**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский государственный политехнический

университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ**

**по предмету «Технология разработки и защиты**

**баз данных»**

**студента 32919/1 группы**

**специальности «Информационные системы и программирование»**

**Веригиной Любови Владимировны**

Санкт-Петербург

2023 г.

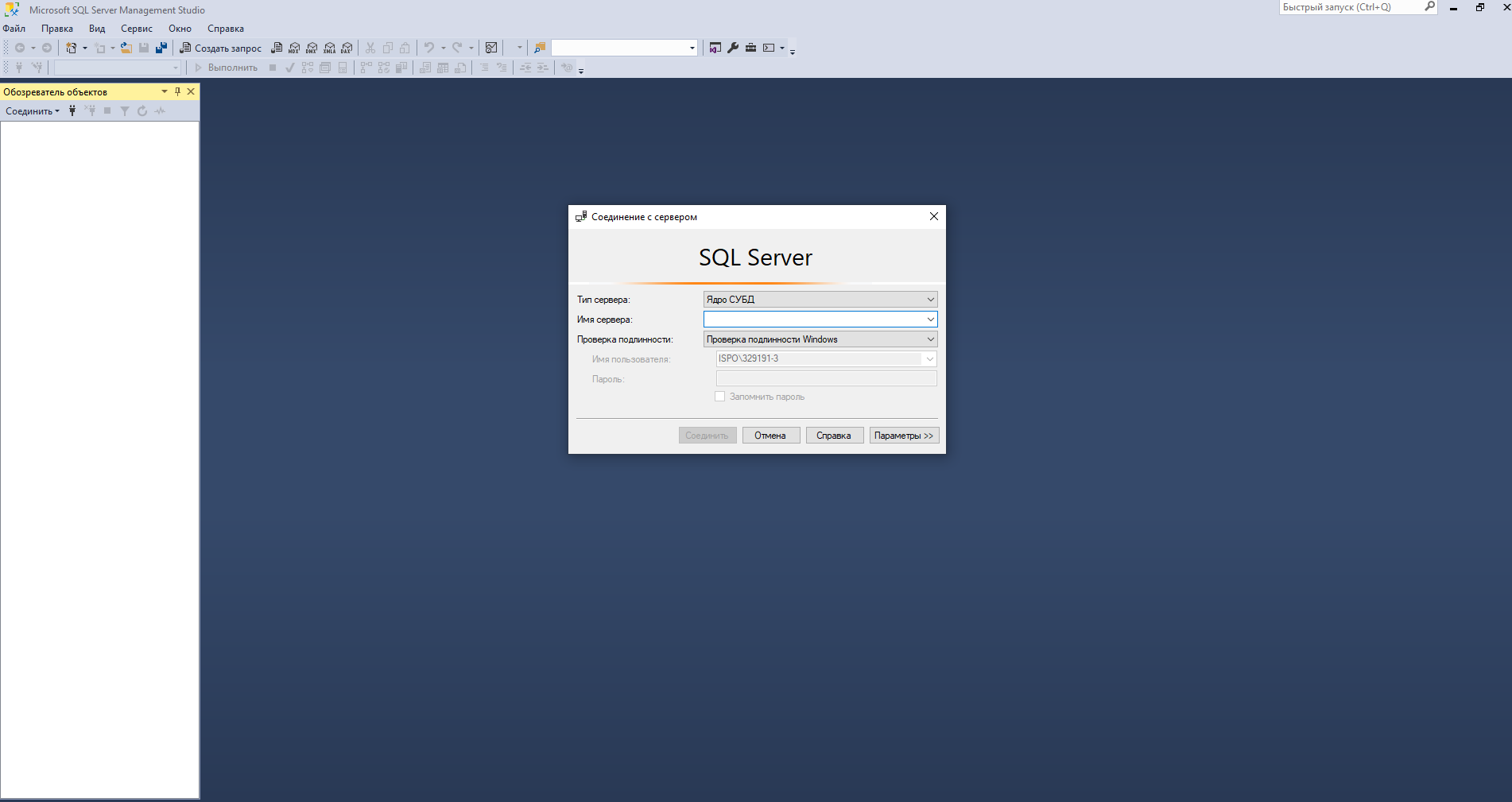
# **Лаборат****орная работа №1**

**«Установка соединения с сервером Microsoft SQL Server.**

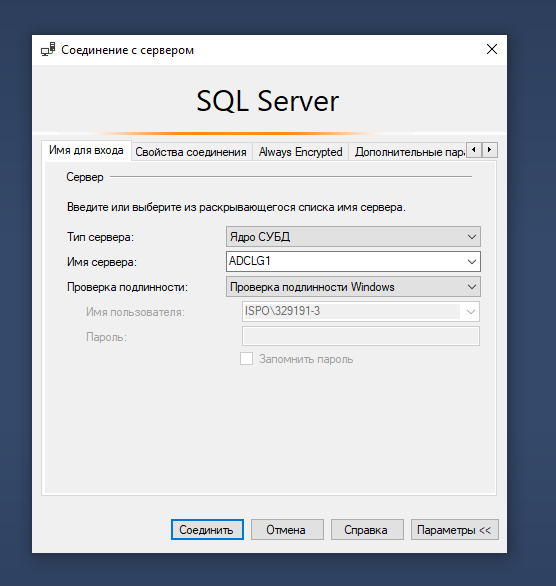
**Создание, копирование и удаление базы данных»**

**Цель:** Познакомиться с основными принципами создания базы данных в MS SQL Server. Получить навыки для создания, удаления, резервного копирования и восстановления базы данных. Изучить SQL-операторы для создания, подключения и удаления базы данных.

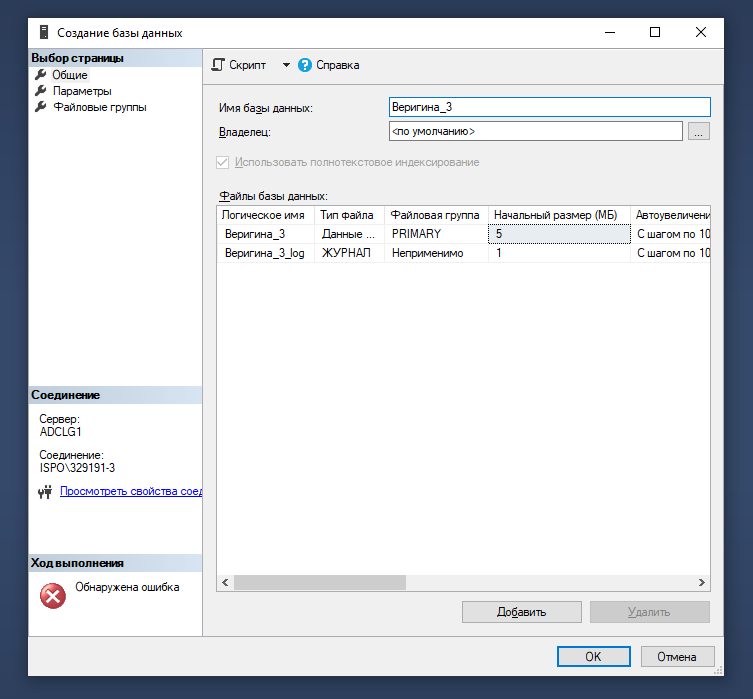
1\_1. Открыть SQL Server Management Studio.



1\_2. Ввести название сервера: ADCLG1 (DESKTOP-HG8OBH6\SQLEXPRESS)



1\_3.Создать БД



1\_4.

Команда для создания БД – CREATE DATABASE.

Синтаксис команды выглядит так:

CREATE DATABASE имя\_бд

ON

( NAME = логическое\_имя\_файла,

FILENAME = 'имя\_файа\_ос',

SIZE = размер, --пример 10

MAXSIZE = максимальный\_размер, --пример 30 или UNLIMITED

FILEGROWTH = шаг\_автоувеличения)

LOG ON --далее данные файла журнала

(NAME = логическое\_имя\_файла,

FILENAME = 'имя\_файа\_ос',

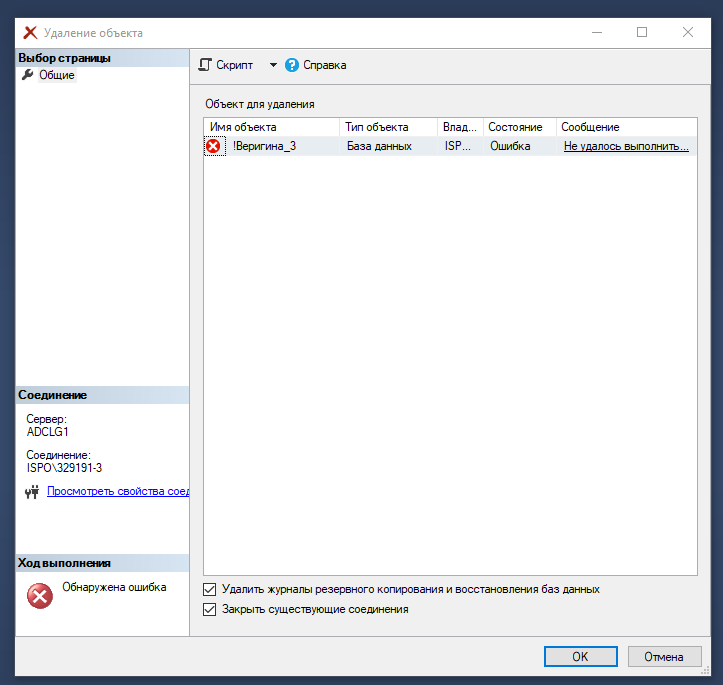
SIZE = размер, --пример 10

MAXSIZE = максимальный\_размер, --пример 30 или UNLIMITED

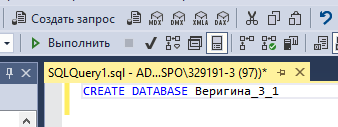
FILEGROWTH = шаг\_автоувеличения)

) ;

1\_5. Удалить с помощью контекстного меню (здесь у нас не прав администратора, поэтому это вызовет ошибку)

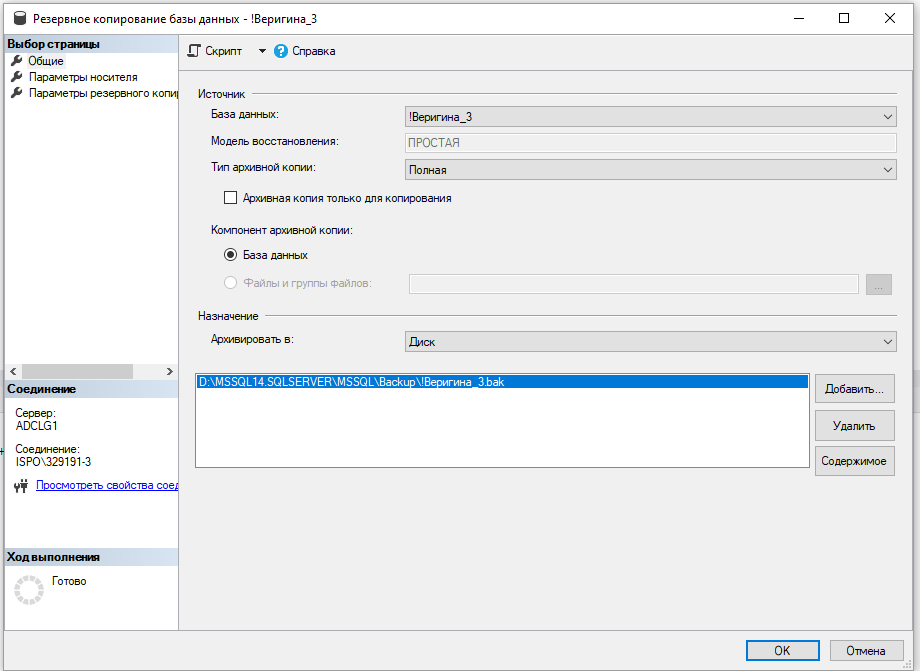


1\_6. Создать БД с помощью скрипта с именем ВашаФамилия\_номер\_журнала.

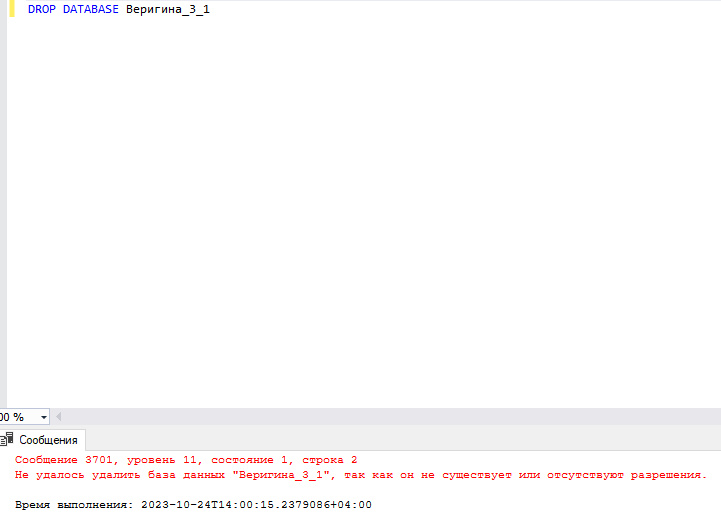


1\_7. Создайте резервную копию этой БД.

Задачи-> Создать резервную копию



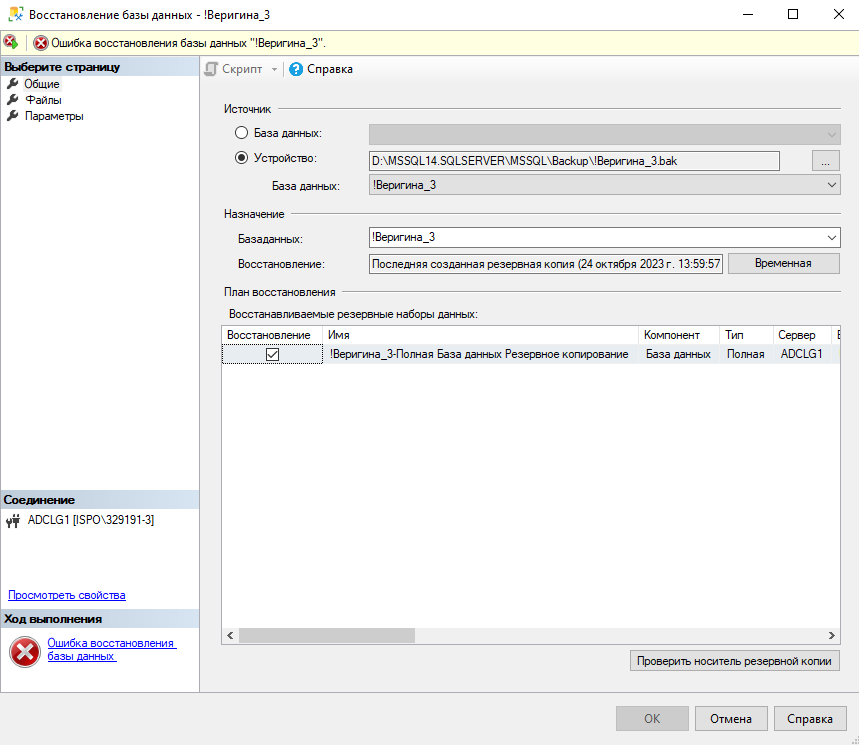
1\_8. Удалите БД с помощью SQL-кода (здесь у нас не прав администратора, поэтому это вызовет ошибку)



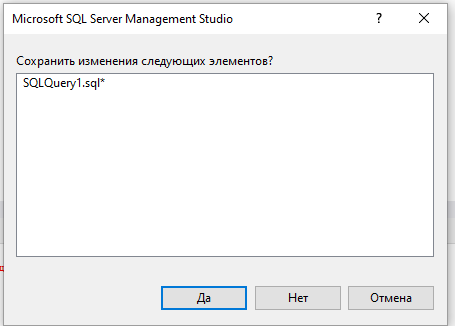
1\_9. Восстановите БД из файла с резервной копией.

База данных -> Восстановление базы данных

(здесь у нас не прав администратора, поэтому это вызовет ошибку, так как удаление тоже невозможно)



1\_10. Сохраните файл сценария.



# **Лабораторная работа №2**

**«Импорт данных. Работа с csv и txt файлами»**

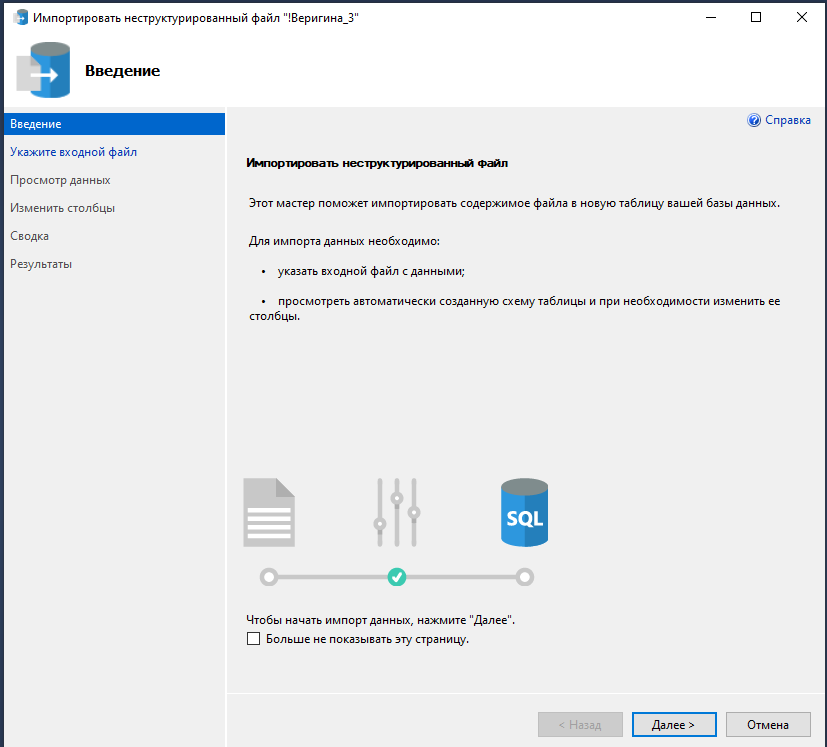
**Цель:** Научиться правильно импортировать данные разных форматов, а также экспортировать данные.

2\_1. Спросите у преподавателя, где находятся файлы для импорта.

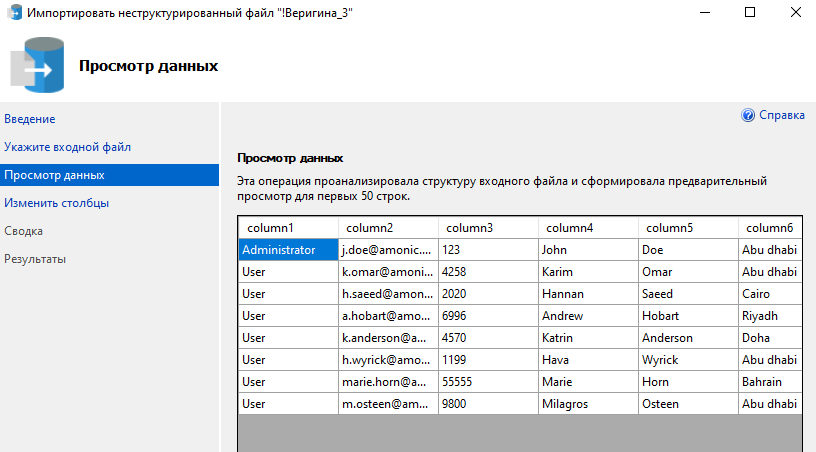
На диске Р в папке ТРЗБД

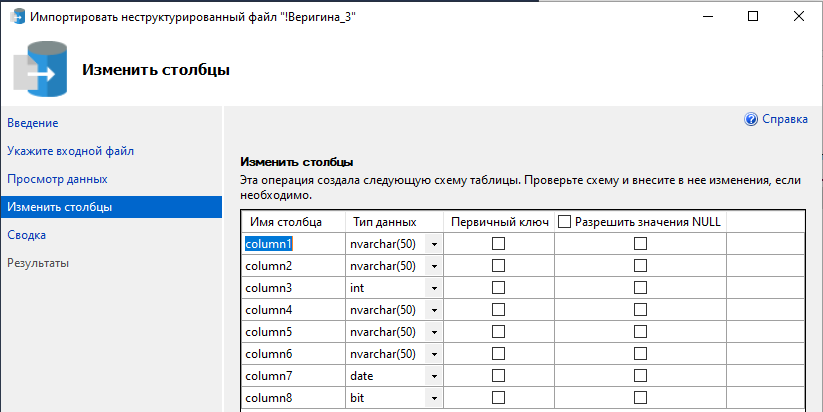
2\_2. Импортируйте каждый и них в вашу БД.

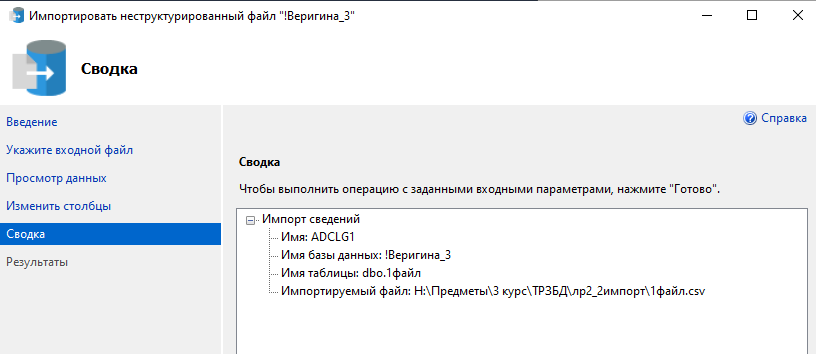
Нажать правой кнопкой мыши на вашу БД в обозревателе объектов, затем выбрать пункт «Задачи», а далее «Импорт неструктурированного файла».

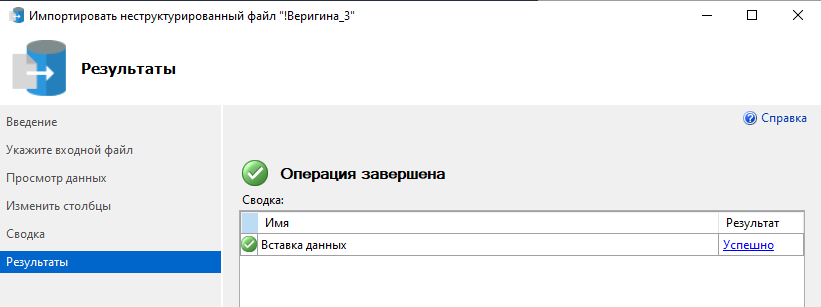




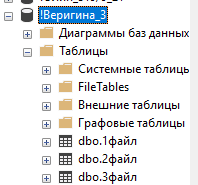






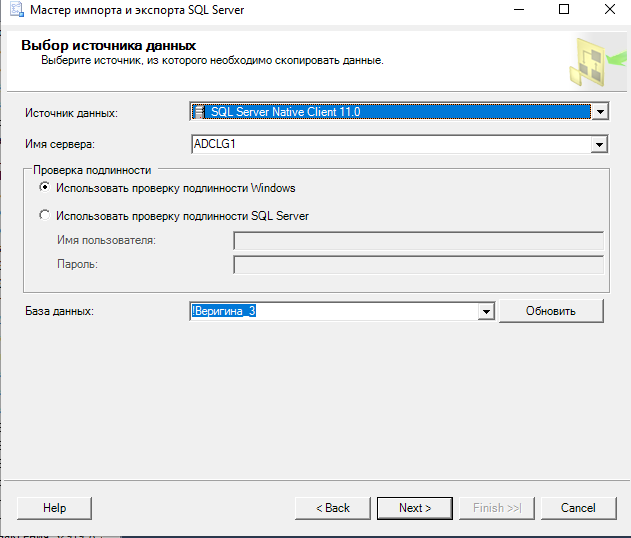


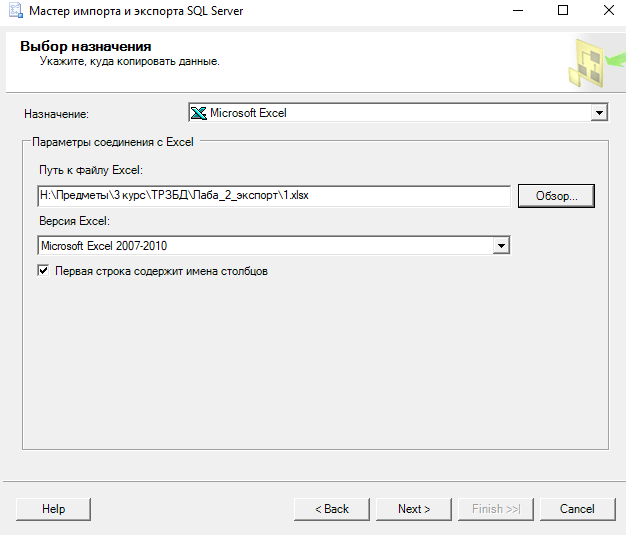
Тоже самое проделала для других двух файлов

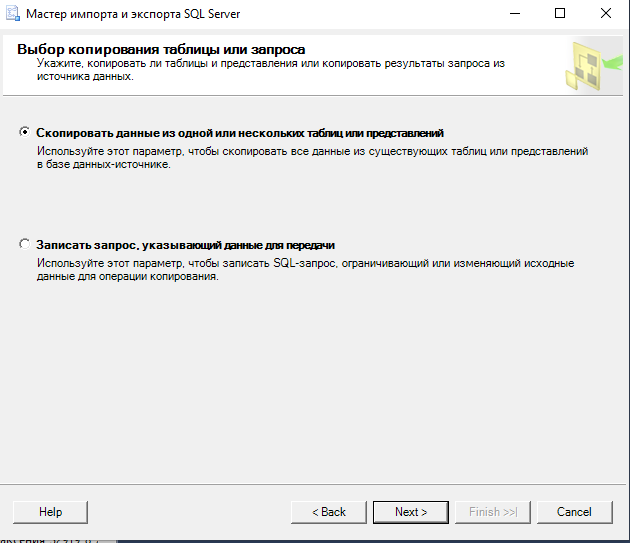


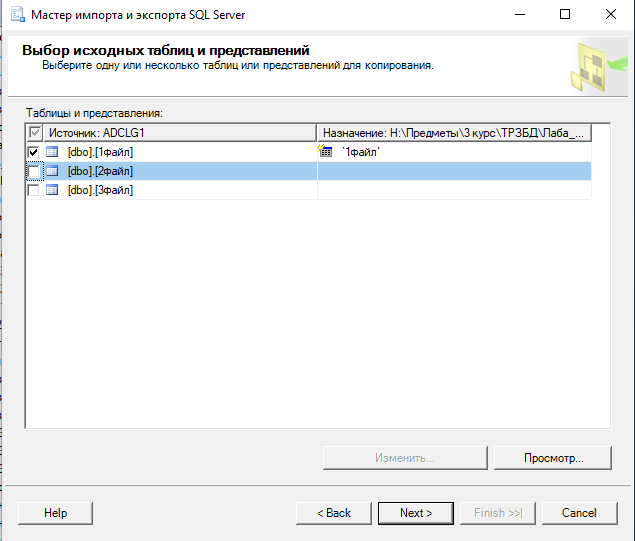
2\_3. Следуя указаниям «мастера импорта и экспорта», экспортируйте любую таблицу в csv или txt файл.

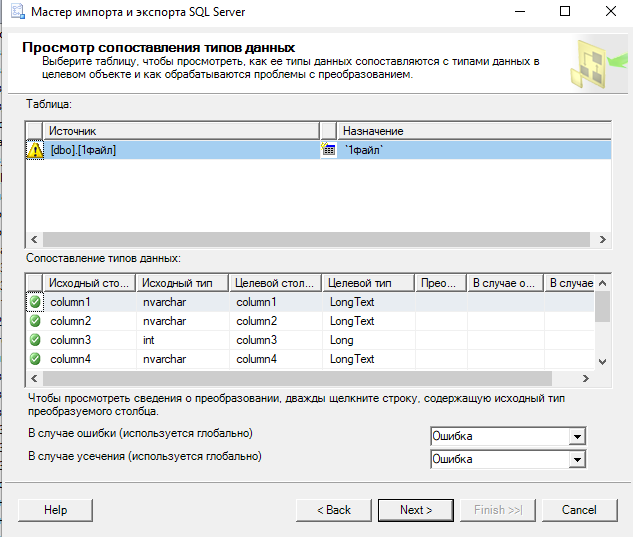
Я экспортировала 1 файл в xlsx

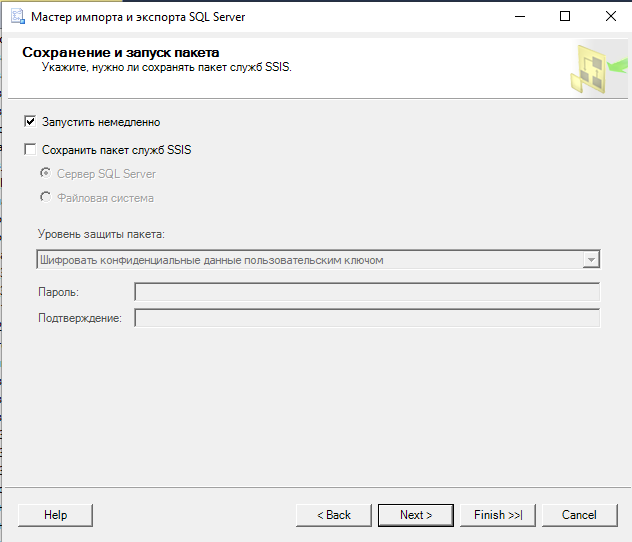


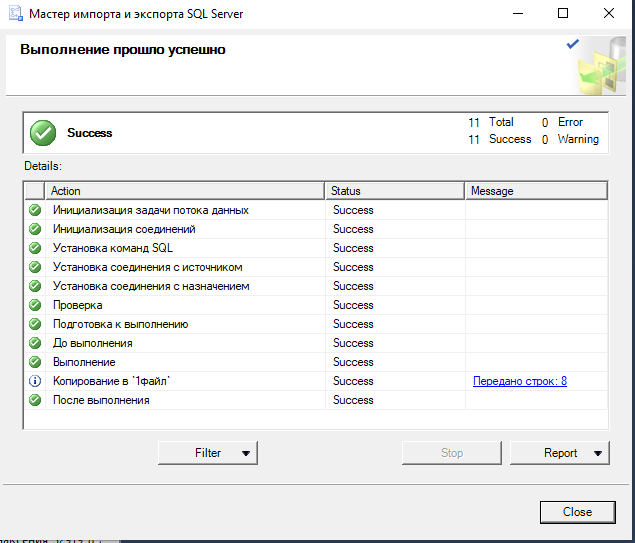




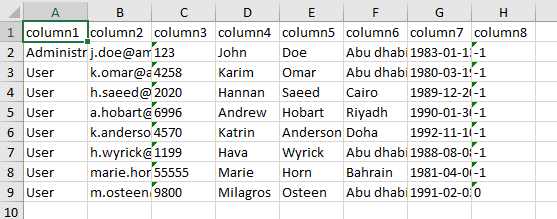






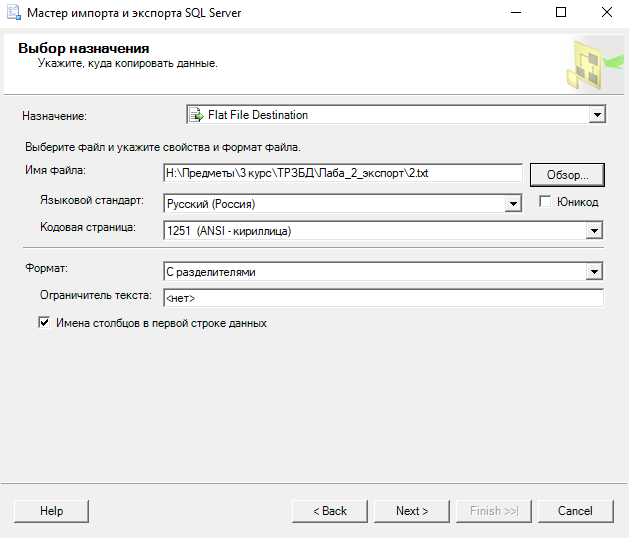


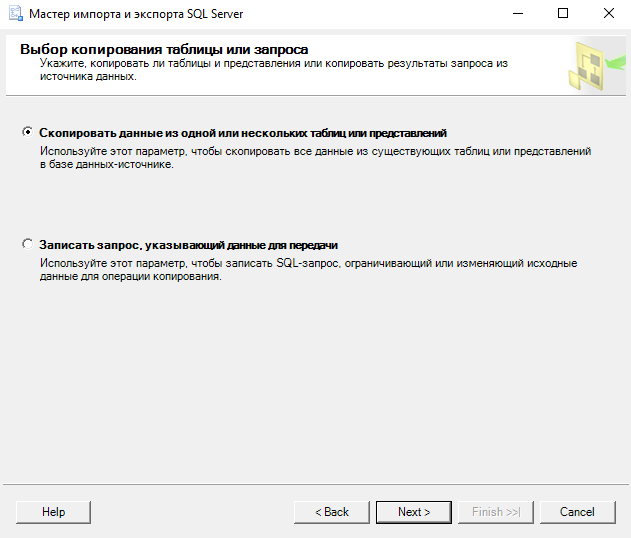
Результат экспорта

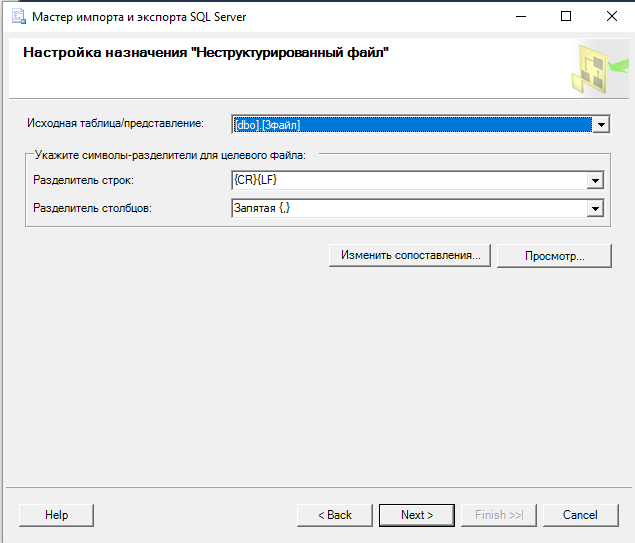


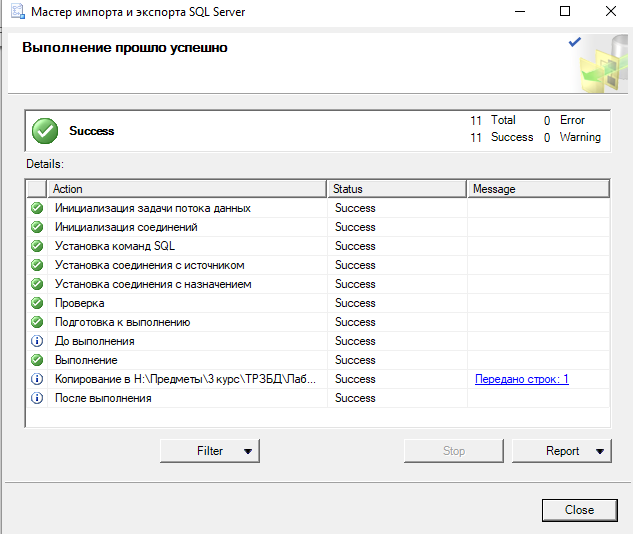
2\_4. Вставьте в отчет выбранные вами свойства и результат экспорта.

Попробовала экспортировать в формат txt файл 3

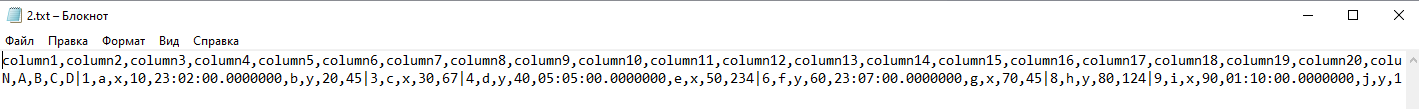




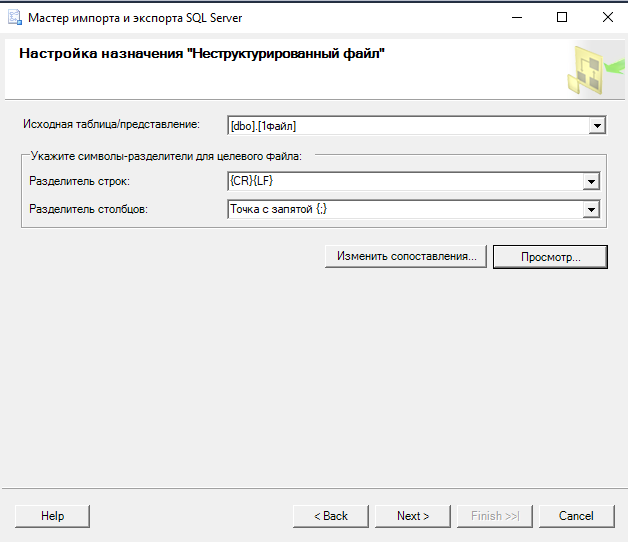




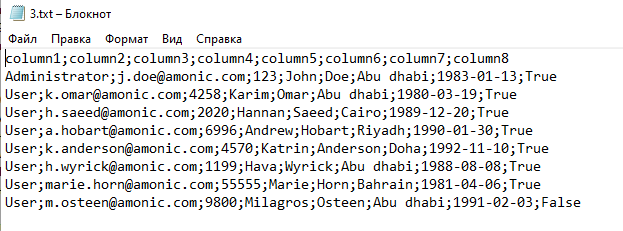
Результат



Попробовала экспортировать 1 файл в txt



Результат



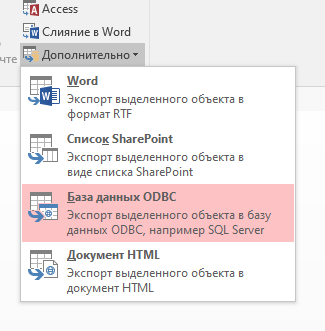
# **Лабораторная работа №3**

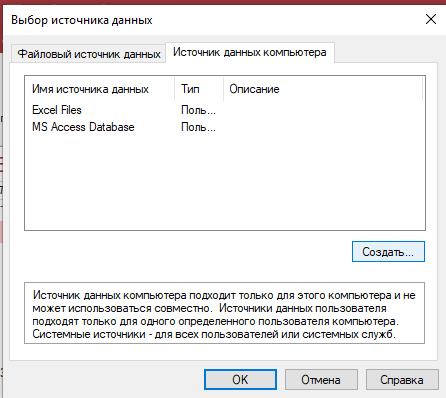
**«Создание и заполнение таблиц»**

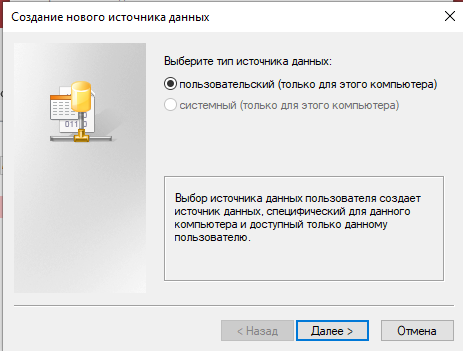
**Цель:** Научиться создавать таблицы (отношения) и заполнять их. Получить навык создания БД с помощью даталогической модели.

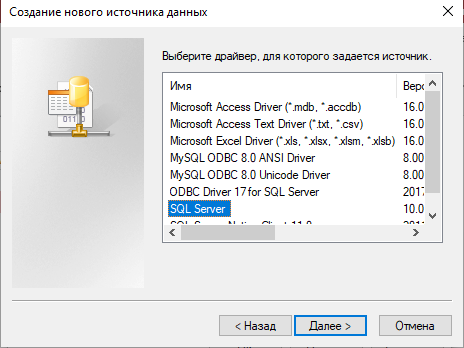
**Задание:** На основании даталогической модели из лабораторной работы создайте в вашей БД таблицы, а также заполните их минимум пятью строками

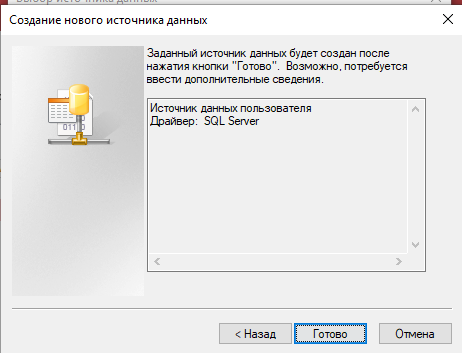
Экспортировала все таблицы:

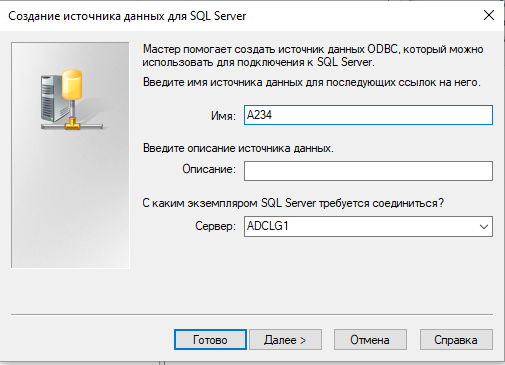


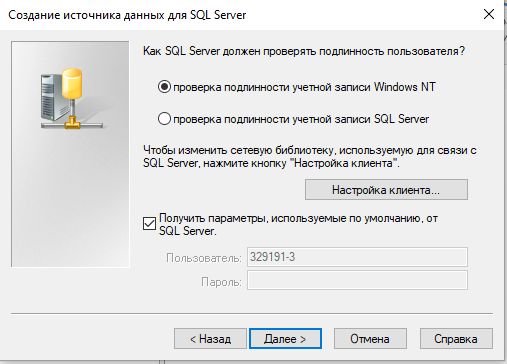


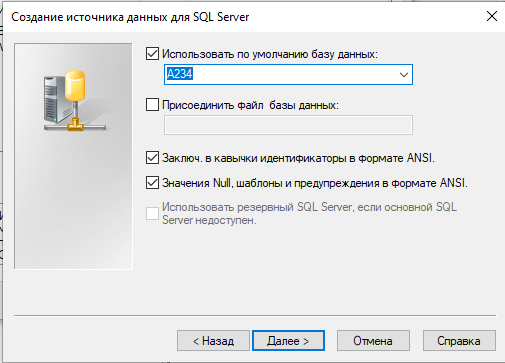


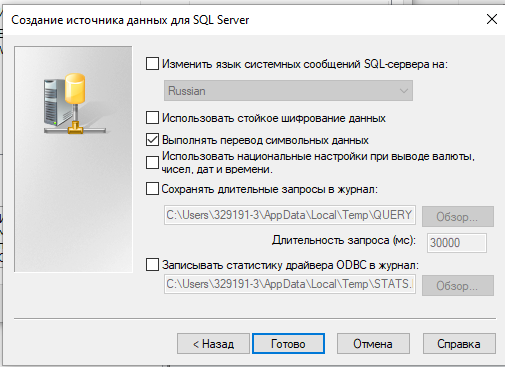


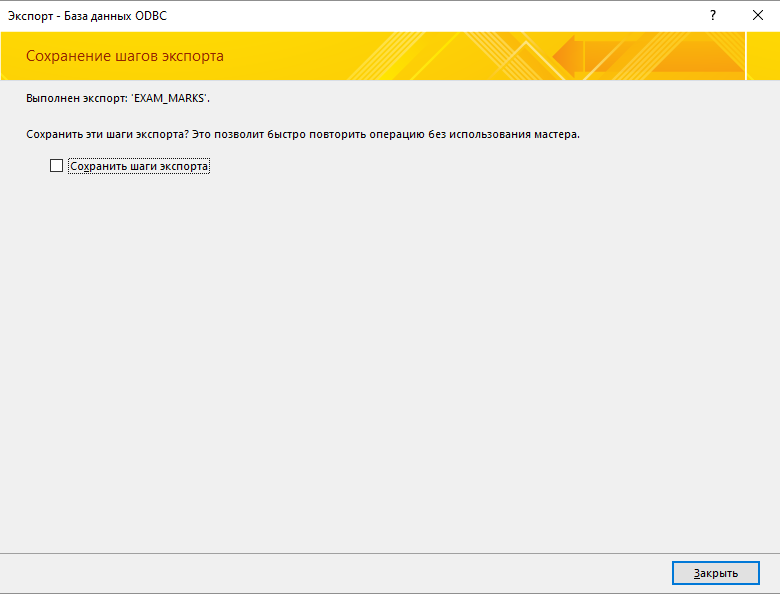


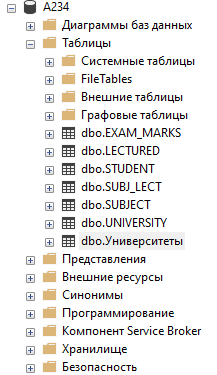












Создала таблицу вручную

CREATE TABLE Университеты(Номер\_Унив int NOT NULL, Название\_Унив nvarchar(5), Рейтинг int, Город nvarchar(15));

Создание записей

INSERT INTO Университеты

VALUES (10, 'ВГУ', 296,'Воронеж'),

(11, 'НГУ', 346, 'Новосибирск'),

(14, 'БГУ', 326, 'Белгород'),

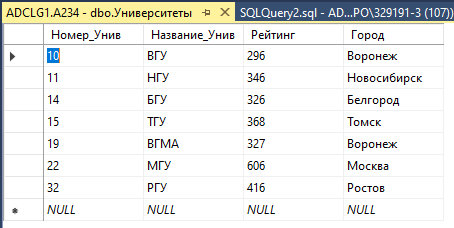
(15, 'ТГУ', 368, 'Томск'),

(19, 'ВГМА', 327, 'Воронеж'),

(22, 'МГУ', 606, 'Москва'),

(32, 'РГУ', 416, 'Ростов');

Итог



# **Лабораторная работа №4**

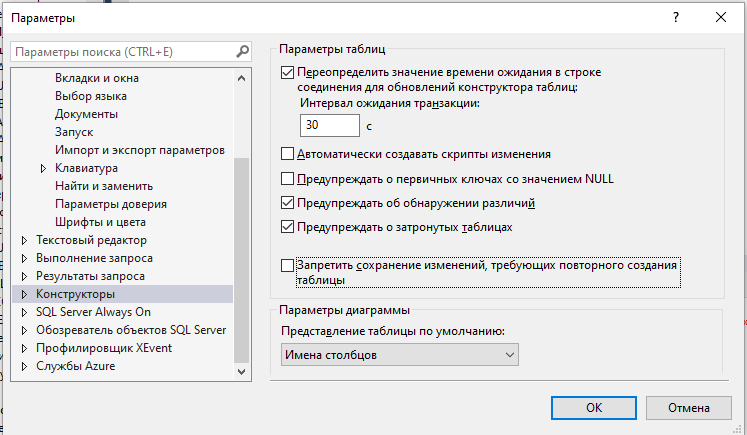
**Разработка ограничений**

**Цель:** Изучить используемые в SQL Server типы ограничений. Изучить SQL-операторы для работы с ограничениями.

**Задание:** На основании даталогической и инфологической моделей, создайте в вашей БД связи между отношениями и другие ограничения.

Создайте таблицу Client, в которой будут храниться данные о клиентах компании. В этом отношении спроектируйте следующие атрибуты и ограничения для них. Id – автоинкримент и первичный ключ, имя, фамилия и отчество имеют только ограничения по длине, возраст - от 14 до 99 лет, номер телефона - начинается с ‘+7’ и содержит еще 10 цифр после этого.

Сняла ограничение на сохранение



Запросы для таблиц:

EXAM\_MARKS:

Первичный ключ

ALTER TABLE EXAM\_MARKS

ADD CONSTRAINT C1

PRIMARY KEY (EXAM\_ID);

Проверила выполнение:



Вторичный ключ

ALTER TABLE EXAM\_MARKS

ADD CONSTRAINT C7

FOREIGN KEY (STUDENT\_ID)

REFERENCES STUDENT (STUDENT\_ID);

ALTER TABLE EXAM\_MARKS

ADD CONSTRAINT C8

FOREIGN KEY (SUBJ\_ID)

REFERENCES SUBJECT (SUDJ\_ID);

Уникальность

ALTER TABLE EXAM\_MARKS

ADD CONSTRAINT C13

UNIQUE ([EXAM\_ID]);

LECTURED:

Первичный ключ

ALTER TABLE LECTURED

ADD CONSTRAINT C2

PRIMARY KEY (LECTURED\_ID);

Вторичный ключ

ALTER TABLE LECTURED

ADD CONSTRAINT C10

FOREIGN KEY (UNIV\_ID)

REFERENCES UNIVERSITY (UNIV\_ID);

Уникальность

ALTER TABLE [LECTURED]

ADD CONSTRAINT C14

UNIQUE ([LECTURED\_ID]);

STUDENT:

Первичный ключ

ALTER TABLE STUDENT

ADD CONSTRAINT C3

PRIMARY KEY (STUDENT\_ID);

Вторичный ключ

ALTER TABLE STUDENT

ADD CONSTRAINT C9

FOREIGN KEY (UNIV\_ID)

REFERENCES UNIVERSITY (UNIV\_ID);

Уникальность

ALTER TABLE [STUDENT]

ADD CONSTRAINT C15

UNIQUE ([STUDENT\_ID]);

SUBJ\_LECT:

Первичный ключ

ALTER TABLE SUBJ\_LECT

ADD CONSTRAINT C4

PRIMARY KEY (SUBJ\_ID, LECTURED\_ID);

Вторичный ключ

ALTER TABLE SUBJ\_LECT

ADD CONSTRAINT C11

FOREIGN KEY (LECTURED\_ID)

REFERENCES LECTURED (LECTURED\_ID);

ALTER TABLE SUBJ\_LECT

ADD CONSTRAINT C12

FOREIGN KEY (SUBJ\_ID)

REFERENCES SUBJECT ([SUDJ\_ID]);

Уникальность

ALTER TABLE [SUBJ\_LECT]

ADD CONSTRAINT C16

UNIQUE ([LECTURED\_ID], [SUBJ\_ID]);

SUBJECT:

Первичный ключ

ALTER TABLE SUBJECT

ADD CONSTRAINT C5

PRIMARY KEY (SUDJ\_ID);

Уникальность

ALTER TABLE [SUBJECT]

ADD CONSTRAINT C17

UNIQUE ([SUDJ\_ID]);

UNIVERSITY:

Первичный ключ

ALTER TABLE UNIVERSITY

ADD CONSTRAINT C6

PRIMARY KEY (UNIV\_ID);

Уникальность

ALTER TABLE [UNIVERSITY]

ADD CONSTRAINT C18

UNIQUE ([UNIV\_ID]);

Создание таблицы клиенты

CREATE TABLE Client( id int NOT NULL primary key IDENTITY (1,1),

Имя nvarchar(10),

Фамилия nvarchar(30),

Отчество nvarchar(20),

Возраст int,

[номер телефона] char(16) check ([номер телефона]

like '+7([0-9][0-9][0-9])[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]'));

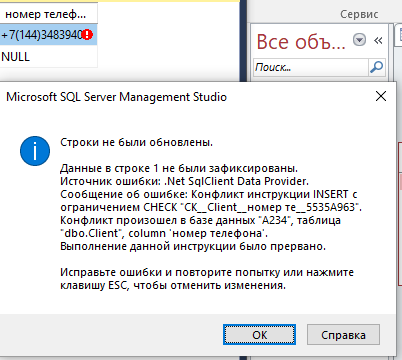
Ограничение:

ALTER TABLE Client

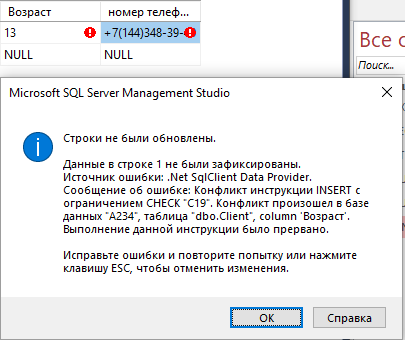
ADD CONSTRAINT C19

CHECK (Возраст>=14 AND Возраст <=99)

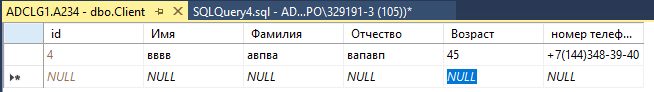
Ошибка при заполнении номера телефона



Ошибка при заполнении возраста вне ограничения



Удачное заполнение



# **Лабораторная работа №5**

**«Работа с внешними ключами»**

**Цель**: Изучить синтаксис Foreign key и понятие ссылочной целостности. Изучить разницу между разными параметрами ключевых слов on delete on update.

**Задание:**

5\_1. Создать четыре отношения и заполнить их любым удобным способом.

5\_2. Должны получиться следующие отношения:

Таблица поставщиков (Shippers)

|  |  |  |  |
| --- | --- | --- | --- |
| Номер поставщика | Фамилия | Рейтинг | Город |
| 1 | Смит | 20 | Лондон |
| 2 | Джонс | 10 | Париж |
| 3 | Блейк | 30 | Париж |
| 4 | Кларк | 20 | Лондон |
| 5 | Адамс | 30 | Афины |

Создала запросом таблицу и заполнила ее

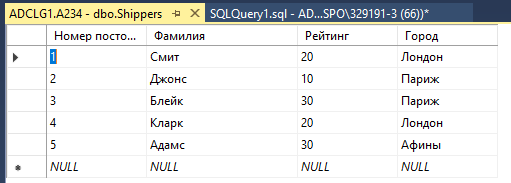


Таблица деталей (Details)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер детали | Название | Цвет | Вес | Город |
| 1 | Гайка | Красный | 12 | Лондон |
| 2 | Болт | Зеленый | 17 | Париж |
| 3 | Винт | Голубой | 17 | Рим |
| 4 | Винт | Красный | 14 | Лондон |
| 5 | Кулачок | Голубой | 12 | Париж |
| 6 | Блюм | Красный | 19 | Лондон |

Создала запросом таблицу и заполнила ее

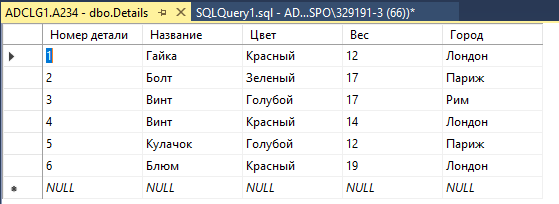


Таблица изделий (Products)

|  |  |  |
| --- | --- | --- |
| Номер изделия | Название | Город |
| 1 | Жесткий диск | Париж |
| 2 | Перфоратор | Рим |
| 3 | Считыватель | Афины |
| 4 | Принтер | Афины |
| 5 | Флоппи-диск | Лондон |
| 6 | Терминал | Осло |
| 7 | Лента | Лондон |

Создала запросом таблицу и заполнила ее

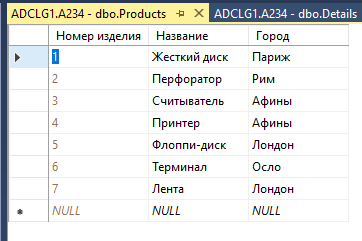
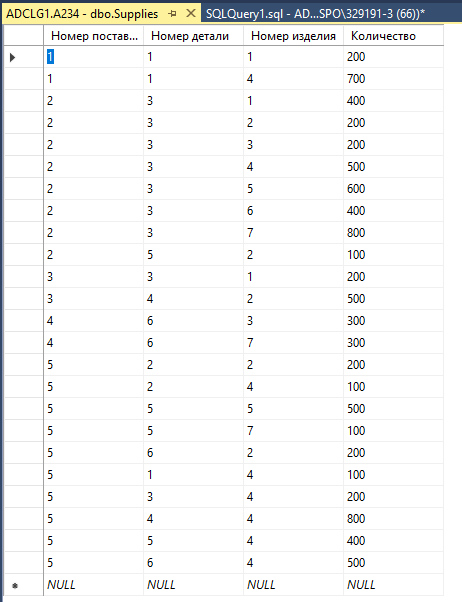


Таблица поставок (Supplies)

|  |  |  |  |
| --- | --- | --- | --- |
| Номер поставщика | Номер детали | Номер изделия | Количество |
| 1 | 1 | 1 | 200 |
| 1 | 1 | 4 | 700 |
| 2 | 3 | 1 | 400 |
| 2 | 3 | 2 | 200 |
| 2 | 3 | 3 | 200 |
| 2 | 3 | 4 | 500 |
| 2 | 3 | 5 | 600 |
| 2 | 3 | 6 | 400 |
| 2 | 3 | 7 | 800 |
| 2 | 5 | 2 | 100 |
| 3 | 3 | 1 | 200 |
| 3 | 4 | 2 | 500 |
| 4 | 6 | 3 | 300 |
| 4 | 6 | 7 | 300 |
| 5 | 2 | 2 | 200 |
| 5 | 2 | 4 | 100 |
| 5 | 5 | 5 | 500 |
| 5 | 5 | 7 | 100 |
| 5 | 6 | 2 | 200 |
| 5 | 1 | 4 | 100 |
| 5 | 3 | 4 | 200 |
| 5 | 4 | 4 | 800 |
| 5 | 5 | 4 | 400 |
| 5 | 6 | 4 | 500 |

Создала запросом таблицу и заполнила ее



5\_3. Установить внешний ключ в последней таблице на соответствующие атрибуты.

Для первой таблицы создать каскадное удаление и обновление, для второй – установку значений null в удаленных строках, а для третьей установить отсутствие действия.

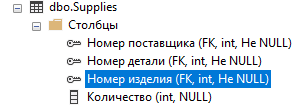
Внешние ключи созданы

ALTER TABLE Supplies

ADD CONSTRAINT C22

FOREIGN KEY ([Номер изделия])

REFERENCES Products ([Номер изделия]);



Для первой каскад

ALTER TABLE Supplies ADD CONSTRAINT С23

FOREIGN KEY ([Номер поставщика]) REFERENCES Shippers ([Номер поставщика])

ON DELETE CASCADE

ON UPDATE CASCADE

Для второй нул

ALTER TABLE Supplies ADD CONSTRAINT С24

FOREIGN KEY ([Номер детали]) REFERENCES Details ([Номер детали])

ON DELETE SET NULL

Для третьей без действий

ALTER TABLE Supplies ADD CONSTRAINT С25

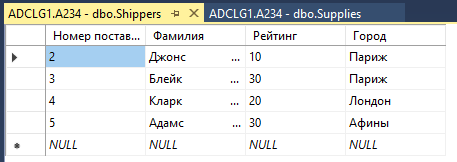
FOREIGN KEY ([Номер изделия]) REFERENCES Products ([Номер изделия])

ON DELETE NO ACTION

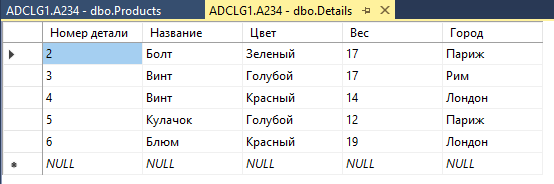
ON UPDATE NO ACTION

5\_4. Удалить из таблиц Shippers, Details, Products строки с первым номером, вывести таблицу Supplies на экран. Проанализировать изменения.

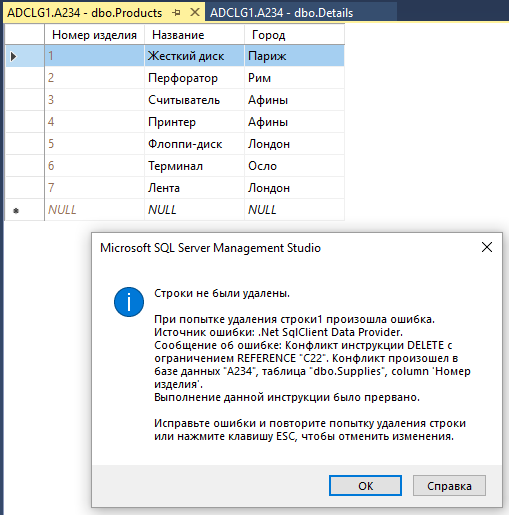
Удалила первую строку



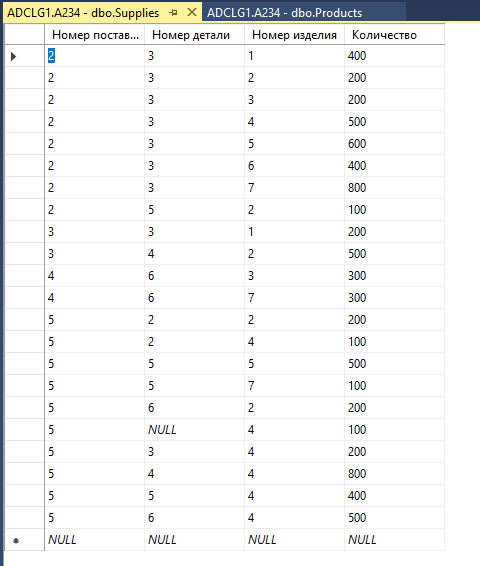
Для второй



Для третьей ошибка из-за установленного параметра



Общая таблица удалилась строка певой таблицы, второй стала нул, а третья не изменилась



5\_5. Сделать вывод, который будет содержать ответ на вопрос: «Как вам кажется, какой вариант параметров on delete on update целесообразнее было бы использовать для всех таблиц в этой работе?»

Вывод: Как показала практика использование каскадного удаления/обновления данных лучше для сохранения целостности и актуальности информации в БД.

**Контрольные вопросы:**

**1.** Какие типы ограничений вы знаете? Какие у них назначения?

NULL - действует на уровне столбца и пользовательского типа данных и либо разрешает (NULL), либо запрещает (NOT NULL) хранение значений NULL.

CHECK - действует на уровне столбца и ограничивает диапазон значений, которые могут быть сохранены в столбце, путем проверки логического условия для вводимых данных. При вводе или изменении данных вводимое значение подставляется в условие. Если полученный результат TRUE, то изменения данных принимаются, иначе - отвергаются и генерируется сообщение об ошибке. Для одного столбца можно задать несколько ограничений типа CHECK.

UNIQUE - действует на уровне столбца и гарантирует уникальность вводимых значений. В отличии от ограничения PRIMARY KEY, это ограничение допускает хранение значений NULL.

PRIMARY KEY - действует на уровне столбца или таблицы и гарантирует уникальность в пределах таблицы первичного ключа, состоящего из одного или нескольких столбцов. Ни для одного из столбцов ключа не должно быть установлено свойство NULL. Когда используется один столбец, то для него необходимо также задать и свойство UNIQUE. В таблице создается только один первичный ключ.

FOREIGN KEY - действует на уровне таблицы и связывается с одним из кандидатов на первичный ключ в другой таблице. Таблица, в которой определен внешний ключ с помощью этого ограничения, называется зависимой, а таблица с кандидатом на первичный ключ - главной. В зависимую таблицу нельзя вставить строку, если внешний ключ не имеет соответствующего значения в главной таблице. Из главной таблицы нельзя удалить строку, если с ней связана хотя бы одна строка в зависимой таблице.

**2.** Зачем нужны ограничения?

Это один из элементов, с помощью которого мы можем обеспечить целостность для нашей БД. Это механизм, обеспечивающий автоматический контроль соответствия данных установленным условиям, или ограничениям целостности.

**3**. Можно ли обойтись без ограничений в БД?

Можно, однако это приведет к потере данных, нарушении целостности, некорректности.

**4**. Существуют ли какие-либо ограничения в таблице по умолчанию?

По умолчанию никакие ограничения не установлены

**5.** Какие условия необходимо соблюсти, чтобы связать две таблицы связью

Необходимо, чтобы эти таблицы имели первичные ключи

**6.** Какие типы ключей вы знаете?

Первичный и внешний

**7.** При связи двух таблиц, накладываются ли какие-либо ограничения на дочернюю или родительскую таблицу?

С помощью конструкций ON DELETE и ON UPDATE и их параметров можно наложить ограничения целостности при связи двух таблиц.

**8.** Зачем нужны ключевые слова on update on delete?

Они нужны для того, чтобы определять действия в связанных таблицах при удалении или обновлении записей.

**9.** Какие параметры on update on delete существуют?

CASCADE — При удалении/обновлении записей в таблице-предке, будут также обновлены/удалены записи из таблицы-потомка с существующим первичным ключом.

SET NULL — При удалении/обновлении записей в таблице-предке, записи из таблицы потомка с существующим первичным ключом будут обновлены на NULL.

NO ACTION — При удалении/обновлении записей в таблице-предке, записи из таблицы-потомка с существующим первичным ключом изменены не будут.

**10.** Какой синтаксис у ограничения типа CHECK?

CHECK (Название поля, условие)

# **Лабораторная работа №6**

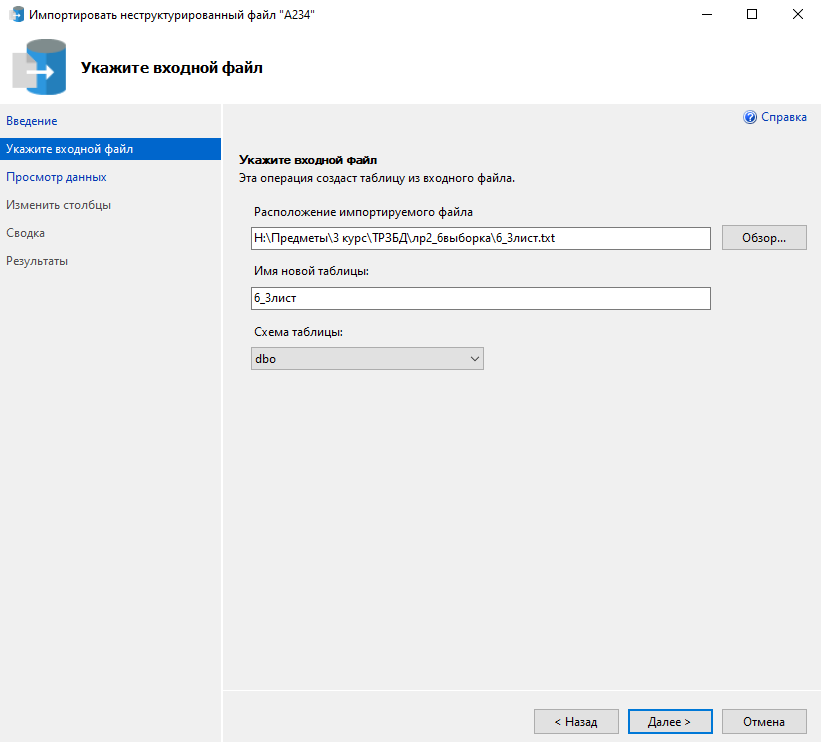
**«Выборка данных. Выбор строк по условию»**

**Цель:** Изучить запросы на выборку к одной таблице, научиться использовать запросы, сочетание нескольких функций, операторов, кодовых слов.

**Задание:**

6\_1. Импортируйте данные из файла, в соответствии с вариантом.

Импортировала лист своего (3 варианта)



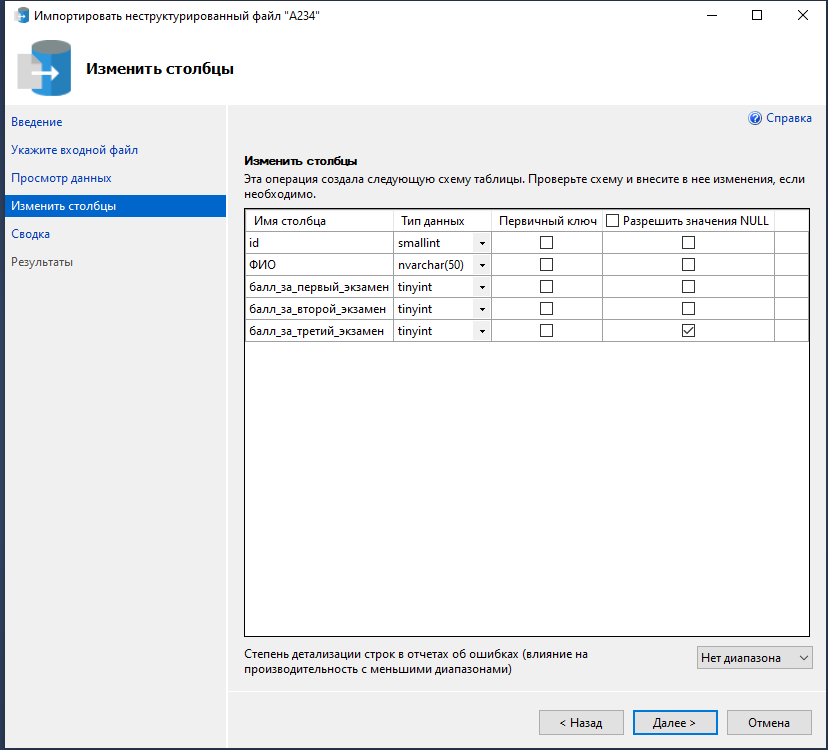
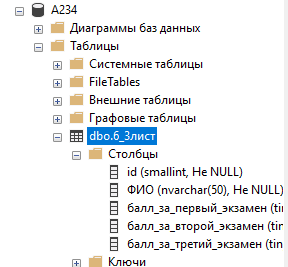


Таблица успешно импортирована



6\_2. Для своего отношения создайте следующие запросы:

a) три запроса с использованием оператора сравнения;

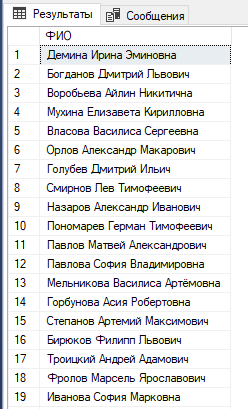
Запрос 1:

select ФИО

from tab

where балл\_за\_второй\_экзамен<балл\_за\_первый\_экзамен;

Результат 1:



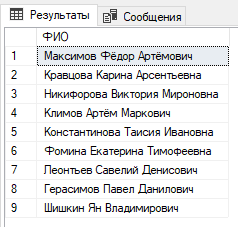
Запрос 2:

select ФИО

from tab

where балл\_за\_третий\_экзамен>99;

Результат 2:



Запрос 3:

select ФИО

from tab

where балл\_за\_первый\_экзамен<3;

Результат 3:



b) три запроса с использованием логических операторов and, or, not;

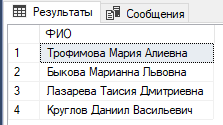
Запрос 1:

select ФИО

from tab

where балл\_за\_первый\_экзамен<3 and балл\_за\_второй\_экзамен>60;

Результат 1:



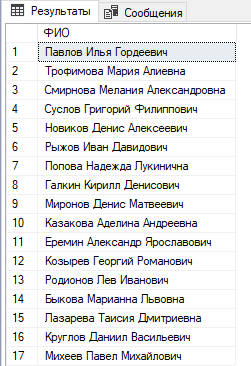
Запрос 2:

select ФИО

from tab

where балл\_за\_первый\_экзамен<3 or балл\_за\_третий\_экзамен=90;

Результат 2:



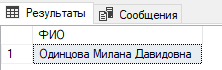
Запрос 3:

select ФИО

from tab

where not(id!=290);

Результат 3:



c) один запрос с комбинацией логических операторов;

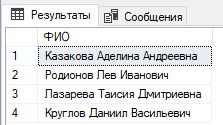
Запрос:

select ФИО

from tab

where балл\_за\_первый\_экзамен<2 and not(id<290);

Результат:



d) два запроса на использование выражений над столбцами;

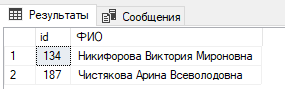
Запрос 1:

select id, ФИО

from tab

where балл\_за\_первый\_экзамен + балл\_за\_второй\_экзамен > 190;

Результат 1:



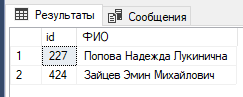
Запрос 2:

select id, ФИО

from tab

where балл\_за\_первый\_экзамен + балл\_за\_второй\_экзамен < 10;

Результат 2:



e) три запроса с использованием агрегатных функций;

Запрос 1:

select AVG(балл\_за\_первый\_экзамен) as ср\_балл\_за\_первый\_экзамен

from tab;

Результат 1:



Запрос 2:

select Sum(балл\_за\_второй\_экзамен) as суммарный\_балл\_за\_второй\_экзамен

from tab;

Результат 2:



Запрос 3:

select Count(id) as количество\_учеников

from tab;

Результат 3:



f) один запрос с проверкой на принадлежность множеству;

Запрос:

select ФИО

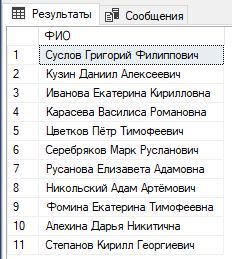
from tab

where балл\_за\_первый\_экзамен in (

select AVG(балл\_за\_первый\_экзамен) as ср\_балл\_за\_первый\_экзамен

from tab);

Результат:



g) два запроса с проверкой на принадлежность диапазону значений;

Запрос 1:

select ФИО

from tab

where балл\_за\_первый\_экзамен between (

select AVG(балл\_за\_первый\_экзамен)

from tab) and (

select Max(балл\_за\_первый\_экзамен)

from tab);

Результат 1:



Запрос 2:

select ФИО

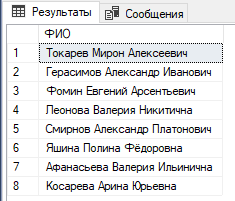
from tab

where id between (

select AVG(id)

from tab) and 307 ;

Результат 2:



h) два запроса с проверкой на соответствие шаблону (разные);

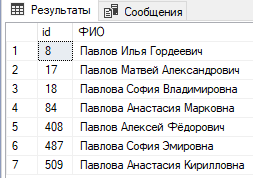
Запрос 1:

select id, ФИО

from tab

where ФИО like('П\_в%');

Результат 1:



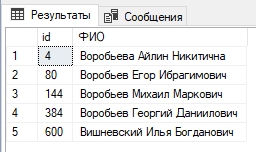
Запрос 2:

select id, ФИО

from tab

where ФИО like('В%б%');

Результат 2:



i) один запрос с проверкой на неопределенное значение.

Запрос:

select ФИО

from tab

where балл\_за\_третий\_экзамен is null ;

Результат:



# **Лабораторная работа №7**

**«Выборка данных. Многотабличные запросы, запросы на соединения»**

**Цель:** Изучить многотабличные запросы на выборку, а также получить навыки работы с оператором SELECT.

**Задания:**

Используйте БД, которая была создана в лабораторной работе №3.

Самостоятельно создать следующие запросы:

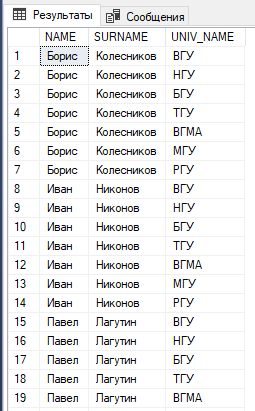
7\_1. один запрос с использованием декартового произведения двух таблиц;

Запрос:

select L.NAME, L.SURNAME,U.UNIV\_NAME

from LECTURED L, UNIVERSITY U;

Результат:



7\_2. два запроса с использованием соединения двух таблиц по равенству;

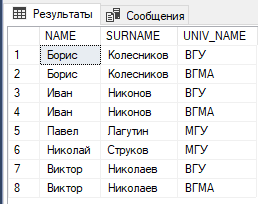
Запрос 1:

select L.NAME, L.SURNAME,U.UNIV\_NAME

from LECTURED L, UNIVERSITY U

where L.CITY=U.CITY;

Результат 1:



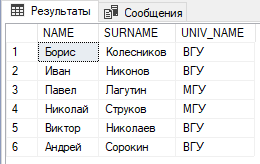
Запрос 2:

select L.NAME, L.SURNAME,U.UNIV\_NAME

from LECTURED L, UNIVERSITY U

where L.UNIV\_ID=U.UNIV\_ID;

Результат 2:



7\_3. один запрос с использованием соединения двух таблиц по равенству и условием отбора;

Запрос:

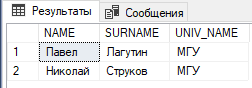
select L.NAME, L.SURNAME,U.UNIV\_NAME

from LECTURED L, UNIVERSITY U

where L.UNIV\_ID=U.UNIV\_ID

And L.CITY='Москва';

Результат:



7\_4. один запрос с использованием соединения по трем таблицам с условием;

Запрос:

select L.NAME, L.SURNAME,U.UNIV\_NAME, S.NAME, S.SURNAME, S.KURS

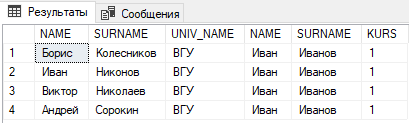
from LECTURED L, UNIVERSITY U, STUDENT S

where L.UNIV\_ID=U.UNIV\_ID

And L.UNIV\_ID=S.UNIV\_ID

And S.CITY='Орел';

Результат:



7\_5. создать копии ранее созданных запросов на соединение по равенству на запросы с использованием внешнего полного соединения таблиц (JOIN).

Запрос:

select L.NAME, L.SURNAME,U.UNIV\_NAME

from LECTURED L FULL OUTER JOIN UNIVERSITY U on L.UNIV\_ID=U.UNIV\_ID;

select L.NAME, L.SURNAME,U.UNIV\_NAME

from LECTURED L FULL OUTER JOIN UNIVERSITY U on L.CITY=U.CITY;

select L.NAME, L.SURNAME,U.UNIV\_NAME

from LECTURED L FULL OUTER JOIN UNIVERSITY U on L.CITY=U.CITY

where L.CITY='Москва';

select L.NAME, L.SURNAME,U.UNIV\_NAME

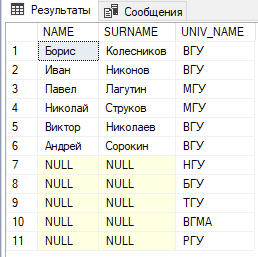
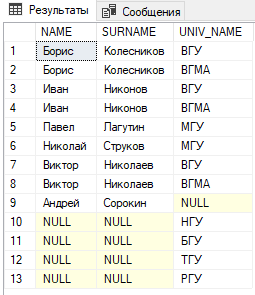
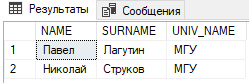
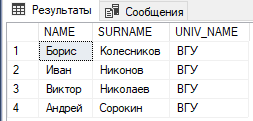
from UNIVERSITY U

FULL OUTER JOIN LECTURED L on L.UNIV\_ID=U.UNIV\_ID

FULL OUTER JOIN STUDENT S on S.UNIV\_ID=U.UNIV\_ID

where S.CITY='Орел';

Результат:

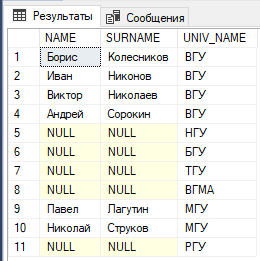
7\_6. один запрос с использованием левого внешнего соединения;

Запрос:

select L.NAME, L.SURNAME,U.UNIV\_NAME

from UNIVERSITY U LEFT JOIN LECTURED L on L.UNIV\_ID=U.UNIV\_ID;

Результат:



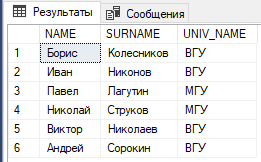
7\_7. один запрос на использование правого внешнего соединения;

Запрос:

select L.NAME, L.SURNAME,U.UNIV\_NAME

from UNIVERSITY U RIGHT JOIN LECTURED L on L.UNIV\_ID=U.UNIV\_ID;

Результат:



7\_8. один запрос с использованием внутреннего соединения.

Запрос:

select \*

from UNIVERSITY U INNER JOIN STUDENT S on S.UNIV\_ID=U.UNIV\_ID;

Результат:



7\_9. один запрос на проверку какого-либо условия через самосоединение.

Запрос:

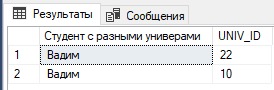
select S.NAME As 'Студент с разными универами', S.UNIV\_ID

from STUDENT S, STUDENT St

where S.Name = St.Name

And S.UNIV\_ID<> St.UNIV\_ID;

Результат:



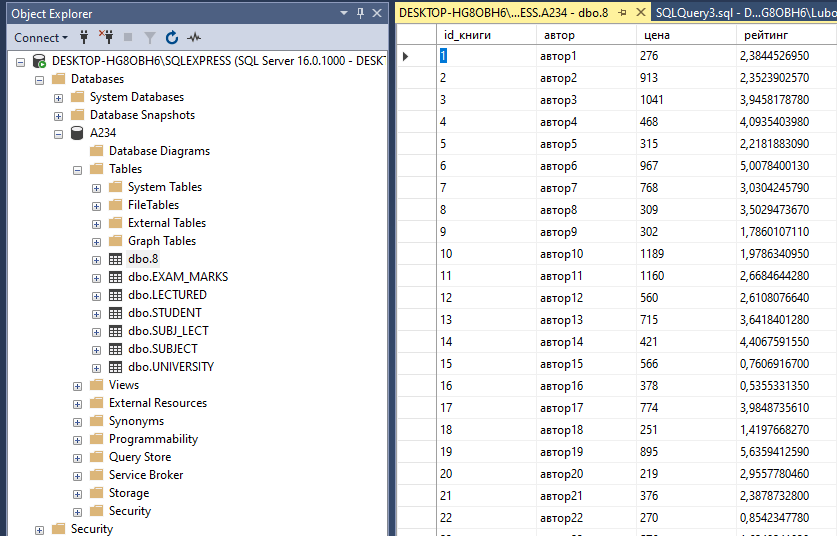
# **Лабораторная работа №8**

**«Выборка данных. Группировка и сортировка данных»**

**Цель:** Изучить используемый в реляционных СУБД оператор извлечения данных из таблиц SELECT и выполнение группировки и сортировки данных. Изучить синтаксис языка модификации данных. Научится использовать встроенные функции в запросах.

**Задания:**

Импортировать данные из ресурса, указанного преподавателем, в соответствии с вариантом.



Для 3 варианта:

8\_1. Запрос, который покажет, сколько книг продается у того или иного автора, сделайте сортировку по убыванию.

Запрос:

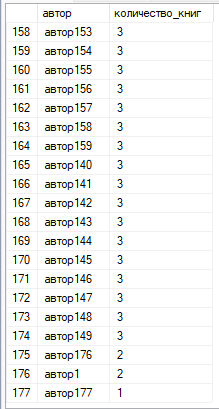
SELECT автор, COUNT(\*) AS количество\_книг

FROM Книги

GROUP BY автор

ORDER BY количество\_книг DESC;

Результат:



8\_2. Запрос, который покажет, какие авторы продают более пяти книг.

Запрос:

select автор

from (SELECT автор, COUNT(\*) AS количество\_книг

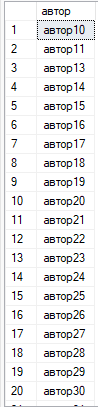
FROM Книги

group by автор) A

where количество\_книг>5

order by количество\_книг desc;

Результат:



8\_3. Запрос, который покажет сколько книг продает Автор1, Автор2, Автор3. Создайте два запроса один с помощью WHERE, другой используя HAVING.

Запрос 1:

SELECT автор, COUNT(\*) AS количество\_продаж

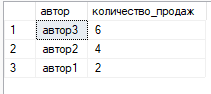
FROM Книги

WHERE автор IN ('Автор1', 'Автор2', 'Автор3')

GROUP BY автор

ORDER BY количество\_продаж DESC;

Результат 1:



Запрос 2:

SELECT автор, COUNT(\*) AS количество\_продаж

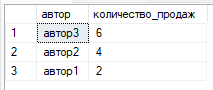
FROM Книги

GROUP BY автор

HAVING автор IN ('Автор1', 'Автор2', 'Автор3')

ORDER BY количество\_продаж DESC;

Результат 2:



8\_4. Запрос, который покажет, сколько книг продает тот или иной автор, считая книги с рейтингом более 3.

Запрос:

SELECT автор, COUNT(\*) AS количество\_продаж

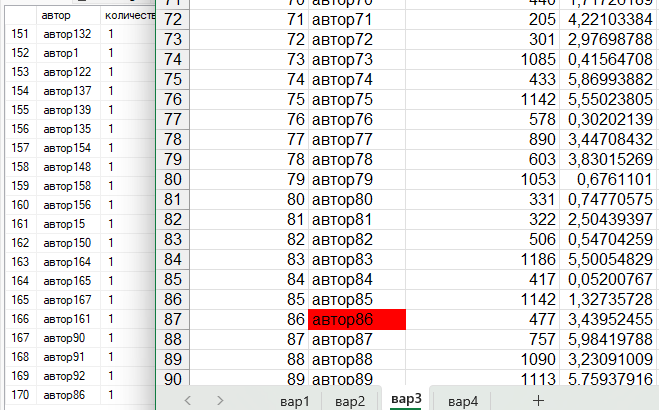
FROM Книги

WHERE рейтинг > 3

GROUP BY автор

ORDER BY количество\_продаж DESC;

Результат:



8\_5. Запрос, который покажет сколько книг с рейтингом менее 3 и ценой более 500 продается в магазине.

Запрос:

SELECT автор, COUNT(\*) AS количество\_продаж

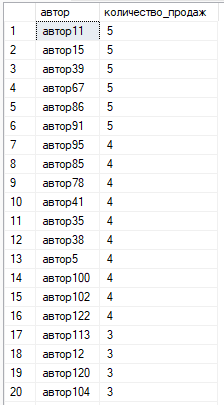
FROM Книги

WHERE рейтинг < 3 and цена>500

GROUP BY автор

ORDER BY количество\_продаж DESC;

Результат:



**Контрольные вопросы:**

1. Что такое агрегатная функция?

Агрегатные функции (агрегации) — это функции, которые вычисляются от группы значений (строк) и объединяют их в одно результирующее значение.

2. Зачем нужно самосоединение?

Самосоединение можно использовать для проверки корректности данных. Например, мы точно знаем, что в нашем вузе нет однофамильцев, занимающих разные должности. С помощью самосоединения таблицы преподавателей мы можем убедиться, что их нет и в базе данных

3. Какие типы соединений вы знаете?

Всего существует пять объединений. Они такие: 1. соединение или внутреннее соединение 2. внешнее соединение 2.1 левое внешнее соединение или левое соединение 2.2 правое внешнее соединение или правое соединение 2.3 полное внешнее соединение или полное соединение 3. естественное соединение 4. перекрестное соединение 5. самостоятельное соединение.

4. Что такое группировка? Приведите пример использования.

Группировка. Для того, чтобы разделить данные в таблице на определенные группы, используется оператор GROUP BY. Пример, я проделала в работе, везде, где присутствуют агрегирующие функции, а также другие столбцы, другие данные необходимо группировать.

5. Как вывести всю строку, содержащую максимальное значение по тому или иному атрибуту?

Для того, чтобы вывести всю строку, содержащую максимальное значение по определенному атрибуту, можно использовать подзапрос в качестве значения этого атрибута. Например, для вывода строки с максимальным значением рейтинга можно использовать следующий запрос:

SELECT \*

FROM Книги

WHERE рейтинг = (SELECT MAX(рейтинг) FROM Книги);

6. В чем разница между операторами WHERE и HAVING?

Основное отличие WHERE от HAVING заключается в том, что WHERE сначала выбирает строки, а затем группирует их и вычисляет агрегатные функции (таким образом, она отбирает строки для вычисления агрегатов), тогда как HAVING отбирает строки групп после группировки и вычисления агрегатных функций.

7. В чем разница между функциями COUNT и SUM?

SUM вычисляет сумму параметров, а COUNT их количество.

8. Можно ли отсортировать данные таблицы по нескольким атрибутам? Если нет, то почему, если да, то какой будет результат?

SQL допускает сортировку сразу по нескольким полям. Для этого после команды ORDER BY необходимые поля указываются через запятую. Порядок в результате запроса будет настраиваться в той же очередности, в которой указаны поля сортировки.

9. Можно ли одновременно использовать в одном запросе операторы WHERE и HAVING?

В запросе могут содержаться оба предложения: WHERE и HAVING. В этом случае: Предложение WHERE применяется сначала к отдельным строкам таблиц или возвращающих табличное значение объектов на панели диаграмм. Группируются только строки, которые удовлетворяют условиям в предложении WHERE. Затем предложение HAVING применяется к строкам в результирующем наборе. Только строки, которые удовлетворяют условиям HAVING, появляются в результирующем запросе.

10. В какой последовательности происходит сортировка по умолчанию?

Порядок сортировки по умолчанию — ASC (по возрастанию)

11. Как можно отсортировать уже сгруппированные записи?

С помощью условия HAVING

# **Лабораторная работа №9**

**«Инструкция MERGE»**

**Цель:** Изучить принципы слияния таблиц, настройки и параметры инструкции MERGE.

**Задания:**

В соревнованиях по бегу участвовало несколько спортсменов. Для соревнования выделили три бегущих группы. У спортсмена была возможность бежать в любой из этих групп, при этом можно было бежать 2 или 3 раза, в этом случае учитывался бы лучший результат. Каждому участвующему спортсмену дается свой номер. Ниже приведены таблицы с результатами по каждой группе с номером атлета и временем, за которое он пробежал дистанцию.

use [А234]

create table [Группа 1](

[Номер спортсмена] int identity(371,1),

[Время, с] int

);

create table [Группа 2](

[Номер спортсмена] int identity(372,2),

[Время, с] int

);

create table [Группа 3](

[Номер спортсмена] int identity(370,1),

[Время, с] int

);

create table [Сводная табллица](

[Номер спортсмена] int,

[Время 1, с] int,

[Время 2, с] int,

[Время 3, с] int

);

use [А234]

GO

INSERT INTO [Группа 1]

([Время, с])

VALUES

(56),

(57),

(58),

(56),

(51),

(62),

(61),

(67),

(48),

(50)

GO

INSERT INTO [Группа 2]

([Время, с])

VALUES

(61),

(58),

(57),

(54),

(53)

GO

INSERT INTO [Группа 3]

([Время, с])

VALUES

(58),

(57),

(67),

(62),

(55),

(52),

(60),

(58),

(59),

(53),

(54)

GO

Создайте в своей базе данных три таблицы и заполните их, аналогично таблицам 1, 2 и 3. Номер спортсмена должен заполняться не с клавиатуры, а с помощью свойства identity (x, y), которое устанавливает автоинкремент столбца (значение, шаг). Создайте сводную таблицу, в которой будут храниться все данные из первых трех. Добавьте в нее данные с помощью операции MERGE. Незаполненные ячейки должны быть пустыми (NULL).

Запрос:

use [А234]

Merge [Сводная таблица] as Base

using (

select g3.[Номер спортсмена] as number, g1.[Время, с] as time1,g2.[Время, с] as time2, g3.[Время, с] as time3

from [Группа 3] as g3

left join [Группа 2] as g2 on g3.[Номер спортсмена] = g2.[Номер спортсмена]

left join [Группа 1] as g1 on g1.[Номер спортсмена] = g3.[Номер спортсмена]) as Source

on Base.[Номер спортсмена] = Source.number

WHEN MATCHED THEN

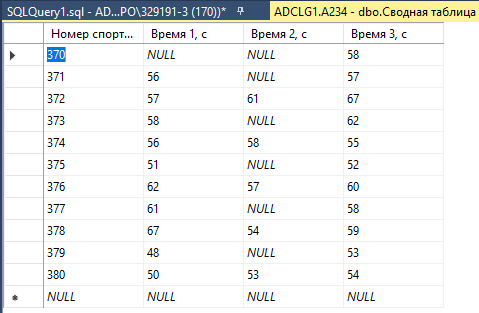
UPDATE SET [Время 1, с] = Source.time1, [Время 2, с] = Source.time2, [Время 3, с] = Source.time3

WHEN NOT MATCHED THEN

INSERT ([Номер спортсмена], [Время 1, с], [Время 2, с], [Время 3, с])

VALUES (Source.number, Source.time1, Source.time2 , Source.time3 );

Результат:



Создайте новую таблицу, в которой будет лучший результат каждого спортсмена. Сортировку выполните с помощью операции MERGE. Обратите внимание, что лучшее время – это минимальное время.

Запрос:

use [А234]

create table [Лучший результат](

[Номер спортсмена] int,

[Лучшее время] int

);

Merge [Лучший результат] as Base

using (

select [Номер спортсмена] as number, MinValue

from [Сводная таблица]

cross apply (SELECT MIN(d) AS MinValue

FROM (VALUES ([Время 1, с]), ([Время 2, с]), ([Время 3, с])) AS a(d)) A) as Source

on Base.[Номер спортсмена] = Source.number

WHEN MATCHED THEN

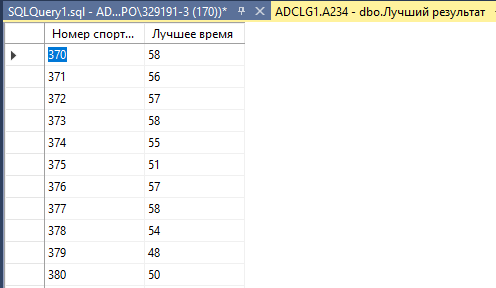
UPDATE SET [Лучшее время] = Source.MinValue

WHEN NOT MATCHED THEN

INSERT ([Номер спортсмена], [Лучшее время])

VALUES (Source.number, Source.MinValue);

Результат:



Выведите на экран номер победителя и его результат любым удобным способом.

Запрос:

use [А234]

Create table [Победитель](

[Номер победителя] int,

[Лучшее время1] int

);

insert into [Победитель]

SELECT [Номер спортсмена], MIN([Лучшее время])

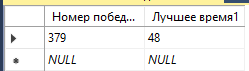
FROM [Лучший результат]

where [Лучшее время] = (SELECT MIN([Лучшее время])

FROM [Лучший результат])

group by [Номер спортсмена]

Результат:



Аналогично выведите на экран худший результат за все соревнование.

Запрос:

delete from [Победитель]

insert into [Победитель]

SELECT [Номер спортсмена], Max([Лучшее время])

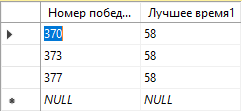
FROM [Лучший результат]

where [Лучшее время] = (SELECT Max([Лучшее время])

FROM [Лучший результат])

group by [Номер спортсмена]

Результат:



# **Лабораторная работа №10**

**«Функции»**

**Цель:** Изучить разные типы функций, научиться создавать и вызывать функции разных типов.

**Задания:**

10\_1. Создайте функцию, выполняющую роль калькулятора, со следующими возможностями: сложение, вычитание, умножение, деление, sin, cos, tg, ctg, возведение в заданную степень.

Запрос:

CREATE FUNCTION Calculator (@n1 numeric(10,0), @oper char(3),@n2 numeric(10,0))

RETURNS numeric(20,2)

BEGIN

DECLARE @Rez numeric(20,2)

RETURN CASE @oper

WHEN '+' THEN @n1 + @n2

WHEN '-' THEN @n1 - @n2

WHEN '\*' THEN @n1 \* @n2

WHEN '/' THEN @n1 / @n2

WHEN 'sin' THEN sin(@n1)

WHEN 'cos' THEN cos(@n1)

WHEN 'tg' THEN tan(@n1)

WHEN 'ctg' THEN COT(@n1)

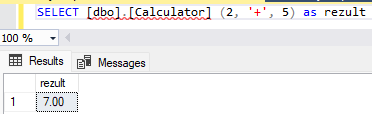
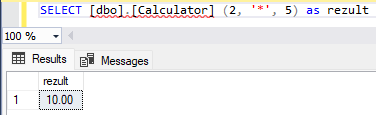
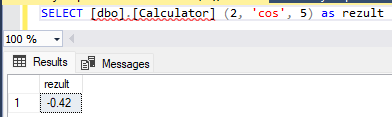
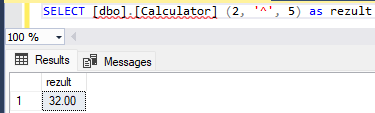
WHEN '^' THEN POWER(@n1, @n2)

ENd

end

10\_2. Протестируйте функцию.

Результат:

10\_3. Создайте функцию, которая при вводе даты, будет возвращать название сезона (Зима, Весна, Лето, Осень).

Запрос:

CREATE FUNCTION GetSizon (@date datetime)

RETURNS CHAR(5)

BEGIN

DECLARE @Rez char(5)

DECLARE @month numeric(2,0)

select @month = MONTH(@date)

if @month =1 or @month=2 or @month=12

SELECT @Rez='Зима'

else if @month =3 or @month=4 or @month=5

SELECT @Rez='Весна'

else if @month =9 or @month=10 or @month=11

SELECT @Rez='Осень'

else if @month =6 or @month=7 or @month=8

SELECT @Rez='Лето'

else

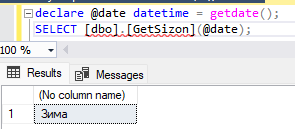
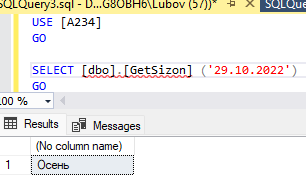
SELECT @Rez='Null'

RETURN @Rez

END

10\_4. Протестируйте функцию.

Результат:

10\_5. Создайте функцию, которая будет разбивать входную строку на отдельные строки. Символ пробела является разделителем строк.

Запрос:

CREATE FUNCTION Words ( @string nvarchar(255))

RETURNS @word TABLE (String nvarchar(255))

AS

Begin

DECLARE @st nVARCHAR(255)

DECLARE @pos INT

WHILE CHARINDEX(' ', @string) > 0

BEGIN

SELECT @pos = CHARINDEX(' ', @string)

SELECT @st = SUBSTRING(@string, 1, @pos-1)

INSERT INTO @word

select @st

SELECT @string = SUBSTRING(@string, @pos+1, LEN(@string)-@pos)

END

INSERT INTO @word

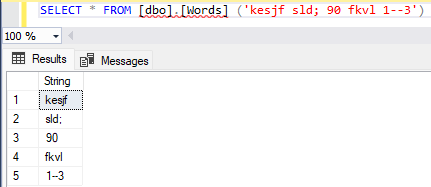
SELECT @string

RETURN

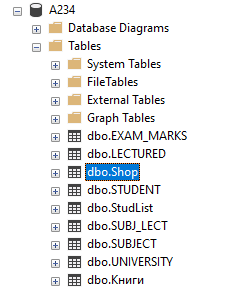
END

10\_6. Протестируйте функцию.

Результат:



10\_7. Импортируйте документ Doc.xsl.



10\_8. Работа с таблицей StudList. Создайте функцию, возвращающую список студентов, обучающихся или отчисленных, в зависимости от входного параметра. Предполагается, что, если в столбце Studying стоит No, студент отчислен, в противном случае – обучается.

Запрос:

CREATE FUNCTION [dbo].[FStudying] ( @question char(3))

RETURNS TABLE

AS

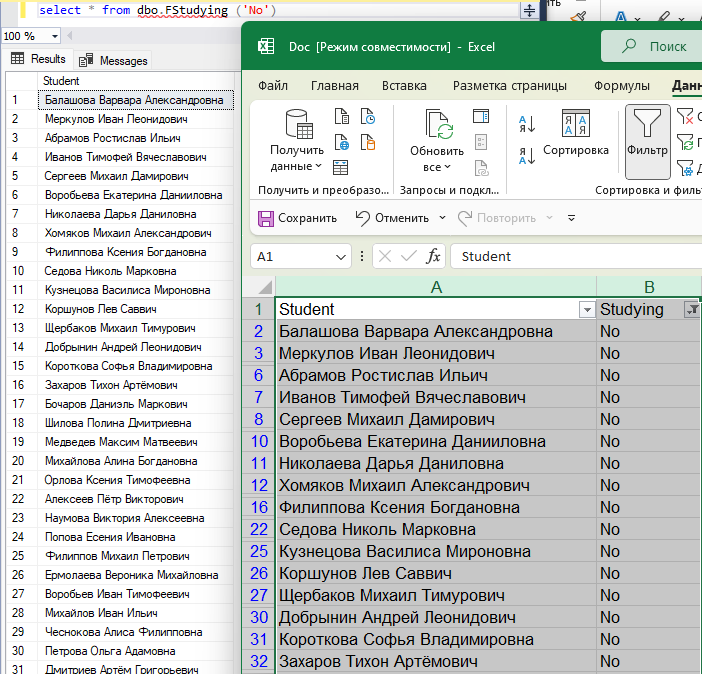
RETURN (select [Student]

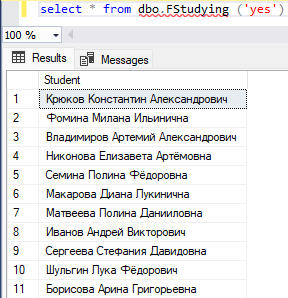
from [dbo].[StudList]

where [Studying] = @question)

10\_9. Протестируйте функцию.

Результат:





# **Лабораторная работа №11**

**«Процедуры»**

**Цель:** Изучить разные типы процедур, научиться создавать и вызывать процедуры без входных и выходных параметров, с входными параметрами.

**Задания:**

11\_1. Работа с таблицей StudList из предыдущей ЛР. Создать процедуру с входным параметром и return. Процедура должна получать входной параметр – Имя студента, а на выходе выдать 1 или 0, в зависимости от того обучается он или нет.

Запрос:

ALTER PROCEDURE [dbo].[PStudying] ( @question varchar(255))

AS

declare @rez int

if EXISTS (select [Studying] from [dbo].[StudList] where [Student] =@question and [Studying] ='No')

set @rez = 0

else set @rez=1

RETURN @rez

2 вариант

ALTER PROCEDURE check\_student

@name VARCHAR(255),

@studying INT OUTPUT

AS

BEGIN

SELECT CASE

WHEN EXISTS(SELECT \* FROM dbo.StudList WHERE [Student] like @name AND [Studying] ='Yes') THEN 1

WHEN EXISTS(SELECT \* FROM dbo.StudList WHERE [Student] like @name AND [Studying] ='No') THEN 0

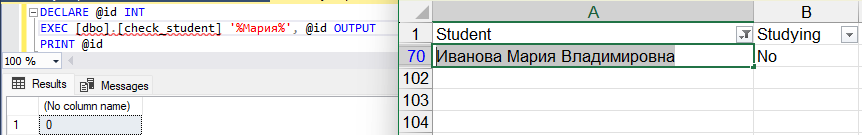
ELSE NULL

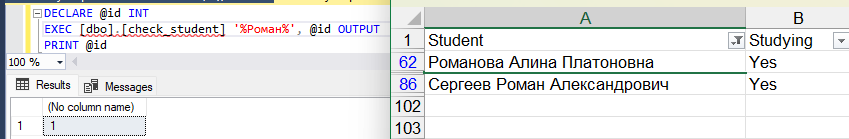
END select @studying;

END;

11\_2. Проверить процедуру.

Результат:





11\_3. Работа с таблицей Shop. Импортируйте таблицу Shop. В магазине проходит акция. Для всех, кто оставил комментарий на сайте, скидка 5%. Создайте процедуру, которая уменьшит цены в таблице Shop на 5% (атрибут Price), в тех строках, где значение столбца Comment yes.

Запрос:

Create PROCEDURE FifePercent AS

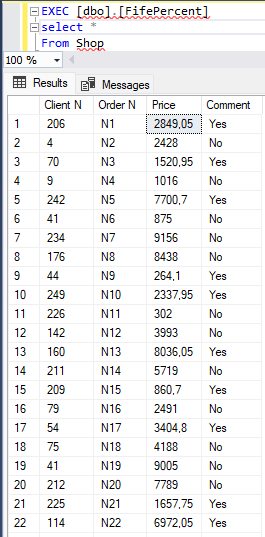
Update Shop

set [Price] = [Price] - [Price]\*0.05

where [Comment] = 'Yes'

11\_4. Проверить процедуру.

Результат:



11\_5. Работа с таблицей Shop. Создать процедуру с входным параметром. Входной параметр – номер клиента. Процедура удаляет все строки с заказами этого клиента.

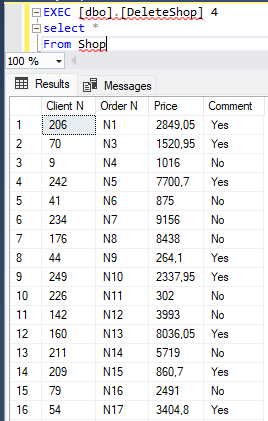
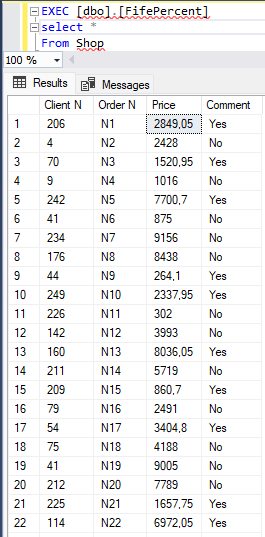
Запрос:

Create PROCEDURE DeleteShop @n numeric(5) AS

Delete from Shop

where [Client N]= @n

Результат:



11\_6. Работа с таблицей Shop. Создать процедуру с входным и выходным параметрами, где входной параметр – номер клиента, а выходной – самый дорогой заказ выбранного клиента.

Запрос:

Create PROCEDURE ExpensiveShop @n numeric(5), @order char(7) OUTPUT AS

select @order=[Order N] from [dbo].[Shop] where [Client N]= @n and [Price] >= ALL(select [Price]

from [dbo].[Shop]

where [Client N]= @n)

2 вариант

CREATE PROCEDURE get\_max\_order

@client\_id INT,

@max\_order DECIMAL(10,2) OUTPUT

AS

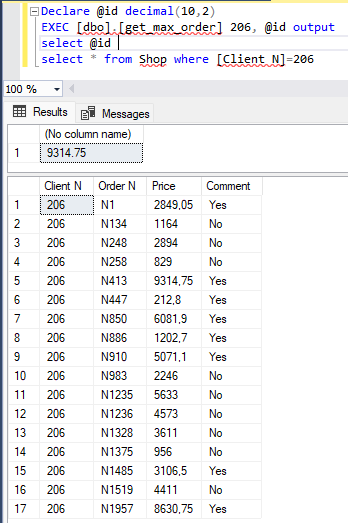
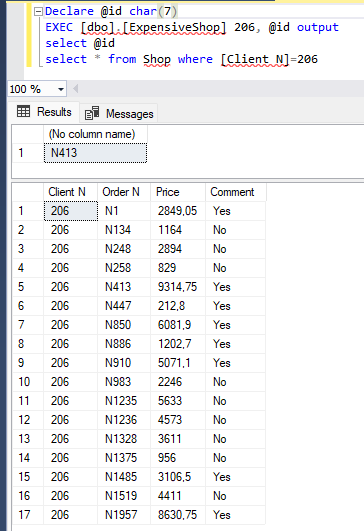
BEGIN

select @max\_order= MAX([Price]) FROM [dbo].[Shop] WHERE [Client N] = @client\_id

END;

11\_7. Проверить процедуру.

Результат:



**Контрольные вопросы:**

1. Какие типы процедур вы знаете?

Пользовательские процедуры, Временные процедуры, Системные процедуры

2. Где хранятся процедуры?

Во внутренней скрытой базе данных Resource

3. Какой синтаксис у создания процедур?

Общий синтаксис выглядит так:

CREATE PROCEDURE Имя\_процедуры AS

Тело процедуры

4. Какой синтаксис у выполнения процедур?

Чтобы вызвать процедуры используется команда EXEC или EXECUTE.

EXEC Имя\_процедуры

5. В чем разница между функцией и процедурой?

Основное отличие между процедурой и функцией в SQL заключается в том, что процедура не обязана возвращать результат, тогда как функция всегда должна возвращать значение. В процедуре можно использовать оператор RETURN, но он не обязателен, а в функции оператор RETURN обязателен и должен содержать возвращаемое значение. И еще один важный аспект - производительность. Хранимые процедуры обычно выполняются быстрее, чем обычные SQL-инструкции. Все потому что код процедур компилируется один раз при первом ее запуске, а затем сохраняется в скомпилированной форме.

6. Зачем нужны процедуры?

Часто операция с данными представляет набор инструкций, которые необходимо выполнить в определенной последовательности или набор действий, которые нужно выполнять с некоторой периодичностью. В этом случае более оптимально будет инкапсулировать все эти действия в один объект - хранимую процедуру.

То есть, по сути, хранимые процедуры представляют набор инструкций, которые выполняются как единое целое. Тем самым хранимые процедуры позволяют упростить комплексные операции и вынести их в единый объект. Также хранимые процедуры позволяют ограничить доступ к данным в таблицах и тем самым уменьшить вероятность преднамеренных или неосознанных нежелательных действий в отношении этих данных.

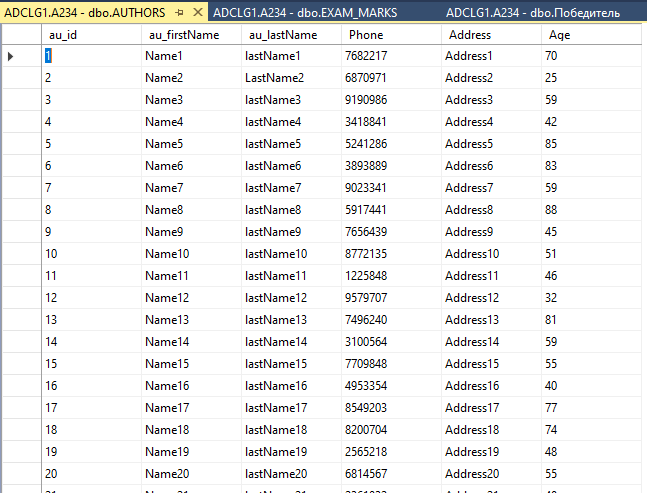
# **Лабораторная работа №12**

**«Создание, программирование и управление триггерами»**

**Цель:** Изучить назначения и типы триггеров, условия их активации, синтаксис команд языка T – SQL для их создания, модификации, переименования, программирования и удаления.

**Задания:**

12\_1. Импортируйте данные из файла AUTH.xls



12\_2. Создайте триггер Trigger\_1, который после изменения данных будет выводить строки, со старыми и новыми значениями, используйте триггер AFTER UP-DATE. Протестируйте ваш триггер.

Запрос:

CREATE TRIGGER Trigger\_1

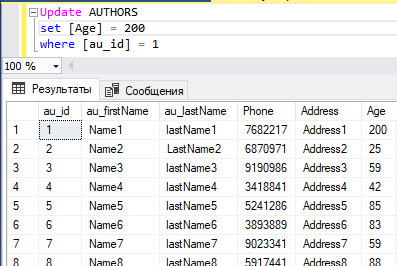
ON AUTHORS

AFTER

UPDATE

AS select \* from AUTHORS

Результат:



12\_3. Создайте триггер Trigger\_2, который вместо удаления выведет следующую информацию: «попытка удаления информации об au\_id». Используйте триггер INSTEAD OF – триггеры вызывается вместо выполнения команд списка. Протестируйте ваш триггер.

Запрос:

ALTER TRIGGER Trigger\_2

ON AUTHORS

INSTEAD OF

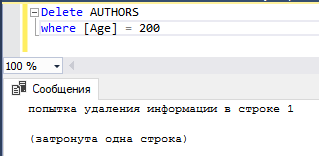
DELETE

AS declare @id int

set @id = (select au\_id from deleted)

print Concat('попытка удаления информации в строке ',@id)

Результат:



12\_4. Создайте триггер Trigger\_3, который будет считать и выводить количество удаленных строк. Протестируйте ваш триггер.

Запрос:

Create TRIGGER Trigger\_3

ON AUTHORS

AFTER

DELETE

AS declare @id int

set @id = (select Count(au\_id) from deleted)

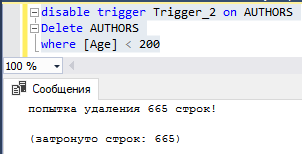
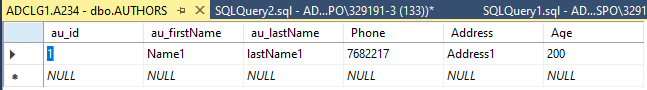
print Concat('попытка удаления ',@id, ' строк!')

Результат:

disable trigger Trigger\_2 on AUTHORS

Delete AUTHORS

where [Age] < 200

**Контрольные вопросы:**

1. Какие типы триггеров вы знаете?

DML

Входа

Системные

2. Можем ли мы обращаться к журналу транзакций при создании триггеров?

Да

3. Как можно протестировать ваш триггер?

Проверить триггер можно написав запрос, воспроизводящий условия триггера, для любой из тестовых записей в таблице.

4. Сколько триггеров может иметь одно отношение?

Можно определить несколько AFTER-триггеров для каждой операции (INSERT, UPDATE, DELETE)

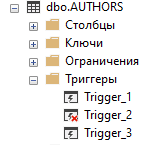
Для каждой операции INSERT, UPDATE, DELETE можно определить только один INSTEAD OF-триггер.

5. Можно ли создать один триггер на несколько операций?

Можно написать триггер сразу на все события (insert, delete, update)

6. Как можно увидеть созданные триггеры?

В папке таблицы, для которой был создан этот триггер.



7. Как можно изменить тело триггера?

ALTER TRIGGER

# **Лабораторная работа №13**

**«Создание и управление транзакциями»**

**Цель:** Изучение способов обеспечения надежной работы SQL Server с помощью механизма транзакций и контрольных точек, приобретение навыков управления транзакциями, а также ознакомление с физической и логической архитектурой журнала транзакций.

**Задания:**

13\_1. Проверьте работу транзакции следующим образом. Выберите любое отношение в вашей базе данных. Создайте транзакцию, в которой будет содержаться один запрос вставки. Запустите ее дважды. В первом случае завершите транзакцию командой COMMIT, второй раз ROLLBACK. Проанализируйте результаты. Вставьте скриншот результата в отчет.

Запрос:

CREATE TABLE table1 (a1 int)

BEGIN TRANSACTION tr1

INSERT INTO table1 VALUES (11)

COMMIT TRANSACTION tr1--фиксирование изменений и конец транзакции

SELECT \* FROM table1

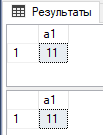
BEGIN TRANSACTION tr1

INSERT INTO table1 VALUES (13)

ROLLBACK TRANSACTION--откат транзакции

SELECT \* FROM table1

Результат: При использовании COMMIT изменения сохраняются, а с ROLLBACK нет.



13\_2. Работа с вложенными транзакциями.

Создать вложенные транзакции, выполнив следующие команды:

CREATE TABLE table\_name (a1 int) -- 0-й уровень

BEGIN TRANSACTION tr1-- 1-й уровень

INSERT INTO table\_name VALUES (11)

BEGIN TRANSACTION tr2-- 2-й уровень

INSERT INTO table\_name VALUES (22)

BEGIN TRANSACTION tr3 -- 3-й уровень

INSERT INTO table\_name VALUES (33)

SELECT \* FROM table\_name

SELECT 'Вложенность транзкций', @@TRANCOUNT

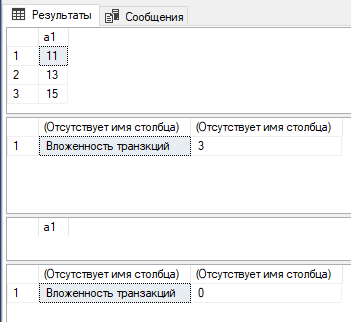
ROLLBACK TRANSACTION

SELECT \* FROM table\_name -- откат на 0-й уровень

SELECT 'Вложенность транзакций', @@TRANCOUNT

Проанализируйте результаты.

Все транзакции были выполнены, затем произошел откат, следовательно не выполнена ни одна транзакция.



13\_3. Работа с данными из файла groups. Импортируйте все листы. В первом листе находятся данные студентов, поступивших в 2018 году. В других листах содержаться списки групп. Задача импортировать данные из списков групп в сводную таблицу всех поступивших. Создайте явную транзакцию, в которой данные о студентах будут вноситься, а, если эти данные уже были внесены раннее, то транзакция будет откатываться. Результаты продемонстрируйте в отчете.

Код:

BEGIN TRANSACTION tr1

declare @group int

select @group = группа

from гр2

if @group not in (select группа from гр2018)

begin

insert into гр2018

select \* from гр2

Commit transaction tr1

end

else rollback transaction tr1

begin transaction tr2

select @group = группа

from гр3

if @group not in (select группа from гр2018)

begin

insert into гр2018

select \* from гр3

Commit transaction tr2

end

else rollback transaction tr2

Результат:



13\_4. Изучите самостоятельно команду SET IMPLICIT\_TRANSACTIONS ON/OFF.

Продемонстрируйте разницу между этими режимами на примере любой таблицы в вашей базе данных.

Код:

SET IMPLICIT\_TRANSACTIONS ON

Select 'Вложенность тарнзакций', @@trancount

delete from гр2018

where фио ='Лукьянова Яна Дмитриевна '

Select 'Вложенность тарнзакций', @@trancount

select \* from гр2018

rollback transaction

SET IMPLICIT\_TRANSACTIONS off

Select 'Вложенность тарнзакций', @@trancount

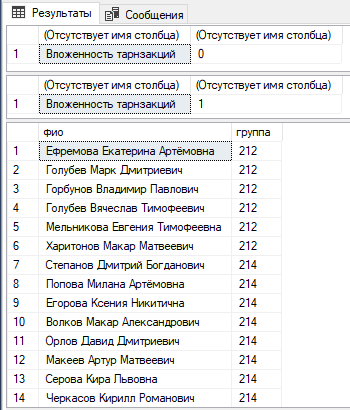
delete from гр2018

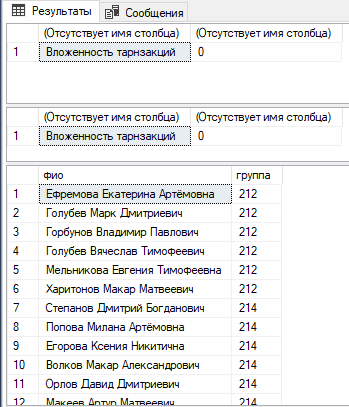
where фио ='Лукьянова Яна Дмитриевна '

Select 'Вложенность тарнзакций', @@trancount

select \* from гр2018

Результат:





**Контрольные вопросы:**

1. Зачем нужны транзакции?

Сохранение целостности данных.

Параллельную работу пользователей с базой данных.

Восстановление данных при откатах и сбоях.

2. С какими режимами транзакции вы работали?

Явные транзакции

Неявные транзакции

3. Какие команды являются признаком начала неявной транзакции?

SET IMPLICIT\_TRANSACTIONS ON

4. Зачем нужны контрольные точки транзакций?

Если в команде ROLLBACK не задано имя транзакции, то откатываются все вложенные транзакции и транзакция самого высокого уровня. Если же имеется необходимость откатить лишь часть транзакций, то предварительно надо создать точку сохранения с помощью команды SAVE TRANSACTION, которую следует указывать при откате.

5. Можно ли узнать количество активных транзакций?

Да, при помощи @@TRANCOUNT

# **Лабораторная работа №14**

**Создание и управление курсорами**

**Цель:** Изучение назначения и типов курсоров, синтаксиса языка T – SQL для создания и открытия курсоров, выборки данных из курсора и изменения строк таблиц с помощью курсоров, удаления данных, закрытия и освобождения курсоров, а также приобретения навыков их применения и управления с помощью команд и системных хранимых процедур SQL Server.

**Задания:**

14\_1. Создайте таблицу Clients со следующими атрибутами: Id, Name, Address.

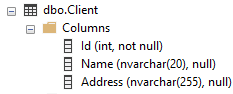
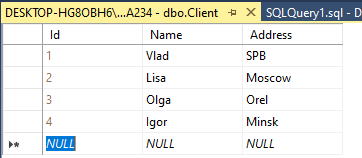
Запрос:

Create table Client(Id int identity(1,1) not null,

[Name] nvarchar(20),

[Address] nvarchar (255));

Результат:

14\_2. С помощью курсора создайте запрос, который будет выводить информацию о всех клиентах c нечетным Id.

Запрос:

DECLARE curs1 CURSOR

FOR

Select \*

From [dbo].[Client]

where [Id]%2<>0

OPEN curs1

DECLARE @id int

DECLARE @name VARCHAR(20)

DECLARE @address VARCHAR(255)

FETCH NEXT FROM curs1 INTO @id, @name, @address

-- Проверить состояние курсора

WHILE @@FETCH\_STATUS = 0

BEGIN

select @id, @name, @address

-- Извлечь следующую запись из курсора

FETCH NEXT FROM curs1 INTO @id, @name, @address

END

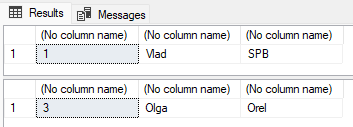
-- Закрыть курсор

CLOSE curs1

-- Освободить ресурсы

DEALLOCATE curs1

Результат:



14\_3. Создайте таблицу OlympMath с результатами Олимпиады по математике за 2019, 2020 и 2021 года. Таблица OlympMath состоит из следующих атрибутов: id, Name, res19, res20, res21.

Запрос:

Create table OlympMath(id int identity(1,1) not null,

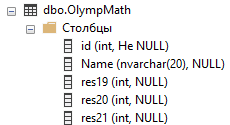
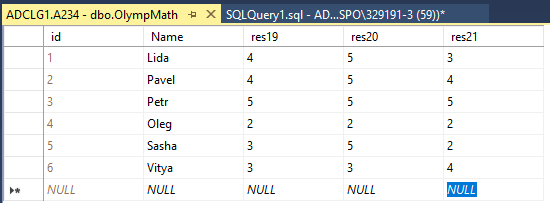
[Name] nvarchar(20),

res19 int,

res20 int,

res21 int);

Результат:

14\_4. С помощью курсора создайте запрос, который будет находить и выводить лучший результат для каждого участника. То есть максимальный в строке.

Запрос:

DECLARE curs1 CURSOR

FOR

Select \*

From [dbo].[OlympMath]

OPEN curs1

DECLARE @id int

DECLARE @name VARCHAR(20)

DECLARE @res1 int

DECLARE @res2 int

DECLARE @res3 int

FETCH NEXT FROM curs1 INTO @id, @name, @res1, @res2, @res3

-- Проверить состояние курсора

WHILE @@FETCH\_STATUS = 0

BEGIN

select @id, @name, Max(v)

FROM (VALUES (@res1), (@res2), (@res3)) AS value(v)

-- Извлечь следующую запись из курсора

FETCH NEXT FROM curs1 INTO @id, @name, @res1, @res2, @res3

END

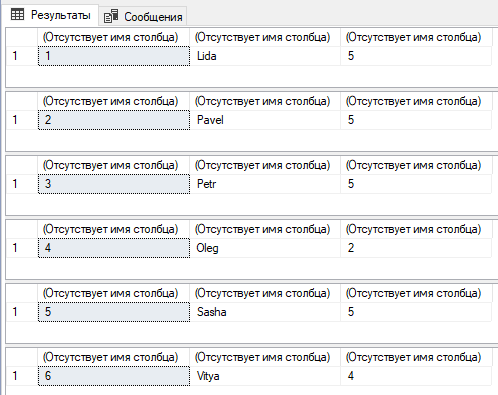
-- Закрыть курсор

CLOSE curs1

-- Освободить ресурсы

DEALLOCATE curs1

Результат:



14\_5. Создайте две таблицы со следующей структурой (папки и файлы):

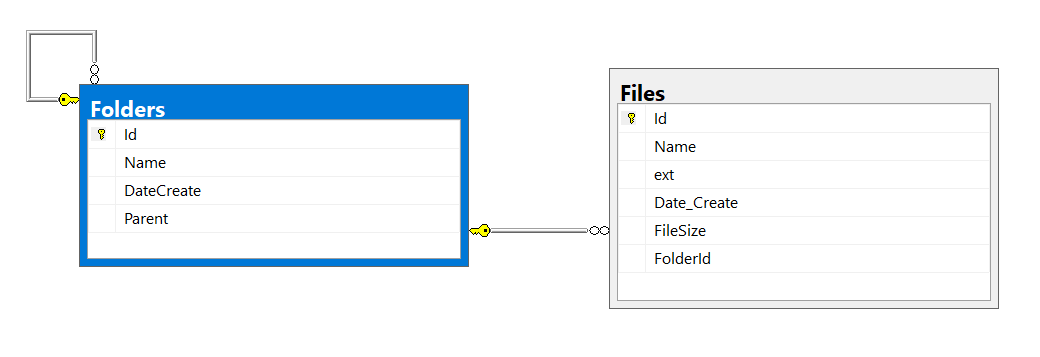
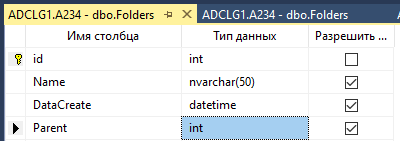
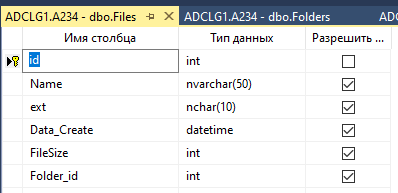
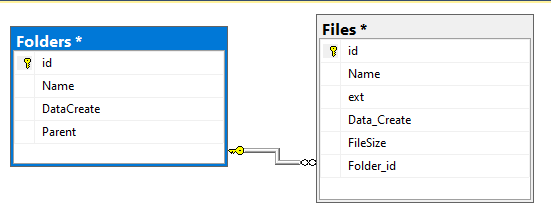


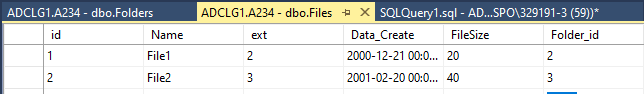
Рис.14 Таблицы Папки и файлы

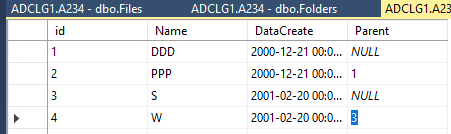
Результат:











Реализуйте функцию, которая будет принимать на входе идентификатор каталога и выводить все файлы и каталоги, которые находятся в заданном каталоге и во всех подкаталогах.

Запрос:

Alter FUNCTION GetFiles (@idInput int)

RETURNS nvarchar(255)

BEGIN

DECLARE @Rez nVARCHAR(255)

if((Select [Name]

From [dbo].[Folders]

where Parent =@idInput)<>'')

SET @Rez = Concat(@Rez,

(Select [Name]

From [dbo].[Folders]

where Parent =@idInput), ' ',[dbo].[GetFiles] ((Select [id]

From [dbo].[Folders]

where Parent =@idInput)))

SET @Rez = Concat(@Rez,

(Select [Name]

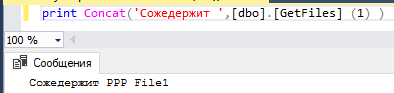
From [dbo].[Files]

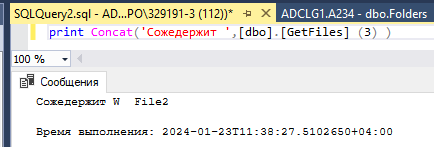
where Folder\_id =@idInput), ' ')

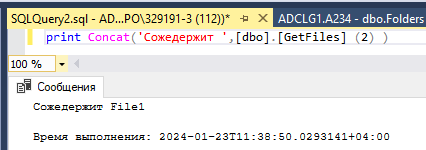
RETURN @Rez

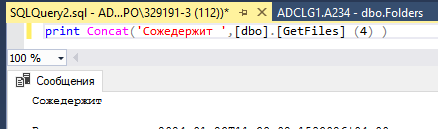
END

Результат:









# **Лабораторная работа №15**

**Подключение к БД, отображение данных в форме с помощью DataGridView**

**Цель:** Изучение синтаксиса для подключения к базе данных через приложение Windows Forms. Простейший способ отображения данных.

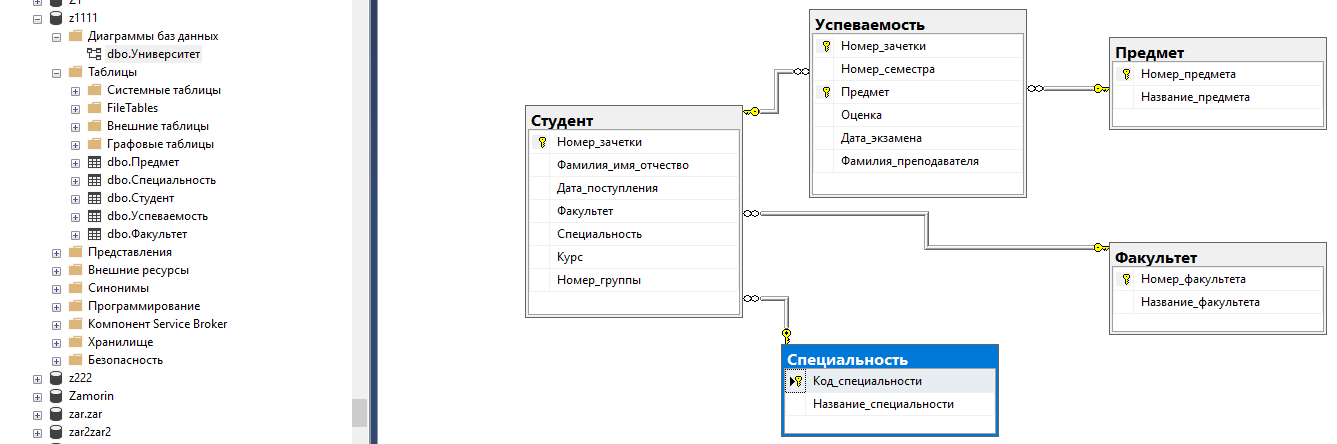
**Задания:**

1. Создайте отношения в базе данных, согласно схеме ниже (Рис. 16).



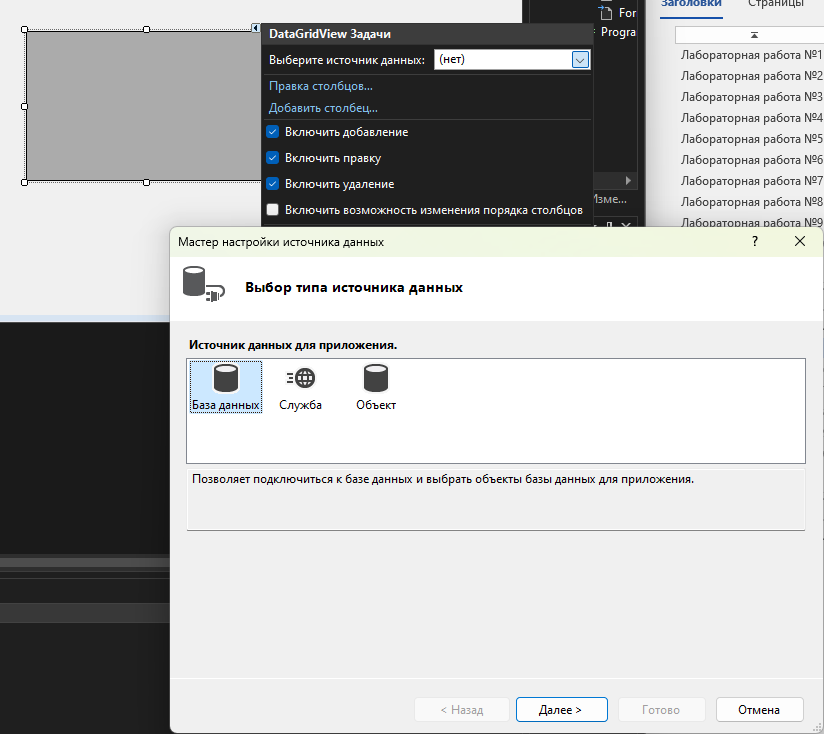
Рис.16 Схема данных.

2. Заполните каждую из таблиц данными (минимум три строки в каждой таблице).



3. Создайте интерфейс с помощью Windows Forms со следующими требованиями:

• Все таблицы должны отображаться с помощью элемента DataGridView.



• Подключения к БД производится через строку подключения.

using System.Data.SqlClient;

namespace WindowsFormsApp1

{

internal class DataBase

{

SqlConnection conn= new SqlConnection(@"Data Source = DESKTOP-HG8OBH6\SQLEXPRESS; Initial catalog = z111; Integrated Security=True;");

public void OpenConection() {

if(conn.State==System.Data.ConnectionState.Closed)

conn.Open();

}

public void CloseConection() {

if (conn.State == System.Data.ConnectionState.Open)

conn.Close();

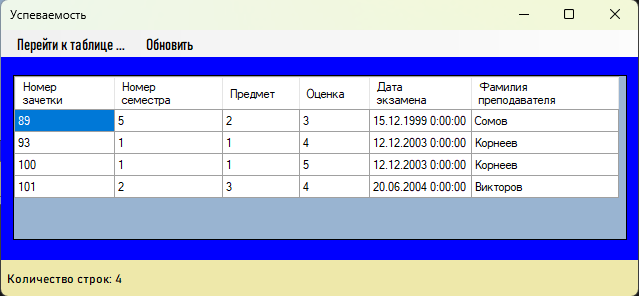
}

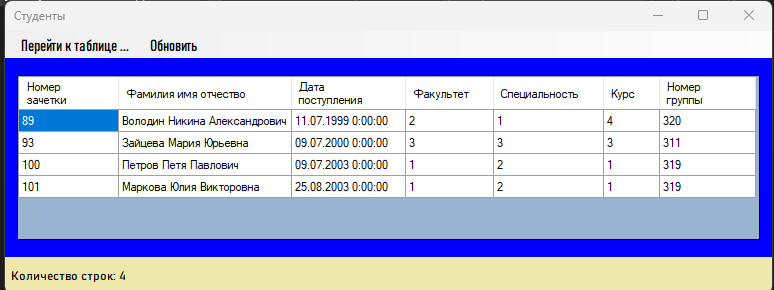
public SqlConnection GetConection() => conn;

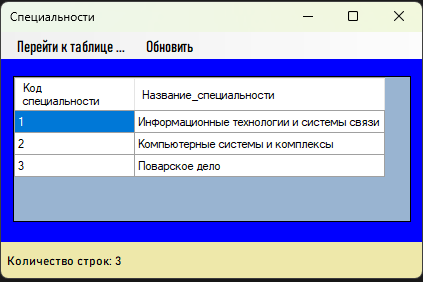
}

}

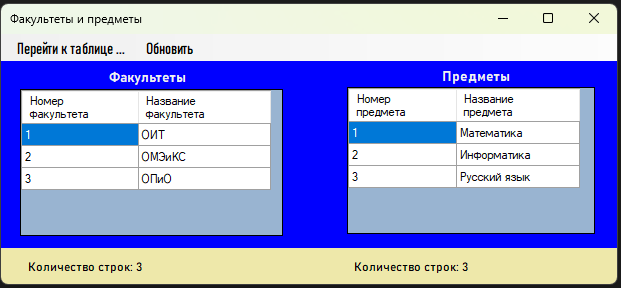
• Таблицы Студент, Успеваемость, Специальность отображаются на отдельных формах.



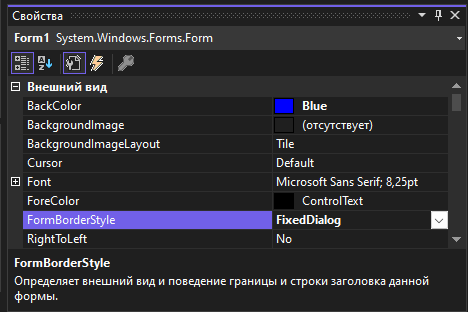




• Таблицы Предмет и Факультет отображаются на одной форме, но на разных элементах panel.



• Размер формы должен быть фиксированный. Изменения размера недоступно для пользователя.



• У каждой формы должно быть имя.

• На формах должны быть кнопки для переключения между таблицами.



• На формах должны быть кнопки для обновления данных.



• Все элементы должны быть понятны пользователю, а текст должен легко читаться.

• Цвет фона формы выберите в соответствии с вариантом:

|  |  |
| --- | --- |
| 3 | Blue |

4. Отчет должен содержать скриншоты отношений из БД, листинг кода для создания форм, а также листинг SQL запросов, скриншоты форм.

**Код форм: (одинаковый функционал на всех формах)**

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

using ClassLibraryDataBase;

namespace WindowsFormsApp1

{

public partial class FormStudent : Form

{

DataBase dataBase = new DataBase();

public FormStudent()

{

InitializeComponent();

}

public void CreateColumns()

{

dataGridView1.ColumnCount = 7;

dataGridView1.Columns[0].HeaderText = "Номер зачетки";

dataGridView1.Columns[1].HeaderText = "Фамилия имя отчество";

dataGridView1.Columns[2].HeaderText = "Дата поступления";

dataGridView1.Columns[3].HeaderText = "Факультет";

dataGridView1.Columns[4].HeaderText = "Специальность";

dataGridView1.Columns[5].HeaderText = "Курс";

dataGridView1.Columns[6].HeaderText = "Номер группы";

}

public void ReadWriteInTable(DataGridView dg)

{

//запрос который будет выполнен

string query = "SELECT \* FROM Студент";

//связь с sql передача туда запроса и строки для соединения

SqlCommand command = new SqlCommand(query, dataBase.GetConection());

dataBase.OpenConection();//открыть соединение

//объект для чтения из бд

SqlDataReader reader = command.ExecuteReader();

dg.Rows.Clear();//очистка строк

while (reader.Read())

{

//запись прочитанной строки в таблицу

dg.Rows.Add(reader.GetInt32(0), reader.GetString(1), reader.GetDateTime(2), reader.GetInt32(3), reader.GetInt32(4), reader.GetInt32(5), reader.GetInt32(6));

}

reader.Close();

}

private void FormStudent\_Load(object sender, EventArgs e)

{

CreateColumns();

ReadWriteInTable(dataGridView1);

labelRowsCount.Text += dataGridView1.RowCount;

}

private void toolStripTextBoxMark\_Click(object sender, EventArgs e)

{

Form ifrm = new FormMark();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void toolStripTextBoxObj\_Click(object sender, EventArgs e)

{

Form ifrm = new FormObject();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void toolStripTextBoxSpecial\_Click(object sender, EventArgs e)

{

Form ifrm = new FormSpecial();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void FormStudent\_FormClosed(object sender, FormClosedEventArgs e)

{

if (Application.OpenForms.Count > 0)

{

// вызываем главную форму, которая открыла текущую, главная форма всегда = 0 - [0]

Form ifrm = Application.OpenForms[0];

ifrm.StartPosition = FormStartPosition.Manual; // меняем параметр StartPosition у Form1, иначе она будет использовать тот, который у неё прописан в настройках и всегда будет открываться по центру экрана

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form1

}

}

private void toolStripMenuItemUpDate\_Click(object sender, EventArgs e)

{

ReadWriteInTable(dataGridView1);

}

}

}

Полезная литература:

[Категории (irisclasson.com)](https://www.irisclasson.com/categories) [Перенос БД из SQLServer в локальную (microsoft.com)](https://social.msdn.microsoft.com/Forums/ru-RU/8ffc0870-ae71-4880-8d44-040d7a8f63e1/-sqlserver-?forum=fordesktopru)

[How to script/export data and or schema from MSSQL tables (irisclasson.com)](https://www.irisclasson.com/2017/02/02/how-to-scriptexport-data-and-or-schema-from-mssql-tables/)

[Связи между таблицами базы данных / Хабр (habr.com)](https://habr.com/ru/articles/488054/)

# **Лабораторная работа №16**

**Удаление, добавление и изменение данных с помощью запросов**

**Цель:** Изучение синтаксиса для управления данными с помощью элементов формы.

**Задания:**

1. Импортируйте документ Bibl.xsl в вашу БД.



2. Создайте интерфейс, в котором будет DataGridView, текстовые поля для добавления и изменения данных, а также интерфейс для удаления данных.

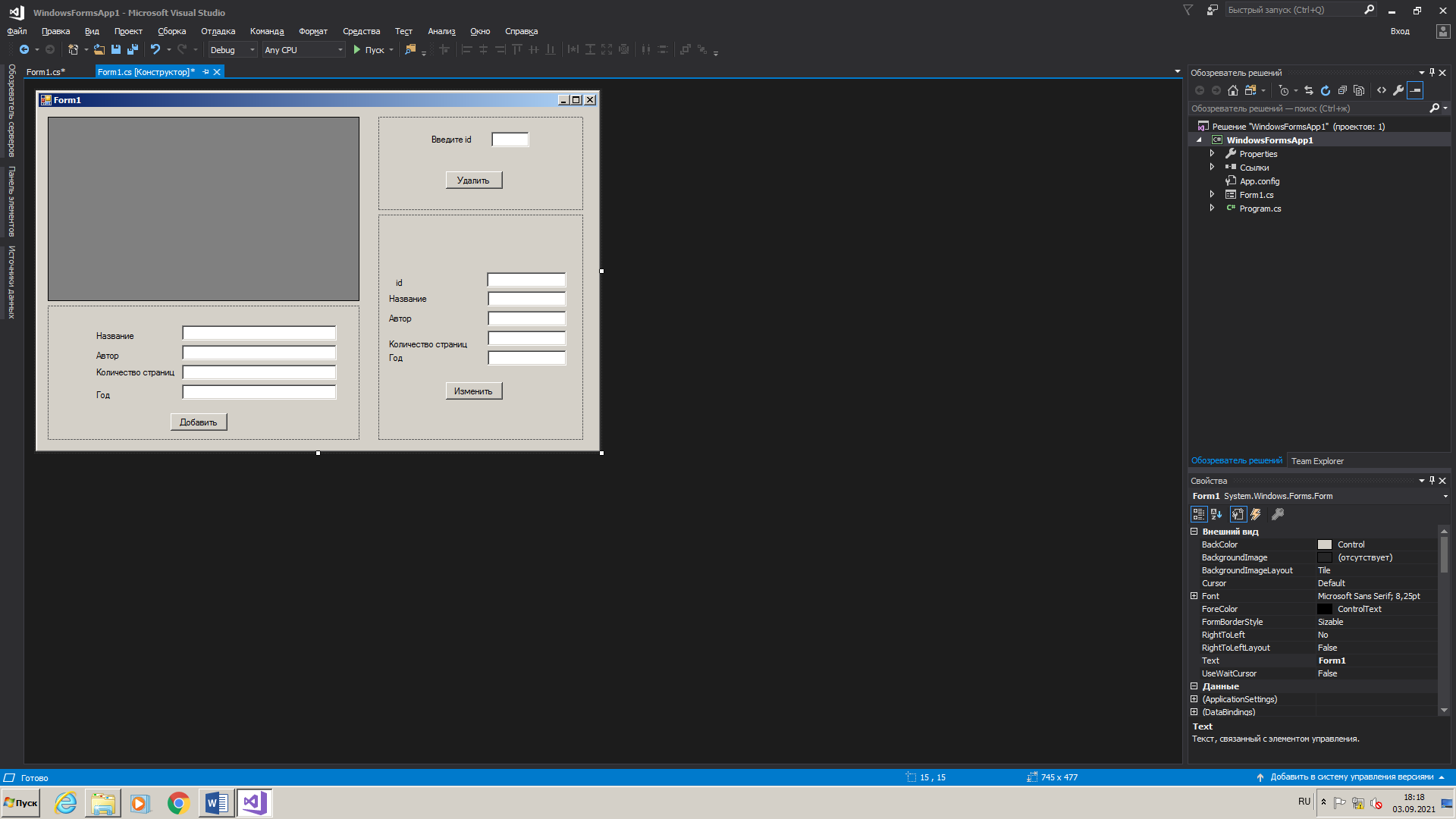
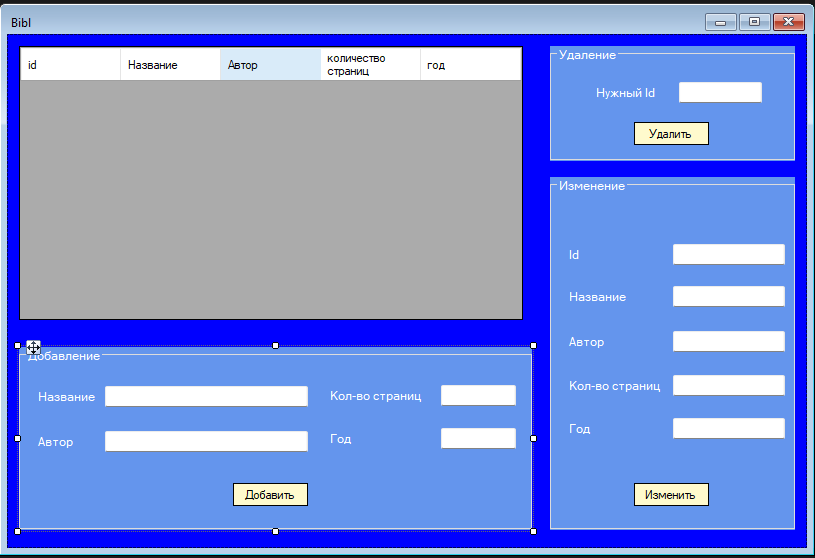


Рис. 18 – Возможный вид вашей формы.



3. Реализуйте функции добавления, изменения и добавления данных с помощью SQL запросов. Обратите внимание, что результат ваших манипуляций должен сразу отображаться на DataGridView.

4. Проверьте правильность работы вашего приложения. Отчёт должен содержать листинг кода и скриншоты результатов.

**Код формы:**

using System;

using System.Configuration;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class FormBibl : Form

{

//строка подключения

string strConnectionString = ConfigurationManager.ConnectionStrings["ConnectionString\_z111"].ConnectionString;

public FormBibl()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

TableUpDate();

}

//запись данных из бд в таблицу

public void TableUpDate()

{

this.biblTableAdapter.Fill(this.z111DataSet.Bibl);

}

private void buttonDel\_Click(object sender, EventArgs e)

{

if (textBoxIdDel.Text != String.Empty)

{

//соединение

SqlConnection MyConnection = new SqlConnection(strConnectionString);

//запрос

string ComDel = " Delete from Bibl where id=@code";

//команда на выполенеие запроса через соединение

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

//запись в параметр значения

SqlParameter pr1 = new SqlParameter("@code", textBoxIdDel.Text);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr1);

MyConnection.Open();

if (cmd1.ExecuteNonQuery()<1) {

MessageBox.Show("Такого id не существует", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else TableUpDate();

MyConnection.Close();

}

else MessageBox.Show("Пустое поле id", "Ошибка",MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void buttonUpDate\_Click(object sender, EventArgs e)

{

if (textBoxIdUp.Text != String.Empty)

{

//соединение

SqlConnection MyConnection = new SqlConnection(strConnectionString);

//запрос

string ComDel = " Update Bibl set Название=@name, Автор=@avtor, [количество страниц]=@page, год=@year where id=@code";

//команда на выполенеие запроса через соединение

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

//запись в параметр значения

SqlParameter pr1 = new SqlParameter("@code", textBoxIdUp.Text);

SqlParameter pr2 = new SqlParameter("@name", textBoxNameUp.Text);

SqlParameter pr3 = new SqlParameter("@avtor", textBoxAvtorUp.Text);

SqlParameter pr4 = new SqlParameter("@page", textBoxCountPageUp.Text);

SqlParameter pr5 = new SqlParameter("@year", textBoxYearUp.Text);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr1);

cmd1.Parameters.Add(pr2);

cmd1.Parameters.Add(pr3);

cmd1.Parameters.Add(pr4);

cmd1.Parameters.Add(pr5);

MyConnection.Open();

if (cmd1.ExecuteNonQuery() <1 )

{

MessageBox.Show("Такого id не существует", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else TableUpDate();

MyConnection.Close();

}

else MessageBox.Show("Пустое поле id", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void buttonInsert\_Click(object sender, EventArgs e)

{

if ((textBoxNameIns.Text != String.Empty)&&(textBoxAvtorIns.Text != String.Empty) &&(textBoxCountPageIns.Text != String.Empty) &&(textBoxYearIns.Text != String.Empty))

{

//соединение

SqlConnection MyConnection = new SqlConnection(strConnectionString);

//запрос

string ComDel = " Insert into Bibl ([Название], [Автор], [количество страниц], [год]) values (@name, @avtor, @page, @year)";

//команда на выполенеие запроса через соединение

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

//запись в параметр значения

SqlParameter pr2 = new SqlParameter("@name", textBoxNameIns.Text);

SqlParameter pr3 = new SqlParameter("@avtor", textBoxAvtorIns.Text);

SqlParameter pr4 = new SqlParameter("@page", textBoxCountPageIns.Text);

SqlParameter pr5 = new SqlParameter("@year", textBoxYearIns.Text);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr2);

cmd1.Parameters.Add(pr3);

cmd1.Parameters.Add(pr4);

cmd1.Parameters.Add(pr5);

MyConnection.Open();

cmd1.ExecuteNonQuery();

TableUpDate();

MyConnection.Close();

}

else MessageBox.Show("Нехватает информации о новой книге", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

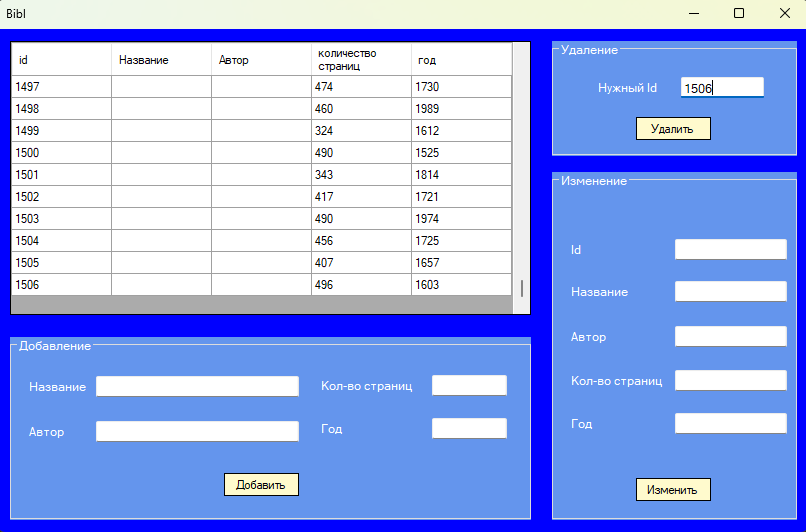
}

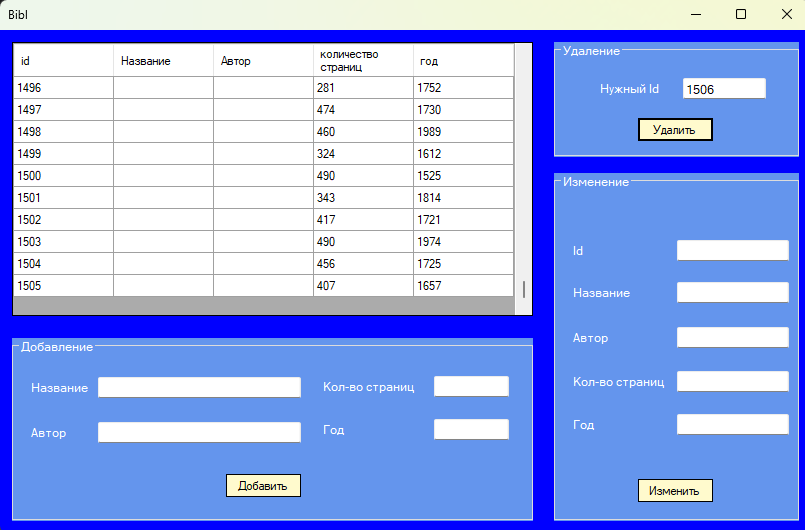
}

}

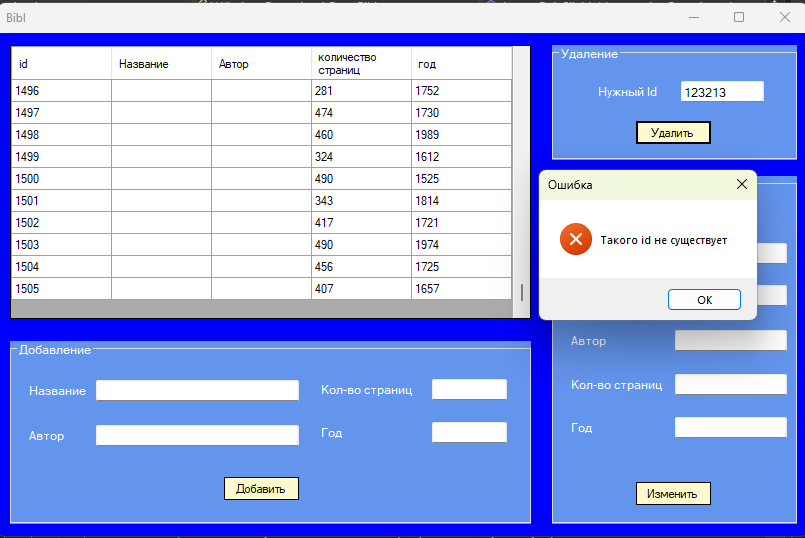
Результат:

Удаление:

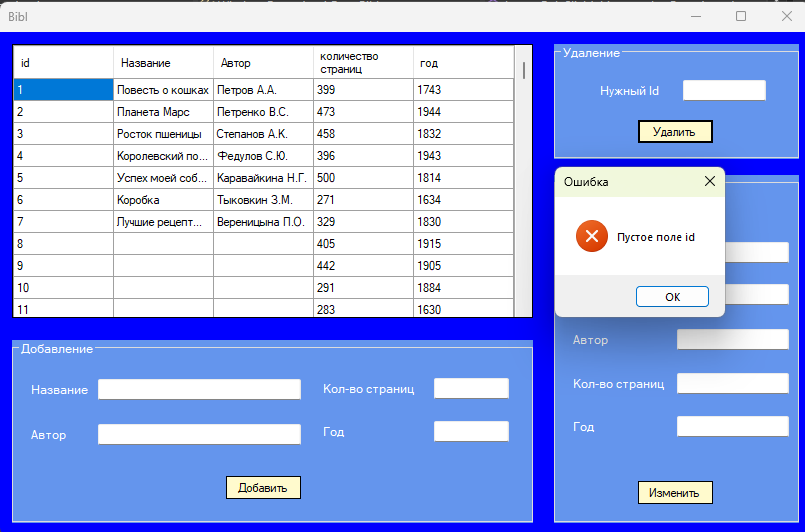




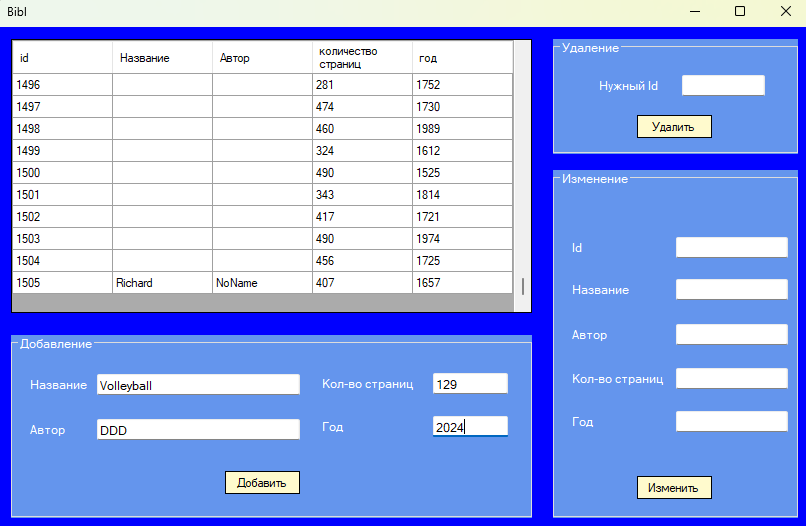
Удаление несуществующей книги

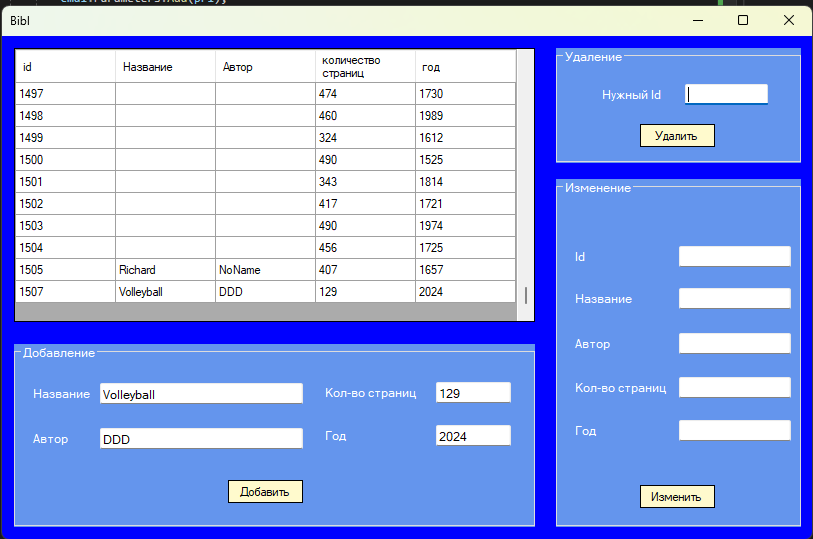


Удаление пустого поля

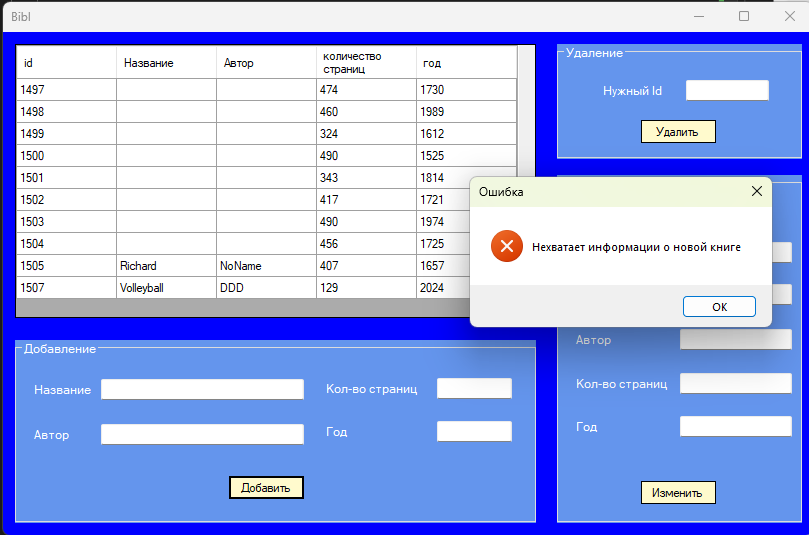


Добавление:

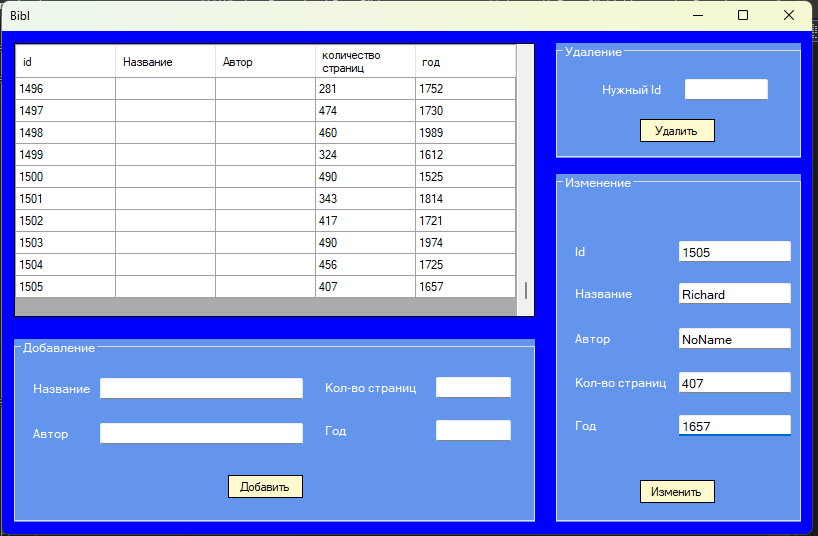


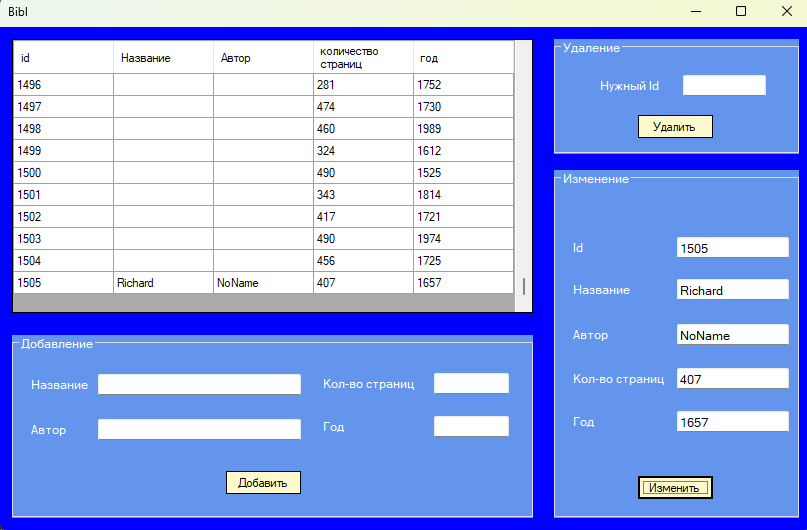


Добавление пустого поля

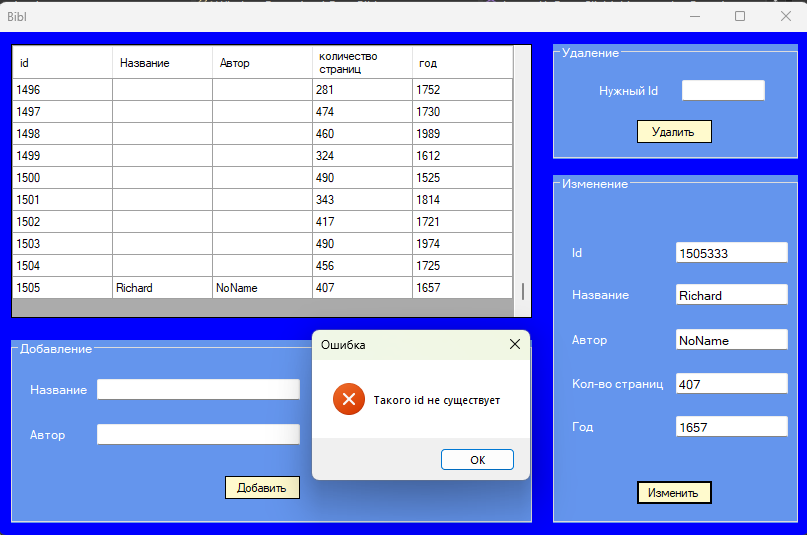


Изменение:



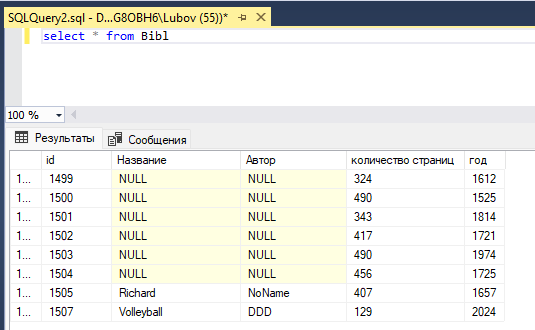


Изменение несуществующей книги



Изменение пустого поля





Полезная литература:

[ConfigurationManager.ConnectionStrings Свойство (System.Configuration) | Microsoft Learn](https://learn.microsoft.com/ru-ru/dotnet/api/system.configuration.configurationmanager.connectionstrings?view=windowsdesktop-8.0)

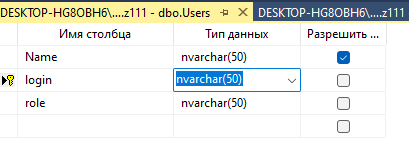
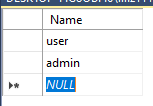
# **Лабораторная работа №17**

**Различные способы отображения данных**

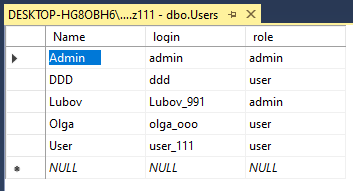
**Цель:** Изучение различных элементов формы, которые можно использовать для отображения данных, а также их синтаксис.

**Задания:**

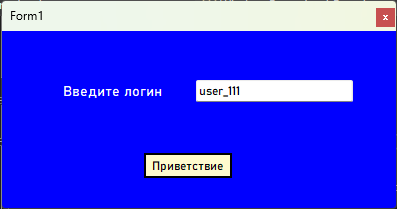
1. Создайте таблицу Users с атрибутами Name, login, role. Причем значение login должно быть уникальным, а role может быть user или admin.

2. Занесите в таблицу Users 5 строк.



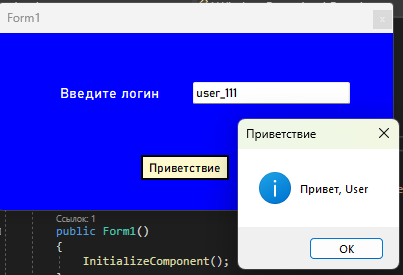
3. Создайте форму. Разместите на ней объекты label, button и textbox.



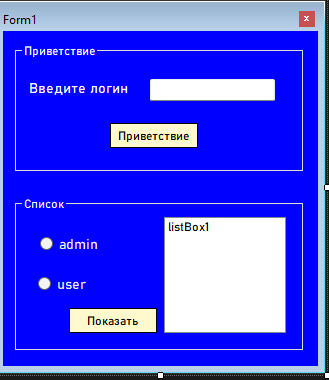
4. Выполните подключение к БД.

SqlConnection conn = new SqlConnection(@"Data Source = DESKTOP-HG8OBH6\SQLEXPRESS; Initial catalog = z111; Integrated Security=True;");

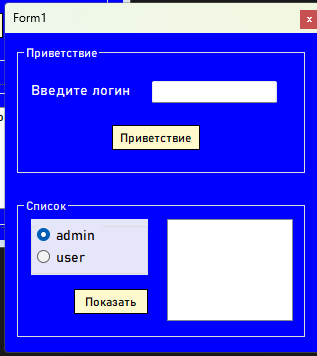
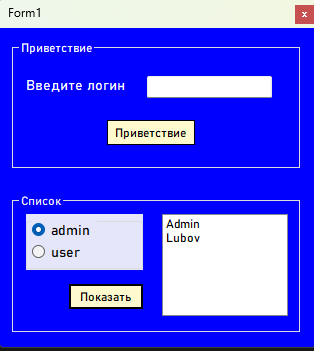
5. Осуществите следующий принцип работы формы: в текстовое поле необходимо ввести логин. По нажатию кнопки в лейбле появляется текст «Привет, Имя\_пользователя». Причем Имя\_пользователя должно соответствовать логину.

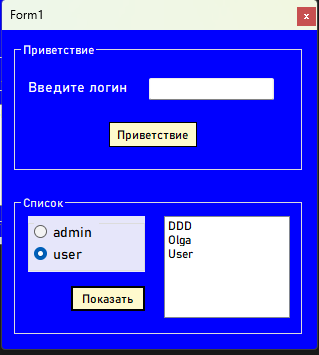


6. Разместите на форме объект listBox и любой элемент, который будет отвечать за выбор роли (textbox, radiobutton, checkbox, combobox и т.д.), а также кнопку.



7. Реализуйте следующий механизм. Пользователь выбирает роль, нажимает кнопку в лист боксе появляются имена пользователей, соответствующие этой роли.



**Код:**

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

using ClassLibraryDataBase;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

DataBase dataBase=new DataBase();

public Form1()

{

InitializeComponent();

}

private void buttonHello\_Click(object sender, EventArgs e)

{

if (textBoxLog.Text != string.Empty)

{

//запрос который будет выполнен

string query = "SELECT Name FROM Users where login = @l";

SqlParameter p1 = new SqlParameter("@l", textBoxLog.Text);

//связь с sql передача туда запроса и строки для соединения

SqlCommand command = new SqlCommand(query, dataBase.GetConection());

command.Parameters.Add(p1);

dataBase.OpenConection();//открыть соединение

string b=null;

dataBase.OpenConection();//открыть соединение

try{

b = command.ExecuteScalar().ToString();//возврат 1 значения

MessageBox.Show("Привет, " + b, "Приветствие", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

}

catch (Exception es) {MessageBox.Show("Пользователя с таким логином не существует", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error); }

dataBase.CloseConection();

}

else MessageBox.Show("Пустое поле login", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void buttonSpisok\_Click(object sender, EventArgs e)

{

Spisok.Items.Clear();//очистка списка

//запрос который будет выполнен

string query = "SELECT Name FROM Users where role = @l";

SqlParameter p1;

if (radioButtonAd.Checked)

p1 = new SqlParameter("@l", radioButtonAd.Text);

else p1 = new SqlParameter("@l", radioButtonUs.Text);

//связь с sql передача туда запроса и строки для соединения

SqlCommand command = new SqlCommand(query, dataBase.GetConection());

command.Parameters.Add(p1);

dataBase.OpenConection();//открыть соединение

SqlDataReader Reader = command.ExecuteReader();//чтнение в массив

while (Reader.Read()) {Spisok.Items.Add(Reader.GetString(0)); }

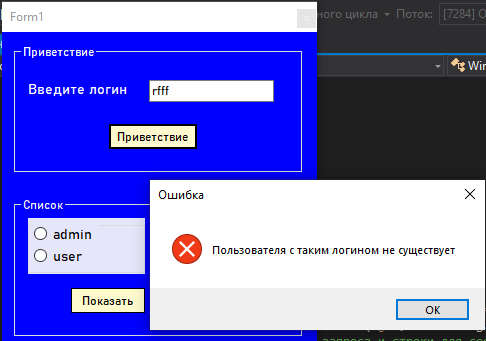
dataBase.CloseConection();

}

}

}

Результат: Если ввести несуществующий логин



# **Лабораторная работа №18**

**Создание окна входа и регистрации пользователя**

**Цель:** Изучение синтаксиса для подключения к базе данных через приложение Windows Forms. Простейший способ отображения данных.

**Задания:**

1. Работайте с таблицей Users из предыдущей ЛР.

2. Создайте форму для входа и регистрации пользователя по описанию из методических указаний, учитывая следующие требования:

• При регистрации по умолчанию пользователю назначается роль User, а данные заносятся в таблицу Users.

• Если пользователь выбрал уже существующий логин, появляется сообщение «Такой логин уже занят, выберите другое имя пользователя»

• Зарегистрироваться с пустыми текстовыми полями невозможно.

• Если пользователь ввел верные данные, появляется Сообщение «Данные введены верно. Добро пожаловать!»

• Если пользователь ввел верный логин, но неверный пароль, то появляется сообщение «Вы ввели неверный пароль».

• Если пользователь ввел неверные данные, то появляется сообщение «Вы ввели неверные данные».

• Текстовое поле с паролем должно скрывать вводимые символы.



**Код:**

**Форма Авторизации:**

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

using ClassLibraryDataBase;

namespace WindowsFormsApp1

{

public partial class Avtorisations : Form

{

//для создания соединения

DataBase dataBase = new DataBase();

public Avtorisations()

{

InitializeComponent();

}

//проверка на существование такой записи в БД

public int VerificationDB(string log)

{

//запрос

string ComDel = " Select count(\*) from Users where login=@code";

//команда на выполенеие запроса через соединение

SqlCommand cmd1 = new SqlCommand(ComDel, dataBase.GetConection());

//запись в параметр значения

SqlParameter pr1 = new SqlParameter("@code", log);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr1);

dataBase.OpenConection();

int k =Convert.ToInt32(cmd1.ExecuteScalar());//кол-во найденных записей

dataBase.CloseConection();

return k;

}

//вход

public string EntranceInDB(string log, string pas)

{

//запрос который будет выполнен

string query = "SELECT Name FROM Users where login = @l and password = @pass";

SqlParameter p1 = new SqlParameter("@l", log);

SqlParameter p2 = new SqlParameter("@pass", pas);

//связь с sql передача туда запроса и строки для соединения

SqlCommand command = new SqlCommand(query, dataBase.GetConection());

command.Parameters.Add(p1);

command.Parameters.Add(p2);

dataBase.OpenConection();//открыть соединение

string b = command.ExecuteScalar()==null?string.Empty: command.ExecuteScalar().ToString();

dataBase.CloseConection();

return b;

}

private void buttonEntrance\_Click(object sender, EventArgs e)

{

if ((textBoxLogin.Text != string.Empty)&&(textBoxPassword.Text!=string.Empty))

{

if (VerificationDB(textBoxLogin.Text) < 1)

{

DialogResult res = MessageBox.Show("Такого логина не существует! " +

"\nЖелаете зарегестрироваться? ", " Предупреждение ",

MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation,

MessageBoxDefaultButton.Button2);

if (res == DialogResult.Yes)

{

FormReg about = new FormReg(textBoxLogin.Text, textBoxPassword.Text);

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

}

}

else

{

string s= EntranceInDB(textBoxLogin.Text, textBoxPassword.Text);

if(s!=string.Empty)

MessageBox.Show("Привет, " + s, "Приветствие", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

else MessageBox.Show("Неверный пароль!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else MessageBox.Show("Пустые поля недопустимы!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void buttonReg\_Click(object sender, EventArgs e)

{

if ((textBoxLogin.Text != string.Empty) && (textBoxPassword.Text != string.Empty))

{

if (VerificationDB(textBoxLogin.Text) == 0)

{

FormReg about = new FormReg(textBoxLogin.Text, textBoxPassword.Text);

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

}

else

{

DialogResult res = MessageBox.Show("Такой логин уже существует! " +

"\nЖелаете войти? ", " Предупреждение",

MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation,

MessageBoxDefaultButton.Button2);

if (res == DialogResult.Yes)

{

string s = EntranceInDB(textBoxLogin.Text, textBoxPassword.Text);

if (s != string.Empty)

MessageBox.Show("Привет, " + s, "Приветствие", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

else MessageBox.Show("Вы ввели неверные данные!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

}

}

**Форма Регистрации:**

using ClassLibraryDataBase;

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class FormReg : Form

{

DataBase dataBase=new DataBase();

public FormReg(string log, string pass)

{

InitializeComponent();

//запрос на получение ролей и записи их в выпадающий список

string query = "SELECT Name FROM Rols";

//связь с sql передача туда запроса и строки для соединения

SqlCommand command = new SqlCommand(query, dataBase.GetConection());

dataBase.OpenConection();//открыть соединение

SqlDataReader Reader = command.ExecuteReader();//чтнение в массив

while (Reader.Read()) { comboBoxRole.Items.Add(Reader[0]); }

dataBase.CloseConection();

//начальный элемент - user

comboBoxRole.SelectedIndex=1;

//ограничение длины вводимых величин

textBoxName.MaxLength = 50;

textBoxLogin.MaxLength = 50;

textBoxPassword.MaxLength = 50;

//перенос данных

textBoxPassword.Text = pass;

textBoxLogin.Text = log;

}

private void buttonReg\_Click(object sender, EventArgs e)

{

if ((textBoxName.Text != String.Empty) && (textBoxLogin.Text != String.Empty) && (textBoxPassword.Text != String.Empty))

{

//запрос

string ComDel = " Insert into Users values (@name, @log, @role, @pass)";

//команда на выполенеие запроса через соединение

SqlCommand cmd1 = new SqlCommand(ComDel, dataBase.GetConection());

//запись в параметр значения

SqlParameter pr2 = new SqlParameter("@name", textBoxName.Text);

SqlParameter pr3 = new SqlParameter("@log", textBoxLogin.Text);

SqlParameter pr4 = new SqlParameter("@pass", textBoxPassword.Text);

SqlParameter pr5 = new SqlParameter("@role", comboBoxRole.Text);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr2);

cmd1.Parameters.Add(pr3);

cmd1.Parameters.Add(pr4);

cmd1.Parameters.Add(pr5);

dataBase.OpenConection();

if(cmd1.ExecuteNonQuery()!=0)

MessageBox.Show("Пользователь успешно зарегистрирован!", "Сообщение", MessageBoxButtons.OK, MessageBoxIcon.Information);

else MessageBox.Show("Что-то пошло не так!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

dataBase.CloseConection();

}

else MessageBox.Show("Пустые поля недопустимы!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

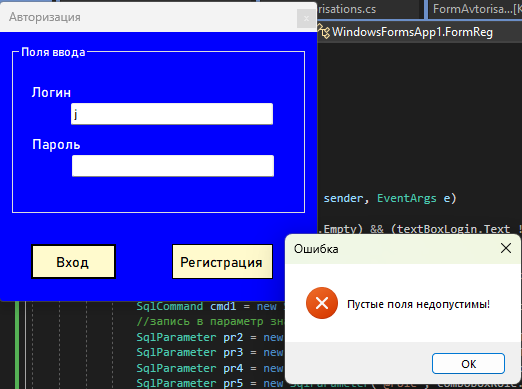
}

}

}

**Результат:**

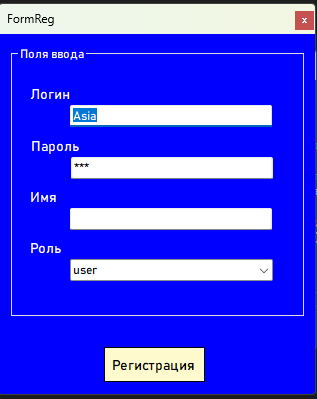
Проверка на пустые поля



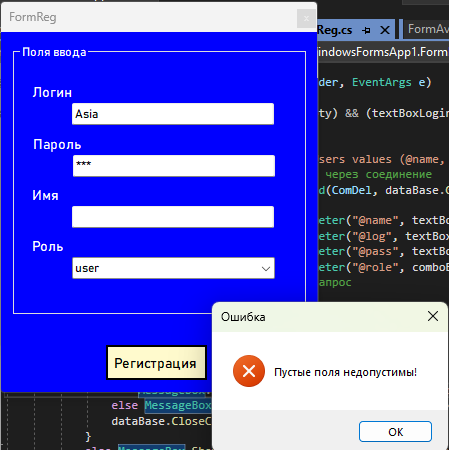
Несуществующий логин и пароль



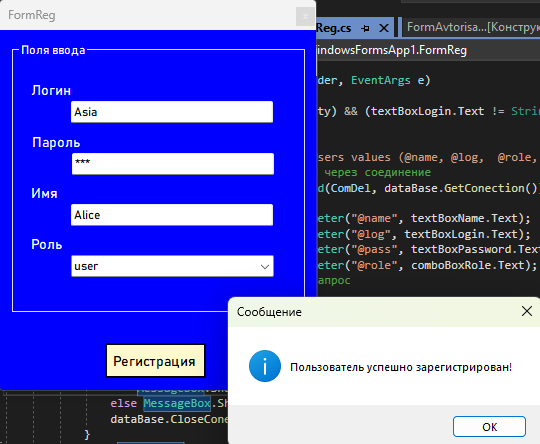
При выборе «нет» ничего не происходит, а если «да», то происходит переход на форму регистрации



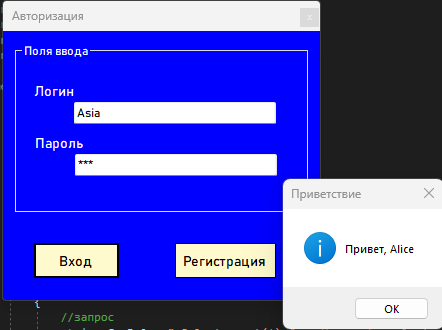
Если не заполнить поле Имя, то выскачет ошибка пустых полей



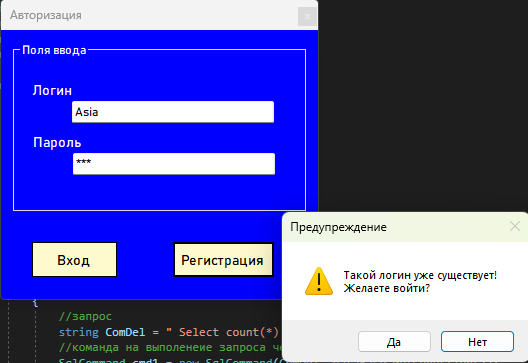
при заполненном варианте



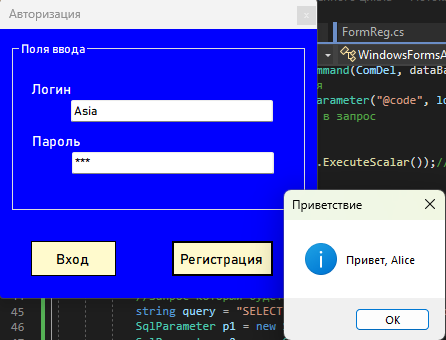
Вход с правильными данными



А теперь попытаемся снова зарегистрироваться

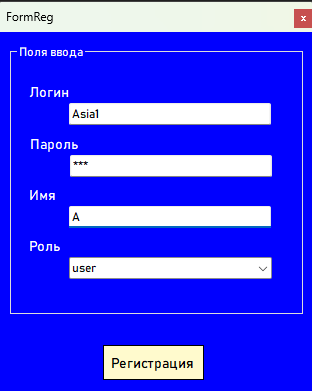


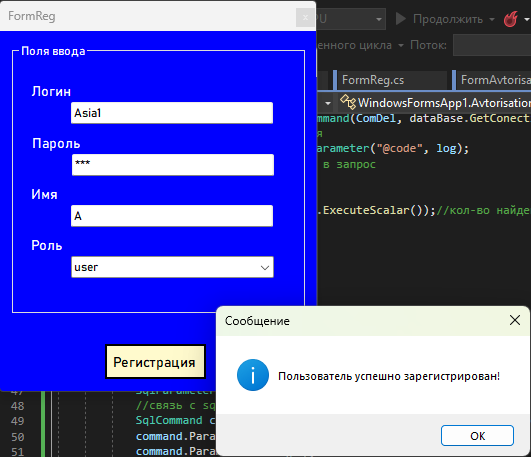
Да



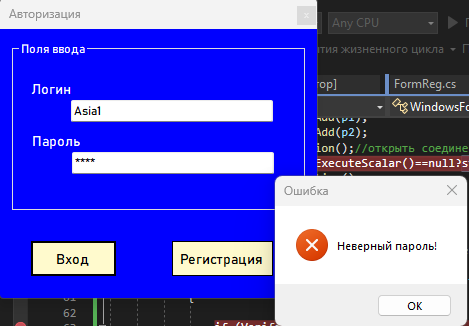
Регистрация с другим логином







Вход с неверным паролем



# **Лабораторная работа №19**

**Создание приложения с опросом**

**Цель:** Вспомнить способы обработки данных с различных элементов формы. Научиться выбирать удобный способ для сбора и обработки информации, а также осуществлять его.

**Задания:**

1. Создайте приложение Windows Forms.

2. В нем создайте приветственную форму. Именно на этом этапе наш пользователь будет выбирать хочет он пройти опрос или нет. В этой форме должны быть две кнопки, по нажатию одной их них, мы переходим в новую форму с опросом, по нажатию второй – форма закрывается. Размер формы должен быть фиксированным, чтобы пользователь не мог менять его сам. Стиль оформления свободный (Рис).

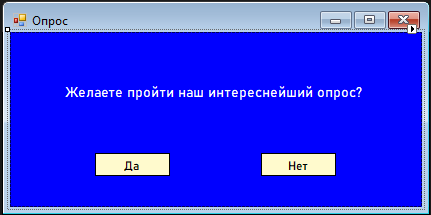


Рис.19 Пример окна формы

3. Создайте вторую форму, на которую перейдет наш пользователь, чтобы пройти опрос. В этой форме будет вопрос о возрасте, оформленный в виде выпадающего списка. Выбор ответа должен быть строго из представленных в списке, невозможно ввести свой вариант. Создайте кнопку «Далее», после нажатия которой будет открываться не новая форма, а в этой же форме появится новый вопрос (Рис.20).

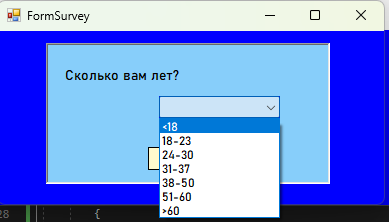


Рис.20 Выпадающий список

4. Следующий вопрос «Любите ли Вы гулять?» должен быть реализован с помощью выбора из двух вариантов «Да/Нет». Также должна присутствовать кнопка «Далее», после нажатия которой мы все еще остаемся в старой форме, но при этом появится новый вопрос (Рис.21).

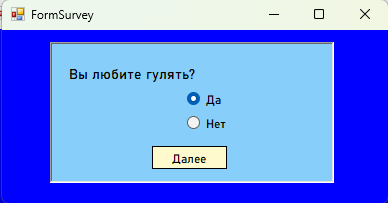


Рис.21 Новый вопрос

5. Если человек ответил да, то его отправляют на новый вопрос, если нет, то выскакивает окно с сообщением «Спасибо, опрос окончен» (Рис.22).

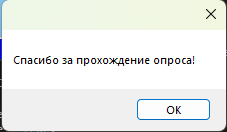


Рис.22

6. Новый и последний опрос совершается с помощью множественного выбора любимых мест для прогулки. Также должна быть кнопка «Отправить!», после нажатия которой форма закрывается, а все данные отправляются в БД (Рис.23).

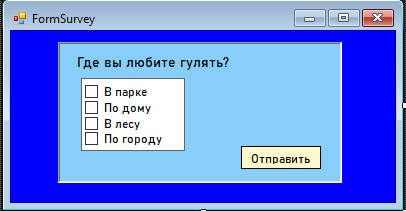
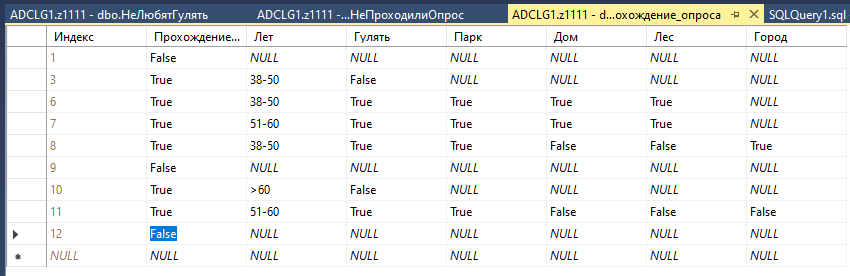


Рис.23

7. Создайте БД, в которую будут отправляться результаты опроса.



8. Подумайте, как лучше и целесообразнее хранить данные (В одной таблице или нескольких, записать ответ или подсчитывать количество того или иного выбора и т.д.)

9. Создайте следующие представления:

• Сколько человек отказалось от прохождения опроса.

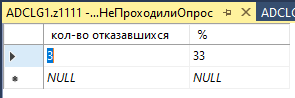
alter view НеПроходилиОпрос

As

select count (\*) as [кол-во отказавшихся], (100\*(select count (\*)from Прохождение\_опроса where [Прохождение опроса] = 0) /(select count (\*) from Прохождение\_опроса))as [%]

from Прохождение\_опроса

where [Прохождение опроса] = 0;



• Какой процент всех опрашиваемых не любит гулять.

SELECT COUNT(\*) AS [кол-во не любящих гулять], 100 \*

(SELECT COUNT(\*) AS Expr1

FROM dbo.Прохождение\_опроса

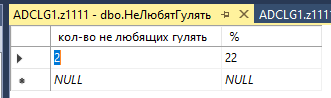
WHERE (Гулять = 0)) /

(SELECT COUNT(\*) AS Expr1

FROM dbo.Прохождение\_опроса) AS [%]

FROM dbo.Прохождение\_опроса

WHERE (Гулять = 0)



• Сколько человек выбрало то или иного место для прогулки, в процентах (Например, в парке 20%, на улице 10% и т.д.).

alter view ГдеГулять

As

select (100\*count (\*) /(select count (\*) from Прохождение\_опроса))as [% в парке],

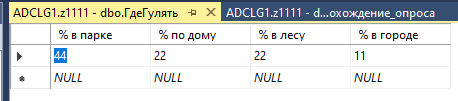
(100\*(select count (\*)from Прохождение\_опроса where [Дом] = 1) /(select count (\*) from Прохождение\_опроса))as [% по дому],

(100\*(select count (\*)from Прохождение\_опроса where [Лес] = 1) /(select count (\*) from Прохождение\_опроса))as [% в лесу],

(100\*(select count (\*)from Прохождение\_опроса where [Город] = 1) /(select count (\*) from Прохождение\_опроса))as [% в городе]

from Прохождение\_опроса

where [Парк] = 1;



• Сколько людей предпочитает загородный отдых, а сколько – городской, в процентах. Будем считать категории «В парке», «На улице», «По дому» городскими, а остальные загородными.

create view ГородПрирода

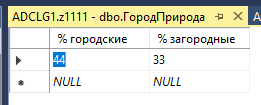
As

select (100\*count (\*) /(select count (\*) from Прохождение\_опроса))as [% городские],

(100\*(select count (\*)from Прохождение\_опроса where [Дом] = 1 or [Город]=1) /(select count (\*) from Прохождение\_опроса))as [% загородные]

from Прохождение\_опроса

where [Парк] = 1 or [Лес]=1;



Примечание: вы можете сделать опрос со своими вопросами, но ваша форма должна содержать те же элементы формы, что и в примерах выше.

Код:

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

using ClassLibraryDataBase;

namespace WindowsFormsApp1

{

public partial class FormSurvey : Form

{

DataBase dataBase = new DataBase();

public FormSurvey()

{

InitializeComponent();

}

public override bool Equals(object obj)

{

if (GetType() == obj.GetType())

{

string a = this.ToString();

string b = obj.ToString();

for (int i = 0; i < a.Length; i++)

{

if (a[i] != b[i]) return false;

}

return true;

}

else return false;

}

public override int GetHashCode()

{

return base.GetHashCode();

}

private void Z2(string s, byte b)

{

//запрос

string ComDel = " Insert into [dbo].[Прохождение\_опроса]([Прохождение опроса], Лет, Гулять, Парк, Дом, Лес, Город) values (1, @year , @code, @1, @2, @3, @4)";

//команда на выполенеие запроса через соединение

SqlCommand cmd1 = new SqlCommand(ComDel, dataBase.GetConection());

//запись в параметр значения

SqlParameter pr1 = new SqlParameter("@year", s);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr1);

SqlParameter pr2 = new SqlParameter("@code", b);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr2);

SqlParameter pr3 = new SqlParameter("@1", checkBox1.Checked);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr3);

SqlParameter pr4 = new SqlParameter("@2", checkBox2.Checked);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr4);

SqlParameter pr5 = new SqlParameter("@3", checkBox3.Checked);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr5);

SqlParameter pr6 = new SqlParameter("@4", checkBox4.Checked);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr6);

dataBase.OpenConection();

cmd1.ExecuteNonQuery();

dataBase.CloseConection();

}

private void Z1(string s, byte b)

{

//запрос

string ComDel = " Insert into [dbo].[Прохождение\_опроса]([Прохождение опроса], Лет, Гулять) values (1, @year , @code)";

//команда на выполенеие запроса через соединение

SqlCommand cmd1 = new SqlCommand(ComDel, dataBase.GetConection());

//запись в параметр значения

SqlParameter pr1 = new SqlParameter("@year", s);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr1);

SqlParameter pr2 = new SqlParameter("@code", b);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr2);

dataBase.OpenConection();

cmd1.ExecuteNonQuery();

dataBase.CloseConection();

}

private void buttonYes\_Click(object sender, EventArgs e)

{

panel1.Visible= false;

panel2.Visible = true;

radioButtonY.Checked = true;

}

private void buttonYes2\_Click(object sender, EventArgs e)

{

if (radioButtonY.Checked)

{

panel2.Visible = false;

panel3.Visible = true;

}

else

{

DialogResult res = MessageBox.Show("Спасибо за прохождение опроса!",

"", MessageBoxButtons.OK, MessageBoxIcon.None);

if (res== DialogResult.OK) { Z1(comboBox1.Text,0);

Close(); }

}

}

private void buttonYes3\_Click(object sender, EventArgs e)

{

Z2(comboBox1.Text, 1);

Close();

}

}

}

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

using ClassLibraryDataBase;

namespace WindowsFormsApp1

{

public partial class FormQ : Form

{

DataBase dataBase = new DataBase();

public FormQ()

{

InitializeComponent();

}

private void Z(byte b)

{

//запрос

string ComDel = " Insert into [dbo].[Прохождение\_опроса]([Прохождение опроса]) values (@code)";

//команда на выполенеие запроса через соединение

SqlCommand cmd1 = new SqlCommand(ComDel, dataBase.GetConection());

//запись в параметр значения

SqlParameter pr1 = new SqlParameter("@code", b);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr1);

dataBase.OpenConection();

cmd1.ExecuteNonQuery();

dataBase.CloseConection();

}

private void buttonNo\_Click(object sender, EventArgs e)

{

DialogResult res = MessageBox.Show("Вы уверены, что не хотите пройти наш опрос?",

"Предупреждение", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (res == DialogResult.Yes) {

Z(0);

Close();}

}

private void buttonYes\_Click(object sender, EventArgs e)

{

FormSurvey about = new FormSurvey();

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

this.Close();

}

}

}

# **Лабораторная работа №20**

**Создание интерфейса администратора**

**Цель:** Усовершенствовать работу номер 18, добавив к ней личный кабинет администратора. Научиться создавать инструменты работы с БД для администратора.

**Задания:**

1. Возьмите за основу вашу БД из ЛР 15 и форму входу из ЛР 18.

2. Реализуйте следующее действие: пользователь, чья роль при входе идентифицируется, как администратор, может иметь доступ ко всем таблицам вашей БД.

3. Самостоятельно выберите способ отображения таблиц. Все на одной форме, на разных формах, панелях или с помощью TabControl.

4. Помимо просмотра, он может добавлять изменять и удалять любые строки из БД.

5. Проверьте каждое из этих действий. Убедитесь, что изменения происходят не только в вашей форме, но и в базе данных.

6. Для обычного пользователя реализуйте просмотр всех данных.

Код:

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

using ClassLibraryDataBase;

namespace WindowsFormsApp1

{

public partial class FormAvtorisations : Form

{

public FormAvtorisations()

{

InitializeComponent();

}

//для создания соединения

public static DataBase dataBase = new DataBase();

//какая таблица активна

public static string table = "Users";

//кол-во полей активной таблицы

public static byte table\_col = 0;

//текущая роль пользователя

public static string rol = "";

//проверка на существование такой записи в БД

public int VerificationDB(string log)

{

//запрос

string ComDel = " Select count(\*) from Users where login=@code";

//команда на выполенеие запроса через соединение

SqlCommand cmd1 = new SqlCommand(ComDel, dataBase.GetConection());

//запись в параметр значения

SqlParameter pr1 = new SqlParameter("@code", log);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr1);

dataBase.OpenConection();

int k = Convert.ToInt32(cmd1.ExecuteScalar());//кол-во найденных записей

dataBase.CloseConection();

return k;

}

//вход

public string EntranceInDB(string log, string pas)

{

//запрос который будет выполнен

string query = "SELECT Name FROM Users where login = @l and password = @pass";

SqlParameter p1 = new SqlParameter("@l", log);

SqlParameter p2 = new SqlParameter("@pass", pas);

//связь с sql передача туда запроса и строки для соединения

SqlCommand command = new SqlCommand(query, dataBase.GetConection());

command.Parameters.Add(p1);

command.Parameters.Add(p2);

dataBase.OpenConection();//открыть соединение

string b = command.ExecuteScalar() == null ? string.Empty : command.ExecuteScalar().ToString();

dataBase.CloseConection();

return b;

}

//роль

public string UserRol(string log)

{

//запрос который будет выполнен

string query = "SELECT role FROM Users where login = @l";

SqlParameter p1 = new SqlParameter("@l", log);

//связь с sql передача туда запроса и строки для соединения

SqlCommand command = new SqlCommand(query, dataBase.GetConection());

command.Parameters.Add(p1);

dataBase.OpenConection();//открыть соединение

string b = command.ExecuteScalar().ToString();

dataBase.CloseConection();

return b;

}

//вход

private void buttonEntrance\_Click(object sender, EventArgs e)

{

if ((textBoxLogin.Text != string.Empty) && (textBoxPassword.Text != string.Empty))

{

if (VerificationDB(textBoxLogin.Text) < 1)

{

DialogResult res = MessageBox.Show("Такого логина не существует! " +

"\nЖелаете зарегестрироваться? ", " Предупреждение ",

MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation,

MessageBoxDefaultButton.Button2);

if (res == DialogResult.Yes)

{

FormReg about = new FormReg(textBoxLogin.Text, textBoxPassword.Text);

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

}

}

else

{

string s = EntranceInDB(textBoxLogin.Text, textBoxPassword.Text);

if (s != string.Empty)

{

MessageBox.Show("Привет, " + s, "Приветствие", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

//роль текущего пользователя

rol = UserRol(textBoxLogin.Text);

Form ifrm = new FormStudent();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

else MessageBox.Show("Неверный пароль!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else MessageBox.Show("Пустые поля недопустимы!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

//регистрация

private void buttonReg\_Click(object sender, EventArgs e)

{

if ((textBoxLogin.Text != string.Empty) && (textBoxPassword.Text != string.Empty))

{

if (VerificationDB(textBoxLogin.Text) == 0)

{

Form ifrm = new FormReg(textBoxLogin.Text, textBoxPassword.Text);

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

else

{

DialogResult res = MessageBox.Show("Такой логин уже существует! " +

"\nЖелаете войти? ", " Предупреждение",

MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation,

MessageBoxDefaultButton.Button2);

if (res == DialogResult.Yes)

{

string s = EntranceInDB(textBoxLogin.Text, textBoxPassword.Text);

if (s != string.Empty)

MessageBox.Show("Привет, " + s, "Приветствие", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

else MessageBox.Show("Вы ввели неверные данные!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

private void FormAvtorisations\_FormClosed(object sender, FormClosedEventArgs e)

{

if (Application.OpenForms.Count > 0)

{

// вызываем главную форму, которая открыла текущую, главная форма всегда = 0 - [0]

Form ifrm = Application.OpenForms[0];

ifrm.StartPosition = FormStartPosition.Manual; // меняем параметр StartPosition у Form1, иначе она будет использовать тот, который у неё прописан в настройках и всегда будет открываться по центру экрана

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form1

}

}

}

}

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class FormUpDate : Form

{

string ComDel;

public FormUpDate(string s)

{

InitializeComponent();

if (FormAvtorisations.table\_col == 2)

{

groupBoxUpDate1.Visible = true;

groupBoxUpDate7.Visible = false;

groupBoxUpDate6.Visible = false;

}

else if (FormAvtorisations.table\_col == 6)//mark

{

groupBoxUpDate1.Visible = false;

groupBoxUpDate7.Visible = false;

groupBoxUpDate6.Visible = true;

}

else

{

groupBoxUpDate1.Visible = false;

groupBoxUpDate6.Visible = false;

groupBoxUpDate7.Visible = true;

}

ComDel = "Update " + s;

}

private void InsertOne()

{

if (textBoxNameUp.Text != String.Empty)

{

//команда на выполенеие запроса через соединение

SqlCommand cmd1 = new SqlCommand(ComDel, FormAvtorisations.dataBase.GetConection());

//запись в параметр значения

SqlParameter pr2 = new SqlParameter("@name", textBoxNameUp.Text);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr2);

SqlParameter pr3 = new SqlParameter("@id", numericUpDownUpId.Text);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr3);

FormAvtorisations.dataBase.OpenConection();

if (cmd1.ExecuteNonQuery() < 1)

MessageBox.Show("Такого id не существует", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

else

MessageBox.Show("Запись успешно обновлена!", "Обновление", MessageBoxButtons.OK, MessageBoxIcon.Information);

FormAvtorisations.dataBase.CloseConection();

}

else MessageBox.Show("Поля не могут быть пустыми!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void InsertSix()//mark

{

if (textBoxTeach.Text != String.Empty)

{

SqlCommand cmd1 = new SqlCommand(ComDel, FormAvtorisations.dataBase.GetConection());

//запись в параметр значения

SqlParameter pr2 = new SqlParameter("@zach", numericUpDownZach6.Value);

SqlParameter pr3 = new SqlParameter("@sem", numericUpDownSem.Value);

SqlParameter pr4 = new SqlParameter("@obj", numericUpDownObj.Value);

SqlParameter pr5 = new SqlParameter("@mark", numericUpDownMark.Value);

SqlParameter pr6 = new SqlParameter("@data", dateTimePickerDataEx.Value);

SqlParameter pr7 = new SqlParameter("@teach", textBoxTeach.Text);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr2);

cmd1.Parameters.Add(pr3);

cmd1.Parameters.Add(pr4);

cmd1.Parameters.Add(pr5);

cmd1.Parameters.Add(pr6);

cmd1.Parameters.Add(pr7);

FormAvtorisations.dataBase.OpenConection();

if (cmd1.ExecuteNonQuery() < 1)

MessageBox.Show("Такого id не существует", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

else

MessageBox.Show("Запись успешно обновлена!", "Обновление", MessageBoxButtons.OK, MessageBoxIcon.Information);

FormAvtorisations.dataBase.CloseConection();

FormAvtorisations.dataBase.CloseConection();

}

else MessageBox.Show("Поля не могут быть пустыми!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void InsertSeven()//student

{

if (textBoxName6.Text != String.Empty)

{

SqlCommand cmd1 = new SqlCommand(ComDel, FormAvtorisations.dataBase.GetConection());

//запись в параметр значения

SqlParameter pr1 = new SqlParameter("@id", numericUpDownZach7.Value);

SqlParameter pr2 = new SqlParameter("@name", textBoxName6.Text);

SqlParameter pr3 = new SqlParameter("@data", dateTimePickerData.Value);

SqlParameter pr4 = new SqlParameter("@fac", numericUpDownFac.Value);

SqlParameter pr5 = new SqlParameter("@spec", numericUpDownSpec.Value);

SqlParameter pr6 = new SqlParameter("@curs", numericUpDownCurs.Value);

SqlParameter pr7 = new SqlParameter("@group", numericUpDownGroup.Value);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr1);

cmd1.Parameters.Add(pr2);

cmd1.Parameters.Add(pr3);

cmd1.Parameters.Add(pr4);

cmd1.Parameters.Add(pr5);

cmd1.Parameters.Add(pr6);

cmd1.Parameters.Add(pr7);

FormAvtorisations.dataBase.OpenConection();

if (cmd1.ExecuteNonQuery() < 1)

MessageBox.Show("Такого id не существует", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

else

MessageBox.Show("Запись успешно обновлена!", "Обновление", MessageBoxButtons.OK, MessageBoxIcon.Information);

FormAvtorisations.dataBase.CloseConection();

FormAvtorisations.dataBase.CloseConection();

}

else MessageBox.Show("Поля не могут быть пустыми!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void buttonUpDate\_Click(object sender, EventArgs e)

{

if (FormAvtorisations.table\_col == 2)

InsertOne();

else if (FormAvtorisations.table\_col == 6)//mark

InsertSix();

else InsertSeven();//student

}

}

}

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class FormInsert : Form

{

string ComDel;

public FormInsert(string s)

{

InitializeComponent();

if (FormAvtorisations.table\_col == 2)

{

groupBoxInsert7.Visible = false;

groupBoxInsert6.Visible = false;

groupBoxInsert1.Visible = true;

}

else if (FormAvtorisations.table\_col == 6)//mark

{

groupBoxInsert1.Visible = false;

groupBoxInsert7.Visible = false;

groupBoxInsert6.Visible = true;

}

else if (FormAvtorisations.table\_col == 7)//mark

{

groupBoxInsert1.Visible = false;

groupBoxInsert6.Visible = false;

groupBoxInsert7.Visible = true;

}

ComDel = "Insert into " + s;

}

private void InsertOne()

{

if (textBoxNameIns.Text != String.Empty)

{

//команда на выполенеие запроса через соединение

SqlCommand cmd1 = new SqlCommand(ComDel, FormAvtorisations.dataBase.GetConection());

//запись в параметр значения

SqlParameter pr2 = new SqlParameter("@name", textBoxNameIns.Text);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr2);

FormAvtorisations.dataBase.OpenConection();

cmd1.ExecuteNonQuery();

MessageBox.Show("Запись успешно добавлена!", "Добавление", MessageBoxButtons.OK, MessageBoxIcon.Information);

FormAvtorisations.dataBase.CloseConection();

}

else MessageBox.Show("Единственное поле не может быть пустым!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void InsertSix()//mark

{

if (textBoxTeach.Text != String.Empty)

{

SqlCommand cmd1 = new SqlCommand(ComDel, FormAvtorisations.dataBase.GetConection());

//запись в параметр значения

SqlParameter pr2 = new SqlParameter("@zach", numericUpDownZach.Value);

SqlParameter pr3 = new SqlParameter("@sem", numericUpDownSem.Value);

SqlParameter pr4 = new SqlParameter("@obj", numericUpDownObj.Value);

SqlParameter pr5 = new SqlParameter("@mark", numericUpDownMark.Value);

SqlParameter pr6 = new SqlParameter("@data", dateTimePickerDataEx.Value);

SqlParameter pr7 = new SqlParameter("@teach", textBoxTeach.Text);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr2);

cmd1.Parameters.Add(pr3);

cmd1.Parameters.Add(pr4);

cmd1.Parameters.Add(pr5);

cmd1.Parameters.Add(pr6);

cmd1.Parameters.Add(pr7);

FormAvtorisations.dataBase.OpenConection();

cmd1.ExecuteNonQuery();

MessageBox.Show("Запись успешно добавлена!", "Добавление", MessageBoxButtons.OK, MessageBoxIcon.Information);

FormAvtorisations.dataBase.CloseConection();

FormAvtorisations.dataBase.CloseConection();

}

else MessageBox.Show("Поля не могут быть пустыми!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void InsertSeven()//student

{

if (textBoxName6.Text != String.Empty)

{

SqlCommand cmd1 = new SqlCommand(ComDel, FormAvtorisations.dataBase.GetConection());

//запись в параметр значения

SqlParameter pr2 = new SqlParameter("@name", textBoxName6.Text);

SqlParameter pr3 = new SqlParameter("@data", dateTimePickerData.Value);

SqlParameter pr4 = new SqlParameter("@fac", numericUpDownFac.Value);

SqlParameter pr5 = new SqlParameter("@spec", numericUpDownSpec.Value);

SqlParameter pr6 = new SqlParameter("@curs", numericUpDownCurs.Value);

SqlParameter pr7 = new SqlParameter("@group", numericUpDownGroup.Value);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr2);

cmd1.Parameters.Add(pr3);

cmd1.Parameters.Add(pr4);

cmd1.Parameters.Add(pr5);

cmd1.Parameters.Add(pr6);

cmd1.Parameters.Add(pr7);

FormAvtorisations.dataBase.OpenConection();

cmd1.ExecuteNonQuery();

MessageBox.Show("Запись успешно добавлена!", "Добавление", MessageBoxButtons.OK, MessageBoxIcon.Information);

FormAvtorisations.dataBase.CloseConection();

FormAvtorisations.dataBase.CloseConection();

}

else MessageBox.Show("Поля не могут быть пустыми!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void buttonInsert\_Click(object sender, EventArgs e)

{

if (FormAvtorisations.table\_col==2)

InsertOne();

else if (FormAvtorisations.table\_col == 6)//mark

InsertSix();

else InsertSeven();//student

}

}

}

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class FormDelete : Form

{

string ComDel;

public FormDelete(string s)

{

InitializeComponent();

ComDel = " Delete from " + s;

if (FormAvtorisations.table == "Успеваемость")

{

groupBoxDel1.Visible = false;

groupBoxDelMark.Visible = true;

}

else { groupBoxDel1.Visible = true; groupBoxDelMark.Visible = false; }

}

private void buttonDel\_Click(object sender, EventArgs e)

{

if (FormAvtorisations.table == "Успеваемость")

{

//команда на выполенеие запроса через соединение

SqlCommand cmd1 = new SqlCommand(ComDel, FormAvtorisations.dataBase.GetConection());

//запись в параметр значения

SqlParameter pr1 = new SqlParameter("@code", numericUpDownZach.Value);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr1);

SqlParameter pr2 = new SqlParameter("@table", numericUpDownObj.Value);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr2);

FormAvtorisations.dataBase.OpenConection();

if (cmd1.ExecuteNonQuery() < 1)

MessageBox.Show("Такого id не существует", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

else

MessageBox.Show("Запись успешно удалена!", "Удаление", MessageBoxButtons.OK, MessageBoxIcon.Information);

FormAvtorisations.dataBase.CloseConection();

}

else

{

//команда на выполенеие запроса через соединение

SqlCommand cmd1 = new SqlCommand(ComDel, FormAvtorisations.dataBase.GetConection());

//запись в параметр значения

SqlParameter pr1 = new SqlParameter("@code", numericUpDownId.Value);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr1);

FormAvtorisations.dataBase.OpenConection();

if (cmd1.ExecuteNonQuery() < 1)

MessageBox.Show("Такого id не существует", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

else

MessageBox.Show("Запись успешно удалена!", "Удаление", MessageBoxButtons.OK, MessageBoxIcon.Information);

FormAvtorisations.dataBase.CloseConection();

}

}

}

}

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class FormMark : Form

{

public FormMark()

{

InitializeComponent();

FormAvtorisations.table = "Успеваемость";

FormAvtorisations.table\_col = 6;

}

public void CreateColumns()

{

dataGridView1.ColumnCount = 6;

dataGridView1.Columns[0].HeaderText = "Номер зачетки";

dataGridView1.Columns[1].HeaderText = "Номер семестра";

dataGridView1.Columns[2].HeaderText = "Предмет";

dataGridView1.Columns[3].HeaderText = "Оценка";

dataGridView1.Columns[4].HeaderText = "Дата экзамена";

dataGridView1.Columns[5].HeaderText = "Фамилия преподавателя";

}

public void ReadWriteInTable(DataGridView dg)

{

//запрос который будет выполнен

string query = "SELECT \* FROM Успеваемость";

//связь с sql передача туда запроса и строки для соединения

SqlCommand command = new SqlCommand(query, FormAvtorisations.dataBase.GetConection());

FormAvtorisations.dataBase.OpenConection();//открыть соединение

//объект для чтения из бд

SqlDataReader reader = command.ExecuteReader();

dg.Rows.Clear();//очистка строк

while (reader.Read())

{

//запись прочитанной строки в таблицу

dg.Rows.Add(reader.GetInt32(0), reader.GetInt32(1), reader.GetInt32(2), reader.GetInt32(3), reader.GetDateTime(4), reader.GetString(5) );

}

reader.Close();

FormAvtorisations.dataBase.CloseConection();

}

private void FormMark\_Load(object sender, EventArgs e)

{

CreateColumns();

ReadWriteInTable(dataGridView1);

labelRowsCount.Text += dataGridView1.RowCount;

if (FormAvtorisations.rol == "user")

{

toolStripMenuItemInsert.Visible = false;

toolStripMenuItemUP.Visible = false;

toolStripMenuItemDelete.Visible = false;

}

}

//закрытие

private void FormMark\_FormClosed(object sender, FormClosedEventArgs e)

{

if (Application.OpenForms.Count > 0) {

// вызываем главную форму, которая открыла текущую, главная форма всегда = 0 - [0]

Form ifrm = Application.OpenForms[0];

ifrm.StartPosition = FormStartPosition.Manual; // меняем параметр StartPosition у Form1, иначе она будет использовать тот, который у неё прописан в настройках и всегда будет открываться по центру экрана

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form1

}

//else Close();

}

private void toolStripTextBoxObj\_Click(object sender, EventArgs e)

{

Form ifrm = new FormObject();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void toolStripTextBoxStud\_Click(object sender, EventArgs e)

{

Form ifrm = new FormStudent();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void toolStripTextBoxSpecial\_Click(object sender, EventArgs e)

{

Form ifrm = new FormSpecial();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

//обновление

private void toolStripMenuItemUpDate\_Click(object sender, EventArgs e)

{

ReadWriteInTable(dataGridView1);

}

//удаление

private void toolStripMenuItemDelete\_Click(object sender, EventArgs e)

{

FormDelete about = new FormDelete(FormAvtorisations.table+ " where [Номер\_зачетки]=@code and [Предмет] =@table");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

}

//добавление

private void toolStripMenuItemInsert\_Click(object sender, EventArgs e)

{

FormInsert about = new FormInsert(FormAvtorisations.table + " ([Номер\_зачетки], [Номер\_семестра], [Предмет], [Оценка], [Дата\_экзамена], [Фамилия\_преподавателя]) values (@zach, @sem, @obj, @mark,@data, @teach)");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

}

//изменение

private void toolStripMenuItemUP\_Click(object sender, EventArgs e)

{

FormUpDate about = new FormUpDate(FormAvtorisations.table + " set [Номер\_семестра]=@sem, [Оценка]=@mark, [Дата\_экзамена]=@data, [Фамилия\_преподавателя]=@teach where [Номер\_зачетки] = @zach and [Предмет]= @obj");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

}

}

}

using System;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace WindowsFormsApp1

{

public partial class FormObject : Form

{

public FormObject()

{

InitializeComponent();

FormAvtorisations.table = "";

FormAvtorisations.table\_col = 2;

}

public void CreateColumnsFac(DataGridView dg)

{

dg.ColumnCount = 2;

dg.Columns[0].HeaderText = "Номер факультета";

dg.Columns[1].HeaderText = "Название факультета";

}

public void CreateColumnsObj(DataGridView dg)

{

dg.ColumnCount = 2;

dg.Columns[0].HeaderText = "Номер предмета";

dg.Columns[1].HeaderText = "Название предмета";

}

public void ReadWriteInTable(DataGridView dg)

{

string query = "SELECT \* FROM " + FormAvtorisations.table;

//связь с sql передача туда запроса и строки для соединения

SqlCommand command = new SqlCommand(query, FormAvtorisations.dataBase.GetConection());

FormAvtorisations.dataBase.OpenConection();//открыть соединение

//объект для чтения из бд

SqlDataReader reader = command.ExecuteReader();

dg.Rows.Clear();//очистка строк

while (reader.Read())

{

//запись прочитанной строки в таблицу

dg.Rows.Add(reader.GetInt32(0), reader.GetString(1));

}

reader.Close();

FormAvtorisations.dataBase.CloseConection();

}

private void FormObject\_Load(object sender, EventArgs e)

{

CreateColumnsObj(dataGridView1);

FormAvtorisations.table = "Предмет";

ReadWriteInTable(dataGridView1);

labelRowsCountObj.Text += dataGridView1.RowCount;

CreateColumnsFac(dataGridView2);

FormAvtorisations.table = " Факультет";

ReadWriteInTable(dataGridView2);

labelRowsCountFac.Text += dataGridView2.RowCount;

if (FormAvtorisations.rol == "user")

{

toolStripMenuItemInsert.Visible = false;

toolStripMenuItemUP.Visible = false;

toolStripMenuItemDelete.Visible = false;

}

}

private void toolStripTextBoxStud\_Click(object sender, EventArgs e)

{

Form ifrm = new FormStudent();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void toolStripTextBoxMark\_Click(object sender, EventArgs e)

{

Form ifrm = new FormMark();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void toolStripTextBoxSpecial\_Click(object sender, EventArgs e)

{

Form ifrm = new FormSpecial();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void FormObject\_FormClosed(object sender, FormClosedEventArgs e)

{

if (Application.OpenForms.Count > 0)

{

// вызываем главную форму, которая открыла текущую, главная форма всегда = 0 - [0]

Form ifrm = Application.OpenForms[0];

ifrm.StartPosition = FormStartPosition.Manual; // меняем параметр StartPosition у Form1, иначе она будет использовать тот, который у неё прописан в настройках и всегда будет открываться по центру экрана

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form1

}

}

//обновление

private void toolStripTextBoxUpFac\_Click(object sender, EventArgs e)

{

FormAvtorisations.table = " Факультет";

ReadWriteInTable(dataGridView2);

}

private void toolStripTextBoxUpObj\_Click(object sender, EventArgs e)

{

FormAvtorisations.table = "Предмет";

ReadWriteInTable(dataGridView1);

}

//добавление

private void toolStripTextBoxInsertFac\_Click(object sender, EventArgs e)

{

FormAvtorisations.table = "Факультет";

FormInsert about = new FormInsert(FormAvtorisations.table + " ([Название\_факультета]) values (@name)");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView2);

}

private void toolStripTextBoxInsertObj\_Click(object sender, EventArgs e)

{

FormAvtorisations.table = "Предмет";

FormInsert about = new FormInsert(FormAvtorisations.table + " ([Название\_предмета]) values (@name)");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

}

//изменение

private void toolStripTextBoxUpDateFac\_Click(object sender, EventArgs e)

{

FormAvtorisations.table = "Факультет";

FormUpDate about = new FormUpDate(FormAvtorisations.table + " set [Название\_факультета]=@name where [Номер\_факультета] = @id");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView2);

}

private void toolStripTextBoxUpDateObj\_Click(object sender, EventArgs e)

{

FormAvtorisations.table = "Предмет";

FormUpDate about = new FormUpDate(FormAvtorisations.table + " set [Название\_предмета]=@name where [Номер\_предмета] = @id");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

}

//удаление

private void toolStripTextBoxDelFac\_Click(object sender, EventArgs e)

{

FormAvtorisations.table = "Факультет";

FormDelete about = new FormDelete(FormAvtorisations.table + " where [Номер\_факультета]=@code");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView2);

}

private void toolStripTextBoxDelObj\_Click(object sender, EventArgs e)

{

FormAvtorisations.table = "Предмет";

FormDelete about = new FormDelete(FormAvtorisations.table + " where [Номер\_предмета]=@code");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

}

}

}

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class FormSpecial : Form

{

public FormSpecial()

{

InitializeComponent();

FormAvtorisations.table ="Специальность";

FormAvtorisations.table\_col = 2;

}

public void CreateColumns()

{

dataGridView1.ColumnCount = 2;

dataGridView1.Columns[0].HeaderText = "Код специальности";

dataGridView1.Columns[1].HeaderText = "Название\_специальности";

}

public void ReadWriteInTable(DataGridView dg)

{

//запрос который будет выполнен

string query = "SELECT \* FROM Специальность";

//связь с sql передача туда запроса и строки для соединения

SqlCommand command = new SqlCommand(query, FormAvtorisations.dataBase.GetConection());

FormAvtorisations.dataBase.OpenConection();//открыть соединение

//объект для чтения из бд

SqlDataReader reader = command.ExecuteReader();

dg.Rows.Clear();//очистка строк

while (reader.Read())

{

//запись прочитанной строки в таблицу

dg.Rows.Add(reader.GetInt32(0), reader.GetString(1));

}

reader.Close();

FormAvtorisations.dataBase.CloseConection();

}

private void FormSpecial\_Load(object sender, EventArgs e)

{

CreateColumns();

ReadWriteInTable(dataGridView1);

labelRowsCount.Text += dataGridView1.RowCount;

if (FormAvtorisations.rol == "user")

{

toolStripMenuItemInsert.Visible = false;

toolStripMenuItemUP.Visible = false;

toolStripMenuItemDelete.Visible = false;

}

}

private void FormSpecial\_FormClosed(object sender, FormClosedEventArgs e)

{

if (Application.OpenForms.Count > 0)

{

// вызываем главную форму, которая открыла текущую, главная форма всегда = 0 - [0]

Form ifrm = Application.OpenForms[0];

ifrm.StartPosition = FormStartPosition.Manual; // меняем параметр StartPosition у Form1, иначе она будет использовать тот, который у неё прописан в настройках и всегда будет открываться по центру экрана

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form1

}

}

private void toolStripTextBoxObj\_Click(object sender, EventArgs e)

{

Form ifrm = new FormObject();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void toolStripTextBoxStud\_Click(object sender, EventArgs e)

{

Form ifrm = new FormStudent();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void toolStripTextBoxMark\_Click(object sender, EventArgs e)

{

Form ifrm = new FormMark();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

//обновление

private void toolStripMenuItemUpDate\_Click(object sender, EventArgs e)

{

ReadWriteInTable(dataGridView1);

}

//добавление

private void toolStripMenuItemInsert\_Click(object sender, EventArgs e)

{

FormInsert about = new FormInsert(FormAvtorisations.table + " ([Название\_специальности]) values (@name)");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

}

//удаление

private void toolStripMenuItemDelete\_Click(object sender, EventArgs e)

{

FormDelete about = new FormDelete(FormAvtorisations.table + " where [Код\_специальности]=@code");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

}

//изменение

private void toolStripMenuItemUP\_Click(object sender, EventArgs e)

{

FormUpDate about = new FormUpDate(FormAvtorisations.table + " set [Название\_специальности]=@name where [Код\_специальности] = @id");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

}

}

}

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class FormStudent : Form

{

public FormStudent()

{

InitializeComponent();

FormAvtorisations.table = "Студент";

FormAvtorisations.table\_col = 7;

}

public void CreateColumns()

{

dataGridView1.ColumnCount = 7;

dataGridView1.Columns[0].HeaderText = "Номер зачетки";

dataGridView1.Columns[1].HeaderText = "Фамилия имя отчество";

dataGridView1.Columns[2].HeaderText = "Дата поступления";

dataGridView1.Columns[3].HeaderText = "Факультет";

dataGridView1.Columns[4].HeaderText = "Специальность";

dataGridView1.Columns[5].HeaderText = "Курс";

dataGridView1.Columns[6].HeaderText = "Номер группы";

}

public void ReadWriteInTable(DataGridView dg)

{

//запрос который будет выполнен

string query = "SELECT \* FROM Студент";

//связь с sql передача туда запроса и строки для соединения

SqlCommand command = new SqlCommand(query, FormAvtorisations.dataBase.GetConection());

FormAvtorisations.dataBase.OpenConection();//открыть соединение

//объект для чтения из бд

SqlDataReader reader = command.ExecuteReader();

dg.Rows.Clear();//очистка строк

while (reader.Read())

{

//запись прочитанной строки в таблицу

dg.Rows.Add(reader.GetInt32(0), reader.GetString(1), reader.GetDateTime(2), reader.GetInt32(3), reader.GetInt32(4), reader.GetInt32(5), reader.GetInt32(6));

}

reader.Close();

FormAvtorisations.dataBase.CloseConection();

}

private void FormStudent\_Load(object sender, EventArgs e)

{

CreateColumns();

ReadWriteInTable(dataGridView1);

labelRowsCount.Text += dataGridView1.RowCount;

if(FormAvtorisations.rol == "user")

{

toolStripMenuItemInsert.Visible = false;

toolStripMenuItemUP.Visible = false;

toolStripMenuItemDelete.Visible = false;

}

}

private void toolStripTextBoxMark\_Click(object sender, EventArgs e)

{

Form ifrm = new FormMark();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void toolStripTextBoxObj\_Click(object sender, EventArgs e)

{

Form ifrm = new FormObject();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void toolStripTextBoxSpecial\_Click(object sender, EventArgs e)

{

Form ifrm = new FormSpecial();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void FormStudent\_FormClosed(object sender, FormClosedEventArgs e)

{

if (Application.OpenForms.Count > 0)

{

// вызываем главную форму, которая открыла текущую, главная форма всегда = 0 - [0]

Form ifrm = Application.OpenForms[0];

ifrm.StartPosition = FormStartPosition.Manual; // меняем параметр StartPosition у Form1, иначе она будет использовать тот, который у неё прописан в настройках и всегда будет открываться по центру экрана

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form1

}

}

//обновление

private void toolStripMenuItemUpDate\_Click(object sender, EventArgs e)

{

ReadWriteInTable(dataGridView1);

}

//добавление

private void toolStripMenuItemInsert\_Click(object sender, EventArgs e)

{

FormInsert about = new FormInsert(FormAvtorisations.table + " ([Фамилия\_имя\_отчество], [Дата\_поступления], [Факультет], [Специальность], [Курс], [Номер\_группы]) values (@name, @data, @fac, @spec,@curs, @group)");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

}

//удаление

private void toolStripMenuItemDelete\_Click(object sender, EventArgs e)

{

FormDelete about = new FormDelete(FormAvtorisations.table + " where [Номер\_зачетки]=@code");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

}

//изменение

private void toolStripMenuItemUP\_Click(object sender, EventArgs e)

{

FormUpDate about = new FormUpDate(FormAvtorisations.table + " set [Фамилия\_имя\_отчество]=@name, [Дата\_поступления]=@data, [Факультет]=@fac, [Специальность]=@spec, [Курс]=@curs, [Номер\_группы]=@group where [Номер\_зачетки] = @id");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

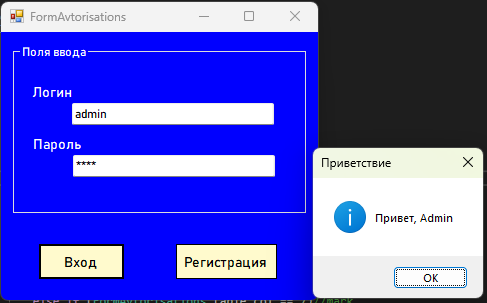
ReadWriteInTable(dataGridView1);

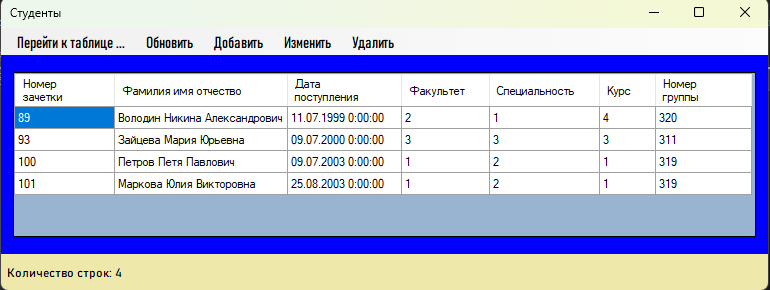
}

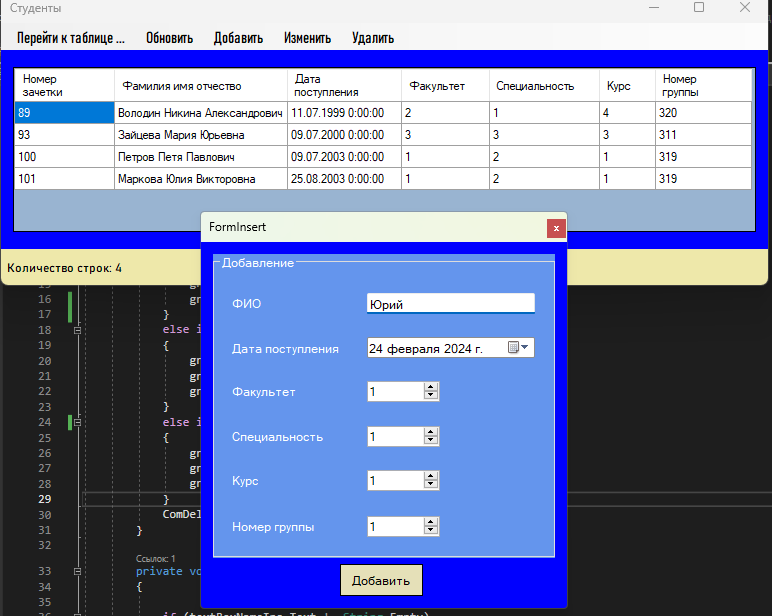
}

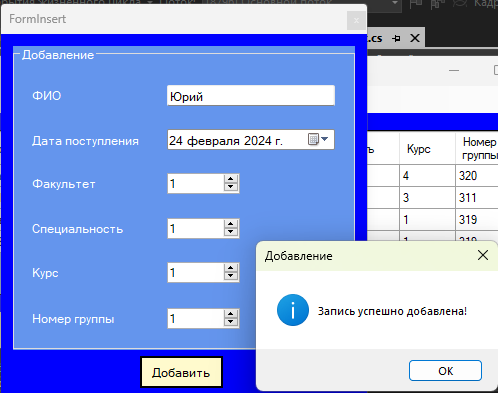
}

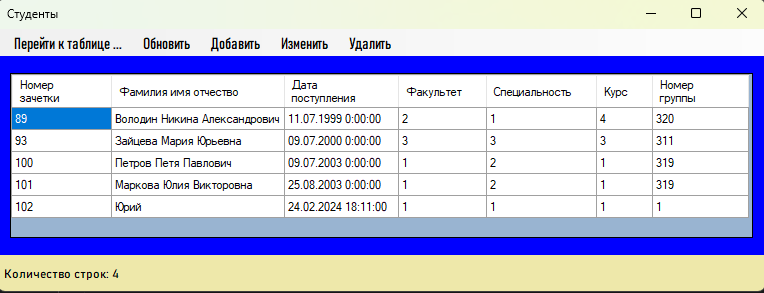
Результат:

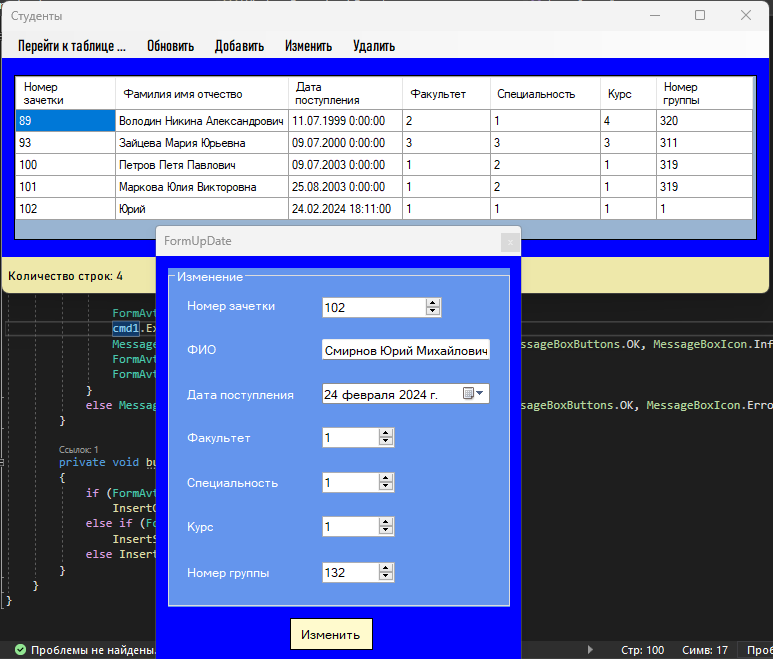


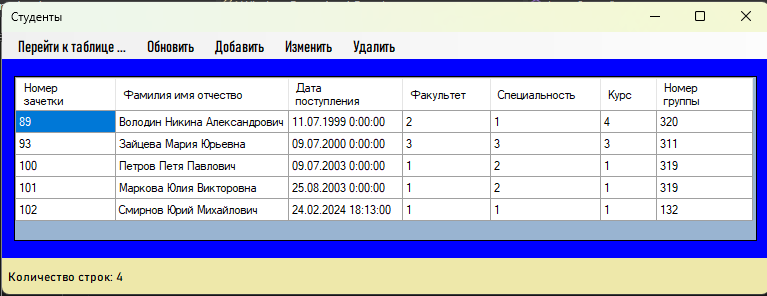


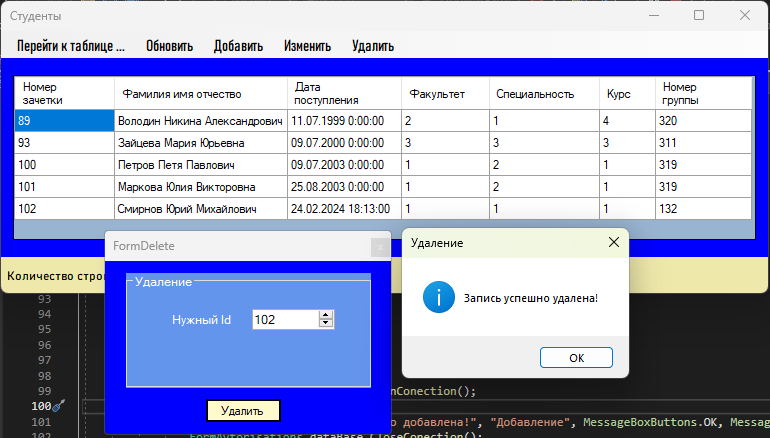


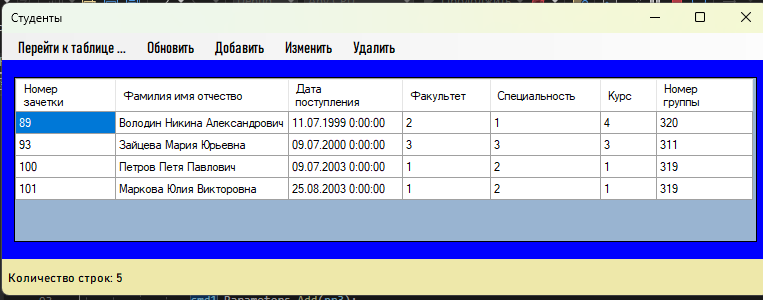


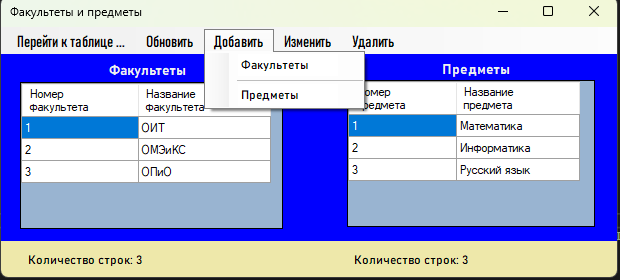


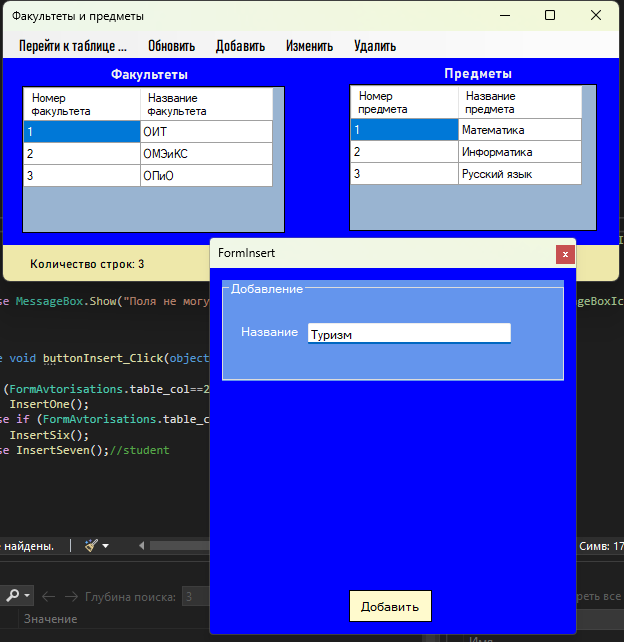


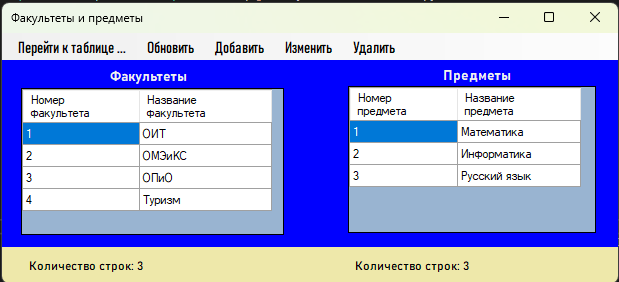


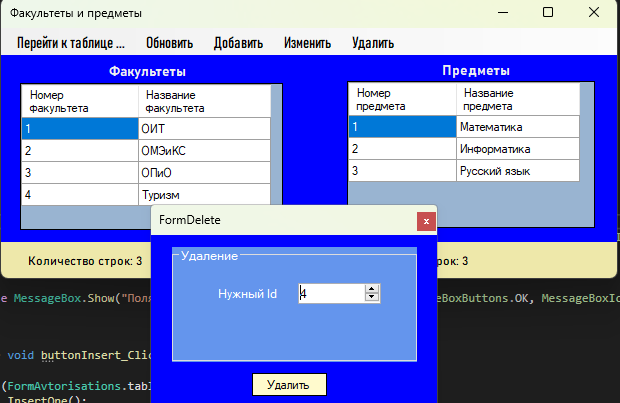


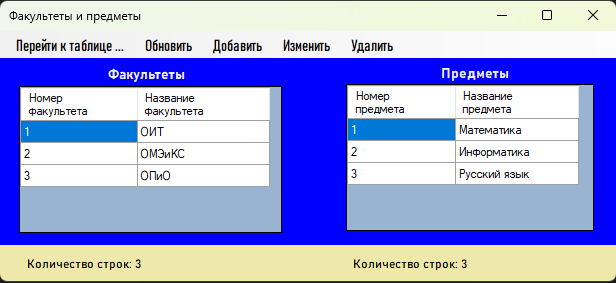












# **Лабораторная работа №21**

**Реализация функции сортировки, поиска и фильтрации в интерфейсе вывода данных**

**Цель:** Усовершенствовать работу номер 20, добавив к ней личный кабинет администратора. Научиться создавать инструменты работы с БД для администратора.

**Задания:**

2. Реализуйте функции фильтра данных по следующим атрибутам: ФИО студента, номер зачетки, ФИ преподавателя, название предмета.

3. Реализуйте функцию поиска по уникальному атрибуту в каждом отношении.

4. Отредактируйте отображение таблиц, скройте таблицы и данные, которые не нужны для работы обычного пользователя. Например, такой столбец, как номер факультета в таблице студент не несет никакой информации. Они полезны лишь для администраторов баз данных. Отобразите вместо номера название факультета. И т.д.

Запрос:

SELECT У.Номер\_зачетки, У.Номер\_семестра, П.Название\_предмета AS Предмет, У.Оценка, У.Дата\_экзамена, У.Фамилия\_преподавателя

FROM dbo.Успеваемость AS У INNER JOIN

dbo.Предмет AS П ON У.Предмет = П.Номер\_предмета

SELECT С.Номер\_зачетки, С.Фамилия\_имя\_отчество, С.Дата\_поступления, Ф.Название\_факультета AS Факультет, Сц.Название\_специальности AS Специяальность, С.Курс, С.Номер\_группы

FROM dbo.Студент AS С INNER JOIN

dbo.Факультет AS Ф ON С.Факультет = Ф.Номер\_факультета INNER JOIN

dbo.Специальность AS Сц ON С.Специальность = Сц.Код\_специальности

Код:

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class FormMark : Form

{

public FormMark()

{

InitializeComponent();

FormAvtorisations.table\_col = 6;

if (FormAvtorisations.rol == "user")

FormAvtorisations.table = "УспеваемостьДляЮзера";

else

FormAvtorisations.table = "Успеваемость";

}

public void CreateColumns()

{

dataGridView1.ColumnCount = 6;

dataGridView1.Columns[0].HeaderText = "Номер зачетки";

dataGridView1.Columns[1].HeaderText = "Номер семестра";

dataGridView1.Columns[2].HeaderText = "Предмет";

dataGridView1.Columns[3].HeaderText = "Оценка";

dataGridView1.Columns[4].HeaderText = "Дата экзамена";

dataGridView1.Columns[5].HeaderText = "Фамилия преподавателя";

}

public void ReadWriteInTable(DataGridView dg, string \_q= "")

{

//связь с sql передача туда запроса и строки для соединения

SqlCommand command = new SqlCommand("SELECT \* FROM "+FormAvtorisations.table+\_q, FormAvtorisations.dataBase.GetConection());

FormAvtorisations.dataBase.OpenConection();//открыть соединение

//объект для чтения из бд

SqlDataReader reader = command.ExecuteReader();

dg.Rows.Clear();//очистка строк

while (reader.Read())

{

if (FormAvtorisations.rol == "user")

dg.Rows.Add(reader.GetInt32(0), reader.GetInt32(1), reader.GetString(2), reader.GetInt32(3), reader.GetDateTime(4), reader.GetString(5));

else

dg.Rows.Add(reader.GetInt32(0), reader.GetInt32(1), reader.GetInt32(2), reader.GetInt32(3), reader.GetDateTime(4), reader.GetString(5) );

}

reader.Close();

FormAvtorisations.dataBase.CloseConection();

}

private void FormMark\_Load(object sender, EventArgs e)

{

CreateColumns();

if (FormAvtorisations.rol == "user")

{

toolStripSeparator3.Visible = false;

toolStripSeparator2.Visible = false;

toolStripSeparator1.Visible = false;

toolStripTextBoxSpecial.Visible = false;

toolStripTextBoxObj.Visible = false;

toolStripMenuItemInsert.Visible = false;

toolStripMenuItemUP.Visible = false;

toolStripMenuItemDelete.Visible = false;

}

ReadWriteInTable(dataGridView1);

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

//закрытие

private void FormMark\_FormClosed(object sender, FormClosedEventArgs e)

{

if (Application.OpenForms.Count > 0) {

// вызываем главную форму, которая открыла текущую, главная форма всегда = 0 - [0]

Form ifrm = Application.OpenForms[0];

ifrm.StartPosition = FormStartPosition.Manual; // меняем параметр StartPosition у Form1, иначе она будет использовать тот, который у неё прописан в настройках и всегда будет открываться по центру экрана

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form1

}

//else Close();

}

private void toolStripTextBoxObj\_Click(object sender, EventArgs e)

{

Form ifrm = new FormObject();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void toolStripTextBoxStud\_Click(object sender, EventArgs e)

{

Form ifrm = new FormStudent();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void toolStripTextBoxSpecial\_Click(object sender, EventArgs e)

{

Form ifrm = new FormSpecial();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

//обновление

private void toolStripMenuItemUpDate\_Click(object sender, EventArgs e)

{

ReadWriteInTable(dataGridView1);

labelRowsCount.Text = dataGridView1.RowCount.ToString();

textBoxObj.Text = string.Empty;

textBoxTeach.Text = string.Empty;

textBoxSObj.Text = string.Empty;

numericUpDownSZach.Value = 0;

}

//удаление

private void toolStripMenuItemDelete\_Click(object sender, EventArgs e)

{

FormDelete about = new FormDelete(FormAvtorisations.table+ " where [Номер\_зачетки]=@code and [Предмет] =@table");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

//добавление

private void toolStripMenuItemInsert\_Click(object sender, EventArgs e)

{

FormInsert about = new FormInsert(FormAvtorisations.table + " ([Номер\_зачетки], [Номер\_семестра], [Предмет], [Оценка], [Дата\_экзамена], [Фамилия\_преподавателя]) values (@zach, @sem, @obj, @mark,@data, @teach)");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

//изменение

private void toolStripMenuItemUP\_Click(object sender, EventArgs e)

{

FormUpDate about = new FormUpDate(FormAvtorisations.table + " set [Номер\_семестра]=@sem, [Оценка]=@mark, [Дата\_экзамена]=@data, [Фамилия\_преподавателя]=@teach where [Номер\_зачетки] = @zach and [Предмет]= @obj");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

//фильтр

private void buttonFiltr\_Click(object sender, EventArgs e)

{

if ((textBoxObj.Text != string.Empty) && (textBoxTeach.Text == string.Empty))

{

ReadWriteInTable(dataGridView1, " where [Предмет] = '" + textBoxObj.Text + "'");

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

else if ((textBoxObj.Text == string.Empty) && (textBoxTeach.Text != string.Empty))

{

ReadWriteInTable(dataGridView1, " where [Фамилия\_преподавателя] = '" + textBoxTeach.Text+"'");

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

else if ((textBoxObj.Text != string.Empty) && (textBoxTeach.Text != string.Empty))

{

ReadWriteInTable(dataGridView1, " where [Фамилия\_преподавателя] = '" + textBoxTeach.Text + "' and [Предмет] = '" + textBoxObj.Text + "'");

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

else MessageBox.Show("Поля не могут быть одновременно заполнены!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

//поиск

private void buttonSearch\_Click(object sender, EventArgs e)

{

if ((textBoxSObj.Text != string.Empty) && (numericUpDownSZach.Value != 0))

{

ReadWriteInTable(dataGridView1, " where [Предмет] = '" + textBoxSObj.Text + "' and [Номер\_зачетки] = "+numericUpDownSZach.Value);

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

else MessageBox.Show("Поля не могут быть пустыми!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class FormStudent : Form

{

public FormStudent()

{

InitializeComponent();

if (FormAvtorisations.rol == "user")

FormAvtorisations.table = "СтудентДляЮзера";

else

FormAvtorisations.table = "Студент";

FormAvtorisations.table\_col = 7;

}

public void CreateColumns()

{

dataGridView1.ColumnCount = 7;

dataGridView1.Columns[0].HeaderText = "Номер зачетки";

dataGridView1.Columns[1].HeaderText = "Фамилия имя отчество";

dataGridView1.Columns[2].HeaderText = "Дата поступления";

dataGridView1.Columns[3].HeaderText = "Факультет";

dataGridView1.Columns[4].HeaderText = "Специальность";

dataGridView1.Columns[5].HeaderText = "Курс";

dataGridView1.Columns[6].HeaderText = "Номер группы";

}

public void ReadWriteInTable(DataGridView dg, string \_q = "")

{

//связь с sql передача туда запроса и строки для соединения

SqlCommand command = new SqlCommand("SELECT \* FROM " + FormAvtorisations.table + \_q, FormAvtorisations.dataBase.GetConection());

FormAvtorisations.dataBase.OpenConection();//открыть соединение

//объект для чтения из бд

SqlDataReader reader = command.ExecuteReader();

dg.Rows.Clear();//очистка строк

while (reader.Read())

{

if (FormAvtorisations.rol == "user")

dg.Rows.Add(reader.GetInt32(0), reader.GetString(1), reader.GetDateTime(2), reader.GetString(3), reader.GetString(4), reader.GetInt32(5), reader.GetInt32(6));

else

//запись прочитанной строки в таблицу

dg.Rows.Add(reader.GetInt32(0), reader.GetString(1), reader.GetDateTime(2), reader.GetInt32(3), reader.GetInt32(4), reader.GetInt32(5), reader.GetInt32(6));

}

reader.Close();

FormAvtorisations.dataBase.CloseConection();

}

private void FormStudent\_Load(object sender, EventArgs e)

{

CreateColumns();

ReadWriteInTable(dataGridView1);

labelRowsCount.Text += dataGridView1.RowCount;

if(FormAvtorisations.rol == "user")

{

toolStripSeparator3.Visible = false;

toolStripSeparator2.Visible = false;

toolStripSeparator1.Visible = false;

toolStripTextBoxSpecial.Visible = false;

toolStripTextBoxObj.Visible = false;

toolStripMenuItemInsert.Visible = false;

toolStripMenuItemUP.Visible = false;

toolStripMenuItemDelete.Visible = false;

}

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

private void toolStripTextBoxMark\_Click(object sender, EventArgs e)

{

Form ifrm = new FormMark();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void toolStripTextBoxObj\_Click(object sender, EventArgs e)

{

Form ifrm = new FormObject();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void toolStripTextBoxSpecial\_Click(object sender, EventArgs e)

{

Form ifrm = new FormSpecial();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void FormStudent\_FormClosed(object sender, FormClosedEventArgs e)

{

if (Application.OpenForms.Count > 0)

{

// вызываем главную форму, которая открыла текущую, главная форма всегда = 0 - [0]

Form ifrm = Application.OpenForms[0];

ifrm.StartPosition = FormStartPosition.Manual; // меняем параметр StartPosition у Form1, иначе она будет использовать тот, который у неё прописан в настройках и всегда будет открываться по центру экрана

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form1

}

}

//обновление

private void toolStripMenuItemUpDate\_Click(object sender, EventArgs e)

{

ReadWriteInTable(dataGridView1);

labelRowsCount.Text = dataGridView1.RowCount.ToString();

textBoxTeach.Text = string.Empty;

numericUpDownSZach.Value = 0;

}

//добавление

private void toolStripMenuItemInsert\_Click(object sender, EventArgs e)

{

FormInsert about = new FormInsert(FormAvtorisations.table + " ([Фамилия\_имя\_отчество], [Дата\_поступления], [Факультет], [Специальность], [Курс], [Номер\_группы]) values (@name, @data, @fac, @spec,@curs, @group)");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

//удаление

private void toolStripMenuItemDelete\_Click(object sender, EventArgs e)

{

FormDelete about = new FormDelete(FormAvtorisations.table + " where [Номер\_зачетки]=@code");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

//изменение

private void toolStripMenuItemUP\_Click(object sender, EventArgs e)

{

FormUpDate about = new FormUpDate(FormAvtorisations.table + " set [Фамилия\_имя\_отчество]=@name, [Дата\_поступления]=@data, [Факультет]=@fac, [Специальность]=@spec, [Курс]=@curs, [Номер\_группы]=@group where [Номер\_зачетки] = @id");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

private void buttonFiltr\_Click(object sender, EventArgs e)

{

if (textBoxTeach.Text != string.Empty)

{

ReadWriteInTable(dataGridView1, " where [Фамилия\_имя\_отчество] = '" + textBoxTeach.Text + "'" );

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

else MessageBox.Show("Поля не могут быть пустыми!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void buttonSearch\_Click(object sender, EventArgs e)

{

if (numericUpDownSZach.Value != 0)

{

ReadWriteInTable(dataGridView1, " where [Номер\_зачетки] = " + numericUpDownSZach.Value);

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

else MessageBox.Show("Поля не могут быть пустыми!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class FormSpecial : Form

{

public FormSpecial()

{

InitializeComponent();

FormAvtorisations.table ="Специальность";

FormAvtorisations.table\_col = 2;

}

public void CreateColumns()

{

dataGridView1.ColumnCount = 2;

dataGridView1.Columns[0].HeaderText = "Код специальности";

dataGridView1.Columns[1].HeaderText = "Название\_специальности";

}

public void ReadWriteInTable(DataGridView dg, string \_q = "")

{

SqlCommand command = new SqlCommand("SELECT \* FROM " + FormAvtorisations.table + \_q, FormAvtorisations.dataBase.GetConection());

FormAvtorisations.dataBase.OpenConection();//открыть соединение

//объект для чтения из бд

SqlDataReader reader = command.ExecuteReader();

dg.Rows.Clear();//очистка строк

while (reader.Read())

{

//запись прочитанной строки в таблицу

dg.Rows.Add(reader.GetInt32(0), reader.GetString(1));

}

reader.Close();

FormAvtorisations.dataBase.CloseConection();

}

private void FormSpecial\_Load(object sender, EventArgs e)

{

CreateColumns();

ReadWriteInTable(dataGridView1);

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

private void FormSpecial\_FormClosed(object sender, FormClosedEventArgs e)

{

if (Application.OpenForms.Count > 0)

{

// вызываем главную форму, которая открыла текущую, главная форма всегда = 0 - [0]

Form ifrm = Application.OpenForms[0];

ifrm.StartPosition = FormStartPosition.Manual; // меняем параметр StartPosition у Form1, иначе она будет использовать тот, который у неё прописан в настройках и всегда будет открываться по центру экрана

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form1

}

}

private void toolStripTextBoxObj\_Click(object sender, EventArgs e)

{

Form ifrm = new FormObject();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void toolStripTextBoxStud\_Click(object sender, EventArgs e)

{

Form ifrm = new FormStudent();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void toolStripTextBoxMark\_Click(object sender, EventArgs e)

{

Form ifrm = new FormMark();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

//обновление

private void toolStripMenuItemUpDate\_Click(object sender, EventArgs e)

{

ReadWriteInTable(dataGridView1);

labelRowsCount.Text = dataGridView1.RowCount.ToString();

numericUpDownS.Value = 0;

}

//добавление

private void toolStripMenuItemInsert\_Click(object sender, EventArgs e)

{

FormInsert about = new FormInsert(FormAvtorisations.table + " ([Название\_специальности]) values (@name)");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

//удаление

private void toolStripMenuItemDelete\_Click(object sender, EventArgs e)

{

FormDelete about = new FormDelete(FormAvtorisations.table + " where [Код\_специальности]=@code");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

//изменение

private void toolStripMenuItemUP\_Click(object sender, EventArgs e)

{

FormUpDate about = new FormUpDate(FormAvtorisations.table + " set [Название\_специальности]=@name where [Код\_специальности] = @id");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

private void buttonSearch\_Click(object sender, EventArgs e)

{

if (numericUpDownS.Value != 0)

{

ReadWriteInTable(dataGridView1, " where [Код\_специальности] = " + numericUpDownS.Value);

labelRowsCount.Text = dataGridView1.RowCount.ToString();

}

else MessageBox.Show("Поля не могут быть пустыми!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

using System;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace WindowsFormsApp1

{

public partial class FormObject : Form

{

public FormObject()

{

InitializeComponent();

FormAvtorisations.table = "";

FormAvtorisations.table\_col = 2;

}

public void CreateColumnsFac(DataGridView dg)

{

dg.ColumnCount = 2;

dg.Columns[0].HeaderText = "Номер факультета";

dg.Columns[1].HeaderText = "Название факультета";

}

public void CreateColumnsObj(DataGridView dg)

{

dg.ColumnCount = 2;

dg.Columns[0].HeaderText = "Номер предмета";

dg.Columns[1].HeaderText = "Название предмета";

}

public void ReadWriteInTable(DataGridView dg, string \_q="")

{

string query = "SELECT \* FROM " + FormAvtorisations.table+\_q;

//связь с sql передача туда запроса и строки для соединения

SqlCommand command = new SqlCommand(query, FormAvtorisations.dataBase.GetConection());

FormAvtorisations.dataBase.OpenConection();//открыть соединение

//объект для чтения из бд

SqlDataReader reader = command.ExecuteReader();

dg.Rows.Clear();//очистка строк

while (reader.Read())

{

//запись прочитанной строки в таблицу

dg.Rows.Add(reader.GetInt32(0), reader.GetString(1));

}

reader.Close();

FormAvtorisations.dataBase.CloseConection();

}

private void FormObject\_Load(object sender, EventArgs e)

{

CreateColumnsObj(dataGridView1);

FormAvtorisations.table = "Предмет";

ReadWriteInTable(dataGridView1);

labelRowsCountObj.Text = dataGridView1.RowCount.ToString();

CreateColumnsFac(dataGridView2);

FormAvtorisations.table = " Факультет";

ReadWriteInTable(dataGridView2);

labelRowsCountFac.Text = dataGridView2.RowCount.ToString();

if (FormAvtorisations.rol == "user")

{

toolStripMenuItemInsert.Visible = false;

toolStripMenuItemUP.Visible = false;

toolStripMenuItemDelete.Visible = false;

}

}

private void toolStripTextBoxStud\_Click(object sender, EventArgs e)

{

Form ifrm = new FormStudent();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void toolStripTextBoxMark\_Click(object sender, EventArgs e)

{

Form ifrm = new FormMark();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void toolStripTextBoxSpecial\_Click(object sender, EventArgs e)

{

Form ifrm = new FormSpecial();

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

}

private void FormObject\_FormClosed(object sender, FormClosedEventArgs e)

{

if (Application.OpenForms.Count > 0)

{

// вызываем главную форму, которая открыла текущую, главная форма всегда = 0 - [0]

Form ifrm = Application.OpenForms[0];

ifrm.StartPosition = FormStartPosition.Manual; // меняем параметр StartPosition у Form1, иначе она будет использовать тот, который у неё прописан в настройках и всегда будет открываться по центру экрана

ifrm.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

ifrm.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

ifrm.Show(); // отображаем Form1

}

}

//обновление

private void toolStripTextBoxUpFac\_Click(object sender, EventArgs e)

{

FormAvtorisations.table = " Факультет";

ReadWriteInTable(dataGridView2);

labelRowsCountFac.Text = dataGridView2.RowCount.ToString();

numericUpDownSF.Value = 0;

}

private void toolStripTextBoxUpObj\_Click(object sender, EventArgs e)

{

FormAvtorisations.table = "Предмет";

ReadWriteInTable(dataGridView1);

labelRowsCountObj.Text = dataGridView1.RowCount.ToString();

numericUpDownSO.Value = 0;

}

//добавление

private void toolStripTextBoxInsertFac\_Click(object sender, EventArgs e)

{

FormAvtorisations.table = "Факультет";

FormInsert about = new FormInsert(FormAvtorisations.table + " ([Название\_факультета]) values (@name)");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView2);

labelRowsCountFac.Text = dataGridView2.RowCount.ToString();

}

private void toolStripTextBoxInsertObj\_Click(object sender, EventArgs e)

{

FormAvtorisations.table = "Предмет";

FormInsert about = new FormInsert(FormAvtorisations.table + " ([Название\_предмета]) values (@name)");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

labelRowsCountObj.Text = dataGridView1.RowCount.ToString();

}

//изменение

private void toolStripTextBoxUpDateFac\_Click(object sender, EventArgs e)

{

FormAvtorisations.table = "Факультет";

FormUpDate about = new FormUpDate(FormAvtorisations.table + " set [Название\_факультета]=@name where [Номер\_факультета] = @id");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView2);

labelRowsCountFac.Text = dataGridView2.RowCount.ToString();

}

private void toolStripTextBoxUpDateObj\_Click(object sender, EventArgs e)

{

FormAvtorisations.table = "Предмет";

FormUpDate about = new FormUpDate(FormAvtorisations.table + " set [Название\_предмета]=@name where [Номер\_предмета] = @id");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

labelRowsCountObj.Text = dataGridView1.RowCount.ToString();

}

//удаление

private void toolStripTextBoxDelFac\_Click(object sender, EventArgs e)

{

FormAvtorisations.table = "Факультет";

FormDelete about = new FormDelete(FormAvtorisations.table + " where [Номер\_факультета]=@code");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView2);

labelRowsCountFac.Text = dataGridView2.RowCount.ToString();

}

private void toolStripTextBoxDelObj\_Click(object sender, EventArgs e)

{

FormAvtorisations.table = "Предмет";

FormDelete about = new FormDelete(FormAvtorisations.table + " where [Номер\_предмета]=@code");

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

ReadWriteInTable(dataGridView1);

labelRowsCountObj.Text = dataGridView1.RowCount.ToString();

}

//поиск

private void buttonSF\_Click(object sender, EventArgs e)

{

if (numericUpDownSF.Value != 0)

{

FormAvtorisations.table = "Факультет";

ReadWriteInTable(dataGridView2, " where [Номер\_факультета] = " + numericUpDownSF.Value);

labelRowsCountFac.Text = dataGridView2.RowCount.ToString();

}

else MessageBox.Show("Поля не могут быть пустыми!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void buttonSO\_Click(object sender, EventArgs e)

{

if (numericUpDownSO.Value != 0)

{

FormAvtorisations.table = "Предмет";

ReadWriteInTable(dataGridView1, " where [Номер\_предмета] = " + numericUpDownSO.Value);

labelRowsCountFac.Text = dataGridView1.RowCount.ToString();

}

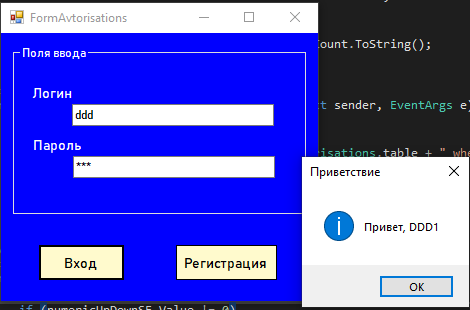
else MessageBox.Show("Поля не могут быть пустыми!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

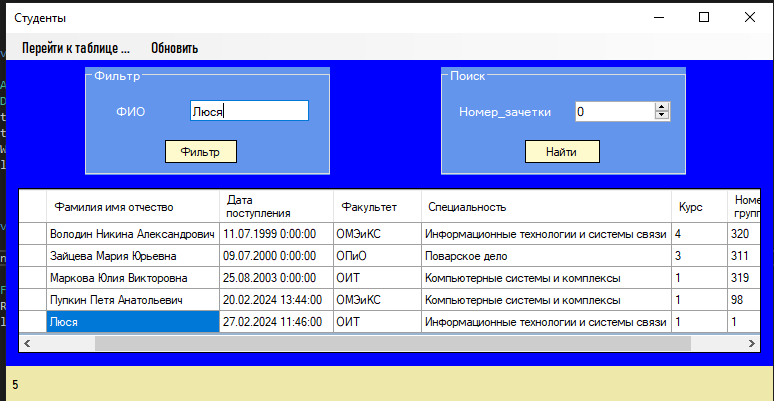
}

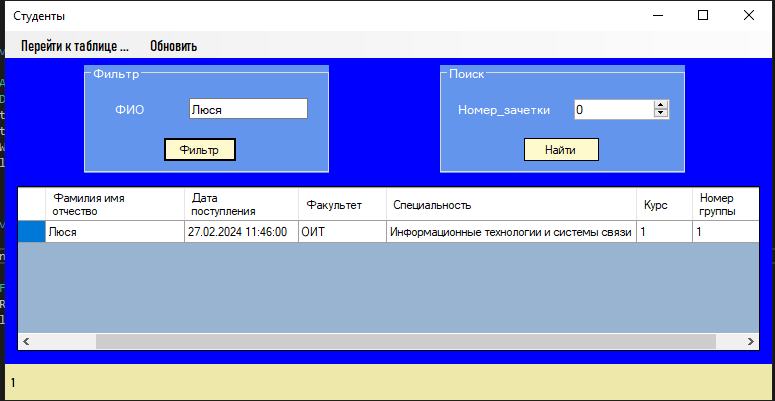
}

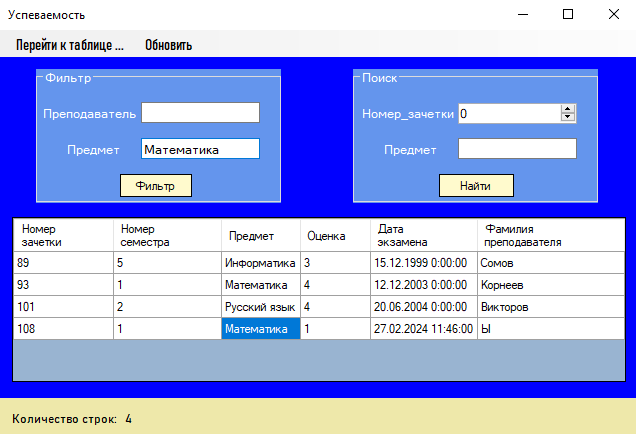
}

Результат:

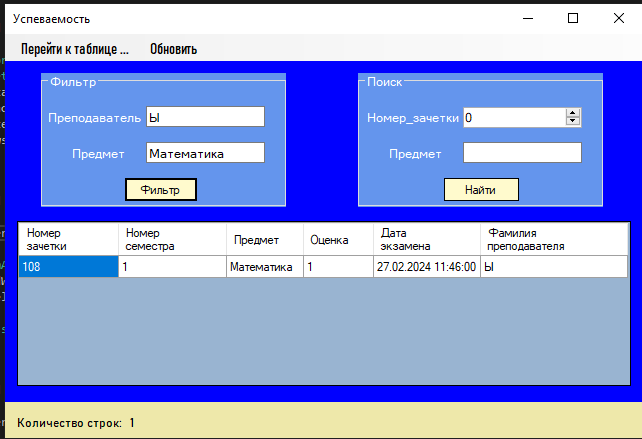


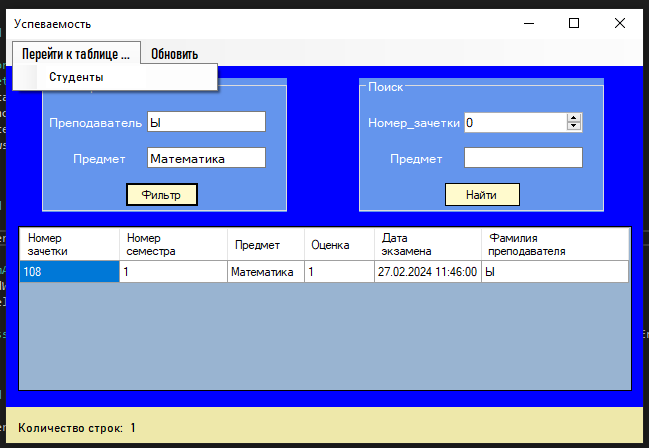


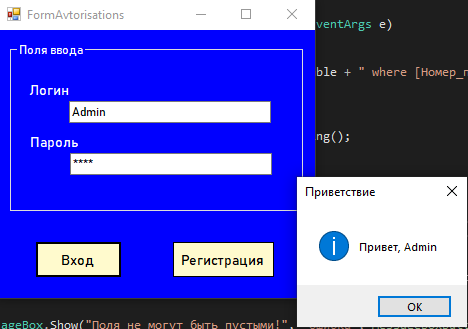










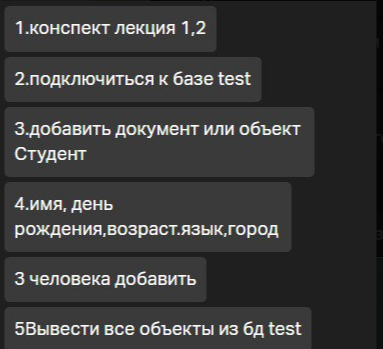


# **Лабораторная работа №22**

**Реализация функции сортировки, поиска и фильтрации в интерфейсе вывода данных**

**Цель:** Познакомиться с MongoDB и поработать в не реляционной бд.

**ЗАДАНИЕ1**

****

**ЗАДАНИЕ2**

1.Создать текстовый файл Users.js на диске со следующим содержимым:

db.users.insertMany([…])

Имя-Алиса, день рождения-20.02.07, возраст-17, язык-английский, французский;

Имя-Лена, день рождения-null, возраст-20, язык-английский, испанский;

Имя-Катя, день рождения-12.01.07, возраст-17, язык-немецкий, русский.

С помощью метода insertMany добавить три документа в коллекцию users.

2. Загрузить файл в текущую базу данных.

3. Выведем все документы, в которых возраст=17;

4. Выведем все документы, в которых день рождения=null;

5. Вывести все документы, в которых испанский в массиве languages находится на втором месте;

5. Вывести только информацию, например, значения поля возраст у всех документов, в которых name=Tom;

6. Найти все поля документа кроме свойства age.

7. Добавить в коллекцию users следующий документ:

db.users.insertOne({"name": "Alex", "age": 28, company: {"name":"Microsoft", "country":"USA"}}). Вернуть все документы, у которых в ключе company вложенноесвойство name=Microsoft

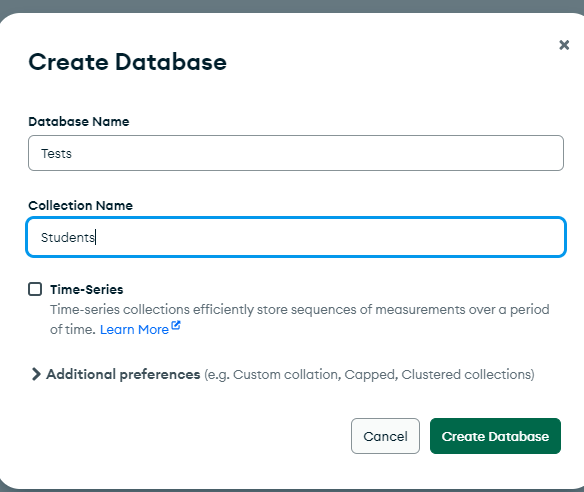
**.**

**Ход работы:**

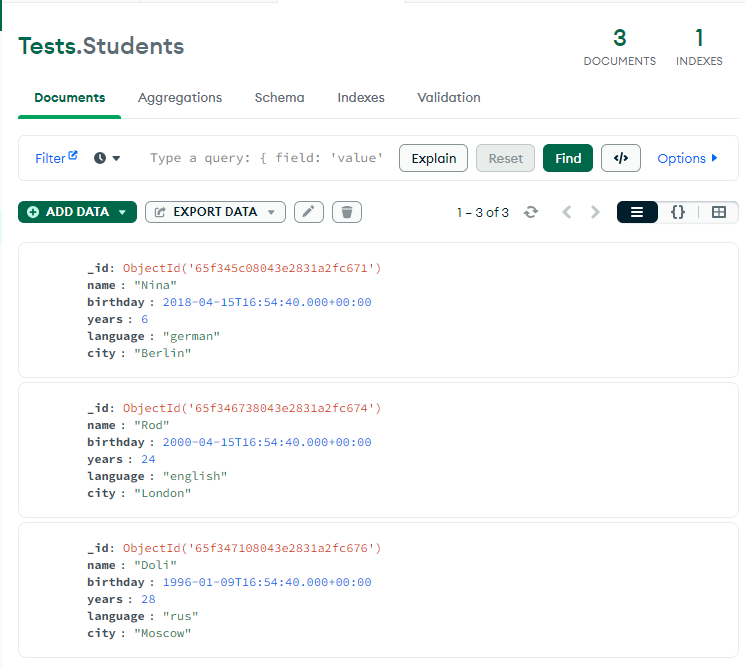
Запустила сервер и подключилась к нему через клиента



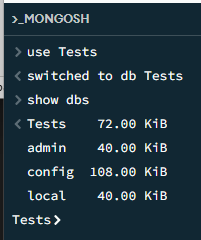
Решила перейти на интерфейс Compass и создала базу данных и коллекцию



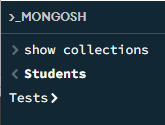
Создала трех студентов



Вывела всю информацию о базах данных



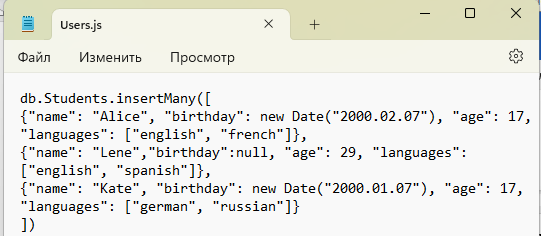
Посмотрела, что содержится в базе Тест



А теперь и внутри коллекции



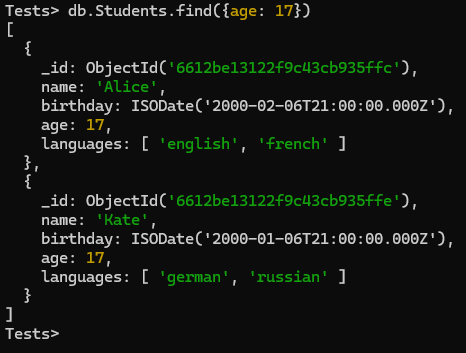
Создала документ



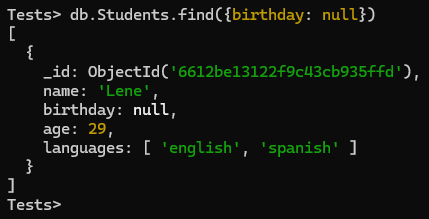
Загрузила эти данные в коллекцию Students бд Tests через файл



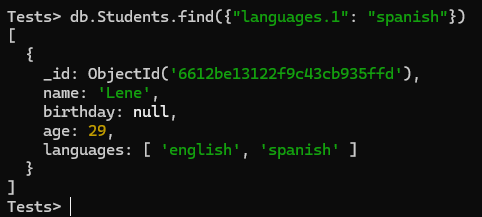
Все ученики 17 лет



Студенты без даты рождения



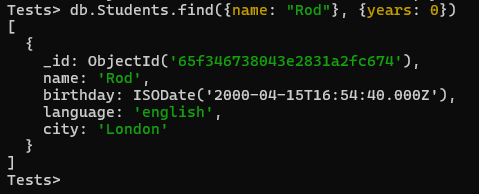
Испанский второй в массиве языков

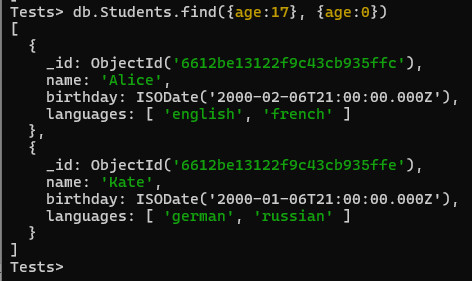


Вывести только информацию, например, значения поля возраст у всех документов, в которых name=Rod;



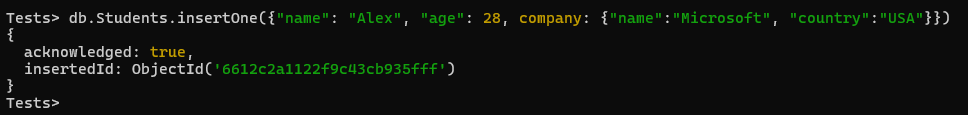
Найти все поля документа кроме свойства age.



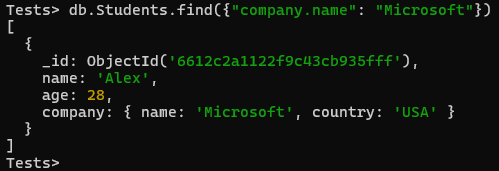


Добавить в коллекцию Students следующий документ:

db.users.insertOne({"name": "Alex", "age": 28, company: {"name":"Microsoft", "country":"USA"}}).



Вернуть все документы, у которых в ключе company вложенное свойство name=Microsoft



# **Лабораторная работа №23**

**Шифрование данных**

**Цель:** Изучить различные способы шифрования данных в MS SQL Server. Научиться их применять для защиты данных в таблицах.

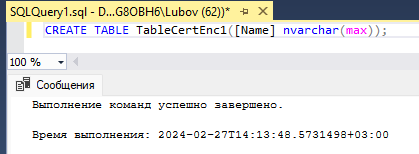
**Задания:**

1. Изучите все способы шифрования в методическом указании и выполните действия, описанные в нем.

**Шифрование с помощью сертификата**

Создайте таблицу:

CREATE TABLE TableCertEnc1(Name nvarchar(max));

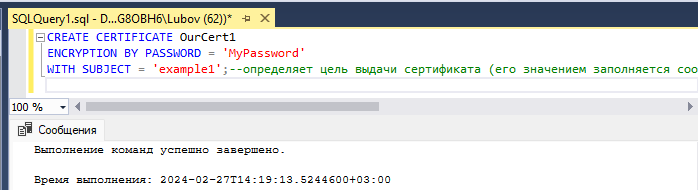


Заведите сертификат:

CREATE CERTIFICATE OurCert1

ENCRYPTION BY PASSWORD = 'MyPassword'

WITH SUBJECT = 'example1';--определяет цель выдачи сертификата (его значением заполняется соответствующее поле сертификата в соответствии со стандартом X.509v1)



Используйте функцию EncryptByCert() для вставки зашифрованной информации в таблицу. EncryptByCert() в качестве аргумента требует id сертификата, который мы получаем с помощью Cert\_ID(). И само значение, которые вы хотите внести в таблицу:

INSERT INTO TableCertEnc1

values(EncryptByCert(Cert\_ID('OurCert1'), 'Secret Name'));

Выведите информацию и посмотрите результат:

Select \* From TableCertEnc1



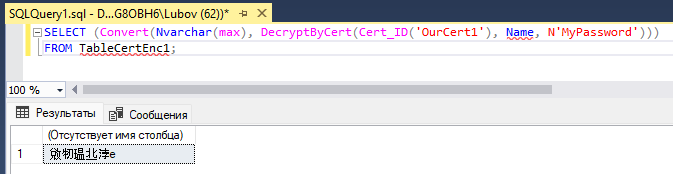
Для дешифровки используется функция DecryptByCert со следующими параметрами:

● 1ый параметр - идентификатор сертификата

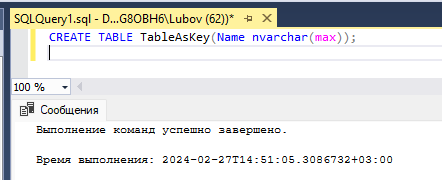
● 2ой параметр - строка, переменная или столбец для расшифровки

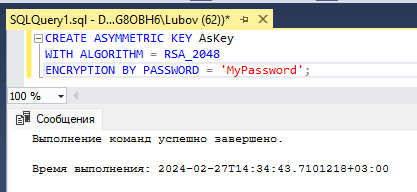
● 3ий параметр - пароль, указанный при создании сертификата

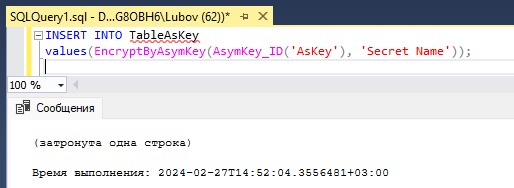
SELECT (Convert(Nvarchar(max), DecryptByCert(Cert\_ID('OurCert1'), Name, N'MyPassword'))) FROM TableCertEnc1;

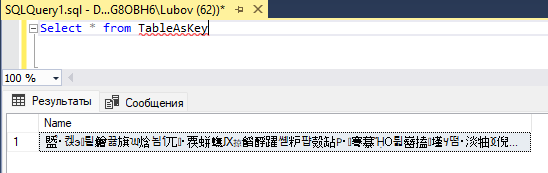


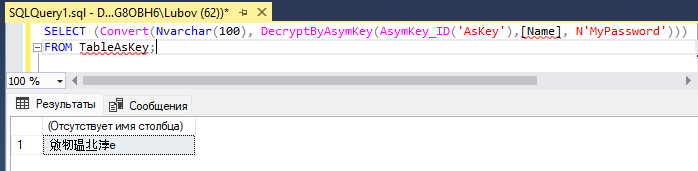
**Шифрование с помощью ассиметричного ключа**



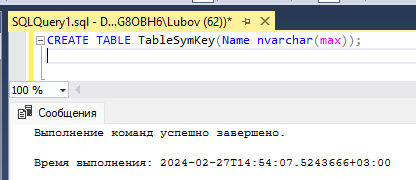


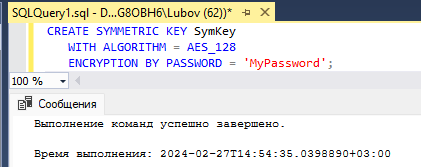


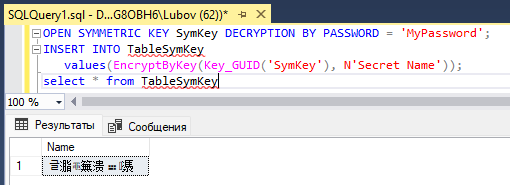
****

****

**Шифрование с помощью симметричного ключа**

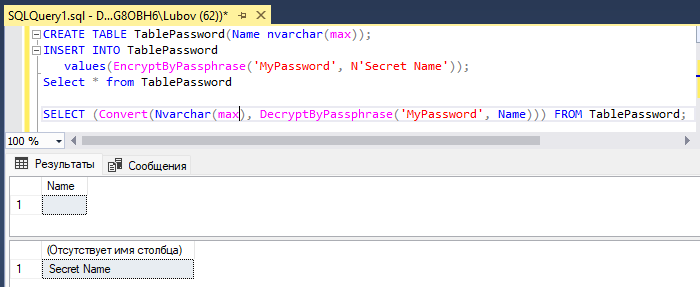






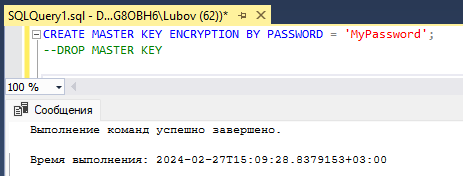


**Шифрование с помощью пароля**

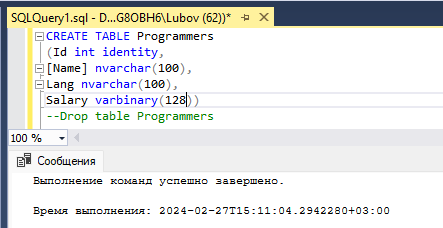


**Защита симметричного ключа пользовательским сертификатом**

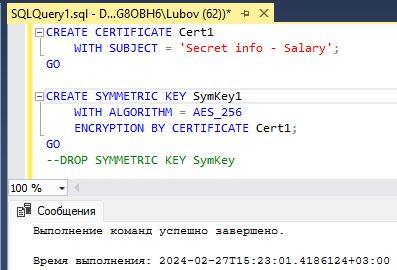
Шаг 1. Создаем главный ключ БД



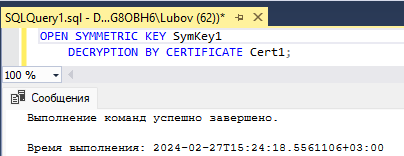
Шаг 2. Создаем таблицу в которой будем шифровать зарплату



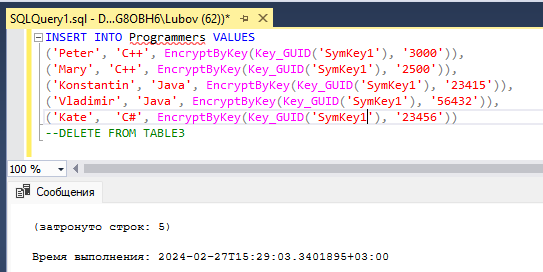
Шаг 3. Создаем сертификат для защиты симметричного ключа



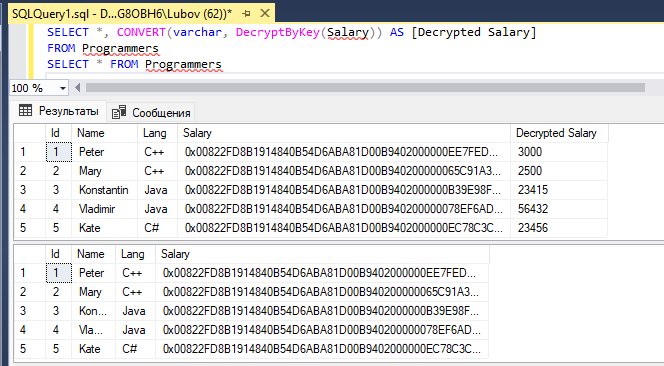
Шаг 4. Открываем симметричный ключ (обратите внимание на использование сертификата)



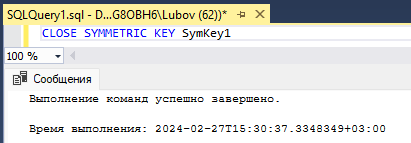
Шаг 5. Вставка нескольких строк в таблицу Programmers с зашифрованной зарплатой



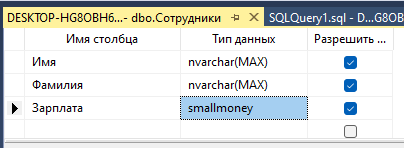
Шаг 6. Вывод информации



Шаг 7. Закрываем симметричный ключ



2. Создайте таблицу Сотрудники со следующими атрибутами (Имя, Фамилия, Зарплата)



3. Зашифруйте и дешифруйте имена и фамилии сотрудников всеми способами, описанными в методическом указании.

**Шифрование с помощью сертификата**

CREATE CERTIFICATE OurCert1

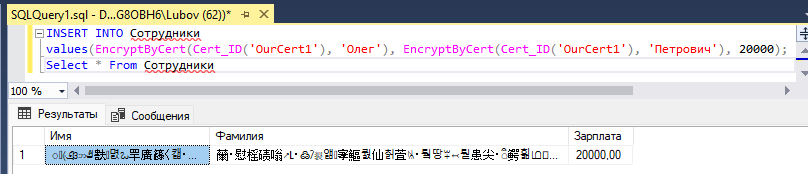
ENCRYPTION BY PASSWORD = 'MyPassword'

WITH SUBJECT = 'example1';--определяет цель выдачи сертификата (его значением заполняется соответствующее поле сертификата в соответствии со стандартом X.509v1)

INSERT INTO Сотрудники

values(EncryptByCert(Cert\_ID('OurCert1'), 'Олег'), EncryptByCert(Cert\_ID('OurCert1'), 'Петрович'), 20000);

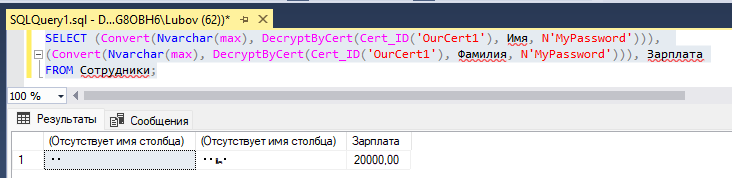
Select \* From Сотрудники



SELECT (Convert(Nvarchar(max), DecryptByCert(Cert\_ID('OurCert1'), Имя, N'MyPassword'))),

(Convert(Nvarchar(max), DecryptByCert(Cert\_ID('OurCert1'), Фамилия, N'MyPassword'))), Зарплата

FROM Сотрудники;



**Шифрование с помощью ассиметричного ключа**

CREATE ASYMMETRIC KEY AsKey

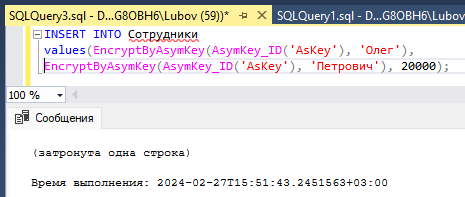
WITH ALGORITHM = RSA\_2048

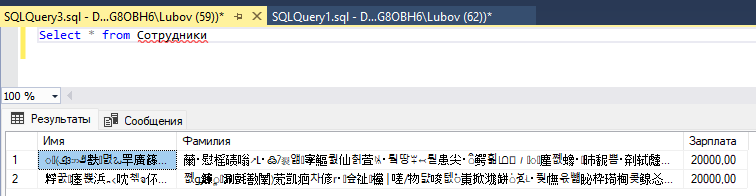
ENCRYPTION BY PASSWORD = 'MyPassword';

INSERT INTO Сотрудники

values(EncryptByAsymKey(AsymKey\_ID('AsKey'), 'Олег'),

EncryptByAsymKey(AsymKey\_ID('AsKey'), 'Петрович'), 30000);



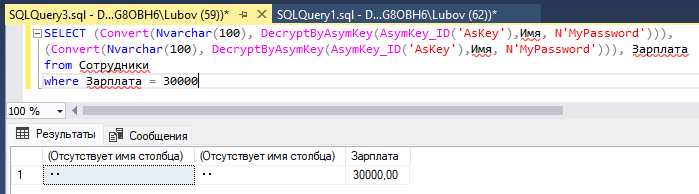


SELECT (Convert(Nvarchar(100), DecryptByAsymKey(AsymKey\_ID('AsKey'),Имя, N'MyPassword'))),

(Convert(Nvarchar(100), DecryptByAsymKey(AsymKey\_ID('AsKey'),Имя, N'MyPassword'))), Зарплата

from Сотрудники

where Зарплата = 30000



**Шифрование с помощью симметричного ключа**

CREATE SYMMETRIC KEY SymKey

WITH ALGORITHM = AES\_128

ENCRYPTION BY PASSWORD = 'MyPassword';

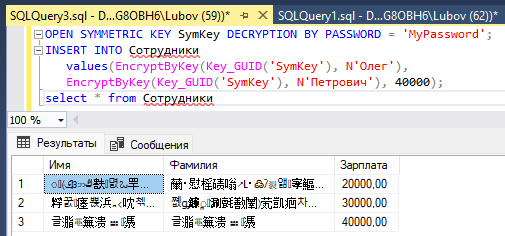
OPEN SYMMETRIC KEY SymKey DECRYPTION BY PASSWORD = 'MyPassword';

INSERT INTO Сотрудники

values(EncryptByKey(Key\_GUID('SymKey'), N'Олег'),

EncryptByKey(Key\_GUID('SymKey'), N'Петрович'), 40000);

select \* from Сотрудники



SELECT (Convert(Nvarchar(max), DecryptByKey(Имя))),

(Convert(Nvarchar(max), DecryptByKey(Фамилия))), Зарплата

FROM Сотрудники

where Зарплата = 40000

CLOSE SYMMETRIC KEY SymKey



**Шифрование с помощью пароля**

INSERT INTO Сотрудники

values(EncryptByPassphrase('MyPassword', N'Юлия'),

EncryptByPassphrase('MyPassword', N'Юрьевна'), 50000);

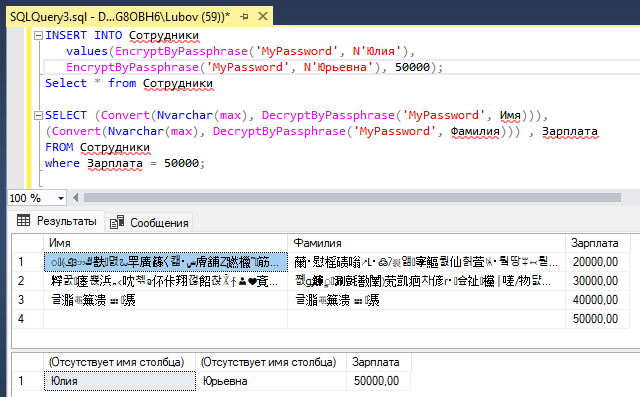
Select \* from Сотрудники

SELECT (Convert(Nvarchar(max), DecryptByPassphrase('MyPassword', Имя))),

(Convert(Nvarchar(max), DecryptByPassphrase('MyPassword', Фамилия))) , Зарплата

FROM Сотрудники

where Зарплата = 50000;



**Защита симметричного ключа пользовательским сертификатом**

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'MyPassword’;

CREATE CERTIFICATE Cert12

WITH SUBJECT = 'Secret info - Имя';

GO

CREATE SYMMETRIC KEY SymKey1

WITH ALGORITHM = AES\_256

ENCRYPTION BY CERTIFICATE Cert12;

GO

OPEN SYMMETRIC KEY SymKey12

DECRYPTION BY CERTIFICATE Cert12;

INSERT INTO Сотрудники1

values(EncryptByKey(Key\_GUID('SymKey12'),'Роман'),

'Ливьер', '60000')

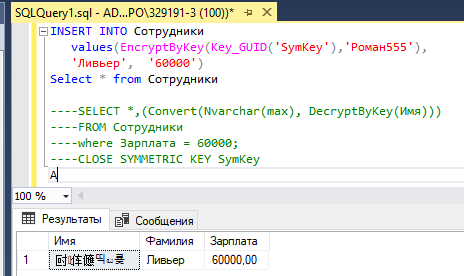
Select \* from Сотрудники1

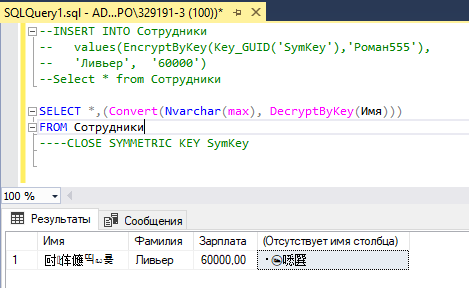
SELECT \*,(Convert(Nvarchar(max), DecryptByKey(Имя)))

FROM Сотрудники1

where Зарплата = 60000;

CLOSE SYMMETRIC KEY SymKey12





# **Лабораторная работа №24**

**Создание окна входа пользователя с использованием функций хеширования**

**Цель:** Усовершенствовать работу номер 18, добавив к ней личный кабинет администратора. Научиться создавать инструменты работы с БД для администратора.

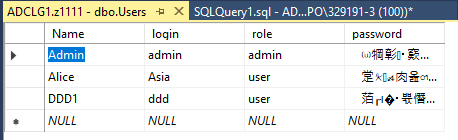
**Задания:**

1. Возьмите за основу ЛР 18.

2. Измените способ хранения паролей в вашей БД с открытого на хешированный для всех пользователей.

UpDate Users Set password = EncryptByPassphrase('MyPassword', N'ddd')

where [login] ='ddd'



3. Измените код приложения так, чтобы мы могли сравнивать введенный пароль с его хешированным вариантом в БД.

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

using ClassLibraryDataBase;

namespace WindowsFormsApp1

{

public partial class Avtorisations : Form

{

//для создания соединения

DataBase dataBase = new DataBase();

public Avtorisations()

{

InitializeComponent();

}

//проверка на существование такой записи в БД

public int VerificationDB(string log)

{

//запрос

string ComDel = " Select count(\*) from Users where login=@code";

//команда на выполенеие запроса через соединение

SqlCommand cmd1 = new SqlCommand(ComDel, dataBase.GetConection());

//запись в параметр значения

SqlParameter pr1 = new SqlParameter("@code", log);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr1);

dataBase.OpenConection();

int k =Convert.ToInt32(cmd1.ExecuteScalar());//кол-во найденных записей

dataBase.CloseConection();

return k;

}

//вход

public string EntranceInDB(string log, string pas)

{

//запрос который будет выполнен

string query = "SELECT Name FROM Users where login = @l and (Convert(Nvarchar(50),DecryptByPassphrase('MyPassword', password))) = @pass";

SqlParameter p1 = new SqlParameter("@l", log);

SqlParameter p2 = new SqlParameter("@pass", pas);

//связь с sql передача туда запроса и строки для соединения

SqlCommand command = new SqlCommand(query, dataBase.GetConection());

command.Parameters.Add(p1);

command.Parameters.Add(p2);

dataBase.OpenConection();//открыть соединение

string b = command.ExecuteScalar()==null?string.Empty: command.ExecuteScalar().ToString();

dataBase.CloseConection();

return b;

}

private void buttonEntrance\_Click(object sender, EventArgs e)

{

if ((textBoxLogin.Text != string.Empty)&&(textBoxPassword.Text!=string.Empty))

{

if (VerificationDB(textBoxLogin.Text) < 1)

{

DialogResult res = MessageBox.Show("Такого логина не существует! " +

"\nЖелаете зарегестрироваться? ", " Предупреждение ",

MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation,

MessageBoxDefaultButton.Button2);

if (res == DialogResult.Yes)

{

FormReg about = new FormReg(textBoxLogin.Text, textBoxPassword.Text);

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

}

}

else

{

string s= EntranceInDB(textBoxLogin.Text, textBoxPassword.Text);

if(s!=string.Empty)

MessageBox.Show("Привет, " + s, "Приветствие", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

else MessageBox.Show("Неверный пароль!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else MessageBox.Show("Пустые поля недопустимы!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void buttonReg\_Click(object sender, EventArgs e)

{

if ((textBoxLogin.Text != string.Empty) && (textBoxPassword.Text != string.Empty))

{

if (VerificationDB(textBoxLogin.Text) == 0)

{

FormReg about = new FormReg(textBoxLogin.Text, textBoxPassword.Text);

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

}

else

{

DialogResult res = MessageBox.Show("Такой логин уже существует! " +

"\nЖелаете войти? ", " Предупреждение",

MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation,

MessageBoxDefaultButton.Button2);

if (res == DialogResult.Yes)

{

string s = EntranceInDB(textBoxLogin.Text, textBoxPassword.Text);

if (s != string.Empty)

MessageBox.Show("Привет, " + s, "Приветствие", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

else MessageBox.Show("Вы ввели неверные данные!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

}

}

using ClassLibraryDataBase;

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class FormReg : Form

{

DataBase dataBase=new DataBase();

public FormReg(string log, string pass)

{

InitializeComponent();

//запрос на получение ролей и записи их в выпадающий список

string query = "SELECT Name FROM Rols";

//связь с sql передача туда запроса и строки для соединения

SqlCommand command = new SqlCommand(query, dataBase.GetConection());

dataBase.OpenConection();//открыть соединение

SqlDataReader Reader = command.ExecuteReader();//чтнение в массив

while (Reader.Read()) { comboBoxRole.Items.Add(Reader[0]); }

dataBase.CloseConection();

//начальный элемент - user

comboBoxRole.SelectedIndex=1;

//ограничение длины вводимых величин

textBoxName.MaxLength = 50;

textBoxLogin.MaxLength = 50;

textBoxPassword.MaxLength = 50;

//перенос данных

textBoxPassword.Text = pass;

textBoxLogin.Text = log;

}

private void buttonReg\_Click(object sender, EventArgs e)

{

if ((textBoxName.Text != String.Empty) && (textBoxLogin.Text != String.Empty) && (textBoxPassword.Text != String.Empty))

{

//запрос

string ComDel = $" Insert into Users values (@name, @log, @role, EncryptByPassphrase('MyPassword', N'{textBoxPassword.Text}'))";

//команда на выполенеие запроса через соединение

SqlCommand cmd1 = new SqlCommand(ComDel, dataBase.GetConection());

//запись в параметр значения

SqlParameter pr2 = new SqlParameter("@name", textBoxName.Text);

SqlParameter pr3 = new SqlParameter("@log", textBoxLogin.Text);

SqlParameter pr5 = new SqlParameter("@role", comboBoxRole.Text);

//добавление этого значения в запрос

cmd1.Parameters.Add(pr2);

cmd1.Parameters.Add(pr3);

cmd1.Parameters.Add(pr5);

dataBase.OpenConection();

if(cmd1.ExecuteNonQuery()!=0)

MessageBox.Show("Пользователь успешно зарегистрирован!", "Сообщение", MessageBoxButtons.OK, MessageBoxIcon.Information);

else MessageBox.Show("Что-то пошло не так!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

dataBase.CloseConection();

}

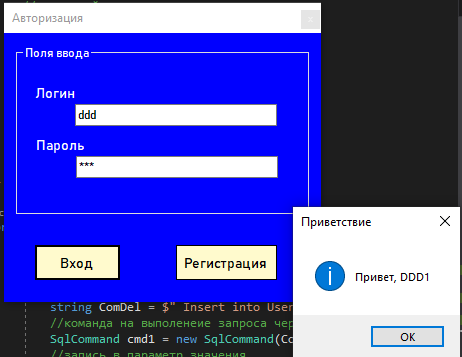
else MessageBox.Show("Пустые поля недопустимы!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

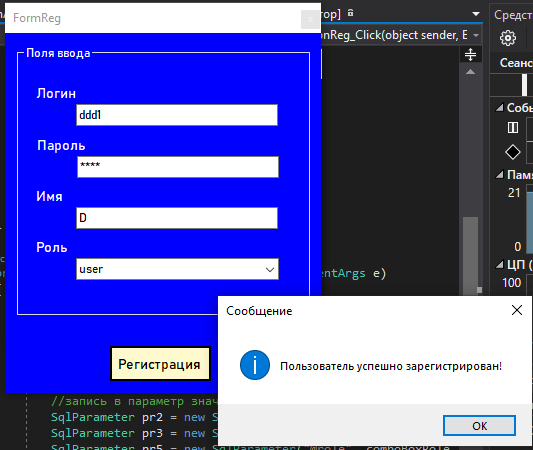
}

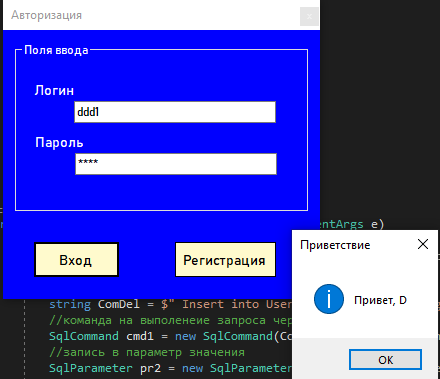
}

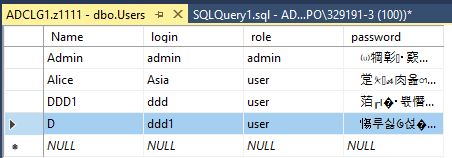
}

4. Осуществите вход и проверьте правильность работы программы.









# **Лабораторная работа №25**

**Определение прав доступа пользователя к данным**

**Цель:** Научиться определять права доступа для защиты данных в базе.

**Задания:**

1. Передайте пользователю PETROV право на изменение в базе данных значений оценок для записей о студентах;

CREATE LOGIN PETROV

WITH PASSWORD = '1234';

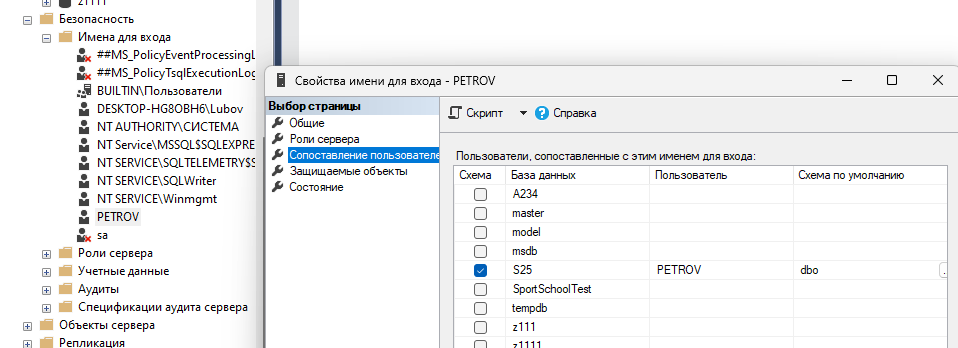
GO

CREATE USER PETROV FOR LOGIN PETROV ;

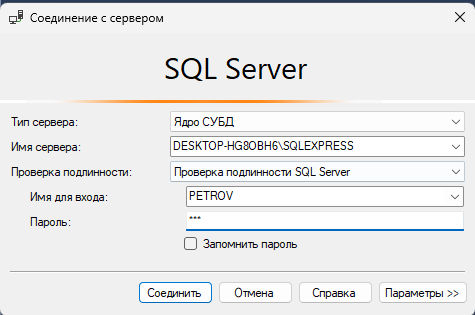
GO

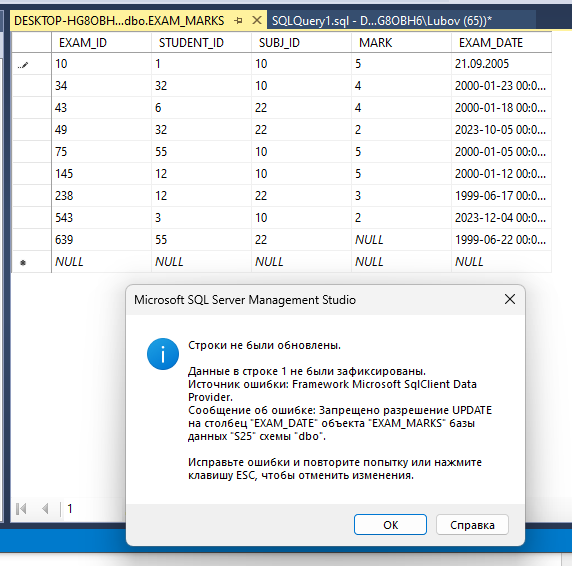


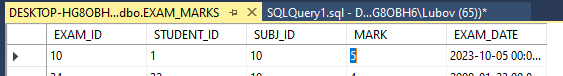
Нужно привязать к этому пользователю бд



GRANT UPDATE(Mark), SELECT ON [dbo].[EXAM\_MARKS] TO PETROV;









2. Передайте пользователю SIDOROV право передавать другим пользователям права на осуществление запросов к таблице EXAM\_MARKS;

CREATE LOGIN SIDOROV

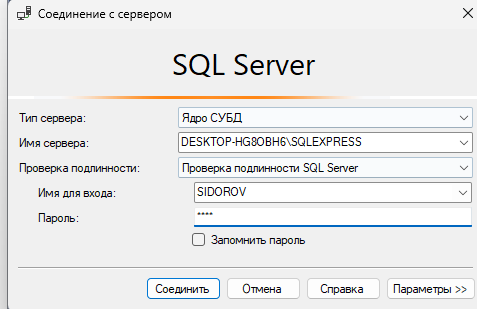
WITH PASSWORD = '1234';

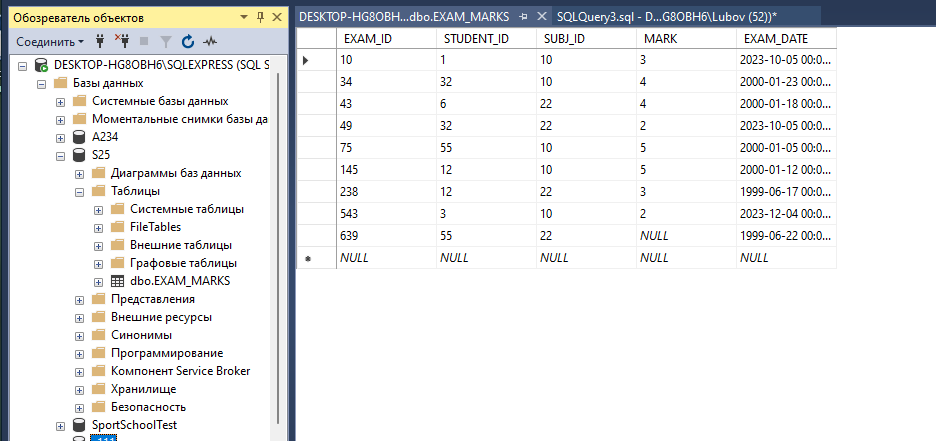
GO

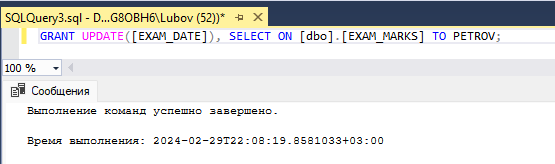
CREATE USER SIDOROV FOR LOGIN SIDOROV ;

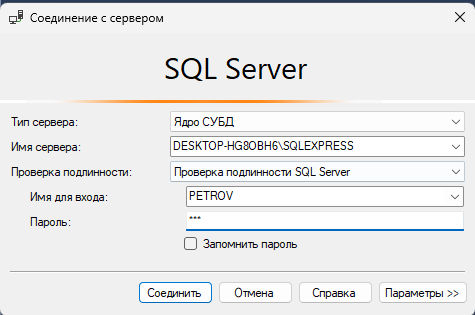
GO

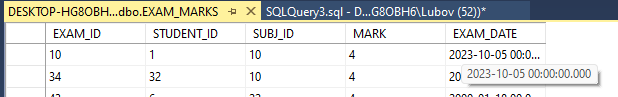
Grant SELECT on EXAM\_MARKS to SIDOROV WITH GRANT OPTION;

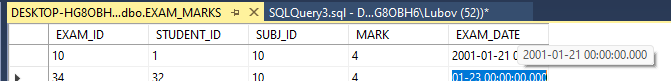












3. Отмените привилегию INSERT по отношению к таблице STUDENT у пользователя IVANOV и у всех других пользователей, которым она была им предоставлена.

CREATE LOGIN IVANOV

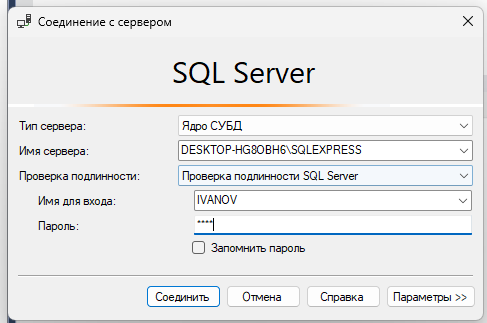
WITH PASSWORD = '1234';

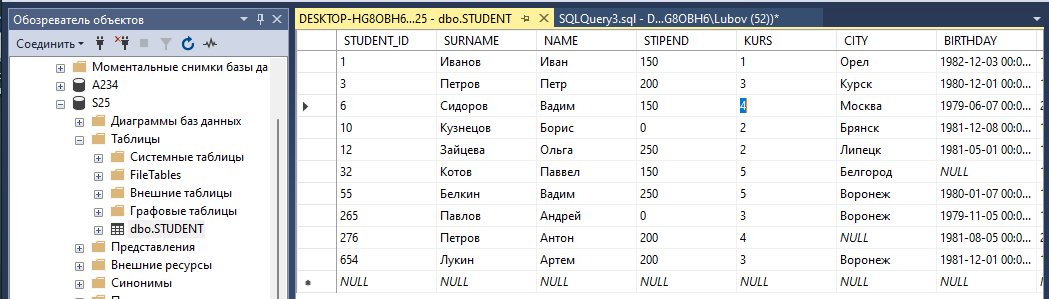
GO

CREATE USER IVANOV FOR LOGIN IVANOV ;

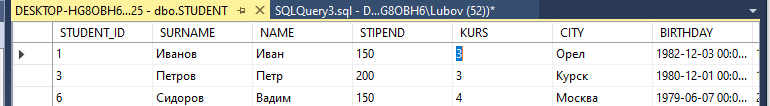
GO

GRANT ALL PRIVILEGES ON STUDENT TO IVANOV





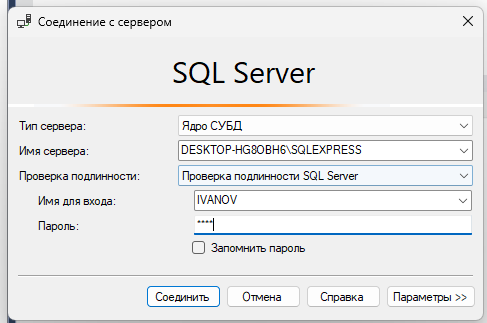
Изменение

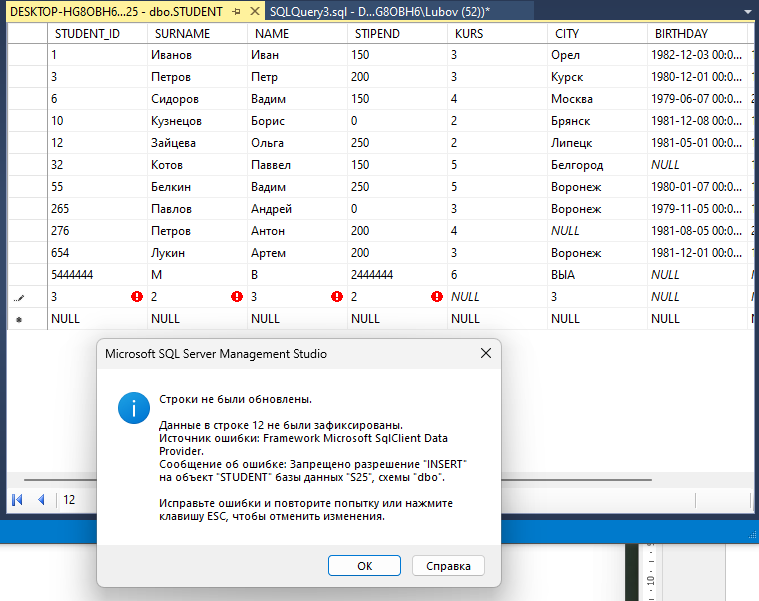


Новая строка

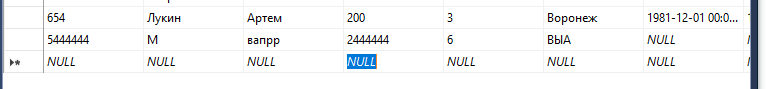


REVOKE INSERT ON STUDENT TO IVANOV

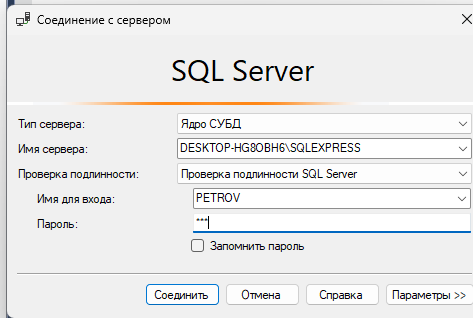




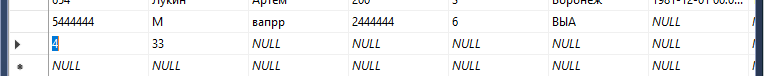
Однако изменения по прежнему доступны



Grant INSERT, select ON STUDENT TO public



Работает привелегия добавления записей



REVOKE INSERT ON STUDENT TO public (эта команда отменить привилегию у всех пользователей, имеющих эту роль)

