Лабораторная работа №2: Работа с базами данных

Цель работы:

- получение навыков работы с базами данных на языке C#;
- знакомство с принципами построения SQL-запросов.

Примечание:

В рамках данной лабораторной работы будет использоваться СУБД SQLite.

Создание и просмотр базы данных лучше всего осуществлять при помощи стороннего ПО. Например, DB Browser for SQLite.

Задание:

Разработать WPF-приложение с графическим интерфейсом и реализовать следующие функции:

1) ввод данных о студентах: уникальный номер, ФИО, оценка по физике, оценка по математике;
2) добавление данных в базу данных SQLite (далее - БД) через интерфейс приложения;
3) чтение данных из БД и отображение их в окне приложения;
4) редактирование данных в БД через интерфейс приложения;
5) удаление данных из таблиц.

БД должна содержать две таблицы, связанные через уникальный номер:

1. таблица, содержащая уникальный номер и ФИО;
2. таблица, содержащая уникальный номер и оценки.

Дополнительное задание в зависимости от последней цифры пароля:

Вариант 1: реализовать хранение и редактирование даты рождения студента;

Вариант 2: реализовать хранение и редактирование номера телефона студента;

Вариант 3: реализовать хранение и редактирование названия группы студента;

Вариант 4: реализовать хранение и редактирование признака наличия стипендии у студента (есть стипендия/нет стипендии).

Вариант выбирается исходя из последней цифры пароля от личного кабинета по следующей таблице:

	Последняя цифра пароля									
	0	1	2	3	4	5	6	7	8	9
Номер варианта	1	2	3	4	1	2	3	4	1	2

Результат выполнения работы предоставить в виде:

- архив с проектом (если размер архива больше 2 Мбайт, то рекомендуется загрузить проект на https://github.com/ или на другое общедоступное хранилище и предоставить ссылку);
- отчет по лабораторной работе в формате Microsoft Word, который содержит следующие разделы:

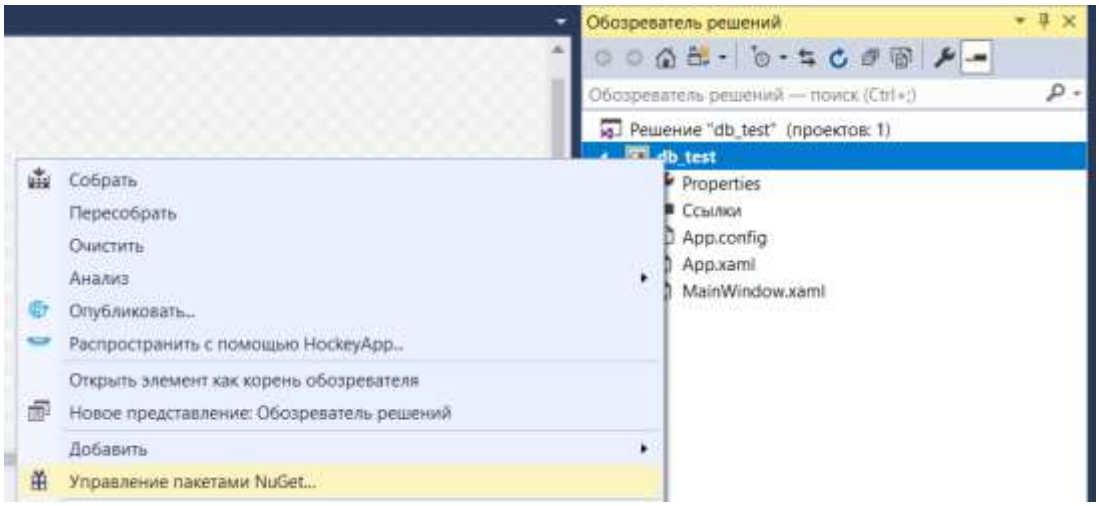
1. титульный лист;
2. задание на лабораторную работу;
3. краткое описание разработанных программ и используемых алгоритмов со скриншотами выполнения;

4. вывод с результатами работы.

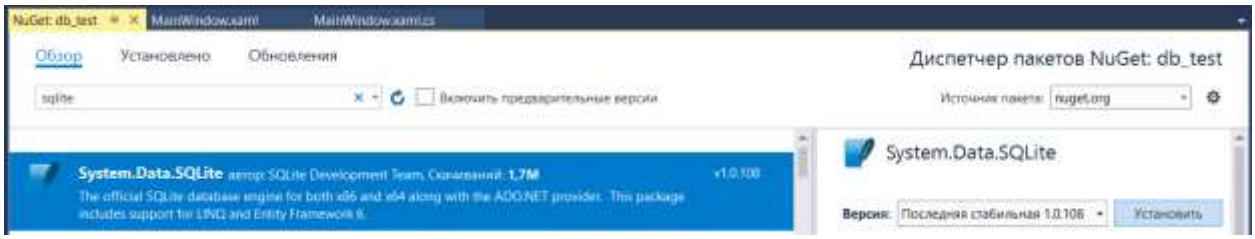
Копирование исходного кода программ не допускается. Проекты с одинаковым исходным кодом зачитываться не будут.

Справочная информация:

Для работы с БД SQLite необходимо подключить к проекту соответствующую библиотеку. Сделать это можно, нажав правой кнопкой мыши на название проекта и выбрав «Управление пакетами NuGet…»:

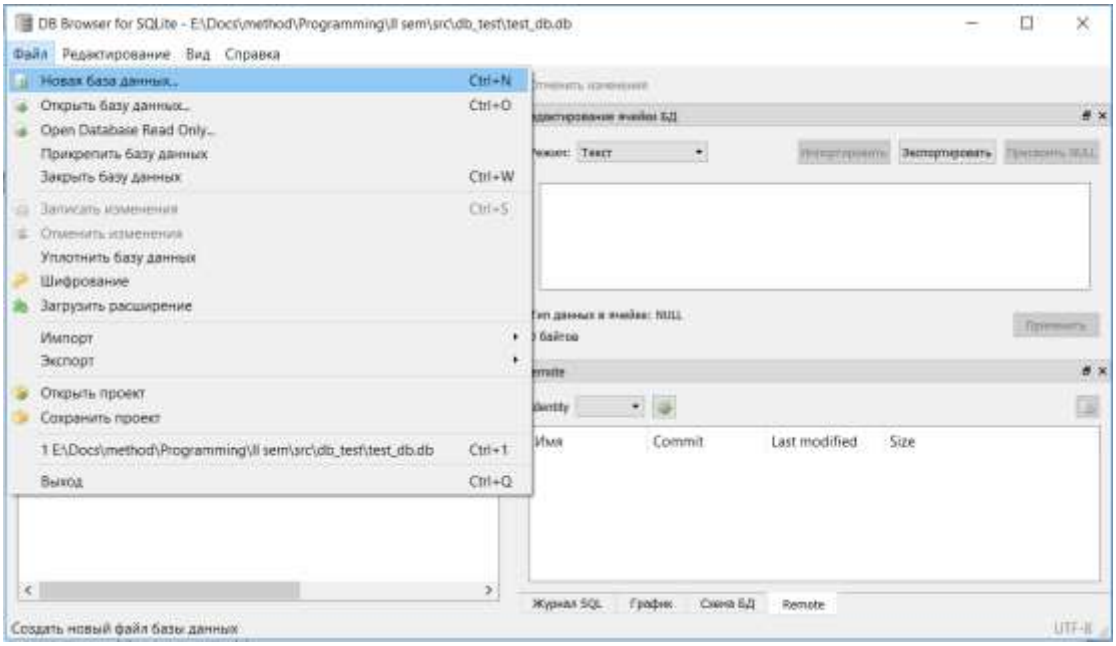


В разделе «Обзор», в строке поиска необходимо ввести «sqlite», после чего выбрать System.Data.SQLite и нажать «Установить»:

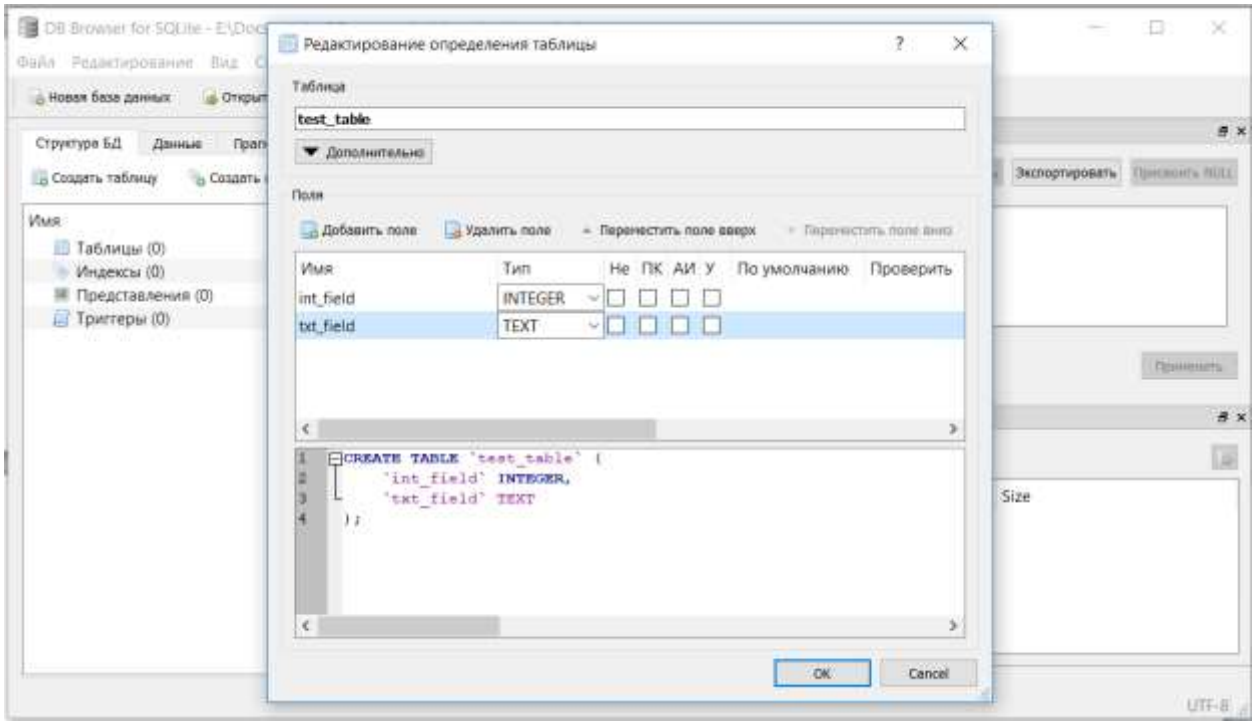


Создание таблиц при помощи DB Browser for SQLite:

После запуска DB Browser for SQLite необходимо выбрать «Файл» -> «Новая база данных»:



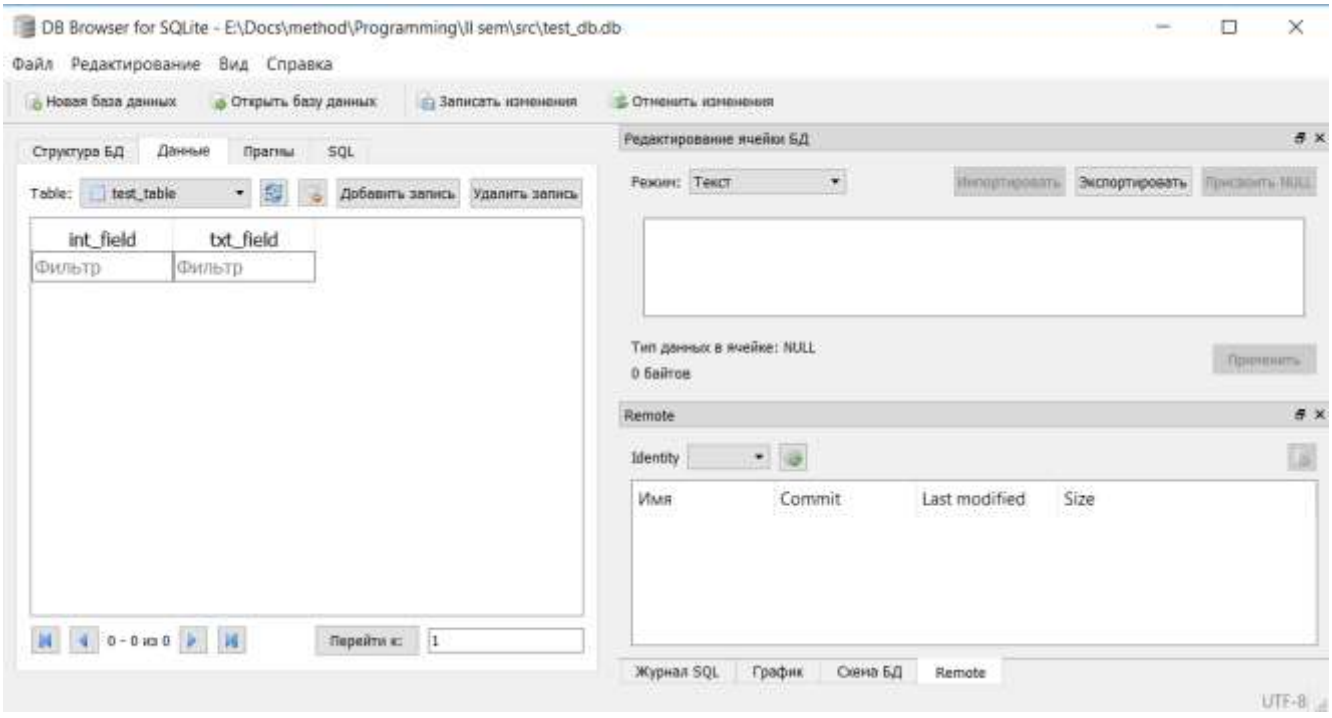
После создания базы данных будет предложено создать первую таблицу. Сделать это можно, указав имя таблицы и добавив поля:



Поля могут иметь один из нескольких типов, а так же дополнительные свойства:

- первичный ключ: поле используется для связи таблиц;
- автоинкремент: значение увеличивается автоматически;
- уникальное: значение должно быть уникальным.

Просмотреть или отредактировать таблицу можно на вкладке «Данные»:



Некоторые функции для работы с базами данных SQLite:

Подключение библиотеки:

```
using System.Data.SQLite;
```

Создание базы данных:

```
SQLiteConnection.CreateFile("D:\\MyDocs\\method\\intsys\\MyDatabase.sqlite");
```

Добавление таблицы:

```
string sql = "CREATE TABLE test (name VARCHAR(20), score INT)";
SQLiteCommand command = new SQLiteCommand(sql, m_dbConnection);
command.ExecuteNonQuery();
```

Подключение к уже существующей базе данных:

```
// имя базы данных
string db_name = dlg.FileName;

SQLiteConnection m_dbConnection;
m_dbConnection = new SQLiteConnection("Data Source=" + db_name + ";Version=3;");
// открытие соединения с базой данных
m_dbConnection.Open();

// выполнение запросов

// закрытие соединения с базой данных
m_dbConnection.Close();
```

Запись данных в уже существующую таблицу:

```
// формирование запроса на добавление данных в поля типа INTEGER и TEXT
// обратите внимание, что в текстовое поле данные добавляются в формате «data»
string sql = "INSERT INTO test_table (int_field, txt_field) VALUES (" + tb1.Text + ",'" + tb2.Text +
"')";
SQLiteCommand command = new SQLiteCommand(sql, m_dbConnection);
// извлечение запроса
command.ExecuteNonQuery();
```

Чтение данных из существующей таблицы:

```
string sql = "SELECT * FROM test_table ORDER BY int_field";
SQLiteCommand command = new SQLiteCommand(sql, m_dbConnection);
SQLiteDataReader reader = command.ExecuteReader();
while (reader.Read())
    list.Items.Add(reader["int_field"] + " : " + reader["txt_field"]);
```

Для получения номера последней добавленной записи можно выполнить SELECT-запрос следующего вида:

```
string sql = "SELECT int_field FROM test_table ORDER BY int_field DESC LIMIT 1";
```

Язык запросов SQL:

Для выполнения лабораторной работы понадобятся всего четыре типа запросов: INSERT, UPDATE, SELECT и DELETE.

Шаблон запроса **INSERT** выглядит следующим образом:

```
INSERT INTO название_таблицы (поле_1 [, поле_2, поле_3 ... ]) VALUES (значение_1 [, значение_2,
значение_3 ... ])
```

Шаблон запроса **UPDATE** выглядит следующим образом:

```
UPDATE название_таблицы SET название_поля = значение [,название_поля = значение ...] [WHERE условие]
```

Пример:

```
UPDATE test_table SET txt_field = ‘ten’ WHERE int_field = 10
```

Шаблон запроса **SELECT** выглядит следующим образом:

```
SELECT

определение, какие именно поля требуется выбрать (* - выбрать все поля)

FROM имя_таблицы

[WHERE условие_выбора]
```

```
[GROUP BY условие_группирования]
[HAVING условие_наличия]
[ORDER BY условие_сортировки]
```

Пример запроса по двум таблицам:

```
SELECT * FROM table_1, table_2 WHERE table_1.id = table_2.user_id
```

Шаблон запроса **DELETE** выглядит следующим образом:

```
DELETE FROM имя_таблицы [WHERE условие];
```

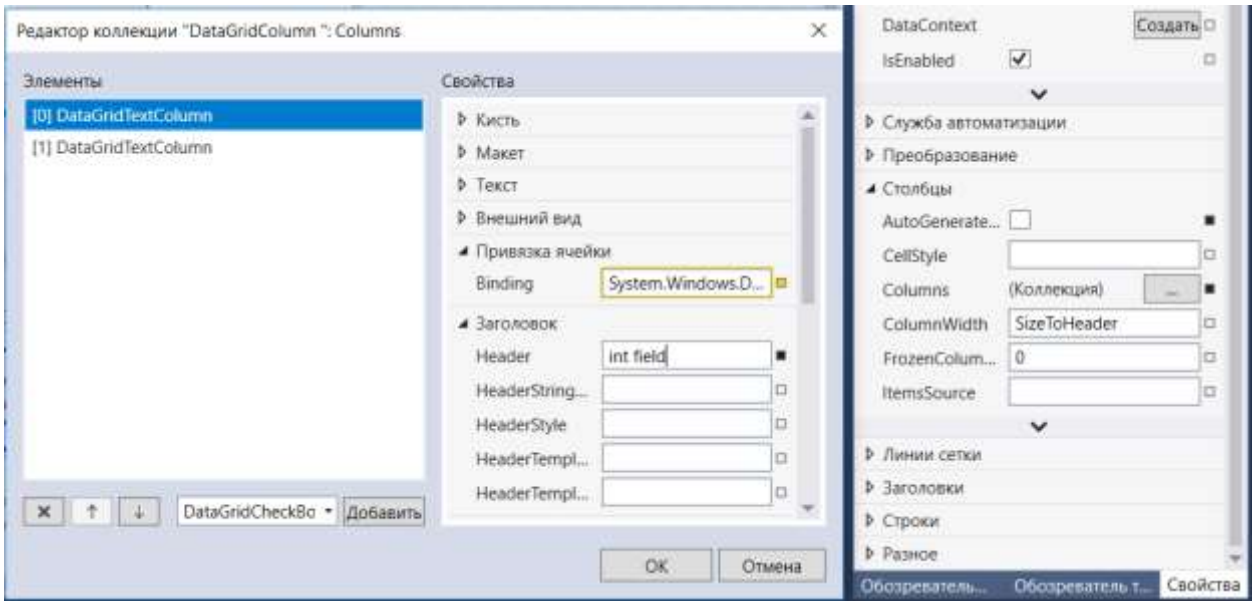
Пример:

```
DELETE FROM test_table WHERE int_field = 10;
```

Компонент DataGrid

Для отображения содержимого базы данных в виде таблицы рекомендуется использовать компонент DataGrid.

Добавить столбцы можно следующим образом:



1. перейти в раздел «Столбцы»;
2. Нажать «Коллекция»;
3. выбрать компонент «DataGridTextColumn»;
4. указать имя столбца в разделе «Заголовок»;
5. связать столбец с данными в разделе «Привязка ячейки» (в данном случае в поле «Binding» нужно прописать `{Binding int_field}`).

Следующий шаг - создание структуры данных, описывающих строку таблицы:

```
public class CTest
{
    public int int_field { get; set; }
    public string txt_field { get; set; }
}
```

Имена полей структуры должны совпадать с теми, что были указаны в «Binding».

Чтобы добавить в DataGrid строку, считанную из таблицы, можно использовать следующий код:

```
// создание строки
var data = new CTest { int_field = int.Parse(reader["int_field"].ToString()), txt_field =
reader["txt_field"].ToString() };
// добавление строки в DataGrid
```

```
datgrid.Items.Add(data);
```

Чтобы получить данные, выбранные в DataGrid, можно использовать:

```
private void datgrid_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    // получение строки из DataGrid
    CTest test = (CTest)datgrid.SelectedItem;

    MessageBox.Show(test.int_field.ToString() + " " + test.txt_field);
}
```

Список литературы:

1. Компонент DataGrid:

   https://msdn.microsoft.com/ru-ru/library/system.windows.controls.datagrid(v=vs.110).aspx

2. Справочник с примерами по языку SQL:

   https://learn.microsoft.com/ru-ru/sql/t-sql/queries/queries?source=recommendations&view=sql-server-ver16

3. Основы проектирования реляционных баз данных:

   http://citforum.ru/database/dbguide/index.shtml