

МИНОБРНАУКИ РОССИИ

**Федеральное государственное бюджетное образовательное учреждение
высшего образования**

«Тульский государственный университет»

Интернет-институт

ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ

по дисциплине

«Базы данных 2»

Семестр 6

Вариант 3

Выполнил: студент гр. ИБ262521-ф

Артемов Александр Евгеньевич

Проверил: канд. техн. наук, доц.

Французова Юлия Вячеславовна

Тула 2024

Лабораторная работа № 1.

Название работы: Запросы на объединение отношений.

Цели работы: Изучение и практическое применение запросов на объединение отношений.

Задание:

1. Напишите запрос, демонстрирующий объединение (UNION) результатов двух запросов. Количество выбираемых столбцов и типы данных соответствующих столбцов обоих запросов должны совпадать.
2. Напишите запрос, демонстрирующий объединение (UNION) результатов трех запросов. Тип данных одного столбца у всех трех запросов должны быть разными (но приводимыми, например, целые числа, действительные числа и строки). Эти типы данных должны быть указаны в дополнительном столбце результирующей выборки.
3. Дополните запрос из пункта 2 сортировкой по двум столбцам (сначала по возрастанию одного, затем по убыванию другого).

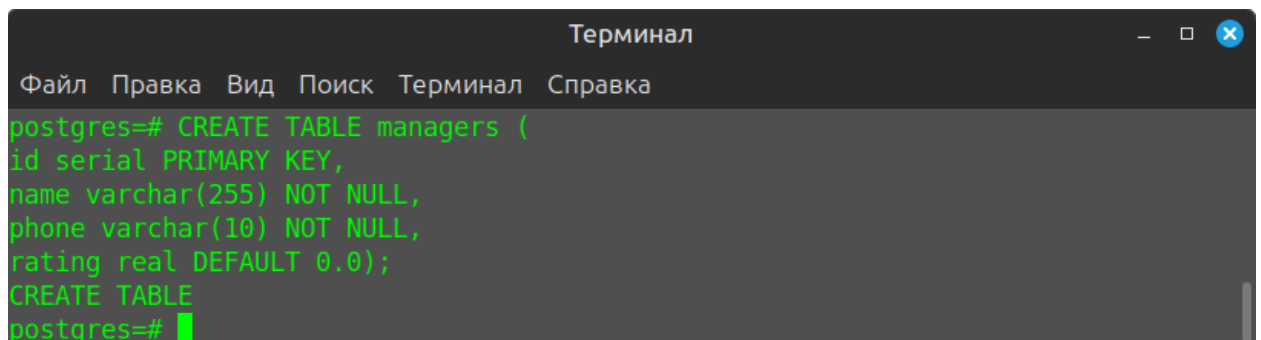
Выполнение лабораторной работы.

Изучены теоретические сведения лабораторной работы по объединению отношений.

1. Запрос, демонстрирующий объединение (UNION) результатов двух запросов. Количество выбираемых столбцов и типы данных соответствующих столбцов обоих запросов должны совпадать.

Для демонстрации запроса будут использованы две таблицы МЕНЕДЖЕРЫ (Managers) и КЛИЕНТЫ (Clients), имеющие одинаковые атрибуты Имя (Name) и Телефон (Phone), а также атрибут Рейтинг (Rating), имеющий в таблице менеджеров вещественный тип, а в таблице клиентов — целый.

Создадим таблицу Managers:



```
postgres=# CREATE TABLE managers (  
id serial PRIMARY KEY,  
name varchar(255) NOT NULL,  
phone varchar(10) NOT NULL,  
rating real DEFAULT 0.0);  
CREATE TABLE  
postgres=#
```

Создадим таблицу Clients:

```
Терминал
Файл Правка Вид Поиск Терминал Справка
postgres=# CREATE TABLE clients (
id serial PRIMARY KEY,
name varchar(255) NOT NULL,
phone varchar(10) NOT NULL,
rating integer DEFAULT 0);
CREATE TABLE
postgres=#
```

Заполним таблицы данными при помощи клиента pgAdmin. Таблица Managers:

	id [PK] integer	name character varying (255)	phone character varying (10)	rating real
1	1	Амина Золотова	0715507563	9.4
2	2	Ева Жукова	7389531048	2.3
3	3	Дмитрий Семенов	5352918907	8.8
4	4	Тимур Добрынин	7581376110	4.5
5	5	София Ларина	2257997770	6.5
6	6	Милана Семенова	0811582503	9.1
7	7	Дарья Исаева	6264329635	3.7
8	8	Александр Назаров	5841025204	1.8
9	9	Фёдор Матвеев	0817284876	5.5
10	10	Полина Свешникова	1546157250	2.6

Таблица Clients:

	id [PK] integer	name character varying (255)	phone character varying (10)	rating integer
1	5	Василиса Кузнецова	4027795052	4
2	6	Сафия Киселева	4131526834	3
3	7	Елизавета Медведева	1228273462	8
4	8	Варвара Скворцова	1952274663	10
5	9	Арсений Федотов	9441738468	9
6	10	Майя Акимова	9558207608	8
7	11	Екатерина Киселева	1075979952	1
8	12	Мария Иванова	4900906128	2
9	13	Вадим Левин	0901879343	6
10	14	Матвей Киселев	6016706843	2
11	15	Захар Морозов	8185285561	7
12	16	Милана Яшина	7981630514	8
13	17	Вера Абрамова	3652433858	5
14	18	Анастасия Родионова	5206719929	10

Выполним запрос, отображающий список клиентов и менеджеров с рейтингом большим 5.

```
SELECT name, phone FROM managers WHERE rating > 5
UNION SELECT name, phone FROM clients WHERE rating > 5.0;
```

```
Терминал
Файл Правка Вид Поиск Терминал Справка
postgres=# SELECT name, phone FROM managers WHERE rating > 5 UNION SELECT name,
phone FROM clients WHERE rating > 5.0;
      name      |      phone
-----+-----
Варвара Скворцова | 1952274663
Захар Морозов    | 8185285561
Милана Семенова  | 0811582503
Фёдор Матвеев    | 0817284876
Амина Золотова   | 0715507563
Вадим Левин      | 0901879343
Арсений Федотов  | 9441738468
Елизавета Медведева | 1228273462
Анастасия Родионова | 5206719929
София Ларина     | 2257997770
Майя Акимова     | 9558207608
Дмитрий Семенов  | 5352918907
Милана Яшина     | 7981630514
(13 rows)

postgres=#
```

2. Запрос, демонстрирующий объединение (UNION) результатов трех запросов. Тип данных одного столбца у всех трех запросов должны быть разными (но приводимыми, например, целые числа, действительные числа и строки). Эти типы данных должны быть указаны в дополнительном столбце результирующей выборки.

Для демонстрации запроса используем таблицы из 1-го задания. Выполним запрос, отображающий список клиентов и менеджеров с рейтингом большим 9 и менеджеров с рейтингом меньше 3.

```
SELECT name, phone, rating FROM managers WHERE rating > 9
UNION SELECT name, phone, rating FROM clients WHERE rating > 9.0
UNION SELECT name, phone, rating FROM clients WHERE rating < 3.0;
```

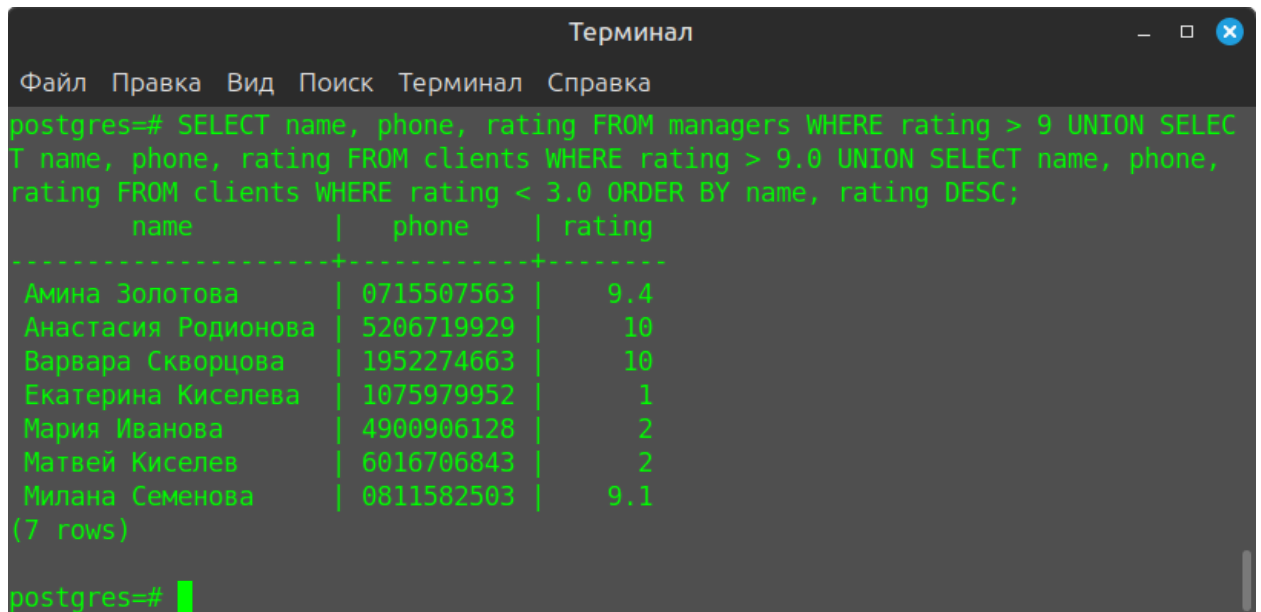
```
Терминал
Файл Правка Вид Поиск Терминал Справка
postgres=# SELECT name, phone, rating FROM managers WHERE rating > 9 UNION SELEC
T name, phone, rating FROM clients WHERE rating > 9.0 UNION SELECT name, phone,
rating FROM clients WHERE rating < 3.0;
      name      |      phone      | rating
-----+-----+-----
Амина Золотова   | 0715507563 | 9.4
Анастасия Родионова | 5206719929 | 10
Варвара Скворцова | 1952274663 | 10
Мария Иванова    | 4900906128 | 2
Екатерина Киселева | 1075979952 | 1
Милана Семенова  | 0811582503 | 9.1
Матвей Киселев   | 6016706843 | 2
(7 rows)

postgres=#
```

3. Дополнение запрос из пункта 2 сортировкой по двум столбцам (сначала по возрастанию одного, затем по убыванию другого).

Для демонстрации запроса используем таблицы из 2-го задания. Добавляем сортировку по возрастанию для столбца имен и по убыванию для рейтинга.

```
SELECT name, phone, rating FROM managers WHERE rating > 8  
UNION SELECT name, phone, rating FROM clients WHERE rating > 8.0  
UNION SELECT name, phone, rating FROM clients WHERE rating < 3.0  
ORDER BY name, rating DESC;
```



The screenshot shows a terminal window titled "Терминал" with a menu bar containing "Файл", "Правка", "Вид", "Поиск", "Терминал", and "Справка". The terminal displays a PostgreSQL query and its results. The query is: `postgres=# SELECT name, phone, rating FROM managers WHERE rating > 9 UNION SELECT name, phone, rating FROM clients WHERE rating > 9.0 UNION SELECT name, phone, rating FROM clients WHERE rating < 3.0 ORDER BY name, rating DESC;` The results are displayed in a table with three columns: name, phone, and rating. The table contains 7 rows of data, sorted by name and rating in descending order. The results are: Амина Золотова (0715507563, 9.4), Анастасия Родионова (5206719929, 10), Варвара Скворцова (1952274663, 10), Екатерина Киселева (1075979952, 1), Мария Иванова (4900906128, 2), Матвей Киселев (6016706843, 2), and Милана Семенова (0811582503, 9.1). The terminal also shows "(7 rows)" and the prompt "postgres=#" with a cursor.

```
postgres=# SELECT name, phone, rating FROM managers WHERE rating > 9 UNION SELECT  
T name, phone, rating FROM clients WHERE rating > 9.0 UNION SELECT name, phone,  
rating FROM clients WHERE rating < 3.0 ORDER BY name, rating DESC;  
      name      |      phone      | rating  
-----+-----+-----  
Амина Золотова  | 0715507563      | 9.4  
Анастасия Родионова | 5206719929      | 10  
Варвара Скворцова | 1952274663      | 10  
Екатерина Киселева | 1075979952      | 1  
Мария Иванова   | 4900906128      | 2  
Матвей Киселев  | 6016706843      | 2  
Милана Семенова | 0811582503      | 9.1  
(7 rows)  
postgres=#
```

Ответы на контрольные вопросы.

1. Вывести список авторов-женщин, работающих в жанре романа, но не в жанре фантастики:

```
CREATE TABLE Автор (  
    Код_Автора      INT,  
    Фамилия         VARCHAR(50),  
    Пол             VARCHAR(50) NOT NULL  
);  
  
CREATE TABLE Книга (  
    Код_Книги INT,  
    название VARCHAR(50) NOT NULL,  
    Тематика VARCHAR(50) NOT NULL,  
    Издательство VARCHAR(50) NOT NULL,  
    Код_Автора INT NOT NULL  
);
```

Ответ:

```
SELECT Автор.Фамилия, Автор.Код_Автора  
FROM Автор  
WHERE Автор.Пол='ж' AND Автор.Код_Автора IN (  
    SELECT Книга.Код_Автора  
    FROM Книга  
    WHERE Книга.Тематика='Роман')  
AND Автор.Код_Автора NOT IN (  
    SELECT Книга.Код_Автора  
    FROM Книга  
    WHERE Книга.Тематика='Фантастика');
```

2. Даны таблицы Город и Разговор:

```
CREATE TABLE Город (  
    Код_Города INT,  
    Название VARCHAR(20) NOT NULL,  
    Тариф MONEY);  
  
CREATE TABLE Разговор (  
    Код_Разговора INT,  
    Код_Города INT NOT NULL  
    Фамилия VARCHAR(20)  
    Дата DATE TIME NOT NULL,  
    Продолжительность INT NOT NULL  
);
```

Вывести список абонентов, которые говорили с Москвой в апреле.

Ответ:

```
SELECT Разговор.Фамилия, Город.Название  
FROM Город, Разговор  
WHERE Город.Код_Города = Разговор.Код_Города  
AND Город.Название='Москва' AND Month(Разговор.Дата)=4;
```

3. Даны таблицы Рейс и Билет:

```
CREATE TABLE Рейс (  
    Номер_рейса INT,  
    Конечный_пункт VARCHAR(30),  
    Дата_вылета DATETIME);  
  
CREATE TABLE БИЛЕТ (  
    Номер_места CHAR(3),  
    Номер_рейса CHAR(6),  
    Дата_продажи DATETIME,  
    Фамилия_пассажира VARCHAR(30)  
);
```

Определить номера мест и дату продажи билетов на рейсы до Москвы с датой вылета 1 мая 2004 года.

Ответ:

```
SELECT Билет.Номер_места, Билет.Дата_продажи  
FROM Билет, Рейс  
WHERE Билет.Номер_рейса = Рейс.Номер_рейса  
AND Рейс.Конечный_пункт='Москва'  
AND Рейс.Дата_вылета='05/01/2004';
```

4. Даны таблицы Автор и Книга:

```
CREATE TABLE Автор (  
    Код_Автора INT ,  
    Фамилия VARCHAR(50),  
    Пол VARCHAR(50) NOT NULL  
);  
  
CREATE TABLE Книга (  
    Код_Книги INT,  
    Название VARCHAR(50) NOT NULL,  
    Тематика VARCHAR(50) NOT NULL,  
    Издательство VARCHAR(50) NOT NULL,  
    Код_Автора INT NOT NULL  
);
```

Вывести список авторов, работающих в жанре детектив.

Ответ:

```
SELECT DISTINCT Автор.Фамилия  
FROM Автор, Книга  
WHERE Автор.Код_Автора = Книга.Код_Автора  
AND Книга.Тематика ='Детектив';
```

5. Даны таблицы:

```
CREATE TABLE Блюдо (  
    Название_блюда VARCHAR(20) NOT NULL,  
    Время_приготовления INT NOT NULL,  
    Общая_калорийность INT NOT NULL,  
    Номер_рецепта INT,  
    Повар VARCHAR(20),
```

```

        Стоимость INT
    );
CREATE TABLE Компонент (
    Название_компонента VARCHAR(20),
    Калорийность INT NOT NULL,
    Жиры INT,
    Белки INT,
    Блюдо VARCHAR(20),
    Углеводы INT,
    Стоимость_100_грамм FLOAT NOT NULL
);

```

Сформировать список поваров, которые используют масло, но обходятся без молока.

Ответ:

```

SELECT Блюдо.Повар
FROM Блюдо
WHERE Блюдо.Повар IN (
    SELECT Блюдо.Повар
    FROM Блюдо
    INNER JOIN Компонент
    ON Блюдо.Название_блюда = Компонент.Блюдо
    WHERE Компонент.Название_компонента="Масло")
AND Блюдо.Повар NOT IN(
    SELECT Блюдо.Повар
    FROM Блюдо
    INNER JOIN Компонент
    ON Блюдо.Название_блюда = Компонент.Блюдо
    WHERE Компонент.Название_компонента="Молоко");

```


Лабораторная работа № 2.

Название работы: Запросы на соединение отношений.

Цели работы: Изучение и практическое применение запросов на соединение отношений.

Задание:

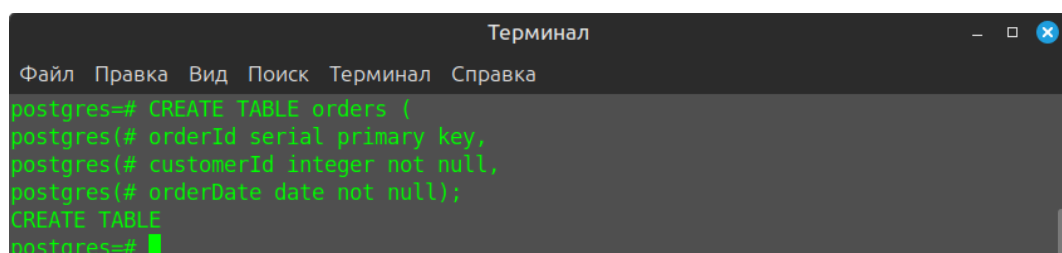
1. Напишите запрос, демонстрирующий соединение двух таблиц с помощью конструкции `SELECT ... FROM TABLE1, TABLE2 WHERE ...`. Перепишите тот же запрос с помощью конструкции `JOIN`. Убедитесь, что результаты выполнения запросов одинаковы.
2. Напишите запрос, демонстрирующий смысл и назначение конструкции `LEFT JOIN`. Перепишите его с помощью конструкции `RIGHT JOIN`. Убедитесь, что результаты выполнения запросов одинаковы.
3. Напишите запрос, в котором таблица соединяется (`JOIN`) сама с собой.
4. Напишите запрос, в котором агрегация происходит по результату соединения таблиц. То есть, в запросе должны присутствовать агрегирующая функция (`SUM`, `AVG`, `MAX`, `MIN` или `COUNT`), `GROUP BY` и `HAVING`, `WHERE` и `JOIN` (внутренний или внешний). Будьте внимательны к этому заданию, оно высоко оценивается.

Выполнение лабораторной работы.

Изучены теоретические сведения лабораторной работы по соединению отношений.

1. Запрос, демонстрирующий соединение двух таблиц с помощью конструкции `SELECT ... FROM TABLE1, TABLE2 WHERE ...`. Тот же запрос с помощью конструкции `JOIN`. Сравнение результатов выполнения запросов.

Для демонстрации соединения используем две таблицы — таблица **ЗАКАЗЫ** (`orders`) и **КЛИЕНТЫ** (`customers`). Таблица заказов имеет такие атрибуты: уникальный идентификатор заказа (`orderId`), идентификатор клиента (`customerId`), дата заказа (`orderDate`). Таблица клиентов имеет такие атрибуты: уникальный идентификатор клиента (`customerId`), наименование клиента (`customerName`) и контактное лицо (`contactName`). Выполним создание таблиц. Таблица `orders`:



```
Терминал
Файл  Правка  Вид  Поиск  Терминал  Справка
postgres=# CREATE TABLE orders (
postgres=#   orderId serial primary key,
postgres=#   customerId integer not null,
postgres=#   orderDate date not null);
CREATE TABLE
postgres=#
```

Таблица customers:

```
Терминал
Файл Правка Вид Поиск Терминал Справка
postgres=# CREATE TABLE customers (
customerid serial primary key,
customerName varchar(255) not null,
contactName varchar(255) not null);
CREATE TABLE
postgres=#
```

Заполним таблицы данными при помощи клиента pgAdmin. Таблица customers:

	customerid [PK] integer	customername character varying (255)	contactname character varying (255)
1	1	Редиска Менеджмент	Пётр Петров
2	2	Рога и копыта	Семён Семёнов
3	3	Балалайка Сервис	Иван Иванов

Таблица orders:

	orderid [PK] integer	customerid integer	orderdate date
1	1	2	2024-07-25
2	2	5	2024-06-20
3	3	3	2024-05-10

Соединим таблицу ЗАКАЗЫ с таблицей КЛИЕНТЫ при помощи конструкции SELECT, заменив столбец customerId столбцами customerName и contactName:

```
Терминал
Файл Правка Вид Поиск Терминал Справка
postgres=# SELECT orders.orderId, customers.customerName, customers.contactName, orders.orderDate
postgres-# FROM orders, customers
postgres-# WHERE orders.customerId = customers.customerId;
orderid | customername | contactname | orderdate
-----+-----+-----+-----
1 | Рога и копыта | Семён Семёнов | 2024-07-25
3 | Балалайка Сервис | Иван Иванов | 2024-05-10
(2 rows)
postgres=#
```

Выполним соединение таблиц при помощи конструкции JOIN и сравним результаты:

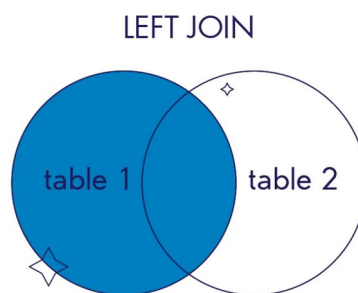
```
Терминал
Файл Правка Вид Поиск Терминал Справка
postgres=# SELECT orders.orderId, customers.customerName, customers.contactName, orders.orderDate
postgres=# FROM orders
postgres=# JOIN customers ON orders.customerId = customers.customerId;
 orderId | customername | contactname | orderdate
-----+-----+-----+-----
      1 | Рога и копыта | Семён Семёнов | 2024-07-25
      3 | Балалайка Сервис | Иван Иванов | 2024-05-10
(2 rows)

postgres=#
```

Как видим. Результаты соединений идентичны.

2. Запрос, демонстрирующий смысл и назначение конструкции LEFT JOIN. Перепишите его с помощью конструкции RIGHT JOIN. Убедитесь, что результаты выполнения запросов одинаковы.

В результат соединения при использовании конструкции LEFT JOIN попадут все записи из левой таблицы, даже если не будет ни одного совпадения с правой, а так же записи из правой таблицы, для которых выполняется условие объединения. Рисунок ниже демонстрирует смысл соединения при использовании конструкции LEFT JOIN:

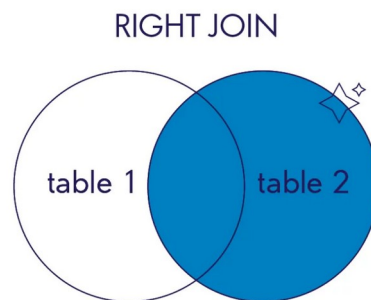


Будем считать, что левая таблица — ЗАКАЗЫ, а правая — КЛИЕНТЫ. Это значит, что в результат соединения попадут все записи из таблицы заказов, но как мы видим в таблице клиентов отсутствует запись с идентификатором клиента равным 5, поэтому в записи соединения, содержащей идентификатор клиента равный 5, значения имени клиента и контактного лица будут иметь значение null (или пустое значение).

```
Терминал
Файл Правка Вид Поиск Терминал Справка
postgres=# SELECT orders.orderId, orders.customerId, customers.customerName, customers.contactName, orders.orderDate
postgres=# FROM orders
postgres=# LEFT JOIN customers ON orders.customerId = customers.customerId;
 orderId | customerId | customername | contactname | orderdate
-----+-----+-----+-----+-----
      1 |          2 | Рога и копыта | Семён Семёнов | 2024-07-25
      2 |          5 |                |                | 2024-06-20
      3 |          3 | Балалайка Сервис | Иван Иванов | 2024-05-10
(3 rows)

postgres=#
```

При использовании конструкции RIGHT JOIN в результат соединения попадают все записи из правой таблицы, а так же записи из левой таблицы, для которых выполняется условие объединения:



Для получения аналогичного результата при помощи конструкции RIGHT JOIN необходимо поменять таблицы местами:

```

Терминал
Файл  Правка  Вид  Поиск  Терминал  Справка
postgres=# SELECT orders.orderId, orders.customerId, customers.customerName, customers.contactName, orders.orderDate
postgres=# FROM customers
postgres=# RIGHT JOIN orders ON orders.customerId = customers.customerId;
 orderId | customerId | customername | contactname | orderdate
-----+-----+-----+-----+-----
      1  |          2 | Рога и копыта | Семён Семёнов | 2024-07-25
      2  |          5 |                |                | 2024-06-20
      3  |          3 | Балалайка Сервис | Иван Иванов | 2024-05-10
(3 rows)
postgres=#

```

3. Запрос, в котором таблица соединяется (JOIN) сама с собой.

Такое соединение используют, когда в запросе нужно соединить несколько записей из одной и той же таблицы, но в одном запросе нельзя дважды использовать имя одной и той же таблицы: иначе запрос вернет ошибку. Поэтому, чтобы выполнить соединение таблицы с самой собой, в запросе ей присваивают два разных временных имени — алиаса.

```

Терминал
Файл  Правка  Вид  Поиск  Терминал  Справка
postgres=# SELECT c1.customername, c2.contactname FROM customers c1
postgres=# JOIN customers c2 ON c1.customerId = c2.customerId;
 customername | contactname
-----+-----
Редиска Менеджмент | Пётр Петров
Рога и копыта      | Семён Семёнов
Балалайка Сервис   | Иван Иванов
(3 rows)
postgres=#

```

4. Запрос, в котором агрегация происходит по результату соединения таблиц. То есть, в запросе должны присутствовать агрегирующая функция (SUM, AVG, MAX, MIN или COUNT), GROUP BY и HAVING, WHERE и JOIN (внутренний или внешний).

Добавим побольше записей в таблицу заказов:

	orderid [PK] integer	customerid integer	orderdate date
1	5	1	2024-08-09
2	10	1	2024-07-15
3	7	1	2024-07-20
4	1	2	2024-07-25
5	6	2	2024-07-01
6	9	2	2024-08-10
7	3	3	2024-05-10
8	8	3	2024-06-08
9	11	3	2024-07-02
10	4	3	2024-09-10
11	2	5	2024-06-20

Выполним запрос, показывающий клиентов, которые сделали в июле более чем 1 заказ:

```
Терминал
Файл  Правка  Вид  Поиск  Терминал  Справка
postgres=# SELECT customers.customerName, COUNT(customers.customerId)
postgres=# FROM orders
postgres=# INNER JOIN customers ON orders.customerId = customers.customerId
postgres=# WHERE EXTRACT(MONTH FROM orders.orderDate) = 7
postgres=# GROUP BY customers.customerName
postgres=# HAVING COUNT(customers.customerId) > 1;
 customername | count
-----+-----
Редиска Менеджмент |      2
Рога и копыта    |      2
(2 rows)

postgres=#
```

В запросе выполняется группировка клиентов по имени для подсчета количества их заказов, выполняется фильтрация по дате заказа, а именно заказы, выполненные в июле, и фильтрация по количеству заказов (более 1).

Ответы на контрольные вопросы.

1. Найти абонентов, которые звонят в Москву, но ни разу не звонили в Самару в мае:

```
CREATE TABLE Город (  
    Код_Города INT,  
    Название VARCHAR(20) NOT NULL,  
    Тариф MONEY);  
  
CREATE TABLE Разговор (  
    Код_Разговора INT ,  
    Фамилия VARCHAR(20),  
    Дата DATE TIME NOT NULL,  
    Продолжительность INT NOT NULL);
```

Ответ:

```
SELECT DISTINCT Разговор.Фамилия  
FROM Разговор  
WHERE Разговор.Фамилия IN (  
    SELECT Разговор.Фамилия  
    FROM Город  
    INNER JOIN Разговор  
    ON Город.Код_Города = Разговор.Код_Города  
    WHERE Город.Название='Москва')  
AND Разговор.Фамилия NOT IN (  
    SELECT Разговор.Фамилия  
    FROM Город  
    INNER JOIN Разговор  
    ON Город.Код_Города = Разговор.Код_Города  
    WHERE Город.Название='Самара'  
    AND Month(Разговор.Дата)=5  
);
```

2. Вывести список авторов-женщин, работающих в жанре романа, но не в жанре фантастики:

```
CREATE TABLE Автор (  
    Код_Автора INT,  
    Фамилия VARCHAR(50),  
    Пол VARCHAR(50) NOT NULL  
);  
  
CREATE TABLE Книга (  
    Код_Книги INT,  
    название VARCHAR(50) NOT NULL,  
    Тематика VARCHAR(50) NOT NULL,  
    Издательство VARCHAR(50) NOT NULL,  
    Код_Автора INT NOT NULL  
);
```

Ответ:

```
SELECT Автор.Фамилия, Автор.Код_Автора  
FROM Автор
```

```

WHERE Автор.Пол='ж' AND Автор.Код_Автора IN (
    SELECT Книга.Код_Автора
    FROM Книга
    WHERE Книга.Тематика='Роман')
AND Автор.Код_Автора NOT IN (
    SELECT Книга.Код_Автора
    FROM Книга
    WHERE Книга.Тематика='Фантастика');

```

3. Даны таблицы Город и Разговор:

```

CREATE TABLE Город (
    Код_Города INT,
    Название VARCHAR(20) NOT NULL,
    Тариф MONEY);

CREATE TABLE Разговор (
    Код_Разговора INT,
    Код_Города INT NOT NULL
    Фамилия VARCHAR(20)
    Дата DATE TIME NOT NULL,
    Продолжительность INT NOT NULL
);

```

Вывести список абонентов, которые говорили с Москвой в апреле.

Ответ:

```

SELECT Разговор.Фамилия, Город.Название
FROM Город, Разговор
WHERE Город.Код_Города = Разговор.Код_Города
AND Город.Название='Москва' AND Month(Разговор.Дата)=4;

```

4. Даны таблицы Рейс и Билет:

```

CREATE TABLE Рейс (
    Номер_рейса INT,
    Конечный_пункт VARCHAR(30),
    Дата_вылета DATETIME);

CREATE TABLE БИЛЕТ (
    Номер_места CHAR(3),
    Номер_рейса CHAR(6),
    Дата_продажи DATETIME,
    Фамилия_пассажира VARCHAR(30)
);

```

Определить номера мест и дату продажи билетов на рейсы до Москвы с датой вылета 1 мая 2004 года.

Ответ:

```

SELECT Билет.Номер_места, Билет.Дата_продажи
FROM Билет, Рейс
WHERE Билет.Номер_рейса = Рейс.Номер_рейса
AND Рейс.Конечный_пункт='Москва'
AND Рейс.Дата_вылета='05/01/2004';

```

5. Даны таблицы Автор и Книга:

```
CREATE TABLE Автор (  
    Код_Автора INT ,  
    Фамилия VARCHAR(50),  
    Пол VARCHAR(50) NOT NULL  
);  
  
CREATE TABLE Книга (  
    Код_Книги INT,  
    Название VARCHAR(50) NOT NULL,  
    Тематика VARCHAR(50) NOT NULL,  
    Издательство VARCHAR(50) NOT NULL,  
    Код_Автора INT NOT NULL  
);
```

Вывести список авторов, работающих в жанре детектив.

Ответ:

```
SELECT DISTINCT Автор.Фамилия  
FROM Автор, Книга  
WHERE Автор.Код_Автора = Книга.Код_Автора  
AND Книга.Тематика = 'Детектив';
```


Лабораторная работа № 3.

Название работы: Подзапросы.

Цели работы: Изучение и практическое применение подзапросов.

Задание:

1. Напишите пример запроса, в котором вместо любой из констант выражения, определяющего условие WHERE, используется скалярный подзапрос.
2. Напишите запрос с векторным подзапросом (ключевое слово IN).
3. Напишите два запроса с ключевыми словами ANY и ALL, делающие одно и то же.
4. Напишите пример запроса с табличным подзапросом (ключевое слово EXISTS).
5. Напишите пример запроса несколькими уровнями вложенности.
6. Напишите пример запроса, в котором вместо таблицы, указываемой после ключевого слова FROM, используется подзапрос.
7. Напишите пример связанного подзапроса.

Выполнение лабораторной работы.

Изучены теоретические сведения лабораторной работы по выполнению подзапросов. Для демонстрации используем таблицы ЗАКАЗЫ (orders) и КЛИЕНТЫ (customers) из Лабораторной работы №2.

1. Пример запроса, в котором вместо любой из констант выражения, определяющего условие WHERE, используется скалярный подзапрос.

Скалярный подзапрос - это обычный запрос SELECT в скобках, который возвращает ровно одну строку и один столбец. После выполнения запроса SELECT его единственный результат используется в окружающем его выражении. В качестве скалярного подзапроса нельзя использовать запросы, возвращающие более одной строки или столбца.

Пусть требуется запрос возвращающий имя клиента, контактное лицо и дату последнего созданного заказа. В качестве подзапроса используем конструкцию SELECT возвращающую дату последнего созданного заказа:

```
SELECT max(orderdate) FROM orders;
```

Согласно данных таблицы orders дата последнего созданного заказа — 2024-09-10, а идентификатор клиента создавшего заказ равен 3. Согласно данных таблицы customers клиент с идентификатором 3 — это «Балалайка Сервис», а контактное лицо - «Иван Иванов».

```
Терминал
Файл Правка Вид Поиск Терминал Справка
postgres=# SELECT customers.customername, customers.contactname, orders.orderdate
postgres=# FROM customers, orders
postgres=# WHERE orders.customerid = customers.customerid
postgres=# AND orders.orderdate = (SELECT max(orderdate) FROM orders);
 customername | contactname | orderdate
-----+-----+-----
Балалайка Сервис | Иван Иванов | 2024-09-10
(1 row)

postgres=#
```

2. Запрос с векторным подзапросом (ключевое слово IN).

Пусть требуется запрос возвращающий имя и контактное лицо клиентов, создавших заказы, например, в августе. В качестве подзапроса используем конструкцию `SELECT` возвращающую идентификаторы клиентов, создавших заказы в августе:

```
SELECT DISTINCT orders.customerid
FROM orders WHERE
EXTRACT(MONTH FROM orders.orderdate) = 8);
```

Конструкция `DISTINCT` необходимо для удаления дубликатов в случае если клиент сделал несколько заказов в месяце.

Согласно данных таблицы `orders` создали заказы в августе клиенты с идентификатором 1 и 2. Согласно данных таблицы `customers` клиенты с идентификаторами 1 и 2 — это «Редиска Менеджмент» и «Рога и копыта», а контактные лица - «Пётр Петров» и «Семён Семёнов» соответственно.

```
Терминал
Файл Правка Вид Поиск Терминал Справка
postgres=# SELECT customers.customername, customers.contactname
postgres=# FROM customers
postgres=# WHERE customers.customerid IN (
postgres=# SELECT DISTINCT orders.customerid
postgres=# FROM orders
postgres=# WHERE EXTRACT(MONTH FROM orders.orderdate) = 8);
 customername | contactname
-----+-----
Редиска Менеджмент | Пётр Петров
Рога и копыта | Семён Семёнов
(2 rows)

postgres=#
```

3. Запросы с ключевыми словами ANY и ALL, делающие одно и то же.

Запрос с ключевым словом `ANY`, возвращающий имя клиента, контактное лицо и дату заказов, сделанных после 01 августа 2024 года.

```
Терминал
Файл Правка Вид Поиск Терминал Справка
postgres=# SELECT customers.customername, customers.contactname, orders.orderdate
postgres=# FROM customers, orders
postgres=# WHERE customers.customerid = orders.customerid
postgres=# AND orders.orderdate >= ANY(
postgres=# SELECT orders.orderdate FROM orders WHERE orders.orderdate > '2024-08-01');
 customername | contactname | orderdate
-----+-----+-----
Балалайка Сервис | Иван Иванов | 2024-09-10
Редиска Менеджмент | Пётр Петров | 2024-08-09
Рога и копыта | Семён Семёнов | 2024-08-10
(3 rows)

postgres=#
```

Запрос с ключевым словом ALL, возвращающий имя клиента, контактное лицо и дату заказов, сделанных после 01 августа 2024 года.

```
Терминал
Файл Правка Вид Поиск Терминал Справка
postgres=# SELECT customers.customername, customers.contactname, orders.orderdate
FROM customers, orders
WHERE customers.customerid = orders.customerid
AND orders.orderdate > ALL(
SELECT orders.orderdate FROM orders WHERE orders.orderdate < '2024-08-01');
 customername | contactname | orderdate
-----+-----+-----
Балалайка Сервис | Иван Иванов | 2024-09-10
Редиска Менеджмент | Пётр Петров | 2024-08-09
Рога и копыта | Семён Семёнов | 2024-08-10
(3 rows)

postgres=#
```

4. Пример запроса с табличным подзапросом (ключевое слово EXISTS).

Запрос возвращает все записи таблицы customers, если в ней есть записи.

```
Терминал
Файл Правка Вид Поиск Терминал Справка
postgres=# SELECT * FROM customers WHERE EXISTS(SELECT * FROM customers);
 customerid | customername | contactname
-----+-----+-----
1 | Редиска Менеджмент | Пётр Петров
2 | Рога и копыта | Семён Семёнов
3 | Балалайка Сервис | Иван Иванов
(3 rows)

postgres=#
```

5. Пример запроса несколькими уровнями вложенности.

Запрос возвращает имя клиента, сделавшего заказ в указанную дату (01 июля 2024 года).

```
Терминал
Файл  Правка  Вид  Поиск  Терминал  Справка
postgres=# SELECT customers.customername FROM customers
postgres=# WHERE customers.customerId IN (
postgres=# SELECT orders.customerId FROM orders
postgres=# WHERE orders.orderdate IN (
postgres=# SELECT orders.orderdate FROM orders
postgres=# WHERE orders.orderdate = '2024-07-01'));
 customername
-----
Рога и копыта
(1 row)

postgres=#
```

6. Пример запроса, в котором вместо таблицы, указываемой после ключевого слова FROM, используется подзапрос.

Запрос возвращает даты совершения заказов клиентом с идентификатором равным 3.

```
Терминал
Файл  Правка  Вид  Поиск  Терминал  Справка
postgres=# SELECT * FROM (SELECT orders.orderdate
postgres=# FROM customers, orders
postgres=# WHERE orders.customerid = 3 AND orders.customerid = customers.customerid);
 orderdate
-----
2024-05-10
2024-09-10
2024-06-08
2024-07-02
(4 rows)

postgres=#
```

7. Пример связанного подзапроса.

Запрос возвращающий идентификатор клиента и дату последнего сделанного заказа в указанном месяце (в августе).

```
Терминал
Файл  Правка  Вид  Поиск  Терминал  Справка
postgres=# SELECT o.* FROM orders o
postgres=# WHERE o.orderdate = (
postgres=# SELECT MAX(o2.orderdate) FROM orders o2
postgres=# WHERE EXTRACT(MONTH FROM o2.orderdate) = 8);
 orderid | customerid | orderdate
-----+-----+-----
      9 |          2 | 2024-08-10
(1 row)

postgres=#
```

Ответы на контрольные вопросы.

1. Даны таблицы:

```
CREATE TABLE Рейс (  
    Номер_рейса INT,  
    Конечный_пункт VARCHAR(30),  
    Дата_вылета DATETIME,  
    Продолжительность_маршрута INT,  
    Число_билетов INT,  
    Стоимость MONEY);
```

```
CREATE TABLE Билет (  
    Номер_места CHAR(3),  
    Номер_рейса CHAR(6),  
    Дата_продажи DATETIME,  
    Продолжительность_маршрута INT,  
    Стоимость MONEY,  
    Фамилия_пассажира VARCHAR(20));
```

Вывести список пассажиров, не летающих в Самару.

Ответ:

```
SELECT DISTINCT Билет.Фамилия_пассажира  
FROM Билет  
WHERE Билет.Фамилия_пассажира NOT IN (  
    SELECT Билет.Фамилия_пассажира  
    FROM Рейс  
    INNER JOIN Билет  
    ON Рейс.Номер_рейса = Билет.Номер_рейса  
    WHERE Рейс.Конечный_пункт="Самара");
```

2. Даны таблицы:

```
CREATE TABLE Автор (  
    Код_Автора INT ,  
    Фамилия VARCHAR(50) NULL,  
    Имя VARCHAR(50) NULL,  
    Отчество VARCHAR(50) NULL,  
    Пол VARCHAR(50) NOT NULL ,  
    Дата_рождения DATETIME ,  
    Телефон CHAR(9));
```

```
CREATE TABLE Книга (  
    Код_Книги INT,  
    Название VARCHAR(50) NOT NULL,  
    Цена MONEY,  
    Тематика VARCHAR(50) NOT NULL  
    Издательство VARCHAR(50) NOT NULL,  
    Код_Автора INT NOT NULL,  
    Количество INT);
```

Определить авторов, не печатающих свои книги в издательстве «АСТ».

Ответ:

```
SELECT Автор.Фамилия, Автор.Код_Автора
FROM Автор
WHERE Автор.Код_Автора NOT IN (
    SELECT Книга.Код_Автора
    FROM Книга WHERE Книга.Издательство='АСТ');
```

3. Даны таблицы:

```
CREATE TABLE Город (
    Код_Города INT ,
    Название VARCHAR(20) NOT NULL,
    Тариф MONEY,
    Регион VARCHAR(20));

CREATE TABLE Разговор (
    Код_Разговора INT,
    Код_Города INT NOT NULL,
    Фамилия VARCHAR(20),
    Дата DATETIME NOT NULL,
    Продолжительность INT NOT NULL);
```

Вывести список городов, телефонные тарифы которых выше среднего.

Ответ:

```
SELECT Город.Название
FROM Город
WHERE Город.Тариф > (SELECT Avg(Город.Тариф) FROM Город);
```

4. Даны таблицы:

```
CREATE TABLE Блюдо (
    Название_блюда VARCHAR(20) NOT NULL,
    Время_приготовления INT NOT NULL,
    Общая_калорийность INT NOT NULL,
    Номер_рецепта INT,
    Повар VARCHAR(20),
    Стоимость MONEY);

CREATE TABLE Компонент (
    Название_компонента VARCHAR(20),
    Вес FLOAT,
    Белки INT,
    Блюдо VARCHAR(20),
    Углеводы INT,
    Стоимость MONEY NOT NULL);
```

Определить блюдо, которое можно приготовить быстрее всех остальных блюд.

Ответ:

```
SELECT Блюдо.Название_блюда, Блюдо.Время_приготовления
FROM Блюдо
WHERE Блюдо.Время_приготовления = (
    SELECT MIN(Блюдо.Время_Приготовления)
    FROM Блюдо);
```

5. Даны таблицы:

```
CREATE TABLE Рейс (  
    Номер_рейса INT,  
    Конечный_пункт VARCHAR(30),  
    Дата_вылета DATETIME,  
    Продолжительность_маршрута INT,  
    Число_билетов INT,  
    Стоимость MONEY);
```

```
CREATE TABLE Билет (  
    Номер_места CHAR(3),  
    Номер_рейса INT,  
    Дата_продажи DATETIME,  
    Стоимость MONEY,  
    Фамилия_пассажира VARCHAR(20));
```

Определить список пассажиров, покупающих билеты на самые дальние рейсы.

Ответ:

```
SELECT Билет.Фамилия_пассажира, Рейс.Продолжительность_маршрута  
FROM Рейс  
INNER JOIN Билет ON Рейс.Номер_рейса = Билет.Номер_рейса  
WHERE Рейс.Продолжительность_маршрута = (  
    SELECT Max(Рейс.Продолжительность_маршрута)  
FROM Рейс);
```