Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение

Санкт-Петербургский политехнический университет Петра Великого

(ФГАОУ ВО «СПБПУ»)

Институт среднего профессионального образования

ОТЧЕТ

по лабораторным работам

по предмету МДК 11.01 ТРЗБД

Выполнил студент

группы 32919/6 В.С.Ершов

Преподаватель: Д.А.Тузова

Санкт-Петербург

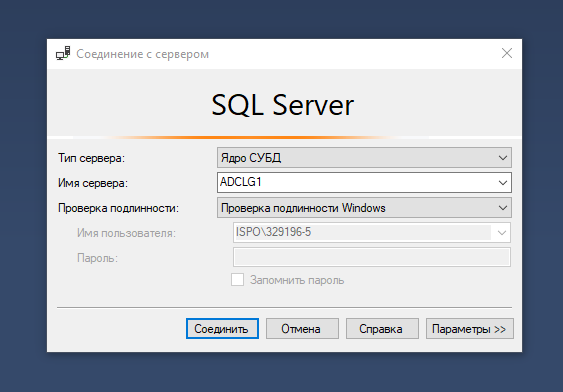
2023

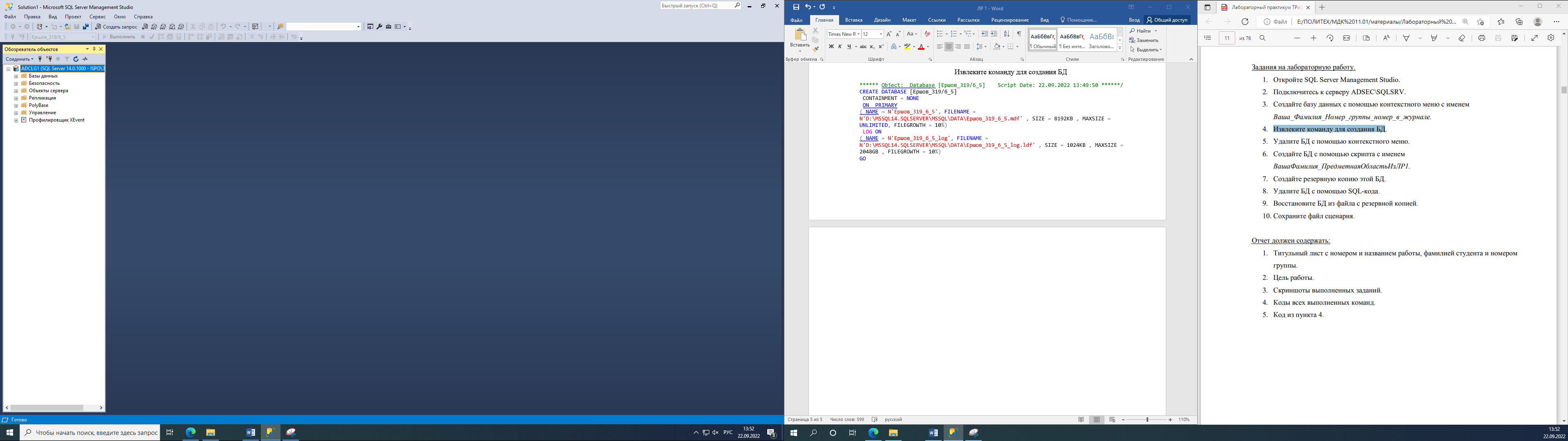
**Лабораторная работа №1**

**«Установка соединения с сервером Microsoft SQL Server, создание, копирование и удаление базы данных»**

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

* 1. Открыл SQL Server Management Studio и подключитесь к серверу ADCLG1.





1. Создал базу данных с помощью контекстного меню с именем Ваша\_Фамилия\_Номер\_группы\_номер\_в\_журнале — Ершов\_319/6\_5

Изображение выглядит как текст

Автоматически созданное описание

1. Извлек команду для создания БД

\*\*\*\*\*\* Object: Database [Ершов\_319/6\_5] Script Date: 22.09.2022 13:49:50 \*\*\*\*\*\*/

CREATE DATABASE [Ершов\_319/6\_5]

CONTAINMENT = NONE

ON PRIMARY

( NAME = N'Ершов\_319\_6\_5', FILENAME = N'D:\MSSQL14.SQLSERVER\MSSQL\DATA\Ершов\_319\_6\_5.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 10%)

LOG ON

( NAME = N'Ершов\_319\_6\_5\_log', FILENAME = N'D:\MSSQL14.SQLSERVER\MSSQL\DATA\Ершов\_319\_6\_5\_log.ldf' , SIZE = 1024KB , MAXSIZE = 2048GB , FILEGROWTH = 10%)

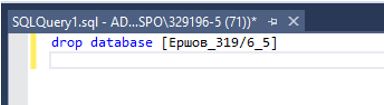
GO

1. Выяснил, что удаление БД с помощью контекстного меню не работает.

Изображение выглядит как текст

Автоматически созданное описание

Удаление с помощью SQL-кода работает.



Изображение выглядит как текст

Автоматически созданное описание

1. Создал БД с помощью скрипта с именем ВашаФамилия\_ПредметнаяОбластьИзЛР1.

Yershov\_football

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

7. Создание резервной копии БД Yershov\_football.

Изображение выглядит как текст

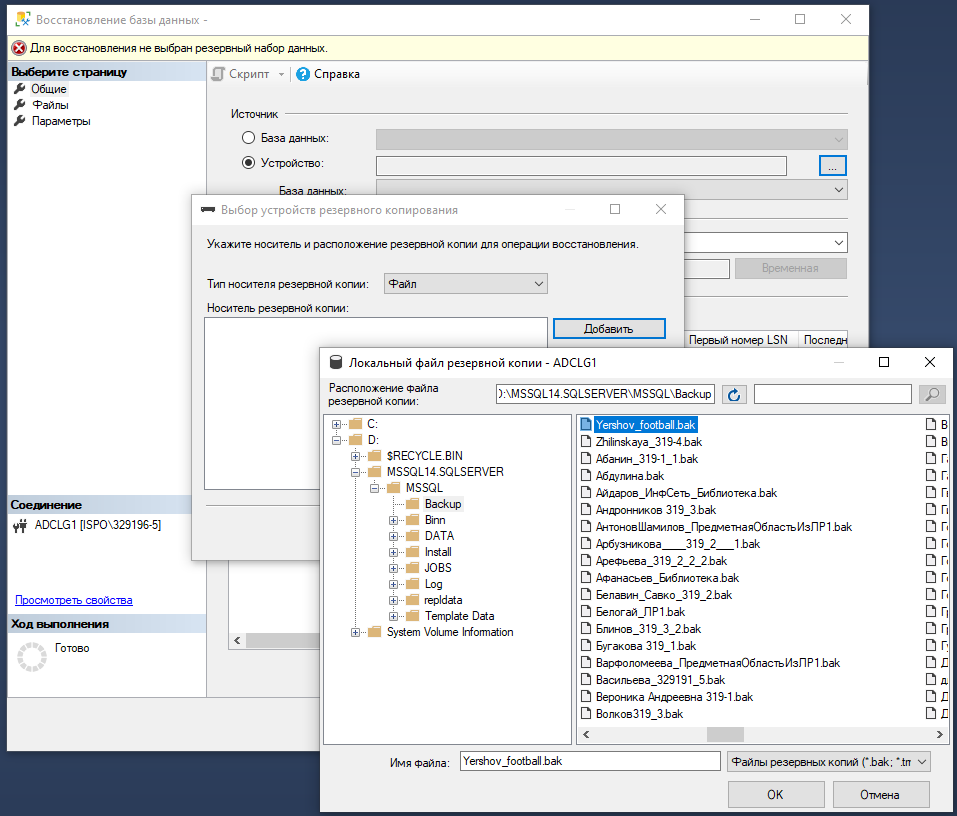
Автоматически созданное описание

8. Удаление БД с помощью SQL-кода.

Изображение выглядит как текст

Автоматически созданное описание

9. Восстановите БД из файла с резервной копией.



Изображение выглядит как текст

Автоматически созданное описание

**Лабораторная работа № 2**

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

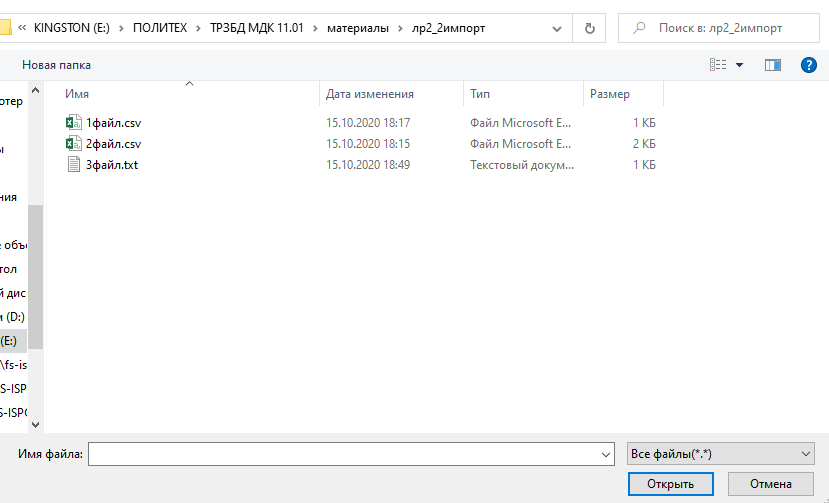
**Цель работы:** научиться правильно импортировать данные разных форматов, а также экспортировать данные.

В качестве источника данных выбрал «Неструктурированный файл». В обзоре файлов выбрал необходимый файл.

Изображение выглядит как текст

Автоматически созданное описание

Убедился, что видны все файлы.



Выбрал вид отношения после импорта

Изображение выглядит как текст

Автоматически созданное описание

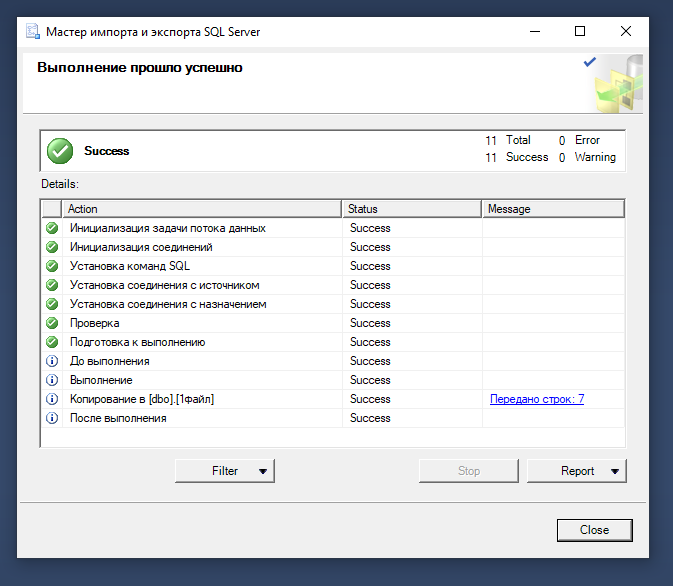
Указал место, куда нужно копировать данные (SQL Server, свою БД)

Изображение выглядит как текст

Автоматически созданное описание

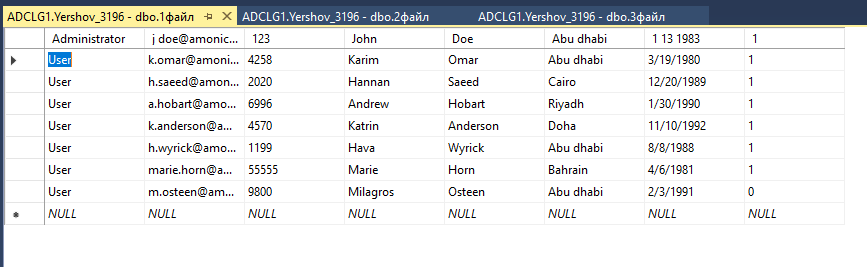
Изображение выглядит как текст

Автоматически созданное описание

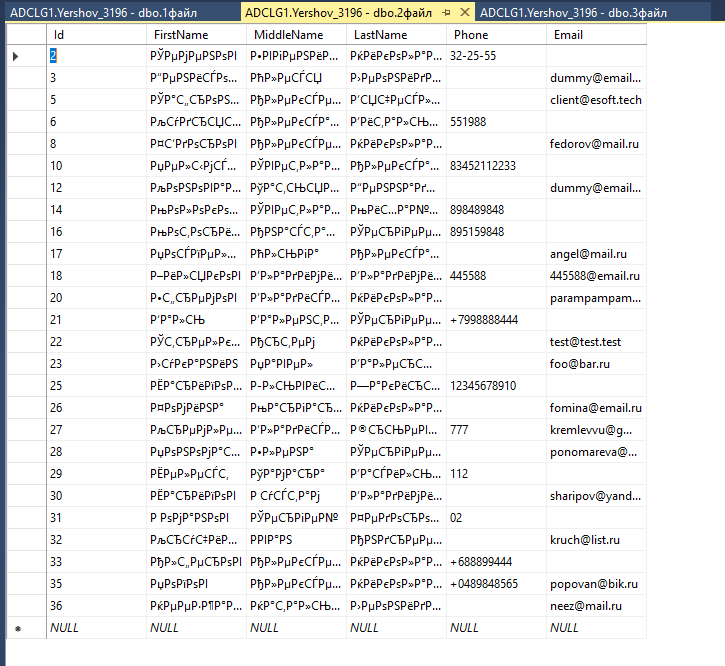


Результаты:

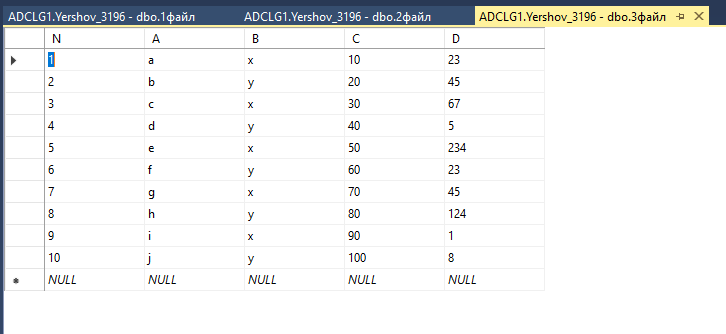
1.



2.



3.



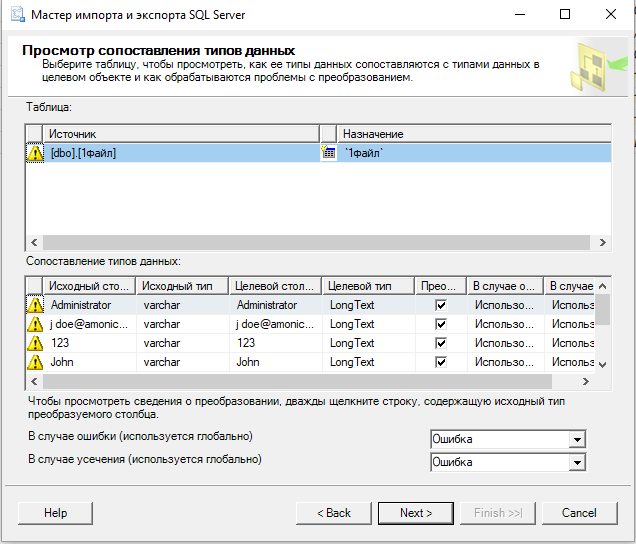
Экспорт

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание



Результат экспортирования:

Изображение выглядит как текст, внутренний

Автоматически созданное описание

**Лабораторная работа №3**

**«Создание и заполнение таблиц»**

**Цель работы:** научиться создавать отношения и заполнять их. Получить навык создания БД с помощью даталогической модели.

Схема данных БД

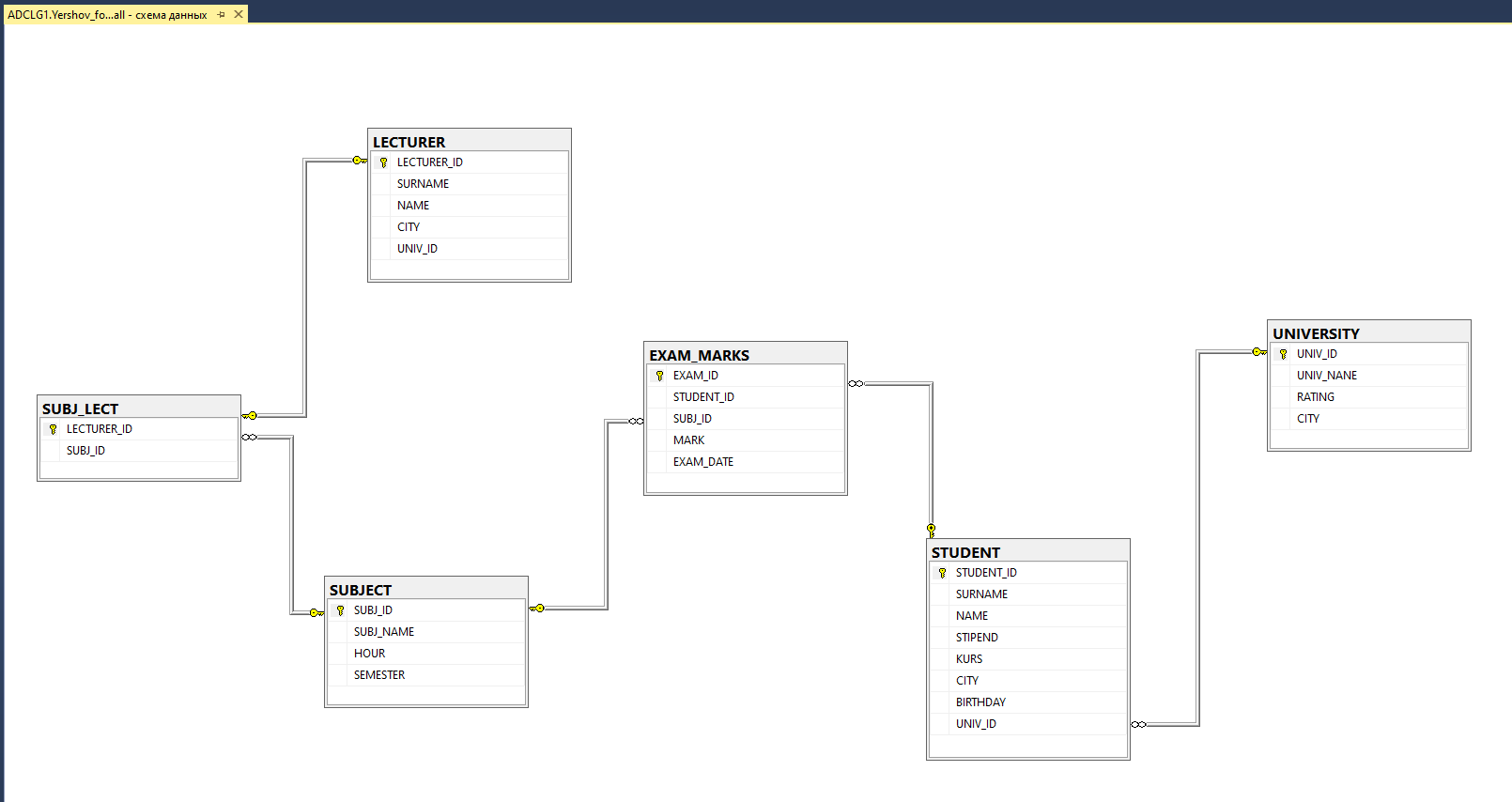


Рис. 1 Схема данных

Таблица EXAM\_MARKS

Изображение выглядит как стол

Автоматически созданное описание

Рис.2 Таблица EXAM\_MARKS

Таблица LECTURER

Изображение выглядит как стол

Автоматически созданное описание

Рис.3 Таблица LECTURER

Таблица STUDENT

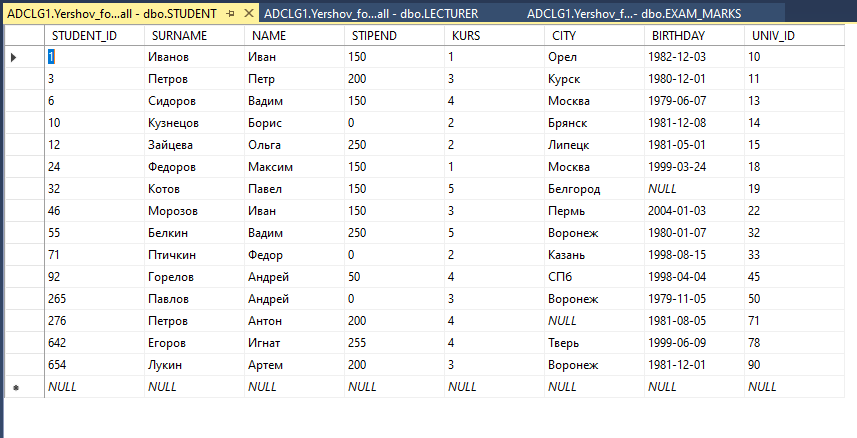


Рис.4 Таблица STUDENT

Таблица SUBJ\_LECT

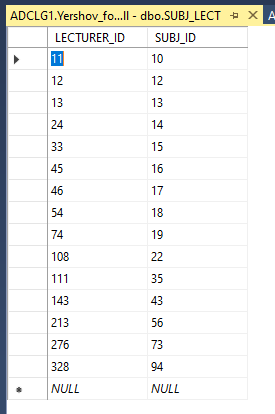


Рис.5 Таблица SUBJ\_LECT

Таблица SUBJECT

Изображение выглядит как стол

Автоматически созданное описание

Рис. 6 Таблица SUBJECT

Таблица UNIVERSITY

Изображение выглядит как стол

Автоматически созданное описание

Рис. 7 Таблица UNIVERSITY

**Лабораторная работа № 4.**

**«Разработка ограничений»**

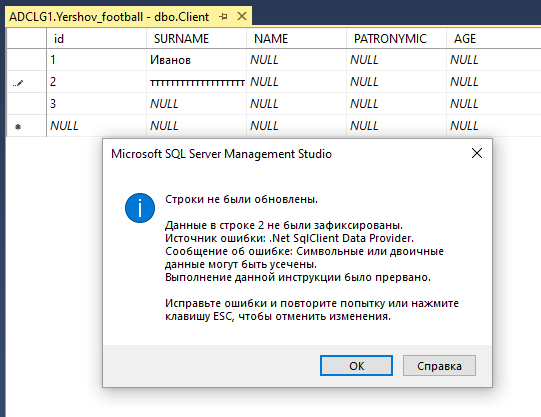
**Цель работы:** изучить используемые в SQL Server типы ограничений. Изучить SQL‑операторы для работы с ограничениями.

Код создания таблицы:

Изображение выглядит как текст

Автоматически созданное описание

Ограничение в столбце «SURNAME» - 40 символов (Аналогичное в «NAME» и «PATRONYMIC»)



Ограничение в столбце «AGE» - от 14 до 99

Изображение выглядит как текст

Автоматически созданное описание

Ограничение в столбце PH NUM (маска: +7(…)…-..-..)

Изображение выглядит как стол

Автоматически созданное описание

**Лабораторная работа № 5**

**«Работа с внешними ключами»**

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

**Ход работы:**

1. Я создал SQL-запрос для создания таблиц, необходимых для работы.

CREATE TABLE SHIPPERS2(

Номер\_Поставщика int NOT NULL primary key,

Фамилия nvarchar(25),

Рейтинг int,

Город nvarchar(50))

CREATE TABLE DETAILS2(

Номер\_Детали int NOT NULL primary key,

Название nvarchar(25),

Цвет nvarchar(25),

Вес int,

Город nvarchar(50))

CREATE TABLE PRODUCTS2(

Номер\_Изделия int NOT NULL primary key,

Название nvarchar(25),

Город nvarchar(50))

CREATE TABLE SUPPLIES2(

Номер int NOT NULL primary key,

Номер\_Поставщика int,

Номер\_Детали int,

Номер\_Изделия int,

Количество int)

1. Установил между ними отношения.

* Для первой таблицы создал каскадное удаление и обновление.
* Для второй — установка значения NULL
* Для третьей — отсутствие действия.

ALTER TABLE SUPPLIES2 ADD CONSTRAINT SHIPPER

FOREIGN KEY (Номер\_Поставщика) REFERENCES SHIPPERS2 (Номер\_Поставщика)

ON DELETE CASCADE

ON UPDATE CASCADE

ALTER TABLE SUPPLIES2 ADD CONSTRAINT DETAIL

FOREIGN KEY (Номер\_Детали) REFERENCES DETAILS2 (Номер\_Детали)

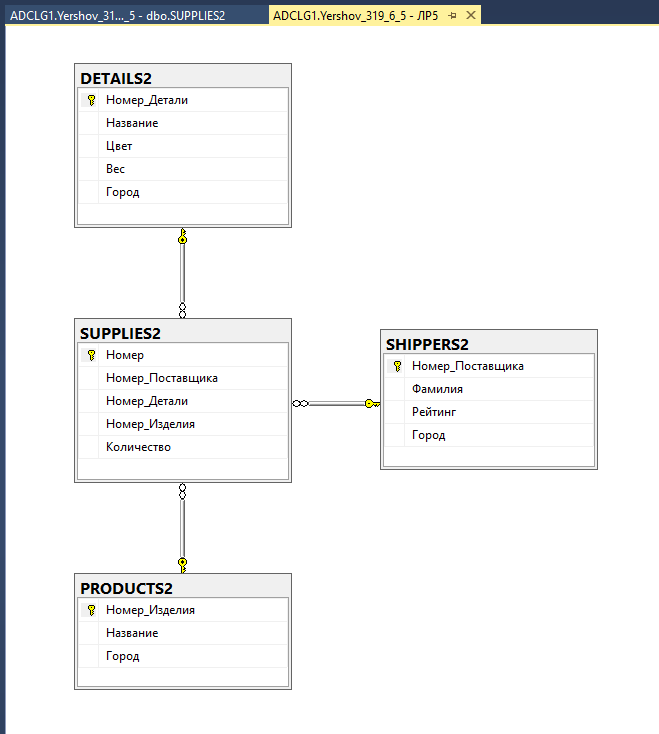
ON DELETE SET NULL

ALTER TABLE SUPPLIES2 ADD CONSTRAINT PRODUCT

FOREIGN KEY (Номер\_Изделия) REFERENCES PRODUCTS2 (Номер\_Изделия)

ON DELETE NO ACTION

Схема данных:



1. Заполнил таблицы, далее удалил первые строки из трех таблиц

Таблица DETAILS2 после удаления 1ой строки

Изображение выглядит как стол

Автоматически созданное описание

Таблица SHIPPERS2 после удаления 1ой строки

Изображение выглядит как стол

Автоматически созданное описание

Таблица PRODUCTS2 после удаления 2ой строки

Изображение выглядит как стол

Автоматически созданное описание

1. Зафиксировал изменения в таблице SUPPLIES2.

Универсальным для всех таблиц является каскадное ограничение, ибо оно упростит управление удаляемыми атрибутами и не засорит итоговую таблицу пустыми значениями.

Таблица SUPPLIES2 после удаления строк

Изображение выглядит как стол

Автоматически созданное описание

**Контрольные вопросы:**

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

Ограничения целостности – NULL, CHECK, UNIQUE, PRIMARY KEY и FOREIGN KEY.

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

1. Зачем нужны ограничения?

Ограничения нужны для обеспечения целостности БД и данных в ней.

1. Можно ли обойтись без ограничений в БД?

Чтобы обойтись без ограничений базы данных, необходимо заранее определить, что можно и что нельзя делать с базой данных, чтобы все приложения могли быть написаны, рассмотрены и тщательно протестированы.

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

Зависит от таблицы, особенно – если таблица связана с другой при помощи связи родитель-потомок.

1. Какие условия необходимо соблюсти, чтобы связать две таблицы связью?

Для связи между таблицами необходимы первичный ключ в основной таблице и внешний ключ во вторичной таблице.

1. Какие типы ключей вы знаете?

* Первичный ключ. Он действует на уровне столбца или таблицы и гарантирует уникальность в пределах таблицы первичного ключа, состоящего из одного или нескольких столбцов.
* Внешний ключ. Он действует на уровне таблицы и связывается с одним из кандидатов на первичный ключ в другой таблице.

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

Основная таблица - 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.

Вторичная таблица - 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.

1. Зачем нужны ключевые слова on update on delete?

Необязательные конструкции ON DELETE и ON UPDATE, определяют поведение MySQL при удалении/обновлении записей из таблицы-предка.

1. Какие параметры on update on delete существуют?

* CASCADE — При удалении/обновлении записей в таблице-предке, будут так же 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.

1. Какой синтаксис у ограничения типа CHECK?

(Название поля) check (Название поля) like (Условие)

**Лабораторная работа № 6**

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

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

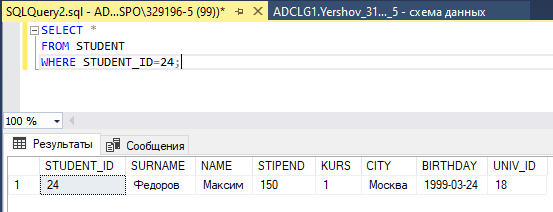
**Ход работы:**

1. 3 запроса с использованием оператора сравнения;

SELECT \*

FROM STUDENT

WHERE STUDENT\_ID=24;



SELECT \*

FROM STUDENT

WHERE KURS>2;

Изображение выглядит как стол

Автоматически созданное описание

SELECT \*

FROM STUDENT

WHERE STIPEND<200;

Изображение выглядит как стол

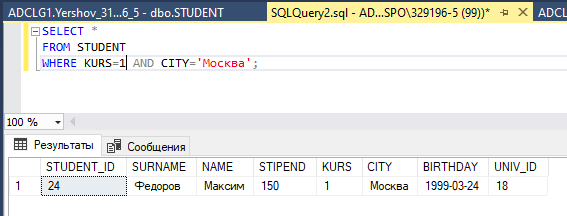
Автоматически созданное описание

1. 3 запроса с использованием логических операторов and, or, not;

SELECT \*

FROM STUDENT

WHERE KURS=1 AND CITY='Москва';



SELECT \*

FROM STUDENT

WHERE KURS=1 OR CITY='Москва';

Изображение выглядит как стол

Автоматически созданное описание

SELECT \*

FROM STUDENT

WHERE NOT(CITY='Москва');

Изображение выглядит как стол

Автоматически созданное описание

1. 1 запрос с комбинацией логических операторов;

SELECT \*

FROM STUDENT

WHERE KURS=2 AND NOT(CITY='Москва');

Изображение выглядит как стол

Автоматически созданное описание

1. 2 запроса на использование выражений над столбцами;

SELECT \*

FROM STUDENT

WHERE KURS+STUDENT\_ID<7;

Изображение выглядит как стол

Автоматически созданное описание

SELECT \*

FROM STUDENT

WHERE STIPEND-KURS-STUDENT\_ID>190;

Изображение выглядит как стол

Автоматически созданное описание

1. 3 запроса с использованием агрегатных функций;

SELECT COUNT(\*)

FROM STUDENT CITY

Изображение выглядит как текст

Автоматически созданное описание

SELECT AVG(STIPEND)

FROM STUDENT

Изображение выглядит как текст

Автоматически созданное описание

SELECT SUM(STIPEND)

FROM STUDENT

Изображение выглядит как текст

Автоматически созданное описание

1. 1 запрос с проверкой на принадлежность множеству;

SELECT \*

FROM STUDENT

WHERE KURS IN(1,2)

Изображение выглядит как стол

Автоматически созданное описание

1. 2 запроса с проверкой на принадлежность диапазону значений;

SELECT \*

FROM STUDENT

WHERE KURS BETWEEN 2 AND 4

Изображение выглядит как стол

Автоматически созданное описание

SELECT \*

FROM LECTURER

WHERE UNIV\_ID BETWEEN 14 AND 40;

Изображение выглядит как стол

Автоматически созданное описание

1. 2 запроса с проверкой на соответствие шаблону (разные);

SELECT \*

FROM STUDENT

WHERE CITY LIKE '%Б%';

Изображение выглядит как стол

Автоматически созданное описание

SELECT \*

FROM STUDENT

WHERE SURNAME LIKE '%Пет%';

Изображение выглядит как стол

Автоматически созданное описание

1. 1 запрос с проверкой на неопределенное значение.

SELECT \*

FROM STUDENT

WHERE CITY IS NULL;

Изображение выглядит как стол

Автоматически созданное описание

**Лабораторная работа № 7**

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

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

**Ход работы:**

1. 1 запрос с использованием декартового произведения двух таблиц;

SELECT SURNAME, UNIV\_NANE

FROM STUDENT, UNIVERSITY

Изображение выглядит как стол

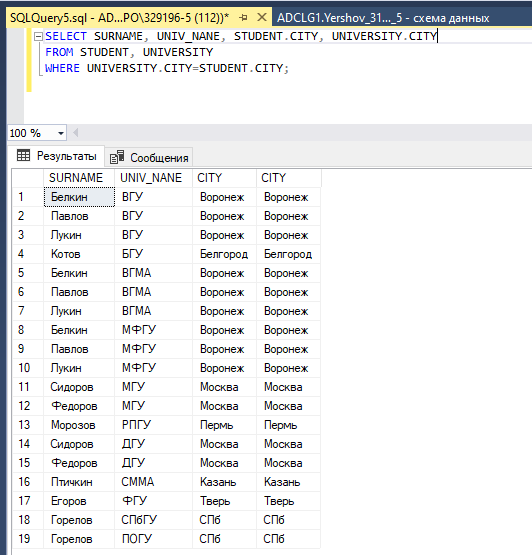
Автоматически созданное описание

1. 2 запроса с использованием соединения двух таблиц по равенству;

SELECT SURNAME, UNIV\_NANE, STUDENT.CITY, UNIVERSITY.CITY

FROM STUDENT, UNIVERSITY

WHERE UNIVERSITY.CITY=STUDENT.CITY;



SELECT STUDENT.SURNAME, STUDENT.[NAME], LECTURER.[NAME], LECTURER.[SURNAME]

FROM STUDENT, LECTURER

WHERE LECTURER.[SURNAME]=STUDENT.SURNAME;

Изображение выглядит как стол

Автоматически созданное описание

1. 1 запрос с использованием соединения двух таблиц по равенству и условием отбора;

SELECT STUDENT.SURNAME, STUDENT.[NAME], LECTURER.[NAME], LECTURER.[SURNAME]

FROM STUDENT, LECTURER

WHERE LECTURER.[SURNAME]=STUDENT.SURNAME AND NOT(LECTURER.[SURNAME]='Петров');

Изображение выглядит как стол

Автоматически созданное описание

1. 1 запрос с использованием соединения по трем таблицам;

SELECT STUDENT.UNIV\_ID, LECTURER.UNIV\_ID, UNIVERSITY.UNIV\_ID

FROM STUDENT, LECTURER, UNIVERSITY

WHERE STUDENT.UNIV\_ID=LECTURER.UNIV\_ID AND STUDENT.UNIV\_ID= UNIVERSITY.UNIV\_ID;

Изображение выглядит как стол

Автоматически созданное описание

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

SELECT SURNAME, UNIV\_NANE, STUDENT.CITY, UNIVERSITY.CITY

FROM

STUDENT FULL OUTER JOIN UNIVERSITY ON STUDENT.CITY = UNIVERSITY.CITY;

Изображение выглядит как стол

Автоматически созданное описание

SELECT STUDENT.SURNAME, STUDENT.[NAME], LECTURER.[NAME], LECTURER.[SURNAME]

FROM

STUDENT FULL OUTER JOIN LECTURER ON LECTURER.[NAME]=LECTURER.[SURNAME];

Изображение выглядит как стол

Автоматически созданное описание

1. 1 запрос с использованием левого внешнего соединения;

SELECT EXAM\_MARKS.STUDENT\_ID, STUDENT.STUDENT\_ID, SURNAME

FROM EXAM\_MARKS LEFT JOIN STUDENT ON EXAM\_MARKS.STUDENT\_ID=STUDENT.STUDENT\_ID;

Изображение выглядит как стол

Автоматически созданное описание

1. 1 запрос на использование правого внешнего соединения;

SELECT EXAM\_MARKS.STUDENT\_ID, STUDENT.STUDENT\_ID, SURNAME

FROM EXAM\_MARKS RIGHT JOIN STUDENT ON EXAM\_MARKS.STUDENT\_ID=STUDENT.STUDENT\_ID;

Изображение выглядит как стол

Автоматически созданное описание

1. 1 запрос с использованием внутреннего соединения.

SELECT UNIVERSITY.CITY, STUDENT.CITY, STUDENT.STUDENT\_ID, SURNAME

FROM UNIVERSITY JOIN STUDENT ON UNIVERSITY.CITY=STUDENT.CITY;

Изображение выглядит как стол

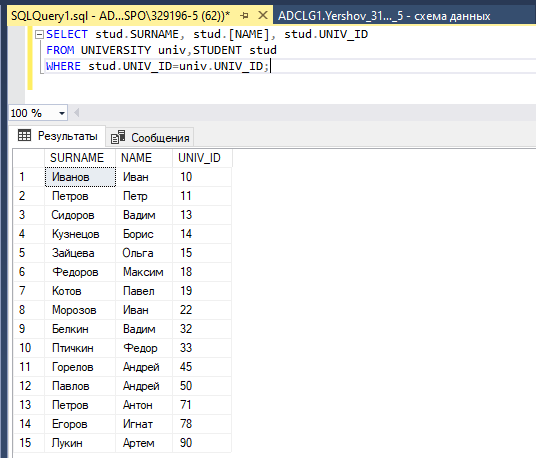
Автоматически созданное описание

1. 1 запрос на проверку какого-либо условия через самосоединение.

SELECT stud.SURNAME, stud.[NAME], stud.UNIV\_ID

FROM UNIVERSITY univ,STUDENT stud

WHERE stud.UNIV\_ID=univ.UNIV\_ID;



**Лабораторная работа № 8**

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

**Цель работы:** 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.

**Вариант 1 (№5)**

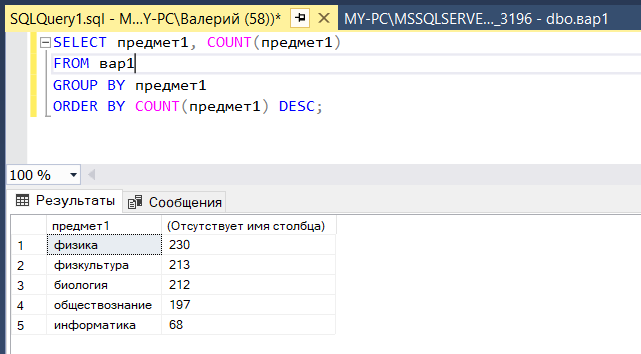
-Запрос, который покажет, сколько человек выбрало тот или иной предмет для первого экзамена, сделайте сортировку по убыванию.

SELECT предмет1, COUNT(предмет1)

FROM вар1

GROUP BY предмет1

ORDER BY COUNT(предмет1) DESC;



-Запрос, который покажет, какие предметы в качестве первого экзамена выбрало более 50 человек.

SELECT предмет1, COUNT(предмет1)

FROM вар1

GROUP BY предмет1

HAVING COUNT(предмет1)>50;

Изображение выглядит как текст

Автоматически созданное описание

-Запрос, который покажет сколько студентов выбрало в качестве второго экзамена физику, химию или биологию. Создайте два запроса один с помощью WHERE, другой используя HAVING.

1)

SELECT COUNT(предмет2)

FROM вар1

WHERE student\_id IN

(SELECT student\_id

FROM вар1

WHERE предмет2='физика' OR предмет2='химия' OR предмет2='биология');

Изображение выглядит как текст

Автоматически созданное описание

2)

SELECT предмет2, COUNT(предмет2)

FROM вар1

GROUP BY предмет2

HAVING предмет2='физика' OR предмет2='химия' OR предмет2='биология';

Изображение выглядит как текст

Автоматически созданное описание

-Запрос, который покажет, сколько студентов выбрало тот или иной предмет в качестве второго экзамена и получивших более 100 баллов за оба экзамена

SELECT student\_id, предмет2, [баллы за 2 экзамена]

FROM вар1

GROUP BY student\_id, предмет2, [баллы за 2 экзамена]

HAVING [баллы за 2 экзамена]>100;

Изображение выглядит как стол

Автоматически созданное описание

-Запрос, который покажет сколько студентов выбрали Биологию в качестве первого или второго предмета.

SELECT COUNT(STUDENT\_ID)

FROM вар1

WHERE предмет1= 'биология' OR предмет2='биология';

Изображение выглядит как текст

Автоматически созданное описание

**Контрольные вопросы:**

1. Что такое агрегатная функция?

Агрегатные функции – встроенные функции T-SQL, обрабатывающие коллекцию значений и возвращающее одно значение. К таким функциям относятся, например, нахождение минимального и максимального числа. Обрабатывая некоторый объем данных, мы получаем лишь одно значение на выходе.

1. Зачем нужно самосоединение?

Самосоединение можно использовать для проверки корректности данных.

1. Какие типы соединений вы знаете?

* Декартово произведение таблиц,
* соединение по равенству,
* левое внешнее соединение,
* правое внешнее соединение,
* внутренне (естественное) соединение,
* полное внешнее соединение,
* самосоединение,
* перекрестное соединение

1. Что такое группировка? Приведите пример использования.

Группировка – разделение данных таблицы на определенные группы.

Например, поиск средней стипендии по каждому курсу:

SELECT AVG(STIPEND), KURS

FROM STUDENT

GROUP BY KURS

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

В SELECT указать \*, а в условии будет сравнение нужного нам атрибута и максимума по нему.

6. В чем разница между операторами WHERE и HAVING?

Оператор HAVING. используется, если вам нужна фильтрация уже сгруппированных данных. Оператор WHERE со сгруппированными данными не используется.

7. В чем разница между функциями COUNT и SUM?

COUNT – возвращает количество значений в группе.

SUM – возвращает сумму всех значений в групп.

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

Да, можно.

Сначала будет происходить сортировка по первому атрибуту, потом по второму и тд.

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. Можно применить предложение HAVING только к тем столбцам, которые появляются в предложении GROUP BY или агрегатной функции.

10. В какой последовательности происходит сортировка по умолчанию?

По возрастанию.

11. Как можно отсортировать уже сгруппированные записи?

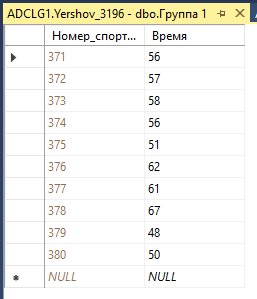
С помощью ORDER\_BY

**Лабораторная работа № 9**

**«Инструкция MERGE»**

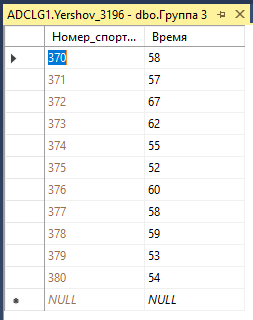
**Цель работы:** изучить принципы слияния таблиц, настройки и параметры инструкции MERGE.

1. Создал в своей базе данных три таблицы и заполнил их, аналогично таблицам 1, 2 и 3. Номер спортсмена заполнялся с помощью свойства identity (x, y).



Изображение выглядит как стол

Автоматически созданное описание



1. Создал сводную таблицу, в которой будут хранятся все данные из первых трех таблиц. Добавил в нее данные с помощью операции MERGE. Незаполненные ячейки — пустые (NULL).

create table dbo.п.2(

[Номер спортсмена] int not null,

[Время 1, с] int,

[Время 2, с] int,

[Время 3, с] int,

)

MERGE [п.2] AS Target

USING [Группа 1] AS SOURSE

ON (Target.[Номер спортсмена] = SOURSE.[Номер спортсмена])

WHEN MATCHED AND TARGET.[Время 1, с] <> SOURSE.[Время] OR TARGET.[Время 1, с] IS NULL

THEN UPDATE SET TARGET.[Время 1, с] = SOURSE.[Время]

WHEN NOT MATCHED BY TARGET

THEN INSERT ([Номер спортсмена], [Время 1, с]) VALUES (SOURSE.[Номер спортсмена], SOURSE.[Время]);

MERGE [п.2] AS Target

USING [Группа 2] AS SOURSE

ON (Target.[Номер спортсмена] = SOURSE.[Номер\_спортсмена])

WHEN MATCHED AND TARGET.[Время 2, с] <> SOURSE.[Время] OR TARGET.[Время 2, с] IS NULL

THEN UPDATE SET TARGET.[Время 2, с] = SOURSE.[Время]

WHEN NOT MATCHED BY TARGET

THEN INSERT ([Номер спортсмена], [Время 2, с]) VALUES (SOURSE.[Номер спортсмена], SOURSE.[Время]);

MERGE [п.2] AS Target

USING [Группа 3] AS SOURSE

ON (Target.[Номер спортсмена] = SOURSE.[Номер спортсмена])

WHEN MATCHED AND TARGET.[Время 3, с] <> SOURSE.[Время] OR TARGET.[Время 3, с] IS NULL

THEN UPDATE SET TARGET.[Время 3, с] = SOURSE.[Время]

WHEN NOT MATCHED BY TARGET

THEN INSERT ([Номер спортсмена], [Время 3, с]) VALUES (SOURSE.[Номер\_спортсмена], SOURSE.[Время]);

Изображение выглядит как стол

Автоматически созданное описание

1. Создал новую таблицу, в которой содержится лучший результат каждого спортсмена. Сортировку выполнил с помощью операции MERGE.

MERGE [п.3] AS TARGET

USING

(SELECT [Номер спортсмена],

(

SELECT MIN([Лучшее время])

FROM (VALUES ([Время 1, с]), ([Время 2, с]), ([Время 3, с])) AS x([Лучшее время])

) AS [Лучшее время]

FROM [п.2]) AS SOURCE

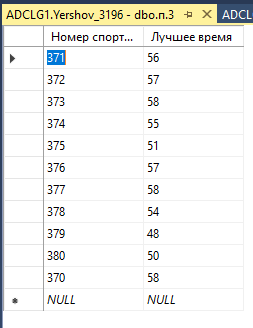
ON TARGET.[Номер спортсмена] = SOURCE.[Номер спортсмена]

WHEN MATCHED AND TARGET.[Лучшее время] <> SOURCE.[Лучшее время] OR TARGET.[Лучшее время] IS NULL

THEN UPDATE SET TARGET.[Лучшее время] = SOURCE.[Лучшее время]

WHEN NOT MATCHED BY TARGET

THEN INSERT ([Номер спортсмена], [Лучшее время]) VALUES (SOURCE.[Номер спортсмена], SOURCE.[Лучшее время]);



1. Вывел номер победителя и его результат.

SELECT [Номер спортсмена], [Лучшее время]

FROM [п.3]

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

FROM [п.3]);

Изображение выглядит как текст

Автоматически созданное описание

1. Аналогично вывел худший результат за всё соревнование.

MERGE [п.5] AS TARGET

USING

(SELECT [Номер спортсмена],

(

SELECT MAX([Худшее время])

FROM (VALUES ([Время 1, с]), ([Время 2, с]), ([Время 3, с])) AS x([Худшее время])

) AS [Худшее время]

FROM [п.2]) AS SOURCE

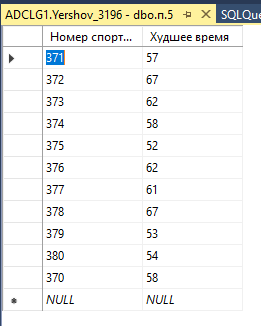
ON TARGET.[Номер спортсмена] = SOURCE.[Номер спортсмена]

WHEN MATCHED AND TARGET.[Худшее время] <> SOURCE.[Худшее время] OR TARGET.[Худшее время] IS NULL

THEN UPDATE SET TARGET.[Худшее время] = SOURCE.[Худшее время]

WHEN NOT MATCHED BY TARGET

THEN INSERT ([Номер спортсмена], [Худшее время]) VALUES (SOURCE.[Номер спортсмена], SOURCE.[Худшее время]);



SELECT [Номер спортсмена], [Худшее время]

FROM [п.5]

WHERE [Худшее время] = (SELECT MAX([Худшее время])

FROM [п.5]);

Изображение выглядит как текст

Автоматически созданное описание

**Лабораторная работа № 10. «Функции».**

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

**Ход работы:**

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

CREATE FUNCTION Calculator(@Число int, @ВтороеЧисло int, @Процедура nvarchar(30))

returns int

as

BEGIN

return

case when @Процедура = 'Сложение'

then @Число + @ВтороеЧисло

when @Процедура = 'Вычитание'

then @Число - @ВтороеЧисло

when @Процедура = 'Умножение'

then @Число \* @ВтороеЧисло

when @Процедура = 'Деление'

then @Число / @ВтороеЧисло

when @Процедура = 'Синус'

then sin(@Число)

when @Процедура = 'Косинус'

then cos(@Число)

when @Процедура = 'Тангенс'

then tan(@Число)

when @Процедура = 'Котангенс'

then cot(@Число)

when @Процедура = 'Возведение в степень'

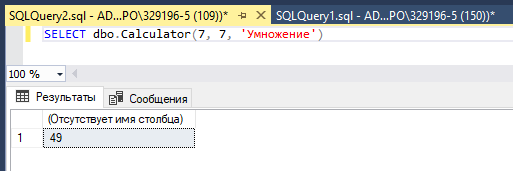
then power(@Число, @ВтороеЧисло)

END

END

GO

1. Протестировал функцию.



Изображение выглядит как стол

Автоматически созданное описание

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

CREATE FUNCTION Сезон(@Дата SmallDateTime)

returns nvarchar(5)

as

BEGIN

return

case when month(@Дата) in (12, 1, 2)

then 'Зима'

when month(@Дата) in (3, 4, 5)

then 'Весна'

when month(@Дата) in (6, 7, 8)

then 'Лето'

when month(@Дата) in (9, 10, 11)

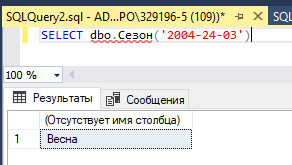
then 'Осень'

END

END

GO

1. Протестировал функцию.



1. Создал функцию, которая будет разбивать входную строку на отдельные строки. Символ пробела является разделителем строк.

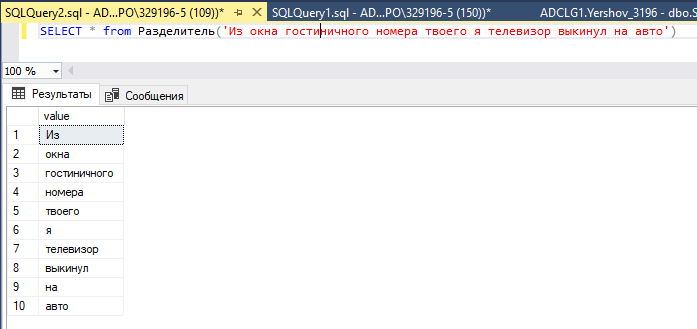
CREATE FUNCTION Разделитель(@Строка nvarchar(500))

returns table

return select \* from string\_split (@Строка, ' ')

GO

1. Протестировал функцию.



1. Импортировал документ Doc.xsl.
2. Работа с таблицей StudList. Создал функцию, возвращающую список студентов, обучающихся или отчисленных, в зависимости от входного параметра. Если в столбце Studying стоит No, студент отчислен, в противном случае – обучается.

CREATE FUNCTION Студлист(@ДаИЛИнет nvarchar(3))

returns table

return select \* from StudList$

where Studying = @ДаИЛИнет

GO

1. Протестировал функцию.



**Лабораторная работа № 11**

**«Процедуры»**

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

**Ход работы:**

1. Я создал 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.

CREATE PROC OBUCHEN @Name nvarchar(max), @maxfactor bit output AS

if (SELECT Studying From StudList$

where Student = @Name) = 'Yes'

select @maxfactor = 1

else select @maxfactor = 0

GO

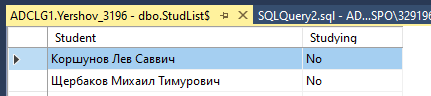
declare @xcringe bit

exec OBUCHEN 'Коршунов Лев Саввич', @xcringe output

print @xcringe

go

drop proc OBUCHEN;



1. Проверить процедуру.

Изображение выглядит как текст

Автоматически созданное описание

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

CREATE PROC skidka AS

update Shop$

set Price = Price\*0.95

where Comment = 'Yes';

GO

exec skidka

go

drop proc skidka;

1. Проверил процедуру.

Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

1. Работа с таблицей Shop. Я создал процедуру с входным параметром. Входной параметр – номер клиента. Процедура удаляет все строки с заказами этого клиента.

CREATE PROC del @ClientNumber int AS

delete from Shop$

where [Client N] = @ClientNumber

GO

exec del 206

go

drop proc del;

1. Проверка

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

1. Создал процедуру с входным и выходным параметрами, где входной параметр – номер клиента, а выходной – самый дорогой заказ выбранного клиента.

CREATE PROC dorogo @ClientNumber nvarchar(max), @ZakazNumber nvarchar(max) output AS

select @ZakazNumber =

[Order N] from Shop$

Where Price = (select MAX(Price) from Shop$)

GO

declare @PAMPARAM nvarchar(max)

exec dorogo 209, @PAMPARAM output

print @PAMPARAM

go

drop proc dorogo;

1. Проверка.

Изображение выглядит как текст

Автоматически созданное описание

**Контрольные вопросы:**

1. Какие типы процедур вы знаете?

* Пользовательские
* Временные
* Системные
* Расширенные пользовательские

1. Где хранятся процедуры?

Процедуры сохраняются в базе, в которой они создаются. Временные помещаются в базу tempbd.

1. Какой синтаксис у создания процедур?

Create procedure <Имя процедуры>

<входные параметры>,

<выходные параметры> output

As

<Тело процедуры>

go

1. Какой синтаксис у выполнения процедур?

Exec <имя процедуры> <входные параметры>, <выходные параметры> output

1. В чем разница между функцией и процедурой?

Функция обязательно возвращает некоторый результат, а процедура может выполнять только ряд задач.

1. Зачем нужны процедуры?

Процедуры необходимы для многоразового выполнения набора команд или ограничения доступа к данным таблиц.

**Лабораторная работа № 12**

**Создание, программирование и управление триггерами**

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

**Ход работы:**

1.Импортировал данные из файла AUTH.xls

2.Создал триггер Trigger\_1, который после изменения данных будет выводить строки, со старыми и новыми значениями.

CREATE TRIGGER Trigger\_1

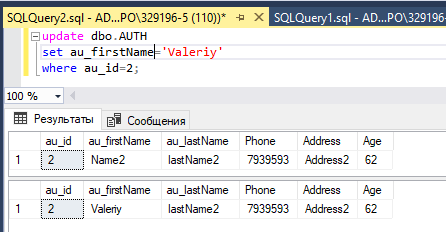
ON AUTH

AFTER UPDATE AS

SELECT \* FROM deleted;

SELECT \* FROM inserted;

Протестировал триггер.



3.Создайте триггер Trigger\_2, который вместо удаления выведет следующую информацию: «попытка удаления информации об au\_id»

CREATE TRIGGER Trigger\_2

ON AUTH

INSTEAD OF DELETE AS

BEGIN

DECLARE @a varchar;

SET @a = (SELECT au\_id FROM deleted);

PRINT ('Попытка УДАЛЕНИЯ информации о ' + @a);

END

Протестировал триггер.

Изображение выглядит как текст

Автоматически созданное описание

4.Создал триггер Trigger\_3, который будет считать и выводить количество удаленных строк

CREATE TRIGGER Trigger\_3

ON AUTH

FOR DELETE AS

BEGIN

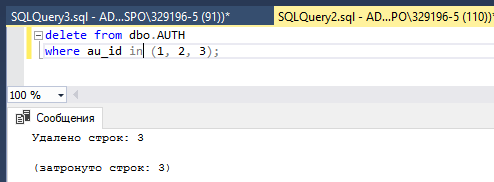
DECLARE @a varchar;

SET @a = (SELECT COUNT(au\_id) FROM deleted);

PRINT ('Удалено строк: ' + @a);

END

Протестировал триггер.



**Контрольные вопросы:**

1. Какие типы триггеров вы знаете?

* FOR – запуск триггера при выполнении заданной в этом списке команды;
* AFTER – запуск триггера после успешного выполнения команд списка;
* INSTEAD OF – триггеры вызывается вместо выполнения команд списка.

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

Да, можем.

3. Как можно протестировать ваш триггер?

Для тестирования триггера следует выполнить действие, согласно которому триггер включится.

4. Сколько триггеров может иметь одно отношение?

В каждой таблице может поддерживаться до трех триггеров: по одному на каждое событие, связанное с выполнением вставки, обновления или удаления.

5. Можно ли создать один триггер на несколько операций?

Триггеры определяются по событиям в таблице, причем на каждое событие в таблице допускается только один триггер.

С несколькими событиями может быть связан единственный триггер, например, после вставки, обновления.

6. Как можно увидеть созданные триггеры?

<База данных> - <нужная таблица> - <триггеры>

7. Как можно изменить тело триггера?

Вместо CREATE TRIGGER указывать ALTER TRIGGER и далее тело триггера с изменениями.

**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.

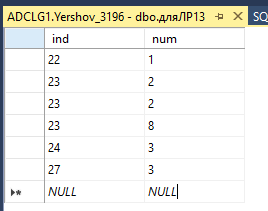
1. Проверка работы транзакции. Выбрал отношение базе данных. Создал транзакцию, в которой содержится один запрос вставки. Запустил ее дважды. В первом случае завершил транзакцию командой COMMIT, второй раз ROLLBACK.

BEGIN TRANSACTION lr13

INSERT INTO dbo.дляЛР13 (ind, num)

VALUES (27,3);

COMMIT TRANSACTION lr13

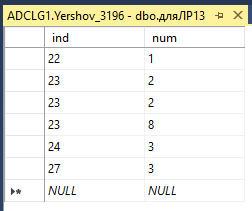


BEGIN TRANSACTION lr13

INSERT INTO dbo.дляЛР13 (ind, num)

VALUES (27,3);

ROLLBACK TRANSACTION lr13



Выводы: в результате использования команды COMMIT транзакция выполняется, а в результате команды ROLLBACK происходит откат изменений транзакции.

1. Работа с вложенными транзакциями. Создать вложенные транзакции, выполнив следующие команды:

CREATE TABLE [lr13-2] (a1 int)

BEGIN TRANSACTION tr1

INSERT INTO [lr13-2] VALUES (11)

BEGIN TRANSACTION tr2

INSERT INTO [lr13-2] VALUES (22)

BEGIN TRANSACTION tr3

INSERT INTO [lr13-2] VALUES (33)

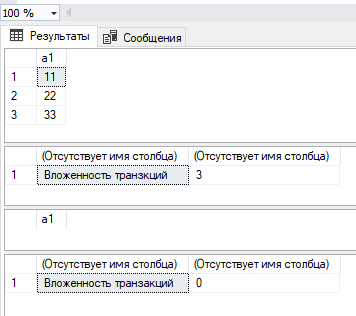
SELECT \* FROM [lr13-2]

SELECT 'Вложенность транзкций', @@TRANCOUNT

ROLLBACK TRANSACTION

SELECT \* FROM [lr13-2]

SELECT 'Вложенность транзакций', @@TRANCOUNT



Выводы: происходит создание таблицы lr13-2, далее код начал 3 транзакции, в которых добавил в таблицу по строчке данных (11,22,33). Затем код выбрал данные из этой таблицы, и стали видны 3 строки данных. Далее с помощью функции @@TRANCOUNT произошел вывод количества активных транзакций – 3. Командой ROLLBACK TRANSACTION закончил все транзакции с откатом изменений (все, тк не указано имя конкретной транзакции). После вывода на экран повторно списка активных транзакций результатом является - 0.

1. Работа с данными из файла groups. Импортировал 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.

BEGIN TRANSACTION [lr13-3]

INSERT INTO dbo.набор2018 (ФИО, группа)

SELECT \* FROM dbo.group212 WHERE фио NOT IN (SELECT ФИО FROM dbo.набор2018);

SELECT \* FROM dbo.group213 WHERE фио NOT IN (SELECT ФИО FROM dbo.набор2018);

COMMIT TRANSACTION [lr13-)]



1. SET IMPLICIT\_TRANSACTIONS {ON | OFF}

Присвоение параметру SET IMPLICIT\_TRANSACTIONS значения ON устанавливает для соединения режим неявных транзакций.

Для транзакций, автоматически открываемых в результате присвоения этому параметру значения ON, пользователю необходимо явно выполнить в конце транзакции фиксацию или откат. Иначе после отключения пользователя будет выполнен откат транзакции и будут потеряны все сделанные ей изменения данных.

Значение OFF возвращает соединение в режим с автоматической фиксацией транзакций.

После фиксации транзакции выполнение одной из указанных инструкций (см. таблицу) начнет новую транзакцию.

Когда соединение находится в режиме неявных транзакций, выполнение любой из следующих инструкций начнет транзакцию.

**Контрольные вопросы:**

1.Зачем нужны транзакции?

Для повышения надежности работы системы. Выполнение транзакций не нарушает целостность данных БД и откатывает их в случае сбоя.

2.С какими режимами транзакции вы работали?

Режим автофиксации – каждый оператор – транзакция. По завершению оператора транзакция завершается.

Явный режим – указание начала и конца транзакции.

Неявный режим – транзакция начинается автоматически при использовании определённых операторов T-SQL. Конец транзакции обозначается COMMIT или ROLLBACK.

3.Какие команды являются признаком начала неявной транзакции?

Операторы, для начала транзакции в неявном режиме:

* ALTER TABLE
* CREATE
* DELETE
* DROP
* FETCH
* GRANT
* INSERT
* OPEN
* REVOKE
* SELECT
* TRUNCATE TABLE
* UPDATE

4.Зачем нужны контрольные точки транзакций? Продемонстрируйте их работу.

Точка транзакции позволяет выполнять откат только до определенной точки в транзакции, а не до самого начала транзакции. Все модификации, выполненные до точки сохранения, остаются в силе и не подвергаются откату, но для всех операторов, выполняемых после точки сохранения вплоть до оператора ROLLBACK, будет выполнен откат. Затем начнут выполняться операторы, следующие за оператором ROLLBACK. Если вы затем зададите откат транзакции без указания точки сохранения, то, как обычно, будет выполнен откат всех модификаций вплоть до начала данной транзакции, т.е. будет отменена вся транзакция.

5.Можно ли узнать количество активных транзакций?

Функция @@TRANCOUNT предназначена для определения количества активных транзакций, начатых в активном соединении.

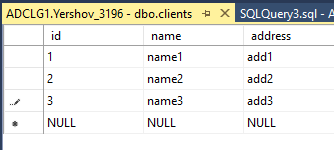
**Лабораторная работа № 14**

**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.

**Ход работы:**

1. Создал таблицу Clients со следующими атрибутами: id, name, address.



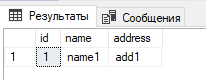
1. С помощью курсора создайте запрос, который будет выводить информацию о всех клиентах c нечетным Id.

DECLARE Fcursor CURSOR

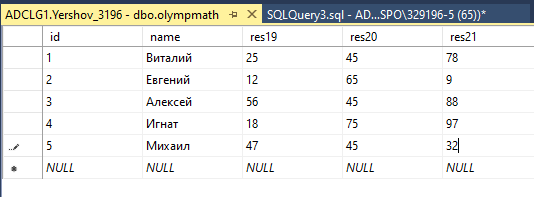
FOR SELECT \* FROM dbo.Clients WHERE id % 2 <> 0

OPEN Fcursor

FETCH NEXT FROM Fcursor;



1. Создал таблицу OlympMath с результатами Олимпиады по математике за 2019, 2020 и 2021 года. Таблица OlympMath состоит из следующих атрибутов: id, name, res19, res20, res21.



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

DECLARE @TEMP TABLE (id int,[Максимальный балл(2019-2021)] int)

DECLARE @ID int

DECLARE @Max\_res int

DECLARE Scursor CURSOR

FOR SELECT T.id, MAX(T.res) AS [Максимальный балл(2019-2021)]

FROM (SELECT id, res19 AS res FROM dbo.olympmath

UNION SELECT id, res20 FROM dbo.olympmath

UNION SELECT id, res21 FROM dbo.olympmath) AS T

GROUP BY t.id;

OPEN Scursor

FETCH NEXT FROM Scursor INTO @ID, @Max\_Res

WHILE @@FETCH\_STATUS = 0

BEGIN

INSERT INTO @TEMP (id, [Максимальный балл(2019-2021)])

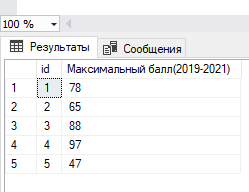
VALUES (@ID, @Max\_Res);

FETCH NEXT FROM Scursor INTO @id, @Max\_Res

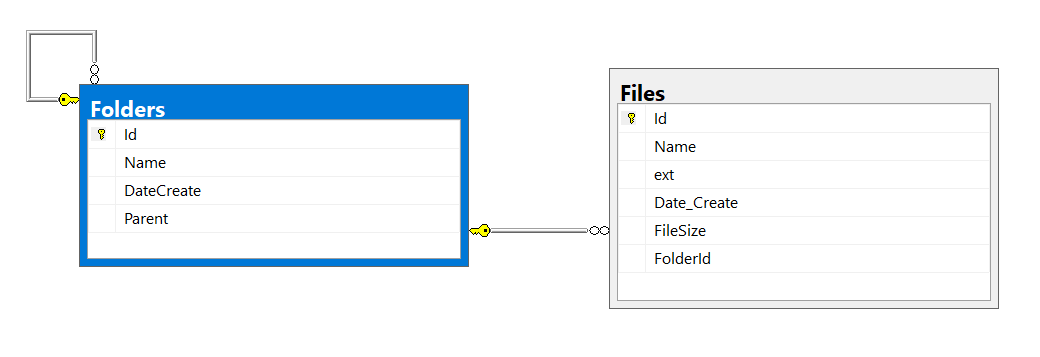
END

CLOSE Scursor

SELECT \* FROM @TEMP



1. Создайте две таблицы со следующей структурой (папки и файлы):



Реализовал функцию, которая принимает на входе идентификатор каталога и выводит все файлы и каталоги, которые находятся в заданном каталоге и во всех подкаталогах.

CREATE FUNCTION Func(@folder\_id int)

RETURNS @table TABLE (ID int, [NAME] nvarchar(30))

AS BEGIN

INSERT INTO @table SELECT id, [NAME] FROM Folders WHERE parent = @folder\_id;

INSERT INTO @table SELECT id, [NAME] FROM Files WHERE folderid = @folder\_id;

BEGIN

DECLARE curs cursor

FOR SELECT ID FROM Folders WHERE parent = @folder\_id

DECLARE @fold\_id int

OPEN curs

FETCH NEXT FROM curs INTO @fold\_id

WHILE @@FETCH\_STATUS = 0 BEGIN

INSERT INTO @table SELECT \* FROM Func(@fold\_id)

FETCH NEXT FROM curs INTO @fold\_id

END

CLOSE curs

DEALLOCATE curs

END

RETURN

END;

