МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

ГЛАВНОЕ УПРАВЛЕНИЕ ПО ОБРАЗОВАНИЮ

МОГИЛЕВСКОГО ОБЛАСТНОГО ИСПОЛНИТЕЛЬНОГО КОМИТЕТА

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

**БАЗЫ ДАННЫХ И СИСТЕМЫ УПРАВЛЕНИЯ**

**БАЗАМИ ДАННЫХ**

ДОМАШНЯЯ КОНТРОЛЬНАЯ РАБОТА №1

Выполнил Учащаяся группы ПО-455 О.И.Любаль

Шифр 14

2023

**25 Назовите функции и команды управления базой данных.**

Реляционная база данных организует данные в структурированные таблицы для поиска общих точек данных. Таблицы похожи на папки в традиционной файловой системе, и каждая таблица хранит набор информации.

SQL — это язык, используемый для взаимодействия с реляционными базами данных. Команды SQL используются для выполнения основных операций с базой данных, таких как создание, чтение, обновление и удаление (CRUD) всего, что связано с базой данных.

Реляционные базы данных являются наиболее популярным типом баз данных, используемых в корпоративных условиях. Эти базы данных помогают обеспечивать работу некоторых крупнейших компаний мира, включая Facebook, Amazon и Google.

Команды также используются для создания, изменения и удаления баз данных и таблиц. В этой статье основное внимание будет уделено командам SQL, часто используемым в управлении базами данных.

Эти команды будут разделены на четыре категории:

* Команды языка обработки данных (DML)
* Команды языка определения данных (DDL)
* Команды языка управления данными (DCL)
* Операторы управления транзакциями (TCS)

**Команды DML** используются для манипулирования и выполнения операций с данными в базе данных. Примеры команд DML включают SELECT, INSERT и UPDATE.

**Команды DDL** используются для определения структуры базы данных. Вы можете изменить схему базы данных, создав новые таблицы и объекты или изменив их атрибуты (например, тип данных, имя таблицы и т. д.). Примеры команд DDL включают CREATE и ALTER.

**Команды DCL** используются для управления разрешениями пользователей и доступом к базе данных. Примеры команд DCL включают GRANT и REVOKE.

**Команды TCS** используются для управления транзакциями в базе данных. Транзакции — это единицы работы, которые можно либо зафиксировать, либо отменить. Примеры команд TCS включают COMMIT и ROLLBACK.

Команда CREATE DATABASE создает новую базу данных. База данных должна быть создана для хранения любых таблиц или данных.

Синтаксис:

CREATE DATABASE database\_name;

Пример:

CREATE DATABASE fruit\_database;

Команда ALTER DATABASE изменяет существующую базу данных. Например, ALTER DATABASE команда может добавлять или удалять файлы из базы данных.

Синтаксис:

ALTER DATABASE database\_name action;

Пример:

ALTER DATABASE fruit\_database ADD FILE 'mango.txt';

### 2. USE

USE выбирает базу данных. Эта команда часто используется для начала работы с только что созданной базой данных.

Синтаксис:

USE database\_name;

Пример:

USE fruit\_database;

После выбора базы данных все последующие команды SQL будут выполняться в этой базе данных.

USE команда может выбирать только уже созданные базы данных.

Если база данных с указанным именем не существует, то будет возвращена ошибка.

### 3. CREATE TABLE, ALTER TABLE и DROP TABLE

Команда CREATE TABLE создает новую таблицу в базе данных. Таблицу необходимо создать до того, как в нее можно будет вставить какие-либо данные.

Синтаксис:

CREATE TABLE table\_name (

    column\_name data\_type,

    column\_name data\_type,

    ...

);

Пример:

CREATE TABLE people\_table (

    id INTEGER,

    name VARCHAR(255),

    age INTEGER

);

В этом примере мы создаем таблицу people\_table с тремя столбцами: id, name и age.

Тип данных для каждого столбца должен быть указан. Некоторые распространенные типы данных включают INTEGER, VARCHAR и DATE.

Команда ALTER TABLE изменяет существующую таблицу. Например, ALTER TABLE команду можно использовать для добавления или удаления столбцов из таблицы.

Синтаксис:

ALTER TABLE table\_name action;

Пример:

ALTER TABLE people\_table

ADD email VARCHAR(255);

В этом примере мы добавляем email в people\_table таблицу новый столбец с именем. Необходимо указать тип данных для нового столбца.

Также можно использовать команду для изменения типа данных ALTER TABLE существующего столбца.

Синтаксис:

ALTER TABLE table\_name

MODIFY COLUMN column\_name data\_type;

Пример:

ALTER TABLE people\_table

MODIFY COLUMN last\_name

VARCHAR(128);

В этом примере мы изменяем last\_name столбец, чтобы он имел тип данных VARCHAR(128).

Обратите внимание, что вы не можете использовать эту ALTER TABLE команду для изменения типа данных столбца, если в этом столбце хранятся какие-либо данные.

Чтобы изменить тип данных столбца, вы должны сначала удалить все данные из этого столбца.

Синтаксис:

ALTER TABLE table\_name

DROP COLUMN column\_name;

Пример:

ALTER TABLE people\_table

DROP COLUMN email;

В этом примере мы удаляем email столбец из файла people\_table. Обратите внимание, что эта команда безвозвратно удалит все данные, хранящиеся в этом столбце.

Команда DROP TABLE удаляет всю таблицу из базы данных. Эта команда безвозвратно удалит все данные, хранящиеся в таблице.

Синтаксис:

DROP TABLE table\_name;

Пример:

DROP TABLE people\_table;

В этом примере мы удаляем people\_table таблицу из базы данных.

Важно быть осторожным при использовании DROP TABLE команды, так как ее нельзя отменить! После удаления таблицы все данные, хранящиеся в этой таблице, безвозвратно теряются.

Альтернативой DROP TABLE является использование TRUNCATE TABLE вместо этого. Эта команда удалит все данные из таблицы, но не саму таблицу.

Синтаксис:

TRUNCATE TABLE table\_name;

Пример:

TRUNCATE TABLE people\_table;

В этом примере мы удаляем все данные из people\_table таблицы. Сама таблица не удаляется, поэтому все сведения о столбцах сохраняются.

Команда INSERT INTO вставляет данные в таблицу.

Синтаксис:

INSERT INTO table\_name (column\_name, column\_name, ...)

VALUES (value, value, ...);

Пример:

INSERT INTO people\_table (id, name, age)

VALUES (NULL, 'Crystal', 64);

В этом примере мы вставляем новую строку в people\_table. Первый столбец в таблице id. Мы указали, что для этого столбца должно быть установлено значение NULL, что означает, что база данных автоматически сгенерирует уникальный идентификатор для этой строки.

Второй и третий столбцы таблицы — это name и age соответственно. Мы указали, что эти столбцы должны быть установлены для этой строки ’Crystal’ и 64для этой строки.

### 5. UPDATE

Команда UPDATE изменяет данные, уже сохраненные в таблице.

Синтаксис:

UPDATE table\_name

SET column\_name = value, column\_name = value, ...

WHERE condition;

Пример:

UPDATE people\_table

SET name = 'Crystal Sequel', age = 65

WHERE id = 100;

В этом примере мы обновляем строку id = 100 в people\_table таблице. Мы устанавливаем name столбец в ’Crystal Sequel’ и age столбец в 65.

Важно: Предложение WHERE обязательно при использовании UPDATE команды. Без WHERE предложения все строки в таблице будут обновлены!

### 6. DELETE

Команда DELETE удаляет данные из таблицы.

Синтаксис:

DELETE FROM table\_name

WHERE condition;

Пример:

DELETE FROM people\_table

WHERE id = 100;

В этом примере мы удаляем из people\_table таблицы строку с id=100.

Как и в случае с UPDATE командой, важно отметить, что это WHERE предложение необходимо при использовании DELETE команды. Как вы, возможно, уже догадались, все строки в таблице будут удалены без WHERE условия.

### 7. SELECT и FROM

Команда SELECT запрашивает данные FROM из таблицы.

Синтаксис:

SELECT column\_name, column\_name, ...

FROM table\_name

WHERE condition;

Пример:

SELECT name, age

FROM people\_table

WHERE id = 100;

В этом примере мы запрашиваем people\_table имя и возраст строки, где id=100.

Команды SELECT и FROM являются двумя наиболее важными командами SQL, поскольку они позволяют указывать и извлекать данные из базы данных.

### 8. ORDER BY

Команда ORDER BY сортирует результаты запроса.

Синтаксис:

SELECT column\_name, column\_name, ...

   FROM table\_name

   WHERE condition

   ORDER BY column\_name [ASC | DESC];

Пример:

SELECT name, age

   FROM people\_table

   WHERE id = 100

   ORDER BY age DESC;

В этом примере мы запрашиваем people\_table имя и возраст строки с помощью id=100. Затем мы сортируем результаты по возрасту в порядке убывания.

Команда ORDER BY часто используется вместе с SELECT командой для извлечения данных из таблицы в определенном порядке.

Важно отметить, что ORDER BY команда работает не только с числовыми данными — ее также можно использовать для сортировки текстовых данных по алфавиту!

ASC: По умолчанию порядок восходящий (A, B, C,… Z)

DESC: в порядке убывания (Z, Y, X,… A)

### 9. GROUP BY

Команда GROUP BY группирует результаты запроса по одному или нескольким столбцам.

Синтаксис:

SELECT column\_name, aggregate\_function(column\_name)

    FROM table\_name

    WHERE condition

    GROUP BY column\_name;

Пример:

SELECT name, count(\*)

   FROM people\_table

   WHERE country='US'

   GROUP BY names;

В этом примере мы запрашиваем people\_table все уникальные имена в таблице. Затем мы используем COUNT()функцию, чтобы подсчитать, сколько раз встречается каждое имя.

Команда GROUP BY часто используется с агрегатными функциями (такими как COUNT(), MIN(), MAX(), SUM()и т. д.) для группировки данных и расчета сводного значения.

Столбцы, указанные в GROUP BY предложении, также должны быть включены в SELECT предложение.

### 10. HAVING

Команда HAVING фильтрует результаты запроса на основе одной или нескольких агрегатных функций.

Синтаксис:

SELECT column\_name, aggregate\_function(column\_name)

    FROM table\_name

    WHERE condition

    GROUP BY column\_name

    HAVING condition;

Пример:

SELECT name, count(\*)

   FROM people\_table

   WHERE country='US'

   GROUP BY names

   HAVING count(\*) > 0;

В этом примере мы запрашиваем people\_table все уникальные имена в таблице. Затем мы используем COUNT()функцию, чтобы подсчитать, сколько раз встречается каждое имя.

Наконец, мы используем HAVING предложение, чтобы отфильтровать любые имена, которые не встречаются в таблице хотя бы раз.

Подобно GROUP BY предложению, мы также можем использовать его HAVING вместе с агрегатными функциями для фильтрации результатов запроса.

Агрегатные функции :

* COUNT(): подсчитывает количество строк в таблице
* MIN(): находит минимальное значение в столбце
* MAX(): находит максимальное значение в столбце
* SUM(): вычисляет сумму значений в столбце
* AVG(): вычисляет среднее значение значений в столбце

Столбцы, указанные в GROUP BY предложении, также должны быть включены в SELECT предложение.

HAVING очень похож на WHERE, но есть несколько важных отличий:

* WHERE используется для фильтрации данных перед агрегированием, а HAVING используется для фильтрации данных после агрегирования.
* WHERE может использоваться с агрегатными функциями, но HAVING может использоваться только со столбцами, включенными в GROUP BY предложение.
* WHERE применяется к отдельным строкам, а HAVING применяется к группам строк.

### 11. UNION и UNION ALL

Команда UNION объединяет результаты двух или более запросов в один набор данных. Он часто используется для объединения данных из нескольких таблиц в один набор данных.

Синтаксис:

SELECT column\_name FROM table\_name1

UNION

SELECT column\_name FROM table\_name2;

Пример:

SELECT names FROM employee\_table

UNION

SELECT email FROM people\_table;

В этом примере мы используем SELECT и UNION для запроса имен из, employee\_table а затем объединяем их с электронными письмами из people\_table в один набор результатов.

Количество и порядок столбцов должны быть одинаковыми во всех операторах SELECT, объединенных с UNION. Кроме того, все столбцы должны быть одного типа данных.

Чтобы объединить данные из нескольких таблиц, где количество и порядок столбцов не совпадают, в один набор данных, используйте UNION ALL вместо UNION.

Синтаксис:

SELECT column\_name FROM table\_name\_one

UNION ALL

SELECT column\_name FROM table\_name\_two;

Пример:

SELECT names FROM people\_table

UNION ALL ALL

SELECT email FROM people\_table;

В этом примере мы запрашиваем people\_table все уникальные имена в таблице. Затем мы используем UNION ALL команду для объединения этого набора данных с другим набором данных, содержащим все уникальные адреса электронной почты в таблице.

**56 Найдите фамилии и оценки юношей определенной группы, родившихся осенью 2000 года.**

Создадим необходимые таблицы и заполним их данными.

Содержимое таблицы Учащиеся представлено на рисунке 1.

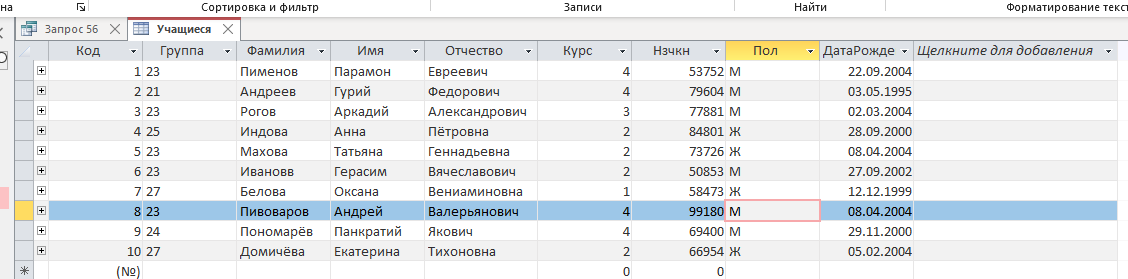


Рисунок 1. Содержимое таблицы Учащиеся.

Запустим мастер запросов и пошагово настроим его. Этапы настройки мастера представлены на рисунках 2-5.

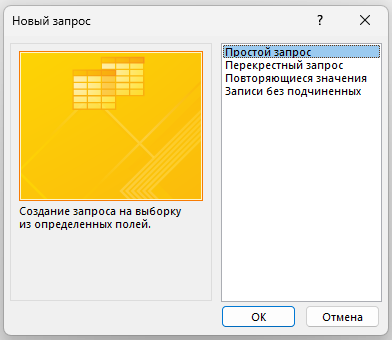


Рисунок 2. Выбор типа запроса.

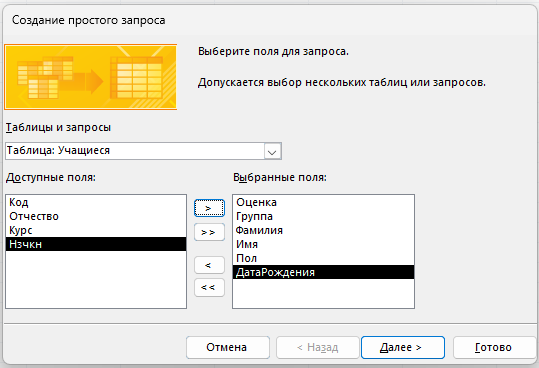


Рисунок 3. Выбор таблиц и полей.

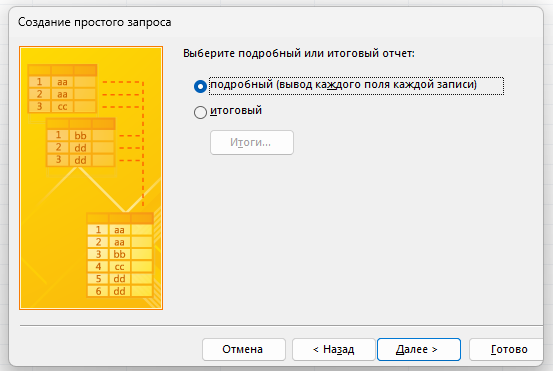


Рисунок 4. Выбор типа отчета.

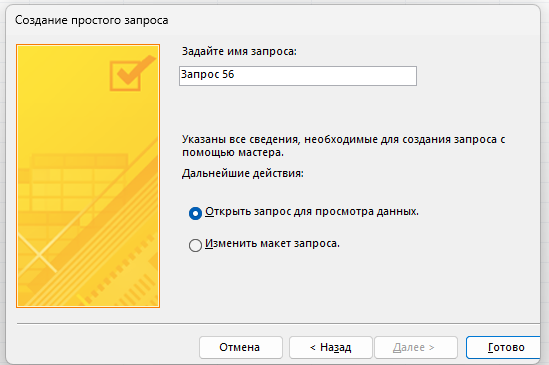


Рисунок 5. Завершение работы мастера.

Откроем запрос в режиме конструктора и укажем критерии отбора записей. Запрос в режиме конструктора представлен на рисунке 6.

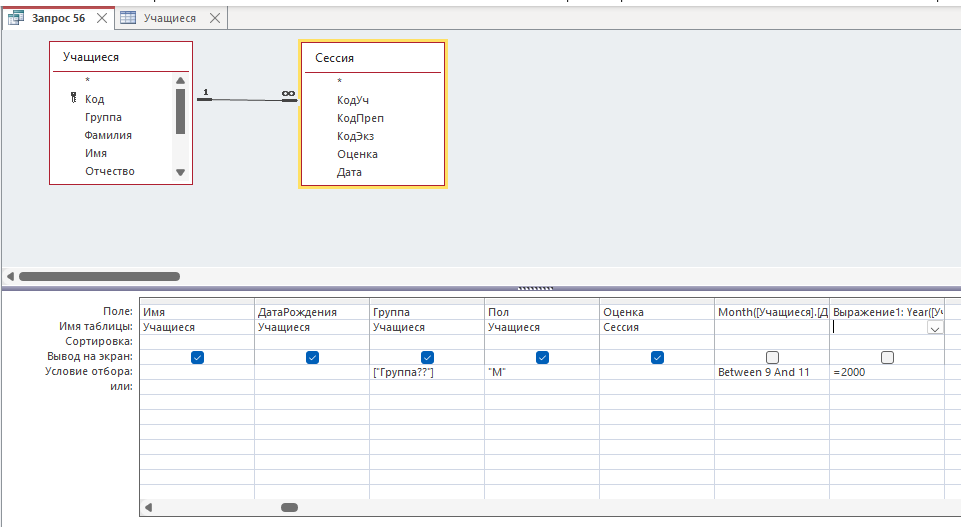


Рисунок 6. Запрос в режиме конструктора.

Сохраним запрос и запустим его на выполнение. Окно выбора группы представлено на рисунке 7.

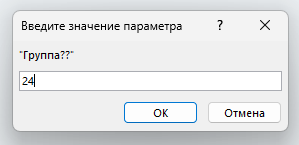


Рисунок 7. Окно выбора группы.

Выполненный запрос представлен на рисунке 8.

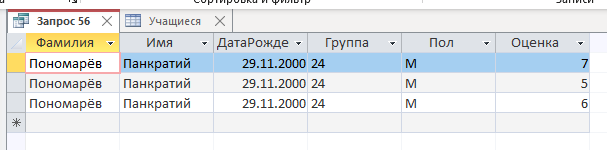


Рисунок 8. Выполненный запрос.

**62 Создайте запрос «Список N группы», содержащий отсортированные фамилии и имена учащихся группы. Номер группы берется из поля «Группа» загруженной формы «Список учащихся по группам».**

Сначала создадим простой запрос при помощи мастера запросов, рисунки 9 и 10.

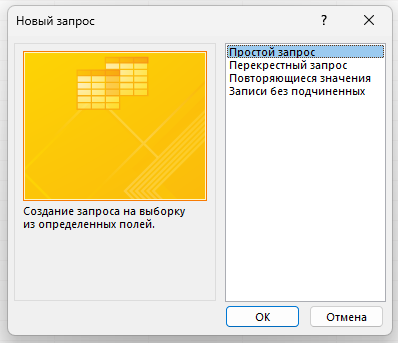


Рисунок 9. Создание запроса.

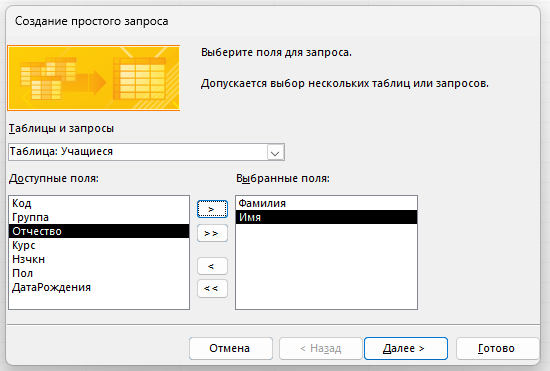


Рисунок 10. Указание полей запроса.

Затем откроем запрос в режиме конструктора и укажем поле Группа, которое будет являться параметром. Вид запроса в режиме конструктора представлен на рисунке 11.

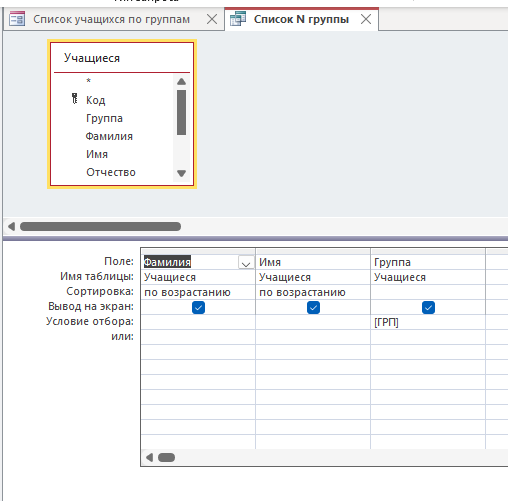


Рисунок 11. Вид запроса в режиме конструктора.

Создадим форму «Список учащихся по группам» и добавим в неё элемент управления – Кнопку. Вид формы в режиме конструктора представлен на рисунке 12.

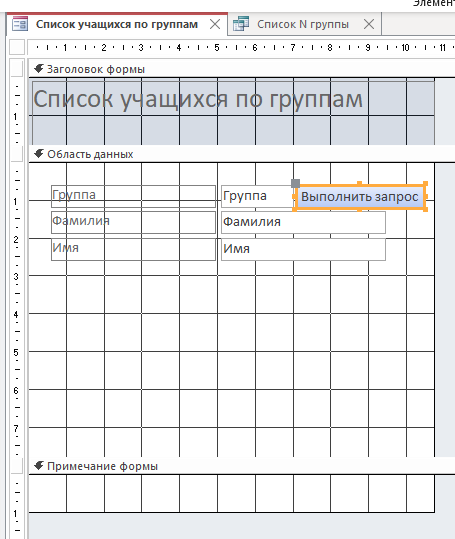


Рисунок 12. Вид формы в режиме конструктора.

Теперь необходимо к кнопке форму подключить вызов запроса с необходимым параметром. Напишем макрос, текст которого представлен на рисунке 13.

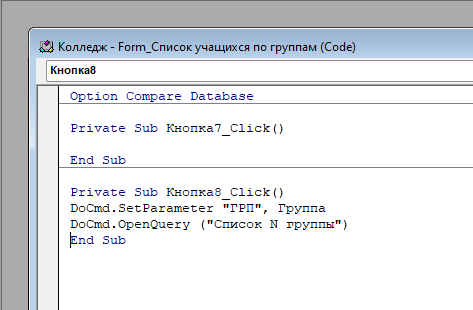


Рисунок 13. Макрос вызова запроса.

Теперь при нажатии на кнопку «Выполнить запрос» будет выведен запрос с переданным в качестве параметра номером группы из поля ввода формы. Результат теста работы представлен на рисунках 14-17.

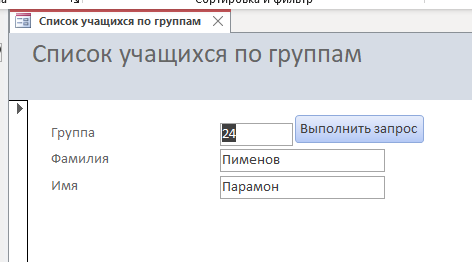


Рисунок 14. Проверка работы кнопки для группы 24.

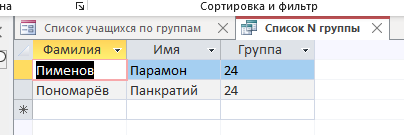


Рисунок 15. Результат выполнения запроса.

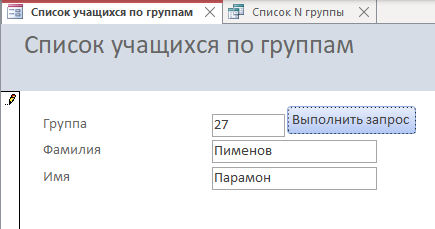


Рисунок 16. Проверка работы кнопки для группы 27.

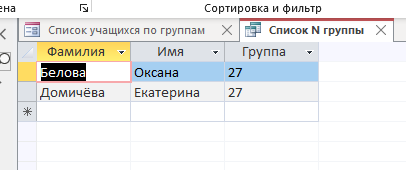


Рисунок 17. Результат выполнения запроса для группы 27

**115 Найдите самого «доброго» преподавателя (имеющего максимальное среднее из поставленных оценок) по данной дисциплине. Наименование дисциплины - параметр запроса.**

Создадим простой запрос для вывода среднего арифметического по дисциплине и преподавателю, результат работы запроса представлден на рисунке 18.

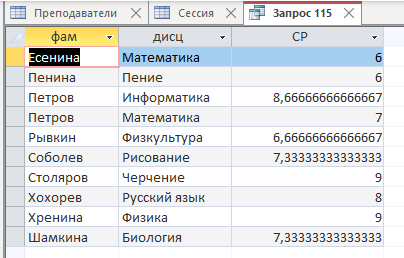


Рисунок 18. Результат работы простого запроса.

Откроем его в режиме SQL, вид запроса в этом режиме представлен на рисунке 19.

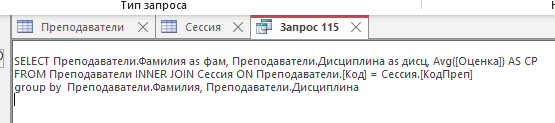


Рисунок 19. Вид запроса в режиме SQL.

Модифицируем SQL запрос, добавим запрос параметра, выбор максимального из средних значений путем сортировки и отброса ненужных значений. Вид модифицированного запроса в режиме SQL представлен на рисунке 20.

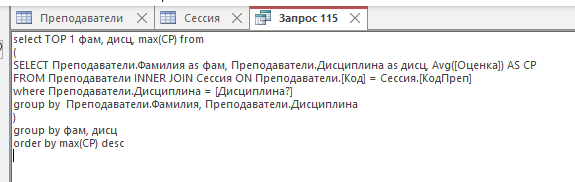


Рисунок 20. Модифицированный запрос.

Сохраним запрос и запустим его на выполнение, Указание параметра запроса представлено на рисунке 21.

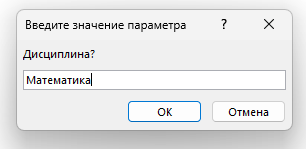


Рисунок 21. Указание параметра запроса.

Результат работы запроса представлен на рисунке 22.

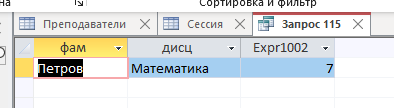


Рисунок 22. Результат работы запроса.

Сравнив результат работы запроса с данными рисунка 18, где выведены все записи о преподавателях и их средних баллах делаем вывод о правильности работы запроса.

**124 Создайте перекрестный запрос «Распределение оценок по экзамену», подсчитывающий количество различных оценок в группах по данному экзамену. Названия строк - номера групп. Названия столбцов -оценки. Код экзамена - параметр запроса.**

Прежде всего создадим простой запрос, который будет являться базой для перекрестного. Этапы создания запроса представлены на рисунках 23-25.

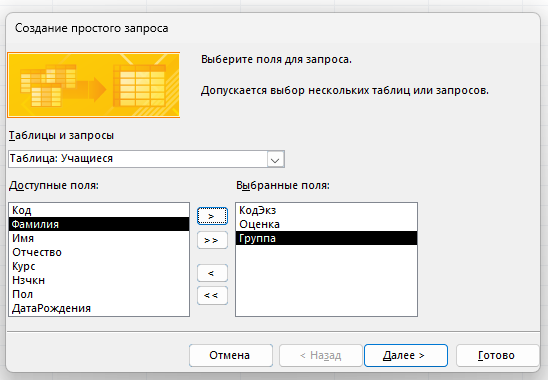


Рисунок 23. Создание запроса и указание полей.

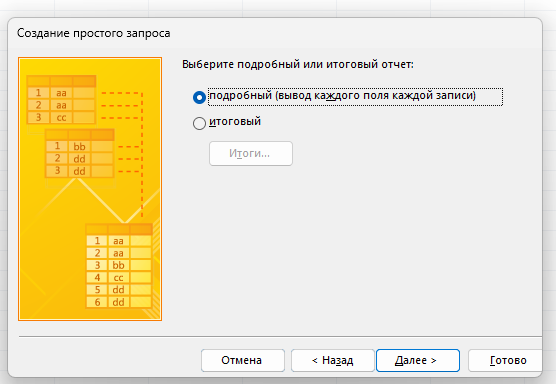


Рисунок 24. Выбор вида отчета.

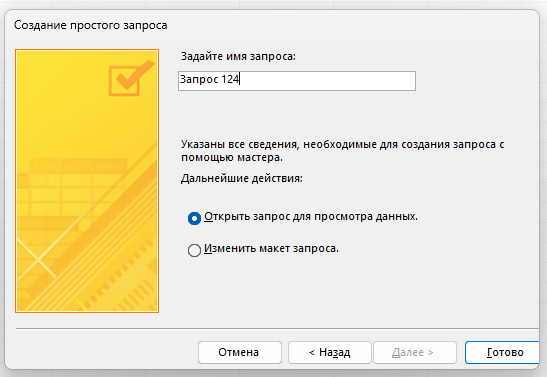


Рисунок 25. Завершение работы мастера.

На рисунке 26. Представлены результаты работы первичного запроса.

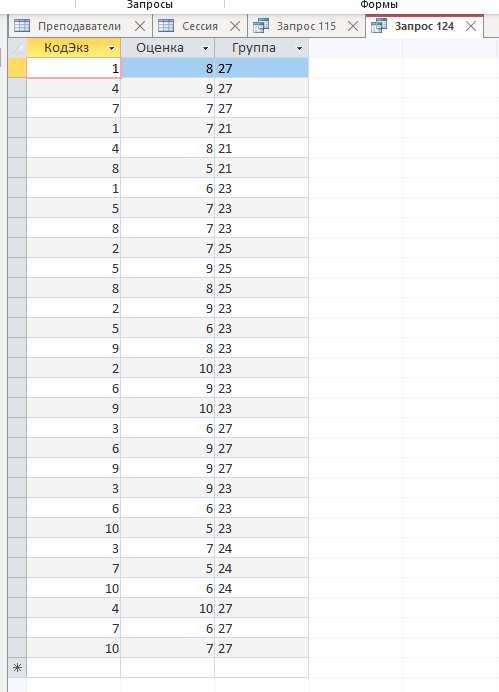


Рисунок 26. Результаты работы первичного запроса.

Далее запустим мастер создания перекрестных запросов и поэтапно настроим внешний вид отчета и его содержимое. Процесс создания перекрестного отчета представлен на рисунках 27-31.

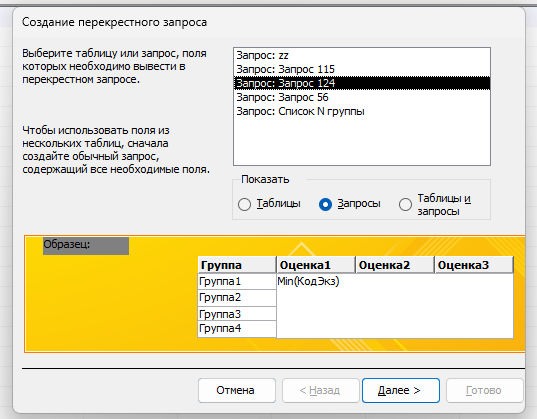


Рисунок 27. Выбор источника запроса.

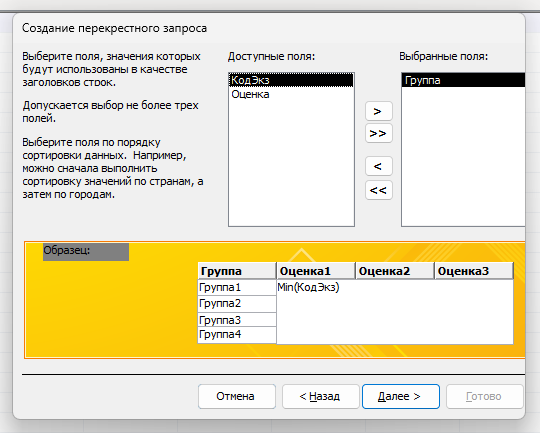


Рисунок 28. Указание заголовков строк.

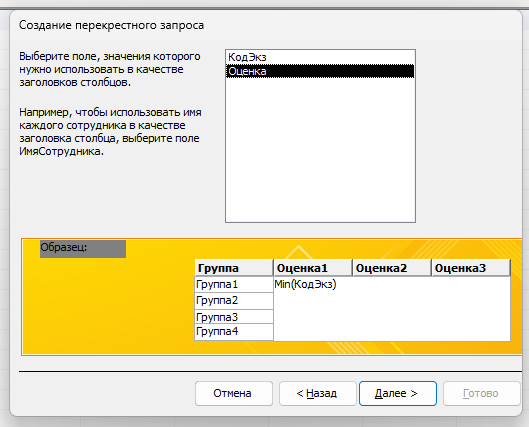


Рисунок 29. Указание заголовков столбцов.

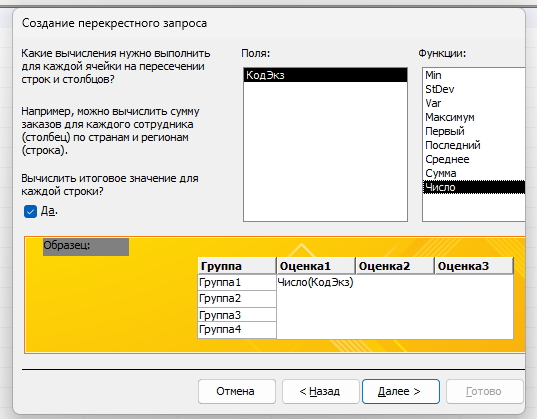


Рисунок 30. Указание содержимого таблицы запроса.

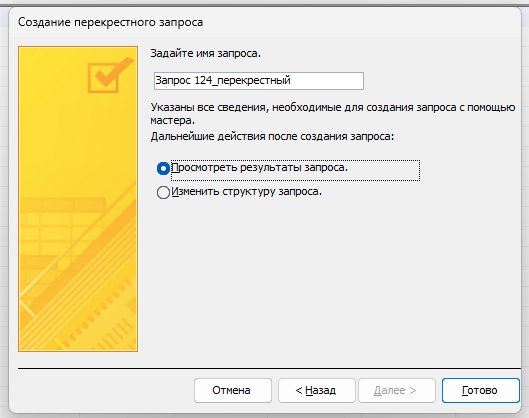


Рисунок 31 Завершение работы мастера.

На рисунке 32 представлен результат работы запроса.

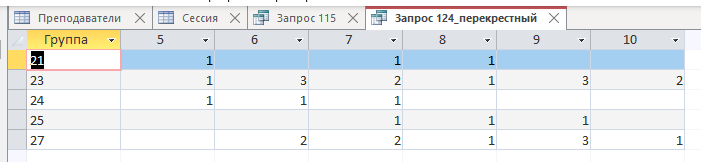


Рисунок 32. Результат работы запроса.

**171 Создайте форму для ввода информации в таблицу «Сессия». Для ввода данных в поля «Код учащихся», «Код экзамена» и «Код преподавателя» использовать поля со списком, подставляющие значения кодов из соответствующих таблиц, а для ввода данных в поле «Оценка» - простой список со значениями оценок.**

На первом этапе создадим пустую форму. Разместим на ней элементы управления. Макет формы представлен на рисунке 33.

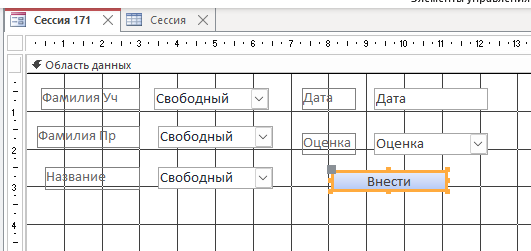


Рисунок 33. Макет формы.

Настроим поля со списком следующим образом: укажем источник данных, присоединенный столбец и подпись для каждого элемента, выбираемого из таблицы Преподавателей, учащихся и экзаменов. Свойства полей представлены на рисунках 34-36.

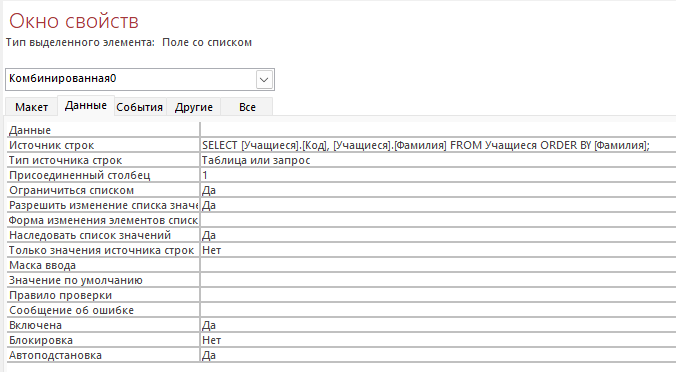


Рисунок 34. Окно свойств Учащиеся

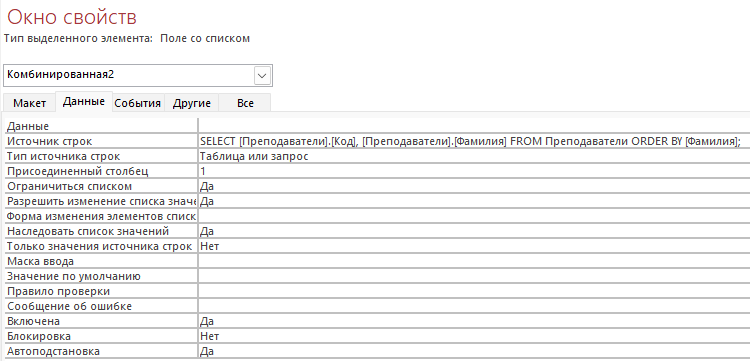


Рисунок 35. Окно свойств Преподаватели

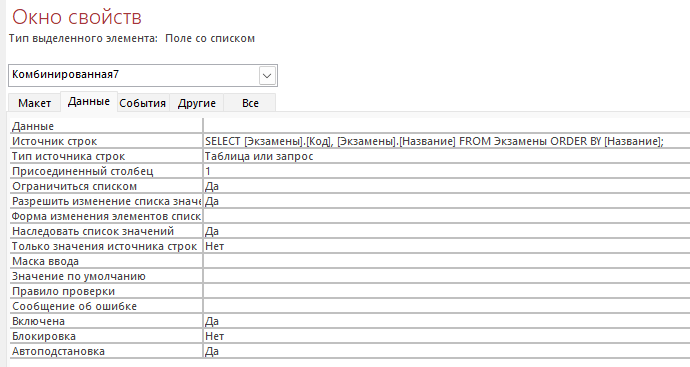


Рисунок 36. Окно свойств Экзамены

Окно свойств поля Оценка представлено на рисунке 37.



Рисунок 37. Окно свойств Оценка.

Для Кнопки укажем выполнение макроса, рисунок 38.

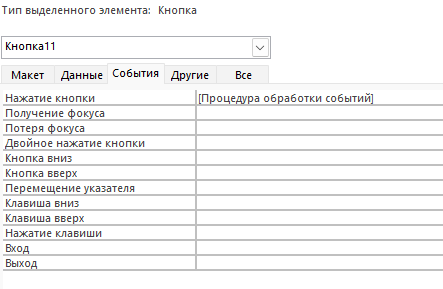


Рисунок 38. Указание действия для кнопки.

Текст макроса представлен на рисунке 39.

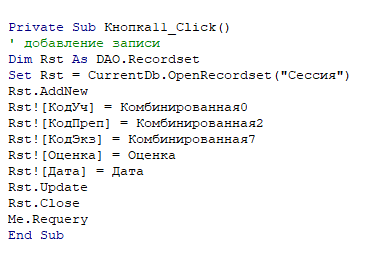


Рисунок 39. Текст макроса.

Проверим работу формы. Вид формы представлен на рисунке 40.

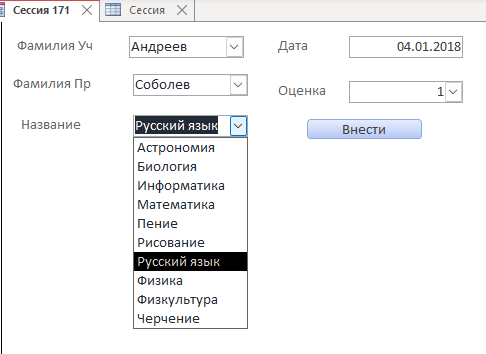


Рисунок 40. Вид Формы добавления записи.

При внесении оценки укажем 1955 год чтобы проще было увидеть результат в таблице сессия. Подготовленная для внесения форма представлена на рисунке 41.

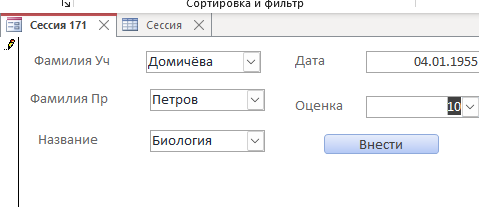


Рисунок 41. Форма с данными для внесения.

Внешний вид таблицы Сессия со внесенной записью представлен на рисунке 42.

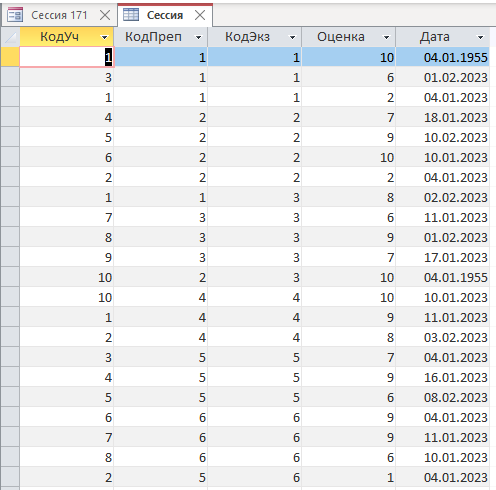


Рисунок 42. Внешний вид таблицы Сессия.

В первой строке внесенная запись.

**197 Постройте отчет «Итоги сессии в группе», содержащий поля «ФИО», названия дисциплин и «Средний балл». Он должен быть отсортирован по фамилиям. В конце отчета указать средний балл по каждому экзамену. Формат вывода средних значений - два десятичных знака после запятой. Номер группы— параметр отчета.**

Создадим запрос Учащиеся Запрос 197, в который выберем данные по учащимся, предметам и оценкам. Конструктор запроса представлен на рисунке 43.

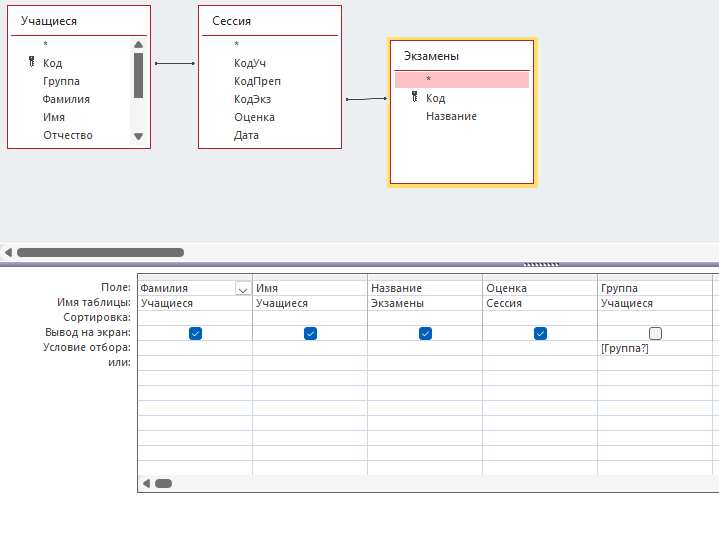


Рисунок 43. Конструктор запроса.

На его базе создадим перекрестный запрос. Конструктор запроса представлен на рисунке 44.

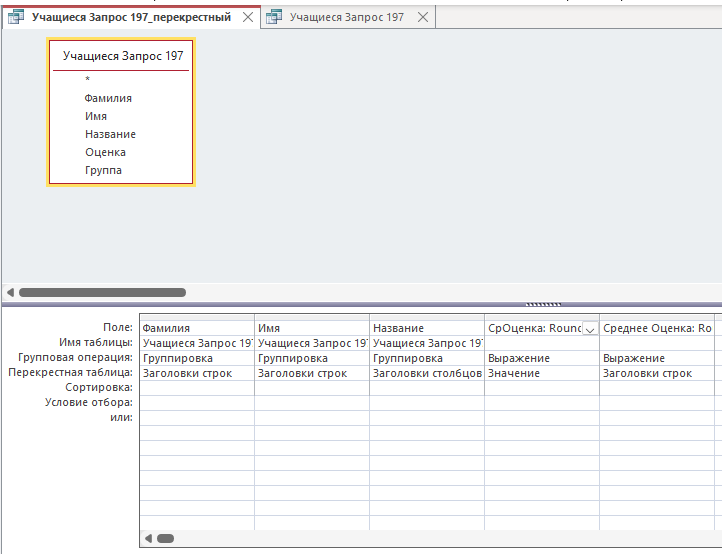


Рисунок 44. Перекрестный запрос, конструктор.

На базе перекрестного запроса создадим отчет. Конструктор отчета представлен на рисунке 45.

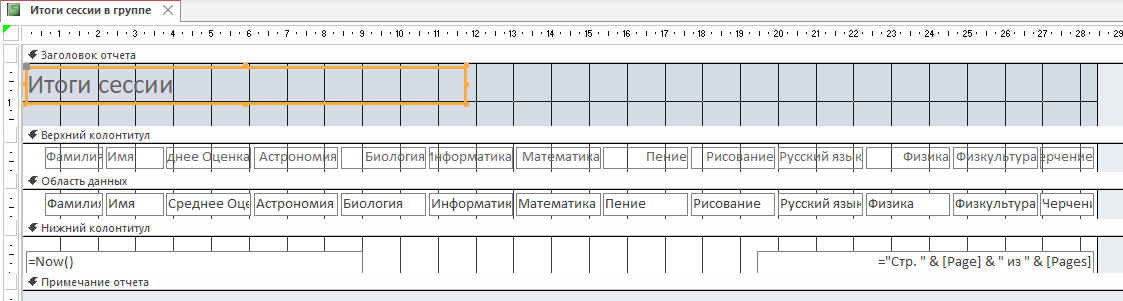


Рисунок 45. Конструктор отчета.

Результирующий отчет представлен на рисунке 46.



Рисунок 46. Итоговый вид отчета.

**Список использованных источников**

1 Бен-Ган, И. Microsoft SQL Server 2012. Основы T-SQL / И. Бен-Ган. М. : Эксмо, 2015. 400 с.

2 Дейт, К.Дж. Введение в системы баз данных / К.Дж. Дейт. М. : Вильямс, 2018. 1382 с.

3 Лазицкас, Е.А. Базы данных и системы управления базами данных / Е.А. Лазицкас, И.Н. Загуменникова, П.Г. Гилевский. Минск : РИПО, 2016. 268 с.

4 Федорова, Г. Разработка и администрирование баз данных / Г. Федорова. М. : Академия, 2015. 313 с.

5 SQL справочник / К. Кляйн [и др.]. СПб. : Символ-плюс, 2016. 56 с.