SQL (*structured query language* — «язык структурированных запросов»)

Куликова Елена Васильевна

Содержание

1. ВВЕДЕНИЕ В SQL
1.1. Общее представление о SQL
1.2. Отличия SQL от языков программирования
1.3. Трехзначная логика. Основные операции в трехзначной логике
1.4. Реализации SQL
1.5. Основные понятия языка
1.6. Типы данных, доступные в SQL
1.7. Предикаты
2. ЗАПРОСЫ SQL
2.1. Оператор SELECT
2.2. Групповые операции в запросах
2.3. Запросы с несколькими таблицами (соединение таблиц)
2.4. Использование подзапросов (подчиненные запросы)
2.5. Запросы на модификацию данных
2.6. Перекрестные запросы
3. СОЗДАНИЕ БД В SQL

1. ВВЕДЕНИЕ В SQL
1.1. Общее представление о SQL

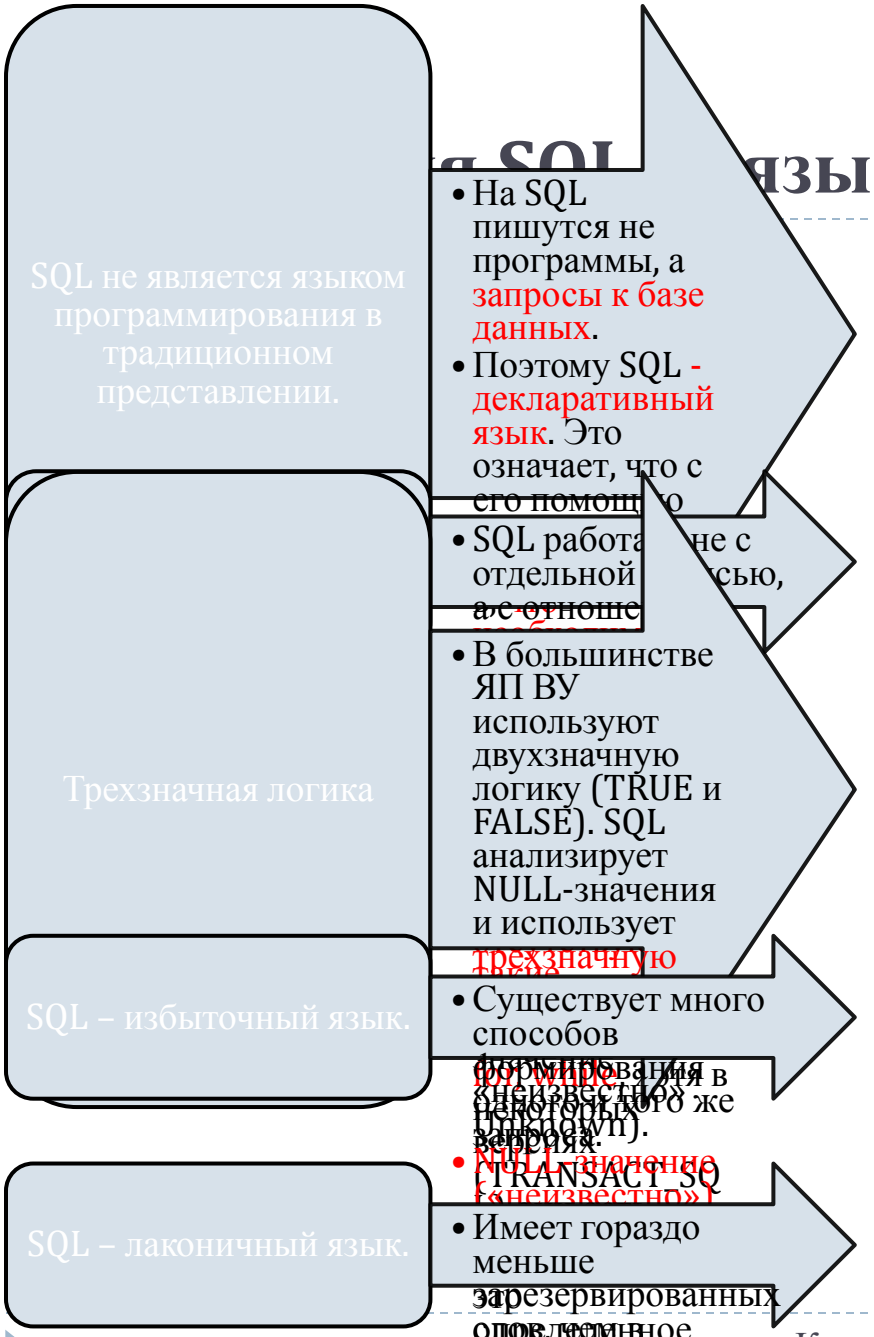
Понятие языка

- SQL (обычно произносимый как «СИКВЭЛ» или «ЭСКЮЭЛЬ») символизирует собой ***структурированный язык запросов***.
- Это язык, который дает возможность работать в реляционных базах данных.
- *Конкретные реализации языка SQL незначительно отличаются в различных SQL-серверах, однако базовые предложения остаются одинаковыми для всех реализаций.*

Официальный стандарт - ANSI/ISO

- Язык SQL имеет **официальный стандарт - ANSI/ISO** (ANSI - Американский национальный институт стандартов, **ISO** – Международный организация по стандартизации).

SQL и языков программирования



1.3. Трехзначная логика. Основные операции в трехзначной логике

- **Операция НЕ**

Операнд	NOT
True (t)	false
False (f)	true
Unknown (u)	Unknown

1.3. Трехзначная логика. Основные операции в трехзначной логике

- **Операция ИЛИ**

OR	true	false	Unknown
true	true	true	true
false	true	false	Unknown
Unknown	true	Unknown	Unknown

1.3. Трехзначная логика. Основные операции в трехзначной логике

- **Операция И**

AND	true	false	Unknown
true	true	false	Unknown
false	false	false	false
Unknown	Unknown	false	Unknown

1.3. Трехзначная логика. Основные операции в трехзначной логике

- **ЯВЛЯЕТСЯ ли (IS)**

IS	true	false	Unknown
true	true	false	false
false	false	true	false
Unknown	false	false	true

1.4. Реализации SQL

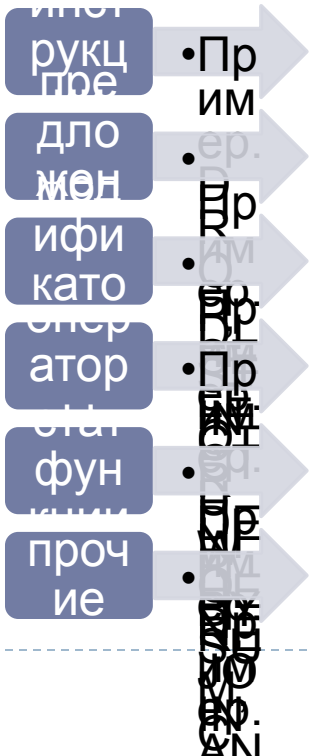
Интерактивный (автономный) SQL

Статический SQL

Динамический SQL

1.5. Основные понятия языка

- Оператор SQL состоит из набора лексем.
- **Лексемы** – это ключевые (зарезервированные) слова, идентификаторы, выражения, операторы.
- Все ключевые слова делят на группы:



Операторы языка разбиты на категории в соответствии с их функциями:

- 1
  - язык определения данных (Data Definition Language, **DDL**);
- 2
  - язык выполнения запросов (Data Query Language, **DQL**);
- 3
  - язык манипулирования данными (Data Manipulation Language, **DML**);
- 4
  - управление курсором (Cursor Control Language, **CCL**);
- 5
  - управление транзакциями (Transaction Processing Language, **TPL**);
- 6
  - язык управления данными (Data Control Language, **DCL**);
- 7
  - обеспечение целостности.

это не отдельные языки, а различные команды одного языка. Такое деление проведено только лишь с точки зрения различного функционального назначения этих команд.

Язык определения данных DDL

- Язык определения данных используется для создания и изменения структуры базы данных и ее составных частей – таблиц, индексов, представлений (виртуальных таблиц), а также триггеров и сохраненных процедур.
- Основными его командами являются:
  - **CREATE (создать),**
  - **ALTER (изменить),**
  - **DROP (удалить).**

Язык выполнения запросов DQL

- Язык выполнения запросов используется для различных выборок из данных и состоит из команды **SELECT (выбрать)**.

Язык манипулирования данными DML

- Язык манипулирования данными используется для выполнения действий над данными.
- Включает команды
  - **INSERT (вставить),**
  - **UPDATE (обновить),**
  - **DELETE (удалить)**.

Язык управления курсором CCL

- Язык управления курсором включает команды, оперирующие курсором.
- Под курсором понимают данные результатов запросов.

Язык управления транзакциями TPL

- Язык управления транзакциями объединяет команды DML в группы (транзакции).
- Состоит из команд:
  - COMMIT,
  - ROLLBACK,
  - SET TRANSACTION.

Язык управления данными DCL

- Язык управления данными используется для управления правами доступа к данным и выполнения процедур в многопользовательской среде. Более точно его можно назвать «язык управления доступом».
- Включает команды:
  - GRANT (права),
  - REVOKE (отмена).

1.6. Типы данных, доступные в SQL

Тип данных	Описание
	1. Строковые данные
1) CARACTER STRING ***2) NATIONAL CHARACTER***	Используются для хранения символьных строк. Второй тип используется для хранения национальных алфавитов. Оба типа имеют подтипы: •**CHAR** •**VARCHAR**. **CHAR(n)** – типы данных, предназначенные для хранения символьных строк фиксированной длины. Параметр “n” задает длину строки. Тип CHAR без параметра является одиночным символом и идентичен типу CHAR(1). **VARCHAR(m)** – тип данных, предназначенный для хранения символьной – строки переменной длины. Параметр “m” задает максимальную длину строки и является обязательным.
	2. Двоичные типы данных
3) BIT STRING	Тип данных, предназначенный для хранения двоичных объектов произвольного объема. Его необходимо использовать для хранения изображений, звука и т. д.

1.6. Типы данных, доступные в SQL

Тип данных	Описание
	3. Числовые данные
4) EXACT NUMERIC	Включает подтипы: •**INTEGER** (4-разрядные целые значения, поддерживаются числа от -2147483648 до 2147483647) •**SMALLINT** (короткое целое, 2-разрядные целые значения, поддерживаются числа от -32768 до 32767) •**DECIMAL**(m,t), **NUMERIC**(m,t) предназначены для хранения величин с фиксированным числом значащих цифр до и после запятой. Параметр “m” задает общее число десятичных цифр, а “t” – их количество после запятой.
5) APPROXIMATE NUMERIC	Включает подтипы **FLOAT**, **REAL**, **DOUBLE PRECISION**, которые предназначены для представления нецелых чисел. Внутреннее представление значений состоит из мантиссы и порядка. То есть значения этих типов данных могут лежать в очень широком диапазоне, но точными будут только несколько цифр. Объем памяти (число байтов), выделяемый для хранения значений, зависит от применяемого компьютера.

1.6. Типы данных, доступные в SQL

Тип данных	Описание
	4. Типы данных даты и времени
6) DATETIME	Включает подтипы: •**DATE**, значения даты •**TIME** значения времени •**DATETIME**, значения даты и времени Предназначены для хранения моментов времени. **Тип DATETIME содержит информацию о годе (YEAR), месяце (MONTH), дне (DAY), часе (HOUR), минуте (MINUTE), секунде (SECOND) и долях секунды (FRACTION).** Возможно выбирать нужный диапазон значений. Например, если требуется зафиксировать момент времени с точностью до секунды в течение дня, то следует указать тип DATETIME HOUR TO SECOND. Если же интересует информация о каком-либо событии с точностью до минуты, но в произвольном году, то тип данных должен быть описан как DATETIME YEAR TO MINUTE. При указании долей секунды надо отмечать точность представления.
7) INTERVAL	Тип данных, предназначенный для хранения временных интервалов. Его значение получают, например, путем вычитания одной даты из другой. Как и для DATETIME, здесь следует уточнить диапазон возможных значений

1.7. Предикаты

- В условиях отбора записей используют различные конструкции языка – предикаты:
  - Операторы сравнения (>, <, =, и т.д.).
  - Оператор BETWEEN AND – проверка на попадание в диапазон значений.
  - Оператор IN – проверка на вхождение элемента в отношение.
  - Оператор LIKE – проверка на приблизительное значение.
  - Оператор NULL – проверка на пустое значение.
  - Оператор SOME, ANY, ALL – проверка при сравнениях в подзапросах.
  - Оператор EXISTS – проверка на пустое отношение.
  - Оператор UNIQUE – проверка на неуникальность записей.
  - Оператор MATCH – проверка на неполное совпадение.
  - Оператор OVERLAP – проверка попадания временного интервала в заданное значение.

2. ЗАПРОСЫ SQL

- Запрос на языке SQL состоит из одного или нескольких операторов, следующих один за другим и разделенных точкой с запятой.

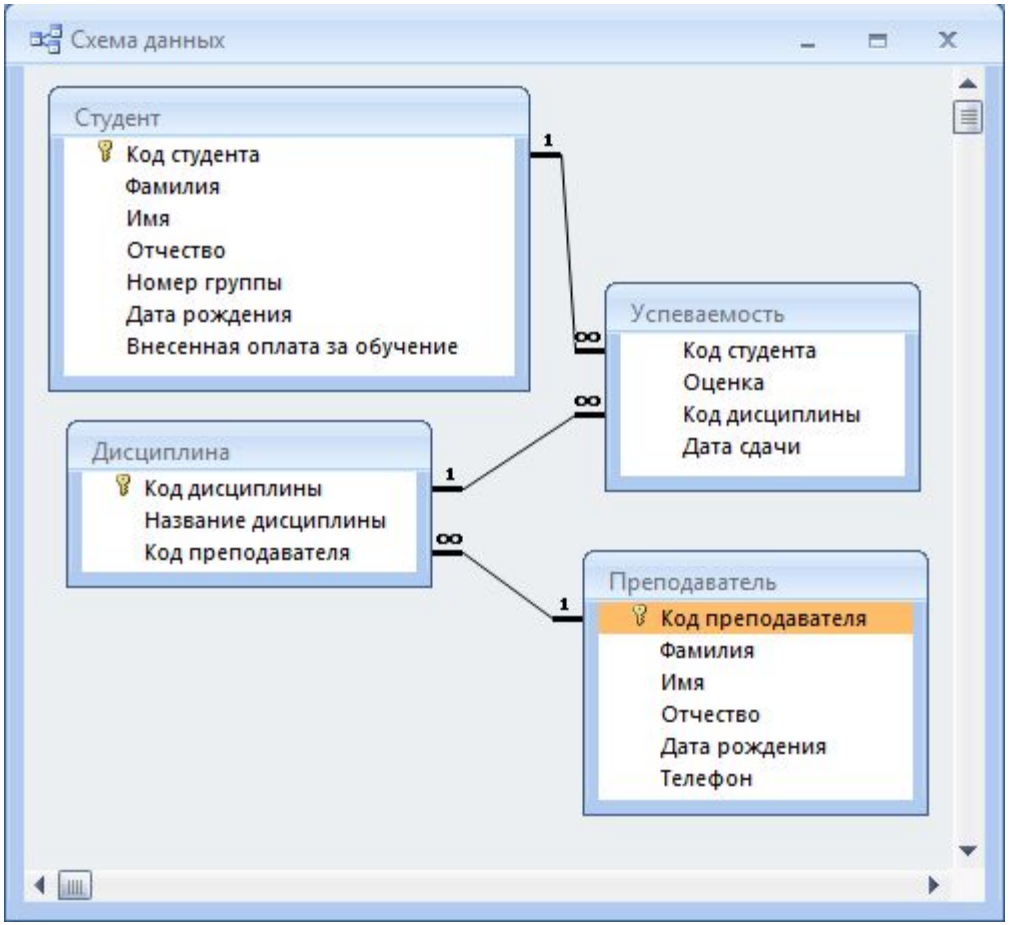
Операторы

- Наиболее важные операторы, которые входят в стандарт ANSI/ISO SQL:

Синтаксис оператора	Выполняемое действие
SELECT	Выбрать данные из базы данных
INSERT	Вставить данные в таблицу
DELETE	Удалить данные из таблицы
UPDATE	Изменить данные в таблице
GRANT	Передать права на действие над объектом
REVOKE	Отобрать права на действие над объектом
COMMIT	Подтвердить транзакцию
ROLLBACK	Откатить транзакцию
CREATE	Создать объект базы данных
DROP	Удалить объект базы данных

2.1. Оператор SELECT

База данных «Успеваемость» для рассмотрения примеров.



Имя поля	Тип данных
Код студента	Числовой
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
Номер группы	Текстовый
Дата рождения	Дата/время
Внесенная оплата за обучение	Денежный

Имя поля	Тип данных
Код преподавателя	Числовой
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
Дата рождения	Дата/время
Телефон	Текстовый

Имя поля	Тип данных
Код дисциплины	Числовой
Название дисциплины	Текстовый
Код преподавателя	Числовой

Имя поля	Тип данных
Код студента	Числовой
Оценка	Числовой
Код дисциплины	Числовой
Дата сдачи	Дата/время

База данных «Успеваемость» для рассмотрения примеров.

Преподаватель

Код преподавателя	Фамилия	Имя	Отчество	Дата рождения	Телефон
200	Волков	Семен	Игоревич	05.12.1975	8-913-8887766
201	Петров	Петр	Васильевич	24.12.1971	8-913-4447766
202	Иванова	Наталья	Владимировн	05.04.1977	8-913-3337766
203	Ковальчук	Сергей	Григорьевич	15.12.1965	8-913-9997766

Дисциплина

Код дисциплины	Название дисциплины	Код преподавателя
101	Математика	Волков
102	Информатика и программирование	
103	История	
104	Экономика	
105	Технология разработки и защиты баз данных	

Волков
Петров
Иванова
Ковальчук

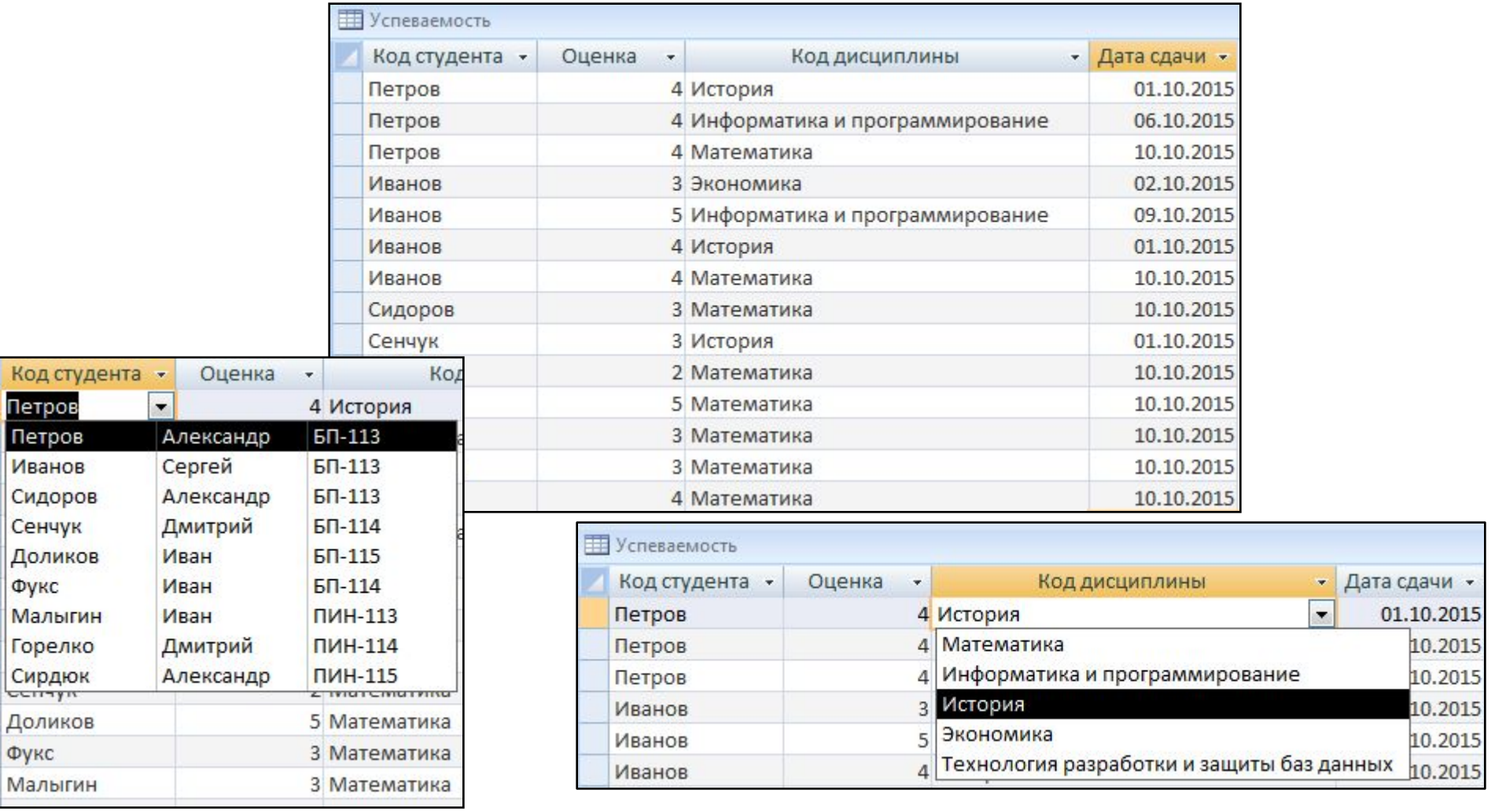
Студент

Код студента	Фамили	Имя	Отчество	Номер групп	Дата рожден	Внесенная оплата за обучение
300	Петров	Александр	Васильевич	БП-113	03.09.1997	5 000,00р
301	Иванов	Сергей	Витальевич		11.05.1998	5 000,00р
302	Сидоров	Александр	Георгиевич		23.09.1996	5 000,00р
303	Сенчук	Дмитрий	Иванович		03.12.1995	10 000,00р
304	Доликов	Иван	Иванович		13.09.1994	45 000,00р
305	Фукс	Иван	Иванович		13.09.1996	0,00р
306	Малыгин	Иван	Иванович		13.09.1994	130 000,00р
307	Горелко	Дмитрий	Иванович		03.12.1995	45 000,00р
308	Сирдюк	Александр	Васильевич		03.09.1990	45 000,00р

БП-113
БП-114
БП-115
КД-113
КД-114
ПИН-113
ПИН-114
ПИН-115

Запись: 1 из 9 | Нет фильтра | Поиск

База данных «Успеваемость» для рассмотрения примеров.



База данных «Успеваемость» (MS SQL Server)

205-11\KIS2.Успева...ость - dbo.Студент

Имя столбца	Тип данных	Разрешить ...
КодСтудента	int	☐
Фамилия	char(30)	☑
Имя	char(30)	☑
Отчество	char(30)	☑
НомерГруппы	char(10)	☑
ДатаРождения	datetime	☑
ВнесеннаяОплата	money	☑
		☐

205-11\KIS2.Успев...dbo.Преподаватель*

Имя столбца	Тип данных	Разрешить ...
КодПреподавателя	int	☐
Фамилия	char(30)	☑
Имя	char(30)	☑
Отчество	char(30)	☑
ДатаРождения	datetime	☑
Телефон	char(15)	☑

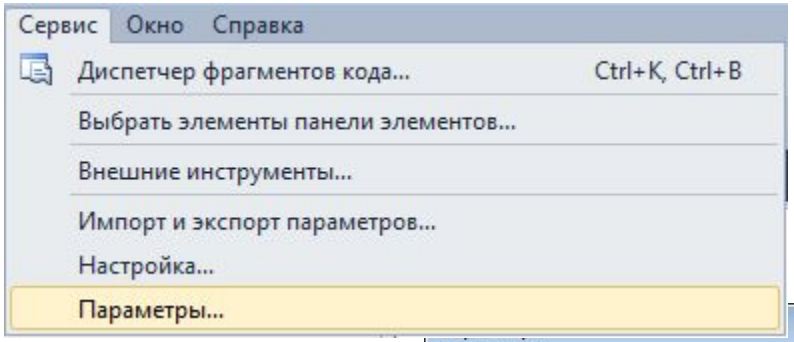
205-11\KIS2.Успев...- dbo.Дисциплина

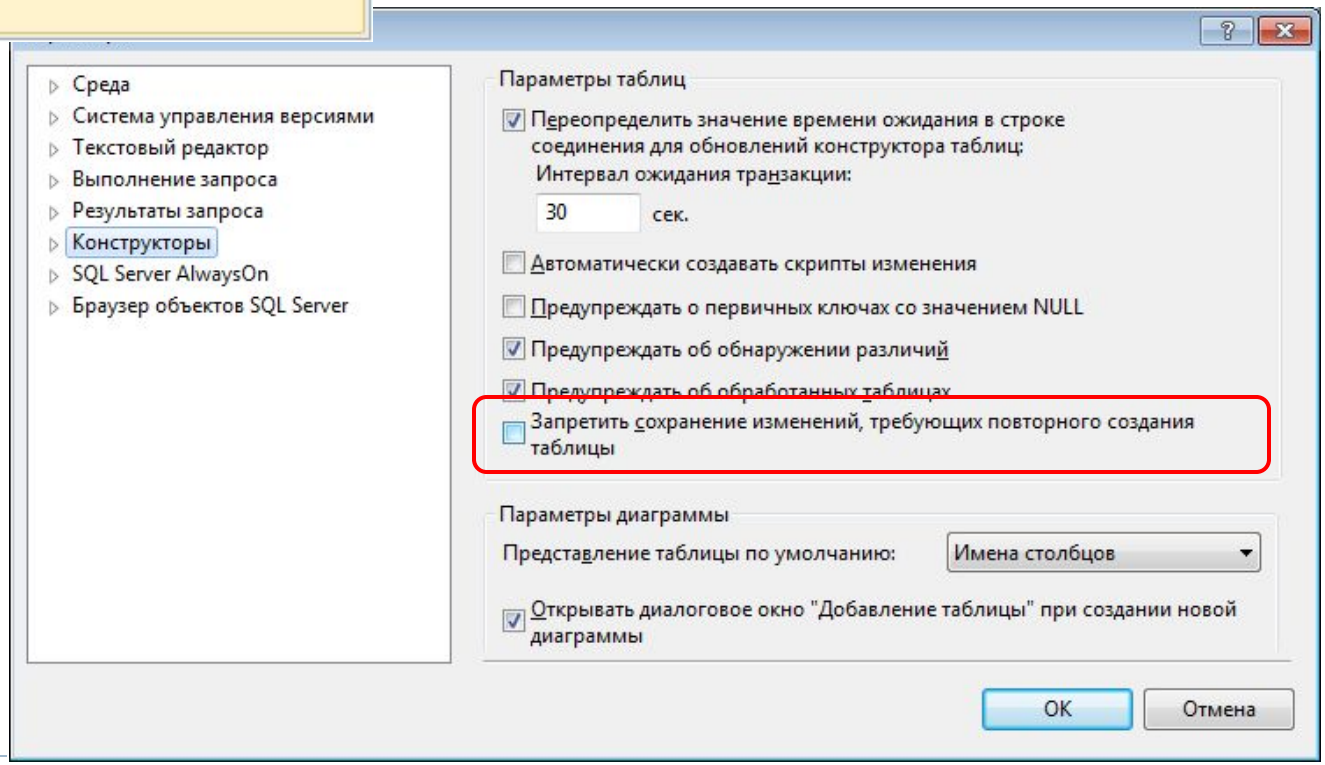
Имя столбца	Тип данных	Разрешить ...
КодДисциплины	int	☐
Дисциплина	char(100)	☑
КодПреподавателя	int	☑

205-11\KIS2.Успев...- dbo.Успеваемость

Имя столбца	Тип данных	Разрешить ...
КодСтудента	int	☐
КодДисциплины	int	☑
Оценка	int	☑
ДатаСдачи	datetime	☑

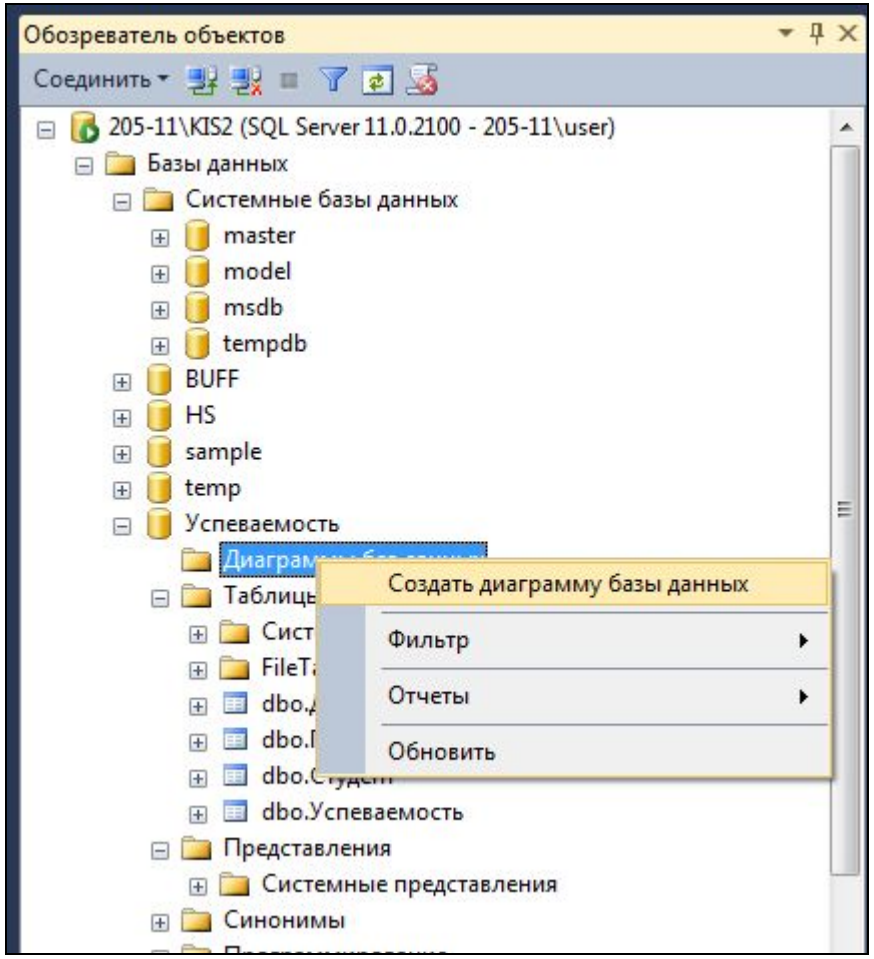
Если появляется сообщение об ошибке при попытке сохранения таблицы…





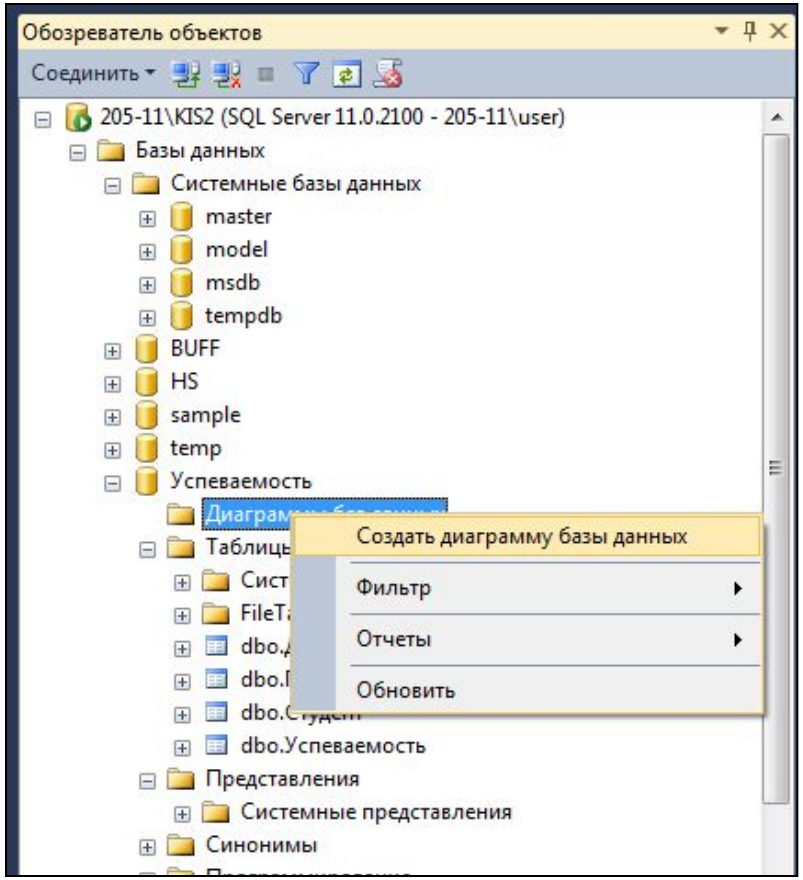
База данных «Успеваемость» (MS SQL Server)

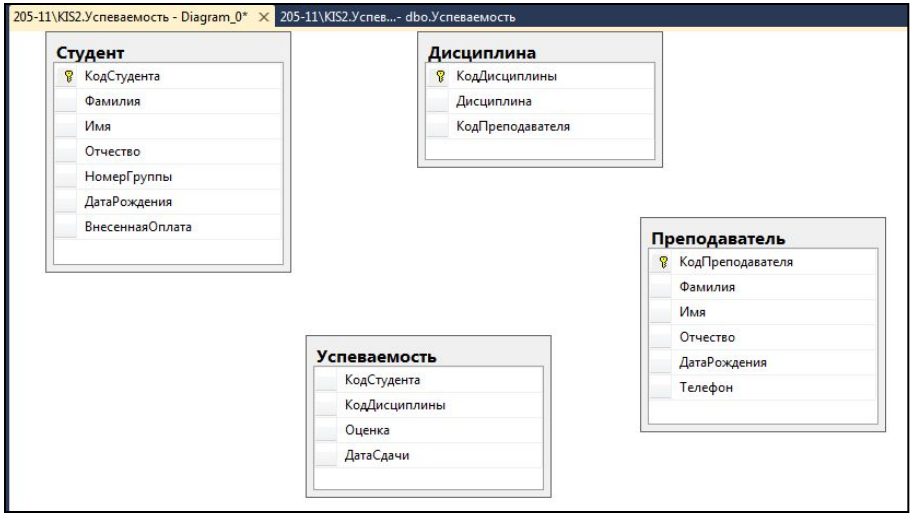
- Создание диаграммы (схемы данных)



База данных «Успеваемость» (MS SQL Server)

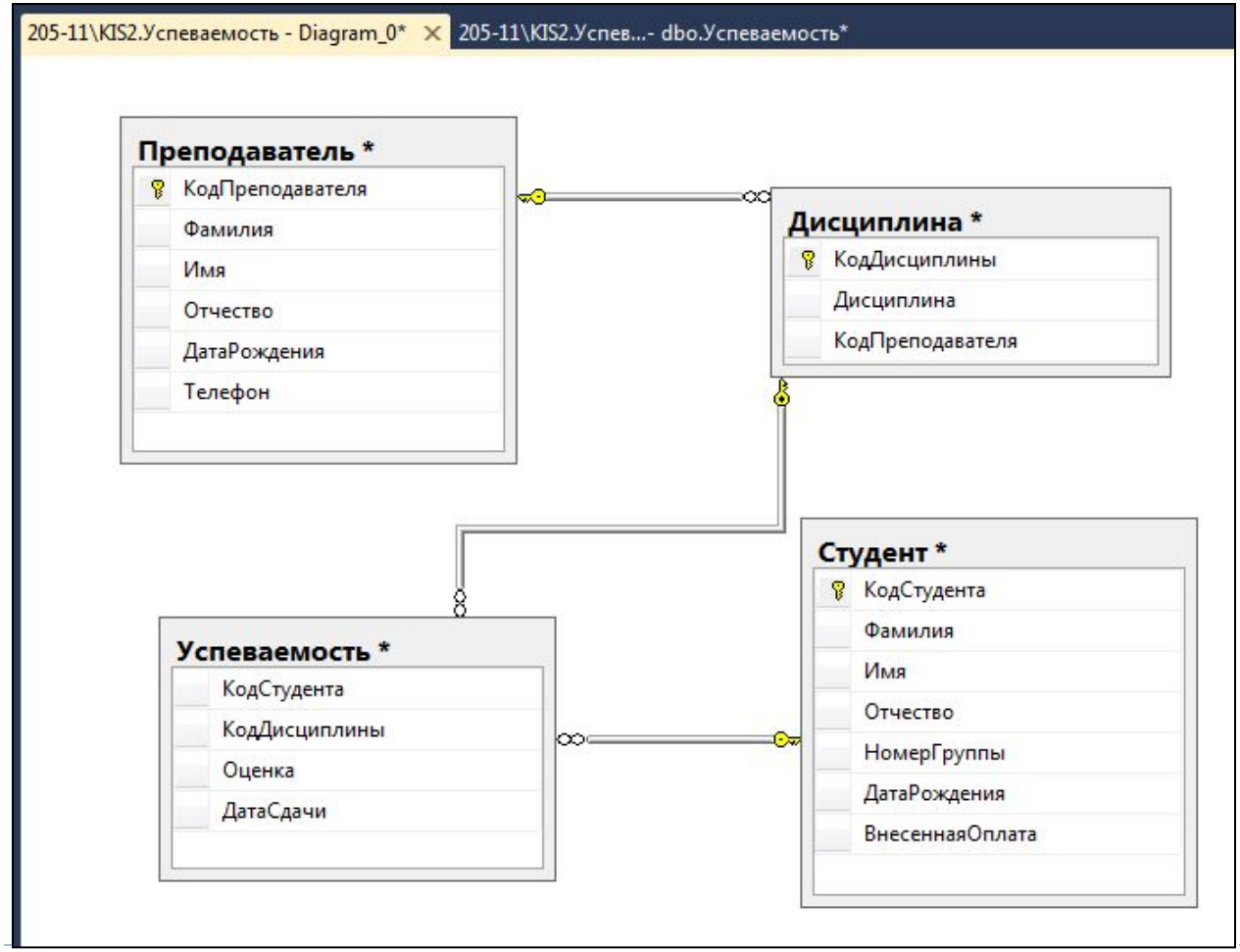
- Создание диаграммы (схемы данных)





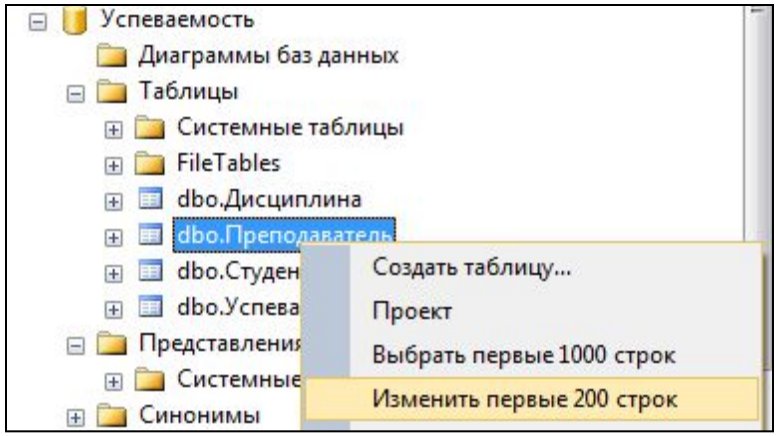
База данных «Успеваемость» (MS SQL Server)

- Создание связей



База данных «Успеваемость» (MS SQL Server)

- Заполнение данными



1. Операция выборки

- Операция выборки позволяет получить все строки (записи) либо часть строк одной таблицы. Каждая таблица или представление, о которых упоминается в запросе, должны быть перечислены в предложении **FROM**.
- Список выбираемых элементов может содержать:
  - имена полей,
  - знак «*»,
  - вычисления,
  - литералы,
  - функции.

1. Операция выборки

Пример 1.1: показать все записи из таблицы «Студент».

SELECT * FROM Студент;

- Знак (*) указывает на то, что требуется вернуть все поля из таблицы.

1. Операция выборки

Пример 1.2: показать записи о студентах из таблицы «Студент», у которых внесенная оплата за обучение составила 45000 рублей.

SELECT * FROM Студент WHERE [внесенная оплата за обучение]=45000;

- Число возвращаемых в результате запроса строк ограничено путем использования предложения **WHERE**, содержащего предикат.

2. Операция проекции

- Операция проекции позволяет выделить **подмножество столбцов** таблицы.
- Требуемые поля перечисляются после **SELECT**.

2. Операция проекции

Пример 2.1: показать записи из таблицы «Студент» (*Фамилия, Номер группы, Внесенная оплата за обучение*).

SELECT Фамилия,[Номер группы],[Внесенная оплата за обучение] FROM Студент;

Комбинация выборки и проекции: выделение **подмножества столбцов и строк** таблицы

Пример 2.2: показать записи из таблицы «Студент» (*Фамилия, Номер группы, Внесенная оплата за обучение*), у которых внесенная оплата за обучение менее 45000 рублей.

SELECT Фамилия, [Номер группы], [Внесенная оплата за обучение] FROM Студент WHERE [внесенная оплата за обучение]<45000;

3. Операция объединения

- Операция объединения позволяет объединять результаты отдельных запросов по нескольким таблицам в единую результирующую таблицу.
- **Основное ограничение: количество полей в запросах должно быть одинаковым и возвращаемые поля должны быть одного типа.**

3. Операция объединения

Пример 3.1: показать коды всех студентов группы БП-115 и студентов, сдавших дисциплину «Информатика и программирование».

```
SELECT [Код студента] FROM Студент WHERE [Номер группы]="БП-115"
UNION
SELECT [Код студента] FROM Успеваемость WHERE [Код дисциплины]=102;
```

4. Вычисления

- Вычисления можно проводить над каждою записью таблицы.
- Заголовок вычисляемого столбца подставляется после ключевого слова AS.

4. Вычисления

Пример 4.1: показать все записи таблицы *Студент (Фамилия, Номер группы, Внесенная оплата за обучение)*, а также вычисляемое поле *Остаток*.

SELECT Фамилия, [Номер группы], [Внесенная оплата за обучение], 45000-[Внесенная оплата за обучение] AS [Остаток] FROM Студент;

4_1

Фамилия	Номер группы	Внесенная оплата за обучение	Остаток
Петров	БП-113	5 000,00р.	40 000,00р.
Иванов	БП-113	5 000,00р.	40 000,00р.
Сидоров	БП-113	5 000,00р.	40 000,00р.
Сенчук	БП-114	10 000,00р.	35 000,00р.
Доликов	БП-115	45 000,00р.	0,00р.
Фукс	БП-114	0,00р.	45 000,00р.
Малыгин	ПИН-113	130 000,00р.	-85 000,00р.
Горелко	ПИН-114	45 000,00р.	0,00р.
Сирдюк	ПИН-115	45 000,00р.	0,00р.
*		0,00р.	

5. Литералы

- Для придания большей наглядности получаемому результату можно использовать литералы.
- Литералы в данном случае представляют собой **строковые константы**, которые применяются наряду с наименованиями.
- Строка символов, представляющая собой литерал, должна быть заключена в одинарные или двойные кавычки.

5. Литералы

Пример 5.1: для предыдущего запроса можно для каждой записи добавить текст *45000-внесенная оплата=*

SELECT Фамилия, [Номер группы], [Внесенная оплата за обучение],"45000-внесенная оплата=" AS Формула, 45000-[Внесенная оплата за обучение] AS Остаток FROM Студент;

5_1

Фамили	Номер группы	Внесенная оплата за обучение	Формула	Остаток
Петров	БП-113	5 000,00р	45000-внесенная оплата=	40 000,00р.
Иванов	БП-113	5 000,00р	45000-внесенная оплата=	40 000,00р.
Сидоров	БП-113	5 000,00р	45000-внесенная оплата=	40 000,00р.
Сенчук	БП-114	10 000,00р	45000-внесенная оплата=	35 000,00р.
Доликов	БП-115	45 000,00р	45000-внесенная оплата=	0,00р.
Фукс	БП-114	0,00р	45000-внесенная оплата=	45 000,00р.
Малыгин	ПИН-113	130 000,00р	45000-внесенная оплата=	-85 000,00р.
Горелко	ПИН-114	45 000,00р	45000-внесенная оплата=	0,00р.
Сирдюк	ПИН-115	45 000,00р	45000-внесенная оплата=	0,00р.
*		0,00р		

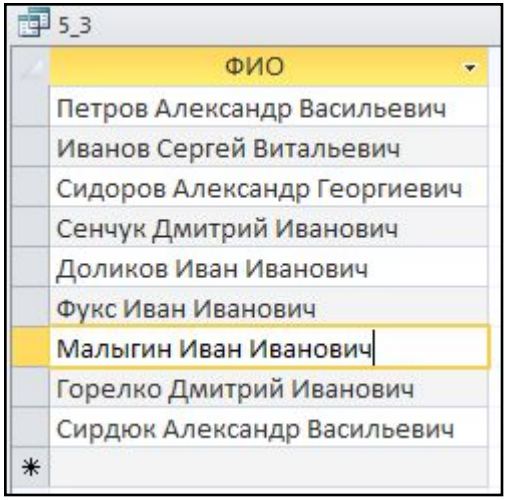
6. Соединение строк

- Имеется возможность соединять два или более столбца, **имеющих строковый тип**, друг с другом, а также соединять их с литералами.
- Для этого используется операция **конкатенации** (в Access это знак +).

6. Соединение строк

Пример 6.1: вывести список студентов, объединив *фамилию, имя, отчество* в один столбец.

SELECT Фамилия+" "+Имя+" "+Отчество AS ФИО FROM Студент;



7. Операции сравнения

- Реляционные операторы могут использоваться с различными элементами.
- **При этом важно соблюдать следующее правило: *элементы должны иметь сравнимые типы*.**
- Если в базе данных определены домены, то сравниваемые элементы должны относиться к одному домену.

7. Операции сравнения

- Операторы, применяемые в построении комплексных запросов:

Оператор	Значение
=	Равно
>	Больше
>=	Больше или равно
<	Меньше
<=	Меньше или равно
<>	Не равно

7. Операции сравнения

Пример 7.1: вывести список студентов (*Фамилия, Имя, Номер группы*) кроме студентов группы БП-113:

SELECT Фамилия, Имя, [Номер группы] FROM Студент WHERE [Номер группы]<>"БП-113";

Замечание. При сравнении литералов конечные пробелы игнорируются. Так, предложение *WHERE Имя = 'Сергей '* будет иметь тот же результат, что и предложение *WHERE Имя = 'Сергей'*.

8. Предикат BETWEEN

- Предикат BETWEEN задает **диапазон значений**, для которого выражение принимает значение **true**.

8. Предикат BETWEEN

Пример 8.1: вывести список студентов (*Фамилия, Имя*), оплативших обучение в размере 10000-20000 рублей:

```
SELECT Фамилия, Имя FROM Студент
WHERE [Внесенная оплата за обучение] BETWEEN
10000 AND 20000;
```

Пример 8.2: тот же запрос с использованием операторов сравнения будет выглядеть следующим образом:

```
SELECT Фамилия, Имя FROM Студент
WHERE [Внесенная оплата за обучение]>= 10000 AND
[Внесенная оплата за обучение]<= 20000;
```

9. Предикат IN

- Предикат IN проверяет, **входит ли заданное значение в указанный в скобках список**.
- Если заданное проверяемое значение равно какому-либо элементу в списке, то предикат принимает значение **true**.

9. Предикат IN

Пример 9.1: вывести данные о преподавателях Волков, Петров, Иванова:

SELECT * FROM Преподаватель WHERE Фамилия IN ("Волков", "Петров", "Иванова");

10. Предикат LIKE

- Предикат LIKE используется только с символьными данными. Он проверяет, соответствует ли данное символьное значение строке с указанной маской.
- В качестве маски используются все разрешенные символы (с учетом верхнего и нижнего регистров), а также специальные символы (% и_):
  - “%” – замещает любое количество символов (в том числе и 0),
  - “_” – замещает только один символ.

В Access знаку % соответствует знак “*”, а знаку “_” соответствует знак “?”.

10. Предикат LIKE

Пример 10.1: вывести данные о студентах с фамилией на букву С:

SELECT * FROM Студент WHERE Фамилия LIKE "С*";

11. Поиск пустых значений

- В SQL-запросах **NULL** означает, что значение столбца неизвестно.
- Предикат **IS NULL** принимает значение *true* только тогда, когда выражение слева от ключевых слов “IS NULL” имеет значение *null* (пусто, не определено).
- Разрешено также использовать конструкцию **IS NOT NULL**, которая означает “не пусто”, “имеет какое-либо значение”.

Пример 11.1: вывести данные о преподавателях с пустым значением Телефон:

SELECT * FROM Преподаватель WHERE Телефон IS NULL;

12. Логические операции

- К логическим относят операторы **AND**, **OR**, **NOT**, позволяющие выполнять различные логические действия.
- Использование этих операторов позволяет гибко “настроить” условия отбора записей.

Оператор	Значение
OR	**Логическое «ИЛИ»**
AND	**Логическое «И»**
NOT	**Не равно (логическое отрицание)**

12. Логические операции

Пример 12.1: вывести данные о студентах из группы БП-115 с оплатой выше 10000 рублей:

SELECT * FROM Студент WHERE [Внесенная оплата за обучение] >10000 AND [Номер группы] = "БП-113";

12. Логические операции

Пример 12.2: вывести данные о студентах с оплатой равной 45000 рублей, кроме студентов группы БП-115 :

```
SELECT * FROM Студент
WHERE [Внесенная оплата за обучение]=45000
AND NOT ([Номер группы]="БП-115");
```

Пример 12.2 (способ 2):

```
SELECT * FROM Студент
WHERE [Внесенная оплата за обучение]=45000
AND ([Номер группы]<>"БП-115");
```

12. Логические операции: порядок выполнения

- В одном предикате логические операторы выполняются в следующем порядке:
  - сначала выполняется оператор **NOT**, затем – **AND** и только после этого – оператор **OR;**
- для изменения порядка выполнения операторов разрешается использовать скобки:
  - сначала проверяется условие, которое находится внутри самых «глубоких» скобок.

13. Сортировка

- Порядок выводимых строк может быть изменен с помощью предложения **ORDER BY** **в конце** **SQL-запроса**.
- Это предложение имеет вид

ORDER BY <порядок> [ASC | DESC]

- Порядок строк может задаваться именами столбцов или номерами столбцов из списка после SELECT.
- Способом по умолчанию – если ничего не указано – является упорядочивание «по возрастанию» (ASC).
- Если же указано слово «DESC», то упорядочивание будет производиться «по убыванию».

13. Сортировка

Пример 13.1: выдать информацию о студентах с внесенной оплатой за обучение по убыванию.

SELECT * FROM Студент ORDER BY [Внесенная оплата за обучение] DESC;

13. Сортировка

- Столбец, определяющий порядок вывода строк, **не обязательно** должен присутствовать в списке выбираемых элементов (столбцов).

Пример 13.2: выдать информацию о студентах (Фамилия, Имя) с внесенной оплатой за обучение по убыванию.

SELECT Фамилия, Имя FROM Студент ORDER BY [Внесенная оплата за обучение] DESC;

13. Сортировка

- Допускается использование нескольких уровней вложенности при упорядочивании выводимой информации по столбцам, при этом разрешается смешивать оба способа.

Пример 13.3: выдать информацию о студентах по убыванию внесенной оплаты за обучение и по алфавиту в каждой «группе» с одинаковой оплатой.

SELECT * FROM Студент ORDER BY [Внесенная оплата за обучение] DESC, Фамилия;

13. Сортировка

Сортировка по полю типа DATETIME. Использование в сортировке функций MONTH, DAY, YEAR

Тип DATETIME содержит информацию о годе (YEAR), месяце (MONTH), дне (DAY), часе (HOUR), минуте (MINUTE), секунде (SECOND) и долях секунды (FRACTION)

Пример 13.4: выдать информацию о студентах по возрастанию числа даты рождения.

SELECT * FROM Студент ORDER BY Day([Дата рождения]);

13. Сортировка

Сортировка по полю типа DATETIME. Использование в сортировке функций MONTH, DAY, YEAR

Тип DATETIME содержит информацию о годе (YEAR), месяце (MONTH), дне (DAY), часе (HOUR), минуте (MINUTE), секунде (SECOND) и долях секунды (FRACTION)

Пример 13.5: выдать информацию о студентах по убыванию числа даты рождения, но по возрастанию года.

SELECT * FROM Студент ORDER BY Day([Дата рождения]) DESC, YEAR ([Дата рождения]);

14. Исключение дубликатов

- Для устранения всех повторов строк из результирующего набора служит модификатор **DISTINCT**.
- Данный модификатор может быть указан только один раз в списке выбираемых элементов и действует на весь список.

14. Исключение дубликатов

Пример 14.1: вывести коды студентов, имеющих оценки (т.е. студентов, коды которых имеются в таблице Успеваемость).

SELECT DISTINCT [Код студента] FROM Успеваемость;

15. Просмотр части данных

- Для отбора нескольких записей из результирующего набора служит модификатор **TOP n** или **TOP n PERCENT**.
- Данный модификатор действует если есть операторы **ORDER BY** или **GROUP BY**.

15. Просмотр части данных

Пример 15.1: показать первые 5 записей студентов (*Дата рождения, Фамилия*) – самых старших из группы.

SELECT TOP 5 [Дата рождения], Фамилия FROM Студент ORDER BY [Дата рождения];

Пример 15.2: показать первые 10 процентов записей студентов (*Дата рождения, Фамилия*) – самых младших из группы.

SELECT TOP 10 PERCENT [Дата рождения], Фамилия FROM Студент ORDER BY [Дата рождения] DESC;

2.2. Групповые операции в запросах

- Использование баз данных на практике ориентировано, прежде всего, на получение итоговых аналитических и справочных отчетов, которые получаются в результате выполнения специальных SQL-запросов.
- В языке SQL **для получения итоговых значений по столбцам** (**агрегирование данных по столбцам**) применяются специальные функции агрегирования

Функции агрегирования

Название функции	Описание функции агрегирования
Count	Подсчитывает количество строк
Sum	Суммирует значение по столбцу
Avg	Рассчитывает среднее значение по столбцу
Max	Определяет максимальное значение по столбцу
Min	Определяет минимальное значение по столбцу
First	…
Last	…
StDev	…
Var	…
VarP	…

Чаще всего такие функции применяются вместе с группировкой.

Функции агрегирования

Пример F1: найти количество студентов, обучающихся в группе БП-113.

```
SELECT COUNT(*) AS Количество
FROM Студент
WHERE [Номер группы]="БП-113";
```

Пример F2: найти количество студентов группы БП-113, родившихся в 1996 году.

```
SELECT COUNT(*) AS Количество
FROM Студент
WHERE [Номер группы]="БП-113" AND
YEAR([Дата рождения])=1996;
```

Функции агрегирования

Пример F3: найти количество студентов, оплативших обучение в размере 45000; найти общую сумму оплат.

```
SELECT COUNT(*) AS Количество, SUM([Внесенная оплата за обучение]) AS [Общая сумма]
FROM Студент
WHERE [Внесенная оплата за обучение]=45000;
```

Пример F4: найти максимальную и минимальную суммы оплат за обучение.

```
SELECT  MAX([Внесенная оплата за обучение]) AS [Максимальная оплата], MIN([Внесенная оплата за обучение]) AS [Минимальная оплата]
FROM Студент;
```

Групповые операции. Предложение GROUP BY

- Операция группировки объединяет записи с одинаковыми значениями в указанном списке полей в одну запись. Поля указываются после GROUP BY.
- Если инструкция SELECT содержит функцию агрегации языка SQL (*например, Sum или Count*), то для каждой записи будет вычислено итоговое значение.
- **Основное правило:** при использовании предложения GROUP BY все поля в списке полей инструкции SELECT должны быть либо включены в предложение GROUP BY, либо использоваться в качестве аргументов статистической функции SQL.

Предложение GROUP BY. Синтаксис

SELECT список_полей

FROM таблица

WHERE условие_отбора

GROUP BY группируемые_поля;

- где группируемые_поля - имена полей (до 10), которые используются для группирования записей.

Предложение GROUP BY

Пример G1: изменим запрос F4: определить максимальную и минимальную суммы оплат за обучение в каждой группе.

SELECT [Номер группы], MAX([Внесенная оплата за обучение]), MIN ([Внесенная оплата за обучение])

FROM Студент

GROUP BY [Номер группы];

Номер групі	Макс оплата в каждой группе	Мин оплата в каждой группе
БП-113	20 000,00р.	5 000,00р.
БП-114	20 000,00р.	0,00р.
БП-115	45 000,00р.	45 000,00р.
ПИН-113	130 000,00р.	130 000,00р.
ПИН-114	45 000,00р.	45 000,00р.
ПИН-115	45 000,00р.	45 000,00р.

Примеры математических функций, которые используются в стандартном SQL

Функция	Описание операции
SIN()	Вычисление синуса
COS()	Вычисление косинуса
LOG()	Вычисление логарифма
ROUND(X,Y)	Округление X до Y знаков после запятой
SQR()	Вычисление квадрата
SIGN()	Вычисление знака
ABS()	Вычисление абсолютного значения

Предложение GROUP BY

Пример G2: найти средний балл по каждой дисциплине, округлить результат до одного знака после запятой.

Без округления:

```
SELECT [Код дисциплины], AVG (Оценка)  AS [Средний балл]
FROM Успеваемость
GROUP BY [Код дисциплины];
```

Код дисциплины	Средний балл
Математика	3,5
Информатика и программирование	4,5
История	3,66666666666667
Экономика	3

С округлением:

```
SELECT [Код дисциплины], ROUND(AVG (Оценка),1) AS [Средний балл]
FROM Успеваемость
GROUP BY [Код дисциплины];
```

Код дисциплины	Средний балл
Математика	3,5
Информатика и программирование	4,5
История	3,7
Экономика	3

Предложение GROUP BY

Пример G3: вывести количество сдач/пересдач студентом зачета (экзамена) по каждой дисциплине.

```
SELECT  [Код студента], [Код дисциплины], Count (Оценка)  AS [Кол_сдач]
FROM Успеваемость
GROUP BY [Код студента], [Код дисциплины];
```

Предложение HAVING

- Определяет, какие сгруппированные записи отображаются при использовании инструкции SELECT **с предложением GROUP BY**.
- **После того как записи будут сгруппированы с помощью предложения GROUP BY, предложение HAVING отберет те из полученных записей, которые удовлетворяют условиям отбора**, указанным в предложении HAVING.

Предложение HAVING, *синтаксис*

SELECT список_полей

FROM таблица

WHERE условие_отбора

GROUP BY группируемые_поля

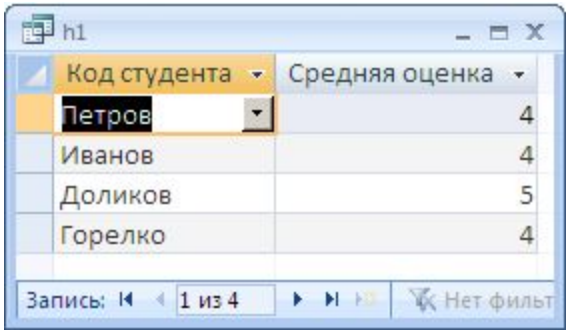
HAVING условие_отбора_групп;

где **условие_отбора_групп** - выражение, определяющее, какие сгруппированные записи отображать.

Предложение HAVING

Пример H1: вывести список студентов (*код студента, средний балл*), средний балл которых выше 3,5

```
SELECT [Код студента], AVG(Оценка) AS [Средняя оценка]
FROM Успеваемость
GROUP BY [Код студента]
HAVING AVG(Оценка)>3.5;
```



2.3. Запросы с несколькими таблицами (соединение таблиц)

- Операция соединения используется в языке SQL для вывода связанной информации, хранящейся в нескольких таблицах.
- В этом проявляется одна из наиболее важных особенностей запросов SQL – способность определять связи между многочисленными таблицами и выводить информацию из них в рамках этих связей.

Виды связей

- Внутренние
- Внешние
- Рекурсивные
- По отношению

Все виды, кроме внешнего, можно задавать в предложении **WHERE** запроса SELECT.

Внешние и внутренние соединения можно задавать с помощью зарезервированного слова **JOIN**.

Особенности связей

- Связывание производится, как правило, по первичному ключу одной таблицы и внешнему ключу другой таблицы – для каждой пары таблиц.
- Соединяемые поля могут (но не обязаны!) присутствовать в списке выбираемых элементов.
- Предложение WHERE может содержать множественные условия соединений.
- Условие соединения может также комбинироваться с другими предикатами в предложении WHERE.

1. Внутреннее соединение с помощью WHERE

- Внутреннее соединение возвращает только те строки, для которых условие соединения принимает значение TRUE.

Пример W1: вывести названия дисциплин и фамилии ведущих преподавателей

SELECT Дисциплина.[Название дисциплины], Преподаватель.Фамилия

FROM Дисциплина, Преподаватель

WHERE Дисциплина.[Код преподавателя] = Преподаватель.[Код преподавателя];

Замечание. Имена столбцов в строке с ключевым словом SELECT записаны в полной синтаксической структуре: <имя таблицы>.<имя столбца>

1. Внутреннее соединение с помощью WHERE. Алиасы

- В вышеприведенном запросе использовался **способ непосредственного указания таблиц с помощью их имен**.
- Возможен (а иногда и просто необходим) также способ указания таблиц с помощью **алиасов** (псевдонимов).
- Алиасы определяются в предложении FROM запроса SELECT и представляют собой любой допустимый идентификатор, написание которого подчиняется таким же правилам, что и написание имен таблиц.
- Потребность в алиасах таблиц возникает тогда, когда названия столбцов, используемых в условиях соединения двух (или более) таблиц, совпадают.
- Часто алиасы используются в подзапросах (см. далее).
- В одном запросе нельзя смешивать использование написания имен таблиц и их алиасов.
- Алиасы таблиц могут совпадать с их именами.

1. Внутреннее соединение с помощью WHERE. Алиасы

Пример W2: рассмотрим вышеприведенный пример с использованием алиасов X иY (показать названия дисциплин и фамилии ведущих преподавателей).

```
SELECT X.[Название дисциплины], Y.Фамилия
FROM Дисциплина AS X, Преподаватель AS Y
WHERE X.[Код преподавателя] =
Y.[Код преподавателя];
```

2. Внутреннее соединение с помощью INNER JOIN

- **INNER JOIN** объединяет записи из двух таблиц, **если связующие поля этих таблиц содержат одинаковые значения**

2. Внутреннее соединение с помощью INNER JOIN. Синтаксис

FROM таблица1

INNER JOIN таблица2

ON таблица1.поле1 оператор_сравнения таблица2. поле2

где:

- таблица1, таблица2 - имена таблиц, записи которых подлежат объединению;
- поле1, поле2 - имена объединяемых полей. Если эти поля не являются числовыми, то должны иметь одинаковый тип данных и содержать данные одного рода, однако они могут иметь разные имена;
- оператор_сравнения - любой оператор сравнения: =, <, >, <=, >=, <>.

2. Внутреннее соединение с помощью INNER JOIN

Пример I1: показать названия дисциплин и фамилии ведущих преподавателей.

```
SELECT Дисциплина.[Название дисциплины],
   Преподаватель.Фамилия
FROM Дисциплина INNER JOIN Преподаватель ON
   Дисциплина.[Код преподавателя]=
   Преподаватель.[Код преподавателя];
```

2. Внутреннее соединение с помощью INNER JOIN

Пример I2: вывести список студентов, дисциплины (по кодам) и полученные по ним оценки.

```
SELECT Студент.Фамилия, Успеваемость.[Код дисциплины], Успеваемость.Оценка
FROM Студент INNER JOIN Успеваемость
 ON Студент.[Код студента]=
Успеваемость.[Код студента];
```

3. Внутреннее соединение (INNER JOIN) с условиями отбора

Пример I3: вывести список студентов и полученные оценки по дисциплине с кодом 102.

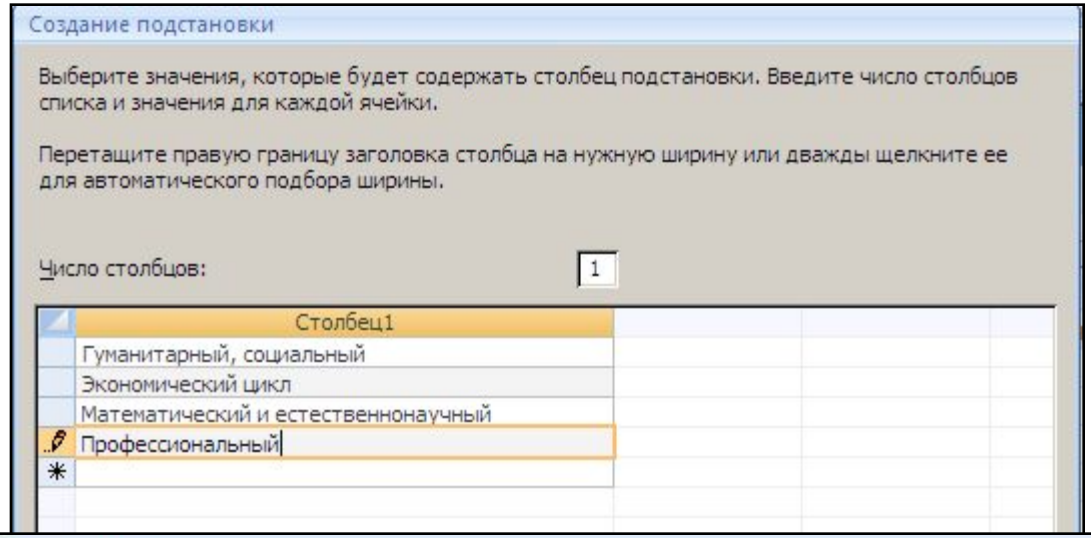
```
SELECT Студент.Фамилия, Успеваемость.Оценка
FROM Студент INNER JOIN Успеваемость ON
  Студент.[Код студента]= Успеваемость.[Код
  студента]
WHERE Успеваемость.[Код дисциплины]=102;
```

4. Косвенные соединения (INNER JOIN)

- С помощью внутренних соединений организуются **косвенные соединения**, когда в запросе указываются **все промежуточные таблицы**, связанные **внутренними связями** (т.е. связываются более двух таблиц, предложение INNER JOIN будет встречаться несколько раз)

4. Косвенные соединения (INNER JOIN)

- **Изменение структуры таблицы Дисциплина: добавлено поле Цикл:**



Дисциплина

Код дисциплины	Название дисциплины	Код преподавателя	Цикл
101	Математика	Волков	Математический и естественнонаучный
102	Информатика и программирование	Петров	Математический и естественнонаучный
103	История	Ковальчук	Гуманитарный, социальный
104	Экономика	Иванова	Экономический цикл
105	Технология разработки и защиты баз данных	Волков	Профессиональный

4. Косвенные соединения (INNER JOIN)

Пример I8 (объединение 3-х таблиц): вывести список студентов (*Фамилии*), дисциплины (*Название, Цикл*) и полученные по ним оценки.

```
SELECT Студент.Фамилия, Дисциплина.[Название дисциплины], Дисциплина.Цикл, Успеваемость. Оценка
FROM (Студент INNER JOIN Успеваемость ON Студент.[Код студента]=Успеваемость.[Код студента])
INNER JOIN Дисциплина ON Успеваемость.[Код дисциплины]=Дисциплина.[Код дисциплины];
```

4. Косвенные соединения (INNER JOIN)

Пример I9 (объединение 4-х таблиц):

```
SELECT Студент.Фамилия, Студент.[Номер группы],
    Дисциплина.[Название дисциплины], Успеваемость.
    Оценка, Преподаватель.Фамилия, Преподаватель.Имя,
    Преподаватель.Отчество
FROM (Студент INNER JOIN Успеваемость ON
    Студент.[Код студента]=Успеваемость.[Код
    студента])
INNER JOIN (Дисциплина INNER JOIN Преподаватель ON
    Дисциплина.[Код преподавателя]= Преподаватель.[Код
    преподавателя]) ON Успеваемость.[Код
    дисциплины]=Дисциплина.[Код дисциплины];
```

5. Соединение таблиц с группировкой записей

- Группировка выполняется в предложении GROUP BY в конце запроса.
- Таблица с группировкой записей указывается после INNER JOIN.
- Все поля после SELECT либо перечисляются в GROUP BY либо записываются со статистическими функциями.

5. Соединение таблиц с группировкой записей

Пример IG1: вывести информацию о студентах (Фамилия, Имя, Номер группы, Средний балл). Группировка записей будет выполняться в таблице *Успеваемость* по полю *Код студента.*

```
SELECT Студент.Фамилия, Студент.Имя,
Студент.[Номер группы], AVG(Успеваемость.
Оценка) AS [Средний балл]
FROM Студент INNER JOIN Успеваемость ON
Студент.[Код студента]=Успеваемость.[Код
студента]
GROUP BY Успеваемость.[Код студента], Студент.
Фамилия, Студент.Имя, Студент.[Номер группы];
```

6. Внешнее соединение

- Внешнее соединение возвращает **все строки из одной таблицы и только те строки из другой таблицы, для которых условие соединения принимает значение true**.
- Строки второй таблицы, не удовлетворяющие условию соединения (т.е. имеющие значение *false*), получают значение *null* в результирующем наборе.

6. Внешнее соединение, виды

- Существуют два вида внешнего соединения:
  - **LEFT JOIN**
  - **RIGHT JOIN**

  В левом соединении (LEFT JOIN) запрос возвращает все строки из левой таблицы (т.е. таблицы, стоящей *слева* от зарезервированного словосочетания “LEFT JOIN”). Для правого соединения – все наоборот.

6. Внешнее соединение, синтаксис

FROM таблица1 [LEFT|RIGHT] JOIN таблица2

ON таблица1.поле1 оператор_сравнения таблица2.поле2

6. Внешнее соединение, синтаксис

Пример LR1 (внешнее соединение):

- Вывести список всех преподавателей (*Фамилия, Имя*). Для тех преподавателей, которые в настоящее время преподают в институте – вывести название дисциплины.

Преподаватель

Код преподавателя	Телефон	Фамилия	Имя	Отчество	Дата рождения	*Щелкните для добавления*
200	8-913-8887766	Волков	Семен	Игоревич	05.12.1975	
201	8-913-4447766	Петров	Петр	Васильевич	24.12.1971	
202	8-913-3337766	Иванова	Наталья	Владимировн	05.04.1977	
203	8-913-3334326	Ковальчук	Сергей	Григорьевич	15.12.1965	
204	8-913-1114326	Лаптева	Галина	Ивановна	31.12.1975	

Код дисциплины	Название дисциплины	Код преподавателя	Цикл
101	Математика	Волков	Математический и естественнонаучный
102	Информатика и программирование	Петров	Математический и естественнонаучный
103	История	Ковальчук	Гуманитарный, социальный
104	Экономика	Иванова	Экономический цикл
105	Технология разработки и защиты баз данных	Волков	Профессиональный

6. Внешнее соединение

Пример LR1 (внешнее соединение):

- Вывести список всех преподавателей (*Фамилия, Имя*). Для тех преподавателей, которые в настоящее время преподают в институте – вывести название дисциплины.

SELECT Преподаватель.Фамилия, Преподаватель.Имя, Дисциплина.[Название дисциплины]

FROM Преподаватель LEFT JOIN Дисциплина ON Преподаватель.[Код преподавателя]= Дисциплина.[Код преподавателя];

lr1

Фамилия	Имя	Название дисциплины
Волков	Семен	Математика
Волков	Семен	Технология разработки и защиты баз данных
Петров	Петр	Информатика и программирование
Иванова	Наталья	Экономика
Ковальчук	Сергей	История
Лаптева	Галина	

6. Внешнее соединение

Пример LR2 (внешнее соединение): если данном запросе использовать *RIGHT JOIN?*

FROM Преподаватель RIGHT JOIN Дисциплина ON Преподаватель.[Код преподавателя]= Дисциплина.[Код преподавателя];

Будут выведены все дисциплины и закрепленные за ними преподаватели.

6. Внешнее соединение

Пример LR2 (внешнее соединение): если данном запросе использовать *RIGHT JOIN?*

Будут выведены все дисциплины и закрепленные за ними преподаватели.

Дисциплина

Код дисциплины	Название дисциплины	Код преподавателя	Цикл
101	Математика	Волков	Математический и естественнонаучный
102	Информатика и программирование	Петров	Математический и естественнонаучный
103	История	Ковальчук	Гуманитарный, социальный
104	Экономика	Иванова	Экономический цикл
105	Технология разработки и защиты баз данных	Волков	Профессиональный
106	Технология разработки ПО		Профессиональный

Преподаватель

Код преподавателя	Телефон	Фамилия	Имя	Отчество	Дата рождения	*Щелкните для добавления*
200	8-913-8887766	Волков	Семен	Игоревич	05.12.1975	
201	8-913-4447766	Петров	Петр	Васильевич	24.12.1971	
202	8-913-3337766	Иванова	Наталья	Владимировн	05.04.1977	
203	8-913-3334326	Ковальчук	Сергей	Григорьевич	15	
204	8-913-1114326	Лаптева	Галина	Ивановна	31	

lr2

Фамилия	Имя	Название дисциплины
		Технология разработки ПО
Волков	Семен	Математика
Волков	Семен	Технология разработки и защиты баз данны
Петров	Петр	Информатика и программирование
Иванова	Наталья	Экономика
Ковальчук	Сергей	История

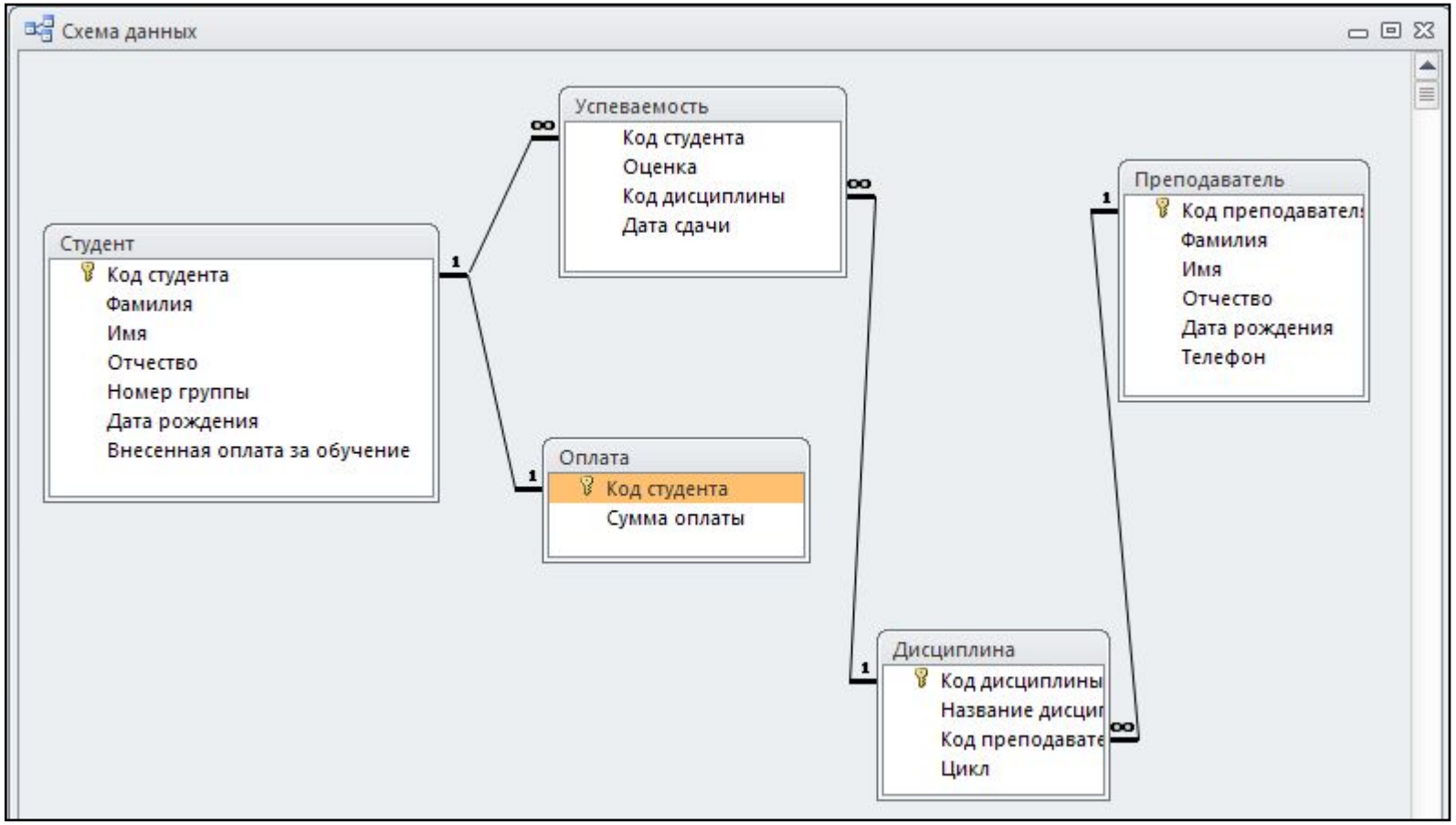
Запись: 1 из 6 | Нет фильтра | Поиск

7. Соединение по отношению

- Соединение по отношению (**тета-соединение**) представляет собой способ соединения по любому отношению, кроме равенства.

7. Соединение по отношению

- Добавление таблицы: **Оплата**



7. Соединение по отношению

- Добавление таблицы: **Оплата**

Студент

Код ст	Фамили	Имя	Отчество	Номе	Дата рожден	Внесенная оплата за обуче
300	Петров	Александр	Васильевич	БП-113	03.09.1997	20 000,00р.
301	Иванов	Сергей	Витальевич	БП-113	11.05.1998	5 000,00р.
302	Сидоров	Александр	Георгиевич	БП-113	23.09.1996	5 000,00р.
303	Сенчук	Дмитрий	Иванович	БП-114	03.12.1995	20 000,00р.
304	Доликов	Иван	Иванович	БП-115	13.09.1994	45 000,00р.
305	Фукс	Иван	Иванович	БП-114	13.09.1996	0,00р.
306	Малыгин	Иван	Иванович	ПИН-11	13.09.1994	130 000,00р.
307	Горелко	Дмитрий	Иванович	ПИН-11	03.12.1995	45 000,00р.
308	Сирдюк	Александр	Васильевич	ПИН-11	03.09.1990	45 000,00р.
*						0,00р.

Оплата

Код студент	Сумма оплаты
300	45 000,00р.
301	45 000,00р.
302	45 000,00р.
303	45 000,00р.
304	45 000,00р.
305	45 000,00р.
306	45 000,00р.
307	45 000,00р.
308	45 000,00р.

7. Соединение по отношению

Пример Т1(тета-соединение): вывести тех студентов (*Фамилия, Имя, Внесенная оплата за обучение, Сумма оплаты*), для которых *Внесенная оплата за обучение* и *Сумма оплаты различны*.

SELECT Студент.Фамилия, Студент.Имя, Студент.[Внесенная оплата за обучение], Оплата.[Сумма оплаты] FROM Студент

INNER JOIN Оплата ON Студент.[Код студента]=Оплата.[Код студента]

WHERE Студент.[Внесенная оплата за обучение]<>Оплата.[Сумма оплаты];

t1

Фамили	Имя	Внесенная оплата за обучение	Сумма опла
Иванов	Сергей	5 000,00р.	45 000,00р.
Сидоров	Александр	5 000,00р.	45 000,00р.
Сенчук	Дмитрий	20 000,00р.	45 000,00р.
Фукс	Иван	0,00р.	45 000,00р.
Малыгин	Иван	130 000,00р.	45 000,00р.
Петров	Александр	20 000,00р.	45 000,00р.

Запись: 1 из 6 Нет фильтра Поиск

8. Рекурсивные соединения

- В некоторых задачах необходимо получить информацию, выбранную особым образом только из одной таблицы. Для этого используются так называемые **самосоединения – рекурсивные соединения**.
- **Рекурсивное соединение** – это соединение таблицы с собой с помощью алиасов.
- Самосоединения полезны в случаях, когда нужно получить **пары аналогичных элементов из одной и той же таблицы**.

8. Рекурсивные соединения

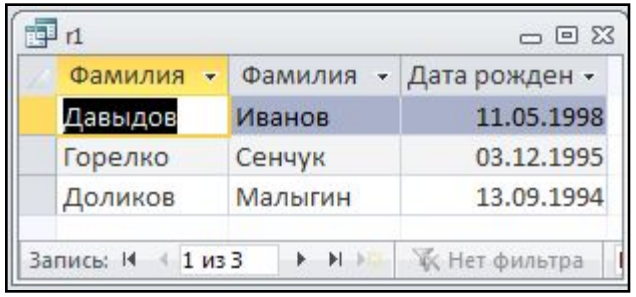
Пример R1: вывести студентов (*Фамилия, Дата рождения*) с одинаковыми датами рождения.

```
SELECT X.Фамилия, Y.Фамилия, Y.[Дата рождения]
FROM Студент AS X, Студент AS Y
WHERE X.Фамилия<Y.Фамилия AND X.[Дата рождения]=Y.[Дата рождения];
```

8. Рекурсивные соединения

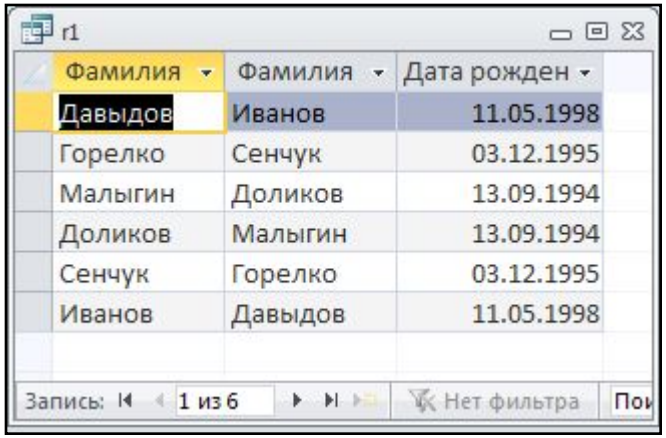
Пример R1: Почему оператор *<?*

WHERE X.Фамилия<Y.Фамилия AND X.[Дата рождения]=Y.[Дата рождения];



Пример R1: Если*<>?*

WHERE X.Фамилия<>Y.Фамилия AND X.[Дата рождения]=Y.[Дата рождения];



2.4. Использование подзапросов (подчиненные запросы)

- **Подзапросы** – это запросы, которые предназначены для обработки внутри другого запроса.
- Подзапросы могут использоваться в предикатах, операторах DELETE , UPDATE, ограничениях, предложениях FROM.
- Подзапросы могут возвращать одно значение (**простой подзапрос**) или множество значений (**табличный подзапрос**).

Типы подзапросов

Существует три типа подзапросов:

1. **Связанный подзапрос** возвращает значение, выбираемое из пересечения одного столбца с одной строкой – то есть единственное значение.
2. **Строковый подзапрос** возвращает значение нескольких столбцов таблицы, но в виде единственной строки.
3. **Табличный подзапрос** возвращает значения одного или более столбцов в более чем одной строке.

Действия

- Подчиненные запросы используются для выполнения следующих действий:
  - проверка в подчиненном запросе существования некоторых результатов (с помощью зарезервированных слов EXISTS или NOT EXISTS);
  - поиск в главном запросе любых значений, которые равны, больше или меньше значений, возвращаемых в подчиненном запросе (с помощью зарезервированных слов ANY, IN, SOME или ALL);
  - создание подчиненных запросов внутри подчиненных запросов (вложенных подчиненных запросов).

Использование подзапросов: синтаксис

SELECT …

WHERE выражение сравнение {ALL | ANY | SOME} (подзапрос)

Подзапрос **всегда заключается в круглые скобки**. Уровень вложенности ограничивается в конкретной реализации SQL.

В SQL выражение **X < > ALL( )** соответствует **«не равен любому»** результату подзапроса, т. е. предикат истинен, если значение X отсутствует среди результатов подзапроса.

Операторы **SOME и ANY** полностью **взаимозаменяемы**; можно использовать тот, который больше нравится. Оператор SOME (ANY) истинен, если какое-нибудь из выведенных подзапросом значений удовлетворяет заданному предикату.

Ограничения

- **Не допускается** использование инструкции SELECT подчиненного запроса в запросе на объединение или в перекрестном запросе.
- Подзапросы заметно замедляют выполнение запросов, поэтому их следует **использовать только обоснованно**, когда другим способом результат получить нельзя (например, при удалении или обновлении).

Необоснованное применение подзапроса

Пример. Показать названия дисциплин, которые ведутся преподавателем Волковым.

С подзапросом:

```
SELECT [Название дисциплины] FROM Дисциплина WHERE [Код преподавателя]=(SELECT [Код преподавателя] FROM Преподаватель WHERE Фамилия="Волков");
```

Без подзапроса :

```
SELECT Дисциплина.[Название дисциплины]
FROM Дисциплина INNER JOIN Преподаватель ON Дисциплина.[Код преподавателя]=Преподаватель.[Код преподавателя]
WHERE  Преподаватель.Фамилия="Волков";
```

Использование подзапросов

Пример Р1: вывести студентов (Фамилия, Имя, Номер группы), которые внесли оплату за обучение больше всех студентов группы БП-113.

```
SELECT Фамилия, Имя, [Номер группы] FROM Студент WHERE
[Внесенная оплата за обучение] > ALL
   (SELECT  [Внесенная оплата за обучение]
   FROM Студент   WHERE [Номер группы] = "БП-113");
```

Студент : таблица

Код студ	Фамилия	Имя	Отчество	Номер группы	Дата рождения	Внесенная оплата за обучение
309	Давыдов	Денис	Сергееивч	БП-113	11.05.1998	0,00р.
302	Сидоров	Александр	Георгиевич	БП-113	23.09.1996	5 000,00р.
301	Иванов	Сергей	Витальевич	БП-113	11.05.1998	5 000,00р.
300	Петров	Александр	Васильевич	БП-113	03.09.1997	20 000,00р.
305	Фукс	Иван	Иванович	БП-114	13.09.1996	0,00р.
303	Сенчук	Дмитрий	Иванович	БП-114	03.12.1995	20 000,00р.
310	Мальцев	Андрей	Сергеевич	БП-115	11.05.1998	0,00р.
304	Доликов	Иван	Иванович	БП-115	13.09.1994	45 000,00р.
306	Малыгин	Иван	Иванович	ПИН-113	14.09.1994	130 000,00р.
307	Горелко	Дмитрий	Иванович	ПИН-114	03.12.1995	45 000,00р.
308	Сирдюк	Александр	Васильевич	ПИН-115	03.09.1990	45 000,00р.
						0,00р.

Запись: 12 из 12

Р1 : запрос на выборку

Фамилия	Имя	Номер групп
Доликов	Иван	БП-115
Малыгин	Иван	ПИН-113
Горелко	Дмитрий	ПИН-114
Сирдюк	Александр	ПИН-115

Запись: 1 из 4

Использование подзапросов

Пример Р2: вывести студентов (Фамилия, Имя, Номер группы) у которых отсутствуют оценки.

(*следовательно, о таких студентах нет записей в таблице успеваемость*)

```
SELECT Фамилия, Имя, [Номер группы]
FROM Студент
WHERE [Код студента] <> ALL
 (SELECT  [Код студента] FROM Успеваемость);
```

Использование подзапросов

Пример Р3: вывести студентов (Фамилия, Имя, Номер группы) у которых есть хотя бы одна оценка.

1 способ:

```
SELECT Фамилия, Имя, [Номер группы]
FROM Студент
WHERE [Код студента] =ANY
 (SELECT  [Код студента] FROM Успеваемость);
```

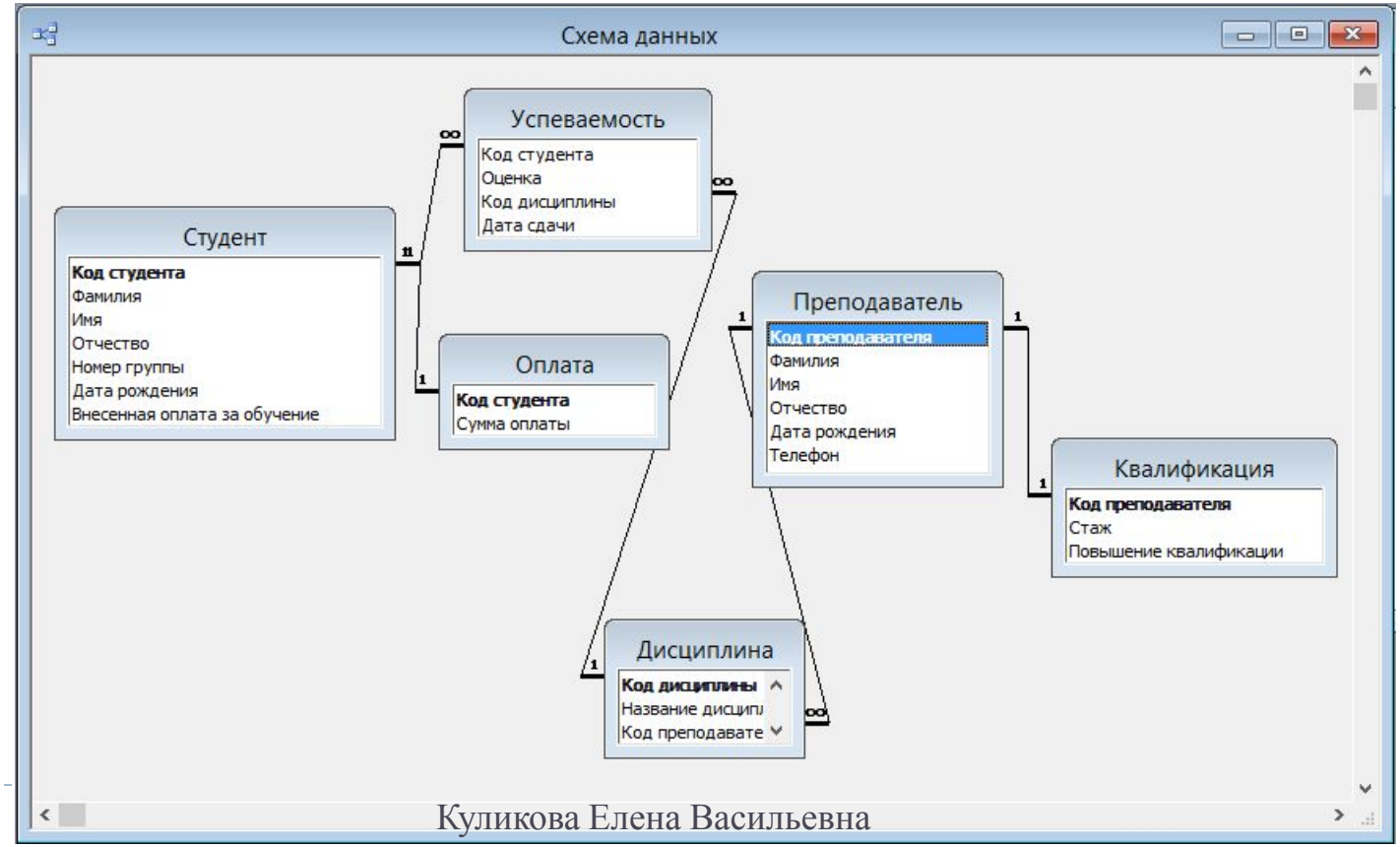
2 способ:

```
SELECT Фамилия, Имя, [Номер группы]
FROM Студент
WHERE [Код студента] =SOME
 (SELECT  [Код студента] FROM Успеваемость);
```

Изменение структуры базы данных

Квалификация : таблица

	Имя поля	Тип данных
🔑	Код преподавателя	Числовой
	Стаж	Числовой
	Повышение квалификации	Числовой

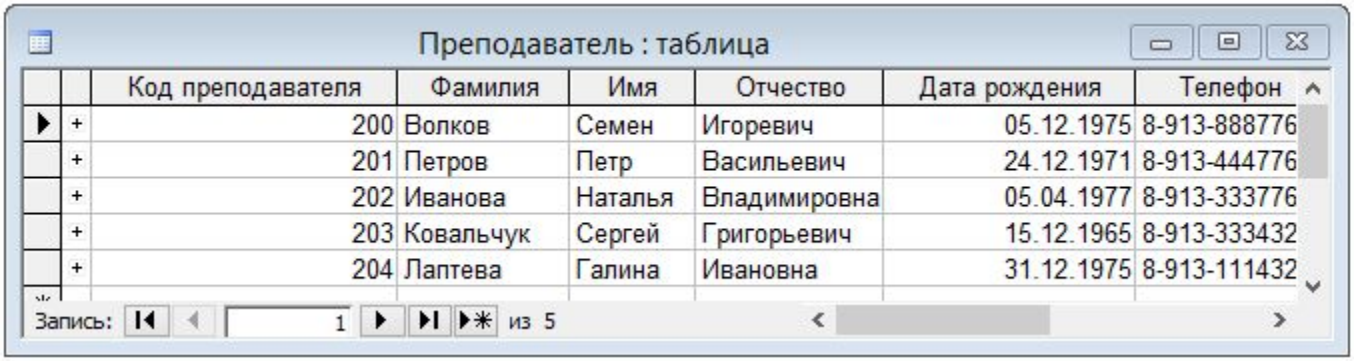


Использование подзапросов

Пример Р4: вывести преподавателей (Фамилия, Имя, Отчество) со стажем работы более 10 лет

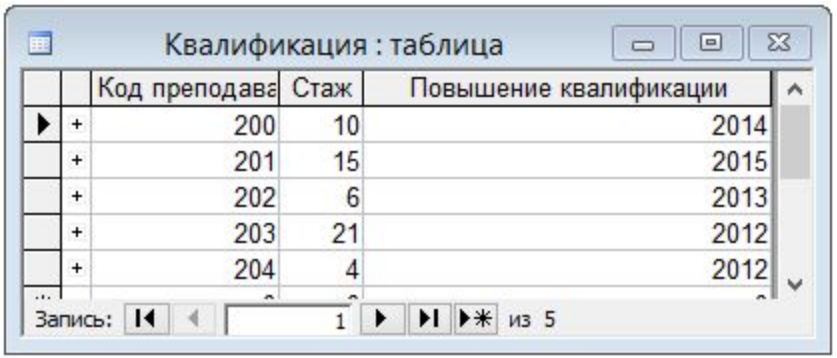
```
SELECT Фамилия, Имя, Отчество
FROM Преподаватель
WHERE [Код преподавателя] IN
(SELECT  [Код преподавателя] FROM Квалификация WHERE
    Стаж>10);
```

Использование подзапросов


Преподаватель : таблица

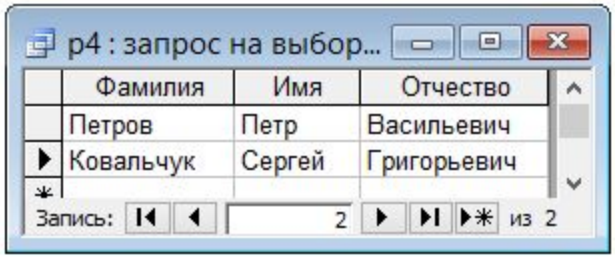
Код преподавателя	Фамилия	Имя	Отчество	Дата рождения	Телефон
200	Волков	Семен	Игоревич	05.12.1975	8-913-888776
201	Петров	Петр	Васильевич	24.12.1971	8-913-444776
202	Иванова	Наталья	Владимировна	05.04.1977	8-913-333776
203	Ковальчук	Сергей	Григорьевич	15.12.1965	8-913-333432
204	Лаптева	Галина	Ивановна	31.12.1975	8-913-111432

Запись: 1 из 5


Квалификация : таблица

Код преподава	Стаж	Повышение квалификации
200	10	2014
201	15	2015
202	6	2013
203	21	2012
204	4	2012

Запись: 1 из 5


p4 : запрос на выбор...

Фамилия	Имя	Отчество
Петров	Петр	Васильевич
Ковальчук	Сергей	Григорьевич

Запись: 2 из 2

2.5. Запросы на модификацию данных

Три основных оператора манипулирования данными:

- INSERT (вставка)
- UPDATE (обновление)
- DELETE (удаление)

1. Запросы на вставку данных в таблицы

Оператор INSERT INTO

Синтаксис:

INSERT INTO <таблица> (<поле1>,<поле2>,...) VALUES (<значение1>, <значение2>,...);

- вставляет в таблицу **одну** новую запись
- после имени таблицы в скобках необходимо указать те поля, которым требуется присвоить некоторые значения явно
- за ключевым словом VALUES в скобках следует список значений для перечисленных полей; число значений в этом списке должно соответствовать числу указанных полей
- полям, не названным в списке присваивается значение NULL

1. Запросы на вставку данных в таблицы

Пример INS1: вставить запись о студенте (*Код студента, Фамилия, Имя, Отчество*).

INSERT INTO Студент ([Код студента], Фамилия, Имя, Отчество) VALUES (311, "Захаров", "Андрей", "Васильевич");

1. Запросы на вставку данных в таблицы

Если необходимо вставить данные по полю с типом данных Счетчик, то значение данного поля указывать не требуется (но при необходимости возможно)

Пример INS1: (если *Код студента - счетчик*): вставить запись о студенте (*Фамилия, Имя, Отчество*).

```
INSERT INTO Студент (Фамилия, Имя, Отчество )
VALUES ("Захаров", "Андрей", "Васильевич");
```

1. Запросы на вставку данных в таблицы

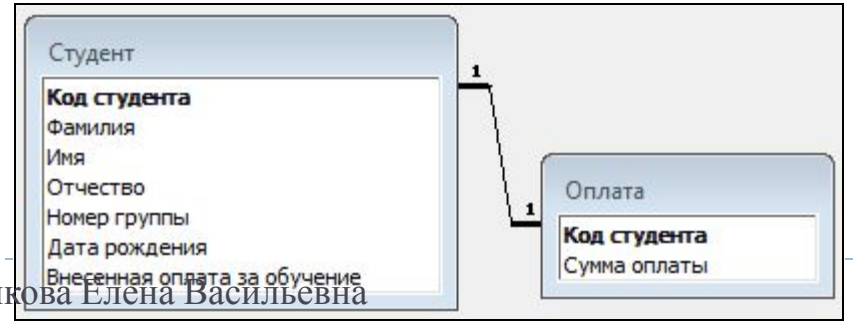
Есть второй вариант команды вставки, который позволяет вводить записи на основании запроса.

Пример INS2: добавить в таблицу Оплата коды тех студентов, которые есть в таблице Студент, но отсутствуют в таблице Оплата .

INSERT INTO Оплата ([Код студента])

SELECT [Код студента] FROM Студент WHERE [Код студента] NOT IN

(SELECT [Код студента] FROM Оплата);



2. Запросы на изменение данных в таблицах

Оператор UPDATE (используется для модификации записей, которые уже есть в таблице)

Синтаксис:

UPDATE <таблица> SET <поле1>=<значение1>, <поле2>=<значение2>, … [WHERE <условие>];

Если слово WHERE не указано, то оператор UPDATE будет применен ко всем записям таблицы

2. Запросы на изменение данных в таблицах

<u>Пример UP1</u>: У студента *Фукс Ивана Ивановича* сменить фамилию (на *Иванов*)

UPDATE Студент SET Фамилия = "Иванов"

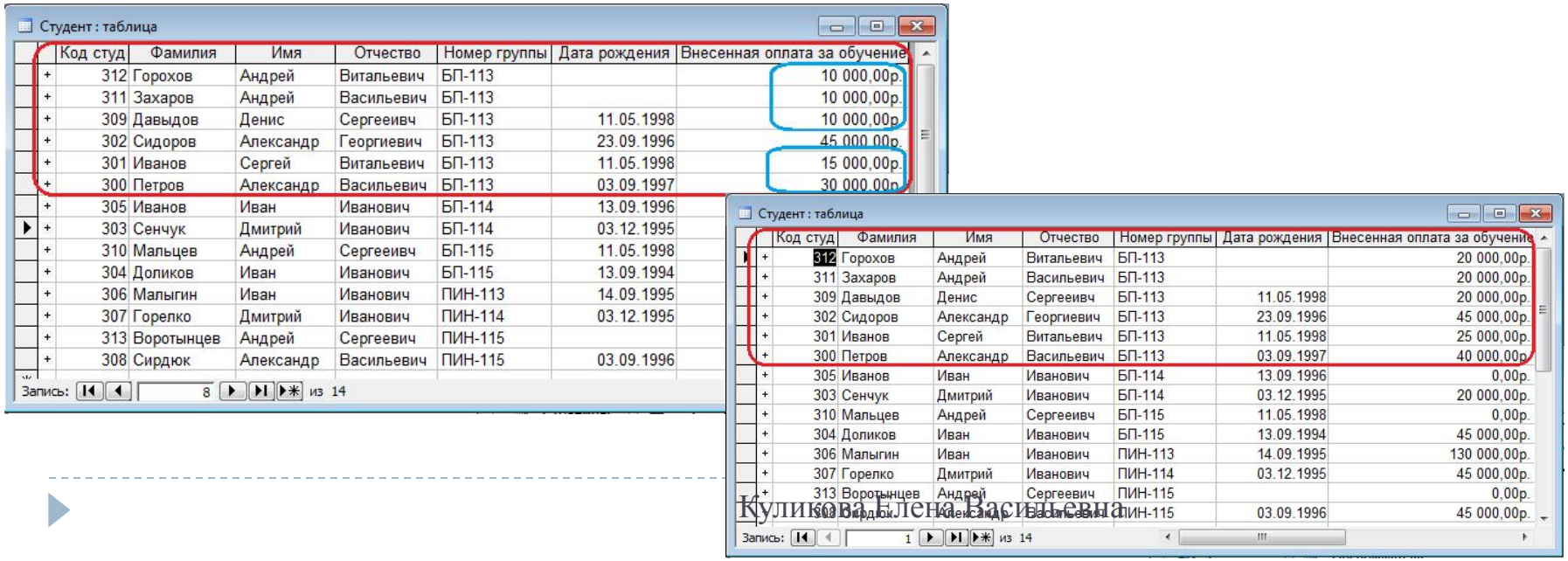
WHERE Фамилия="Фукс" AND Имя="Иван" AND Отчество="Иванович";

2. Запросы на изменение данных в таблицах

Пример UP2: изменить внесенную оплату за обучение у студентов группы БП-113 (добавить к существующей оплате 10000 рублей), не учитывая тех студентов, которые оплатили сумму в размере 45000 рублей

UPDATE Студент SET [Внесенная оплата за обучение] = [Внесенная оплата за обучение]+10000

WHERE [Номер группы]="БП-113" AND [Внесенная оплата за обучение]<>45000;



Студент : таблица

Код студ	Фамилия	Имя	Отчество	Номер группы	Дата рождения	Внесенная оплата за обучение
312	Горохов	Андрей	Витальевич	БП-113		10 000,00р.
311	Захаров	Андрей	Васильевич	БП-113		10 000,00р.
309	Давыдов	Денис	Сергеевич	БП-113	11.05.1998	10 000,00р.
302	Сидоров	Александр	Георгиевич	БП-113	23.09.1996	45 000,00р.
301	Иванов	Сергей	Витальевич	БП-113	11.05.1998	15 000,00р.
300	Петров	Александр	Васильевич	БП-113	03.09.1997	30 000,00р.
305	Иванов	Иван	Иванович	БП-114	13.09.1996	
303	Сенчук	Дмитрий	Иванович	БП-114	03.12.1995	
310	Мальцев	Андрей	Сергеевич	БП-115	11.05.1998	
304	Доликов	Иван	Иванович	БП-115	13.09.1994	
306	Малыгин	Иван	Иванович	ПИН-113	14.09.1995	
307	Горелко	Дмитрий	Иванович	ПИН-114	03.12.1995	
313	Воротынцев	Андрей	Сергеевич	ПИН-115		
308	Сирдюк	Александр	Васильевич	ПИН-115	03.09.1996	

Запись: 8 из 14

Студент : таблица

Код студ	Фамилия	Имя	Отчество	Номер группы	Дата рождения	Внесенная оплата за обучение
312	Горохов	Андрей	Витальевич	БП-113		20 000,00р.
311	Захаров	Андрей	Васильевич	БП-113		20 000,00р.
309	Давыдов	Денис	Сергеевич	БП-113	11.05.1998	20 000,00р.
302	Сидоров	Александр	Георгиевич	БП-113	23.09.1996	45 000,00р.
301	Иванов	Сергей	Витальевич	БП-113	11.05.1998	25 000,00р.
300	Петров	Александр	Васильевич	БП-113	03.09.1997	40 000,00р.
305	Иванов	Иван	Иванович	БП-114	13.09.1996	0,00р.
303	Сенчук	Дмитрий	Иванович	БП-114	03.12.1995	20 000,00р.
310	Мальцев	Андрей	Сергеевич	БП-115	11.05.1998	0,00р.
304	Доликов	Иван	Иванович	БП-115	13.09.1994	45 000,00р.
306	Малыгин	Иван	Иванович	ПИН-113	14.09.1995	130 000,00р.
307	Горелко	Дмитрий	Иванович	ПИН-114	03.12.1995	45 000,00р.
313	Воротынцев	Андрей	Сергеевич	ПИН-115		0,00р.
308	Сирдюк	Александр	Васильевич	ПИН-115	03.09.1996	45 000,00р.

Запись: 1 из 14

3. Запросы на удаление записей из таблицы

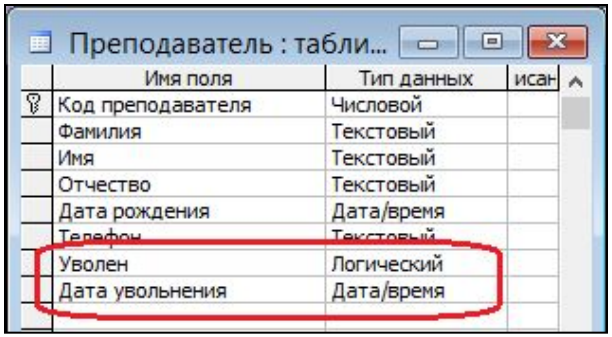
Оператор DELETE

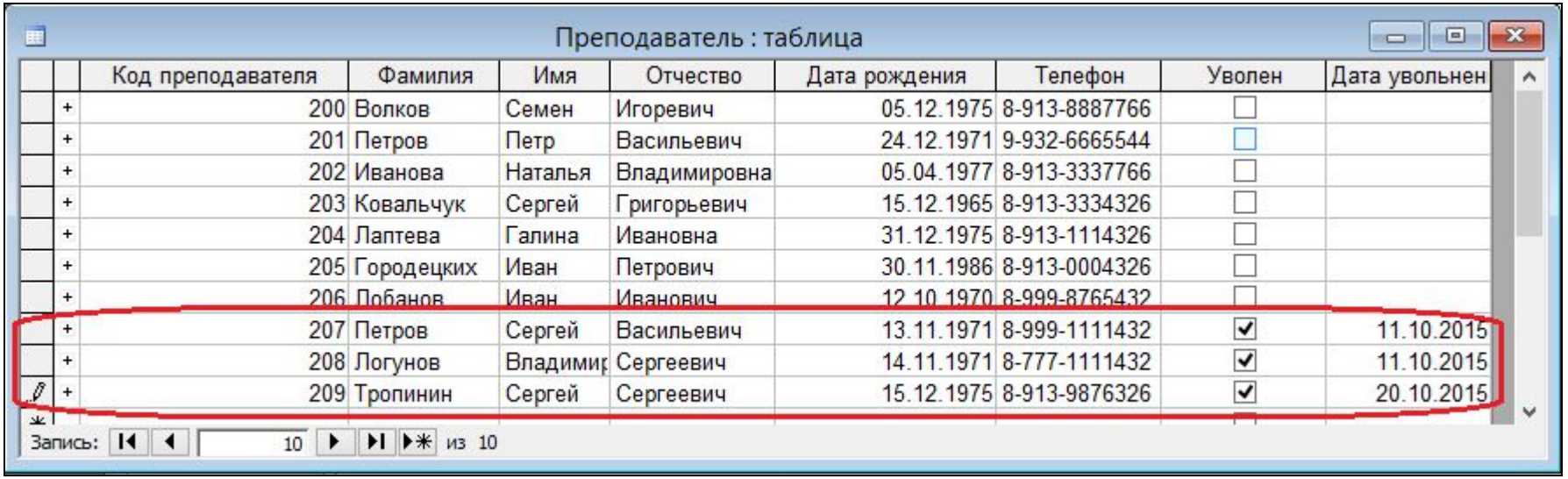
Синтаксис:

DELETE FROM <имя таблицы > [WHERE < условие >];

Если условие не задано, то из таблицы будут удалены все записи

Изменение структуры таблицы Преподаватель





3. Запросы на удаление записей из таблицы

Пример D1: удалить запись об уволенном преподавателе (Петрове)

DELETE * FROM Преподаватель WHERE Фамилия="Петров" AND Уволен=true;

Преподаватель : таблица

Код преподавателя	Фамилия	Имя	Отчество	Дата рождения	Телефон	Уволен	Дата увольнен
200	Волков	Семен	Игоревич	05.12.1975	8-913-8887766	☐	
201	Петров	Петр	Васильевич	24.12.1971	9-932-6665544	☐	
202	Иванова	Наталья	Владимировна	05.04.1977	8-913-3337766	☐	
203	Ковальчук	Сергей	Григорьевич	15.12.1965	8-913-3334326	☐	
204	Лаптева	Галина	Ивановна	31.12.1975	8-913-1114326	☐	
205	Городецких	Иван	Петрович	30.11.1986	8-913-0004326	☐	
206	Лобанов	Иван	Иванович	12.10.1970	8-999-8765432	☐	
207	Петров	Сергей	Васильевич	13.11.1971	8-999-1111432	☑	11.10.2015
208	Логунов	Владимир	Сергеевич	14.11.1971	8-777-1111432	☑	11.10.2015
209	Тропинин	Сергей	Сергеевич	15.12.1975	8-913-9876326	☑	20.10.2015

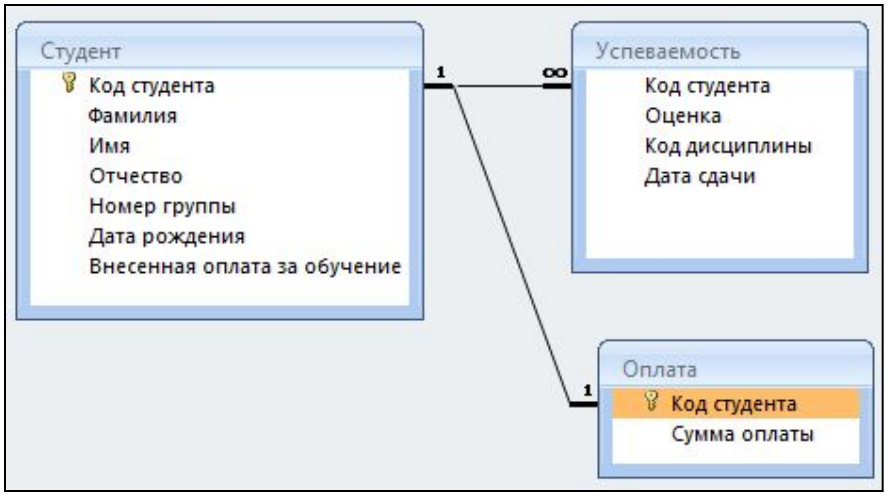
Запись: 10 из 10

3. Запросы на удаление записей из таблицы

Пример D2: удалить запись о студентах, которые будут отчислены (есть хотя бы одна двойка)

```
DELETE * FROM Студент
WHERE [Код студента]=ANY (SELECT [Код студента]
  FROM Успеваемость WHERE Оценка=2);
```

Удаление записей в родительской таблице повлечет за собой удаление связанных записей в дочерних таблицах



2.6. Перекрестные запросы

Перекрестные запросы относятся к **статистическим запросам**, когда **значения в строках таблицы становятся заголовками строк или столбцов**, на пересечениях таких строк и столбцов **значения формируются с помощью групповых операций**.

2.6. Перекрестные запросы, синтаксис

TRANSFORM статФункция
ИнструкцияSelect
PIVOT поле [IN (значение_1[, значение_2[, ...]])]

Описание параметров:

статФункция. Статистическая функция SQL (*Sum, Count* и т.д.), обрабатывающая данные, которые будут отображаться в области значений итоговой выборки (справа от заголовков строк и под строкой заголовков столбцов).

ИнструкцияSelect. Текст SQL-запроса на выборку. Может включать предложения *SELECT*, *FROM*, *GROUP BY* и т.д.

поле. Поле или выражение, возвращающее поле, которое содержит заголовки столбцов для итогового набора данных.

значение_1, значение_2. Фиксированные значения, используемые при создании заголовков столбцов.

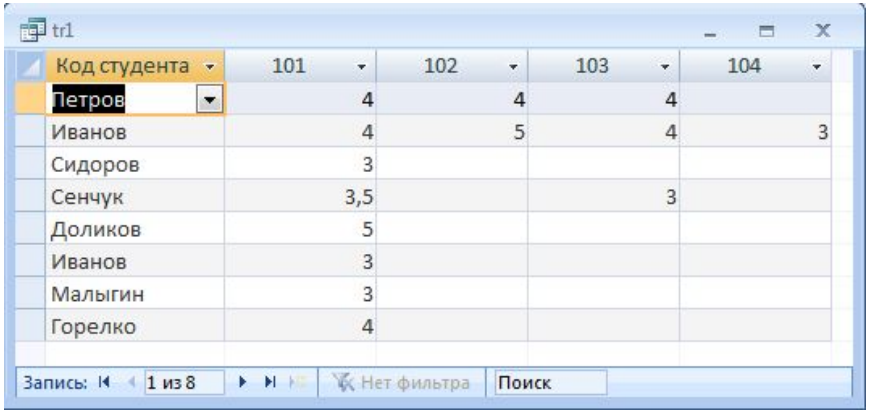
2.6. Перекрестные запросы, синтаксис

- В перекрестном запросе в инструкции ***SELECT***, перечисляются поля, являющиеся заголовками строк.
- В инструкции ***PIVOT*** указывается поле, значения которого будут служить в качестве заголовков столбцов.
- Инструкция ***TRANSFORM*** служит для указания поля, значения которого, после произведенной групповой операции (*Sum*, *Count* и т.д.), будут помещены справа от полей, перечисленных в инструкции ***SELECT***

	PIVOT	*PIVOT*	*PIVOT*
SELECT	*TRANSFORM*	*TRANSFORM*	…
SELECT	*TRANSFORM*	*TRANSFORM*	…
SELECT	*TRANSFORM*	*TRANSFORM*	…
SELECT	*TRANSFORM*	*TRANSFORM*	…
SELECT	…	…	…

2.6. Перекрестные запросы

Пример TR1: вывести для каждого студента средний балл по каждой дисциплине , представив в виде:



tr1

Код студента	101	102	103	104
Петров	4	4	4	
Иванов	4	5	4	3
Сидоров	3			
Сенчук	3,5		3	
Доликов	5			
Иванов	3			
Малыгин	3			
Горелко	4			

Запись: 1 из 8 Нет фильтра Поиск

TRANSFORM AVG(Оценка) AS [Средний балл по дисциплине]

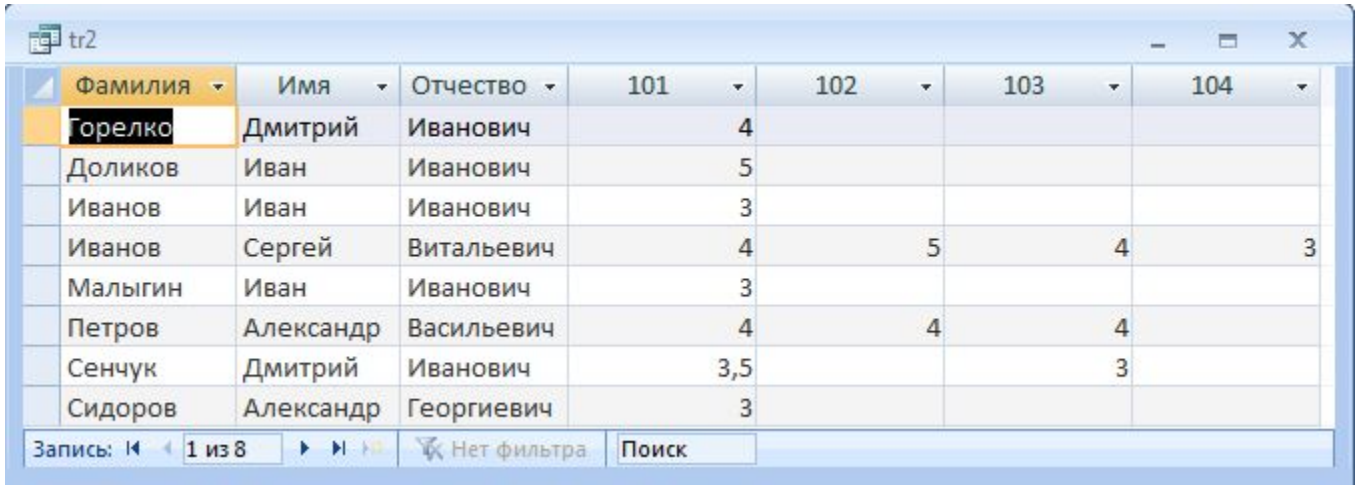
SELECT [Код студента] FROM Успеваемость

GROUP BY [Код студента]

PIVOT [Код дисциплины];

2.6. Перекрестные запросы

Пример TR2: вывести для каждого студента средний балл по каждой дисциплине , представив в виде:



tr2

Фамилия	Имя	Отчество	101	102	103	104
Горелко	Дмитрий	Иванович	4			
Доликов	Иван	Иванович	5			
Иванов	Иван	Иванович	3			
Иванов	Сергей	Витальевич	4	5	4	3
Малыгин	Иван	Иванович	3			
Петров	Александр	Васильевич	4	4	4	
Сенчук	Дмитрий	Иванович	3,5		3	
Сидоров	Александр	Георгиевич	3			

Запись: 1 из 8 Нет фильтра Поиск

```
TRANSFORM AVG(Оценка) AS [Средний балл по дисциплине]
SELECT Студент.Фамилия, Студент.Имя, Студент.Отчество
FROM Студент INNER JOIN Успеваемость ON Студент.[Код студента]=Успеваемость.[Код студента]
GROUP BY Студент.Фамилия, Студент.Имя, Студент.Отчество  PIVOT Успеваемость.[Код дисциплины];
```

3. СОЗДАНИЕ БД на SQL. Язык описания данных (Data Definition Language, DDL)

Ключевое слово	Использование
CREATE	Служит для создания индекса или таблицы, если они еще не существуют.
ALTER	Изменяет существующую таблицу или столбец.
DROP	Удаляет существующую таблицу, столбец или ограничение.
ADD	Добавляет в таблицу столбец или ограничение.
COLUMN	Используется вместе с ключевым словом ADD, ALTER или DROP.
CONSTRAINT	Используется вместе с ключевым словом ADD, ALTER или DROP.
INDEX	Используется вместе с ключевым словом CREATE.
TABLE	Используется вместе с ключевым словом ALTER, CREATE или DROP

3.1. Запросы на создание таблиц базы данных

Синтаксис:

CREATE TABLE <имя таблицы >

(<имя поля> <тип /домен> [описание /ограничения столбца],...,[ограничения таблицы])

Тип может быть любой из типов SQL-сервера или вместо типа используют имя домена.

Типы данных

Типы данных ANSI SQL	Типы данных Microsoft Jet SQL	Описание	Тип данных Microsoft SQLServer
BIT, BIT VARYING	**BINARY**		BINARY, VARBINARY
DATE, TIME, TIMESTAMP	**DATETIME**		DATETIME
DECIMAL	**DECIMAL**		DECIMAL
REAL	**REAL**		REAL
DOUBLE PRECISION, FLOAT	**FLOAT**		FLOAT
SMALLINT	**SMALLINT**		SMALLINT
INTEGER	**INTEGER**		INTEGER
CHARACTER, CHARACTER VARYING, NATIONAL CHARACTER, NATIONAL CHARACTER VARYING	**CHAR**		CHAR, VARCHAR, NCHAR, NVARCHAR

Ограничения таблицы

Описание каждого поля может включать следующие конструкции:

- *DEFAULT* – конструкция, определяющая значение поля по умолчанию;
- *NOT NULL* – конструкция, указывающая на то, что поле не может быть пустым;
- *COLLATE* – предложение, определяющее порядок сортировки для выбранного набора символов. Например, в INTERBASE русский набор символов WIN1251 имеет два порядка сортировки – WIN1251 и PXW_CYRL. Для правильной сортировки, включающей большие буквы, следует выбрать порядок PXW_CYRL.

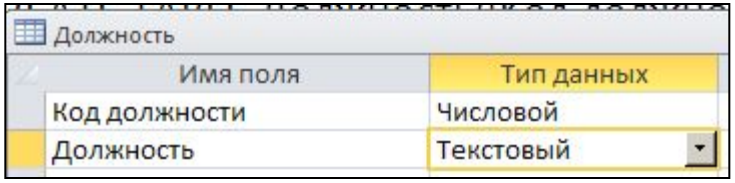
Описание ограничений включает в себя предложения *CONSTRAINT* или предложения:

- *PRIMARY KEY | UNIQUE (список полей)* – определяет создание индексов (первичного или просто уникального);
- *FOREIGN KEY (список полей связи) REFERENCES <имя таблицы связи> (список полей первичного ключа в таблице связи)* – определяет связи между двумя таблицами;
- *CHECK (<предикат>)* определяет другие ограничения с использованием предикатов сравнения *BETWEEN, LIKE, IN* и других.

3.1. Запросы на создание таблиц базы данных

Пример CrT1: создать таблицу Должность, содержащую поля: *Код должности* (целое), *Должность* (текстовый, 30 символов)

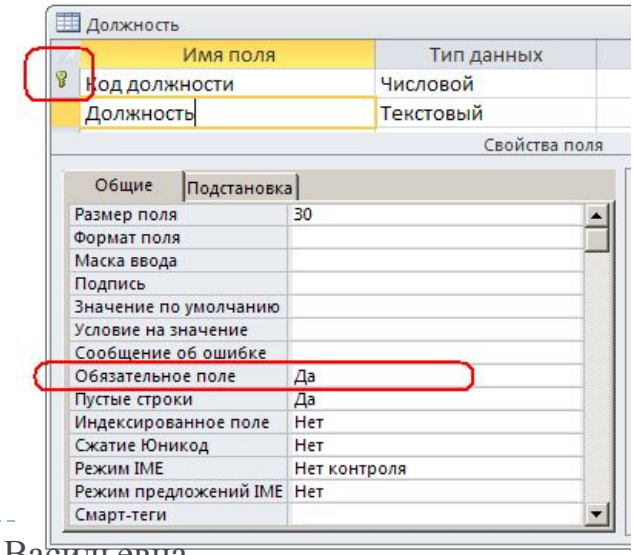
```
CREATE TABLE Должность
(
[Код должности] INTEGER,
Должность CHAR (30)
);
```



3.1. Запросы на создание таблиц базы данных

Пример CrT2: создать таблицу Должность, содержащую поля: *Код должности* (целое, ключ), *Должность* (текстовый, 30 символов, обязательное поле)

```
CREATE TABLE Должность
(
[Код должности] INTEGER PRIMARY KEY,
Должность CHAR(30) NOT NULL
);
```



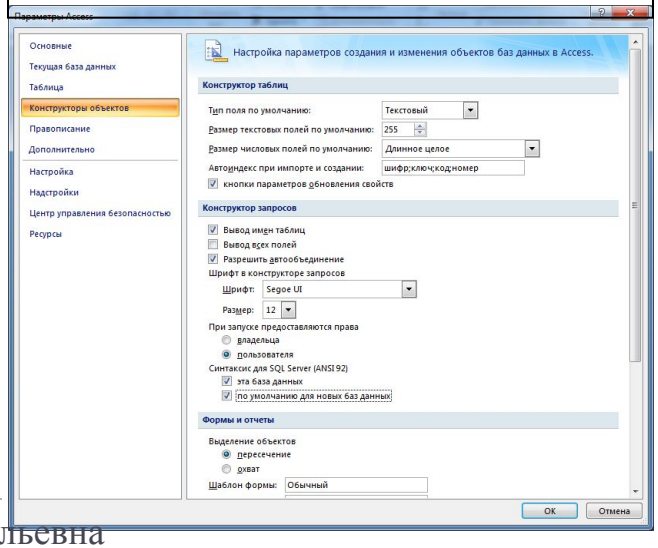
3.1. Запросы на создание таблиц базы данных

Пример СrТЗ: создать таблицу Сотрудник (*КодСотрудника , Фамилия , Пол, Возраст*), определив значение по умолчанию (пол “ж”), ограничение по возрасту (не менее 18 лет)

```
CREATE TABLE Сотрудник
(
КодСотрудника INTEGER PRIMARY KEY,
Фамилия CHAR(30) NOT NULL,
Пол CHAR(1) NOT NULL DEFAULT “Ж”,
Возраст INTEGER NOT NULL,
CHECK (Возраст >=18)
);
```

Перед выполнением запроса в параметрах MS Access необходимо включить Синтаксис для SQL Server (ANSI 92)

Ограничение целостности ***check*** *позволяет задать для определённой колонки выражение, которое будет осуществлять проверку, помещаемого в эту колонку значения.*



3.1. Запросы на создание таблиц базы данных

Пример CrT3 (измененный): создать таблицу Сотрудник (*КодСотрудника , Фамилия , Пол, Возраст*), определив значение по умолчанию (пол “ж”), ограничение по возрасту (не менее 18 лет)

```
CREATE TABLE Сотрудник
(
КодСотрудника INTEGER PRIMARY KEY,
Фамилия CHAR(30) NOT NULL,
Пол CHAR(1) NOT NULL DEFAULT “Ж”,
Возраст INTEGER NOT NULL,
CONSTRAINT VOZRAST
CHECK (Возраст >=18)
);
```

Можно дать ограничению конкретное **имя**. Это сделает более понятными сообщения об ошибках и позволит ссылаться на это ограничение, когда понадобится его изменить.

VOZRAST – *имя ограничения*

3.2. Запросы на удаление таблиц базы данных

Синтаксис:

DROP TABLE <имя таблицы>;

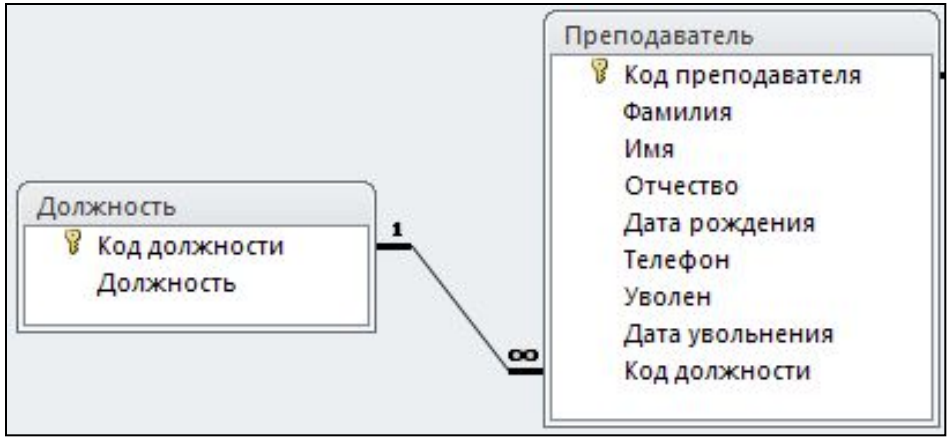
3.2. Запросы на удаление таблиц базы данных

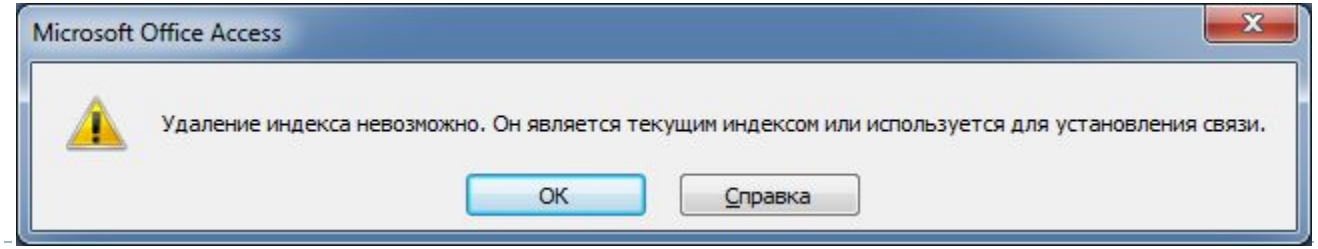
Пример DR1: удалить таблицу Должность

DROP TABLE Должность;

Удаление связанной таблицы таким запросом невозможно:

См. Пример AT8





3.3. Запросы на модификацию таблиц

Оператор **ALTER TABLE** позволяет модифицировать все то, что указывается при выполнении оператора **CREATE TABLE**

Синтаксис:

ALTER TABLE <имя таблицы> предикат

где *предикат* может принимать одно из указанных ниже значений.

- **ADD COLUMN** *имя тип поля*[(*размер*)] [NOT NULL] [CONSTRAINT *ограничение*]
- **ADD CONSTRAINT** *ограничение_нескольких_полей*
- **ALTER COLUMN** *поле тип поля*[(*размер*)]
- **DROP COLUMN** *поле*
- **DROP CONSTRAINT** *ограничение*

3.3. Запросы на модификацию таблиц

Пример АТ1: добавить в таблицу Преподаватель поля Адрес (текстовый, 100 символов), Email (текстовый, 30 символов)

ALTER TABLE Преподаватель ADD COLUMN Адрес CHAR(100), Email CHAR(30);

3.3. Запросы на модификацию таблиц

Пример АТ3: удалить из таблицы Преподаватель поле Email

```
ALTER TABLE Преподаватель DROP COLUMN
Email;
```

3.3. Запросы на модификацию таблиц

Пример АТ4: создать ограничение поля *Фамилия* таблицы *Преподаватель (ограничение: обязательное поле)*

ALTER TABLE Преподаватель ALTER COLUMN Фамилия CHAR

CONSTRAINT FamNotNull NOT NULL;

FamNotNull *– имя ограничения*

3.3. Запросы на модификацию таблиц

Пример АТ5: создать ограничение поля *Код должности* таблицы *Должность (ограничение: Primary Key)*

ALTER TABLE Должность

ALTER COLUMN [Код должности] INTEGER

CONSTRAINT КодДолжности PRIMARY KEY;

КодДолжности *– имя ограничения*

3.3. Запросы на модификацию таблиц

Пример АТ6: создать связь таблиц Должность и Преподаватель по полю Код должности

```
ALTER TABLE Преподаватель
ADD CONSTRAINT ПреподавательДолжность
FOREIGN KEY ([Код должности])
REFERENCES Должность ([Код должности]);
```

ПреподавательДолжность – имя ограничения

3.3. Запросы на модификацию таблиц

Пример АТ8: удалить связь таблиц Должность и Преподаватель

ALTER TABLE Преподаватель

DROP CONSTRAINT ПреподавательДолжность;

ПреподавательДолжность *– имя ограничения*