МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Воронежский государственный технический университет»

Кафедра радиоэлектронных устройств и систем

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторной работы №13
для студентов специальности 11.05.01
«Радиоэлектронные системы и комплексы»
очной формы обучения

Воронеж 2024

УДК 681.3.06(07)
ББК 32.97я7

Составитель А. И. Сукачев

Информационные технологии: методические указания к выполнению лабораторной работы №13 для студентов специальности 11.05.01 «Радиоэлектронные системы и комплексы» очной формы обучения / ФГБОУ ВО «Воронежский государственный технический университет»; сост.: А. И. Сукачев. – Воронеж: Изд-во ВГТУ, 2024. – 25 с.

В соответствии с рабочими учебными программами дисциплин приведены описания методов измерений и методик выполнения лабораторных работ, изложены теоретические сведения, лежащие в основе программирования на языке С++. По каждой лабораторной работе в описание включены: цель, основные теоретические сведения, порядок подготовки и проведения работы, перечень положений, которые необходимо отразить в выводах.

Предназначены для студентов специальности 11.05.01 «Радиоэлектронные системы и комплексы» очной формы обучения.

Методические указания подготовлены в электронном виде и содержатся в файле МУ_ИТ_ЛР_13.pdf.

Ил. 27. Табл. 1. Библиогр.: 3 назв.

УДК 681.3.06(07)
ББК 32.97я7

Рецензент – А. В. Останков, д-р техн. наук, профессор кафедры радиотехники ВГТУ

Издается по решению редакционно-издательского совета Воронежского государственного технического университета

ЛАБОРАТОРНАЯ РАБОТА № 13
ПОЛУЧЕНИЕ БАЗОВЫХ ЗНАНИЙ ПО РАБОТЕ С БАЗАМИ ДАННЫХ

1. ОБЩИЕ УКАЗАНИЯ
1.1. ЦЕЛЬ РАБОТЫ

Познакомиться с основами разработки баз данных. Получить базовые навыки работы с СУБД PostgreSQL.

1.2. ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Основным содержанием работы является разработка базы данных, её реализация и выполнение простейших запросов. В ходе работы идёт ознакомление с приведённым примером базы данных, производится работа с базой данных с помощью СУБД PostgreSQL. На основе полученных знаний выполняется индивидуальное задание по проектированию базы данных.

2. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ
2.1. ER-ДИАГРАММЫ

В процессе проектирования базы данных появляется необходимость построения ER диаграммы.

Диаграмма отношений сущностей (ER) - это тип блок-схемы, которая иллюстрирует, как «сущности», такие как люди, объекты или концепции, связаны друг с другом в системе. Диаграммы ER чаще всего используются для проектирования или отладки реляционных баз данных в области разработки программного обеспечения, информационных систем для бизнеса, образования и исследований. Также известные как ERD или ER-модели, они используют определенный набор символов, таких как прямоугольники, ромбы, овалы и соединительные линии, чтобы изобразить взаимосвязь сущностей, отношений и их атрибутов. Они отражают грамматическую структуру, с сущностями в виде существительных и отношений в виде глаголов.

Диаграммы ER связаны с диаграммами структуры данных (DSD), которые фокусируются на отношениях элементов внутри объектов, а не на отношениях между самими объектами. Диаграммы ER также часто используются в сочетании с диаграммами потоков данных (DFD), которые отображают поток информации для процессов или систем.

Питеру Чену (также известному как Питер Пин-Шен Чен), в настоящее время преподаватель в Университете Карнеги-Меллона в Питтсбурге, приписывают разработку ER-моделирования для проектирования баз данных в 1970-х годах. Работая в качестве ассистента профессора в Школе управления Слоана при Массачусетском технологическом институте, в 1976 году он опубликовал основную статью под названием «Модель сущности-отношения: на пути к унифицированному представлению данных».

В более широком смысле изображение взаимосвязанности вещей восходит к наименее древней Греции с работами Аристотеля, Сократа и Платона. Это было замечено совсем недавно в работах философов-логиков XIX и XX веков, таких как Чарльз Сандерс Пирс и Готтлоб Фреге.

ER-диаграммы используются в следующих областях:

- Проектирование базы данных: диаграммы ER используются для моделирования и проектирования реляционных баз данных с точки зрения логики и бизнес-правил (в логической модели данных) и с точки зрения конкретной технологии, которая должна быть реализована (в физической модели данных). В разработке программного обеспечения, Диаграмма ER часто является начальным шагом в определении требований для проекта информационных систем. Позже он также используется для моделирования конкретной базы данных или баз данных. Реляционная база данных имеет эквивалентную реляционную таблицу и потенциально может быть выражена таким образом при необходимости.
- Устранение неполадок с базой данных. Диаграммы ER используются для анализа существующих баз данных с целью поиска и решения проблем в логике или развертывании. Рисование диаграммы должно показать, где происходит ошибка.
- Бизнес-информационные системы: диаграммы используются для проектирования или анализа реляционных баз данных, используемых в бизнес-процессах. Любой бизнес-процесс, который использует полевые данные, включающие сущности, действия и взаимодействие, потенциально может извлечь выгоду из реляционной базы данных. Это может упростить процессы, легче находить информацию и улучшать результаты.
- Реинжиниринг бизнес-процессов (BPR): ER-диаграммы помогают анализировать базы данных, используемые при реинжиниринге бизнес-процессов, и моделировать новую настройку базы данных.
- Образование. Базы данных - это современный метод хранения реляционной информации в образовательных целях и последующего поиска, поэтому диаграммы ER могут быть полезны при планировании этих структур данных.
- Исследования: Поскольку так много исследований сосредоточено на структурированных данных, диаграммы ER могут играть ключевую роль в создании полезных баз данных для анализа данных.

Компоненты и особенности диаграммы ER:

- Сущность - определенная вещь, такая как человек, объект, концепция или событие, в которой могут храниться данные. Думайте о сущностях как о существительных. Примеры: клиент, студент, автомобиль или продукт. Обычно отображается в виде прямоугольника. У сущностей есть ключи. Первичный ключ: ключ, выбранный разработчиком базы данных для уникальной идентификации набора сущностей. Внешний ключ: определяет отношения между объектами.

- Отношения - как сущности действуют друг на друга или связаны друг с другом. Думайте об отношениях как о глаголах. Например, названный студент может записаться на курс. Эти две сущности будут студентом и курсом, и изображенные отношения являются актом зачисления, таким образом соединяя две сущности. Отношения обычно отображаются в виде ромбов или меток непосредственно на соединительных линиях.
- Атрибут - свойство или характеристика объекта. Часто изображается в виде овала или круга.
- Связь - определяет числовые атрибуты отношений между двумя сущностями или наборами сущностей. Три основных типа связи: один-к-одному, один-ко-многим и многие-многие. Пример один-к-одному - один студент, связанный с одним почтовым адресом. Пример « один ко многим» (или «многие к одному», в зависимости от направления отношений): один студент регистрируется на нескольких курсах. Пример «многие ко многим»: студенты как группа связаны с несколькими преподавателями, а преподаватели, в свою очередь, связан с несколькими студентами.
- Модели ER и модели данных бывают трех уровней детализации:
- Концептуальная модель данных: представление самого высокого уровня, содержащее наименьшее количество деталей. Его значение показывает общий объем модели и изображает архитектуру системы.
- Логическая модель данных: содержит больше деталей, чем концептуальная модель. Определены более подробные операционные и транзакционные объекты. Логическая модель не зависит от технологии, в которой она будет реализована.
- Физическая модель данных: физические модели показывают достаточно подробно технологию для производства и реализации фактической базы данных.

Процесс создания ER диаграммы:

1. Цель и область применения: определите цель и область применения анализируемого или моделируемого объекта.
2. Сущности: определить сущности, которые участвуют. Когда вы будете готовы, начните рисовать их в прямоугольниках (или по выбору формы вашей системы) и пометить их как существительные.
3. Отношения: определите, как все сущности связаны. Нарисуйте линии между ними, чтобы обозначить отношения и пометить их. Некоторые сущности могут быть не связаны, и это нормально. В разных системах обозначений взаимосвязь может быть помечена ромбом, другим прямоугольником или непосредственно над соединительной линией.

4. Атрибуты: добавить ключевые атрибуты сущностей. Атрибуты часто показывают как овалы.
5. Тип связи: показать тип отношения: 1-1, 1-многим или многие-ко-многим.

2.2. POSTGRESQL

Для проектирования базы данных в лабораторной работе используется СУБД PostgreSQL.

pgAdmin 4 – клиент для работы с PostgreSQL – поддерживает все функции PostgreSQL, от написания простых SQL-запросов до разработки сложных баз данных. Он предназначен для запроса активной базы данных (в режиме реального времени), что позволяет вам быть в курсе изменений и реализаций.

Особенности pgAdmin 4 включают в себя:

- автоопределение и поддержка объектов, обнаруженных во время выполнения
- SQL Query Tool с прямым редактированием данных
- поддержка административных запросов
- редактор SQL с подсветкой синтаксиса
- переработан графический интерфейс
- мощные диалоги управления и инструменты для решения общих задач
- отзывчивое, контекстно-зависимое поведение
- вспомогательные сообщения об ошибках
- полезные советы
- интерактивная справка и информация об использовании диалогов и инструментов pgAdmin.

Когда открывается pgAdmin, интерфейс имеет строку меню и окно, разделенное на две панели: элемент управления дерева *браузера* на левой панели и браузер с вкладками на правой панели.

2.3. ЗАПРОСЫ SQL

Вы можете создать новую таблицу, указав имя таблицы, а также все имена столбцов и их типы. Создадим две таблицы:

```
CREATE TABLE weather (
    city            varchar(80),
    temp_lo         int,           -- минимальная температура
    temp_hi         int,           -- максимальная температура
    prcp            real,          -- уровень осадков
    date            date
);
CREATE TABLE cities (
    name            varchar(80),
    location        point
```

);

Если вам больше не нужна какая-то таблица, вы можете удалить её, используя следующую команду: DROP TABLE имя_таблицы.

Для заполнения таблицы строками используется оператор INSERT:

```
INSERT INTO weather VALUES ('San Francisco', 46, 50, 0.25, '1994-11-27');
```

Синтаксис выше требует, чтобы вы запомнили порядок столбцов. Альтернативный синтаксис позволяет явно перечислять столбцы:

```
INSERT INTO weather (city, temp_lo, temp_hi, prcp, date)
   VALUES ('San Francisco', 43, 57, 0.0, '1994-11-29');
```

Вы можете перечислить столбцы в другом порядке, если хотите, или даже опустить некоторые столбцы, например, если осадки неизвестны:

```
INSERT INTO weather (date, city, temp_hi, temp_lo)
   VALUES ('1994-11-29', 'Hayward', 54, 37);
```

Многие разработчики считают явное перечисление столбцов лучшим вариантом, чем неявное использование порядка.

Для извлечения данных из таблицы выполняется запрос к ней. Для этого используется оператор SELECT. Оператор делится на список выбора (часть, в которой перечислены возвращаемые столбцы), список таблиц (часть, в которой перечислены таблицы, из которых извлекаются данные) и необязательное условие (часть, которая задает любые ограничения). Например, чтобы получить все строки таблицы weather, введите:

```
SELECT * FROM weather;
```

Здесь * - это сокращение для “всех столбцов". Таким образом, тот же самый результат был бы получен с:

```
SELECT city, temp_lo, temp_hi, prcp, date FROM weather;
```

Запрос можно дополнить, добавив предложение WHERE, которое указывает, какие строки требуются. Предложение WHERE содержит логическое выражение (значение истинности), и возвращает только строки, для которых логическое выражение является истинным. В квалификации допускаются обычные булевы операторы (AND, OR, и NOT). Например, ниже приводится информация о погоде в Сан-Франциско в дождливые дни:

SELECT * FROM weather WHERE city = 'San Francisco' AND prcp > 0.0;

Вы можете сделать так, чтобы результаты запроса возвращались в отсортированном порядке:

SELECT * FROM weather ORDER BY city;

Можно удалить повторяющиеся строки из результата запроса:

SELECT DISTINCT city FROM weather ORDER BY city;

Существующие строки можно обновить с помощью команды UPDATE. Предположим, вы обнаружите, что после 28 ноября все показания температуры отклоняются на 2 градуса. Вы можете исправить эти данные следующим образом:

```
UPDATE weather
    SET temp_hi = temp_hi - 2,  temp_lo = temp_lo - 2
    WHERE date > '1994-11-28';
```

Строки могут быть удалены из таблицы с помощью команды DELETE. Предположим, вы больше не интересуетесь погодой в Хейворде. Чтобы удалить эти строки из таблицы:

DELETE FROM weather WHERE city = 'Hayward';

Без уточнения DELETE удалит все строки из данной таблицы, оставив ее пустой. Следует остерегаться высказываний такого рода. Система не будет запрашивать подтверждение перед этим!

DELETE FROM tablename;

Рассмотрим следующую проблему: вы хотите убедиться, что никто не может вставить строки в таблицу погоды, которые не имеют соответствующей записи в таблице городов. Это называется поддержанием ссылочной целостности ваших данных. В упрощенных системах баз данных это было бы реализовано (если вообще возможно) путем первого просмотра таблицы городов, чтобы проверить, существует ли соответствующая запись, а затем вставить или отклонить новые записи погоды. Этот подход имеет ряд проблем и очень неудобен, поэтому PostgresPro может сделать это за вас.

Новое объявление таблиц будет выглядеть следующим образом:

CREATE TABLE cities (

```
    city     varchar(80) primary key,
    location point
);

CREATE TABLE weather (
    city     varchar(80) references cities(city),
    temp_lo  int,
    temp_hi  int,
    prcp     real,
    date     date
);
```

Предложение FROM выводит таблицу из одной или нескольких других таблиц, указанных в списке ссылок на таблицы, разделенные запятыми. Ссылка на таблицу может быть именем таблицы (возможно, с определением схемы) или производной таблицей, такой как подзапрос, конструкция соединения или их сложные комбинации.

FROM table_reference [, table_reference [, ...]]

Объединенная таблица-это таблица, производная от двух других (вещественных или производных) таблиц в соответствии с правилами конкретного типа соединения. Доступны внутренние, внешние и поперечные соединения. Общий синтаксис объединенной таблицы таков:

T1 join_type T2 [join_condition]

Типы соединения:

1. Перекрестное соединение

T1 CROSS JOIN T2

Для каждой возможной комбинации строк из T1 и T2 (т. е. Декартового произведения) объединенная таблица будет содержать строку, состоящую из всех столбцов в T1, за которыми следуют все столбцы в T2. Если таблицы имеют N и M строк соответственно, то объединенная таблица будет иметь N * M строк.

Условие соединения указывается в предложении ON или USING или неявно словом NATURAL. Условие соединения определяет, какие строки из двух исходных таблиц считаются "совпадающими", как подробно описано ниже.

2. Возможные типы квалифицированного соединения:

a) INNER JOIN

Для каждой строки R1 из T1 объединенная таблица имеет строку для каждой строки в T2, которая удовлетворяет условию соединения с R1.

б) LEFT OUTER JOIN

Сначала выполняется внутреннее соединение. Затем для каждой строки в T1, которая не удовлетворяет условию соединения с любой строкой в T2, присоединенная строка добавляется с нулевыми значениями в Столбцах T2. Таким образом, объединенная таблица всегда имеет по крайней мере одну строку для каждой строки в T1.

в) RIGHT OUTER JOIN

Сначала выполняется внутреннее соединение. Затем для каждой строки в T2, которая не удовлетворяет условию соединения с любой строкой в T1, присоединенная строка добавляется с нулевыми значениями в Столбцах T1. Это противоположно левому соединению: результирующая таблица всегда будет иметь строку для каждой строки в T2.

г) FULL OUTER JOIN

Сначала выполняется внутреннее соединение. Затем для каждой строки в T1, которая не удовлетворяет условию соединения с любой строкой в T2, присоединенная строка добавляется с нулевыми значениями в Столбцах T2. Кроме того, для каждой строки T2, которая не удовлетворяет условию соединения с любой строкой в T1, добавляется объединенная строка с нулевыми значениями в столбцах T1.

3. ЛАБОРАТОРНЫЕ ЗАДАНИЯ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ИХ ВЫПОЛНЕНИЮ

Задание 1. Спроектировать базу данных, построить ER-диаграмму (Entity-Relationship, диаграммы «сущность-связь»).

Здесь необходимо определиться с функциями, которые будет выполнять разрабатываемое программное обеспечение. Пусть в нашей программе, основной задачей которой будет хранение заметок, будет возможна авторизация и регистрация нового пользователя, работа с заметками, поиск заметок по категориям. Также возможен обмен личными и групповыми сообщениями. Последовательность действий при проектировании базы данных выглядит следующим образом:

1) Выделить задачи, которая должна выполнять разрабатываемая система:

а) хранение данных о пользователях;

б) хранение данных о группах пользователей;

в) хранение данных о заметках;

г) хранение данных о категориях заметок.

2) Выделить сущности рассматриваемой предметной области:

а) «пользователь»;

б) «группа пользователей»;

в) «заметка»;

г) «категория заметок»;

3) Определить необходимое число связей между сущностями:

а) связь «пользователь – категория заметок»: один пользователь может добавить много категорий заметок, но одна и та же категория не может быть добавлена многими пользователями (рис. 1):

Рис. 1. ER-диаграмма связи «пользователь – категория»

б) связь «категория – заметка»: одна категория может включать в себя много заметок, и одна заметка может входить в состав множества категорий (Рис. 2):

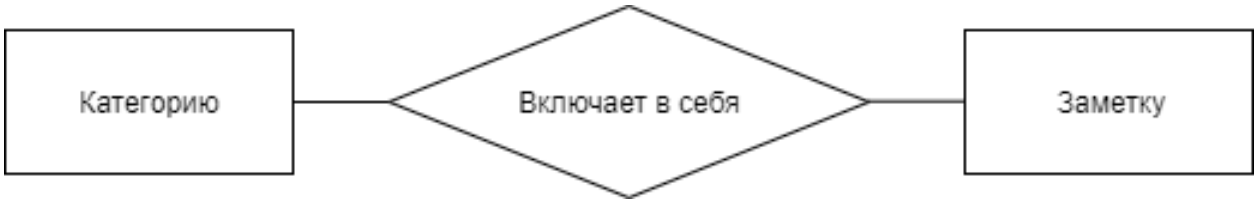

Рис. 2. ER-диаграмма связи «Категория – Заметка»

в) связь «пользователь – группа пользователей»: один пользователь может входить в несколько групп пользователей, и одна группа пользователей может включать в себя множество пользователей (рис. 3):

Рис. 3. ER-диаграмма связи «пользователь – группа»

3) Определить атрибуты сущностей (рис. 4):

а) сущность «пользователь»:

- «ID пользователя» - ключ сущности;
- «логин»;
- «пароль»;

Рис. 4. Атрибуты сущности «пользователь» на ER-диаграмме

б) сущность «категория» (рис. 5):
- «ID категории» - ключ сущности;
- «название категории»;
- «ID заметок».

Рис. 5. Атрибуты сущности «категория» на ER-диаграмме

в) сущность «заметка» (рис. 6):
- «ID заметки» - ключ сущности;
- «название заметки»;

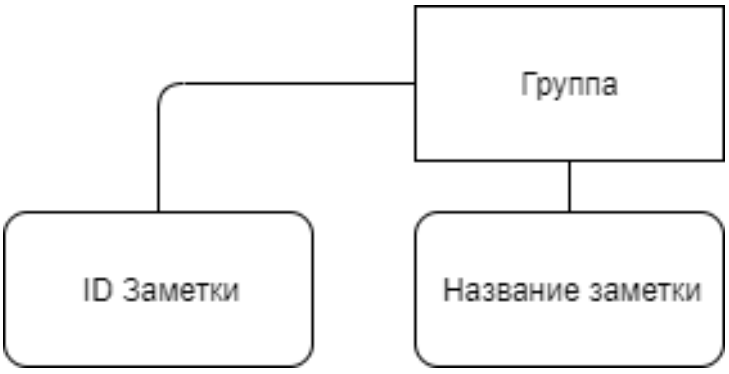

Рис. 6. Атрибуты сущности «ингредиент» на ER-диаграмме

г) сущность «группа пользователей» (рис. 7):
- «ID группы» - ключ сущности;
- «название группы».

- «ID пользователей»

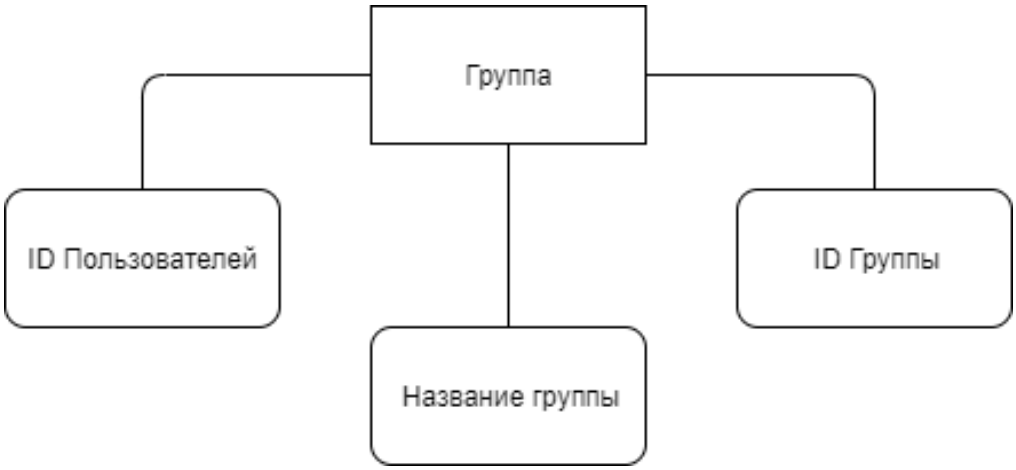

Рис. 7. Атрибуты сущности «оборудование» на ER-диаграмме

4) Составить окончательный вариант ER-диаграммы (рис. 8).

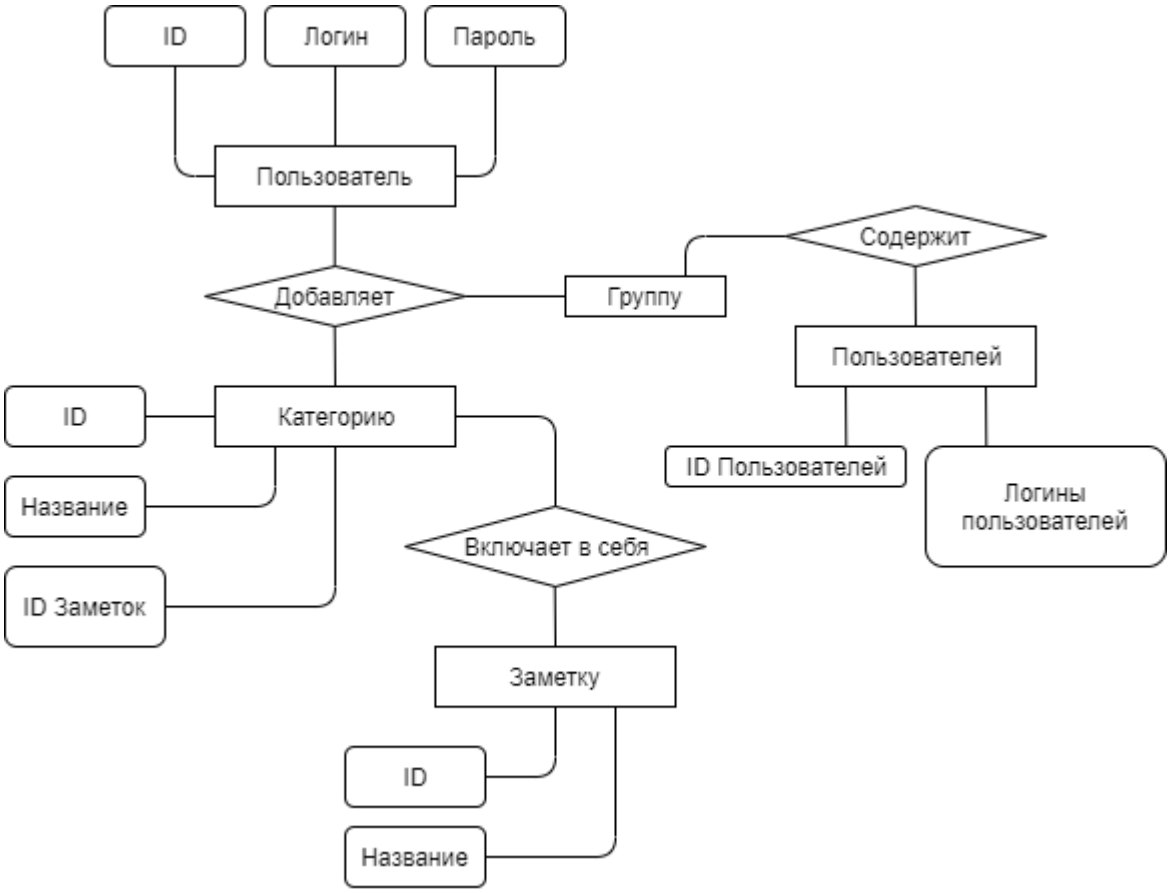

Рис. 8. ER-диаграмма разрабатываемой системы

5) Составить табличную схему БД системы электронных записей (рис. 9).

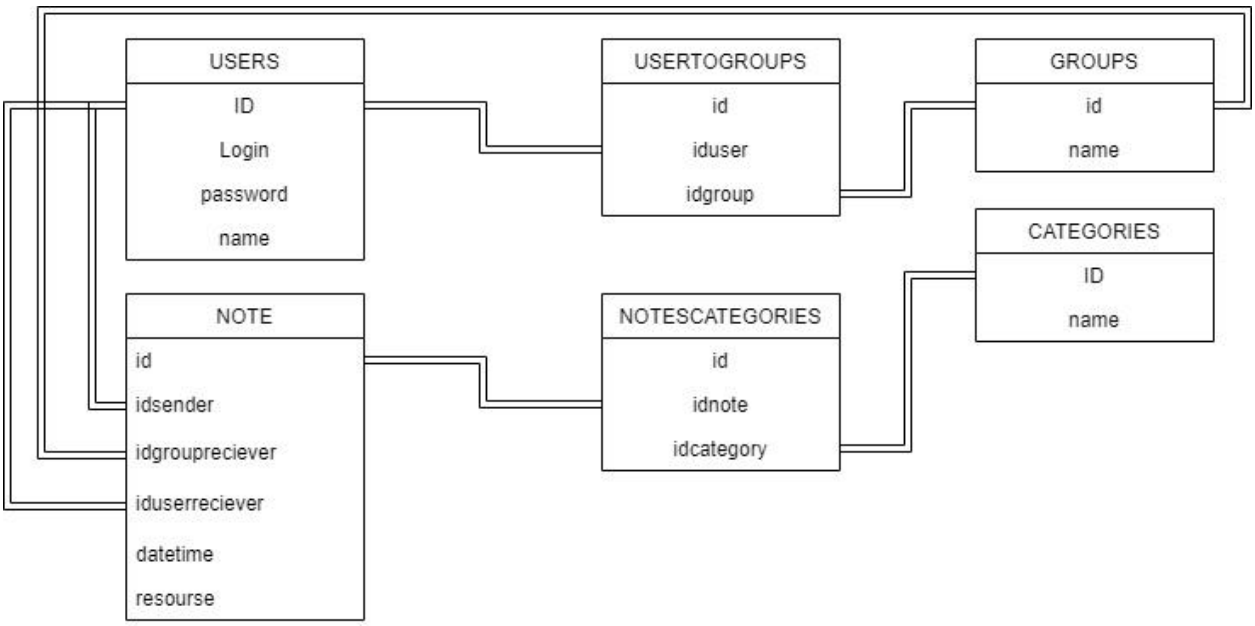

Рис. 9. Табличная схема базы данных «Notepad»

Задание 2. Создать базу данных с помощью СУБД PostgreSQL

1. Для начала работы с PostgreSQL нужно запустить pgAdmin4.exe. Это инструмент для упрощения администрирования на сервере PostgreSQL, включенный в ее базовый комплект. По сути он представляет собой графический клиент для работы с сервером, через который в удобном виде можно создавать, удалять, изменять базы данных и управлять ими.

После запуска pgAdmin4 на компьютере создается сервер для работы с базами данных.

Интерфейс pgAdmin4 состоит из меню (рис. 10) и обозревателя (рис. 11).

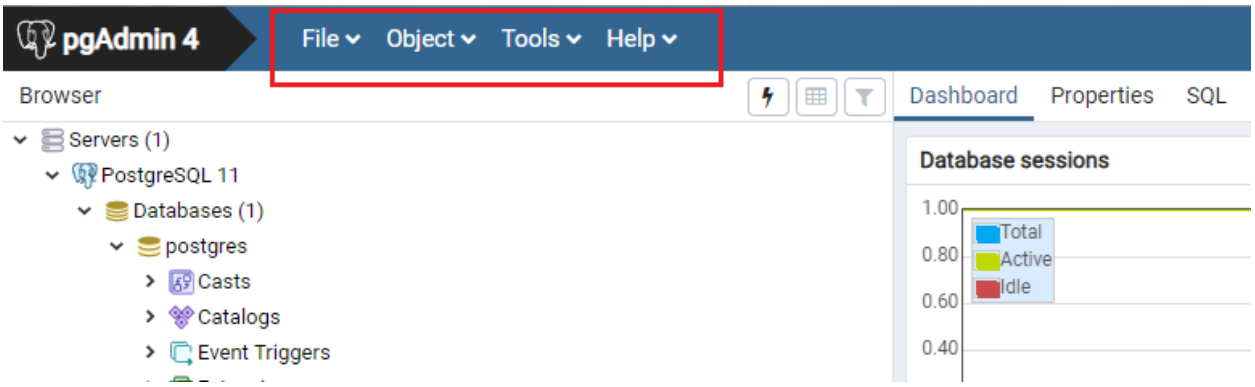

Рис. 10. Меню pgAdmin4

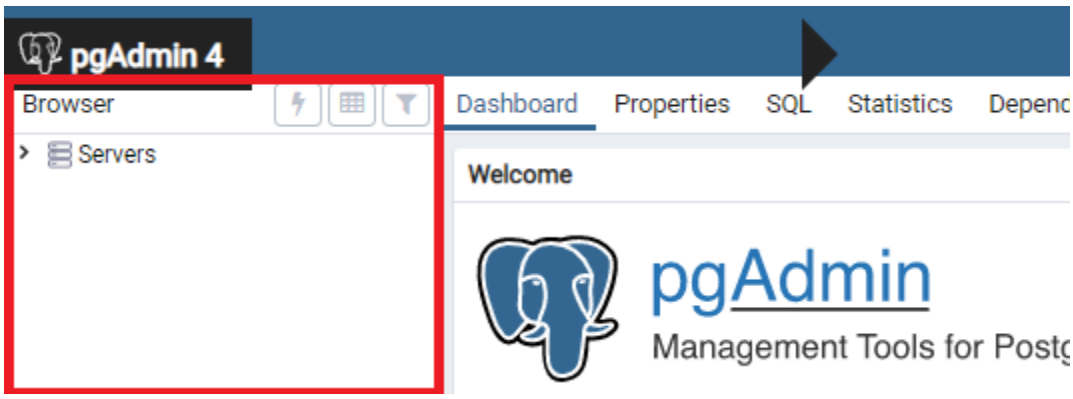

Рис. 11. Обозреватель pgAdmin4

Сначала в обозревателе отображена только вкладка Servers. При щелчке по ней появится дочерняя ветвь с названием PostgreSQL 11. После нее идут ветви Databases, Login/Group Roles и Tablespaces.

Login/Group Roles содержит т.н. роли. Это пользователи с разными правами доступа. По умолчанию их 9, среди них postgres. Обозреватель с раскрытыми упомянутыми ветвями приведен на рис.12.

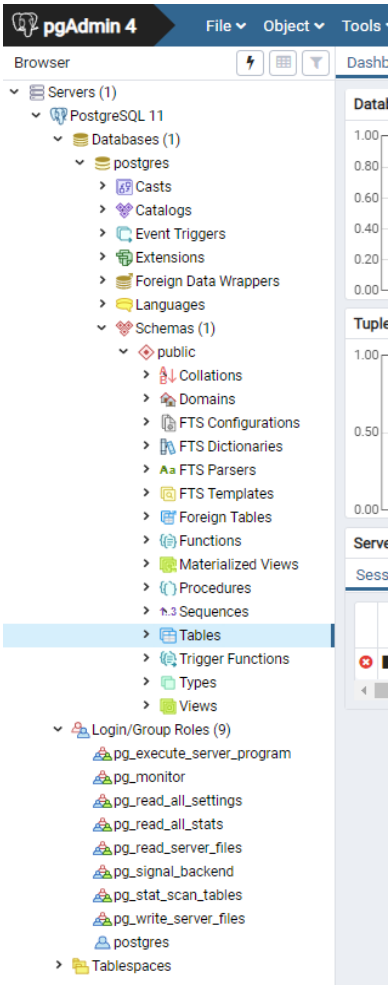

Рис. 12. Обозреватель pgAdmin4 с раскрытыми Databases и Login/Group Roles

При открытии Databases появится список имеющихся баз данных. По умолчанию такая база одна, и это postgres. Это стандартная БД, создаваемая системой при инициализации.

2. Создать новую базу данных. Для этого требуется нажать правой кнопкой мыши по Databases, затем выбрать Create—Database (Рис.13). В поле Owner (Владелец) задается пользователь, имеющий полные права доступа к создаваемой таблице. По умолчанию это postgres, в данном случае менять ничего не нужно.

3. Задать имя БД — Notepad. Для завершения создания базы данных нажать кнопку Save.

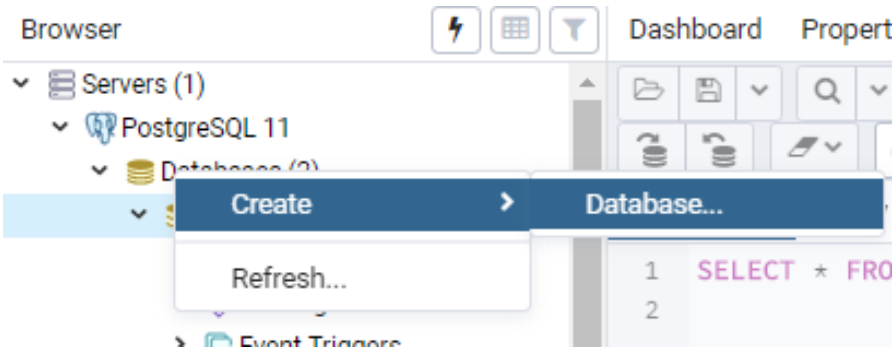

Рис. 13. Создание БД

4. Главным достоинством pgAdmin является то, что он позволяет управлять СУБД без непосредственного ввода SQL-команд. Однако для создания таблиц мы будем использовать SQL-команды. Для этого нужно нажать правой кнопкой мыши на Databases и выбрать Query Tool (рис. 14).

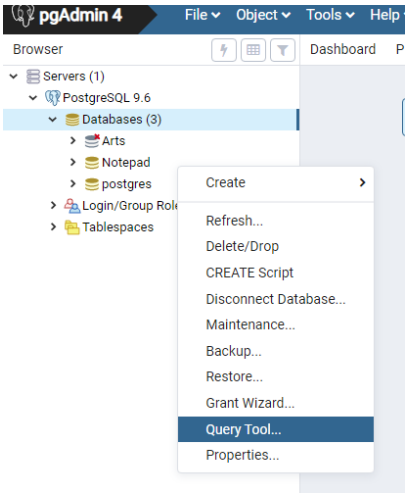

Рис. 14. Query Tool

5. Прописать набор SQL команд создания таблиц базы данных (рис. 15).

```
CREATE TABLE Users
(
ID Serial PRIMARY KEY,
Login VARCHAR(30) NOT NULL,
Password VARCHAR(30) NOT NULL
Fio VARCHAR(50) NOT NULL
);
CREATE TABLE Categories
(
ID Serial PRIMARY KEY,
Name VARCHAR(30) NOT NULL
);
CREATE TABLE NotestoCategories
(
ID Serial PRIMARY KEY,
IdNote INTEGER NOT NULL,
IdCategory INTEGER NOT NULL
);
CREATE TABLE Groups
(
ID Serial PRIMARY KEY,
Name VARCHAR(30) NOT NULL
);
CREATE TABLE UserstoGroups
(
ID Serial PRIMARY KEY,
IdUser INTEGER NOT NULL,
IdGroup INTEGER NOT NULL
);
CREATE TABLE Note
(
ID Serial PRIMARY KEY,
IdSender INTEGER NOT NULL,
IdUserReciever INTEGER,
IdGroupReciever INTEGER,
DateTime TIMESTAMP,
Resource TEXT
);
```

Рис. 15. набор SQL команд создания таблиц базы данных

Здесь задать название каждой таблицы, имена столбцов и типы данных в них. Когда команды написаны, нажать клавишу F5 или кнопку с изображением молнии на панели инструментов. После этого ваши таблицы добавятся в базу данных и будут отображаться во вкладке Tables схемы public ветви Schemas (рис. 16).

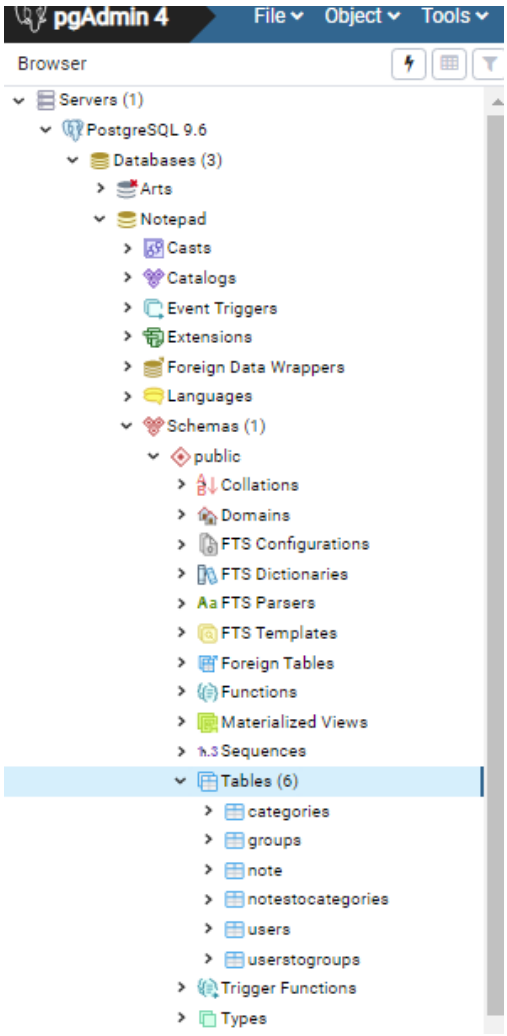

Рис. 16. Расположение таблиц

6. Теперь необходимо связать таблицы. Для этого нужно выбрать таблицу и нажать правой кнопкой мыши на Costraints, выбрать Create – Foreign key (рис. 17).

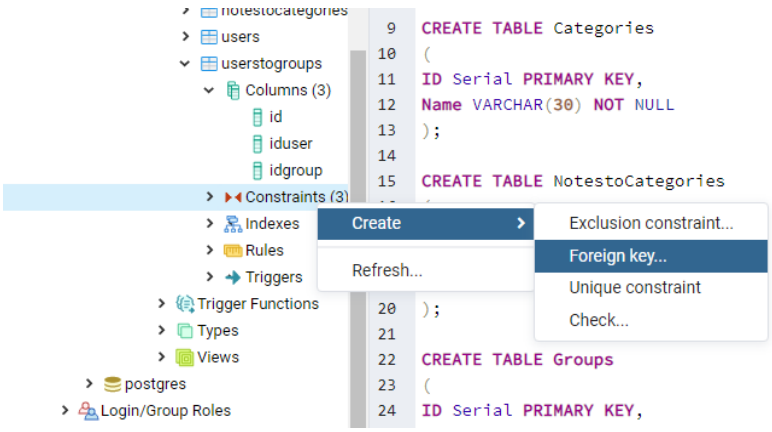

Рис. 17. Связывание таблиц

7. В открывшемся диалоговом окне во вкладке General задать имя связи. Во вкладке Columns первой задать колонку выбранной таблицы, которую мы будем связывать, далее – таблицу, с которой будет осуществляться связь, затем – колонку добавленной таблицы, и нажать + (рис. 18). Во вкладке Action в обоих полях указать CASCADE. Это необходимо для того, чтобы при удалении записи из базы данных удалялись все связанные с ней записи. Для сохранения нажать Save.

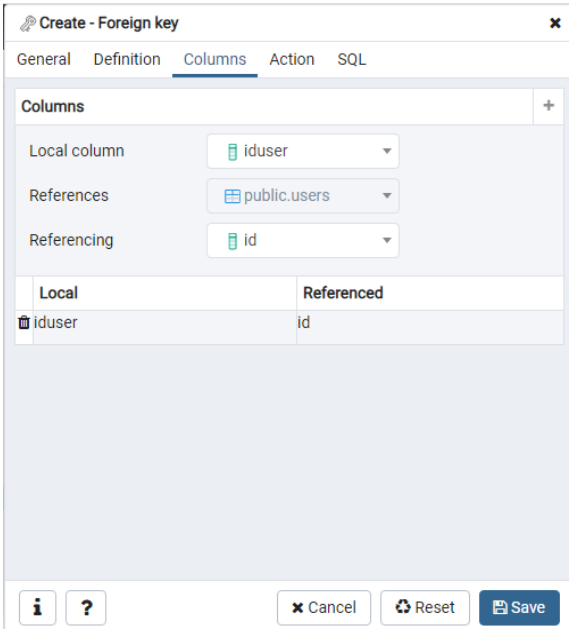

Рис. 18. Задание колонки таблицы

8. Создать все связи, показанные на рис. 9. На этом создание базы данных завершено.

Задание 3. Познакомиться с базовыми конструкциями запросов языка SQL

9. При помощи команды INSERT заполнить таблицу «users» (рис. 19). Результат можно увидеть нажав на нужную таблицу правой кнопкой мыши и выбрав View/Edit Data – All Rows (рис. 20). Аналогичным образом заполнить данными другие таблицы.

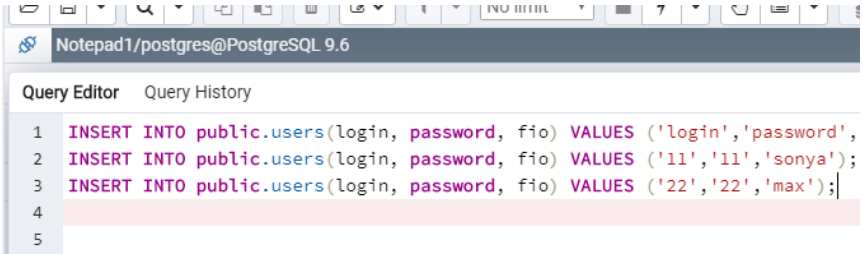

Рис. 19. Заполнение таблицы «users»

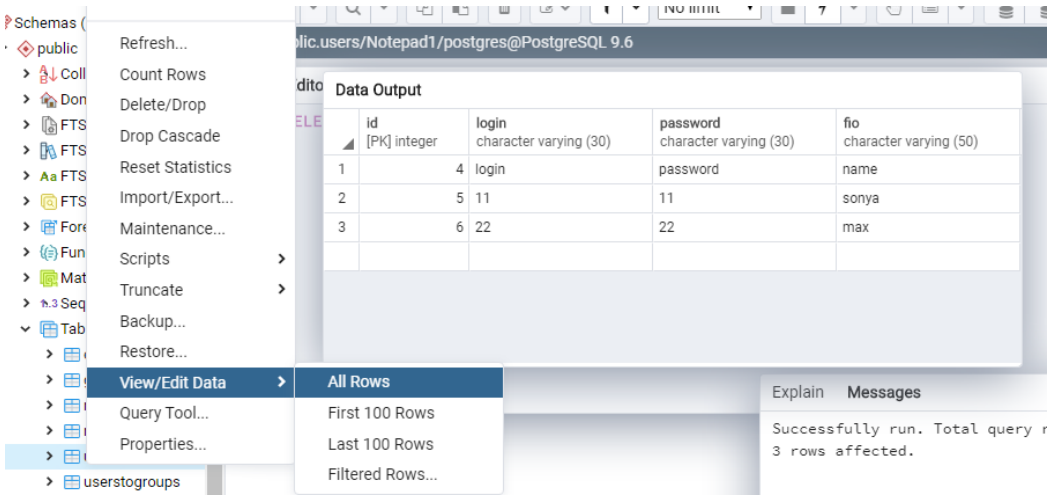

Рис. 20. Просмотр результата

10. Осуществить удаление записей с помощью команды DELETE (рис. 21). При выполнении приведенной на рисунке команды из таблицы удалится запись с id = 4 (рис. 22).

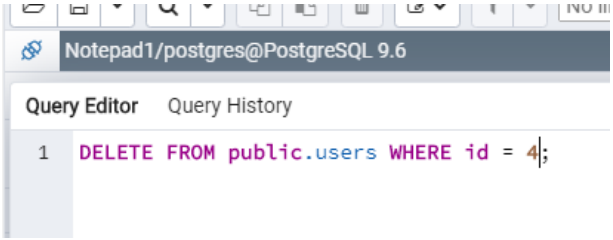

Рис. 21. Удаление записей

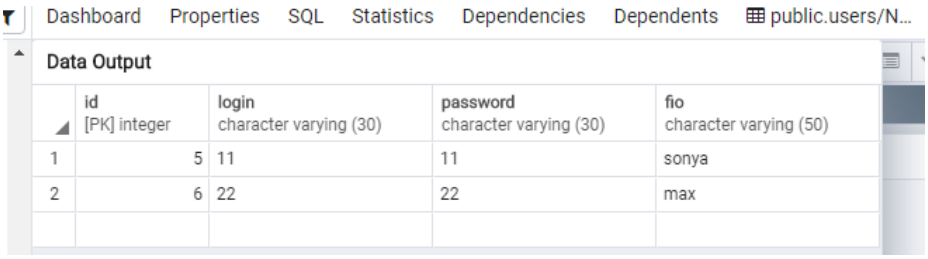

Рис. 22. Проверка удаления записи

11. Применить команду UPDATE для редактирования данных (Рис. 23). Результат выполнения команды, представленной на рис. 24, можно увидеть на рис. 24.

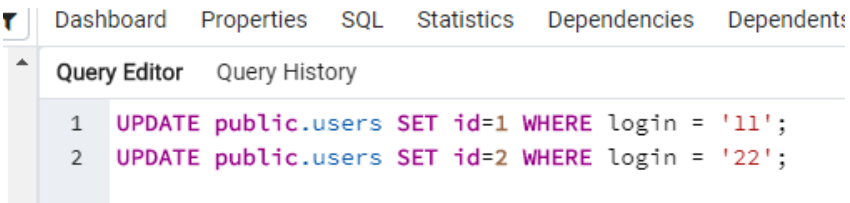

Рис. 23. Редактирование данных

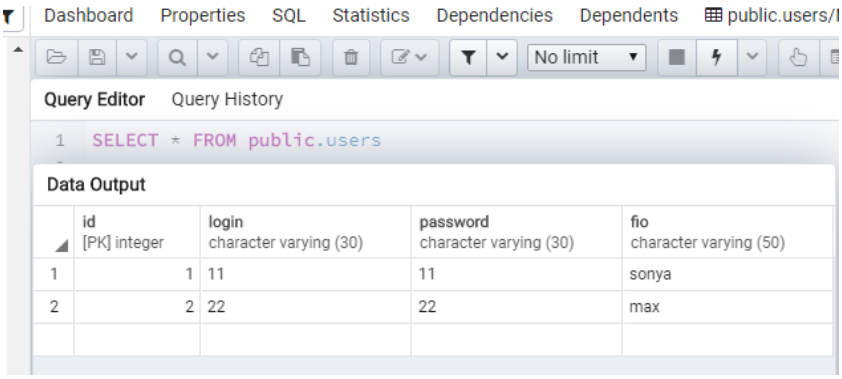

Рис. 24. Проверка измененной записи

11. Применить команду SELECT для выполнения различных запросов. На рис. 25 представлен запрос, позволяющий вывести все записи таблицы users. Есть возможность выводить только выбранные столбцы. Такие запросы могут быть с условием (рис. 25) и выводиться в определенном порядке (рис. 26).

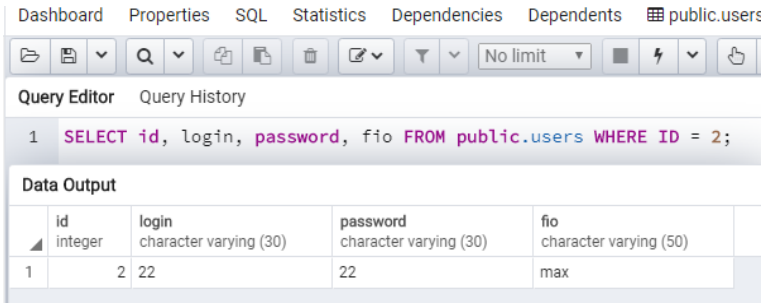

Рис. 25. Запрос вывода данных с условием

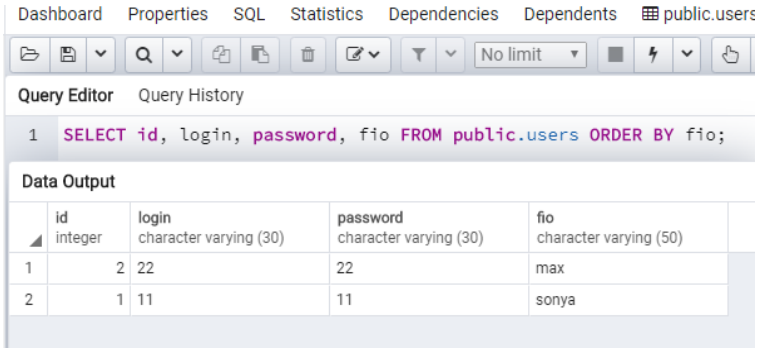

Рис. 26. Запрос на вывод данных в определенном порядке

12. Сделать запрос из связанных таблиц. Для этого используется команда INNER JOIN. На рис. 27 представлен запрос из двух связанных таблиц, позволяющий отобрать логин и имя пользователей, состоящих в группе с названием «it», и его результат.

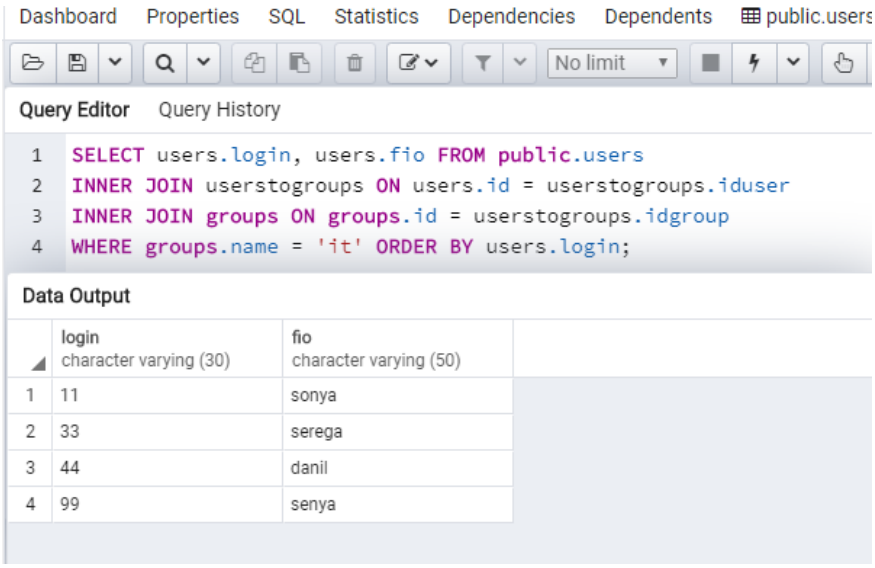

Рис. 27. Запрос из двух связанных таблиц

4. ВАРИАНТЫ ЛАБОРАТОРНЫХ ЗАДАНИЙ

Варианты заданий даны в табл. 1.

Таблица 1

Варианты заданий

Вариант	Задание
1	Разработать и программно реализовать базу данных для магазина спортивных товаров
2	Разработать и программно реализовать базу данных для социальной сети художников
3	Разработать и программно реализовать базу данных для пекарни
4	Разработать и программно реализовать базу данных для службы доставки еды
5	Разработать и программно реализовать базу данных для электронного журнала
6	Разработать и программно реализовать базу данных для туристического агентства
7	Разработать и программно реализовать базу данных для продуктового магазина
8	Разработать и программно реализовать базу данных для парикмахерской
9	Разработать и программно реализовать базу данных для магазина автозапчастей
10	Разработать и программно реализовать базу данных для социальной сети репетиторов и учеников

5. ДОМАШНЕЕ ЗАДАНИЕ

Потренироваться делать различные запросы

6. УКАЗАНИЯ ПО ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать название и цель работы, краткие теоретические сведения, а также код с комментариями. В конце отчета должен быть вывод.

7. КОНТРОЛЬНЫЕ ВОПРОСЫ К ЛАБОРАТОРНЫМ ЗАДАНИЯМ

1. Что такое ER-диаграмма?
2. Назовите основные элементы ER-диаграммы?
3. Каковы особенности СУБД PostgreSQL?
4. Как создать таблицу в PostgreSQL?
5. Как добавлять, создавать, редактировать данные в PostgreSQL?
6. Как делать запросы в PostgreSQL?
7. Как делать запросы из связанных таблиц в PostgreSQL?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Саммерфилд М. Qt. Профессиональное программирование. Разработка кроссплатформенных приложений на C++ / М. Саммерфилд – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 560 с.
2. Шлее М. Qt 5.10. Профессиональное программирование на C++ / М. Шлее – СПб: БХВ-Петербург, 2018. – 1072 с.
3. Бланшет Ж. QT 4: программирование GUI на C++ / Ж. Бланшет – М.: КУДИЦ-ПРЕСС, 2007. – 546 с.

ОГЛАВЛЕНИЕ

1. Лабораторная работа № 13. Получение базовых знаний по работе с базами данных……..3
1. Общие указания 3
1.1. Цель работы 3
1.2. Общая характеристика работы 3
2. Основные теоретические сведения 3
2.1. ER-диаграммы 3
2.2. PostgreSQL 6
2.3. Запросы SQL 6
3. Лабораторные задания и методические указания по их выполнению .. 10
4. Варианты лабораторных заданий 22
5. Домашнее задание 22
6. Указания по оформлению отчета 23
7. Контрольные вопросы к лабораторным заданиям 23
Библиографический список 23

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторной работы №13
для студентов специальности 11.05.01
«Радиоэлектронные системы и комплексы»
очной формы обучения

Составитель
Сукачев Александр Игоревич

Издается в авторской редакции

Подписано к изданию 18.03.2024.
Уч.-изд. л. 1,3.

ФГБОУ ВО «Воронежский государственный технический университет»
394006 Воронеж, ул. 20-летия Октября, 84