1,4	2
2	Редактор	1,4	3

- ✓ (58) Увеличение требуемого для работы программы объема оперативной памяти на 1 Гб если ее тестировал пользователь с id\_polzovatel равным 2:

UPDATE programma SET op\_pamyat\_prog = op\_pamyat\_prog + 1  
WHERE id\_programmy IN (SELECT id\_programmy FROM testirovanie WHERE id\_polzovatel = 2);

```

279 UPDATE programma SET op_pamyat_prog = op_pamyat_prog + 1
280 WHERE id_programmy IN (SELECT id_programmy FROM testirovanie WHERE id_polzovatel = 2);

```

	id_programm [PK] integer	id_tip smallint	chastota_pr_p character (3)	op_pamyat_p smallint	mesto_disk_p smallint	data_rasrabot date	obem smallint	imya character vary	status character vary
1	6	2	2,2	16	512	2021-05-14	128	Игра	Checked
2	3	1	2,1	4	2048	2023-02-01	256	Система	Checked
3	10	2	1,9	8	1024	2021-05-14	128	Навигатор	Checked
4	7	2	1,4	2	1024	2019-02-24	64	Поиск	Testing
5	9	1	2,7	32	4096	2023-07-01	256	Оператор	Testing
6	11	1	1,2	2	256	2018-02-13	512	Структура	Testing
7	5	1	1,8	8	[null]	2020-02-01	512	Оболочка	Checked
8	8	2	1,7	5	512	2020-11-08	128	Организ...	Checked
9	4	2	1,4	3	512	2021-05-14	128	Редактор	Testing

- ✓ (59) Выборка программ с объемом меньше или равно 128 мегабайт и датой разработки не позже максимальной среди дат тестирований:

```

SELECT
pr.imya,
ras.data_razrabotki,

```

```

pr.obem,
tes.id_programmy
FROM programma pr
JOIN testirovanie tes ON pr.id_programmy = tes.id_programmy
JOIN rasrabotka ras ON ras.id_programmy = tes.id_programmy
WHERE pr.obem <= 128
AND ras.data_razrabotki < (SELECT MAX(data_testirovania) FROM testirovanie
tes1
WHERE tes1.id_programmy = tes.id_programmy);

```

```

390 SELECT
391 pr.imya,
392 ras.data_razrabotki,
393 pr.obem,
394 tes.id_programmy
395 FROM programma pr
396 JOIN testirovanie tes ON pr.id_programmy = tes.id_programmy
397 JOIN rasrabotka ras ON ras.id_programmy = tes.id_programmy
398 WHERE pr.obem <= 128
399 AND ras.data_razrabotki < (SELECT MAX(data_testirovania) FROM testirovanie tes1
400 WHERE tes1.id_programmy = tes.id_programmy);

```

	imya character varying (20)	data_razrabotki date	obem smallint	id_programmy smallint
1	Редактор	2021-05-14	128	4
2	Поиск	2019-02-24	64	7
3	Организер	2020-11-08	128	8

- ✓ (60) Изменение величины требуемой оперативной памяти для программы с id\_programmy = 11 до величины, соответствующей минимальной оперативной памяти на компьютерах пользователей:

```

UPDATE programma
SET op_pamyat_prog = (SELECT MIN(op_pamyat_pol) FROM polzovatel)
WHERE id_programmy = 11;

```

```

262 UPDATE programma
263 SET op_pamyat_prog = (SELECT MIN(op_pamyat_pol) FROM polzovatel)
264 WHERE id_programmy = 11;

```

Data Output Messages Notifications

	id_programm [PK] integer	id_tip smallint	chastota_pr_ character (3)	op_pamyat_p smallint	mesto_disk_p smallint	data_rasrabot date	obem smallint	imya character var	status character var
1	6	2	2,2	16	512	2021-05-14	128	Игра	Checked
2	3	1	2,1	4	2048	2023-02-01	256	Система	Checked
3	10	2	1,9	8	1024	2021-05-14	128	Навигатор	Checked
4	7	2	1,4	2	1024	2019-02-24	64	Поиск	Testing
5	9	1	2,7	32	4096	2023-07-01	256	Оператор	Testing
6	5	1	1,8	8	[null]	2020-02-01	512	Оболочка	Checked
7	8	2	1,7	5	512	2020-11-08	128	Органайз...	Checked
8	4	2	1,4	3	512	2021-05-14	128	Редактор	Testing
9	11	1	1,2	8	256	2018-02-13	512	Структура	Testing

## 9.2 Функция EXISTS

✓ (61) Вывод данных о когда-либо тестируемых программах:

```

SELECT
pr.imya,
pr.mesto_disk_prog
FROM programma pr
WHERE EXISTS(SELECT 1 FROM testirovanie tes
              WHERE pr.id_programmy = tes.id_programmy);

```

```

343 SELECT
344 pr.imya,
345 pr.mesto_disk_prog
346 FROM programma pr
347 WHERE EXISTS(SELECT 1 FROM testirovanie tes
348              WHERE pr.id_programmy = tes.id_programmy);

```

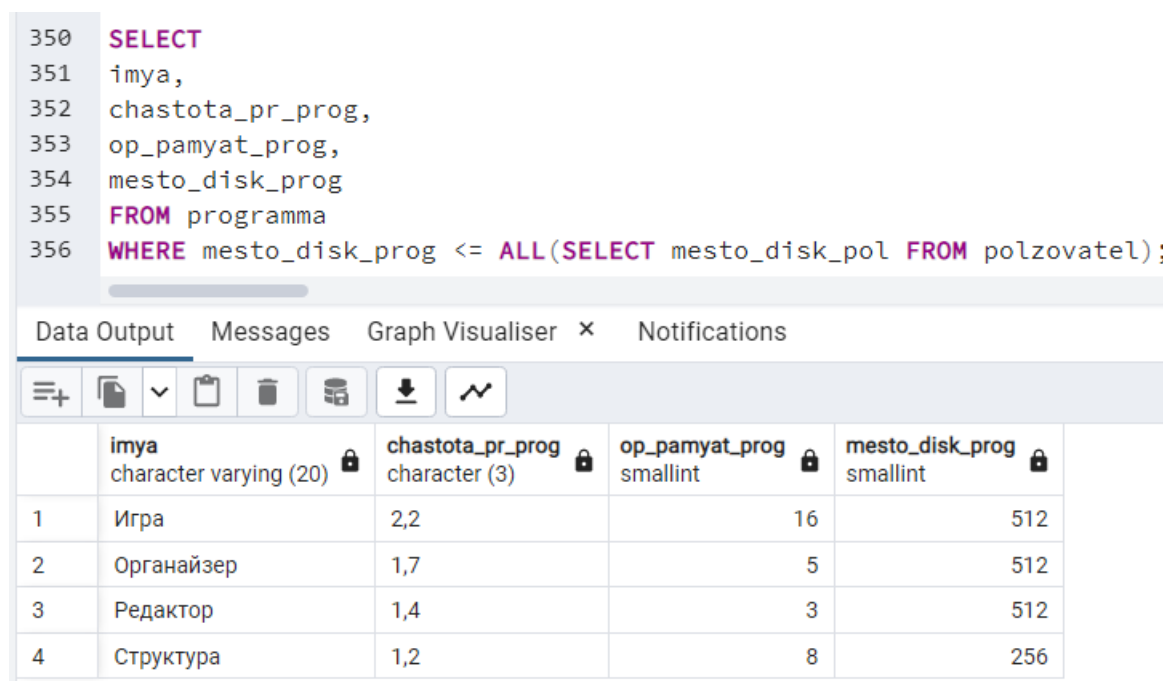
Data Output Messages Graph Visualiser × Notifications

	imya character varying (20)	mesto_disk_prog smallint
1	Оболочка	[null]
2	Редактор	512
3	Поиск	1024
4	Органайзер	512

### 9.3 Функция ALL

- ✓ (62) Вывод данных о программах, требуемое место на диске для которых меньше или равно самого маленького свободного места на дисках пользователей:

```
SELECT
имya,
chastota_pr_prog,
op_pamyat_prog,
mesto_disk_prog
FROM programma
WHERE mesto_disk_prog <= ALL(SELECT mesto_disk_pol FROM polzovatel);
```



The screenshot shows a database query interface. The query editor displays the following SQL code:

```
350 SELECT
351 имya,
352 chastota_pr_prog,
353 op_pamyat_prog,
354 mesto_disk_prog
355 FROM programma
356 WHERE mesto_disk_prog <= ALL(SELECT mesto_disk_pol FROM polzovatel);
```

Below the query editor, there is a toolbar with icons for query execution, and a results table with 4 rows of data. The table has the following columns: **имya** (character varying (20)), **chastota\_pr\_prog** (character (3)), **op\_pamyat\_prog** (smallint), and **mesto\_disk\_prog** (smallint). The data is as follows:

	имya	chastota_pr_prog	op_pamyat_prog	mesto_disk_prog
1	Игра	2,2	16	512
2	Организер	1,7	5	512
3	Редактор	1,4	3	512
4	Структура	1,2	8	256

- ✓ (63) Вывод данных о программах, требуемая оперативная память для работы которых больше или равна самой большой оперативной памяти, установленной на компьютеры пользователей:

```
SELECT
имya,
chastota_pr_prog,
op_pamyat_prog,
mesto_disk_prog
FROM programma
WHERE op_pamyat_prog >= ALL(SELECT op_pamyat_pol FROM polzovatel);
```

```

350 SELECT
351 imya,
352 chastota_pr_prog,
353 op_pamyat_prog,
354 mesto_disk_prog
355 FROM programma
356 WHERE op_pamyat_prog >= ALL(SELECT op_pamyat_pol FROM polzovatel);

```

Data Output Messages Graph Visualiser × Notifications

	imya character varying (20)	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint
1	Оператор	2,7	32	4096

## 10 Написание запросов по вариантам

✓ (64) Вывод полной информации о программе с id\_programmy = 8:

```

SELECT
pr.*,
tip.tip_programmy,
ras.data_razrabotki,
prst.fio_programmista,
tes.data_testirovania,
pol.fio_polzovatelya
FROM programma pr
JOIN tip_programmy tip ON tip.id_tip = pr.id_tip
JOIN rasrabotka ras ON ras.id_programmy = pr.id_programmy
JOIN programmist prst ON prst.id_programmista = ras.id_programmista
JOIN testirovanie tes ON pr.id_programmy = tes.id_programmy
JOIN polzovatel pol ON tes.id_polzovatel = pol.id_polzovatel
WHERE pr.id_programmy = 8;

```

```

374 SELECT
375 pr.*,
376 tip.tip_programmy,
377 ras.data_razrabotki,
378 prst.fio_programmista,
379 tes.data_testirovania,
380 pol.fio_polzovatelya
381 FROM programma pr
382 JOIN tip_programmy tip ON tip.id_tip = pr.id_tip
383 JOIN rasrabotka ras ON ras.id_programmy = pr.id_programmy
384 JOIN programmist prst ON prst.id_programmista = ras.id_programmista
385 JOIN testirovanie tes ON pr.id_programmy = tes.id_programmy
386 JOIN polzovatel pol ON tes.id_polzovatel = pol.id_polzovatel
387 WHERE pr.id_programmy = 8;

```

Data Output Messages Graph Visualiser × Notifications

	id_programm integer	id_tip smallint	chastota_pr_f character (3)	op_pamyat_p smallint	mesto_disk_f smallint	obem smallint	imya character varying	tip_programmy character varying (35)	data_razrabot date	fio_programmista character varying (50)	data_testirov date	fio_polzovatelya character varying (50)
1	8	2	1,7	5	512	128	Организатор	Прикладное программное обеспечение	2020-11-08	Иванов Иван Иванович	2022-12-02	Петров Семен Андреевич

✓ (65) Вывод названий типов программ вместе с количеством программ данного типа:



```

SELECT
tip.tip_programmy,
COUNT (pr.id_tip)
FROM tip_programmy tip
JOIN programma pr ON tip.id_tip = pr.id_tip
GROUP BY tip.tip_programmy;

```

```

389 SELECT
390 tip.tip_programmy,
391 COUNT (pr.id_tip)
392 FROM tip_programmy tip
393 JOIN programma pr ON tip.id_tip = pr.id_tip
394 GROUP BY tip.tip_programmy;

```

Data Output Messages Graph Visualiser × Notifications

	tip_programmy character varying (35)	count bigint
1	Прикладное программное обеспечение	5
2	Системное программное обеспечение	4

# Лабораторная работа №4. Программируемые объекты базы данных

## 1 Начало работы

Проведено знакомство со схемой БД, для которой написаны запросы-примеры.

## 2 Дополнительные возможности SELECT-запросов

### 2.1 Заполнение таблиц из выборки

- ✓ (1) Создание отдельной таблицы для хранения информации о программах, принадлежащих к типу системного программного обеспечения:

```
SELECT * INTO sys FROM programma WHERE id_tip = 1;  
SELECT * FROM sys;
```

```
389 SELECT  
390 tip.tip_programmy,  
391 COUNT (pr.id_tip)  
392 FROM tip_programmy tip  
393 JOIN programma pr ON tip.id_tip = pr.id_tip  
394 GROUP BY tip.tip_programmy;  
395  
396 SELECT * INTO sys FROM programma WHERE id_tip = 1;  
397 SELECT * FROM sys;
```

	id_programm integer	id_tip smallint	chastota_pr_ character (3)	op_pamyat_p smallint	mesto_disk_p smallint	obem smallint	imya character var
1	3	1	2,1	4	2048	256	Система
2	9	1	2,7	32	4096	256	Оператор
3	5	1	1,8	8	[null]	512	Оболочка
4	11	1	1,2	8	256	512	Структура

### 2.2 Объединение запросов

- ✓ (2) Объединение двух запросов с UNION, показывающих размер старой и новой средней требуемой оперативной памяти при увеличении памяти для системного программного обеспечения на 5%, а для прикладного программного обеспечения на 10%:

```
(SELECT  
tip.tip_programmy AS "Тип программы",  
FLOOR(AVG(pr.mesto_disk_prog)) AS "Старая средняя память",
```

```

FLOOR(AVG(pr.mesto_disk_prog)*1.05) AS "Новая средняя память"
FROM programma pr
JOIN tip_programmy tip ON tip.id_tip = pr.id_tip
WHERE pr.id_tip = 1
GROUP BY tip.tip_programmy)
UNION
(SELECT
tip.tip_programmy AS "Тип программы",
FLOOR(AVG(pr.mesto_disk_prog)) AS "Старая средняя память",
FLOOR(AVG(pr.mesto_disk_prog)*1.1) AS "Новая средняя память"
FROM programma pr
JOIN tip_programmy tip ON tip.id_tip = pr.id_tip
WHERE pr.id_tip = 2
GROUP BY tip.tip_programmy);

```

```

399 (SELECT
400 tip.tip_programmy AS "Тип программы",
401 FLOOR(AVG(pr.mesto_disk_prog)) AS "Старая средняя память",
402 FLOOR(AVG(pr.mesto_disk_prog)*1.05) AS "Новая средняя память"
403 FROM programma pr
404 JOIN tip_programmy tip ON tip.id_tip = pr.id_tip
405 WHERE pr.id_tip = 1
406 GROUP BY tip.tip_programmy)
407 UNION
408 (SELECT
409 tip.tip_programmy AS "Тип программы",
410 FLOOR(AVG(pr.mesto_disk_prog)) AS "Старая средняя память",
411 FLOOR(AVG(pr.mesto_disk_prog)*1.1) AS "Новая средняя память"
412 FROM programma pr
413 JOIN tip_programmy tip ON tip.id_tip = pr.id_tip
414 WHERE pr.id_tip = 2
415 GROUP BY tip.tip_programmy);

```

	Тип программы character varying (35)	Старая средняя память numeric	Новая средняя память numeric
1	Прикладное программное обеспечение	716	788
2	Системное программное обеспечение	2112	2217

- ✓ (3) Объединение двух запросов с UNION ALL; в выборку попали одинаковые значения количества программ, разработанных одним программистом:

```

(SELECT
prst.fio_programmista,
COUNT(ras.id_programmy)
FROM programmist prst
JOIN rasrabotka ras ON prst.id_programmista = ras.id_programmista
JOIN programma pr ON pr.id_programmy = ras.id_programmy

```

```

WHERE pr.id_tip = 1
GROUP BY prst.fio_programmista)
UNION
(SELECT
prst.fio_programmista,
COUNT (ras.id_programmy)
FROM programmist prst
JOIN rasrabotka ras ON prst.id_programmista = ras.id_programmista
JOIN programma pr ON pr.id_programmy = ras.id_programmy
WHERE pr.id_tip = 2
GROUP BY prst.fio_programmista);

```

```

482 (SELECT
483 prst.fio_programmista,
484 COUNT (ras.id_programmy)
485 FROM programmist prst
486 JOIN rasrabotka ras ON prst.id_programmista = ras.id_programmista
487 JOIN programma pr ON pr.id_programmy = ras.id_programmy
488 WHERE pr.id_tip = 1
489 GROUP BY prst.fio_programmista)
490 UNION ALL
491 (SELECT
492 prst.fio_programmista,
493 COUNT (ras.id_programmy)
494 FROM programmist prst
495 JOIN rasrabotka ras ON prst.id_programmista = ras.id_programmista
496 JOIN programma pr ON pr.id_programmy = ras.id_programmy
497 WHERE pr.id_tip = 2
498 GROUP BY prst.fio_programmista);

```

Data Output Messages Graph Visualiser × Notifications

	fio_programmista character varying (50)	count bigint
1	Сидоров Андрей Иванович	1
2	Иванов Иван Иванович	2
3	Петров Иван Семенович	2
4	Сидоров Андрей Иванович	1
5	Иванов Иван Иванович	3
6	Петров Иван Семенович	1

- ✓ (4) Объединение двух запросов в EXCEPT; в выборку попадали все строки, которые содержатся в результате первого запроса, но при этом отсутствуют в результате второго:

```

(SELECT
prst.fio_programmista,
COUNT (ras.id_programmy)
FROM programmist prst

```

```

JOIN rasrabotka ras ON prst.id_programmista = ras.id_programmista
JOIN programma pr ON pr.id_programmy = ras.id_programmy
WHERE pr.id_tip = 1
GROUP BY prst.fio_programmista)
EXCEPT
(SELECT
prst.fio_programmista,
COUNT (ras.id_programmy)
FROM programmist prst
JOIN rasrabotka ras ON prst.id_programmista = ras.id_programmista
JOIN programma pr ON pr.id_programmy = ras.id_programmy
WHERE pr.id_tip = 2
GROUP BY prst.fio_programmista);

```

```

482 (SELECT
483 prst.fio_programmista,
484 COUNT (ras.id_programmy)
485 FROM programmist prst
486 JOIN rasrabotka ras ON prst.id_programmista = ras.id_programmista
487 JOIN programma pr ON pr.id_programmy = ras.id_programmy
488 WHERE pr.id_tip = 1
489 GROUP BY prst.fio_programmista)
490 EXCEPT
491 (SELECT
492 prst.fio_programmista,
493 COUNT (ras.id_programmy)
494 FROM programmist prst
495 JOIN rasrabotka ras ON prst.id_programmista = ras.id_programmista
496 JOIN programma pr ON pr.id_programmy = ras.id_programmy
497 WHERE pr.id_tip = 2
498 GROUP BY prst.fio_programmista);

```

	fio_programmista character varying (50)	count bigint
1	Петров Иван Семенович	2
2	Иванов Иван Иванович	2

### 3 Представления

- ✓ (5) Создание временного представления Petrov с данными о разработанных программистом Петровым программах; после перезапуска сервера PostgreSQL повторная выборка данных из этого представления приводит к ошибке:

```

CREATE TEMP VIEW Petrov AS
SELECT pr.* FROM programma AS pr
WHERE pr.id_programmy IN (SELECT id_programmy FROM rasrabotka

```

WHERE id\_programmista = 6);

SELECT \* FROM Petrov;

```
513 CREATE TEMP VIEW Petrov AS
514 SELECT pr.* FROM programma AS pr
515 WHERE pr.id_programmy IN (SELECT id_programmy FROM rasrabotka WHERE id_programmista = 6);
516
517 SELECT * FROM Petrov;
```

Data Output Messages Graph Visualiser × Notifications

	id_programmy integer	id_tip smallint	chastota_pr_prog character (3)	op_pamyat_prog smallint	mesto_disk_prog smallint	obem smallint	imya character varying (20)
1	3	1	2,1	4	2048	256	Система
2	7	2	1,4	2	1024	64	Поиск
3	11	1	1,2	8	256	512	Структура

```
8 SELECT * FROM Petrov;
```

Data Output Messages Notifications

ERROR: ОШИБКА: отношение "petrov" не существует  
LINE 1: SELECT \* FROM Petrov;  
^

SQL state: 42P01

Character: 16

- ✓ (6) Создание постоянного представления Иванов с данными о разработанных программистом Ивановым программах с последующим запросом с использованием этого представления:

```
CREATE or REPLACE VIEW Ivanov AS
SELECT pr.* FROM programma AS pr
WHERE pr.id_programmy IN (SELECT id_programmy FROM rasrabotka
WHERE id_programmista = 5);
```

```
SELECT
iv.imya, ras.data_razrabotki, tip.tip_programmy, iv.chastota_pr_prog,
iv.op_pamyat_prog, iv.mesto_disk_prog
FROM Ivanov AS iv
JOIN rasrabotka AS ras ON iv.id_programmy = ras.id_programmy
JOIN tip_programmy AS tip ON tip.id_tip = iv.id_tip;
```

```

502 CREATE or REPLACE VIEW Ivanov AS
503 SELECT pr.* FROM programma AS pr
504 WHERE pr.id_programmy IN (SELECT id_programmy FROM rasrabotka WHERE id_programmista = 5);
505
506 SELECT
507 iv.imya, ras.data_razrabotki, tip.tip_programmy, iv.chastota_pr_prog,
508 iv.op_pamyat_prog, iv.mesto_disk_prog
509 FROM Ivanov AS iv
510 JOIN rasrabotka AS ras ON iv.id_programmy = ras.id_programmy
511 JOIN tip_programmy AS tip ON tip.id_tip = iv.id_tip;

```

Data Output Messages Graph Visualiser × Notifications

	imya character varying (35)	data_razrabotki date	tip_programmy character varying (35)	chastota_pr_ character (3)	op_pamyat_p smallint	mesto_disk_p smallint
1	Редактор	2021-05-14	Прикладное программное обеспечение	1,4	2	512
2	Оболочка	2020-02-01	Системное программное обеспечение	1,8	8	2048
3	Организер	2020-11-08	Прикладное программное обеспечение	1,7	4	512
4	Навигатор	2021-05-14	Прикладное программное обеспечение	1,9	8	1024
5	Оператор	2023-07-01	Системное программное обеспечение	2,7	32	4096

## 4 Общие табличные выражения

### 4.1 Общая информация

- ✓ (7) Вывод информации о программах, требуемый объем оперативной памяти для работы которых не превышает среднего объема требуемой оперативной памяти среди всех разработанных коллективом программ, каждая из которых написана программистом моложе 21 года:

```

WITH pamyat AS(
SELECT
pr.id_programmy,
pr.imya,
pr.op_pamyat_prog,
ras.id_programmista
FROM vozrast vs
JOIN rasrabotka ras ON vs.id_programmista = ras.id_programmista
JOIN programma pr ON ras.id_programmy = pr.id_programmy
WHERE pr.op_pamyat_prog < (SELECT AVG(pr.op_pamyat_prog) FROM
programma pr)
AND ras.id_programmista IN
(SELECT
prst.id_programmista
FROM programmist prst
WHERE (date_part('year', now()) - date_part('year', prst.data_rozhdeniya)) < 21))

SELECT

```

```

prst.fio_programmista,
pr.imya,
pr.chastota_pr_prog,
pr.op_pamyat_prog
FROM pamyat pam
JOIN programma pr ON pam.id_programmy = pr.id_programmy
JOIN rasrabotka ras ON ras.id_programmy = pr.id_programmy
JOIN programmist prst ON prst.id_programmista = ras.id_programmista

```

```

527 WITH pamyat AS(
528 SELECT
529 pr.id_programmy,
530 pr.imya,
531 pr.op_pamyat_prog,
532 ras.id_programmista
533 FROM vozrast vs
534 JOIN rasrabotka ras ON vs.id_programmista = ras.id_programmista
535 JOIN programma pr ON ras.id_programmy = pr.id_programmy
536 WHERE pr.op_pamyat_prog < (SELECT AVG(pr.op_pamyat_prog) FROM programma pr)
537 AND ras.id_programmista IN
538 (SELECT
539 prst.id_programmista
540 FROM programmist prst
541 WHERE (date_part('year', now()) - date_part('year', prst.data_rozhdeniya)) < 21))
542
543 SELECT
544 prst.fio_programmista,
545 pr.imya,
546 pr.chastota_pr_prog,
547 pr.op_pamyat_prog
548 FROM pamyat pam
549 JOIN programma pr ON pam.id_programmy = pr.id_programmy
550 JOIN rasrabotka ras ON ras.id_programmy = pr.id_programmy
551 JOIN programmist prst ON prst.id_programmista = ras.id_programmista

```

Data Output Messages Notifications						
	fio_programmista character varying (50)	id_programmy integer	imya character varying (20)	chastota_pr_prog character (3)	op_pamyat_prog smallint	
1	Петров Иван Семенович	3	Система	2,1	4	
2	Петров Иван Семенович	7	Поиск	1,4	2	
3	Петров Иван Семенович	11	Структура	1,2	8	
4	Сидоров Андрей Иванович	12	Схема	2,3	8	

## 4.2 Иерархические запросы

- ✓ (8) Вывод всех начальников сотрудника с именем «Сергей» рекурсивным Common Table Expressions:

```

WITH RECURSIVE example AS(
SELECT

```



```

t1.manager_id,
t1.empl_name,
t1.empl_id
FROM demoWith AS t1
WHERE empl_name = 'Сергей'
UNION
SELECT
t2.manager_id,
t2.empl_name,
t2.empl_id
FROM demoWith AS t2
JOIN example AS ex ON ex.manager_id = t2.empl_id)
SELECT * FROM example;

```

```

567 WITH RECURSIVE example AS(
568 SELECT
569 t1.manager_id,
570 t1.empl_name,
571 t1.empl_id
572 FROM demoWith AS t1
573 WHERE empl_name = 'Сергей'
574 UNION
575 SELECT
576 t2.manager_id,
577 t2.empl_name,
578 t2.empl_id
579 FROM demoWith AS t2
580 JOIN example AS ex ON ex.manager_id = t2.empl_id)
581 SELECT * FROM example;

```

Data Output Messages Notifications

	manager_id integer	empl_name character varying (20)	empl_id integer
1	6	Сергей	7
2	5	Федор	6
3	1	Петр	5
4	[null]	Иван	1

## 5 Введение в PL/PGSQL

### 5.1 Общая информация

Знакомство с PL/pgSQL.

## 5.2 Функции и процедуры

- ✓ (9) Вычисление синуса угла, вводимого в градусах:

```
CREATE OR REPLACE FUNCTION function1(x NUMERIC) RETURNS
NUMERIC
LANGUAGE plpgsql
AS $$
DECLARE
y NUMERIC:=0;
BEGIN
y := sind(x);
RETURN y;
END $$;
```

```
SELECT function1(45);
```

```
595 CREATE OR REPLACE FUNCTION function1(x NUMERIC) RETURNS NUMERIC
596 LANGUAGE plpgsql
597 AS $$
598 DECLARE
599 y NUMERIC:=0;
600 BEGIN
601 y := sind(x);
602 RETURN y;
603 END $$;
604
605 SELECT function1(45);
```

Data Output		Messages	Notifications
function1 numeric			
1	0.707106781186547		

- ✓ (10) Вывод списка программ, подходящих для тестирования на компьютере с заданными параметрами объема оперативной памяти и свободного места на диске:

```
CREATE OR REPLACE PROCEDURE pamyat1 (op_pamyat_in INT,
mesto_disk_in INT)
LANGUAGE plpgsql
AS $$
DECLARE
    op_pamyat NUMERIC:=0;
```

```

    mesto_disk NUMERIC:=0;
    imua CHAR(10):= NULL;
BEGIN
RAISE NOTICE 'Для тестирования на компьютере с заданными параметрами
подходят следующие программы:';
    FOR op_pamyat, mesto_disk, imua IN SELECT op_pamyat_prog,
mesto_disk_prog, imya FROM programma
    LOOP
        IF op_pamyat > $1 AND mesto_disk > $2
            THEN RAISE NOTICE '%', imua;
        END IF;
    END LOOP;
END $$;

```

```

614 CREATE OR REPLACE PROCEDURE pamyat1 (op_pamyat_in INT, mesto_disk_in INT)
615 LANGUAGE plpgsql
616 AS $$
617 DECLARE
618     op_pamyat NUMERIC:=0;
619     mesto_disk NUMERIC:=0;
620     imua CHAR(10):= NULL;
621 BEGIN
622     RAISE NOTICE 'Для тестирования на компьютере с заданными параметрами подходят следующие программы:';
623
624     FOR op_pamyat, mesto_disk, imua IN SELECT op_pamyat_prog, mesto_disk_prog, imya FROM programma
625     LOOP
626         IF op_pamyat > $1 AND mesto_disk > $2
627             THEN RAISE NOTICE '%', imua;
628         END IF;
629     END LOOP;
630
631 END $$;
632
633 CALL pamyat1 (4,256);

```

Data Output   Messages   Notifications

ЗАМЕЧАНИЕ: Для тестирования на компьютере с заданными параметрами подходят следующие программы:  
 ЗАМЕЧАНИЕ: Навигатор  
 ЗАМЕЧАНИЕ: Оболочка  
 ЗАМЕЧАНИЕ: Схема  
 CALL

Query returned successfully in 39 msec.

✓ (11) Реализация процедуры, создающей таблицу с тремя столбцами:

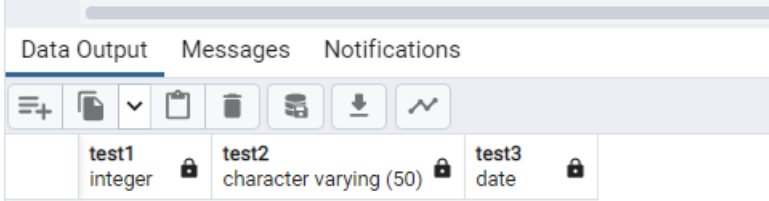
```

CREATE OR REPLACE PROCEDURE tabl()
LANGUAGE plpgsql
AS $$
BEGIN
CREATE TABLE test
(
test1 INT NOT NULL GENERATED ALWAYS AS IDENTITY,
test2 VARCHAR(50) NOT NULL,
test3 DATE NOT NULL

```

```
);
END $$;
CALL tabl();
```

```
538 CREATE OR REPLACE PROCEDURE tabl()
539 LANGUAGE plpgsql
540 AS $$
541 BEGIN
542 CREATE TABLE test
543 (
544 test1 INT NOT NULL GENERATED ALWAYS AS IDENTITY,
545 test2 VARCHAR(50) NOT NULL,
546 test3 DATE NOT NULL
547 );
548 END $$;
549 CALL tabl();
550 SELECT * FROM test;
```



test1 integer	test2 character varying (50)	test3 date
------------------	---------------------------------	---------------

- ✓ (12) Вывод на экран результатов умножения столбцов таблицы на величины, заданные во входных параметрах:

```
CREATE OR REPLACE PROCEDURE zadacha12 (multipl1 NUMERIC,
multipl2 NUMERIC)
LANGUAGE plpgsql
AS $$
DECLARE
    column1 NUMERIC:=0;
    column2 NUMERIC:=0;
BEGIN
    FOR column1 IN SELECT op_pamyat_prog FROM programma
    LOOP
        RAISE NOTICE '%', column1*multipl1;
    END LOOP;
    FOR column2 IN SELECT mesto_disk_prog FROM programma
    LOOP
        RAISE NOTICE '%', column2*multipl2;
    END LOOP;
END $$;
```

```

587 CREATE OR REPLACE PROCEDURE zadacha12 (multipl1 NUMERIC, multipl2 NUMERIC)
588 LANGUAGE plpgsql
589 AS $$
590 DECLARE
591     column1 NUMERIC:=0;
592     column2 NUMERIC:=0;
593 BEGIN
594     FOR column1 IN SELECT op_pamyat_prog FROM programma
595     LOOP
596         RAISE NOTICE '%', column1*multipl1;
597     END LOOP;
598     FOR column2 IN SELECT mesto_disk_prog FROM programma
599     LOOP
600         RAISE NOTICE '%', column2*multipl2;
601     END LOOP;
602 END $$;
603 CALL zadacha12 (1.5,2);

```

Data Output	Messages	Notifications
-------------	----------	---------------

ЗАМЕЧАНИЕ:	6.0
ЗАМЕЧАНИЕ:	12.0
ЗАМЕЧАНИЕ:	3.0
ЗАМЕЧАНИЕ:	12.0
ЗАМЕЧАНИЕ:	12.0
ЗАМЕЧАНИЕ:	6.0
ЗАМЕЧАНИЕ:	3.0
ЗАМЕЧАНИЕ:	12.0
ЗАМЕЧАНИЕ:	3.0
ЗАМЕЧАНИЕ:	3.0
ЗАМЕЧАНИЕ:	4096
ЗАМЕЧАНИЕ:	2048
ЗАМЕЧАНИЕ:	2048
ЗАМЕЧАНИЕ:	512
ЗАМЕЧАНИЕ:	4096
ЗАМЕЧАНИЕ:	1024
ЗАМЕЧАНИЕ:	1024
ЗАМЕЧАНИЕ:	2048
ЗАМЕЧАНИЕ:	8192
ЗАМЕЧАНИЕ:	1024
CALL	

## 5.3 Триггеры

- ✓ (13) Реализация триггера для аудита изменения данных таблицы «programma»:

```

CREATE OR REPLACE FUNCTION audit_function() RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    IF (NEW.op_pamyat_prog IS NOT NULL AND NEW.op_pamyat_prog <>
        OLD.op_pamyat_prog) THEN
        INSERT INTO audit(id_programmy,old_op_pamyat_prog,
            new_op_pamyat_prog, change)
            VALUES(OLD.id_programmy, OLD.op_pamyat_prog, NEW.op_pamyat_prog,
                now());

```

```
END IF;
IF (NEW.mesto_disk_prog IS NOT NULL AND NEW.mesto_disk_prog <>
OLD.mesto_disk_prog) THEN
    INSERT INTO audit(id_programmy,old_mesto_disk_prog,
new_mesto_disk_prog, change)
    VALUES(OLD.id_programmy, OLD.mesto_disk_prog,
NEW.mesto_disk_prog, now());
END IF;
RETURN NEW;
END $$;
```

```
CREATE TABLE audit
(id_audit INT NOT NULL GENERATED ALWAYS AS IDENTITY,
id_programmy INT,
old_op_pamyat_prog INT,
new_op_pamyat_prog INT,
old_mesto_disk_prog INT,
new_mesto_disk_prog INT,
change TIMESTAMP);
```

```
CREATE TRIGGER audit
AFTER INSERT OR UPDATE OR DELETE ON programma FOR EACH ROW
EXECUTE FUNCTION audit_function();
```

```
UPDATE programma SET op_pamyat_prog = 8, mesto_disk_prog = 2048
WHERE id_programmy = 3;
```

```
SELECT * FROM audit;
```

```

605 CREATE OR REPLACE FUNCTION audit_function() RETURNS TRIGGER
606 LANGUAGE plpgsql
607 AS $$
608 BEGIN
609     IF (NEW.op_pamyat_prog IS NOT NULL AND NEW.op_pamyat_prog <> OLD.op_pamyat_prog) THEN
610         INSERT INTO audit(id_programmy,old_op_pamyat_prog, new_op_pamyat_prog, change)
611         VALUES(OLD.id_programmy, OLD.op_pamyat_prog, NEW.op_pamyat_prog, now());
612     END IF;
613     IF (NEW.mesto_disk_prog IS NOT NULL AND NEW.mesto_disk_prog <> OLD.mesto_disk_prog) THEN
614         INSERT INTO audit(id_programmy,old_mesto_disk_prog, new_mesto_disk_prog, change)
615         VALUES(OLD.id_programmy, OLD.mesto_disk_prog, NEW.mesto_disk_prog, now());
616     END IF;
617     RETURN NEW;
618 END $$;
619
620 CREATE TABLE audit
621 (id_audit INT NOT NULL GENERATED ALWAYS AS IDENTITY,
622 id_programmy INT,
623 old_op_pamyat_prog INT,
624 new_op_pamyat_prog INT,
625 old_mesto_disk_prog INT,
626 new_mesto_disk_prog INT,
627 change TIMESTAMP);
628
629 CREATE TRIGGER audit
630 AFTER INSERT OR UPDATE OR DELETE ON programma FOR EACH ROW EXECUTE FUNCTION audit_function();
631
632 UPDATE programma SET op_pamyat_prog = 8, mesto_disk_prog = 2048 WHERE id_programmy = 3;
633
634 SELECT * FROM audit;

```

Data Output Messages Notifications							
	id_audit integer	id_programmy integer	old_op_pamyat_prog integer	new_op_pamyat_prog integer	old_mesto_disk_prog integer	new_mesto_disk_prog integer	change timestamp without time zone
1	1	3	4	8	[null]	[null]	2023-02-05 01:02:40.213198
2	2	3	[null]	[null]	1024	2048	2023-02-05 01:02:40.213198

- ✓ (14) Реализация триггера для аудита изменения данных таблицы «programmist»:

```

CREATE OR REPLACE FUNCTION audit_tel_function() RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    IF (NEW.telefon_dom IS NOT NULL AND NEW.telefon_dom <>
    OLD.telefon_dom) THEN
        INSERT INTO audit_telefon(id_programmista, old_telefon_dom,
        new_otelefon_dom, change)
        VALUES(OLD.id_programmista, OLD.telefon_dom, NEW.telefon_dom,
        now());
    END IF;
    IF (NEW.telefon_con IS NOT NULL AND NEW.telefon_con <>
    OLD.telefon_con) THEN
        INSERT INTO audit_telefon(id_programmista, old_telefon_con,
        new_otelefon_con, change)
        VALUES(OLD.id_programmista, OLD.telefon_con, NEW.telefon_con,

```

```
now());  
    END IF;  
    RETURN NEW;  
END $$;
```

```
CREATE TABLE audit_telefon  
(id_audit_tel INT NOT NULL GENERATED ALWAYS AS IDENTITY,  
id_programmista INT,  
old_telefon_dom CHAR(11),  
new_otelefon_dom CHAR(11),  
old_telefon_con CHAR(11),  
new_otelefon_con CHAR(11),  
change TIMESTAMP);
```

```
CREATE TRIGGER audit_telefon  
AFTER INSERT OR UPDATE OR DELETE ON programmist FOR EACH ROW  
EXECUTE FUNCTION audit_tel_function();
```

```
UPDATE programmist SET telefon_dom = 79112228822, telefon_con =  
79994447757 WHERE id_programmista = 7;
```

```
SELECT * FROM audit_telefon;
```



```

638 CREATE OR REPLACE FUNCTION audit_tel_function() RETURNS TRIGGER
639 LANGUAGE plpgsql
640 AS $$
641 BEGIN
642     IF (NEW.telefon_dom IS NOT NULL AND NEW.telefon_dom <> OLD.telefon_dom) THEN
643         INSERT INTO audit_telefon(id_programmista, old_telefon_dom, new_otelefon_dom, change)
644         VALUES(OLD.id_programmista, OLD.telefon_dom, NEW.telefon_dom, now());
645     END IF;
646     IF (NEW.telefon_con IS NOT NULL AND NEW.telefon_con <> OLD.telefon_con) THEN
647         INSERT INTO audit_telefon(id_programmista, old_telefon_con, new_otelefon_con, change)
648         VALUES(OLD.id_programmista, OLD.telefon_con, NEW.telefon_con, now());
649     END IF;
650     RETURN NEW;
651 END $$;
652
653 CREATE TABLE audit_telefon
654 (id_audit_tel INT NOT NULL GENERATED ALWAYS AS IDENTITY,
655 id_programmista INT,
656 old_telefon_dom CHAR(11),
657 new_otelefon_dom CHAR(11),
658 old_telefon_con CHAR(11),
659 new_otelefon_con CHAR(11),
660 change TIMESTAMP);
661
662 CREATE TRIGGER audit_telefon
663 AFTER INSERT OR UPDATE OR DELETE ON programmist FOR EACH ROW EXECUTE FUNCTION audit_tel_function();
664
665 UPDATE programmist SET telefon_dom = 79112228822, telefon_con = 79994447757 WHERE id_programmista = 7;
666
667 SELECT * FROM audit_telefon;

```

Data Output Messages Notifications

	id_audit_tel integer	id_programmista integer	old_telefon_dom character (11)	new_otelefon_dom character (11)	old_telefon_con character (11)	new_otelefon_con character (11)	change timestamp without time zone
1	1	7	71112223322	79112228822	[null]	[null]	2023-02-05 07:05:03.561859
2	2	7	[null]	[null]	73334448833	79994447711	2023-02-05 07:05:03.561859
3	3	7	71112223322	79112228822	[null]	[null]	2023-02-05 07:05:03.561859
4	4	7	[null]	[null]	73334448833	79994447711	2023-02-05 07:05:03.561859
5	5	7	[null]	[null]	79994447711	79994447757	2023-02-05 07:06:47.760977
6	6	7	[null]	[null]	79994447711	79994447757	2023-02-05 07:06:47.760977