

# Основы информационных технологий

Глава 7 (продолжение II).  
Реляционная алгебра.  
Язык SQL

# § 8. Операции реляционной алгебры

---

А.Е. АНИСИМОВ

- Реляционная теория баз данных основана на понятии реляционной алгебры (алгебры отношений). Объектами в ней выступают отношения.
- Э.Ф. Кодд предложил 8 операций реляционной алгебры (реляционных операций). Каждая из них основана на отношениях как аргументах и результатом является всегда отношение.

- **Операции реляционной алгебры:**
- **Теоретико-множественные:**
  1. Объединение
  2. Пересечение
  3. Разность
  4. Расширенное декартово произведение
- **Специальные:**
  5. Фильтрация (горизонтальный выбор)
  6. Проекция (вертикальный выбор)
  7. Условное соединение
  8. Деление.

# Теоретико- множественные операции

- **1. Объединение**
- Опр. **Объединением** двух отношений с называют отношение, которое состоит из кортежей, принадлежащих первому, либо второму отношениям, либо обоим одновременно.
- $R_1 \cup R_2 = \{r \mid r \in R_1 \vee r \in R_2\}$
- Атрибуты отношений должны быть эквиваленты!

- 2. Пересечение

- Опр. Пересечением двух отношений с эквивалентными схемами называют отношение, которое состоит из кортежей, принадлежащих одновременно первому и второму отношениям.

- $R_1 \cap R_2 = \{r \mid r \in R_1 \wedge r \in R_2\}$

- **3. Разность**

- Опр. **Разностью** двух отношений с эквивалентными схемами называют отношение, которое состоит из кортежей, принадлежащих первому отношению и не принадлежащих второму отношению.

- $$R_1 \setminus R_2 = \{r \mid r \in R_1 \wedge r \notin R_2\}$$

## § 8. Операции реляционной алгебры

M (мужчины)	
НСБ	Фамилия
01	Иванов
02	Петров
03	Сидоров

O (отличники)	
НСБ	Фамилия
02	Петров
04	Кузнецова

M ∪ O	
НСБ	Фамилия
01	Иванов
02	Петров
03	Сидоров
04	Кузнецова

M ∩ O	
НСБ	Фамилия
02	Петров

M \ O	
НСБ	Фамилия
01	Иванов
03	Сидоров

- 4. Расширенное декартово произведение.
- Опр. Расширенным декартовым произведением отношения  $R_1$  и отношения  $R_2$  называется отношение, кортежи которого получены сцеплением **каждого** кортежа отношения  $R_1$  с **каждым** кортежем отношения  $R_2$ .
- $R_1 \otimes R_2 = \{(p, q) \mid p \in R_1 \wedge q \in R_2\}$

## § 8. Операции реляционной алгебры

- Пример.

S (студенты)	
НСБ	Фамилия
01	Иванов
02	Петров
03	Сидоров

G (группы)	
Индекс	Специальность
ИТ-21	Информационные технологии
ПИ-21	Прикладная информатика

S $\otimes$ G			
НСБ	Фамилия	Индекс	Специальность
01	Иванов	ИТ-21	Информационные технологии
01	Иванов	ПИ-21	Прикладная информатика
02	Петров	ИТ-21	Информационные технологии
02	Петров	ПИ-21	Прикладная информатика
03	Сидоров	ИТ-21	Информационные технологии
03	Сидоров	ПИ-21	Прикладная информатика

# Специальные операции

- **5. Фильтрация (горизонтальный выбор, выборка)**
- Опр. **Фильтрацией** отношения  $R$  по условию  $\alpha(r)$  называется отношение, содержащее только те кортежи из  $R$ , для которых истинно  $\alpha(r)$ .
- $R[\alpha(r)] = \{r \mid r \in R \wedge \alpha(r)\}$
- Условие  $\alpha(r)$  может содержать термы сравнения ( $=$ ,  $<$ ,  $>$ ,  $<=$ ,  $>=$ ,  $<$ ,  $>$ ), константы, логические связки, скобки.

## § 8. Операции реляционной алгебры

- Пример.

S	
Фамилия	Пол
Иванов	М
Петрова	Ж
Сидоров	М
Кузнецова	Ж

S [Пол = "Ж"]	
Фамилия	Пол
Петрова	Ж
Кузнецова	Ж

- **6. Проекция (вертикальный выбор)**
- Опр. Проекцией отношения  $R(A_1, A_2, \dots, A_n)$  по подмножеству его атрибутов  $B = (A_{i1}, A_{i2}, \dots, A_{ik}) \subseteq (A_1, A_2, \dots, A_n)$  называется отношение со схемой, соответствующей набору атрибутов  $B$ , содержащее кортежи, получаемые их кортежей исходного отношения  $R$  путем удаления из них значений, не принадлежащих атрибутам из  $B$ .
- $S_R = (A_1, A_2, \dots, A_n),$   
 $B = (A_{i1}, A_{i2}, \dots, A_{ik}) \subseteq S_R$
- $R[B] = \{r \mid r = \langle r_{i1}, r_{i2}, \dots, r_{ik} \rangle \wedge r_{it} \in A_{it} \wedge$   
 $(\exists p \in R: p = \langle r_1, r_2, \dots, r_n \rangle \forall j \in 1..k \exists z: r_{ij} = r_z) \}$
-

## § 8. Операции реляционной алгебры

### • Пример

Book				
ИнвN	Автор	Название	Из-во	ГодВып
01234	Попов В.Н.	Язык Паскаль	ФиС	1994
23567	Вирт Н.	Алгоритмы и структуры данных	Мир	1987
34562	Вирт Н.	Алгоритмы и структуры данных	Мир	2003
91873	Карпова Т.	Базы данных	Питер	2002

Book [Автор, Название]	
Автор	Назв
Попов В.Н.	Язык Паскаль
Вирт Н.	Алгоритмы и структуры данных
Карпова Т.	Базы данных

- **7. Условное соединение**

- Опр. **Условным соединением** отношения  $R_1(A_1, A_2, \dots, A_n)$  и отношения  $R_2(B_1, B_2, \dots, B_m)$ , некоторые атрибуты которых сравнимы, по условию  $\alpha(r)$  называется подмножество декартова произведения  $R_1 \otimes R_2$ , кортежи которого удовлетворяют условию  $\alpha(r)$ .

- $R_1(A_1, A_2, \dots, A_n), R_2(B_1, B_2, \dots, B_m)$ , и пусть  $V = (A_{i_1}, A_{i_2}, \dots, A_{i_k}), V \subseteq S_{R_1} \wedge V \subseteq S_{R_2}, \alpha(r)$  - предикат на  $V$ . Тогда

- $R_1 [\alpha(r)] R_2 = \{ \langle r_1, r_2 \rangle \mid r_1 \in R_1 \wedge r_2 \in R_2 \wedge \alpha(r_1 \cap r_2) \}$

## § 8. Операции реляционной алгебры

### 1. Пример.

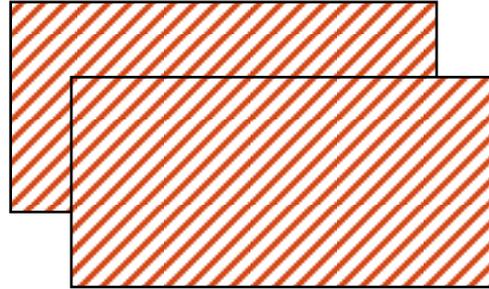
S		
НСБ	Фамилия	Группа
01	Иванов	39-21
02	Петров	39-21
03	Сидоров	ИТ-21
04	Кузнецов	ИТ-21

P	
Группа	Предмет
39-21	Базы Данных
39-21	МИР
ИТ-11	Алгебра

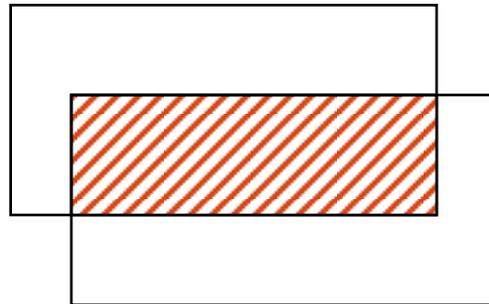
S [S.Группа=P.Группа] P			
НСБ	Фамилия	Группа	Предмет
01	Иванов	39-21	Базы Данных
01	Иванов	39-21	МИР
02	Петров	39-21	Базы Данных
02	Петров	39-21	МИР
03	Сидоров	ИТ-21	Алгебра
04	Кузнецов	ИТ-21	Алгебра

1. Поясняющие диаграммы (1-2):

1. Объединение

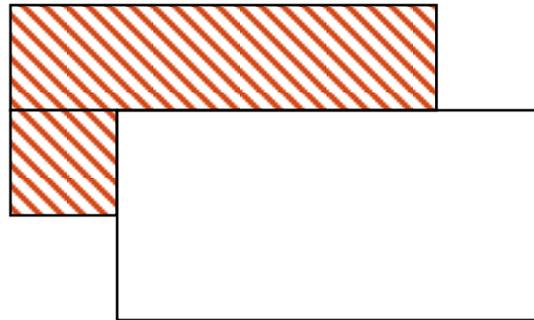


2. Пересечение

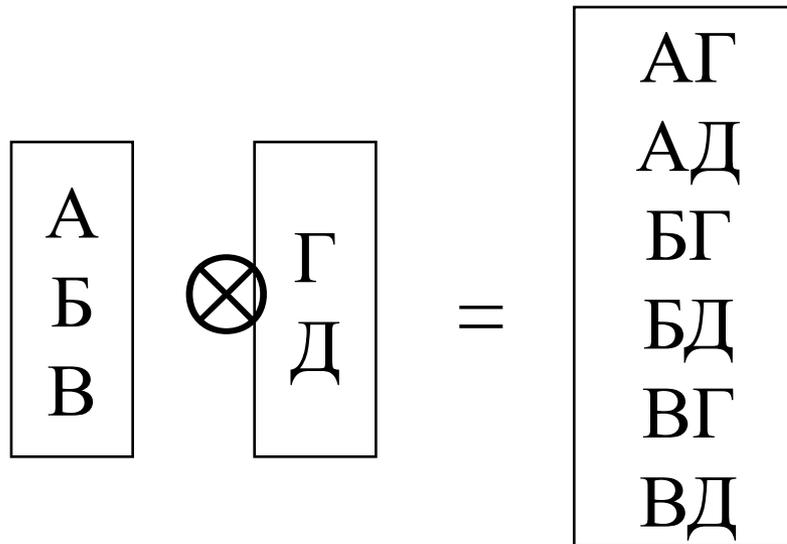


1. Поясняющие диаграммы (3-4):

3. Разность

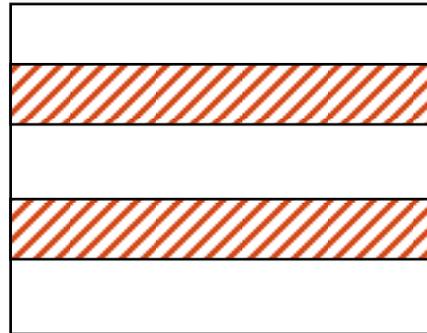


4. Декартово произведение

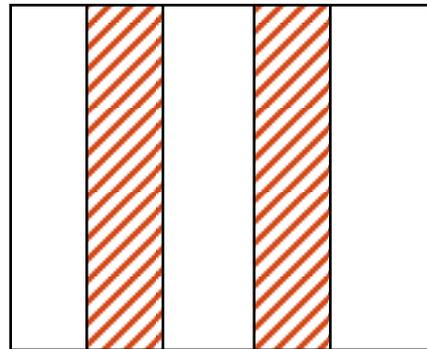


1. Поясняющие диаграммы (5-6):

5. Фильтрация

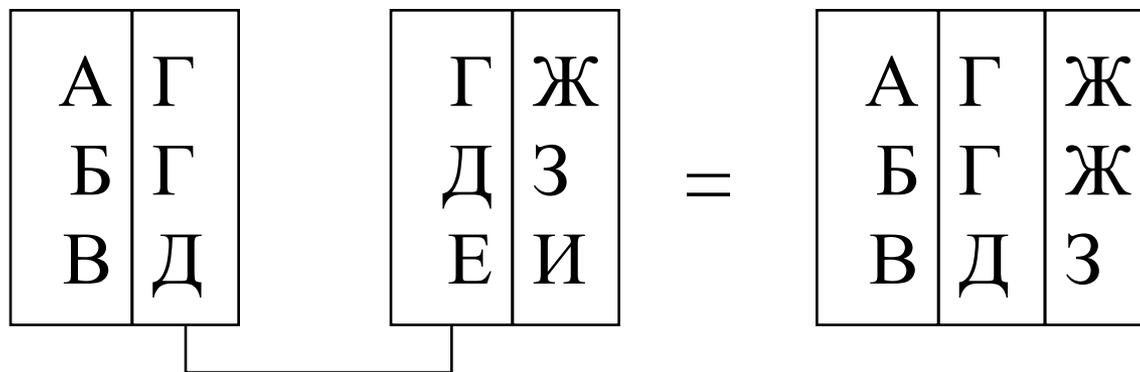


6. Проекция

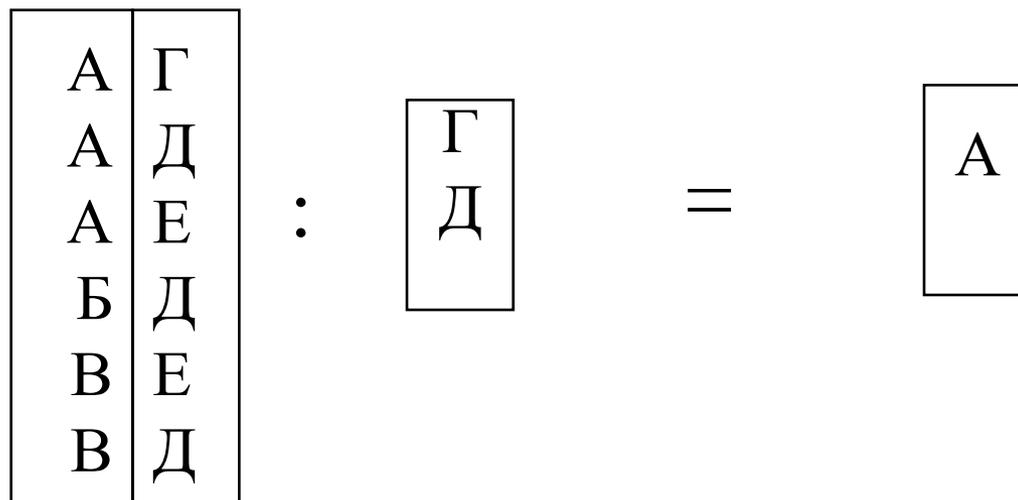


1. Поясняющие диаграммы (7-8):

7. Условное  
соединение



8. Деление



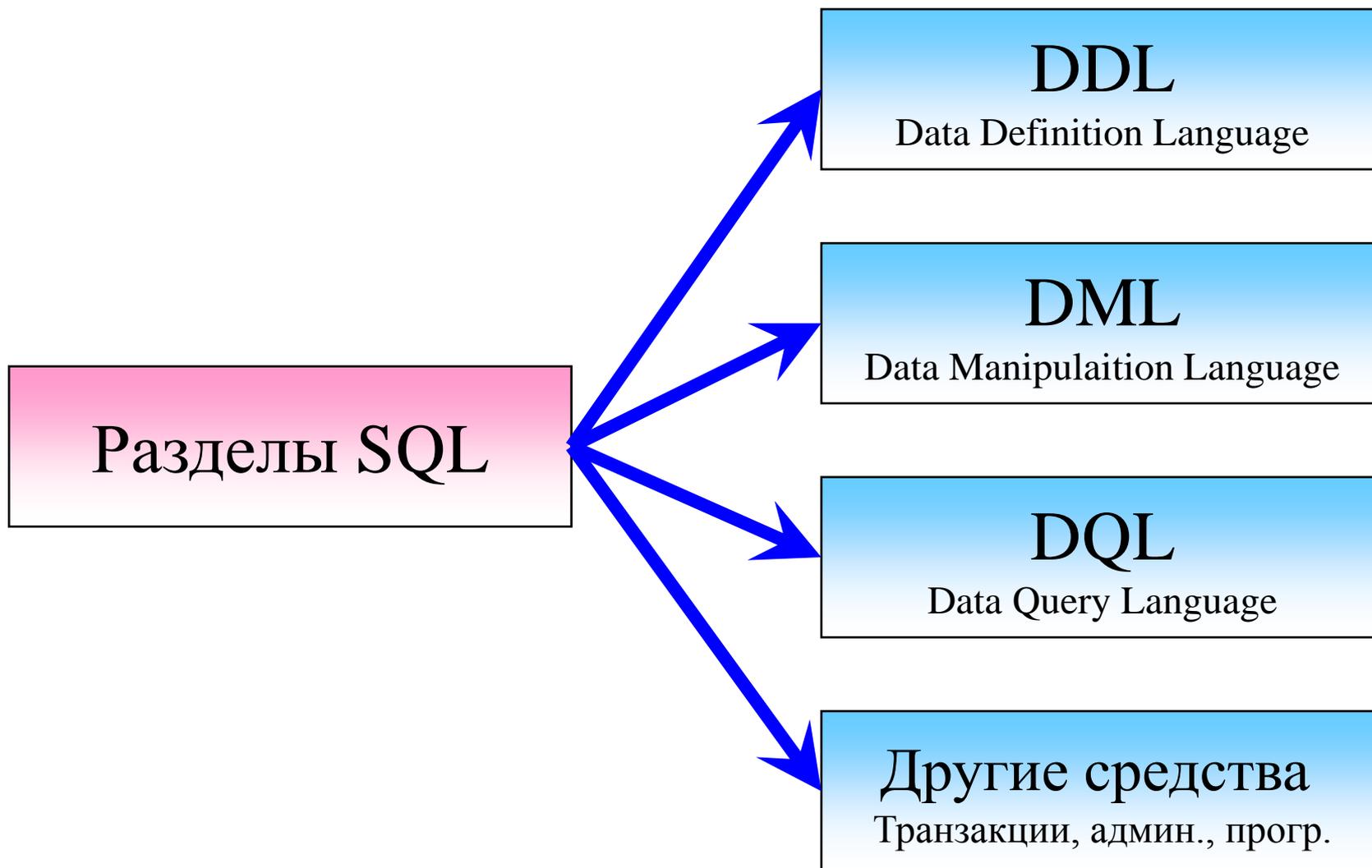
# § 9. Язык SQL

---

А.Е. АНИСИМОВ

# SQL

- **Structured Query Language** – Язык Структурированных Запросов - универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных.



## •Data Definition Language

- язык описания (определения) данных

Оператор	Действие
CREATE TABLE	Создать таблицу
DROP TABLE	Удалить таблицу
ALTER TABLE	Изменить таблицу
CREATE VIEW	Создать представление
DROP VIEW	Удалить представление
ALTER VIEW	Изменить представление
CREATE INDEX	Создать индекс
DROP INDEX	Удалить индекс

- **Таблица (TABLE)** – основное информационное хранилище базы данных в реляционной модели;
- **Представление (VIEW)** - виртуальная (логическая) таблица, представляющая собой результат поименованного запроса.

В отличие от обычных таблиц реляционной БД, представление не является самостоятельной частью набора данных, хранящегося в базе. Содержимое представления динамически вычисляется на основании данных, находящихся в реальных таблицах;

- **Индекс (INDEX)** - объект базы данных, создаваемый с целью повышения производительности поиска данных.

## •Data Manipulation Language

- язык манипулирования данными

Оператор	Действие
DELETE	Удаление записей из таблицы в соответствии с условиями запроса
INSERT	Вставляет записи в таблицу из другой таблицы или запроса
UPDATE	Обновляет значения одного или нескольких полей в записях, соответствующих условиям запроса

- Data Query Language

- - язык запросов

Оператор	Действие
SELECT	Формирует новое отношение (результат запроса) в соответствии с условиями запроса

## Оператор SELECT

Этот оператор реализует все операции реляционной алгебры. Синтаксис SELECT:

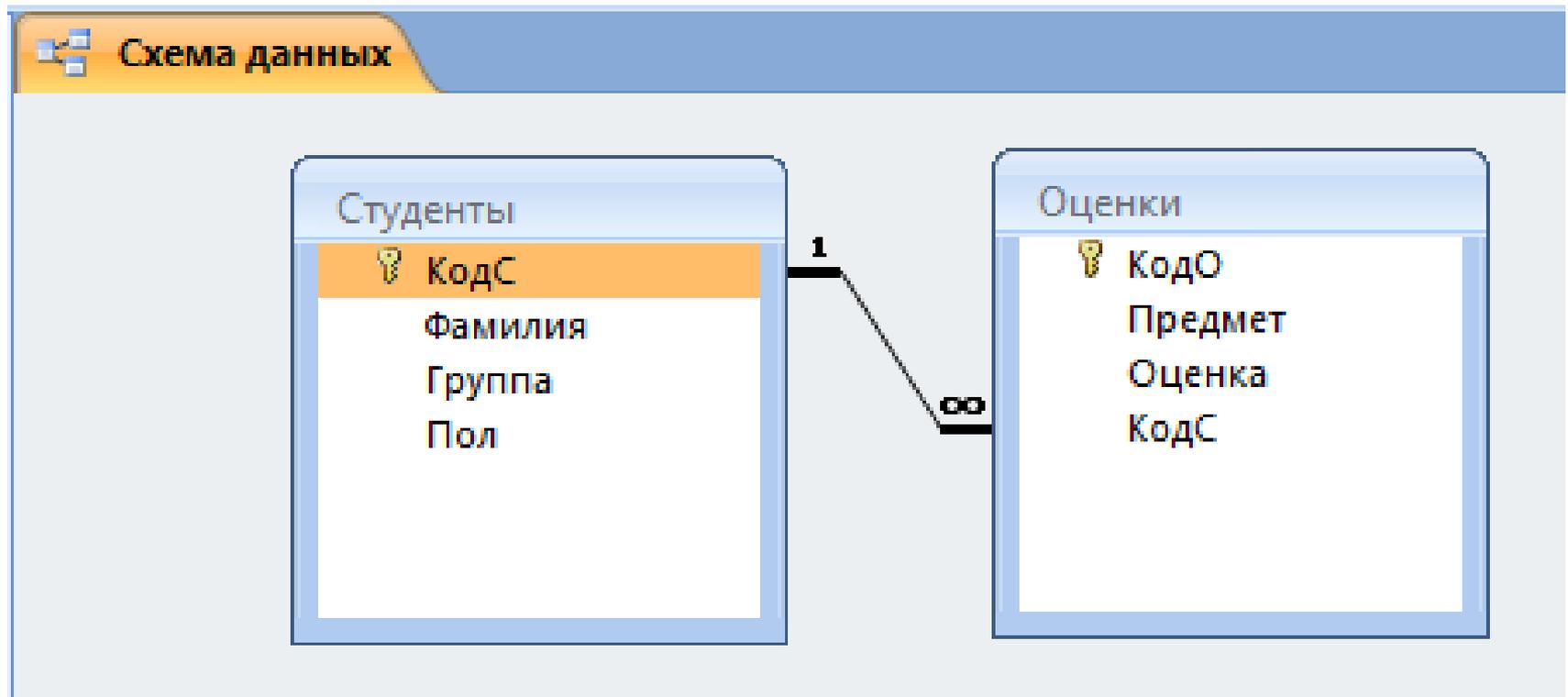
<b>SELECT</b> (ALL   DISTINCT)(<список_полей> *)	проекция
<b>FROM</b> <список_таблиц>	декартово произведение
<b>WHERE</b> <условие_выборки/соединения>	фильтрация/ условное соединение
<b>GROUP BY</b> <поля_группировки>	группировка
<b>HAVING</b> <условие_на_группу>	фильтрация групп
<b>ORDER BY</b> <поля_сортировки>	сортировка

- Пояснения:
  1. **ALL** – в результат включаются *все* строки запроса (в том числе одинаковые; по умолчанию); **DISTINCT** – в результат включаются только *различные* строки.
  2. \* означает выбор всех полей таблиц; иначе – только тех, которые указаны в разделе **SELECT**.
  3. В разделе **FROM** задается перечень исходных таблиц, над которыми выполняется декартово произведение.
  4. В разделе **WHERE** задаются условия запроса: либо условие выборки, либо условие соединения.

- Пояснения:
- 5. В разделе **GROUP BY** задается список полей, по значениям которых выполняется группировка записей.
- 6. В разделе **HAVING** задаются условия, накладываемые на каждую группу записей.
- 7. В разделе **ORDER BY** перечисляются поля, определяющие порядок сортировки записей в результате.
- 8. Имена полей – составные:  
**ИмяТаблицы.ИмяПоля**
- 9. В разделе **SELECT** можно задавать псевдонимы полей (**AS**); также **AS** можно использовать в разделе **FROM** для задания псевдонимов таблиц.

- Условия (предикаты) в разделе WHERE:
- Сравнения =, <>, <, <=, >, >=
- **Between A and B** – значение между A и B
- **Not Between A and B** – значение не между A и B
- **In (множество)** – принадлежность множеству
- **Not In (множество)** – непринадлежность множеству
- **Like “шаблон”** – сравнение с шаблоном (образцом):
  - \_ - любой одиночный символ
  - % - любая последовательность символов
  - другие символы обозначают сами себя
- **Is Null** – сравнение с неопределенным значением
- **Not Is Null** – отрицание неопределенного значения

- Пример. Создание запросов к базе данных на SQL
- Схема данных:



- Таблицы:

Студенты			
КодС	Фамилия	Группа	Пол
1	Иванов	801	М
2	Петров	801	М
3	Сидорова	801	Ж
4	Кузнецов	803	М
5	Малышева	803	Ж

Оценки			
КодО	Предмет	Оценка	КодС
1	Информатика	4	1
2	Информатика	3	2
3	Алгебра	5	2
4	Алгебра	4	5
5	Алгебра	3	4
6	Информатика	4	4
7	Алгебра	2	3
8	Информатика	3	5

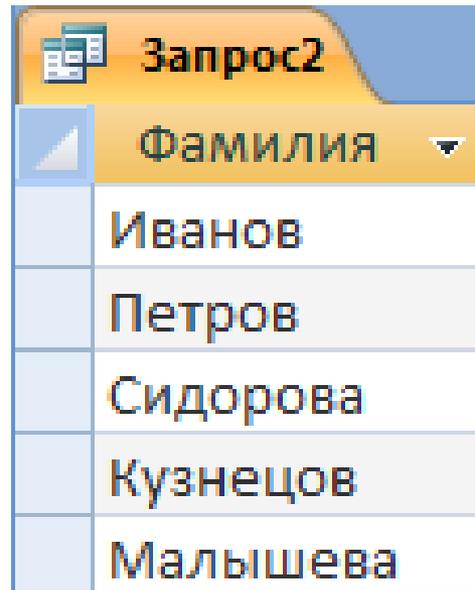
01. Список всех студентов (копия таблицы Студенты)

```
SELECT *  
FROM Студенты;
```

Запрос1				
	КодС	Фамилия	Группа	Пол
	1	Иванов	801	М
	2	Петров	801	М
	3	Сидорова	801	Ж
	4	Кузнецов	803	М
	5	Малышева	803	Ж

02. Фамилии  
студентов

```
SELECT Студенты.Фамилия  
FROM Студенты;
```

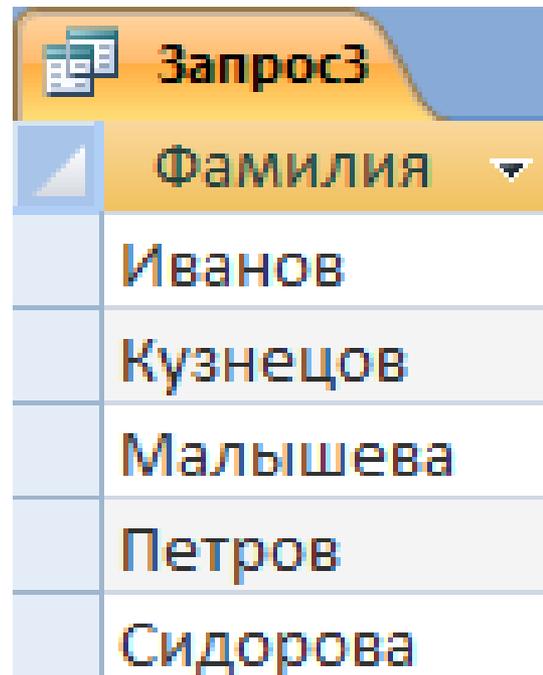


The screenshot shows a window titled "Запрос2" (Query2) displaying the results of an SQL query. The results are presented in a table with a single column labeled "Фамилия" (Surname). The surnames listed are: Иванов, Петров, Сидорова, Кузнецов, and Малышева.

Фамилия
Иванов
Петров
Сидорова
Кузнецов
Малышева

03. Список студентов в алфавитном порядке

```
SELECT Студенты.Фамилия  
FROM Студенты  
ORDER BY Студенты.Фамилия;
```

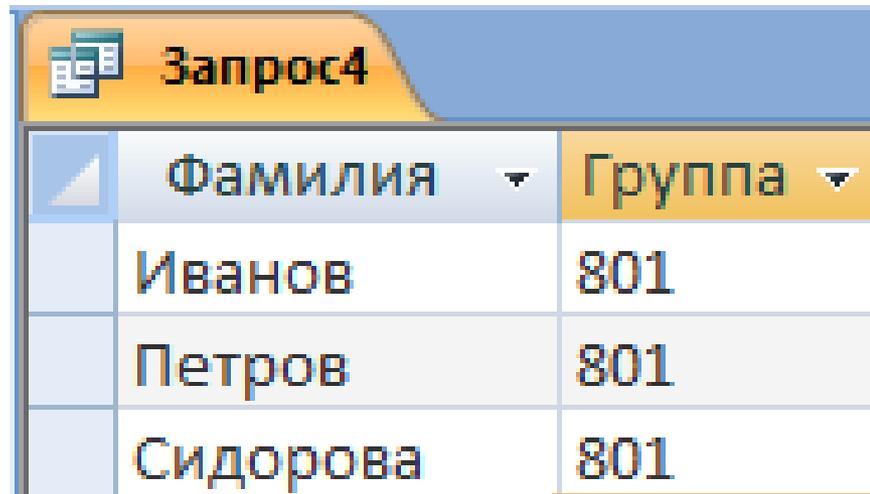


The screenshot shows a query window titled "Запрос3" (Query3). The column header is "Фамилия" (Surname). The results are listed in alphabetical order: Иванов, Кузнецов, Малышева, Петров, and Сидорова.

Фамилия
Иванов
Кузнецов
Малышева
Петров
Сидорова

#### 04. Список студентов группы 801

```
SELECT Студенты.Фамилия, Студенты.Группа  
FROM Студенты  
WHERE Студенты.Группа="801";
```



The screenshot shows a query window titled 'Запрос4' (Query4). The query results are displayed in a table with two columns: 'Фамилия' (Surname) and 'Группа' (Group). The results show three rows of data, all with the group number 801.

Фамилия	Группа
Иванов	801
Петров	801
Сидорова	801

## 05. Список оценок

```
SELECT Оценки.Предмет, Оценки.Оценка,  
Оценки.КодС  
FROM Оценки  
ORDER BY Оценки.Предмет;
```

Предмет	Оценка	КодС
Алгебра	2	3
Алгебра	3	4
Алгебра	4	5
Алгебра	5	2
Информатика	3	5
Информатика	4	4
Информатика	3	2
Информатика	4	1

Хотелось бы  
видеть фамилии!  
Но они в таблице  
Студенты

## 06. Список студентов и их оценок

```
SELECT *
FROM Студенты , Оценки ;    !!!
```

КодО	Предмет	Оценка	Оценки.Код	Студе	Фамилия	Группа	Пол
1	Информатика	4	1	1	Иванов	801	М
1	Информатика	4	1	2	Петров	801	М
1	Информатика	4	1	3	Сидорова	801	Ж
1	Информатика	4	1	4	Кузнецов	803	М
1	Информатика	4	1	5	Мальшева	803	Ж
2	Информатика	3	2	1	Иванов	801	М
2	Информатика	3	2	2	Петров	801	М
2	Информатика	3	2	3	Сидорова	801	Ж
2	Информатика	3	2	4	Кузнецов	803	М
2	Информатика	3	2	5	Мальшева	803	Ж
3	Алгебра	5	2	1	Иванов	801	М
3	Алгебра	5	2	2	Петров	801	М
3	Алгебра	5	2	3	Сидорова	801	Ж
3	Алгебра	5	2	4	Кузнецов	803	М
3	Алгебра	5	2	5	Мальшева	803	Ж
4	Алгебра	4	5	1	Иванов	801	М
4	Алгебра	4	5	2	Петров	801	М
4	Алгебра	4	5	3	Сидорова	801	Ж
4	Алгебра	4	5	4	Кузнецов	803	М
4	Алгебра	4	5	5	Мальшева	803	Ж
5	Алгебра	3	4	1	Иванов	801	М
5	Алгебра	3	4	2	Петров	801	М
5	Алгебра	3	4	3	Сидорова	801	Ж
5	Алгебра	3	4	4	Кузнецов	803	М
5	Алгебра	3	4	5	Мальшева	803	Ж
6	Информатика	4	4	1	Иванов	801	М
6	Информатика	4	4	2	Петров	801	М
6	Информатика	4	4	3	Сидорова	801	Ж

Не вышло, т.к. не было указано, как соединяются записи двух таблиц (каждая с каждой)

## 07. Список студентов и их оценок (еще попытка)

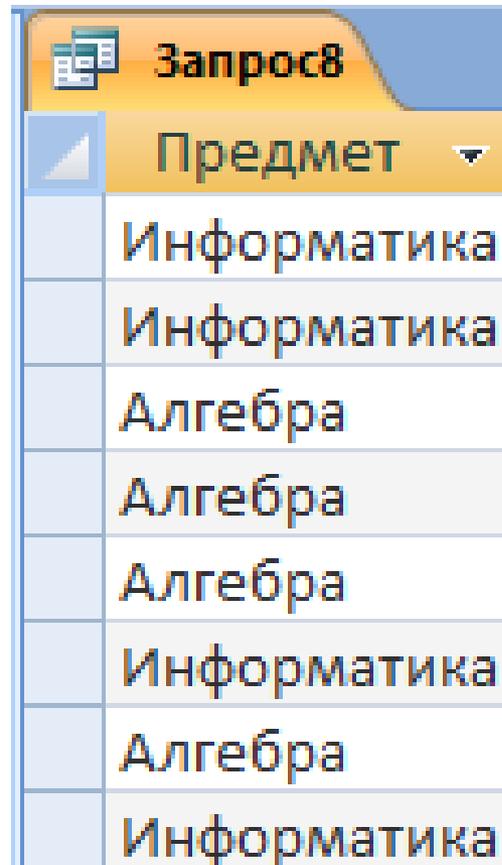
```
SELECT Студенты.Фамилия, Оценки.Предмет, Оценки.Оценка  
FROM Студенты,Оценки  
WHERE Студенты.КодС = Оценки.КодС;
```

Фамилия	Предмет	Оценка
Иванов	Информатика	4
Петров	Информатика	3
Петров	Алгебра	5
Сидорова	Алгебра	2
Кузнецов	Алгебра	3
Кузнецов	Информатика	4
Малышева	Алгебра	4
Малышева	Информатика	3

Теперь  
правильно

## 08. Список предметов

```
SELECT Оценки.Предмет  
FROM Оценки;
```



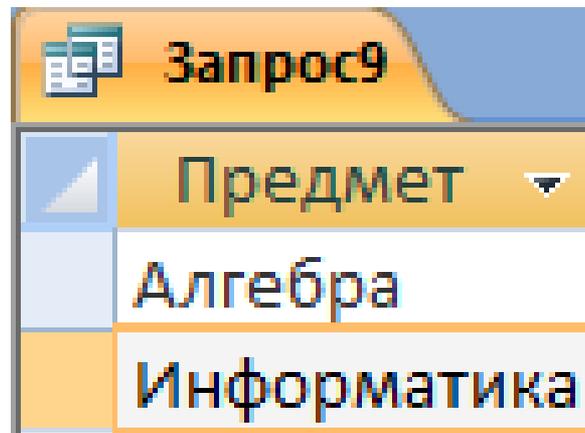
The screenshot shows a query window titled "Запрос8" (Query8) displaying the results of the SQL query. The results are shown in a table with a single column labeled "Предмет" (Subject). The table contains the following rows:

Предмет
Информатика
Информатика
Алгебра
Алгебра
Алгебра
Информатика
Алгебра
Информатика

Но здесь  
есть  
повторы!

## 09. Список предметов (без повторов)

```
SELECT DISTINCT Оценки.Предмет  
FROM Оценки;
```



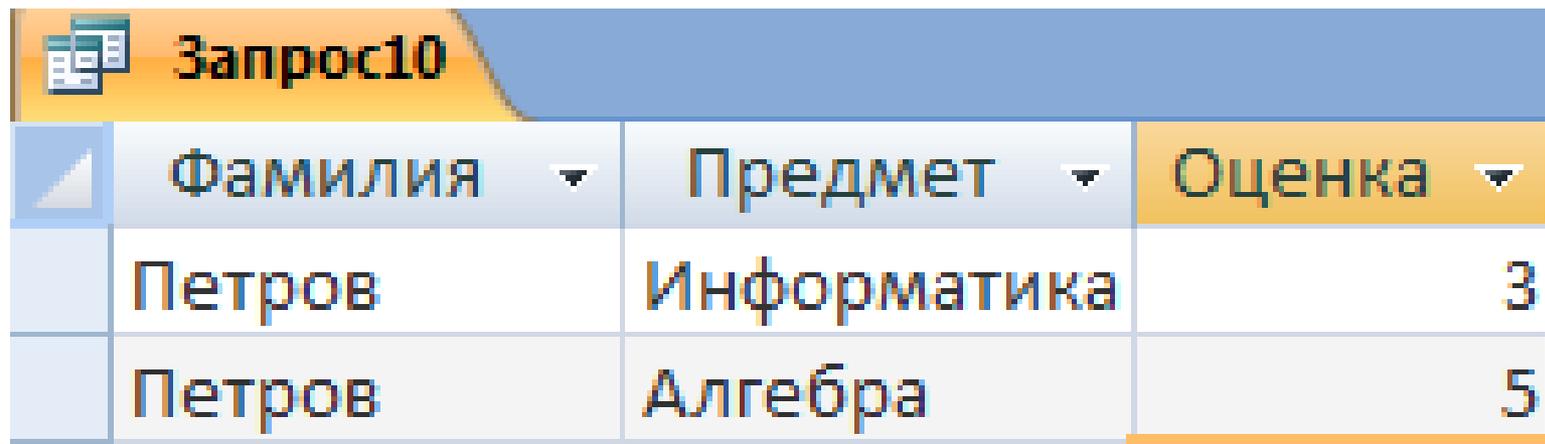
The screenshot shows a query window titled "Запрос9" (Query9). The result is a table with one column named "Предмет" (Subject). The table contains two rows: "Алгебра" (Algebra) and "Информатика" (Informatics). The "Информатика" row is highlighted with an orange background.

Предмет
Алгебра
Информатика

Теперь без  
повторов

## 10. Все оценки студента Петрова

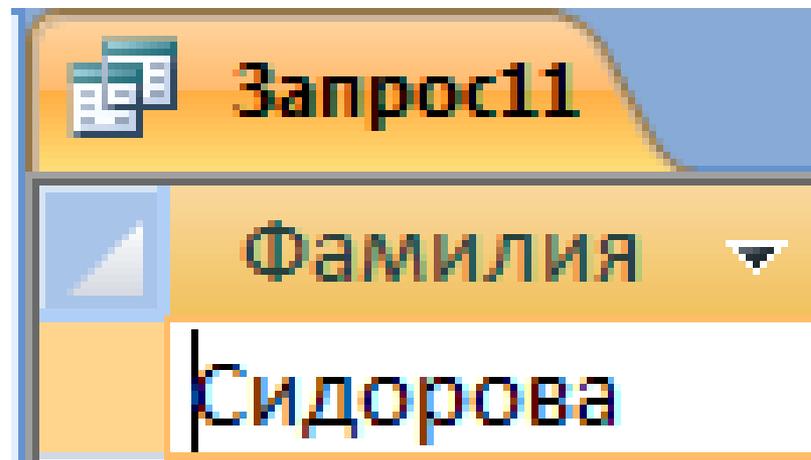
```
SELECT Студенты.Фамилия, Оценки.Предмет, Оценки.Оценка  
FROM Студенты, Оценки  
WHERE (Студенты.КодС = Оценки.КодС)  
AND (Студенты.Фамилия="Петров");
```



Фамилия	Предмет	Оценка
Петров	Информатика	3
Петров	Алгебра	5

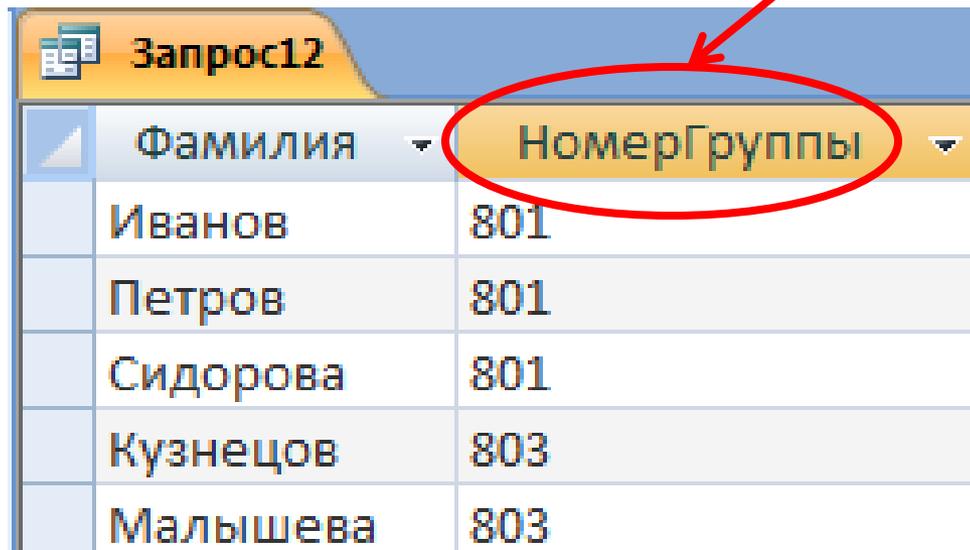
11. Список студентов, у которых есть хотя бы одна двойка

```
SELECT DISTINCT Студенты.Фамилия  
FROM Студенты , Оценки  
WHERE (Студенты.КодС = Оценки.КодС)  
AND (Оценки.Оценка=2);
```



## 12. Переименование столбца

```
SELECT Студенты.Фамилия, Студенты.Группа AS НомерГруппы  
FROM Студенты;
```



Фамилия	НомерГруппы
Иванов	801
Петров	801
Сидорова	801
Кузнецов	803
Малышева	803

## •4. Группировка и агрегатные функции

•**Группировка** – разбиение всего множества записей таблицы на группы, в которые собираются записи, имеющие одинаковые значения полей группировки.

Студенты				
КодС	Фамилия	Группа	Пол	
1	Иванов	801	М	
2	Петров	801	М	
3	Сидорова	801	Ж	
4	Кузнецов	803	М	
5	Малышева	803	Ж	

Если группировать записи по полю «Пол», то они будут разбиты на две группы записей («М» и «Ж»).

Если группировать записи по полю «Группа», то они будут разбиты на две группы записей (801 и 803).

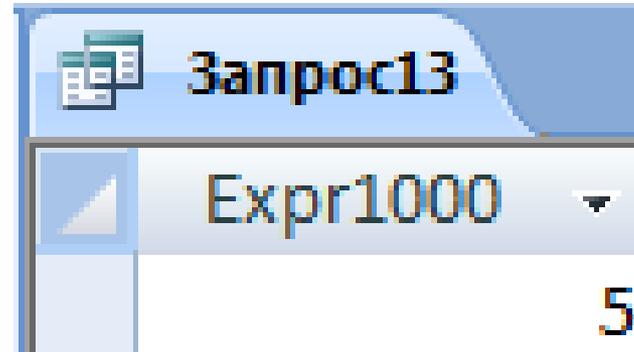
- **Агрегатная функция** – вычисляет обобщенное групповое значение для всех записей каждой группы.

ФУНКЦИЯ	РЕЗУЛЬТАТ
<b>COUNT()</b>	Количество записей группы / непустых значений поля
<b>SUM()</b>	Сумма
<b>AVG()</b>	Среднее арифметическое
<b>MIN()</b>	Наименьшее
<b>MAX()</b>	Наибольшее

- Агрегатные функции используются подобно именам полей в `SELECT`
- Аргументами агрегатных функций являются имена полей (либо \* для `COUNT`)
- При использовании группировки в разделе `SELECT` можно использовать **только** имена полей группировки либо агрегатные функции.
- Можно применять агрегатные функции и без группировки, тогда вся таблица рассматривается как одна группа.

### 13. Общее количество студентов

```
SELECT COUNT(*)  
FROM Студенты;
```

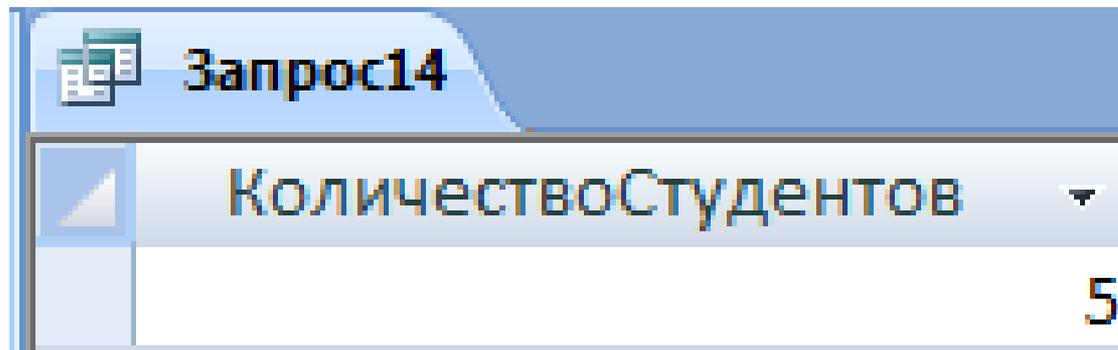


The screenshot shows a window titled "Запрос13" (Query13) with a table icon. The table has one column labeled "Expr1000" and one row containing the value "5".

Expr1000
5

14. Общее количество студентов (заголовок столбца)

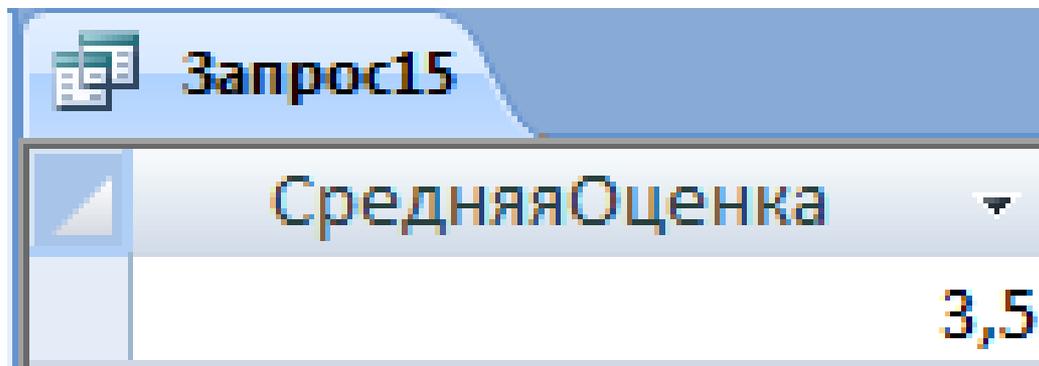
```
SELECT COUNT(*) AS КоличествоСтудентов  
FROM Студенты;
```



КоличествоСтудентов
5

15. Средняя оценка всех студентов

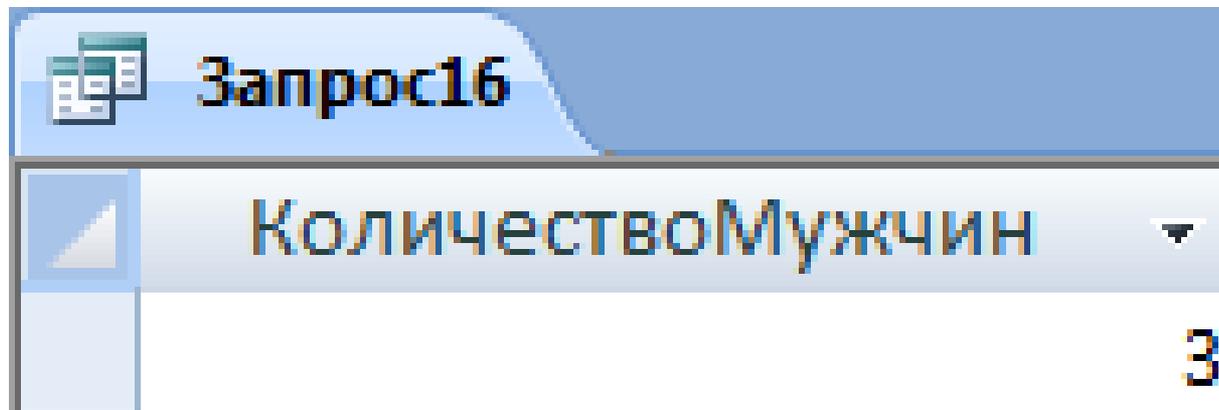
```
SELECT AVG(Оценка) AS СредняяОценка  
FROM Оценки;
```



СредняяОценка
3,5

## 16. Количество студентов-мужчин

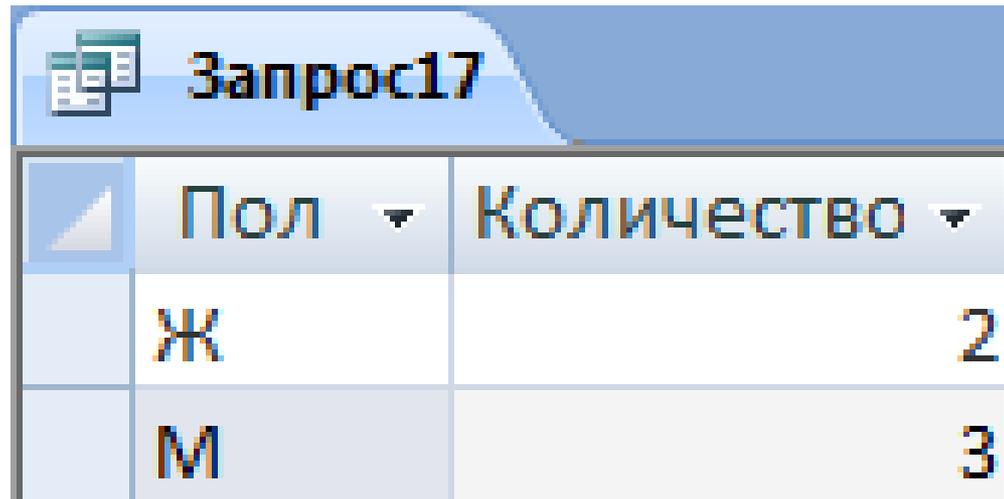
```
SELECT COUNT(*) AS КоличествоМужчин  
FROM Студенты  
WHERE Пол="М";
```



КоличествоМужчин
3

17. Количество студентов каждого пола

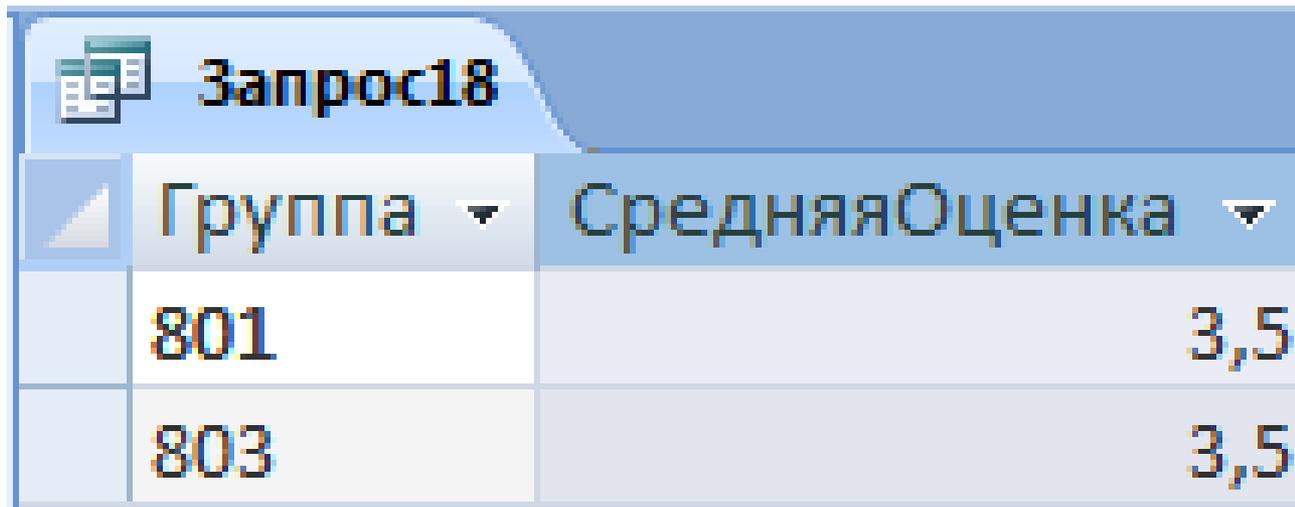
```
SELECT Пол, COUNT(*) AS Количество  
FROM Студенты  
GROUP BY Пол;
```



	Пол ▼	Количество ▼
	Ж	2
	М	3

## 18. Средняя оценка в каждой группе

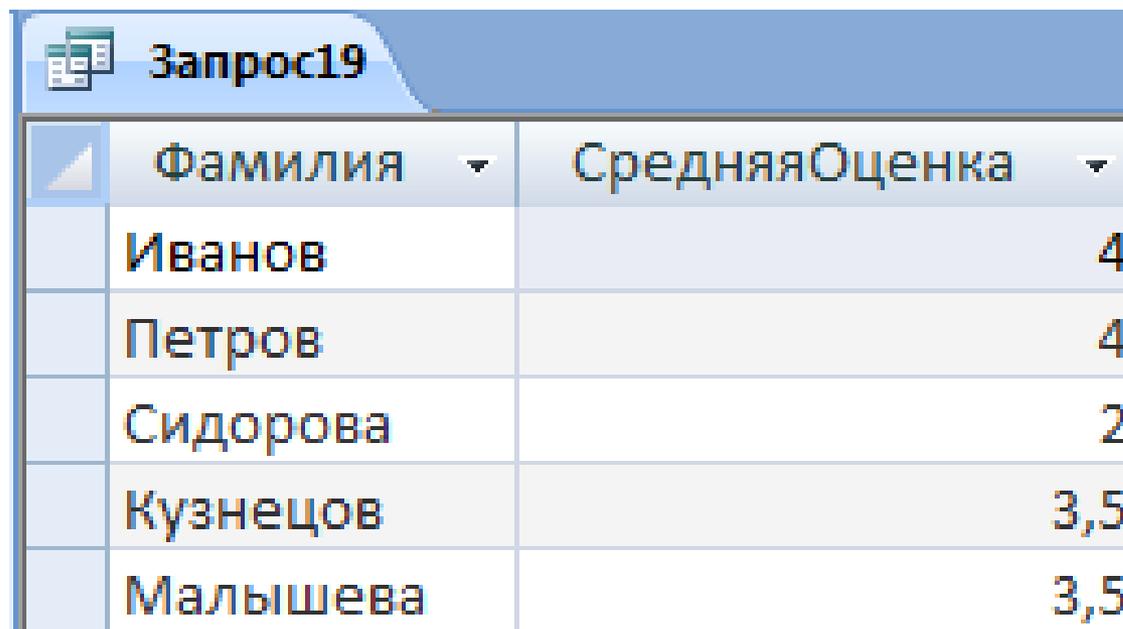
```
SELECT Студенты.Группа, Avg(Оценки.Оценка) AS  
СредняяОценка  
FROM Студенты, Оценки  
WHERE Студенты.КодС=Оценки.КодС  
GROUP BY Студенты.Группа;
```



Группа	СредняяОценка
801	3,5
803	3,5

## 19. Средняя оценка каждого студента

```
SELECT MAX(Студенты.Фамилия) AS Фамилия,  
       AVG(Оценки.Оценка) AS СредняяОценка  
FROM Студенты, Оценки  
WHERE Студенты.КодС = Оценки.КодС  
GROUP BY Студенты.КодС;
```



Фамилия	СредняяОценка
Иванов	4
Петров	4
Сидорова	2
Кузнецов	3,5
Малышева	3,5

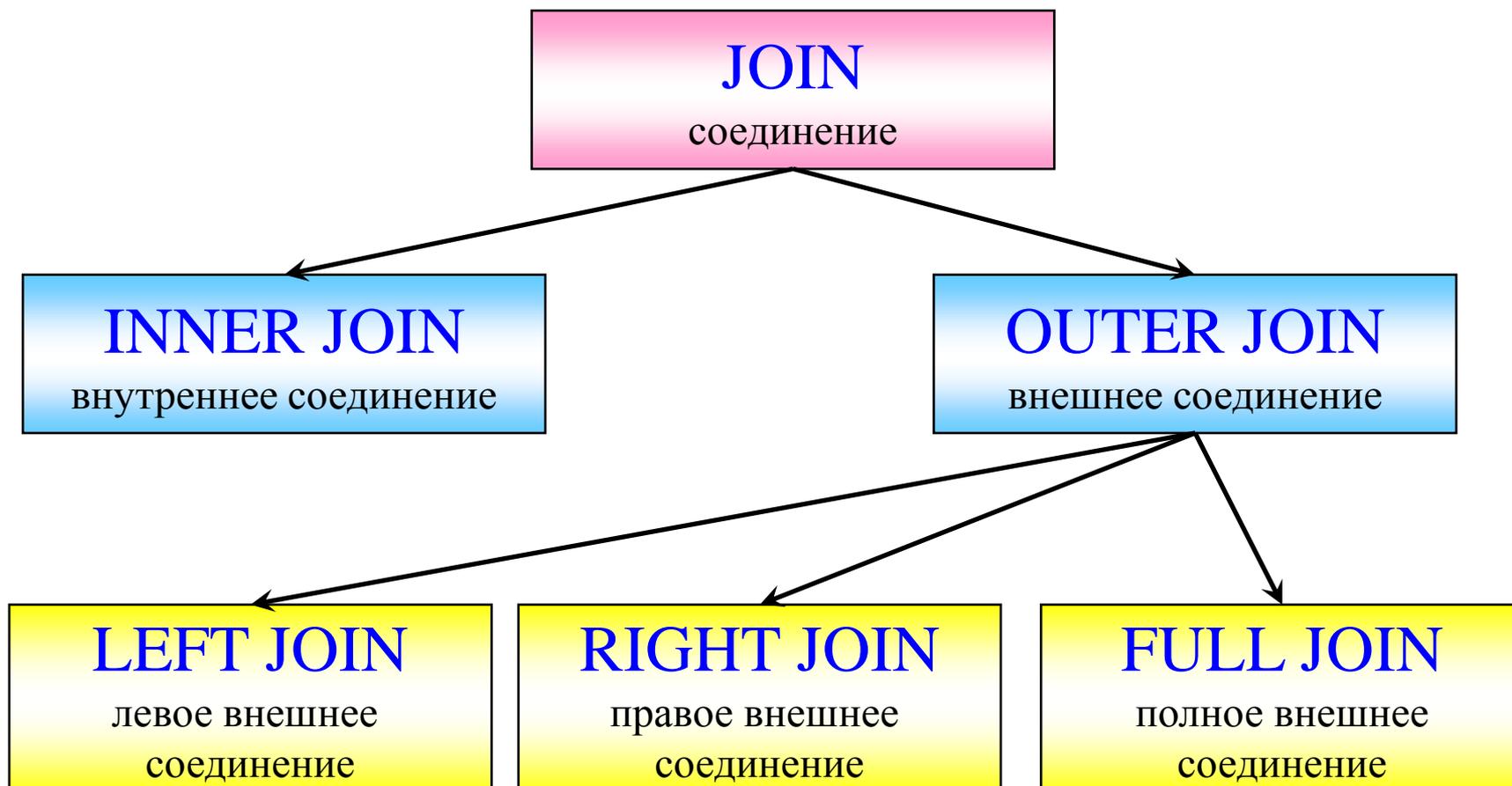
## •5. Внутреннее и внешнее соединение таблиц

- Соединение (**JOIN**) – формирование представления таблиц сцеплением их записей.

- Ранее были введены операции расширенного декартова произведения (каждый кортеж с каждым) и условного соединения (только те кортежи, для которых истинен предикат соединения).

- JOIN – специальная операция SQL, позволяющая указать способ соединения двух или более таблиц в разделе FROM.

## Виды соединений



## Синтаксис оператора соединения:

Таблица1 **JOIN** Таблица2 **ON** УсловиеСоединения

## • 1. Внутреннее соединение INNER JOIN

Соединяются только те записи из T1, для которых существуют записи из T2 и наоборот.

T1	
A1	B1
1	10
2	20
3	30

T2	
A2	B2
1	20
2	30
3	40

```
SELECT *  
FROM T1 INNER JOIN T2 ON T1.B1=T2.B2;
```

Q1			
A1	B1	A2	B2
2	20	1	20
3	30	2	30

## •2. Левое внешнее соединение LEFT JOIN

Соединяются все записи из T1 с соответствующими записями из T2. Отсутствующие записи заменяются на NULL.

```
SELECT *  
FROM T1 LEFT JOIN T2 ON T1.B1=T2.B2;
```

T1	
A1	B1
1	10
2	20
3	30

T2	
A2	B2
1	20
2	30
3	40

Q2			
A1	B1	A2	B2
1	10		
2	20	1	20
3	30	2	30

### •3. Правое внешнее соединение RIGHT JOIN

Соединяются все записи из T2 с соответствующими записями из T1. Отсутствующие записи заменяются на NULL.

```
SELECT *  
FROM T1 RIGHT JOIN T2 ON T1.B1=T2.B2;
```

T1	
A1	B1
1	10
2	20
3	30

T2	
A2	B2
1	20
2	30
3	40

Q3			
A1	B1	A2	B2
2	20	1	20
3	30	2	30
		3	40

#### •4. Полное внешнее соединение FULL JOIN

Соединяются все записи из T1 и все записи из T2.

Отсутствующие записи заменяются на NULL.

**SELECT \***  
**FROM T1 FULL JOIN T2 ON T1.B1=T2.B2;**

T1	
A1	B1
1	10
2	20
3	30

T2	
A2	B2
1	20
2	30
3	40

Q4			
A1	B1	A2	B2
1	10		
2	20	1	20
3	30	2	30
		3	40

Задание:

Написать запрос, находящий количество оценок у каждого студента

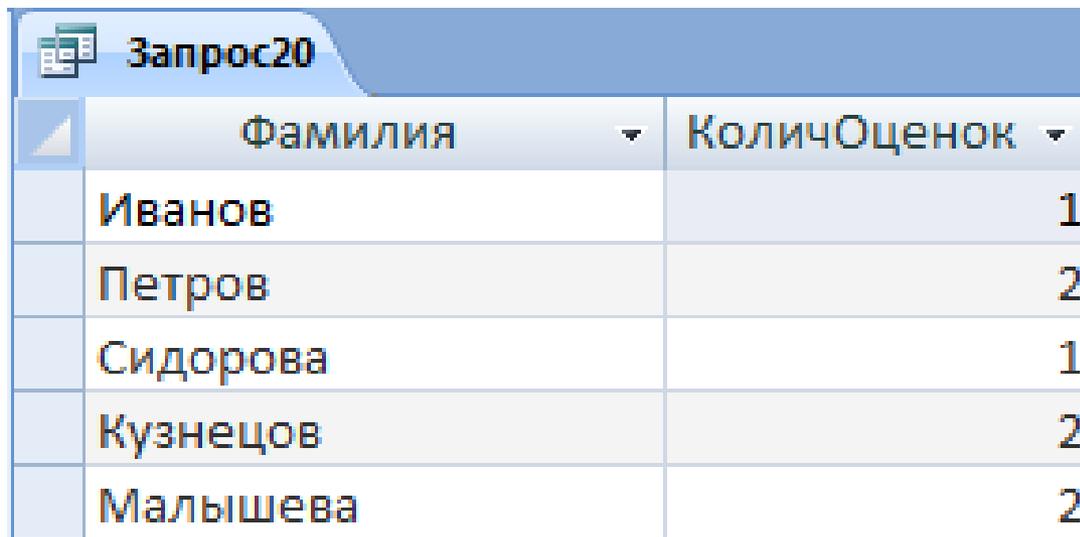
Добавим ещё одного студента, но без оценок:

Студенты					
		КодС	Фамилия	Группа	Пол
	+	1	Иванов	801	М
	+	2	Петров	801	М
	+	3	Сидорова	801	Ж
	+	4	Кузнецов	803	М
	+	5	Малышева	803	Ж
	+	6	Герман	803	М

В таблице «Оценки» для Германа ни одной записи не введено

## 20. Версия 1. Количество (без соединения)

```
SELECT MAX(Студенты.Фамилия) AS Фамилия,  
       COUNT(*) AS КоличОценок  
FROM Студенты, Оценки  
WHERE Студенты.КодС=Оценки.КодС  
GROUP BY Студенты.КодС;
```



Фамилия	КоличОценок
Иванов	1
Петров	2
Сидорова	1
Кузнецов	2
Малышева	2

А где  
Герман?  
Как быть?

## 21. Версия 2. Количество оценок (соединение)

```
SELECT MAX(Студенты.Фамилия) AS Фамилия,  
       COUNT(*) AS КоличОценок  
FROM Студенты INNER JOIN Оценки  
     ON Студенты.КодС=Оценки.КодС  
GROUP BY Студенты.КодС;
```

Запрос20	
Фамилия	КоличОценок
Иванов	1
Петров	2
Сидорова	1
Кузнецов	2
Малышева	2

А Германа  
все нет...

## 22. Версия 3. Количество оценок (левое соединение)

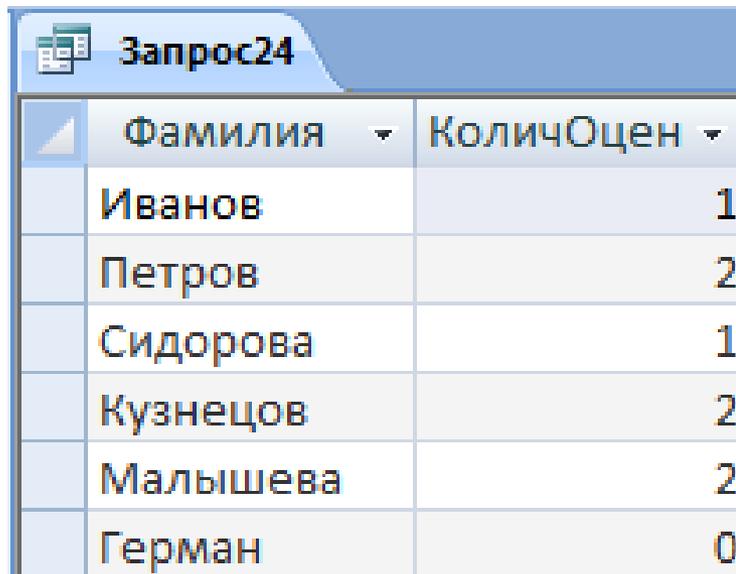
```
SELECT MAX(Студенты.Фамилия) AS Фамилия,  
       COUNT (*) AS КоличОценок  
FROM Студенты LEFT JOIN Оценки  
       ON Студенты.КодС = Оценки.КодС  
GROUP BY Студенты.КодС;
```

Запрос22	
Фамилия	КоличОцен
Иванов	1
Петров	2
Сидорова	1
Кузнецов	2
Малышева	2
Герман	1

А почему у  
Германа одна  
оценка? Ведь  
их нет!

## 23. Версия 4. Количество оценок (левое соединение)

```
SELECT MAX(Студенты.Фамилия) AS Фамилия,  
       COUNT (Оценка) AS КоличОценок  
FROM Студенты LEFT JOIN Оценки  
       ON Студенты.КодС = Оценки.КодС  
GROUP BY Студенты.КодС;
```



Фамилия	КоличОцен
Иванов	1
Петров	2
Сидорова	1
Кузнецов	2
Малышева	2
Герман	0

Теперь  
правильно

Отличие Count(\*)  
от Count(Поле)  
состоит в том, что  
вторая функция  
не учитывает  
нулевые значения.

## 6. Оператор INSERT

```
INSERT INTO <имя таблицы> [(  
<список полей>)]  
VALUES (<список констант>)
```

### Пакетная передача данных:

```
INSERT INTO <имя таблицы> [(  
<список полей>)]  
SELECT (<список полей>) FROM <имя таблицы2>
```

## •Пример

```
INSERT INTO Сотрудники (Фам, Детей)  
VALUES(“Сергеев”, 2);
```

```
INSERT INTO Сотрудники  
VALUES(“Сергеев”, 2);
```

```
INSERT INTO Студенты (Фамилия, ДатаРожд )  
SELECT Фамилия, ДатаРожд FROM Абитуриенты;
```

## •7. Оператор DELETE

DELETE FROM <имя таблицы>

[WHERE <условия поиска>]

## •Пример

```
DELETE FROM Сотрудники;
```

```
DELETE FROM Сотрудники
```

```
WHERE ((Возраст>=60) AND (Пол="М")  
OR (Возраст>=55) AND (Пол="Ж"));
```

## •8. Оператор UPDATE

UPDATE < имя таблицы >

SET < имя колонки > = < новое значение > , < имя колонки > =  
< новое значение > , ...

[WHERE < условия поиска >]

## •Пример

```
UPDATE Сотрудники SET Зарплата = Зарплата+1000;
```

```
UPDATE Сотрудники  
SET Зарплата = Зарплата-1000  
WHERE Опоздания > 0;
```

## •9. Оператор CREATE DATABASE

```
CREATE DATABASE [IF NOT EXISTS] <имя_бд>
```

### Пример

```
CREATE DATABASE students;
```

```
CREATE DATABASE IF NOT EXISTS teachers;
```

## •10. Оператор DROP DATABASE

DROP DATABASE [IF EXISTS] <имя\_бд>

### Пример

DROP DATABASE students;

DROP DATABASE IF EXISTS teachers;

# • 11. Оператор CREATE TABLE

**CREATE** [TEMPORARY] **TABLE** [IF NOT EXISTS]  
<имя\_таб> [(create\_definition,...)] [table\_options]  
[select\_statement]

**create\_definition:**

col\_name type [NOT NULL | NULL] [DEFAULT  
default\_value] [AUTO\_INCREMENT] [PRIMARY KEY]  
[reference\_definition]

**ИЛИ** PRIMARY KEY (index\_col\_name,...)

**ИЛИ** KEY [index\_name] (index\_col\_name,...)

**ИЛИ** INDEX [index\_name] (index\_col\_name,...)

**ИЛИ** UNIQUE [INDEX] [index\_name] (index\_col\_name,...)

**ИЛИ** FULLTEXT [INDEX] [index\_name] (index\_col\_name,...)

**ИЛИ** [CONSTRAINT symbol] FOREIGN KEY [index\_name]  
(index\_col\_name,...) [reference\_definition]

**ИЛИ** CHECK (expr)

## •12. Оператор DROP TABLE

```
DROP TABLE [IF EXISTS] <имя_таб> [, <имя_таб>,...]  
[RESTRICT | CASCADE]
```

### Пример

```
DROP TABLE users;
```

```
DROP TABLE IF EXISTS teachers, pupils CASCADE;
```

## •13. Оператор ALTER TABLE

ALTER [IGNORE] TABLE <имя\_таб> alter\_spec [, alter\_spec ...]

alter\_specification:

ADD [COLUMN] create\_definition [FIRST | AFTER column\_name ]

или ADD [COLUMN] (create\_definition, create\_definition,...)

или ADD INDEX [index\_name] (index\_col\_name,...)

или ADD PRIMARY KEY (index\_col\_name,...)

или ADD UNIQUE [index\_name] (index\_col\_name,...)

или ADD FULLTEXT [index\_name] (index\_col\_name,...)

или ADD [CONSTRAINT symbol] FOREIGN KEY index\_name  
(index\_col\_name,...) [reference\_definition]

или ALTER [COLUMN] col\_name {SET DEFAULT literal | DROP  
DEFAULT} или CHANGE [COLUMN] old\_col\_name create\_definition  
[FIRST | AFTER column\_name]

...

В SQL еще масса различных возможностей и средств.  
Рекомендуется изучать литературу и интернет.

<http://www.sql-ex.ru/>

<http://citforum.ru>

<http://www.sql.ru/>

<http://www.mysql.ru/docs>

и др.