Комитет по образованию Правительства Санкт-Петербурга
САНКТ-ПЕТЕРБУРГСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КУРСОВОЙ ПРОЕКТ

по дисциплине «Управление и автоматизация баз данных»

Специальность 09.02.07

Информационные системы и программирование (администратор баз данных)

Выполнил студен группы 225

Седнев Матвей Дмитриевич

Сдал _____

Проверил _____

Оценка _____

Санкт-Петербург 2024 г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Введение является важной составной частью данного проекта, посвященного разработке базы данных и созданию соответствующего приложения. В данном разделе обсуждаются ключевые аспекты проекта:

Актуальность темы: в свете постоянного развития информационных технологий и их все более широкого использования создание эффективных баз данных становится насущной задачей. В контексте учета и мониторинга данных о потреблении ресурсов, таких как энергия, важность оптимизации и управления данными становится особенно яркой.

Сущность работы: Основной целью данного проекта является разработка базы данных, предназначенной для хранения информации о счетчиках и их показаниях. Это предполагает создание структуры данных, которая будет обеспечивать эффективное и удобное хранение информации, необходимой для учета и анализа энергопотребления.

Замысел работы: Проект предполагает создание надежной и функциональной базы данных, а также разработку приложения для взаимодействия с этой базой данных. Основной задачей будет определение требований к базе данных, разработка ее структуры, а также создание пользовательского интерфейса для удобного доступа к данным.

Цель и задачи: Целью проекта является создание эффективной системы учета и мониторинга данных о счетчиках, что позволит оптимизировать управление энергоресурсами. Для достижения этой цели необходимо определить требования к базе данных, разработать ее структуру, а также создать приложение, обеспечивающее удобное взаимодействие с этой базой данных.

Предполагаемые методы и способы решения: для разработки базы данных и приложения будут использованы современные методы и инструменты программирования, такие как языки SQL и C#, а также соответствующие фреймворки и библиотеки. Это позволит обеспечить эффективное и надежное функционирование создаваемой системы.

1. СТРУКТУРА БАЗЫ ДАННЫХ

1.1 СХЕМА БАЗЫ ДАННЫХ

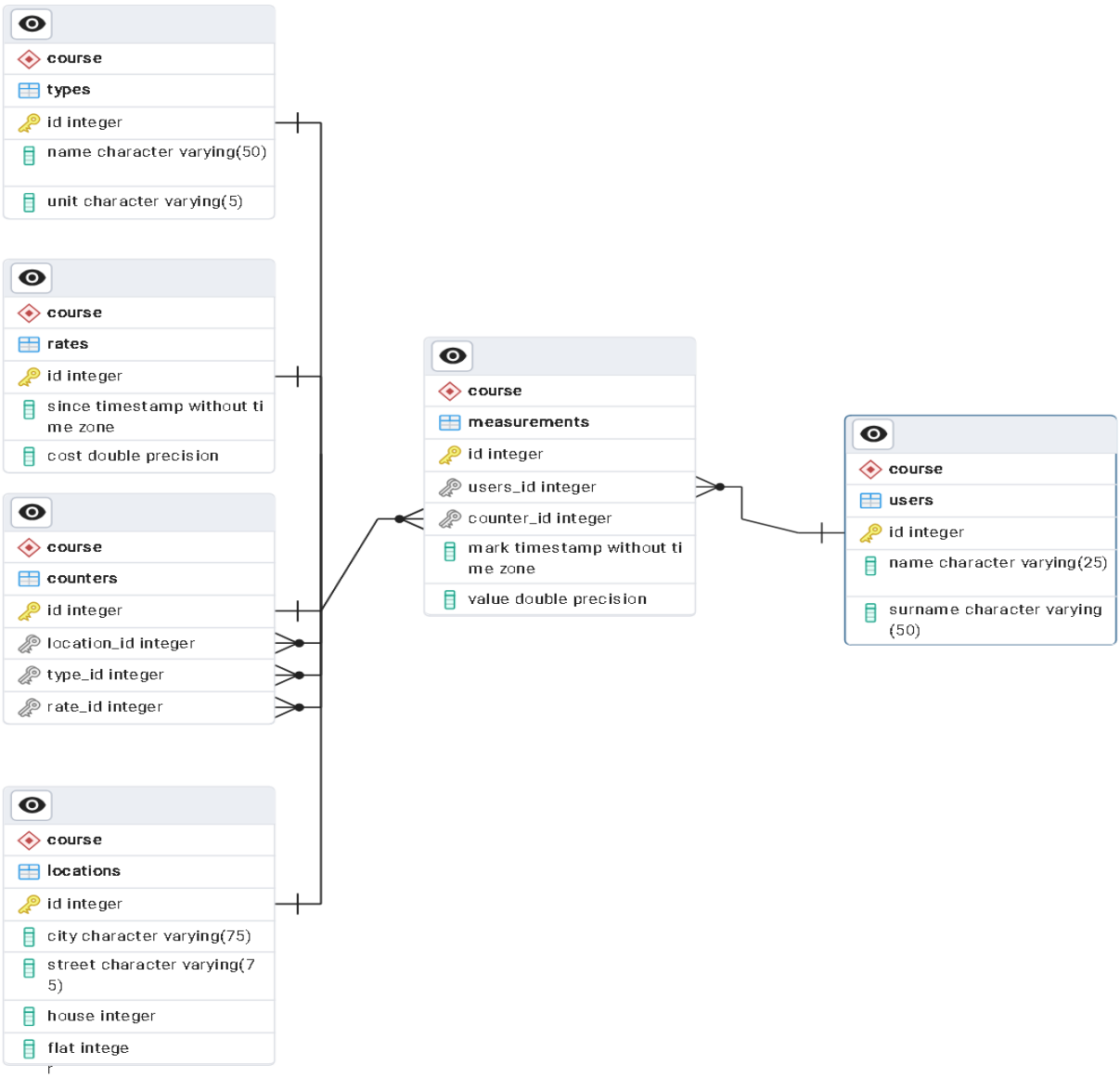


Рисунок 1 – ER диаграмма базы данных

База данных состоит из 6 таблиц, размещённых в схеме course:

1. Locations – содержит полные адреса город, улица, дом, квартира
2. Users – содержит личную информацию о пользователях
3. Rates – содержит историю тарифов по счётчикам
4. Types – содержит типы счётчиков
5. Counters – содержит полную информацию о счётчике(-ах)
6. Measurments – содержит информацию последнего измерения счётчика

7.

1.2 Структура таблиц

Рассмотрим таблицу types (Рис. 2)

№	Имя	Тип	Размер	Значение по умолчаию	Первичный ключ	Уникально	Описание
1	id	integer	-	Nextval(‘sq_types_id’)	Yes	Yes	Номер типа счётчика
2	name	varchar	50	-	No	No	Названия типа счётчика
3	unit	varchar	5	-	No	No	Единицы измерения

Рисунок 2 – Описание полей таблицы types

Таблица содержит 3 столбца:

1. Id – содержит уникальные номера каждого типа счётчика
2. name – содержит названия типа счётчика, данный столбец обязан содержать название типов больше 5 символов
3. unit – содержит единицы измерения типа счётчика, данный столбец обязан содержать единицы измерения больше или равно 2 символам

На рисунке 3 приведён пример данных, находящихся в таблице types

id [PK] integer	name character varying (20)	unit character varying (5)
1	power	kW
2	power_day	kW
3	power_night	kW
4	cold water	m^3
5	gas meter	m^3
6	hot water	m^3
7	heat meters	Gcal

Рисунок 3 – Пример данных в таблице types

На рисунке 4 изображён код для данной таблицы types

```
create table types (
    id integer default nextval('sq_types_id') primary key,
    name varchar(50) not null,
    unit varchar(5) not null);
```

Рисунок 4 – код таблицы types

Рассмотрим таблицу locations (Рис. 5)

№	Название	Тип	Значения по умолчанию	Первичный ключ	Внешний ключ	Уникальность	Описание
1	id	integer	Nextval ('sq_locations_id')	Yes	-	yes	Уникальный номер
2	city	Char	-	No	-	no	Город
3	street	Char	-	no	-	no	Улица
4	house	Integer	-	no	-	no	Номер дома
5	flat	Integer	-	no	-	no	Номер квартиры

Рисунок 5 - Описание полей таблицы locations

Таблица содержит 5 столбцов:

1. id – уникальный номер каждой локации
2. city – город
3. street – улица
4. house – номер дома
5. flat – номер квартиры

На рисунке 6 приведён пример данных, находящихся в таблице locations

	id [PK] integer	city character varying (75)	street character varying (75)	house integer	flat integer
1	1	Санкт-Петербург	Лиговский	71	23
2	2	Санкт-Петербург	Новгородская	31	56
3	3	Санкт-Петербург	Лиговский	43	67
4	4	Санкт-Петербург	Старорусская	10	57
5	5	Санкт-Петербург	Кирочная ул.	14	58
6	6	Санкт-Петербург	Кирочная ул.	66	78
7	7	Санкт-Петербург	Лиговский пр.	74	56

Рисунок 6 – Пример данных в таблице locations

На рисунке 7 изображён код для данной таблицы locations

```
create table locations (
    id integer default nextval('sq_locations_id') primary key,
    city varchar(75) not null,
    street varchar(75) not null,
    house integer not null,
    flat integer not null);
```

Рисунок 7 - код таблицы locations

Рассмотрим таблицу users (Рис. 8)

№	Название	Тип	Значения по умолчанию	Первичный ключ	Внешний ключ	Уникальность	Описание
1	Id	Integer	Nextval(‘sq_users_id)	yes	-	yes	Уникальный номер
2	Name	char	25	no	-	no	Имя пользователя
3	Surname	char	50	no	-	no	Фамилия пользователя

Рисунок 8 - Описание полей таблицы users

Таблица содержит 3 столбца:

1. id -Уникальный номер каждого владельца счётчика
2. name - Имя владельца
3. surname - Фамилия владельца

На рисунке 9 приведён пример данных, находящихся в таблице users

	id [PK] integer	name character varying (25)	surname character varying (50)
5	5	JONES	MICHAEL
6	6	BROWN	WILLIAM
7	7	DAVIS	DAVID
8	8	MILLER	RICHARD
9	9	WILSON	CHARLES
10	10	MOORE	JOSEPH
11	11	TAYLOR	THOMAS
12	12	ANDERSON	CHRISTOPHER

Рисунок 9 – Пример данных в таблице users

На рисунке 10 изображён код для данной таблицы users

```
create table users (
    id integer default nextval('sq_users_id') primary key,
    name varchar(25) not null,
    surname varchar(50) not null);
```

Рисунок 10 – код таблицы owners

№	Название	Тип	Значения по умолчанию	Первичный ключ	Внешний ключ	Уникальность	Описание
1	id	integer	nextval ('sq_couners_id')	Yes	no	Yes	Номер счётчика
2	location_id	integer	-	No	location.id	Yes	Адрес счётчика
3	type_id	integer	-	No	types.id	No	Тип счётчика
4	rate_id	integer	-	No	rates.id	No	Тип счётчика

Рисунок 11 – Описание полей таблицы counters

Таблица имеет 3 столбца:

1. id – уникальный номер счётчика
2. location_id – уникальный номер локации расположения счётчика
3. type_id –уникальный номер типа счётчика
4. rate_id -уникальный номер тарифа счётчика

На рисунке 12 приведён пример данных, находящихся в таблице

	id [PK] integer	location_id integer	type_id integer	rate_id integer
1	1	1	1	[null]
2	2	2	4	[null]
3	3	3	3	[null]
4	4	4	2	[null]
5	5	5	5	[null]
6	6	6	6	[null]
7	7	7	2	[null]

Рисунок 12 – Пример данных в таблице counters

На рисунке 13 изображён код для данной таблицы counters

```
create table counters (
    id integer default nextval('sq_counters_id') primary key,
    location_id integer not null,
    type_id integer not null,
    rate_id integer,
    foreign key (location_id) references locations(id),
    foreign key (type_id) references types(id),
    foreign key (rate_id) references rates(id));
```

Рисунок 13 – код таблицы counters

№	Название	Тип	Значения по умолчанию	Первичный ключ	Уникальность	Описание
1	id	Integer	nextval('sq_rates_id')	yes	yes	Номер тарифа
2	since	timestamp	-	no	no	Время и дата вступления тарифа
3	cost	Float	-	no	no	Стоимость тарифа

Рисунок 14 – описание столбцов в таблице rates

Таблица имеет 3 столбца:

1. id – уникальный номер каждого нового тарифа
2. since – дата вступления нового тарифа
3. cost – стоимость тарифов

На рисунке 15 приведён пример данных, находящихся в таблице rates

	id [PK] integer	since timestamp without time zone	cost double precision
1	1	2020-01-01 00:00:00	5
2	2	2020-02-01 00:00:00	7
3	3	2020-03-01 00:00:00	9
4	4	2020-04-01 00:00:00	11
5	5	2020-05-01 00:00:00	4
6	6	2020-06-01 00:00:00	6
7	7	2020-07-01 00:00:00	8
8	8	2020-08-01 00:00:00	10
9	9	2020-09-01 00:00:00	12
10	10	2020-10-01 00:00:00	5
11	11	2020-11-01 00:00:00	7

Рисунок 15 – Пример данных в таблице rates

На рисунке 16 изображён код для данной таблицы rates

```
CREATE TABLE rates
(
    id integer primary key not null default nextval('sq_rates_id'),
    since timestamp not null,
    cost float not null
);
```

Рисунок 16 – код таблицы rates

Рассмотрим таблицу measurments (Рис. 17)

№	Название	Тип	Размер	Значения по умолчанию	Первичный ключ	Внешний ключ	Уникальность	Описание	
1	Id	integer	-	Nextval ('sq_measurments_id')	yes	no	yes	Номер показателя	
2	users_id	integer	-	-	no	users.id	no	Владелец счётчика	

3	counter_id	integer	-	-	no	counters.id	no	Счётчик	
4	mark	timestamp	-	-	no	no	no	Время замера счётчика	
5	value	float	-	-	no	no	no	Значение счётчика	

Рисунок 17 – описание столбцов таблицы measurments

Таблица содержит 5 столбцов:

1. Id – уникальный номер показателя счетчиков
2. users_id - номер владельца счётчика
3. counter_id – номер счётчика
4. mark – временная отметка измерений счётчиков
5. value – значение счётчиков

На рисунке 18 приведён пример данных, находящихся в таблице measurments

	id [PK] integer	users_id integer	counter_id integer	mark timestamp without time zone	value double precision
1	1	1	1	2021-11-21 15:36:00	1348.05
2	2	1	1	2021-12-09 20:52:00	1379.42
3	3	1	1	2022-01-01 01:56:00	1456.51
4	4	1	1	2022-01-07 14:30:00	1467.27
5	5	1	1	2022-01-22 14:10:00	1493.59
6	6	1	1	2022-03-08 14:24:00	1545.94
7	7	1	1	2022-03-15 21:06:00	1554.2

Рисунок 18 – Пример данных в таблице measurments

На рисунке 19 изображён код для данной таблицы measurments

```
create table measurements (
    id integer default nextval('sq_measurements_id') primary key,
    users_id integer not null,
    counter_id integer not null,
    mark timestamp not null,
    value float not null,
    foreign key (counter_id) references counters(id),
    foreign key (users_id) references users(id));
```

Рисунок 19 – Код таблицы measurments

2. СТРУКТУРА ПРИЛОЖЕНИЯ

Приложение написано на C# WinForms. Для этого внутри самого приложения надо указать данные для подключения к пустой базе данных.

Рисунок 20 представляет из себя наглядный пример конструктора приложения. В данном конструкторе можно заметить 2 Button, 1 ListBox, 1 monthCalendar.

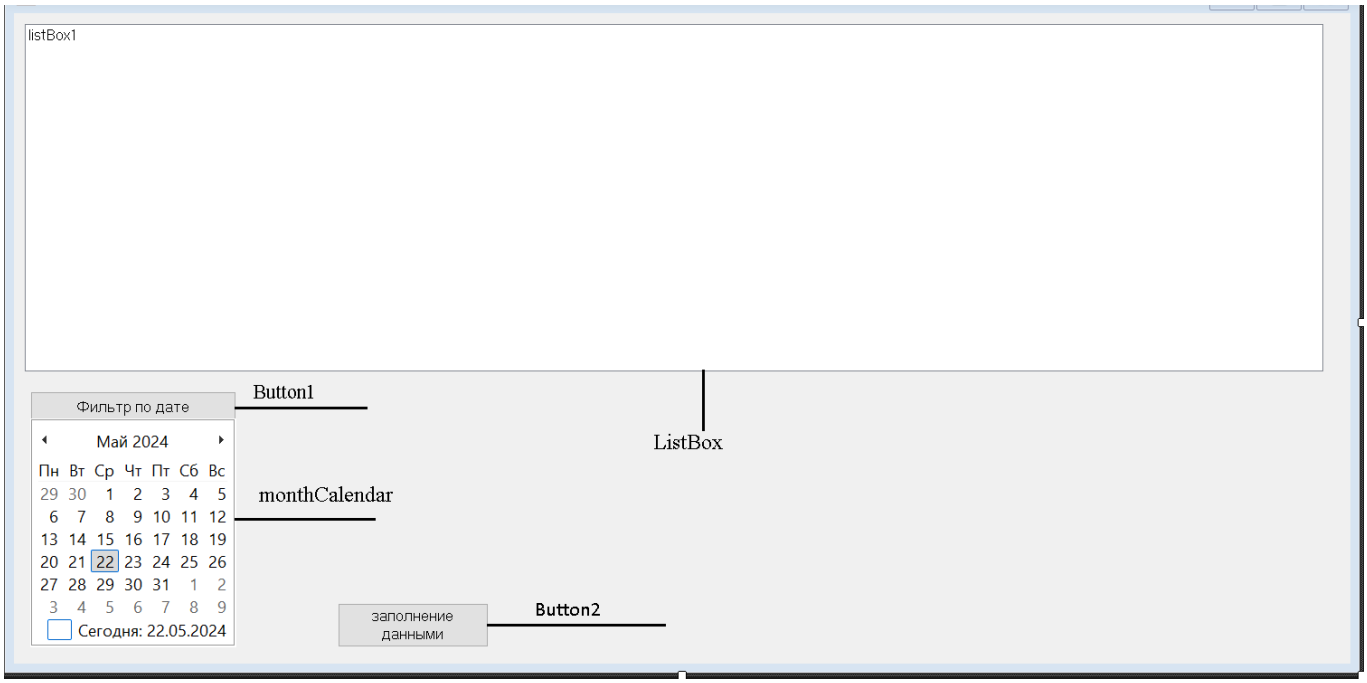
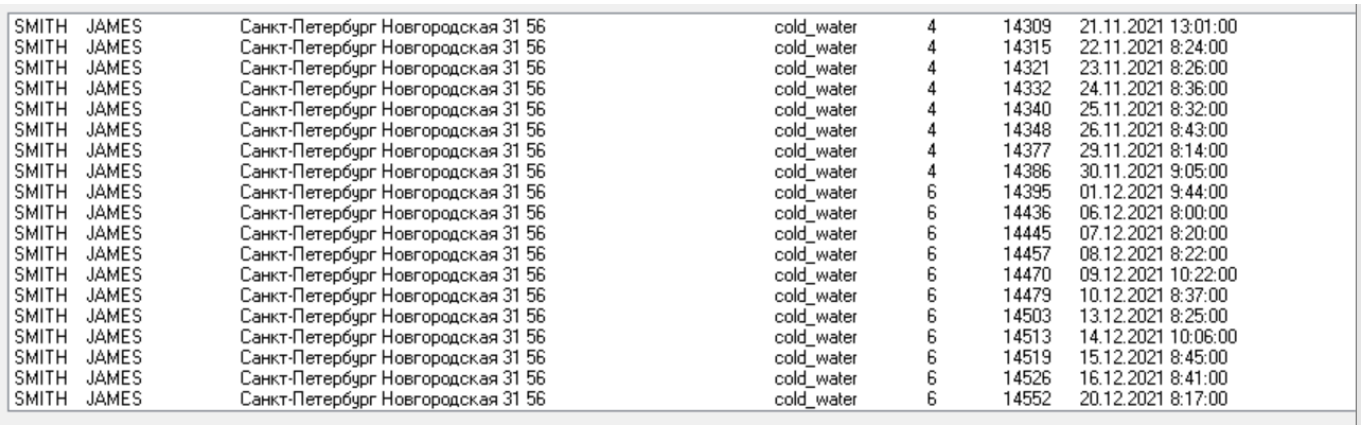


Рисунок 20 – Изображение приложения в конструкторе

В приложении всего 6 элементов:

1. ListBox – Данный элемент выводит данные из БД. (Рис.21)



SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	4	14309	21.11.2021 13:01:00
SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	4	14315	22.11.2021 8:24:00
SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	4	14321	23.11.2021 8:26:00
SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	4	14332	24.11.2021 8:36:00
SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	4	14340	25.11.2021 8:32:00
SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	4	14348	26.11.2021 8:43:00
SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	4	14377	29.11.2021 8:14:00
SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	4	14386	30.11.2021 9:05:00
SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	6	14395	01.12.2021 9:44:00
SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	6	14436	06.12.2021 8:00:00
SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	6	14445	07.12.2021 8:20:00
SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	6	14457	08.12.2021 8:22:00
SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	6	14470	09.12.2021 10:22:00
SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	6	14479	10.12.2021 8:37:00
SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	6	14503	13.12.2021 8:25:00
SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	6	14513	14.12.2021 10:06:00
SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	6	14519	15.12.2021 8:45:00
SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	6	14526	16.12.2021 8:41:00
SMITH	JAMES	Санкт-Петербург Новгородская 31 56	cold_water	6	14552	20.12.2021 8:17:00

Рисунок 21 – Пример работы ListBox

2. Button1 – Фильтр по дате
3. monthCalendar - Данный элемент позволяет выбрать дату до какого числа нужно вывести данные в ListBox
4. Button2 – Фильтрует данные до даты выбранной в monthCalendar.

Коды каждого элемента:

1. ListBox – код принимает входящие данные из бд (Рис. 22)

```
while (reader.Read())
{
    // Собираем строку для отображения в listBox1
    string value = $"{reader["users_name"]}\t\t" +
    $"{reader["locations"]}\t\t\t" +
    $"{reader["counter_type"]}\t" +
    $"{reader["rate_cost"]}\t" +
    $"{reader["value"]}\t" +
    $"{reader["mark"]}";

    // Добавляем строку в listBox1
    listBox1.Items.Add(value);
}
```

Рисунок 22 – Код заполнения ListBox

2. Button1 – Фильтрация данных по дате. Этот запрос SQL выбирает данные из нескольких таблиц базы данных PostgreSQL на основе определенных условий. Он соединяет таблицы "measurements", "counters", "users", "locations", "types" и "rates" с помощью операторов INNER JOIN. Затем он выбирает определенные столбцы из этих таблиц и выполняет несколько операций над данными, таких как объединение и преобразование типов. Наконец, запрос фильтрует данные по имени и фамилии владельца и сортирует результаты по идентификатору измерения "m.id".

```
using (NpgsqlConnection conn = new NpgsqlConnection(connString))
{
    // Открываем подключение
    conn.Open();

    string sql = "SELECT " +
    "m.id, " +
    "u.name || ' ' || u.surname as users_name, " +
    "l.city || ' ' || l.street || ' ' || l.house || ' ' || l.flat as locations," +
    "t.name as counter_type, " +
    "CASE " +
    "WHEN r.cost IS NULL THEN 'N/A' " +
    "ELSE CAST(r.cost AS TEXT) " +
    "END as rate_cost, " +
    "m.value, " +
    "m.mark, " +
    "r.id as rate_id " + // Выбор rate_id из таблицы rates
    "FROM " +
    "measurements m " +
    "INNER JOIN counters c ON m.counter_id = c.id " +
    "INNER JOIN users u ON m.users_id = u.id " +
    "INNER JOIN locations l ON c.location_id = l.id " +
    "INNER JOIN types t ON c.type_id = t.id " +
    "INNER JOIN rates r ON m.mark >= r.since " + // Сравнение даты measurments(mark) с rates(since)
    "AND r.since = (SELECT MAX(since) FROM rates WHERE since <= m.mark) " + // Выбор ближайшей даты rates(since)
    "WHERE m.mark <= @selectedDate " +
    "ORDER BY m.id";
```

Рисунок 23 – Фильтрация по дате

3. MonthCalendar – Выбор даты до которой надо отфильтровать (Рис. 24)

```
DateTime selectedDate = monthCalendar1.SelectionStart;
```

Рисунок 24 – Вывод даты из элемента

ЗАКЛЮЧЕНИЕ

Данный курсовой проект посвящен разработке базы данных и созданию соответствующего приложения для управления и автоматизации баз данных, в контексте учета и мониторинга данных о потреблении ресурсов, таких как энергия.

В ходе работы были рассмотрены ключевые аспекты проекта, начиная с актуальности темы в свете постоянного развития информационных технологий. Разработка эффективных баз данных становится неотъемлемой частью современного мира, где эффективное управление данными играет ключевую роль в оптимизации управления энергоресурсами.

Основной целью проекта было создание надежной и функциональной базы данных, а также разработка приложения для взаимодействия с этой базой данных. Для достижения этой цели были определены требования к базе данных, разработана ее структура, а также создан пользовательский интерфейс для удобного доступа к данным.

Структура базы данных включает в себя восемь таблиц, каждая из которых имеет четко определенные поля и отвечает за хранение определенных категорий информации, начиная от данных об улицах и районах, заканчивая данными о тарифах и измерениях счетчиков.

Приложение, разработанное на языке программирования C# с использованием WinForms, предоставляет удобный интерфейс для работы с базой данных. Возможности фильтрации данных по различным параметрам, а также просмотр данных в удобном формате обеспечивают удобство использования приложения как для администраторов, так и для обычных пользователей.

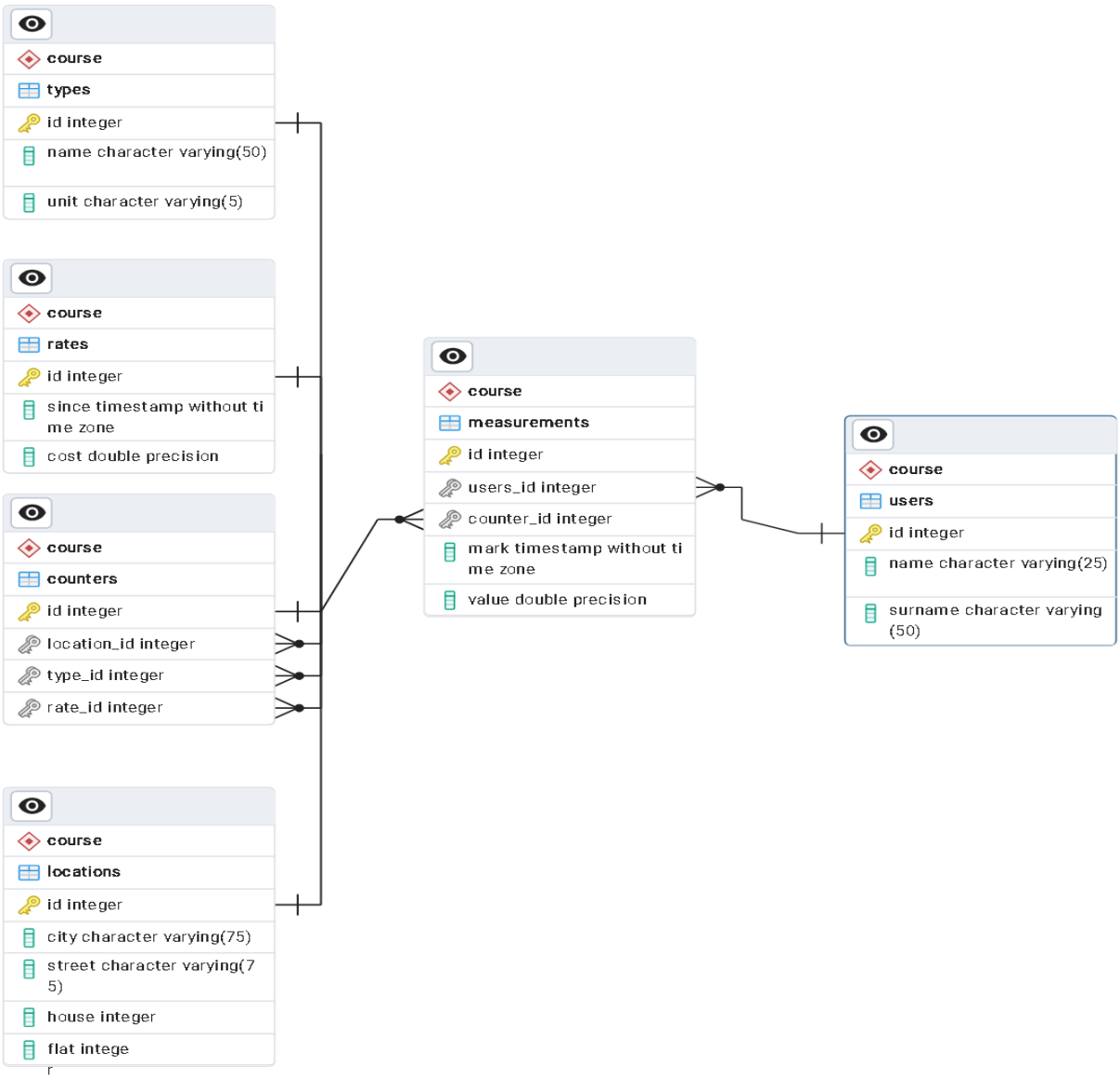
Таким образом, данный проект не только соответствует поставленным целям и задачам, но и представляет собой значимый вклад в область управления и автоматизации баз данных, способствуя оптимизации управления энергоресурсами и повышению эффективности процессов учета и мониторинга данных.

ЛИТЕРАТУРА

1. "Официальная документация PostgreSQL" (https://www.postgresql.org/docs/)
2. "Apress. PostgreSQL для профессионалов" (Айман Гиллеспи, 2020)
3. "Packt Publishing. C# 8.0 и платформа .NET Core 3.0" (Пол Деккер и др., 2019)
4. "Технология разработки программного обеспечения: учебное пособие" (Г.Н Федорова, 2020)
5. "C# Programming for Beginners" (Тим Академия, 2019)
6. "Основы разработки приложений на платформе Microsoft .NET Framework с использованием Windows Forms" (Microsoft)
7. "WinForms Programming in C#" (Различные авторы)
8. "Документация по среде разработки Microsoft Visual Studio" (https://docs.microsoft.com/)
9. "Документация по системе управления базами данных PostgreSQL" (https://www.postgresql.org/docs/)
10. "Metanit. Документация по фреймворку базы данных" (https://metanit.com/)
11. "Microsoft. Документация по Windows Forms" (https://docs.microsoft.com/)
12. "Microsoft. Документация по .NET Framework" (https://docs.microsoft.com/)
13. "Microsoft. Документация по C# programming language" (https://docs.microsoft.com/)
14. "C# Station. WinForms Tutorial" (https://www.csharp-station.com/tutorial.aspx)
15. "Codecademy. Learn C#" (https://www.codecademy.com/learn/learn-c-sharp)

ПРИЛОЖЕНИЕ

A Приложение ERD схема в расширениях pgerd и png



B Приложение Скрипт создание схемы в БД

```
drop schema if exists course cascade;

create schema course;


set search_path = 'course';


create sequence sq_users_id;
```

```
create sequence sq_types_id;
create sequence sq_locations_id;
create sequence sq_rates_id;
create sequence sq_counters_id;
create sequence sq_measurements_id;

create table users (
        id integer default nextval('sq_users_id') primary key,
        name varchar(25) not null,
        surname varchar(50) not null);
create table types (
        id integer default nextval('sq_types_id') primary key,
        name varchar(50) not null,
        unit varchar(5) not null);

create table locations (
        id integer default nextval('sq_locations_id') primary key,
        city varchar(75) not null,
        street varchar(75) not null,
        house integer not null,
        flat integer not null);

create table rates (
        id integer default nextval('sq_rates_id') primary key,
        since timestamp,
        cost float);

create table counters (
```

```
    id integer default nextval('sq_counters_id') primary key,
    location_id integer not null,
    type_id integer not null,
    rate_id integer,
    foreign key (location_id) references locations(id),
    foreign key (type_id) references types(id),
    foreign key (rate_id) references rates(id));

create table measurements (
    id integer default nextval('sq_measurements_id') primary key,
    users_id integer not null,
    counter_id integer not null,
    mark timestamp not null,
    value float not null,
    foreign key (counter_id) references counters(id),
    foreign key (users_id) references users(id));
```

alter table counters add constraint fk_counters_locations foreign key (location_id) references locations (id) on delete cascade;

alter table counters add constraint fk_counters_types foreign key (type_id) references types (id) on delete cascade;

alter table counters add constraint fk_counters_rates foreign key (rate_id) references rates (id) on delete cascade;

alter table measurements add constraint fk_measurements_counters foreign key (counter_id) references counters (id) on delete cascade;

--check

alter table locations add constraint chk_locations_city_length check (length (city) >= 2);

```
alter table locations add constraint chk_locations_street_length check (length (street) >= 2);

alter table locations add constraint chk_locations_house_positive check (house > 0);

alter table locations add constraint chk_locations_flat_positive check (flat > 0);


alter table rates add constraint chk_rates_since_after_date check (since >= '2020-01-01'::timestamp);

alter table rates add constraint chk_rates_since_cost_positive_or_zero check (cost > 0.0);


create function fn_last_counter_value(counter integer) returns float language sql as $$

        select value from measurements where counter_id = counter order by mark desc limit 1;

    $$;


alter table measurements add constraint chk_measurements_mark_after_date check (mark >= '2020-01-01'::timestamp);

alter table measurements add constraint chk_measurements_mark_not_in_future check (mark < now());

alter table measurements add constraint chk_measurements_value_positive_or_zero check (value >= 0.0);
```

C Приложение Разработка приложения C# WinForms

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System;
using System.Data;
```

```
using System.Data.SqlClient;
using Npgsql;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
using System.Xml.Linq;

namespace cource_db
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();

      }
      string connString =
"Host=localhost;Port=5432;Username=postgres;Password=1;Database=postgres";


      private void MainForm_Load(object sender, EventArgs e)
      {
         using (NpgsqlConnection conn = new NpgsqlConnection(connString))
         {
            conn.Open();
            string copyTypesCountersSql = "COPY types(name, unit) FROM
'C:/hhh/db/type_counters.csv' DELIMITER ',' CSV;";
            using (NpgsqlCommand copyTypesCountersCmd = new
NpgsqlCommand(copyTypesCountersSql, conn))
            {
               copyTypesCountersCmd.ExecuteNonQuery();
            }
         }

         using (NpgsqlConnection conn = new NpgsqlConnection(connString))
         {
            conn.Open();
            string CopyLocationsSql = "COPY locations(city, street, house, flat) FROM
'C:/hhh/db/course_work/Locations.csv' DELIMITER ',' CSV; ";
            using (NpgsqlCommand generateAreasStreetsCmd = new
NpgsqlCommand(CopyLocationsSql, conn))
            {
               generateAreasStreetsCmd.ExecuteNonQuery();
            }
         }
```

```
        using (NpgsqlConnection conn = new NpgsqlConnection(connString))
        {
            conn.Open();
            string copyRatesSql = "COPY rates(since, cost) FROM
'C:/hhh/db/course_work/Rates.csv' DELIMITER ',' CSV;";
            using (NpgsqlCommand copyRatesCmd = new NpgsqlCommand(copyRatesSql,
conn))
            {
                copyRatesCmd.ExecuteNonQuery();
            }
        }

        using (NpgsqlConnection conn = new NpgsqlConnection(connString))
        {
            conn.Open();
            string copyNameSql = "COPY users(name, surname) FROM
'C:/hhh/db/Code_data_all_name.csv' DELIMITER ',' CSV;";
            using (NpgsqlCommand copyNameCmd = new
NpgsqlCommand(copyNameSql, conn))
            {
                copyNameCmd.ExecuteNonQuery();
            }
        }



    LoadMeasurements();
    }

    private void LoadMeasurements()
    {
        using (NpgsqlConnection conn = new NpgsqlConnection(connString))
        {
            conn.Open();

            string sql = "SELECT " +
            "m.id, " +
            "u.name || ' ' || u.surname as users_name, " +
            "l.city || ' ' || l.street || ' ' || l.house || ' ' || l.flat as locations," +
            "t.name as counter_type, " +
            "CASE " +
            "WHEN r.cost IS NULL THEN 'N/A' " +
            "ELSE CAST(r.cost AS TEXT) " +
```

```
            "END as rate_cost, " +
            "m.value, " +
            "m.mark, " +
            "r.id as rate_id " +
            "FROM " +
            "measurements m " +
            "INNER JOIN counters c ON m.counter_id = c.id " +
            "INNER JOIN users u ON m.users_id = u.id " +
            "INNER JOIN locations l ON c.location_id = l.id " +
            "INNER JOIN types t ON c.type_id = t.id " +
            "INNER JOIN rates r ON m.mark >= r.since " +
            "AND r.since = (SELECT MAX(since) FROM rates WHERE since <= m.mark)
" +
            "ORDER BY m.id";


            using (NpgsqlCommand cmd = new NpgsqlCommand(sql, conn))
            {

               using (NpgsqlDataReader reader = cmd.ExecuteReader())
               {

                  listBox1.Items.Clear();


                  while (reader.Read())
                  {

                     string value = $"{reader["users_name"]}\t\t" +
                     $"{reader["locations"]}\t\t\t" +
                     $"{reader["counter_type"]}\t" +
                     $"{reader["rate_cost"]}\t" +
                     $"{reader["value"]}\t" +
                     $"{reader["mark"]}";


                     listBox1.Items.Add(value);
                  }
               }
            }
         }
      }

      private void button1_Click(object sender, EventArgs e)
```

```
{
    DateTime selectedDate = monthCalendar1.SelectionStart;

    using (NpgsqlConnection conn = new NpgsqlConnection(connString))
    {

        conn.Open();

        string sql = "SELECT " +
        "m.id, " +
        "u.name || ' ' || u.surname as users_name, " +
        "l.city || ' ' || l.street || ' ' || l.house || ' ' || l.flat as locations," +
        "t.name as counter_type, " +
        "CASE " +
        "WHEN r.cost IS NULL THEN 'N/A' " +
        "ELSE CAST(r.cost AS TEXT) " +
        "END as rate_cost, " +
        "m.value, " +
        "m.mark, " +
        "r.id as rate_id " +
        "FROM " +
        "measurements m " +
        "INNER JOIN counters c ON m.counter_id = c.id " +
        "INNER JOIN users u ON m.users_id = u.id " +
        "INNER JOIN locations l ON c.location_id = l.id " +
        "INNER JOIN types t ON c.type_id = t.id " +
        "INNER JOIN rates r ON m.mark >= r.since " +
        "AND r.since = (SELECT MAX(since) FROM rates WHERE since <= m.mark)
" +
        "WHERE m.mark <= @selectedDate " +
        "ORDER BY m.id";


        using (NpgsqlCommand cmd = new NpgsqlCommand(sql, conn))
        {

            cmd.Parameters.AddWithValue("@selectedDate", selectedDate);

            using (NpgsqlDataReader reader = cmd.ExecuteReader())
            {

                listBox1.Items.Clear();
```

```
                while (reader.Read())
                {

                    string value = $"{reader["users_name"]}\t" +
                    $"{reader["locations"]}\t" +
                    $"{reader["counter_type"]}\t" +
                    $"{reader["rate_cost"]}\t" +
                    $"{reader["value"]}\t" +
                    $"{reader["mark"]}";


                    listBox1.Items.Add(value);
                }
            }
        }
    }

    private void button2_Click(object sender, EventArgs e)
    {
        LoadMeasurements();
    }
}
}
```