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.Е.Листопадова

Шифр 21

2023

**13 Дайте определение понятию нормализации базы данных. Охарактеризуйте нормальную форму Бойса-Кодда.**

## Реляционная база данных

В целом под базой данных можно понимать любой набор информации, которую можно найти в этой базе данных и воспользоваться ей, однако если говорить в контексте SQL, то речь будет идти, конечно, о реляционных базах данных, а что же это такое?

**Реляционная база данных** – это упорядоченная информация, связанная между собой определёнными отношениями.

Логически такая база данных представлена в виде таблиц, в которых и лежит вся эта информация.

## Нормализация баз данных

В реляционных базах данных есть такое понятия, как «Нормализация».

**Нормализация** – это процесс удаления избыточных данных.

Также нормализацию можно рассматривать и с позиции проектирования базы данных, в таком случае мы можем сформулировать определение нормализации следующим образом.

**Нормализация** – это метод проектирования базы данных, который позволяет привести базу данных к минимальной избыточности.

Избыточность устраняется, как правило, за счёт декомпозиции отношений (таблиц), т.е. разбиения одной таблицы на несколько.

### Зачем нормализовать базу данных?

У Вас может возникнуть вопрос – а зачем вообще нормализовать базу данных и бороться с этой избыточностью?

Дело в том, что избыточность данных создает предпосылки для появления различных аномалий, снижает производительность, и делает управление данными не гибким и не очень удобным. Отсюда можно сделать вывод, что нормализация нужна для:

* Устранения аномалий
* Повышения производительности
* Повышения удобства управления данными

Теперь давайте поговорим о самой избыточности данных, что же это такое.

**Избыточность данных** – это когда одни и те же данные хранятся в базе в нескольких местах, именно это и приводит к аномалиям.

Так как в этом случае необходимо добавлять, изменять или удалять одни и те же данные в нескольких местах. Например, если не выполнить операцию в каком-нибудь одном месте, то возникает ситуация, когда одни данные не соответствуют вроде как точно таким же данным в другом месте.

Каждая сущность должна храниться отдельно, а в случае необходимости использования этой сущности в другой таблице на нее делается всего лишь ссылка, т.е. выстраивается связь.

### Нормальные формы базы данных

В целом процесс нормализации базы данных выглядит следующим образом: мы, следуя определённым правилам и соблюдая определенные требования, проектируем таблицы в базе данных.

При этом все эти правила и требования можно сгруппировать в несколько наборов, и если спроектировать базу данных с соблюдением всех правил и требований, которые включаются в тот или иной набор, то база данных будет находиться в определённом состоянии, т.е. форме, и такая форма называется нормальная форма базы данных.

Иными словами, следуя определённым правилам и соблюдая определенные требования мы приводим базу данных к определенной нормальной форме.

**Нормальная форма базы данных** – это набор правил и критериев, которым должна отвечать база данных.

Каждая следующая нормальная форма содержит более строгие правила и критерии, тем самым приводя базу данных к определённой нормальной форме мы устраняем определённый набор аномалий.

Отсюда можно сделать вывод, что чем выше нормальная форма, тем меньше аномалий в базе будет.

**Процесс нормализации** – это последовательный процесс приведения базы данных к эталонному виду, т.е. переход от одной нормальной формы к следующей.

Иными словами, процесс перехода от одной нормальной формы к следующей – это усовершенствование базы данных. Так как если база данных находится в какой-то определённой нормальной форме – это означает, что в базе данных отсутствует определенный вид аномалий.

Существует 5 основных нормальных форм базы данных:

* Первая нормальная форма (1NF)
* Вторая нормальная форма (2NF)
* Третья нормальная форма (3NF)
* Четвертая нормальная форма (4NF)
* Пятая нормальная форма (5NF)

Однако выделяют еще дополнительные нормальные формы:

* Ненормализованная форма или нулевая нормальная форма (UNF)
* Нормальная форма Бойса-Кодда (BCNF)
* Доменно-ключевая нормальная форма (DKNF)
* Шестая нормальная форма (6NF)

Если объединить оба этих списка и упорядочить нормальные формы от менее нормализованной до самой нормализованной, т.е. начиная с формы, при которой база данных по своей сути не является нормализованной, и заканчивая самой строгой нормальной формой, то мы получим следующий перечень:

1. Ненормализованная форма или нулевая нормальная форма (UNF)
2. Первая нормальная форма (1NF)
3. Вторая нормальная форма (2NF)
4. Третья нормальная форма (3NF)
5. Нормальная форма Бойса-Кодда (BCNF)
6. Четвертая нормальная форма (4NF)
7. Пятая нормальная форма (5NF)
8. Доменно-ключевая нормальная форма (DKNF)
9. Шестая нормальная форма (6NF)

База данных считается нормализованной, если она находится как минимум в третьей нормальной форме (3NF).

В реальном мире нормализация до третьей нормальной формы (3NF) является обычной, стандартной практикой, так как 3NF устраняет достаточное количество аномалий, при этом производительность базы данных, а также удобство ее использования не снижается, что нельзя сказать о всех последующих формах.

Ситуации, при которых требуется нормализовать базу данных до четвертой нормальной формы (4NF), в реальном мире встречаются достаточно редко.

Между 3 и 4 нормальной формой есть еще и промежуточная нормальная форма, она называется – Нормальная форма Бойса-Кодда (BCNF). Иногда ее еще называют «Усиленная третья нормальная форма». Промежуточной или усиленной третьей нормальной формой ее называют потому, что ситуации, в которых могут предъявляться требования нормальной формы Бойса-Кодда, возникают не всегда, т.е. это некий частный случай, именно поэтому данная форма не включена в основную градацию. Однако во всех источниках эта форма рассматривается, поэтому и мы ее тоже рассмотрим.

Перед тем как переходить к процессу приведения таблиц базы данных до нормальной формы Бойса-Кодда, необходимо, чтобы эти таблицы уже находились в третьей нормальной форме

## Требования нормальной формы Бойса-Кодда

Требования нормальной формы Бойса-Кодда следующие:

* Таблица должна находиться в третьей нормальной форме. Здесь все как обычно, т.е. как и у всех остальных нормальных форм, первое требование заключается в том, чтобы таблица находилась в предыдущей нормальной форме, в данном случае в третьей нормальной форме;
* Ключевые атрибуты составного ключа не должны зависеть от неключевых атрибутов.

Отсюда следует, что требования нормальной формы Бойса-Кодда предъявляются только к таблицам, у которых первичный ключ составной. Таблицы, у которых первичный ключ простой, и они находятся в третьей нормальной форме, автоматически находятся и в нормальной форме Бойса-Кодда.

Главное правило нормальной формы Бойса-Кодда (BCNF) звучит следующим образом:

Часть составного первичного ключа не должна зависеть от неключевого столбца.

## Пример приведения таблиц базы данных к нормальной форме Бойса-Кодда

Представим, что у нас есть организация, которая реализует множество различных проектов. При этом в каждом проекте работа ведётся по нескольким функциональным направлениям, в каждом из которых есть свой куратор. Сотрудник может быть куратором только того направления, на котором он специализируется, т.е. если сотрудник программист, он не может курировать в проекте направление, связанное с бухгалтерией.

Допустим, что нам нужно хранить информацию о кураторах всех проектов по каждому направлению.

В итоге мы реализуем следующую таблицу, в которой первичный ключ составной «Проект + Направление», так как в каждом проекте есть несколько направлений работы и поэтому, зная только проект, мы не можем определить куратора направления, так же как зная только направление, мы не сможем определить куратора, нам нужно знать и проект и направление, чтобы определить куратора этого направления в этом проекте.

*Таблица проектов и кураторов.*

|  |  |  |
| --- | --- | --- |
| **Проект** | **Направление** | **Куратор** |
| 1 | Разработка | Иванов И.И. |
| 1 | Бухгалтерия | Сергеев С.С. |
| 2 | Разработка | Иванов И.И. |
| 2 | Бухгалтерия | Петров П.П. |
| 2 | Реализация | John Smith |
| 3 | Разработка | Андреев А.А. |

Наша таблица находится в третьей нормальной форме, так как у нас есть первичный ключ, а неключевой столбец зависит от всего ключа, а не от какой-то его части.

Но в данном случае таблица не находится в нормальной форме Бойса-Кодда, дело в том, что зная куратора, мы можем четко определить, какое направление он курирует, иными словами, часть составного ключа, т.е. *«Направление»*, зависит от неключевого атрибута, т.е. *«Куратора»*.

Чтобы привести данную таблицу к нормальной форме Бойса-Кодда, необходимо, как всегда сделать декомпозицию данного отношения, т.е. разбить эту таблицу на несколько таблиц.

*Таблица кураторов.*

|  |  |  |
| --- | --- | --- |
| **Идентификатор**  **куратора** | **ФИО** | **Направление** |
| 1 | Иванов И.И. | Разработка |
| 2 | Сергеев С.С. | Бухгалтерия |
| 3 | Петров П.П. | Бухгалтерия |
| 4 | John Smith | Реализация |
| 5 | Андреев А.А. | Разработка |

*Таблица связи кураторов и проектов.*

|  |  |
| --- | --- |
| **Проект** | **Идентификатор куратора** |
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |
| 2 | 3 |
| 2 | 4 |
| 3 | 5 |

Таким образом, в таблице кураторов у нас хранится список кураторов и их специализация, т.е. направление, которое они могут курировать, а в таблице связи кураторов и проектов отражается связь проектов и кураторов.

**55 Найдите фамилии и адреса девушек определенной группы, родившихся зимой 1999 года.**

Создадим базу данных, создадим таблицы, заполним их данными и настроим связи базы данных. Схема связей базы данных представлена на рисунке 1.

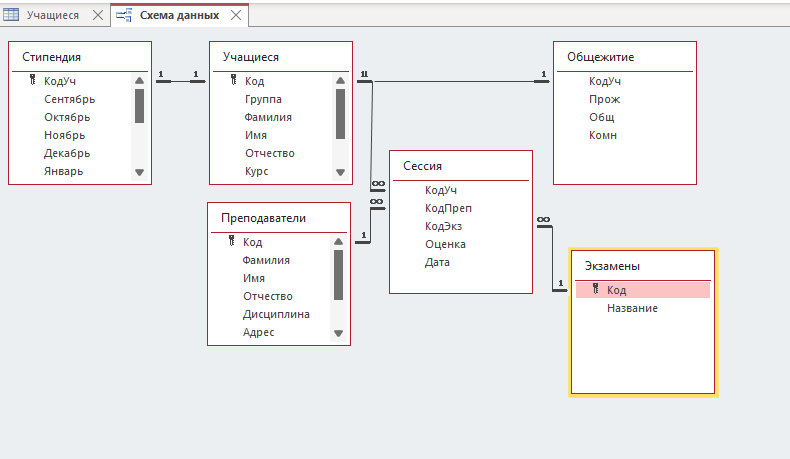


Рисунок 1. Схема базы данных.

Содержимое таблицы учащиеся представлено на рисунке 2.

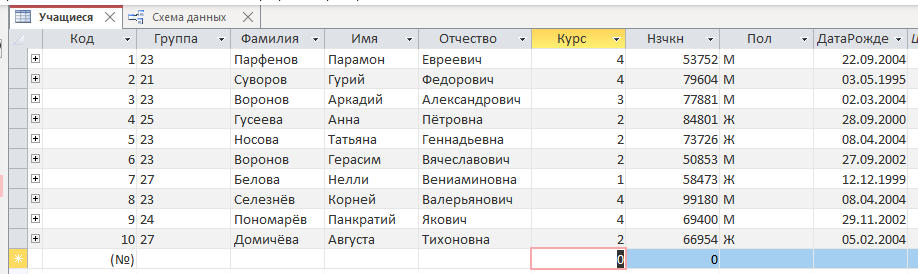


Рисунок 2. Содержимое таблицы Учащиеся.

Создадим запрос для выбора необходимых данных в режиме конструктора. Запрос представлен на рисунке 3.

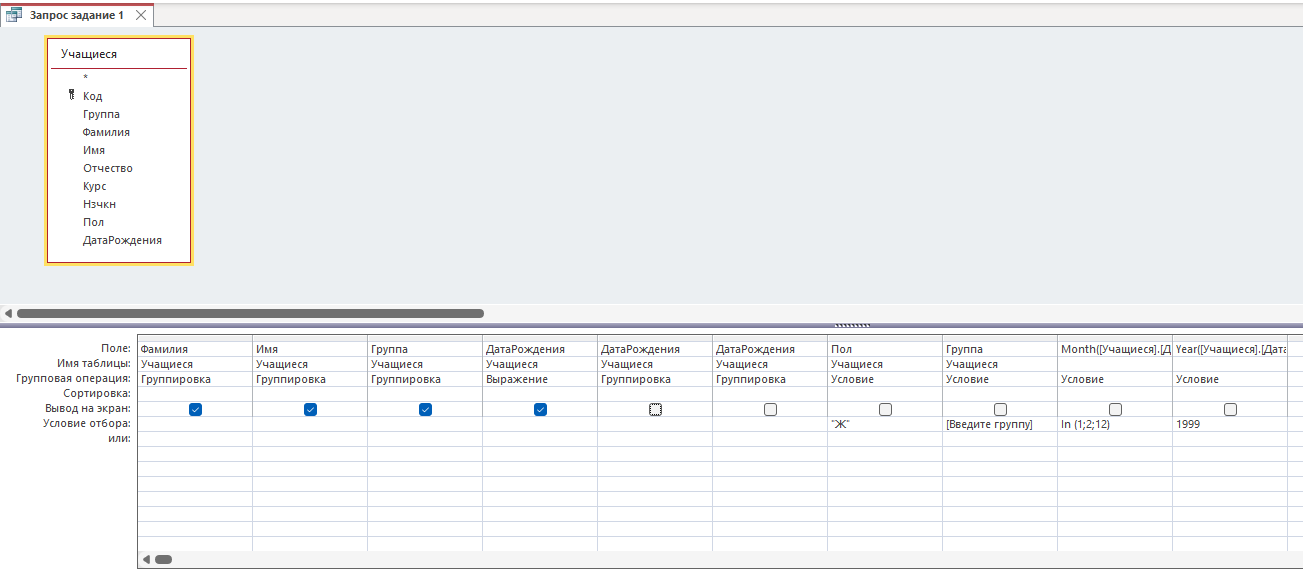


Рисунок 3. Запрос в режиме конструктора.

SQL текст запроса:

SELECT Учащиеся.Фамилия, Учащиеся.Имя, Учащиеся.Группа, Учащиеся.ДатаРождения FROM Учащиеся

WHERE (((Учащиеся.Пол)="Ж") AND ((Учащиеся.Группа)=[Введите группу]) AND ((Month([Учащиеся].[ДатаРождения])) In (1,2,12)) AND ((Year([Учащиеся].[ДатаРождения]))=1999))

GROUP BY Учащиеся.Фамилия, Учащиеся.Имя, Учащиеся.Группа, Учащиеся.ДатаРождения, Учащиеся.ДатаРождения;

Выполним запрос, для этого необходимо указать группу. Выполнение запроса представлено на рисунках 4-5.

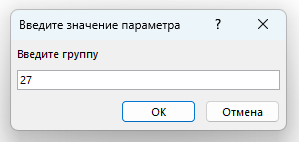


Рисунок 4. Указание группы.

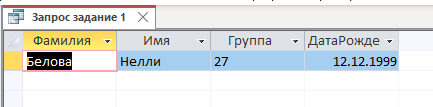


Рисунок 5. Результат выполнения запроса.

**67 Найдите фамилии учащихся, имеющих средний балл больше заданного числа X. Число X - параметр запроса.**

Сначала выведем список всех студентов и их средний балл. Средний балл студентов представлен на рисунке 6.

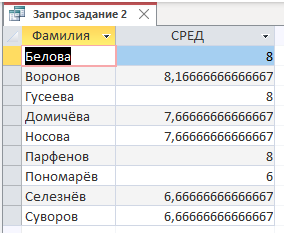


Рисунок 6. Средний балл.

Для выбора среднего балла студентов использовался SQL запрос:

SELECT Учащиеся.Фамилия, Avg(Сессия.Оценка) AS СРЕД

FROM Учащиеся INNER JOIN Сессия ON Учащиеся.Код = Сессия.КодУч

group by Учащиеся.Фамилия

Теперь из этого запроса необходимо выбрать студентов со средним баллом выше указанного, допишем запрос следующим образом:

SELECT Учащиеся.Фамилия, СРЕД

from(

SELECT Учащиеся.Фамилия, Avg(Сессия.Оценка) AS СРЕД

FROM Учащиеся INNER JOIN Сессия ON Учащиеся.Код = Сессия.КодУч

group by Учащиеся.Фамилия

) where сред>[Введите балл]

Запрос в режиме конструктора представлен на рисунке 7.

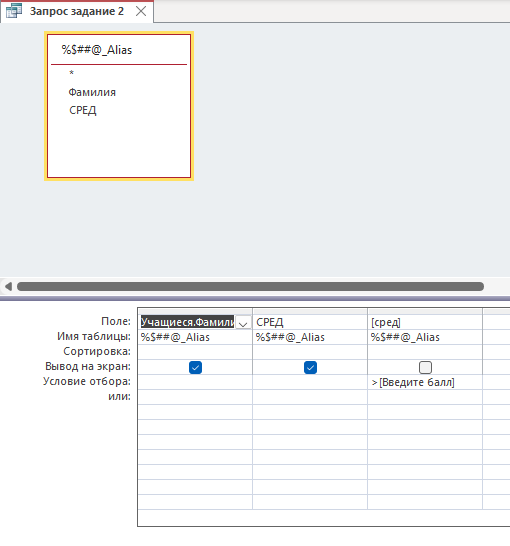


Рисунок 7. Запрос в режиме конструктора.

Результат работы запроса для балла 7 представлен на рисунке 8.

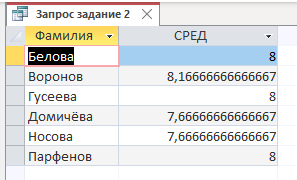


Рисунок 8. Результат работы запроса для балла 7.

**98 Создайте список учащихся -однофамильцев, содержащий их фамилии, имена и номера групп.**

Для выполнения задания сделаем запрос в котором одной таблице Учащиеся назначим 2 псевдонима – а1 и а2. И выберем из этих таблиц записи где совпадают фамилии а имена не совпадают. SQL запрос будет выглядеть так:

SELECT а1.Фамилия, а1.Имя, а1.Группа

FROM Учащиеся AS а1, Учащиеся AS а2

WHERE (((а1.Имя)<>[а2].[Имя]) AND ((а1.Фамилия)=[а2].[Фамилия]));

Конструктор запроса представлен на рисунке 9.

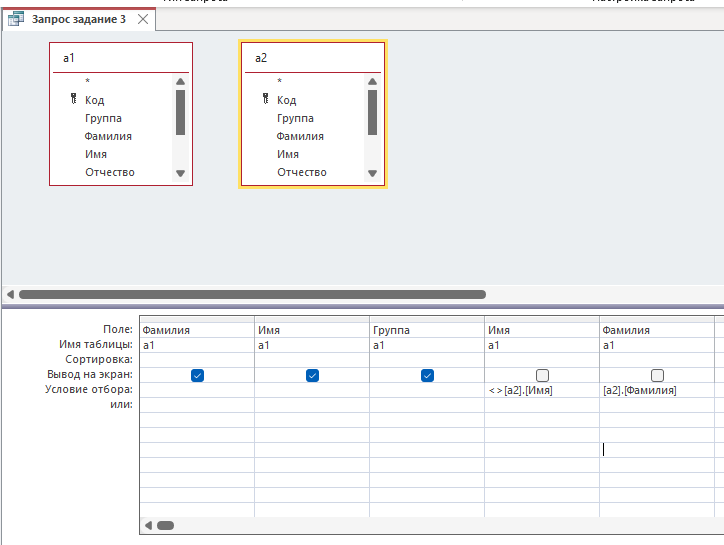


Рисунок 9. Запрос в режиме конструктора.

Результат выполнения запроса представлен на рисунке 10.

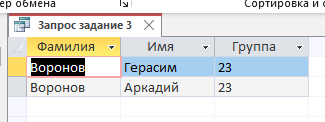


Рисунок 10. Результат выполнения запроса.

**126 Создайте перекрестный запрос «Средние баллы в группах», дающий значения среднего балла в группах по каждому экзамену. Названия строк - экзамены. Названия столбцов - номера групп. Формат вывода среднего балла - два десятичных знака после запятой.**

Сначала создадим запрос в котором выберем средний палл по показателям Группа и Предмет. Конструктор запроса представлен на рисунке 11.

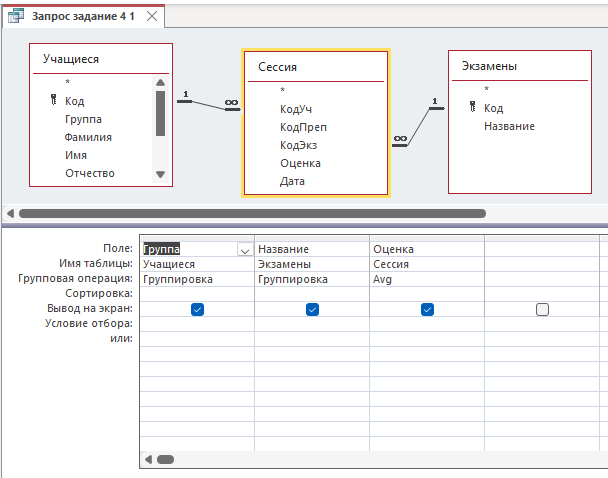


Рисунок 11. Конструктор запроса.

Создадим и настроим перекрестный запрос при помощи мастера. Этапы создания запроса представлены на рисунках 11-17.

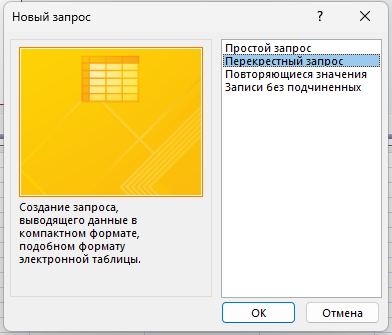


Рисунок 12 Выбор типа запроса

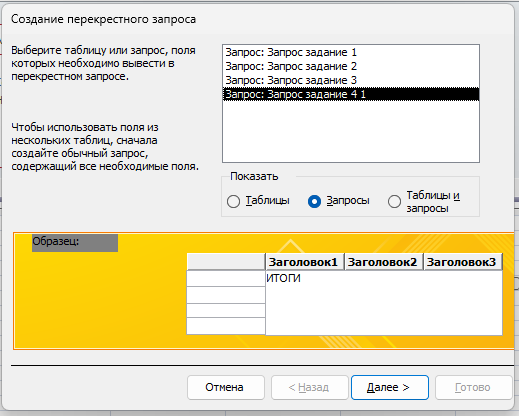


Рисунок 13. Выбор источника запроса.

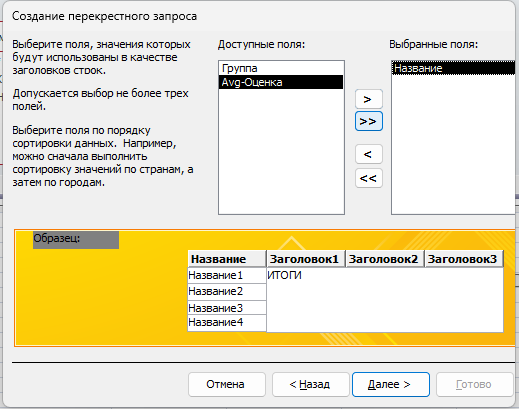


Рисунок 14. Выбор заголовков строк.

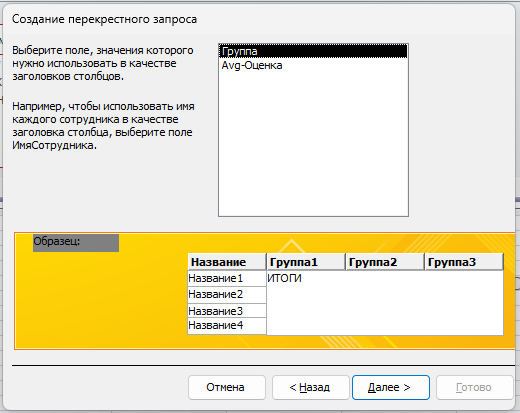


Рисунок 15. Выбор заголовков столбцов.

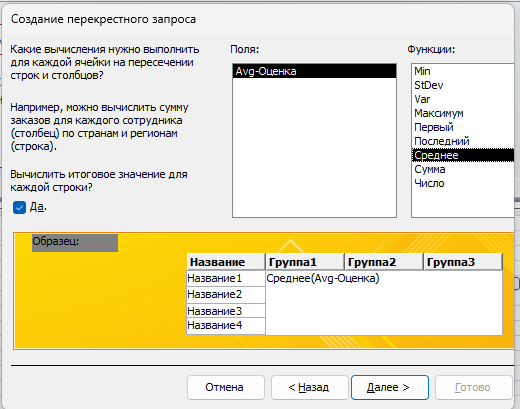


Рисунок 16. Выбор значения на пересечении.

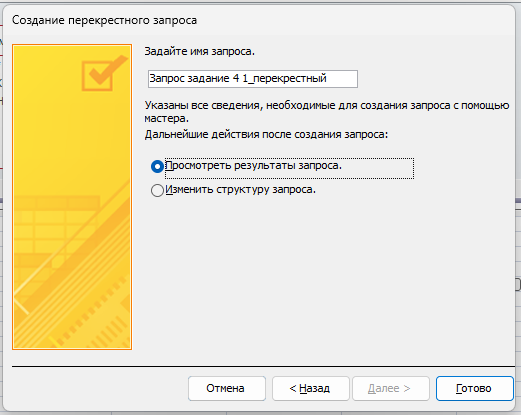


Рисунок 17. Задание имени запроса.

Результат запроса представлен на рисунке 18.

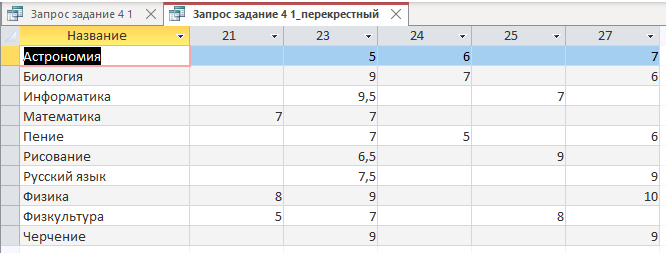


Рисунок 18. Результат выполнения запроса.

**160 Постройте составную форму «Преподаватели и учащиеся». Главная форма должна содержать фамилию преподавателя и название дисциплины, а подчиненная форма - список учащихся, у которых он принимал экзамен, с номерами их групп и полученными оценками. Список должен быть отсортирован по группам и фамилиям учащихся.**

**206 Постройте отчет «Итоги сдачи дисциплины». Он должен включать поле «Группа» с номерами учебных групп, а также поля «10», «9»,… ,«2» и «1», содержащие количество оценок, полученных по дисциплине в каждой группе. Дисциплина-параметр.**

**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.