

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«МИРЭА—Российский технологический университет»**

**РТУ МИРЭА**

**Институт комплексной безопасности и специального приборостроения**

(наименование института, филиала)

Кафедра КБ-2 «Прикладные информационные технологии»

(наименование кафедры)

**Курсовая работа**

по дисциплине \_\_\_\_\_\_\_\_Базы данных и экспертные системы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование дисциплины)

**Тема курсовой работы** Создание клиент-серверного приложения для работы с базой данных

**Студент группы** БИСО-03-19Костерева Дарья Викторовна  **\_\_\_\_\_\_\_\_\_**

(подпись студента)

**Руководитель курсовой работы** Кульков Дмитрий Александрович **\_\_\_\_\_\_\_\_\_\_\_**

(подпись руководителя)

**Рецензент** (при наличии) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность, звание, ученая степень) (подпись рецензента)

Работа представлена к защите « 29 » мая 2021 г.

Допущен к защите «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«МИРЭА—Российский технологический университет»**

**РТУ МИРЭА**

**Институт комплексной безопасности и специального приборостроения**

(наименование института, филиала)

Кафедра КБ-2 «Прикладные информационные технологии»

(наименование кафедры)

Утверждаю

Заведующий кафедрой:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (подпись)

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине \_\_\_\_\_\_\_\_\_Базы данных и экспертные системы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование дисциплины)

**Тема курсовой работы** Создание клиент-серверного приложения для работы с базой данных**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Студент группы** БИСО-03-19Костерева Дарья Викторовна  **\_\_\_\_\_\_\_\_\_\_\_\_**

(подпись студента)

**Исходные данные** Производственная деятельность студенческих строительных отрядов **Перечень вопросов, подлежащих обработке, и обязательного графического материала:**

Создание клиента для работы с БД\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Реализация прав доступа в клиентской части\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Создание БД, таблиц, функций, процедур, триггеров индексов, транзакций\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Реализация в клиенте запросов согласно заданию\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Добавление индексов для ускорения запросов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Срок предоставление к защите курсовой работы до** «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

**Задание на курсовую работу выдал** Кульков Д.А.  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  (подпись руководителя)

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

**Задание на курсовую работу получил** Костерева Д.В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись исполнителя)

**Содержание**

Введение ……………………………………………………………………..4

Наименование и цель работы ………………………………………………5

Постановка и цель работы ………………………………………….………7

Схема базы данных …………………………………………………………10

Выполненные задания ……………………………………………………...14

Листинг программы …………………………………………………….….29

Демонстрация работы приложения ………………………………………..40

Вывод ……………………………………………………………………...…47

Список используемых источников …………………………………………48

**Введение**

С ростом количества информации, необходимой для жизнедеятельности человека, возникает острая потребность в автоматизации ее обработки и решения вопроса об организации хранения необходимых данных. За все время существования данной проблемы было предложено множество различных решений. Одним из современных и наиболее востребованным способом является использование баз данных (БД). Современные системы управления базами данных (СУБД) предоставляют обширный перечень возможностей, как разработчикам, так и рядовым пользователя.

В курсовой работе ставиться задача - разработать проект базы данных для предметной области производственной деятельности студенческих строительных отрядов. В качестве основного инструмента должно быть представлено клиентское приложение, выполняющее всю основную предметной области.

В качестве средств разработки будут использоваться СУБД Microsoft SQL Server Management Studio 18, а также фреймворк для разработки кроссплатформенного программного обеспечения на языке программирования C++ Qt.

**Наименование и цель работы**

Дисциплина «Базы данных и экспертные системы» имеет целью сформировать у обучающихся компетенции ОПК-3; ПК-14; ПК-15 в соответствии с требованиями ФГОС ВО по направлению подготовки 10.05.04 Информационно-аналитические системы безопасности «специализация N 3 «Технологии информационно-аналитического мониторинга»».

В результате изучения дисциплины обучающийся должен:

Знать:

• принципы инфологического и даталогического подходов к проектированию БД;

• этапы концептуального и логического проектирования баз данных; • классические модели представления данных;

• принципы разработки систем искусственного интеллекта, основные подходы к созданию искусственного интеллекта.

Уметь:

• проектировать реляционную базу данных, составлять программы взаимодействия с базой данных;

• разрабатывать ER-диаграммы;

• программировать на языке SQL;

• применять аппарат математической логики и математические модели из теории машинного обучения для разработки систем искусственного интеллекта.

Владеть:

• навыками разработки систем искусственного интеллекта (экспертных систем);

• навыками использования языка SQL для реализации запросов к БД;

• навыками создания прикладных программных средств для реальных БД;

• навыками использования языка SQL для реализации запросов к БД;

• навыками создания прикладных программных средств для реальных БД;

• навыками использования языка SQL для реализации запросов к БД;

• навыками создания прикладных программных средств для реальных БД.

**Постановка задачи**

Разработать клиент-серверное приложение, серверная часть которой реализована на любом SQL сервере, представляющая собой модель предметной области в соответствии с вариантом задания. В рамках заданной предметной области реализовать заданную (по варианту) схему отношений, т.е. выделить сущности и их атрибуты, так чтобы связи между сущностями соответствовали представленной схеме. В рамках курсовой работы необходимо на стороне сервера реализовать и использовать при демонстрации приложения следующие компоненты:

1. Постоянные таблицы и связи между ними, количество таблиц и наличие связей должно соответствовать заданию, допускается увеличение числа таблиц и их полей для более адекватного представления предметной области;

2. В приложении реализовать не менее пяти запросов, включая (для демонстрации навыков работы):

a. Составной многотабличный запрос с параметром, включающий соединение таблиц и CASE-выражение;

b. На основе обновляющего представления (многотабличного VIEW), в котором критерий упорядоченности задает пользователь при выполнении;

c. Запрос, содержащий коррелированные и некоррелированные подзапросы в разделах SELECT, FROM и WHERE (в каждом хотя бы по одному);

d. Многотабличный запрос, содержащий группировку записей, агрегативные функции и параметр, используемый в разделе HAVING;

e. Запрос, содержащий предикат ANY(SOME) или ALL;

3. Создать индексы для увеличения скорости выполнения запросов;

4. В таблице (в соответствии с вариантом) предусмотреть поле, которое заполняется автоматически по срабатыванию триггера при добавлении, обновлении и удалении данных, иметь возможность продемонстрировать работу триггера при работе приложения. Триггеры должны обрабатывать только те записи, которые были добавлены, изменены или удалены в ходе текущей операции (транзакции).

5. Операции добавления, удаления и обновления реализовать в виде хранимых процедур (функций) с параметрами для всех таблиц;

6. Реализовать отдельную хранимую процедуру(функцию), состоящую из нескольких отдельных операций в виде единой транзакции, которая при определенных условиях может быть зафиксирована или откатана;

7. В триггере или хранимой процедуре реализовать курсор на обновления отдельных данных;

8. В запросе (из пункта 2 или в дополнительном к тому перечню) использовать собственную скалярную функцию, а в хранимой процедуре –векторную (или табличную) функцию. Функции сохранить в базе данных

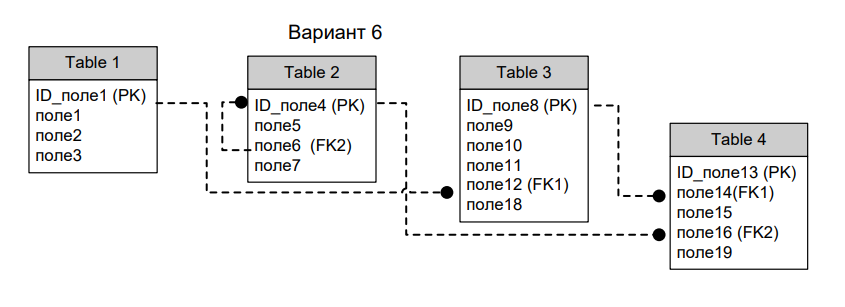
9. Распределение прав пользователей: предусмотреть не менее двух пользователей с разным набором привилегий. Каждый набор привилегий оформить в виде роли.

Клиент должен обеспечивать добавление, модификацию и удаление данных по всей предметной области. Добавление, редактирование данных в таблице производить в отдельном окне.

Запрещено в качестве вводимых данных, в том числе для связи таблиц, указывать значения первичных и внешних ключей – для обеспечения ссылочной целостности пользователь должен выбирать значения из справочника, а соответствующие значения должны подставляться программно (тем или иным способом – автоматически).

Вариант 26

|  |  |  |  |
| --- | --- | --- | --- |
| **Предметная область** | **Примерные отношения**  **предметной области** | **№ схемы**  **(вариант**  **схемы)** | **Триггер**  **(таблица № …)** |
| Производственная деятельность студенческих строительных отрядов | Какие студенты, каких групп участвуют в каком стройотряде (и где), выполненный объем и характер работ и учет оплаты за нее. | 6 | 2 |

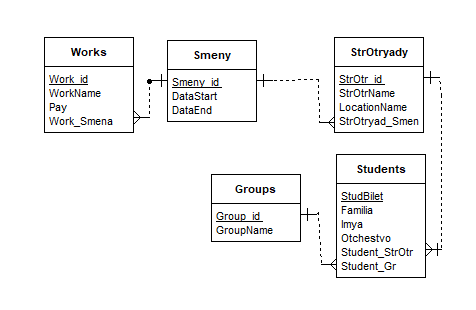


**Схема базы данных**

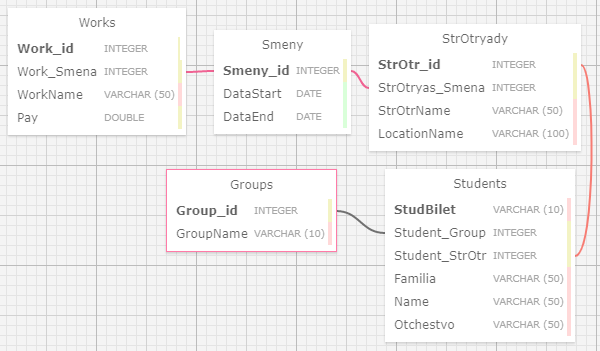
1. Постоянные таблицы и связи между ними, количество таблиц и

наличие связей должно соответствовать заданию, допускается увеличение числа таблиц и их полей для более адекватного представления предметной области.

Инфологическая модель базы данных:



Даталогическая модель базы данных:



Создание таблиц в MS SQL:

-- Смены

CREATE TABLE Smeny

(

Smeny\_id int PRIMARY KEY IDENTITY(1,1),

DataStart date,

DataEnd date,

)

GO

-- Строительные отряды

CREATE TABLE StrOtryady

(

StrOtrName NVARCHAR(50) NOT NULL,

LocationName NVARCHAR(100),

StrOtryad\_Smena int REFERENCES Smeny (Smeny\_id) ON DELETE SET DEFAULT,

StrOtr\_id int PRIMARY KEY IDENTITY(1,1),

)

GO

-- Учебные группы студентов

CREATE TABLE Groups

(

Group\_id int PRIMARY KEY IDENTITY(1,1),

GroupName NVARCHAR(50) NOT NULL,

)

GO

-- Студенты

CREATE TABLE Students

(

StudBilet NVARCHAR(50) PRIMARY KEY NOT NULL,

Familia NVARCHAR(50) NOT NULL,

Imya NVARCHAR(50) NOT NULL,

Otchestvo NVARCHAR(50),

Student\_StrOtr int REFERENCES StrOtryady (StrOtr\_id) ON DELETE SET DEFAULT,

Student\_Group int REFERENCES Groups (Group\_id) ON DELETE SET DEFAULT,

)

GO

-- Работы

CREATE TABLE Works

(

Work\_id int PRIMARY KEY IDENTITY(1,1) NOT NULL,

WorkName NVARCHAR(50),

Pay REAL,

Work\_Smena int REFERENCES Smeny (Smeny\_id) ON DELETE SET DEFAULT,

)

GO

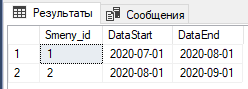
Заполнение таблиц данными.

-- Заполнение смен

INSERT INTO Smeny (DataStart,DataEnd)

VALUES ('2020-07-01', '2020-08-01'), ('2020-08-01', '2020-09-01')

GO

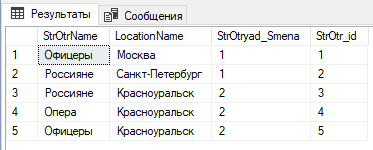


-- Заполнение таблицы стройотрядов

INSERT INTO StrOtryady (StrOtrName, LocationName, StrOtryad\_Smena)

VALUES ('Офицеры','Москва',1),('Россияне','Санкт-Петербург',1), ('Россияне','Красноуральск',2),('Опера','Красноуральск',2),('Офицеры','Красноуральск',2)

GO

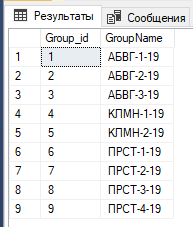


-- Заполнение таблицы учебных групп

INSERT INTO Groups (GroupName)

VALUES ('АБВГ-1-19'),('АБВГ-2-19'),('АБВГ-3-19'),('КЛМН-1-19'),('КЛМН-2-19'),('ПРСТ-1-19'),('ПРСТ-2-19'),('ПРСТ-3-19'),('ПРСТ-4-19')

GO



-- Заполнение таблицы студентов

INSERT INTO Students (StudBilet, Familia, Imya, Otchestvo, Student\_StrOtr, Student\_Group)

VALUES ('19Б001','Вейдер','Дарт','Дарт',4,1),('19Б028','Балбес','Пётр','Петрович',3,4),('19Б062','Бывалый','Фёдор','Фёдорович',3,4),

('19Б090','Трус','Иван','Иванович',3,4),('19Б111','Штирлиц','Отто','',2,6),('19Б139','Романова','Екатерина','Алексеевна',1,1),

('19Б300','Красносолнышков','Владимир','',4,3),('19Б302','Вещий','Олег','',4,8),('19Б311','Романов','Пётр','Алекчеевич',1,2),

('19Б331','Грозный','Иван','Васильевич',4,5),('19Б333','Исаев','Максим','Максимович',2,6),('19Б340','Каренина','Анна','Аркадьевна',2,7),

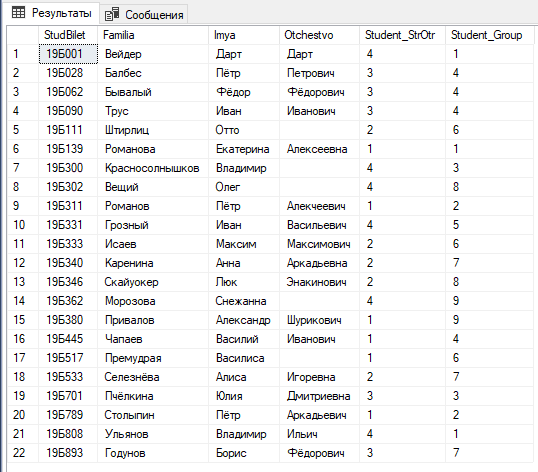
('19Б346','Скайуокер','Люк','Энакинович',2,8),('19Б362','Морозова','Снежанна','',4,9),('19Б380','Привалов','Александр','Шурикович',1,9),

('19Б445','Чапаев','Василий','Иванович',1,4),('19Б517','Премудрая','Василиса','',1,6),('19Б533','Селезнёва','Алиса','Игоревна',2,7),

('19Б701','Пчёлкина','Юлия','Дмитриевна',3,3),('19Б808','Ульянов','Владимир','Ильич',4,1),('19Б893','Годунов','Борис','Фёдорович',3,7),

('19Б789','Столыпин','Пётр','Аркадьевич',1,2)

GO

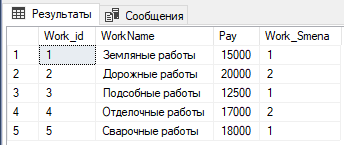


-- Заполнение таблицы работ

INSERT INTO Works (WorkName, Pay, Work\_Smena)

VALUES ('Земляные работы',15000,1),('Дорожные работы',20000,2),('Подсобные работы',12500,1),('Отделочные работы',17000,2),('Сварочные работы',18000,1)

GO



**Выполненные задания**

2. В приложении реализовать не менее пяти запросов, включая (для демонстрации навыков работы):

a. Составной многотабличный запрос с параметром, включающий соединение таблиц и CASE-выражение;

-- Какие студенты к какому направлению относятся

SELECT Distinct Familia, GroupName,

CASE

WHEN GroupName LIKE 'АБВГ%' THEN 'АБВГ'

WHEN GroupName LIKE 'КЛМН%' THEN 'КЛМН'

WHEN GroupName LIKE 'ПРСТ%' THEN 'ПРСТ'

end Napravleniya

FROM Students, Groups where Student\_Group=Group\_id order by Napravleniya;



b. На основе обновляющего представления (многотабличного VIEW), в котором критерий упорядоченности задает пользователь при выполнении;

-- Какой студент какой группы находится

-- в каком стройотряде

CREATE VIEW st\_gr\_ot AS

SELECT S.Familia as Familia, G.GroupName as Gruppa, SO.StrOtrName as StroyOtryad

FROM Groups G, Students S, StrOtryady SO

WHERE G.Group\_id=S.Student\_Group AND S.Student\_StrOtr=SO.StrOtr\_id;

SELECT \* FROM st\_gr\_ot ORDER BY Familia;



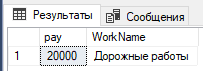
c. Запрос, содержащий коррелированные и некоррелированные подзапросы в разделах SELECT, FROM и WHERE (в каждом хотя бы по одному);

-- максимальная оплата за вторую смену

select pay ,(select workname from works where Work\_id>=1) as WorkName

from (select pay, WorkName from works where Work\_Smena=2 ) as work

where pay=(select max(pay) from works)



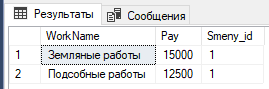
d. Многотабличный запрос, содержащий группировку записей, агрегативные функции и параметр, используемый в разделе HAVING;

-- Название работ, оплачиваемых ниже среднего, их оплаты и рабочие смены

select works.WorkName, works.Pay, smeny.Smeny\_id

from (Works join Smeny on Work\_Smena=Smeny\_id)

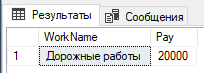
Group by workname, pay, Smeny\_id Having pay < (select avg(pay) from works);



e. Запрос, содержащий предикат ANY(SOME) или ALL;

-- Работы, прибыльнее среднего во вторую смену

SELECT Works.WorkName, Works.Pay FROM Works WHERE Works.Pay > any (SELECT avg(Works.Pay) FROM Works, Smeny where Smeny\_id=2 and Smeny\_id=Work\_Smena )



3. Создать индексы для увеличения скорости выполнения запросов;

CREATE INDEX i\_group\_name on Groups (GroupName)

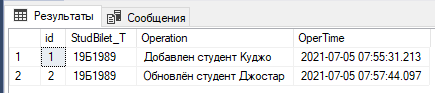
CREATE INDEX i\_student\_familiya ON Students(Familia)

CREATE INDEX i\_StrOtryad\_Location ON StrOtryady(LocationName)

CREATE INDEX i\_work\_name ON Works (WorkName)

4. В таблице (в соответствии с вариантом) предусмотреть поле, которое заполняется автоматически по срабатыванию триггера при добавлении, обновлении и удалении данных, иметь возможность продемонстрировать работу триггера при работе приложения. Триггеры должны обрабатывать только те записи, которые были добавлены, изменены или удалены в ходе текущей операции (транзакции).

Триггер создан к таблице Students.



CREATE TABLE Timing

(

id INT PRIMARY KEY IDENTITY NOT NULL,

StudBilet\_T NVARCHAR(50) NOT NULL,

Operation NVARCHAR(50) NOT NULL,

OperTime DATETIME NOT NULL DEFAULT GETDATE()

)GO

CREATE TRIGGER tr\_add on [dbo].[Students] AFTER INSERT AS

INSERT INTO Timing (StudBilet\_T, Operation)

SELECT StudBilet, 'Добавлен студент ' + Familia FROM inserted

GO

CREATE TRIGGER tr\_upd on [dbo].[Students] AFTER UPDATE AS

INSERT INTO Timing (StudBilet\_T, Operation)

SELECT StudBilet, 'Обновлён студент ' + Familia FROM inserted

GO

CREATE TRIGGER tr\_del on [dbo].[Students] AFTER INSERT AS

INSERT INTO Timing (StudBilet\_T, Operation)

SELECT StudBilet, 'Удалён студент ' + Familia FROM deleted

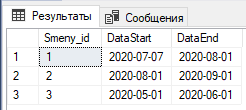
GO

5. Операции добавления, удаления и обновления реализовать в виде хранимых процедур (функций) с параметрами для всех таблиц;

-- Добавляем / обновляем / удаляем смену по дате начала (все смены длятся одинаковое количество дней)

CREATE PROCEDURE INSERTsmena (@Smeny\_id INT, @DataStart DATE, @DataEnd DATE) AS BEGIN INSERT INTO Smeny (Smeny\_id, DataStart,DataEnd) VALUES (@Smeny\_id, @DataStart, @DataEnd) END;

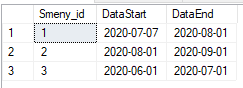
GO



CREATE PROCEDURE UPDATEsmena (@Smeny\_id INT, @DataStart DATE, @DataEnd DATE) AS BEGIN UPDATE Smeny SET DataStart=@DataStart, DataEnd=@DataEnd

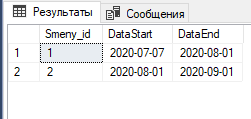
WHERE Smeny\_id=@Smeny\_id END;

GO



CREATE PROCEDURE DELETEsmena (@Smeny\_id INT) AS BEGIN DELETE Smeny WHERE Smeny\_id=@Smeny\_id END;

GO



EXEC INSERTsmena '2020-05-01','2020-06-01';

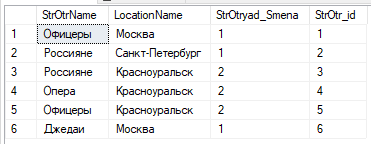
EXEC UPDATEsmena '2020-05-01', '2020-06-01', '2020-07-01';

EXEC DELETEsmena '2020-06-01';

-- Добавляем / обновляем / удаляем строительные отряды по их названию

CREATE PROCEDURE INSERTstrotr (@StrOtrName NVARCHAR(50), @LocationName NVARCHAR(50), @StrOtryad\_Smena int) AS BEGIN INSERT INTO StrOtryady(StrOtrName, LocationName, StrOtryad\_Smena) VALUES ((@StrOtrName, @LocationName, @StrOtryad\_Smena) END;

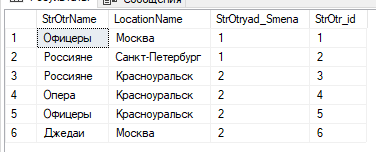
GO



CREATE PROCEDURE UPDATEstrotr (@StrOtrName NVARCHAR(50), @LocationName NVARCHAR(50), @StrOtryad\_Smena int) AS BEGIN UPDATE StrOtryady SET LocationName=@LocationName, StrOtryad\_Smena=@StrOtryad\_Smena

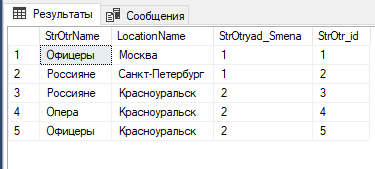
where StrOtrName=@StrOtrName END;

GO



CREATE PROCEDURE DELETEstrotr (@StrOtrName NVARCHAR(50)) AS BEGIN DELETE StrOtryady WHERE StrOtrName=@StrOtrName END;

GO



EXEC INSERTstrotr 'Джедаи', 'Москва', 1;

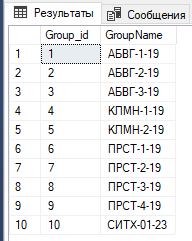
EXEC UPDATEstrotr 'Джедаи', 'Москва', 2;

EXEC DELETEstrotr 'Джедаи';

-- Добавляем / обновляем / удаляем учебные группы студентов их названию

CREATE PROCEDURE INSERTgroup (@GroupName NVARCHAR(50)) AS BEGIN INSERT INTO Groups (GroupName) VALUES (@GroupName) END;

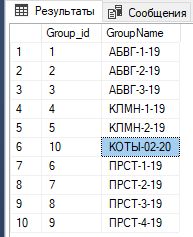
GO



CREATE PROCEDURE UPDATEgroup (@GroupName NVARCHAR(50), @GroupName\_new NVARCHAR(50)) AS BEGIN UPDATE Groups SET GroupName=@GroupName\_new

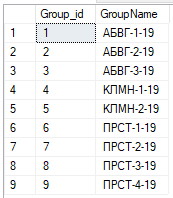
where GroupName=@GroupName END;

GO



CREATE PROCEDURE DELETEgroup (@GroupName NVARCHAR(50)) AS BEGIN DELETE Groups WHERE GroupName=@GroupName END;

GO



EXEC INSERTgroup 'СИТХ-01-23';

EXEC UPDATEgroup 'СИТХ-01-23','КОТЫ-02-20';

EXEC DELETEgroup 'КОТЫ-02-20';

-- Добавляем / обновляем / удаляем студентов по студ.билету

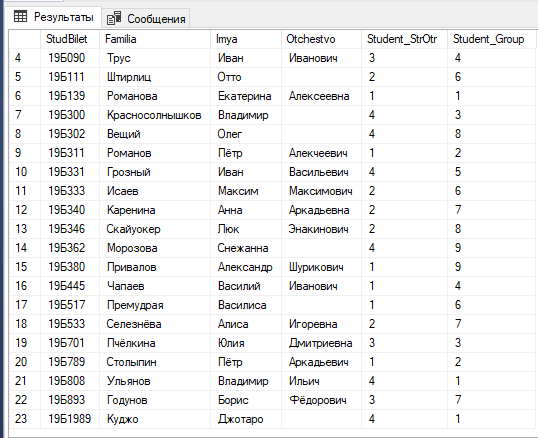
CREATE PROCEDURE INSERTstudent (@StudBilet NVARCHAR(50), @Familia NVARCHAR(50), @Imya NVARCHAR(50), @Otchestvo NVARCHAR(50), @StrOtrName NVARCHAR(50), @GroupName NVARCHAR(50)) AS BEGIN

INSERT INTO Students(StudBilet, Familia, Imya, Otchestvo, Student\_StrOtr, Student\_Group) VALUES(@StudBilet, @Familia, @Imya, @Otchestvo, (select distinct Students.Student\_StrOtr from students, StrOtryady where @StrOtrName=StrOtrName and StrOtr\_id=Student\_StrOtr),

(select distinct Groups.GroupId from Groups, Students where @GroupName=GroupName and Group\_id=Student\_Group))

END;

GO



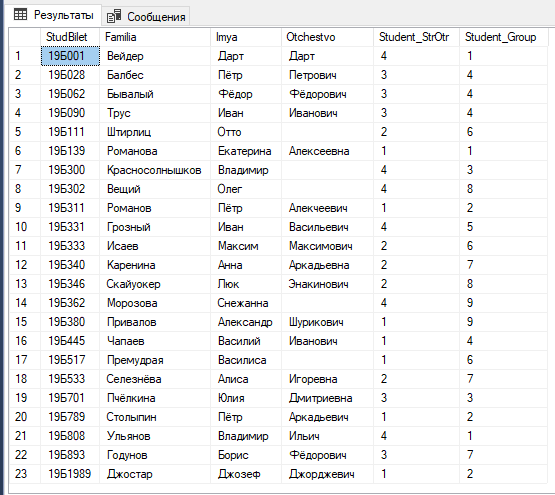
CREATE PROCEDURE UPDATEstudent (@StudBilet NVARCHAR(50), @Familia NVARCHAR(50), @Imya NVARCHAR(50), @Otchestvo NVARCHAR(50), @StrOtrName NVARCHAR(50), @GroupName NVARCHAR(50)) AS BEGIN

UPDATE Students(StudBilet, Familia, Imya, Otchestvo, Student\_StrOtr, Student\_Group) SET Familia=@Familia, Imya=@Imya, Otchestvo=@Otchestvo, Student\_StrOtr=(select Students.Student\_StrOtr from students, StrOtryady where @StrOtrName=StrOtrName and StrOtr\_id=Student\_StrOtr),

Student\_Group=(select Groups.GroupId from Groups, Students where @GroupName=GroupName and Group\_id=Student\_Group) where StudBilet=@StudBilet

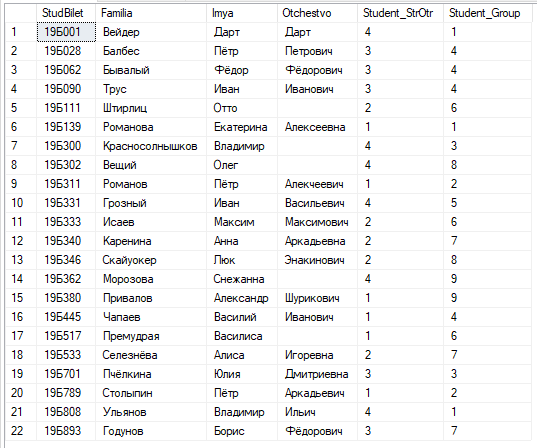
END;

GO



CREATE PROCEDURE DELETEstudent (@StudBilet NVARCHAR(50)) AS BEGIN DELETE Students WHERE StudBilet=@StudBilet END;

GO



EXEC INSERTstudent '19Б1989','Куджо','Джотаро','','Опера','АБВГ-1-19';

EXEC UPDATEstudent '19Б1989','Джостар','Джозеф','Джорджевич','Офицеры','АБВГ-2-19';

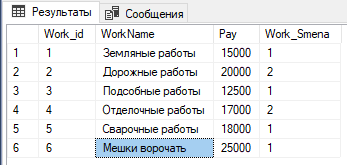
EXEC DELETEstudent '19Б1989';

-- Добавляем / обновляем / удаляем работы по их названию

CREATE PROCEDURE INSERTwork ( @WorkName NVARCHAR(50), @Pay REAL, @Smeny\_id INT) AS BEGIN INSERT INTO Works (WorkName,Pay,Work\_Smena)

VALUES (@WorkName, @Pay, @Smeny\_id) END;

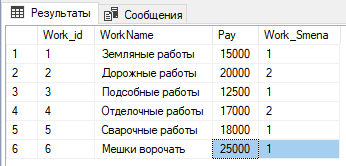
GO



CREATE PROCEDURE UPDATEwork ( @WorkName NVARCHAR(50), @Pay REAL, @Smeny\_id INT) AS BEGIN UPDATE Works SET Pay=@Pay, Work\_Smena=@Smeny\_id

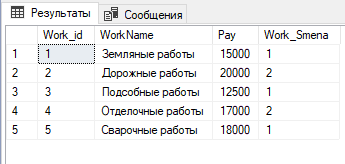
WHERE WorkName=@WorkName END;

GO



CREATE PROCEDURE DELETEwork (@WorkName NVARCHAR(50)) AS BEGIN DELETE Works WHERE WorkName=@WorkName END;

GO



EXEC INSERTwork 'Мешки ворочать', 25000, 1;

EXEC UPDATEwork 'Мешки ворочать',22500,2;

EXEC DELETEwork 'Мешки ворочать';

6. Реализовать отдельную хранимую процедуру(функцию), состоящую из нескольких отдельных операций в виде единой транзакции, которая при определенных условиях может быть зафиксирована или откатана;

-- Проверка отсутствия цифр в ФИО

CREATE PROCEDURE StudCheck (@StudBilet NVARCHAR(50), @Familia NVARCHAR(50), @Imya NVARCHAR(50), @Otchestvo NVARCHAR(50)) AS BEGIN

BEGIN TRANSACTION

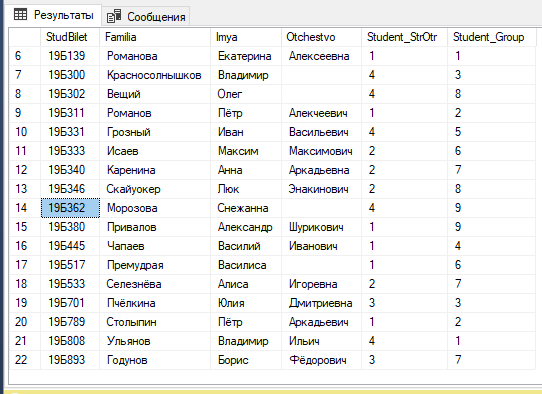
UPDATE Students SET Familia=@Familia, Imya=@Imya, Otchestvo=@Otchestvo WHERE StudBilet=@StudBilet;

IF (ISNUMERIC(@Familia)=1 OR ISNUMERIC(@Imya)=1 OR ISNUMERIC(@Otchestvo)=1) ROLLBACK;

ELSE COMMIT TRANSACTION

END;

GO

****

EXEC StudCheck '19Б362','К0стерева','Дарья','Викторовна';

****

EXEC StudCheck '19Б362','Изотопов','Распад','';

7. В триггере или хранимой процедуре реализовать курсор на обновления отдельных данных;

-- Зарплата в сменах, где количество работников меньше половины их полного количества, увеличивается на 1234 рубля

CREATE PROCEDURE payCursor AS BEGIN DECLARE @WorkName NVARCHAR(50), @Pay REAL,@Work\_Smena INT,@Studbilet INT

DECLARE P\_Cur CURSOR FOR SELECT Works.WorkName, Works.Pay, Works.Work\_Smena FROM Works OPEN P\_Cur

FETCH NEXT FROM P\_Cur INTO @WorkName, @Pay, @Work\_Smena

WHILE @@FETCH\_STATUS = 0

BEGIN IF (SELECT COUNT(@WorkName) FROM Works , SMENY WHERE Works.Work\_Smena=@Work\_Smena) < (SELECT COUNT(@WorkName)/2 FROM Works, Smeny WHERE Work\_Smena LIKE '%')

UPDATE WORKS SET Pay=Pay + 1234

WHERE WorkName=@WorkName

FETCH NEXT FROM P\_Cur INTO @WorkName, @Pay, @Work\_Smena

END

CLOSE P\_CUR

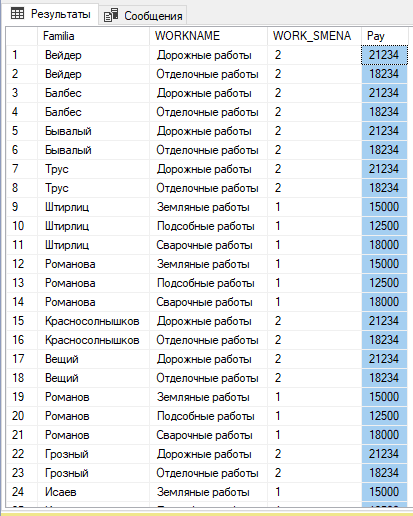
DEALLOCATE P\_Cur

END

GO

EXEC payCursor

SELECT S.Familia, W.WORKNAME, W.WORK\_SMENA, W.Pay FROM WORKS W, STUDENTS S, Smeny SM, StrOtryady ST WHERE Work\_Smena=Smeny\_id AND Smeny\_id=StrOtryad\_Smena AND StrOtr\_id=Student\_StrOtr



8. В запросе (из пункта 2 или в дополнительном к тому перечню) использовать собственную скалярную функцию, а в хранимой процедуре –векторную (или табличную) функцию. Функции сохранить в базе данных

--скалярная функция - итоговый заработок студентов

CREATE FUNCTION dbo.PaySum (@pay real) RETURNS REAL

AS BEGIN

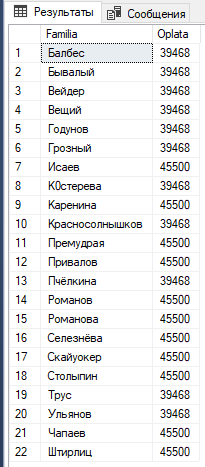
DECLARE @RESULT REAL, @studbilet NVARCHAR(50)

SELECT @RESULT=SUM(Pay) FROM WORKS W, STUDENTS S, Smeny SM, StrOtryady ST WHERE Work\_Smena=Smeny\_id AND Smeny\_id=StrOtryad\_Smena AND StrOtr\_id=Student\_StrOtr

return @result;

END;

SELECT distinct Familia, Oplata=dbo.paysum(@pay) FROM WORKS, STUDENTS, Smeny, StrOtryady WHERE Work\_Smena=Smeny\_id AND Smeny\_id=StrOtryad\_Smena AND StrOtr\_id=Student\_StrOtr



--табличная функция (ссылается на пункт 6, ФИО, не содержащие цифр)

CREATE FUNCTION [dbo].[f\_table]

(@STROTR INT)

RETURNS TABLE AS RETURN

(SELECT \* FROM STUDENTS WHERE @STROTR=STUDENT\_STROTR)

GO

--DROP FUNCTION [dbo].[f\_table]

CREATE PROCEDURE dbo.StudCheckTable (@StudBilet NVARCHAR(50), @Familia NVARCHAR(50), @Imya NVARCHAR(50), @Otchestvo NVARCHAR(50)) AS BEGIN

BEGIN TRANSACTION

UPDATE Students SET Familia=@Familia, Imya=@Imya, Otchestvo=@Otchestvo WHERE StudBilet=@StudBilet;

--IF (ISNUMERIC(@Familia)=1 OR ISNUMERIC(@Imya)=1 OR ISNUMERIC(@Otchestvo)=1) ROLLBACK;

if ( @Familia like '%[0123456789]%' or @Imya like '%[0123456789]%' or @Otchestvo like '%[0123456789]%') ROLLBACK;

ELSE COMMIT TRANSACTION;

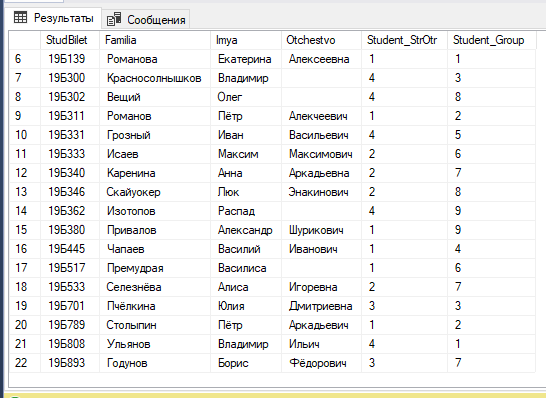
IF ( @Familia not like '%[0123456789]%' and @Imya not like '%[0123456789]%' and @Otchestvo not like '%[0123456789]%')

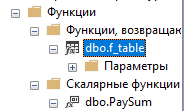
select \* from [dbo].[F\_TABLE] (1)

END;

GO

EXEC dbo.StudCheckTable '19Б362','иван0в','Иван','Иван0вич';



****

9. Распределение прав пользователей: предусмотреть не менее двух пользователей с разным набором привилегий. Каждый набор привилегий оформить в виде роли.

CREATE LOGIN ADDmin WITH PASSWORD='1'

CREATE USER ADDmin FOR LOGIN ADDmin;

CREATE LOGIN ADDmax WITH PASSWORD='admin'

CREATE USER ADDmax FOR LOGIN ADDmax;

CREATE ROLE ODMEN;

CREATE ROLE UZVER;

--Предоставить привилегии на роль

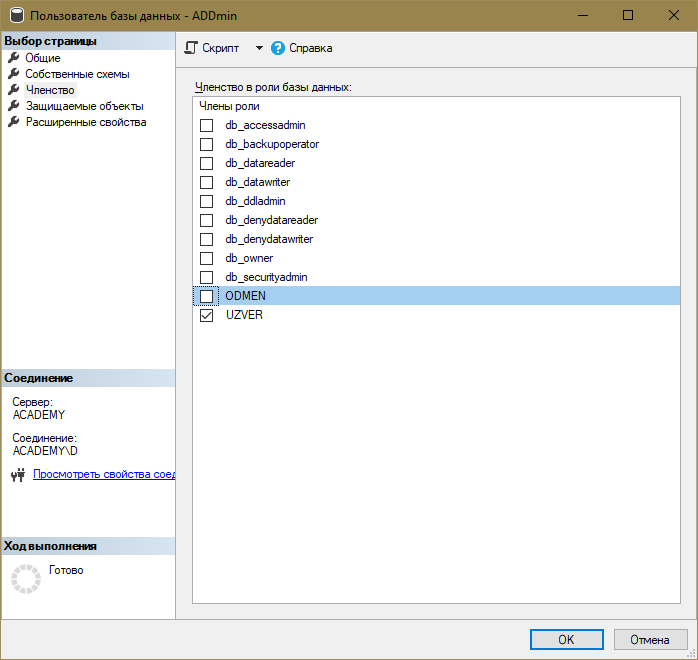
GRANT SELECT, UPDATE, INSERT, DELETE ON Students TO ODMEN;

GO

GRANT SELECT, UPDATE, INSERT, DELETE ON StrOtryady TO ODMEN;

GO

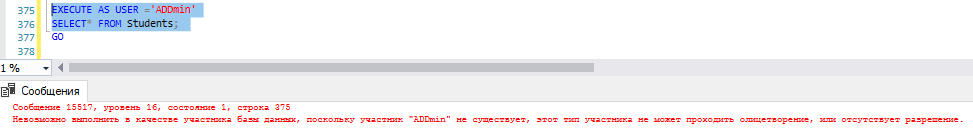
--Предоставить роль пользователям

****

--Отзыв привилегии из роли:

REVOKE SELECT ON Students FROM ODMEN;

REVOKE INSERT ON StrOtryady FROM UZVER;

****

**Листинг программы**

**main.cpp**

#include "mainwindow.h"

#include <QApplication>

int **main**(int argc, char \*argv[])

{

QApplication a(*argc*, *argv*);

MainWindow w;

w.show();

return a.exec();

}

**mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(*parent*)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~***MainWindow***()

{

delete ui;

}

void MainWindow::**on\_BDconnect\_triggered**()

{

logwin = new login();

logwin->show();

}

void MainWindow::**on\_tableStudents\_clicked**()

{

studwin = new tableStudents();

studwin->show();

}

void MainWindow::**on\_tableSmeny\_clicked**()

{

smenywin = new tableSmeny();

smenywin->show();

}

void MainWindow::**on\_tableStrOtr\_clicked**()

{

strotrwin = new tableStrOtryady();

strotrwin->show();

}

void MainWindow::**on\_tableGroups\_clicked**()

{

groupwin = new tableGroups();

groupwin->show();

}

void MainWindow::**on\_tableWorks\_clicked**()

{

workwin = new tableWorks();

workwin->show();

}

**login.cpp**

#include "login.h"

#include "ui\_login.h"

login::**login**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::login)

{

ui->setupUi(this);

ui->lineEdit\_2->setEchoMode(QLineEdit::Password);

message = new QMessageBox();

}

login::~***login***()

{

delete ui;

}

void login::**on\_DB\_Connected\_clicked**()

{

db = QSqlDatabase::addDatabase("QODBC");

db.setDatabaseName("DRIVER={SQL Server};SERVER=ACADEMY;DATABASE=Cursach;");

db.setUserName(ui->lineEdit->text());

db.setPassword(ui->lineEdit\_2->text());

if(db.open())

{

message->setText("Соединение успешно");

}

else

{

message->setText("Соединение не установлено");

}

message->show();

}

**tablegroups.cpp**

#include "tablegroups.h"

#include "ui\_tablegroups.h"

#include <QDebug>

tableGroups::**tableGroups**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::tableGroups)

{

ui->setupUi(this);

}

tableGroups::~***tableGroups***()

{

delete ui;

}

void tableGroups::**on\_update\_clicked**()

{

groups\_model = new QSqlQueryModel();

groups\_model->setQuery("SELECT GroupName FROM Groups");

groups\_model->*setHeaderData*(0,Qt::Horizontal,"Название группы");

ui->tableView->*setModel*(*groups\_model*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void tableGroups::**on\_Change\_clicked**()

{

QSqlQuery\* query=new QSqlQuery();

query->prepare("UPDATE Groups SET GroupName=:GroupName\_new where GroupName=:GroupName");

query->bindValue(":GroupName\_new",ui->lineEdit->text());

query->bindValue(":GroupName",prim\_key);

query->exec();

on\_update\_clicked();

}

void tableGroups::**on\_Delete\_clicked**()

{

QSqlQuery\* query=new QSqlQuery();

query->prepare("DELETE FROM Groups WHERE GroupName=:GroupName");

query->bindValue(":GroupName",ui->lineEdit->text());

query->exec();

on\_update\_clicked();

}

void tableGroups::**on\_Add\_clicked**()

{

QSqlQuery\* query=new QSqlQuery();

query->prepare("INSERT INTO Groups (GroupName) VALUES (:GroupName)");

query->bindValue(":GroupName",ui->lineEdit->text());

QMessageBox\*mes=new QMessageBox();

if (query->exec()){

mes->setText(" Запись добавлена. ");

mes->show();}

else {mes->setText(" Ошибка при добавлении. ");

mes->show();};

on\_update\_clicked();

}

void tableGroups::**on\_tableView\_clicked**(const QModelIndex &index)

{

prim\_key=ui->tableView->model()->*data*(ui->tableView->model()->*index*(index.row(),0)).toString();

qDebug()<<prim\_key;

ui->lineEdit->setText(prim\_key);

QSqlQuery\* query=new QSqlQuery();

query->prepare("SELECT GroupName FROM Groups WHERE GroupName=:GroupName");

query->bindValue(":GroupName", prim\_key);

}

#include "tablesmeny.h"

#include "ui\_tablesmeny.h"

tableSmeny::**tableSmeny**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::tableSmeny)

{

ui->setupUi(this);

ui->dateEdit->setDate(QDate::currentDate());

ui->dateEdit\_2->setDate(QDate::currentDate());

}

tableSmeny::~***tableSmeny***()

{

delete ui;

}

void tableSmeny::**on\_update\_clicked**()

{

smeny\_model = new QSqlQueryModel();

smeny\_model->setQuery("SELECT \* FROM Smeny");

smeny\_model->*setHeaderData*(0,Qt::Horizontal,"Номер смены");

smeny\_model->*setHeaderData*(1,Qt::Horizontal,"Дата начала");

smeny\_model->*setHeaderData*(2,Qt::Horizontal,"Дата окончания");

ui->tableView->*setModel*(*smeny\_model*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void tableSmeny::**on\_Change\_clicked**()

{

QSqlQuery\* query=new QSqlQuery();

query->prepare("UPDATE Smeny SET DataStart=:DataStart, DataEnd=:DataEnd WHERE :DataStart=DataStart");

query->bindValue(":DataStart",ui->dateEdit->text());

query->bindValue(":DataEnd",ui->dateEdit\_2->text());

query->exec();

on\_update\_clicked();

}

**tablesmeny.cpp**

void tableSmeny::**on\_Delete\_clicked**()

{

QSqlQuery\* query=new QSqlQuery();

query->prepare("DELETE FROM Smeny WHERE DataStart=:DataStart");

query->bindValue(":DataStart",ui->dateEdit->text());

query->exec();

on\_update\_clicked();

}

void tableSmeny::**on\_Add\_clicked**()

{

QSqlQuery\* query=new QSqlQuery();

query->prepare("INSERT INTO Smeny (DataStart,DataEnd) VALUES ( :DataStart, :DataEnd)");

query->bindValue(":DataStart",ui->dateEdit->text());

query->bindValue(":DataEnd",ui->dateEdit\_2->text());

QMessageBox\*mes=new QMessageBox();

if (query->exec()){

mes->setText(" Запись добавлена. ");

mes->show();}

else {mes->setText(" Ошибка при добавлении. ");

mes->show();};

on\_update\_clicked();

}

void tableSmeny::**on\_tableView\_clicked**(const QModelIndex &index)

{

int prim\_key;

prim\_key=ui->tableView->model()->*data*(ui->tableView->model()->*index*(index.row(),0)).toInt();

ui->ID\_Smeny->setText(QString::number(prim\_key));

QSqlQuery\* query=new QSqlQuery();

query->prepare("SELECT \* FROM Smeny WHERE Smeny\_id=:Smeny\_id");

query->bindValue(":Smeny\_id",prim\_key);

if (query->exec()){

query->next();

ui->dateEdit->setDate(QDate(query->value(1).toDate()));

ui->dateEdit\_2->setDate(QDate(query->value(2).toDate()));

}

}

**tablestroyotrady.cpp**

#include "tablestrotryady.h"

#include "ui\_tablestrotryady.h"

#include <QString>

tableStrOtryady::**tableStrOtryady**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::tableStrOtryady)

{

ui->setupUi(this);

QSqlQuery\* query = new QSqlQuery();

QSqlQuery\* query2 = new QSqlQuery();

query->exec("SELECT LocationName FROM StrOtryady");

while (query->next())

{

ui->comboBox\_2->addItem(query->value(0).toString());

}

query2->exec("SELECT Smeny\_id FROM Smeny");

while (query2->next())

{

ui->comboBox\_3->addItem(query2->value(0).toString());

}

location=0;

smena=0;

}

tableStrOtryady::~***tableStrOtryady***()

{

delete ui;

}

void tableStrOtryady::**on\_update\_clicked**()

{

strotr\_model = new QSqlQueryModel();

strotr\_model->setQuery("SELECT StrOtrName,LocationName, StrOtryad\_Smena FROM StrOtryady");

strotr\_model->*setHeaderData*(0,Qt::Horizontal,"Название строй.отряда");

strotr\_model->*setHeaderData*(1,Qt::Horizontal,"Местоположение");

strotr\_model->*setHeaderData*(2,Qt::Horizontal,"Смена");

ui->tableView->*setModel*(*strotr\_model*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void tableStrOtryady::**on\_Change\_clicked**()

{

QSqlQuery\* query=new QSqlQuery();

query->prepare("UPDATE StrOtryady SET LocationName=:LocationName, StrOtryad\_Smena=:StrOtryad\_Smena WHERE StrOtrName=:StrOtrName");

query->bindValue(":StrOtrName",ui->lineEdit->text());

query->bindValue(":LocationName",ui->comboBox\_2->itemText(location));

query->bindValue(":StrOtryad\_Smena",ui->comboBox\_3->itemText(smena));

query->exec();

on\_update\_clicked();

}

void tableStrOtryady::**on\_Delete\_clicked**()

{

QSqlQuery\* query=new QSqlQuery();

query->prepare("DELETE FROM StrOtryady WHERE StrOtrName=:StrOtrName");

query->bindValue(":StrOtrName", ui->lineEdit->text());

query->exec();

on\_update\_clicked();

}

void tableStrOtryady::**on\_Add\_clicked**()

{

QSqlQuery\* query=new QSqlQuery();

query->prepare("INSERT INTO StrOtryady(StrOtrName, LocationName, StrOtryad\_Smena) VALUES (:StrOtrName, :LocationName, :StrOtryad\_Smena)");

query->bindValue(":StrOtrName",ui->lineEdit->text());

query->bindValue(":LocationName",ui->comboBox\_2->itemText(location));

query->bindValue(":StrOtryad\_Smena",ui->comboBox\_3->itemText(smena));

QMessageBox\*mes=new QMessageBox();

if (query->exec()){

mes->setText(" Запись добавлена. ");

mes->show();}

else {mes->setText(" Ошибка при добавлении. ");

mes->show();};

on\_update\_clicked();

}

void tableStrOtryady::**on\_tableView\_clicked**(const QModelIndex &index)

{

QString prim\_key;

prim\_key=ui->tableView->model()->*data*(ui->tableView->model()->*index*(index.row(),0)).toString();

ui->lineEdit->setText(prim\_key);

QSqlQuery\* query=new QSqlQuery();

query->prepare("SELECT StrOtrName, LocationName, StrOtryad\_Smena FROM StrOtryady WHERE StrOtrName=:StrOtrName");

query->bindValue(":StrOtrName",prim\_key);

}

void tableStrOtryady::**on\_comboBox\_2\_currentIndexChanged**(int index)

{

location=index;

}

void tableStrOtryady::**on\_comboBox\_3\_currentIndexChanged**(int index)

{

smena=index;

}

**tablestudents.cpp**

#include "tablestudents.h"

#include "ui\_tablestudents.h"

#include <QString>

tableStudents::**tableStudents**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::tableStudents)

{

ui->setupUi(this);

QSqlQuery\* query = new QSqlQuery();

QSqlQuery\* query2 = new QSqlQuery();

query->exec("SELECT StrOtrName FROM StrOtryady");

while (query->next())

{

ui->comboStrOtr->addItem(query->value(0).toString());

}

query2->exec("SELECT GroupName FROM Groups");

while (query2->next())

{

ui->comboGroup->addItem(query2->value(0).toString());

}

combo\_group=0;

combo\_strotr=0;

}

tableStudents::~***tableStudents***()

{

delete ui;

}

void tableStudents::**on\_update\_clicked**()

{

stud\_model = new QSqlQueryModel();

stud\_model->setQuery("SELECT StudBilet,Familia, Imya, Otchestvo FROM Students");

stud\_model->*setHeaderData*(0,Qt::Horizontal,"Номер студ.билета");

stud\_model->*setHeaderData*(1,Qt::Horizontal,"Фамилия");

stud\_model->*setHeaderData*(2,Qt::Horizontal,"Имя");

stud\_model->*setHeaderData*(3,Qt::Horizontal,"Отчество");

ui->tableView->*setModel*(*stud\_model*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void tableStudents::**on\_Add\_clicked**()

{

int id=0, id\_g=0;

QSqlQuery\* query ,\*query2,\*query3=new QSqlQuery();

//query->prepare("INSERT INTO Students (StudBilet, Familia, Imya, Otchestvo) VALUES(:StudBilet, :Familia, :Imya, :Otchestvo)");

query->prepare("INSERT INTO Students (StudBilet, Familia, Imya, Otchestvo,Student\_StrOtr,Student\_Group) VALUES(:StudBilet, :Familia, :Imya, :Otchestvo, :Student\_StrOtr, :Student\_Group)");

query->bindValue(":StudBilet",ui->lineEdit->text());

query->bindValue(":Familia",ui->lineEdit\_2->text());

query->bindValue(":Imya",ui->lineEdit\_3->text());

query->bindValue(":Otchestvo",ui->lineEdit\_4->text());

query2->prepare("SELECT StrOtrName, StrOtr\_id FROM StrOtryady where StrOtrName=:StrOtrName");

query2->bindValue(":StrOtrName",ui->comboStrOtr->itemText(combo\_strotr));

query2->bindValue("StrOtr\_id",id);

query3->prepare("SELECT GroupName, Group\_id FROM Groups where GroupName=:GroupName");

query3->bindValue(":GroupName",ui->comboGroup->itemText(combo\_group));

query2->bindValue("Group\_id",id\_g);

query->bindValue(":Student\_StrOtr",id);

query->bindValue(":Student\_Group",id\_g);

QMessageBox\*mes=new QMessageBox();

if (query->exec()){

mes->setText(" Запись добавлена. ");

mes->show();}

else {mes->setText(" Ошибка при добавлении. ");

mes->show();};

on\_update\_clicked();

}

void tableStudents::**on\_Delete\_clicked**()

{

QSqlQuery\* query=new QSqlQuery();

query->prepare("DELETE FROM Students WHERE StudBilet=:StudBilet");

query->bindValue(":StudBilet",ui->lineEdit->text());

query->exec();

on\_update\_clicked();

}

void tableStudents::**on\_Change\_clicked**()

{

QSqlQuery\* query=new QSqlQuery();

query->prepare("UPDATE Students SET Familia=:Familia, Imya=:Imya, Otchestvo=:Otchestvo WHERE StudBilet=:StudBilet");

query->bindValue(":StudBilet",ui->lineEdit->text());

query->bindValue(":Familia",ui->lineEdit\_2->text());

query->bindValue(":Imya",ui->lineEdit\_3->text());

query->bindValue(":Otchestvo",ui->lineEdit\_4->text());

query->exec();

on\_update\_clicked();

}

void tableStudents::**on\_tableView\_clicked**(const QModelIndex &index)

{

QString prim\_key;

prim\_key=ui->tableView->model()->*data*(ui->tableView->model()->*index*(index.row(),0)).toString();

ui->lineEdit->setText(prim\_key);

QSqlQuery\* query=new QSqlQuery();

query->prepare("SELECT Familia,Imya, Otchestvo FROM Students WHERE StudBilet=:StudBilet");

query->bindValue(":StudBilet",prim\_key);

if (query->exec()){

query->next();

ui->lineEdit\_2->setText(query->value(0).toString());

ui->lineEdit\_3->setText(query->value(1).toString());

ui->lineEdit\_4->setText(query->value(2).toString());

}

}

void tableStudents::**on\_comboGroup\_currentIndexChanged**(int index)

{

combo\_group=index;

}

void tableStudents::**on\_comboStrOtr\_currentIndexChanged**(int index)

{

combo\_strotr=index;

}

**tableworks.cpp**

#include "tableworks.h"

#include "ui\_tableworks.h"

#include <QString>

tableWorks::**tableWorks**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::tableWorks)

{

ui->setupUi(this);

QSqlQuery\* query = new QSqlQuery();

query->exec("SELECT Smeny\_id FROM Smeny");

while (query->next())

{

ui->comboBox\_2->addItem(query->value(0).toString());

}

combo\_id=0;

}

tableWorks::~***tableWorks***()

{

delete ui;

}

void tableWorks::**on\_tableView\_clicked**(const QModelIndex &index)

{

QString prim\_key;

prim\_key=ui->tableView->model()->*data*(ui->tableView->model()->*index*(index.row(),0)).toString();

ui->lineEdit->setText(prim\_key);

QSqlQuery\* query=new QSqlQuery();

query->prepare("SELECT WorkName,Pay, Work\_Smena FROM Works WHERE WorkName=:WorkName");

query->bindValue(":WorkName",prim\_key);

}

void tableWorks::**on\_update\_clicked**()

{

work\_model = new QSqlQueryModel();

work\_model->setQuery("SELECT WorkName,Pay,Work\_Smena FROM Works");

work\_model->*setHeaderData*(0,Qt::Horizontal,"Название работы");

work\_model->*setHeaderData*(1,Qt::Horizontal,"Оплата");

work\_model->*setHeaderData*(2,Qt::Horizontal,"Номер смены");

ui->tableView->*setModel*(*work\_model*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void tableWorks::**on\_Change\_clicked**()

{

QSqlQuery\* query=new QSqlQuery();

query->prepare("UPDATE Works SET Pay=:Pay, Work\_Smena=:Smeny\_id WHERE WorkName=:WorkName");

query->bindValue(":Pay",ui->lineEdit\_2->text());

query->bindValue(":WorkName",ui->lineEdit->text());

query->bindValue(":Smeny\_id",ui->comboBox\_2->itemText(combo\_id));

query->exec();

on\_update\_clicked();

}

void tableWorks::**on\_Delete\_clicked**()

{

QSqlQuery\* query=new QSqlQuery();

query->prepare("DELETE FROM Works WHERE WorkName=:WorkName");

query->bindValue(":WorkName",ui->lineEdit->text());

query->exec();

on\_update\_clicked();

}

void tableWorks::**on\_Add\_clicked**()

{

QSqlQuery\* query=new QSqlQuery();

query->prepare("INSERT INTO Works (WorkName,Pay,Work\_Smena) VALUES (:WorkName, :Pay, :Smeny\_id)");

query->bindValue(":WorkName",ui->lineEdit->text());

query->bindValue(":Pay",ui->lineEdit\_2->text());

query->bindValue(":Smeny\_id",ui->comboBox\_2->itemText(combo\_id));

query->exec();

on\_update\_clicked();

}

void tableWorks::**on\_comboBox\_2\_currentIndexChanged**(int index)

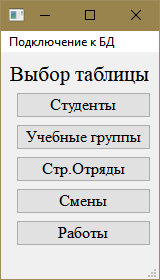
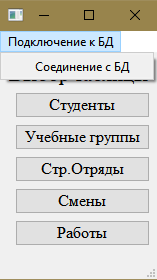
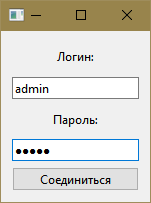
{

combo\_id=index;

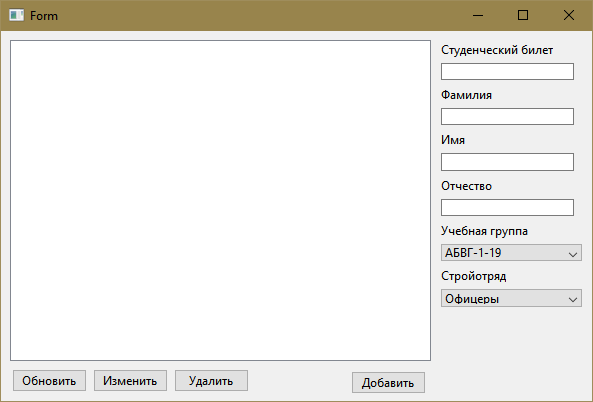
}

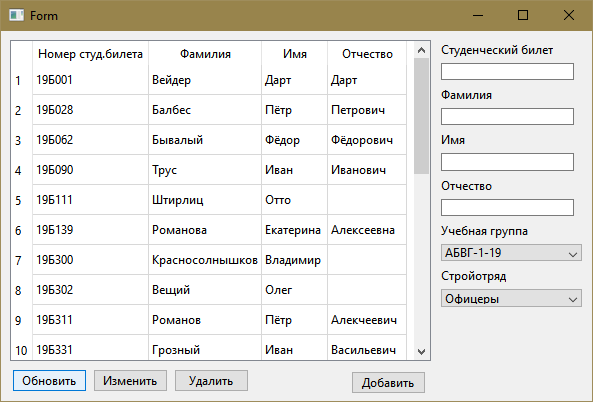
**Демонстрация работы приложения**

При запуске открывается контекстное меню, из которого можно подключиться к базе данных, введя пароль и логии от неё, и выбрать нужную таблицу.

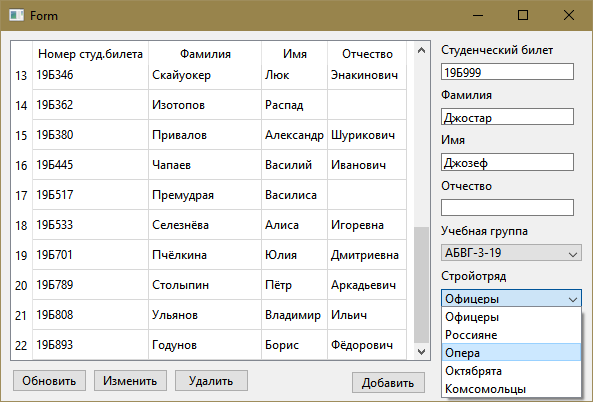
  

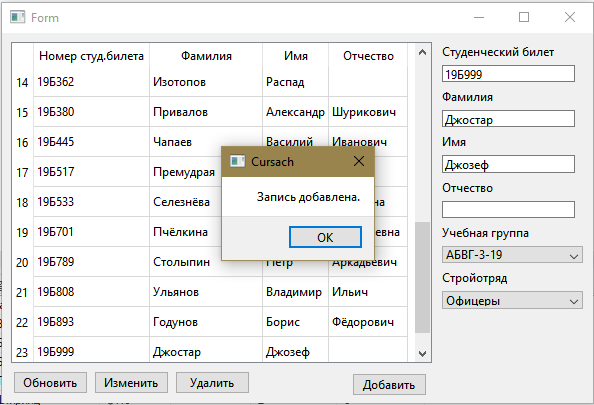
Таблицы представляют из себя окна с кнопками и пустой областью. При нажатии на кнопку «Обновить» данные из соответствующей таблицы поступают в выбранное окно. Рассмотрим на примере таблицы «Студенты».



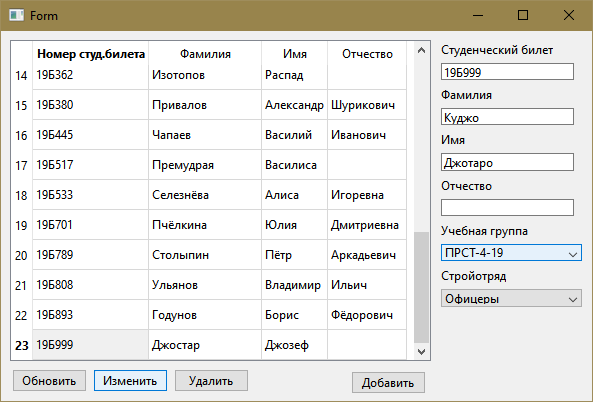


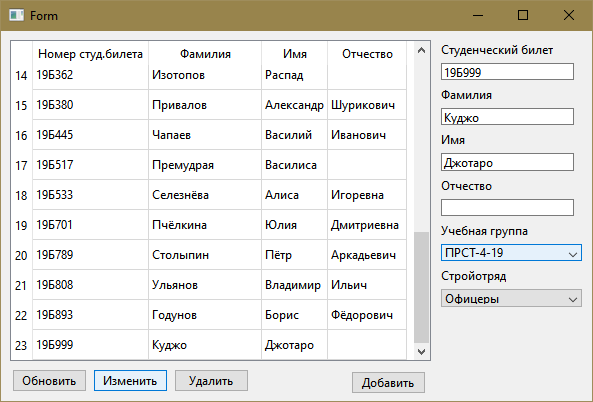
Справа от таблицы располагаются поля ввода для столбцов таблицы с уникальными значениями и выпадающее меню для выбора определённых значений. При заполнении этих полей и нажатии кнопки «Добавить» произойдёт добавление данных в таблицу с её автоматическим обновлением.



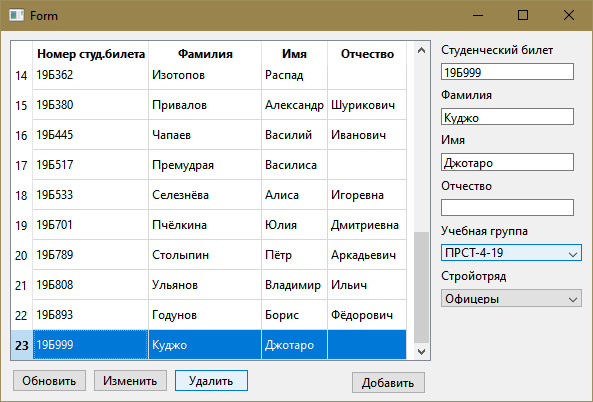


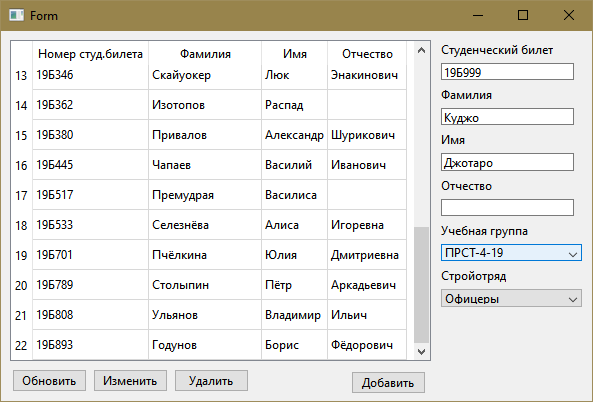
Для изменения записи нужно нажать на строку, данные которой нужно поменять, и ввести новые данные в поля справа, после чего нажать кнопку «Изменить», таблица автоматически обновится.



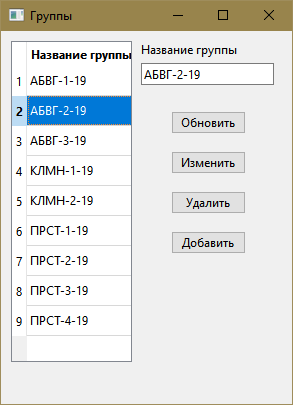


Для удаления записи из базы данных нужно нажать на удаляемую строку и выбрать кнопку «Удалить», после чего запись удалится из базы данных, таблица обновится.

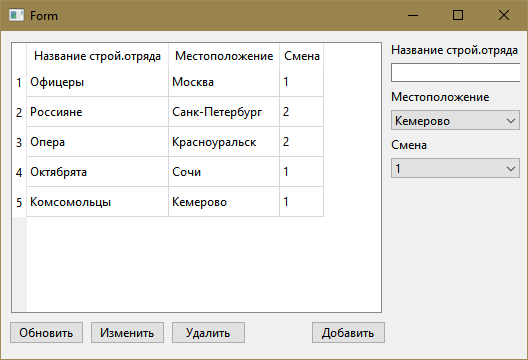


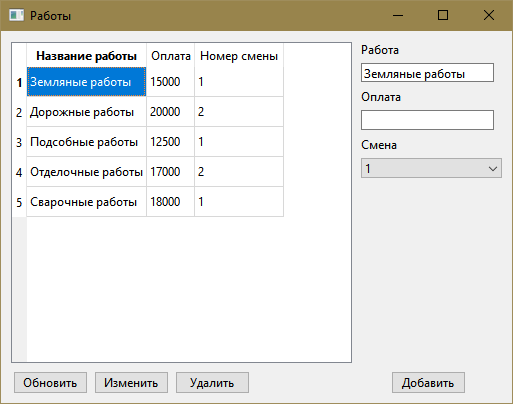


Функционал таблицы «Группы» практически не отличается «Студентов», поэтому подробно останавливаться мы на нём не будем.

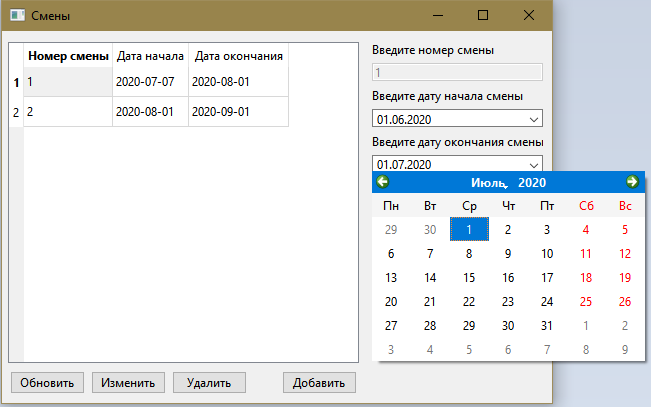


Таблицы «Строительные отряды» и «Работы» заполняются аналогично предыдущим.





В таблице «Смены» присутствует уникальная для этой базы данных механика – поля даты заполняются при помощи календаря. Остальные функции аналогичны функциям других таблиц.



**Вывод**

После выполнения данной работы я закрепила теорию создания баз данных и научилась проектировать реляционную базу данных, составлять программы взаимодействия с базой данных, разрабатывать ER-диаграммы а также программировать на языке SQL.

В дальнейшем полученные знания могут помочь мне на других дисциплинах

**Список используемых источников**

1. <https://www.youtube.com/playlist?list=PL47zejIkBQvTJqvh9ZtNn01zwJ2lGXk-E>
2. <https://www.youtube.com/watch?v=8MT20sMEmRc>
3. <https://docs.microsoft.com/ru-ru/azure/azure-sql/database/develop-cplusplus-simple>
4. <https://info-comp.ru/obucheniest/626-insert-into-in-t-sql.html>
5. <https://studme.org/93825/informatika/kursory>
6. <https://www.sql.ru/articles/mssql/2005/032101usingcorrelatedsubquery.shtml>