

1. База данных

Задача:

Создать базу данных (БД) для табеля учета рабочего времени.

Сущности БД:

1. Департаменты;
2. Сотрудники;
3. Производственный календарь – календарь рабочих, выходных, предпраздничных (эти рабочие дни сокращаются на 1 час) и праздничных дней на год;
4. Виды отметок в табеле:
 - Я – полный рабочий день;
 - Н – отсутствие на рабочее место по невыясненным причинам;
 - В – выходные и праздничные дни;
 - Рв – работа в праздничные и выходные дни; а также работа в праздничные и выходные дни, при нахождении в командировке;
 - Б – дни временной нетрудоспособности;
 - К – командировочные дни, а также выходные дни при нахождении в командировке, когда сотрудник отдыхает;
 - ОТ – ежегодный основной оплаченный отпуск;
 - До – неоплачиваемый отпуск (отпуск за свой счет);
 - Хд – хозяйственный день;
 - У – отпуск на период обучения;
 - Ож – отпуск по уходу за ребенком.

Результат:

Схема спроектированной БД в графическом виде и с текстовым описанием каждого объекта (таблицы, поля, связи).

Выполнение:

В базе данных представлено 4 сущности: **Департамент**, **Сотрудники**, **Производственный календарь**, **Отметка в табеле**.

В таблице **Департамент** я выделил 4 поля: **Код департамента**, **Название департамента**, **Местоположение** и **Задача**. Таким образом: таблица имеет простую форму, в ней отсутствуют составные поля и нет одинаковых по смыслу полей. Т.е. таблица приведена к 1 нормальной форме.

Первичным ключом в таблице **Департамент** я выделил поле **Код департамента**, т.к. оно однозначно идентифицирует запись в таблице. Т.к. в разделении таблицы на несколько

таблиц нет необходимости, а транзитивных зависимостей в таблице тоже нет, то получается, что таблица приведена к 3 нормальной форме.

По тому же принципу были созданы остальные таблицы. Первичным ключом в таблице сотрудники я не стал брать поля **Имя, Фамилия**, так как есть вероятность того, что имя и фамилия сотрудников совпадут, и причем вместо этих полей у нас есть другое поле однозначно определяющее запись. Это поле **Табельный №**.

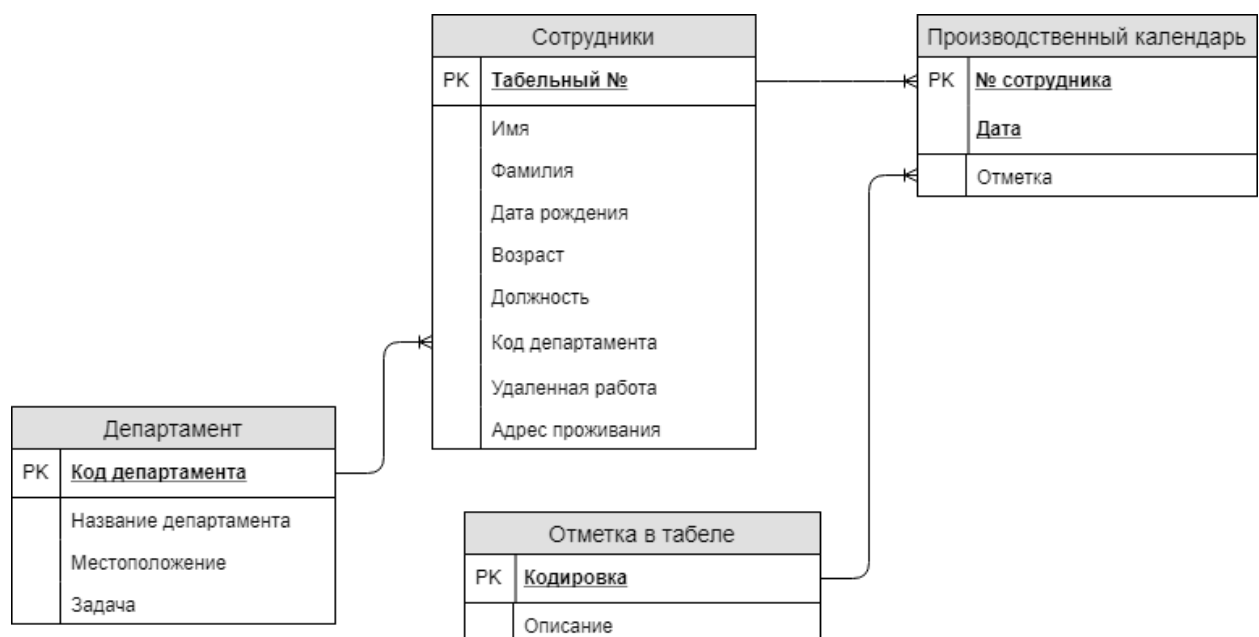
В других полях всё намного проще. В таблице **Производственный календарь** первичным ключом является **№ сотрудника, Дата**, а в таблице **Отметка в табеле** – **Кодировка**.

Опишем связи между таблицами. Каждый сотрудник (таблица **Сотрудники**) может работать в нескольких департаментах, а в таблице **Департамент** имеются всего по одной записи об одном конкретном департаменте с его уникальным **Кодом департамента**. Поэтому связь между таблицами **Департамент-Сотрудники** Один-ко-Многим. Внешний ключ таблицы **Сотрудники** – **Код департамента**.

В таблице **Производственный календарь** имеются множество дат со своим отметками для каждого сотрудника. А в таблице **Сотрудники** имеются всего по одной записи об одном конкретном сотруднике с его уникальным **Табельным №**. Таким образом, связь между таблицами **Сотрудники-Производственный календарь** Один-ко-Многим. Внешний ключ таблицы **Производственный календарь** – **Табельным №**.

В таблице **Отметка в табеле** имеются по одной уникальной записи кодировки со своим описанием. А в таблице **Производственный календарь** имеются множество отметок, относящихся к своим датам. Таким образом, снова связь между таблицами **Отметка в табеле-Производственный календарь** Один-ко-Многим. Внешний ключ таблицы **Производственный календарь** – **Отметка**.

Графическое представление базы данных выполнено с помощью инструментов сайта draw.io и представлено ниже:



Замечание:

В задании в сущности Производственный календарь перечислена информация о календаре рабочих, выходных, предпраздничных (эти рабочие дни сокращаются на 1 час) и праздничных дней на год. Но в видах отметок не сказано ничего о сокращенных на 1 час предпраздничных днях. Возможно требуется добавить отметку “Сокращенный день”. Тогда в программе нужно будет просто в таблицу **Отметка в таблице** добавить нужную кодировку с соответствующим описанием.

Скрипты:

В работе использовалась база данных - SQL Server. Ниже представлены скрипты базы данных:

1) Таблица – Департамент

```
1 CREATE TABLE [dbo].[Department] (  
2     [Код департамента] INT NOT NULL,  
3     [Название департамента] NVARCHAR (50) NULL,  
4     [Местоположение] NVARCHAR (50) NULL,  
5     [Задача] NVARCHAR (MAX) NULL,  
6     PRIMARY KEY CLUSTERED ([Код департамента] ASC)  
7 );
```

2) Таблица – Сотрудники

```
1 CREATE TABLE [dbo].[Employees] (  
2     [Табельный №] INT NOT NULL,  
3     [Имя] NVARCHAR (50) NOT NULL,  
4     [Фамилия] NVARCHAR (50) NOT NULL,  
5     [Дата рождения] DATE NULL,  
6     [Возраст] INT NULL,  
7     [Должность] NVARCHAR (50) NULL,  
8     [Код департамента] INT NOT NULL,  
9     [Удаленная работа] BIT NULL,  
10    [Адрес проживания] NVARCHAR (50) NULL,  
11    PRIMARY KEY CLUSTERED ([Табельный №] ASC),  
12    CONSTRAINT [FK_Employees_ToDepartment] FOREIGN KEY ([Код департамента]) REFERENCES [dbo].[Department] ([Код департамента])  
13 );
```

3) Таблица – Производственный календарь

```
1 CREATE TABLE [dbo].[Production_calendar] (  
2     [Дата] DATE NOT NULL,  
3     [№ сотрудника] INT NOT NULL,  
4     [Отметка] NVARCHAR (50) NULL,  
5     PRIMARY KEY CLUSTERED ([Дата] ASC, [№ сотрудника] ASC),  
6     CONSTRAINT [FK_Production_calendar_ToEmployees] FOREIGN KEY ([№ сотрудника]) REFERENCES [dbo].[Employees] ([Табельный №]),  
7     CONSTRAINT [FK_Production_calendar_ToMark_in_timetable] FOREIGN KEY ([Отметка]) REFERENCES [dbo].[Mark_in_timetable] ([Кодировка])  
8 );
```

4) Таблица – **Отметка в таблице**

```
1 CREATE TABLE [dbo].[Mark_in_timetable] (  
2     [Кодировка] NVARCHAR (50) NOT NULL,  
3     [Описание] NVARCHAR (MAX) NULL,  
4     PRIMARY KEY CLUSTERED ([Кодировка] ASC)  
5 );
```

2. Просмотр данных табеля

Задача:

Разработать программное обеспечение (ПО), которое позволит просматривать находящиеся в БД данные табеля.

Язык программирования – на свой выбор.

Пользователь ПО может просматривать информацию об отметках работников департаментов в разрезе месяца.

Результат:

Представить в виде исходного кода и исполняемого приложения.

Выполнение:

Язык программирования был выбран – C#. Вся разработка велась в среде разработки Visual Studio с помощью фреймворка Windows Forms.

Ниже представлен код главной формы Form1 с комментариями (комментарии указаны только в этом документе):

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Data.SqlClient; //Подключим библиотеки для работы с данными  
using System.Windows.Forms;  
  
namespace TestNauka  
{  
    public partial class Form1 : Form  
    {  
        SqlConnection sqlConnection; //К базе было много запросов из разных методов,  
        поэтому для удобства класс для подключения к базе SQL и следующий класс для чтения  
        потока данных сделал глобальными  
        SqlDataReader sqlReader;  
        List<string> listKodeDepartment = new List<string>();  
  
        public Form1()  
        {  
            InitializeComponent();  
        }  
    }  
}
```

```

    }

    private async void Form1_Load(object sender, EventArgs e) //почти все методы
асинхронные, чтобы можно было работать с приложением, не дожидаясь ответа
    {
        string stringConnection = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\TestNaukaBase.mdf;Integr
ated Security=True"; //Подключаемся к базе (причем путь относительный)
        sqlConnection = new SqlConnection(stringConnection);

        await sqlConnection.OpenAsync(); //Открываем подключение асинхронно
        sqlReader = null;
        SqlCommand commandListBox = new SqlCommand("SELECT [Код
департамента],[Название департамента] FROM [Department]", sqlConnection);
        SqlCommand commandTimetablesYears = new SqlCommand("SELECT [Дата] FROM
[Production_calendar]", sqlConnection); //Здесь пишем запросы к базе данных
        try
        {
            sqlReader = await commandListBox.ExecuteReaderAsync(); //Здесь
асинхронно эти запросы отправляются
            while (await sqlReader.ReadAsync()) //Здесь считывать данные по
запросу (асинхронно)
            {
                listBox1.Items.Add(sqlReader["Название департамента"].ToString());
                listKodeDepartment.Add(sqlReader["Код департамента"].ToString());
            }

            sqlReader.Close();
            listBox1.SetSelected(0, true);

            sqlReader = await commandTimetablesYears.ExecuteReaderAsync();
            string year = "";
            while (await sqlReader.ReadAsync())
            {
                if (year != sqlReader["Дата"].ToString().Substring(6, 4))
                {
                    year = sqlReader["Дата"].ToString().Substring(6, 4);
                    comboBoxYears.Items.Add(year);
                }
            }
        }
        catch(Exception ex) //Ловим ошибки, если вдруг не получилось подключиться к
базе или выполнить запрос
        {
            MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        Finally //Обязательно закрываем поток
        {
            if(sqlReader!=null)
                sqlReader.Close();
        }
    }

    //Обязательно закрываем соединение, чтобы не было утечки данных. Делаем это и
по нажатию крестика и по нажатию кнопки "Выход" в меню "Файл"
    private void выходToolStripMenuItem_Click_1(object sender, EventArgs e)
    {
        if (sqlConnection != null && sqlConnection.State != ConnectionState.Closed)
            sqlConnection.Close();
        Application.Exit();
    }

    private void Form1_FormClosing(object sender, FormClosingEventArgs e)
    {
        if (sqlConnection != null && sqlConnection.State != ConnectionState.Closed)
            sqlConnection.Close();
    }

```

```

        Application.Exit();
    }

    //При выборе Департамента в listBox1 (другие объекты имеют более адекватное
название) закрывается кнопка Поиска (т.к. для поиска нужно выбрать хотя бы табельный №)
и отправляются в соответствующий ComboBox список Табельных № сотрудников выбранного
департамента
    private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        Search.Enabled = false;

        string Inquiry = listKodeDepartment[listBox1.SelectedIndex];
        SqlCommand commandTimetablesEmployees = new SqlCommand("SELECT [Табельный
№] FROM [Employees] WHERE [Код департамента]=" + Inquiry, sqlConnection);
        sqlReader = commandTimetablesEmployees.ExecuteReader();
        comboBoxTimetables.Items.Clear();
        while (sqlReader.Read())
            comboBoxTimetables.Items.Add(sqlReader["Табельный №"].ToString());
        sqlReader.Close();
    }

    //Здесь идет вывод кодировок в раздел "Справки"
    private async void кодировкаОтметокToolStripMenuItem1_Click(object sender,
EventArgs e)
    {
        sqlReader = null;
        SqlCommand commandMark = new SqlCommand("SELECT * FROM
[Mark_in_timetable]", sqlConnection);
        try
        {
            sqlReader = await commandMark.ExecuteReaderAsync();
            string message="";
            while (await sqlReader.ReadAsync())
                message+=sqlReader["Кодировка"].ToString()+"\t"+
sqlReader["Описание"].ToString()+"\n";
            MessageBox.Show(message, "Кодировки");
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString() + "\n\nПовторите снова!",
ex.Source.ToString(), MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            if (sqlReader != null)
                sqlReader.Close();
        }
    }

    //Здесь идет вывод таблиц сотрудников в раздел "Справки"
    private async void таблицыСотрудниковToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        sqlReader = null;
        SqlCommand commandTimetablesYears = new SqlCommand("SELECT
[Фамилия],[Имя],[Табельный №] FROM [Employees] ORDER BY [Фамилия]", sqlConnection);
        try
        {
            sqlReader = await commandTimetablesYears.ExecuteReaderAsync();
            List<string> message=new List<string>();
            while (await sqlReader.ReadAsync())
                message.Add(sqlReader["Фамилия"].ToString() + " " +
sqlReader["Имя"].ToString() + "\t\t\t" + sqlReader["Табельный №"].ToString());

```

```

        FormTimeTableEmployees newForm = new FormTimeTableEmployees(this,
message);
        newForm.ShowDialog();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString()+"\n\nПовторите снова!",
ex.Source.ToString(), MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (sqlReader != null)
            sqlReader.Close();
    }
}

//По нажатию кнопки "Поиск" обновляются данные в разделе "Информация о
сотруднике"
private async void Search_Click(object sender, EventArgs e)
{
    sqlReader = null;
    listBoxFirstName.Items.Clear();
    listBoxLastName.Items.Clear();
    listBoxBirthDay.Items.Clear();
    listBoxAge.Items.Clear();
    listBoxPosition.Items.Clear();
    listBoxRemoteWork.Items.Clear();
    listBoxAddress.Items.Clear();

    SqlCommand commandEmployees = new SqlCommand("SELECT * FROM [Employees]
WHERE [Табельный №]="+comboBoxTimetables.SelectedItem, sqlConnection);
    try
    {
        sqlReader = await commandEmployees.ExecuteReaderAsync();
        await sqlReader.ReadAsync();
        listBoxFirstName.Items.Add(sqlReader["Имя"].ToString());
        listBoxLastName.Items.Add(sqlReader["Фамилия"].ToString());
        listBoxBirthDay.Items.Add(sqlReader["Дата
рождения"].ToString().Substring(0,10));
        listBoxAge.Items.Add(sqlReader["Возраст"].ToString());
        listBoxPosition.Items.Add(sqlReader["Должность"].ToString());
        if (sqlReader["Удаленная работа"].ToString() == "True")
            listBoxRemoteWork.Items.Add("Да");
        else if (sqlReader["Удаленная работа"].ToString() == "False")
            listBoxRemoteWork.Items.Add("Нет");
        else { }
        listBoxAddress.Items.Add(sqlReader["Адрес проживания"].ToString());

        sqlReader.Close();

        if (comboBoxMonth.SelectedItem != null && comboBoxYears.SelectedItem !=
null)
            CalendarInGridView(sender, e);

    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString() + "\n\nПовторите снова!",
ex.Source.ToString(), MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (sqlReader != null)
            sqlReader.Close();
    }
}

```

```

    }

    //По нажатию кнопки "Поиск" обновляются данные в разделе "Календарь". Здесь я
    использовал очень удобный инструмент GridView
    private void CalendarInGridView(object sender, EventArgs e)
    {
        string Inquiry = comboBoxTimetables.SelectedItem.ToString();
        string dateMonth = comboBoxMonth.SelectedIndex.ToString();
        dateMonth = dateMonth.Length == 1 ? "0" + dateMonth : dateMonth;
        string dateYears = comboBoxYears.SelectedItem.ToString();

        SqlCommand commandCalendar = new SqlCommand("SELECT [Дата],[Отметка] FROM
[Production_calendar] WHERE [№ сотрудника]=' + Inquiry, sqlConnection);
        sqlReader = commandCalendar.ExecuteReader();

        dataGridViewCalendar.Rows.Clear();
        List<string[]> listCalendar = new List<string[]>();
        while (sqlReader.Read())
            if (sqlReader[0].ToString().Substring(3, 7) == dateMonth + "." +
dateYears)
            {
                listCalendar.Add(new string[3]);
                listCalendar[listCalendar.Count - 1][0] =
sqlReader[0].ToString().Substring(0, 10);
                listCalendar[listCalendar.Count - 1][1] = sqlReader[1].ToString();
            }
        foreach (string[] s in listCalendar)
            dataGridViewCalendar.Rows.Add(s);
    }

    //При выборе табельного № в ComboBox мы можем использовать кнопку
    (блокирование кнопки предотвращает появление ошибок)
    private void comboBoxTimetables_SelectedIndexChanged(object sender,
EventArgs e)
    {
        Search.Enabled = true;
    }

    private void добавитьToolStripMenuItem_Click(object sender, EventArgs e)
    {
        FormLogin newForm = new FormLogin(this);
        newForm.ShowDialog();
    }
}

```

Далее представлен код той формы в разделе "Справки"=>"Табели сотрудников". Справка упрощает жизнь пользователя системой. Выполнен этот код в форме, а не в MessageBox, чтобы при большом количестве сотрудников было проще ориентироваться в их списке.

```

using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace TestNauka
{
    public partial class FormTimeTableEmployees : Form
    {
        public FormTimeTableEmployees(Form1 ParrentForm, List<string> message)
        {
            InitializeComponent();
            foreach(var i in message)
                listBox1.Items.Add(i);
        }
    }
}

```



```
}  
}  
}
```

3. Просмотр данных табеля

Задача:

Разработать функциональность, которая позволит вносить новые записи в БД.

Для ПО, разработанного на II этапе, создать функциональность по учету времени и ведению работников и отделов для пользователей разных ролей:

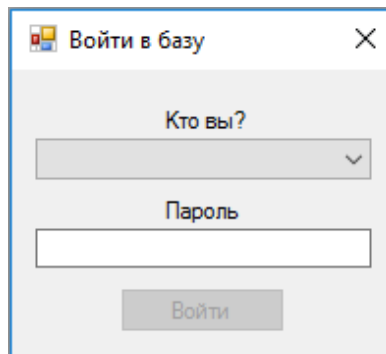
- Табельщик,
- Администратор справочника отделов (специалист отдела кадров),
- Администратор справочника работников (специалист отдела кадров).

Роли и функции:

1. Роль: табельщик – ежедневная отметка о рабочем времени работника.
Функции: отметка в календаре месяца в соответствующей дню месяца ячейке.
2. Роль: администратор справочника департаментов – содержание в актуальном состоянии справочника департаментов.
Функции: добавление/изменение департаментов в справочнике департаментов – отдельное окно.
3. Роль: администратор справочника работников – поддержание в актуальном состоянии данных о работниках предприятия.
Функции: добавление/изменение справочника работников - отдельное окно (пример прототипа ниже).

Выполнение:

В этом задании я очень сильно упростил себе жизнь и для по ролям я использовал обычную формочку с паролем. Она находится во вкладке “Функции”=> “Вход в систему”:



Внимание! Пароли для ролей:

Табельщик: 1111

Админ. справ. департамент.: 2222

Админ. справ. сотрудников: 3333

При переходе по каждой из ролей следуют отдельные формы для работы с базой данных. Они сделаны исключительно инструментами GridView. В них можно добавлять записи, удалять, изменять и сортировать по столбцам. Для того, чтобы изменения вступили в силу при закрытии формы “Войти в базу” программа перезагружается по методу:

```
private void FormLogin_FormClosed(object sender, FormClosedEventArgs e)
{
    Application.Restart();
}
```

Далее представлен код формы “Войти в базу”:

```
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;

namespace TestNauka
{
    public partial class FormLogin : Form
    {
        public FormLogin(Form1 ParrentForm)
        {
            InitializeComponent();
            buttonEnter.Enabled = false;
        }

        private void buttonEnter_Click(object sender, EventArgs e)
        {
            string role = comboBoxRoles.SelectedItem.ToString();
            string password = textBoxPassword.Text.ToString();
            if (role == "Табельщик" && password == "1111")
            {
                FormTimekeeper newForm = new FormTimekeeper(this);
                newForm.ShowDialog();
            }
            else if (role == "Админ. справ. департамент." && password == "2222")
            {
                FormAdminDepartment newForm = new FormAdminDepartment(this);
                newForm.ShowDialog();
            }
            else if (role == "Админ. справ. сотрудников" && password == "3333")
            {
                FormAdminEmployees newForm = new FormAdminEmployees(this);
                newForm.ShowDialog();
            }
            else
                MessageBox.Show("Пароль введен неверно!", "В доступе отказано");
        }

        private void comboBoxRoles_SelectedIndexChanged(object sender, EventArgs e)
        {
            buttonEnter.Enabled = true;
        }
    }
}
```

```
private void FormLogin_FormClosed(object sender, FormClosedEventArgs e)
{
    Application.Restart();
}
}
```