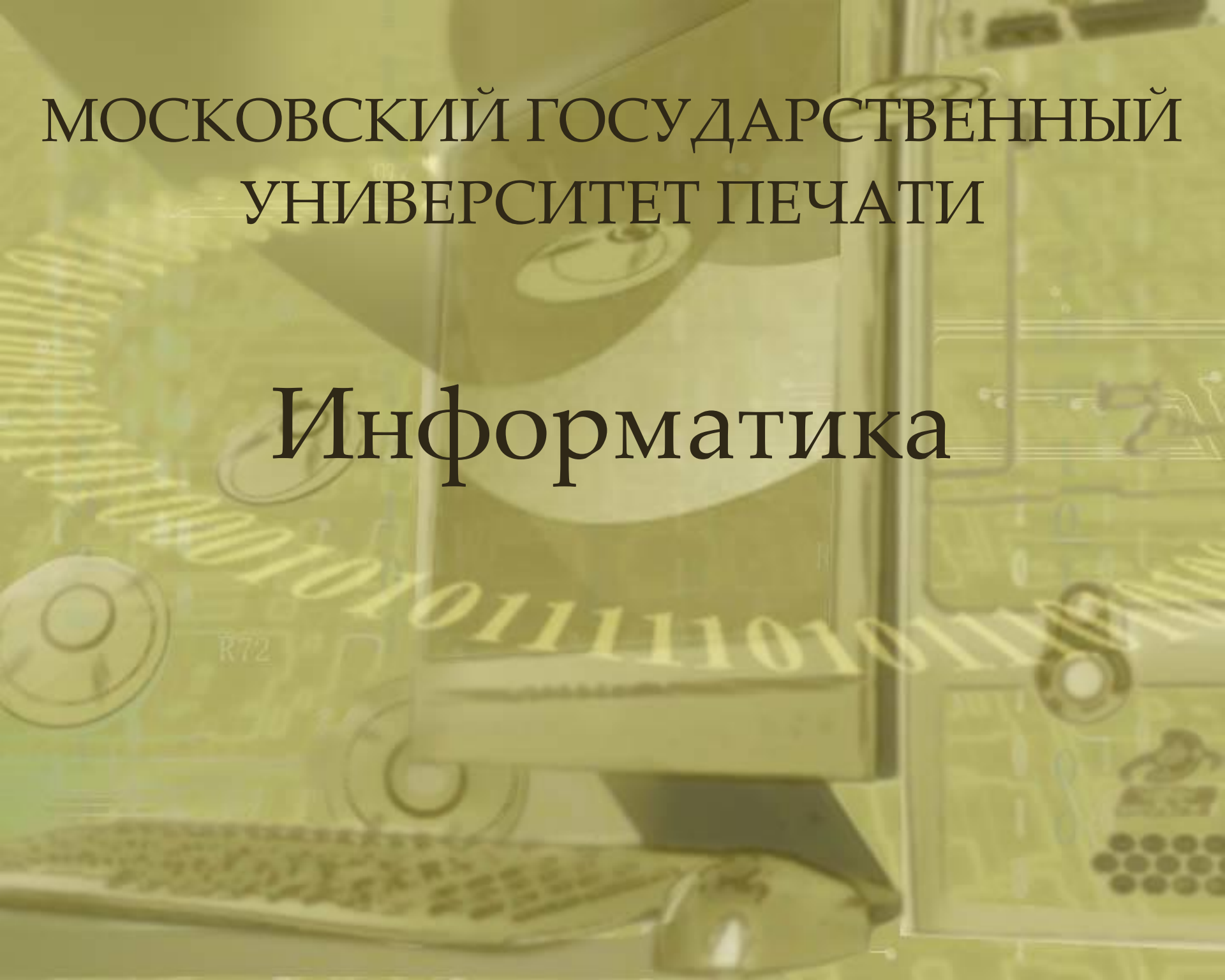
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПЕЧАТИ

ЗАОЧНАЯ ФОРМА ОБУЧЕНИЯ



МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ ПЕЧАТИ

Информатика

ЛЕКЦИЯ

Базы данных

Лектор

Попов Дмитрий Иванович

доктор технических наук,
заведующий кафедрой Информатика
и вычислительная техника МГУП,
директор Института открытого
образования МГУП

План лекции

1. Язык запросов SQL
2. Использование SQL для выборки данных из таблиц. Оператор SELECT.
3. Операции сравнения, логические и арифметические операции
4. Использование специальных операторов IN, BETWEEN, LIKE, NULL, NOT
5. Пример создания простой БД в СУБД MS-ACCESS-2002

Язык запросов SQL

SQL (Structured Query Language – структурированный язык запросов) предназначен для обработки реляционных БД. Многие конструкции и операторы языка SQL связаны с операциями реляционной алгебры. С этой точки зрения методы реляционной алгебры являются базовыми для более глубокого понимания и использования языка SQL.

В различных СУБД используются два типа языка SQL:

В различных СУБД используются два типа языка SQL:

1. Интерактивный SQL – используется для выполнения действий непосредственно над БД в оперативном режиме. Как правило, интерактивному SQL сопутствует некоторая программа-сервер SQL. Наиболее популярны в настоящее время Oracle SQL-Server, MS-SQL Server, My-SQL, Inter Base и некоторые другие

В различных СУБД используются два типа языка SQL:

1. Интерактивный SQL – используется для выполнения действий непосредственно над БД в оперативном режиме. Как правило, интерактивному SQL сопутствует некоторая программа-сервер SQL. Наиболее популярны в настоящее время Oracle SQL-Server, MS-SQL Server, My-SQL, Inter Base и некоторые другие
2. Встроенный SQL – состоит из команд SQL, включенных непосредственно в программы, которые написаны на другом языке программирования. Например, в таких системах программирования, как Delphi, C++ Builder, VBasic, VC++ имеется встроенный SQL. В различных СУБД, например в MS-Access, также имеются возможности использовать команды встроенного SQL

Можно разделить SQL на определенные подмножества:

Можно разделить SQL на определенные подмножества:

DQL – язык запросов. Команды предназначены для извлечения данных из таблиц

Можно разделить SQL на определенные подмножества:

DQL – язык запросов. Команды предназначены для извлечения данных из таблиц

DML – язык манипуляция данными

Можно разделить SQL на определенные подмножества:

DQL – язык запросов. Команды предназначены для извлечения данных из таблиц

DML – язык манипуляция данными

TPL – язык обработки транзакций. Команды позволяют объединить команды языка DML в группы транзакций. Если одна из команд не может быть выполнена, то отменяются все предыдущие команды из этой же транзакции, – происходит «откат транзакции»

Можно разделить SQL на определенные подмножества:

DQL – язык запросов. Команды предназначены для извлечения данных из таблиц

DML – язык манипуляция данными

TPL – язык обработки транзакций. Команды позволяют объединить команды языка DML в группы транзакций. Если одна из команд не может быть выполнена, то отменяются все предыдущие команды из этой же транзакции, – происходит «откат транзакции»

DDL – язык определение данных. Включает в себя также инструкции обеспечения целостности данных. Например, команды создания таблиц и организации связей между ними

Можно разделить SQL на определенные подмножества:

DQL – язык запросов. Команды предназначены для извлечения данных из таблиц

DML – язык манипуляция данными

TPL – язык обработки транзакций. Команды позволяют объединить команды языка DML в группы транзакций. Если одна из команд не может быть выполнена, то отменяются все предыдущие команды из этой же транзакции, – происходит «откат транзакции»

DDL – язык определение данных. Включает в себя также инструкции обеспечения целостности данных. Например, команды создания таблиц и организации связей между ними

CCL – язык управления курсором. Позволяет выбрать для обработки одну строку из результирующего множества запросов

Можно разделить SQL на определенные подмножества:

DQL – язык запросов. Команды предназначены для извлечения данных из таблиц

DML – язык манипуляция данными

TPL – язык обработки транзакций. Команды позволяют объединить команды языка DML в группы транзакций. Если одна из команд не может быть выполнена, то отменяются все предыдущие команды из этой же транзакции, – происходит «откат транзакции»

DDL – язык определение данных. Включает в себя также инструкции обеспечения целостности данных. Например, команды создания таблиц и организации связей между ними

CCL – язык управления курсором. Позволяет выбрать для обработки одну строку из результирующего множества запросов

DCL – язык управления данными. Содержит инструкции с помощью которых выполняется присваивание прав доступа к БД, множеству таблиц или представлений

Использование SQL для выборки данных из таблиц. Оператор SELECT

Запрос – это команда, которая выдается программе базы данных для поиска определенной информации из таблиц. Эта информация обычно посылается непосредственно на экран компьютера или терминала, которым вы пользуетесь, хотя в большинстве случаев, ее можно также послать на принтер, сохранить в файле (как объект в памяти компьютера) или представить как вводную информацию для другой команды или процесса.

Запросы обычно рассматриваются как часть языка DML. Однако, так как запрос не меняет информацию в таблицах, а просто показывает ее пользователю, будем рассматривать запросы как самостоятельную категорию среди команд DML, которые производят действие, а не просто показывают содержание базы данных.

Все запросы в SQL состоят из одиночной команды. Эта команда называется SELECT (выбор).

Например, для того чтобы вывести на экран таблицу Студенты достаточно использовать такую команду:

```
SELECT  StudNo, Имя, Фамилия, Адрес, Телефон
FROM Студенты;
```

Запрос может быть сформирован в несколько строк или же в одну.

В списке полей можно использовать символ * для указания, что необходимо выводить все поля данной таблицы.

Другими словами, можно составить следующий запрос, полностью аналогичный предыдущему:

```
SELECT * FROM Студенты;
```

Общий вид оператора SELECT

```
SELECT  [ DISTINCT | ALL ] <выражение или имя поля>, ...
  FROM <ссылка на таблицу>, ...
  [WHERE <предикат>]
  [GROUP BY <столбец, по которому выполняется группировка>, ...]
  [HAVING <предикат>]
  [ORDER BY <столбец, по которому выполняется упорядочивание>
                                    [ ASC | DESC ], ... ]
```

Команда извлекает только те строки из таблицы для, которых такое утверждение верно.

Например, если необходимо выбрать информацию по студенту Петров, то можно использовать такой запрос:

```
SELECT Имя, Фамилия,  Телефон, СтудNo
FROM Студенты
WHERE Фамилия = ‘Петров’;
```

Заметим, что если поле не является текстовым, то в операции сравнения кавычки указывать не нужно.

Например, выберем студентов, стипендия которых больше 1400 рублей:

```
SELECT Имя, Фамилия, Телефон
FROM Студенты
WHERE Стипендия > 1400;
```

Операции сравнения, логические и арифметические операции

Логические и другие операции в SQL

Операция	Назначение
>	Сравнивает два значения и возвращает «истину», если значение слева от оператора больше значения справа
<	Сравнивает два значения и возвращает «истину», если левое значение меньше значения справа
>=	Сравнивает два значения и возвращает «истину», если левое значение больше или равно значения справа
<=	Сравнивает два значения и возвращает «истину», если левое значение меньше или равно значения справа
=	Сравнивает два значения и возвращает «истину», если левое значение равно значения справа
<>	Сравнивает два значения и возвращает «истину», если левое значение не равно значения справа
AND	Логическая операция И (конъюнкция)
OR	Логическая операция ИЛИ (дизъюнкция)
NOT	Логическая операция НЕ (отрицание)
+	Сложение двух чисел
-	Вычитание
*	Умножение
/	Деление
&	Конкатенация (объединение двух строк или двух любых разнотипных значений)

Пример.

Пусть необходимо найти жилье в Пензе стоимостью не более 400 рублей в сутки.

```
Select Улица, Город, Плата
From Жилье
Where Город=’Пенза’ AND Плата<=400;
```

Для формирования более сложных предикатов можно использовать скобки, определяющие порядок вычисления значения предиката.

Предположим, нужно вывести цену за аренду жилья и рядом налог на имущество, выплачиваемый за это жилье. Результат выдать отсортированный по полям «город» и «адрес». Пусть налог на имущество за месяц представляет собой фиксированную сумму, равную 40 рублям плюс 13% от стоимости аренды:

```
SELECT Жилье.ЖNo, Жилье.Город, Жилье.Улица, Жилье.Плата,
                '40 руб. + 13%' AS СтавкаНалога,
                (40+ Жилье.Плата*13/100)&'рублей' AS Налог
FROM Жилье
ORDER BY Город, Улица;
```

Ключевые слова ORDER BY используются для окончательной сортировки, полученных данных. Указываются столбцы, по которым нужно сортировать, при этом ключевое слово ASC означает сортировку по возрастанию, DESC – сортировку по убыванию.

Результат выполнения запроса

ЖNo	Город	Улица	Плата	Ставка Налога	Налог
ЖА14	Астрахань	ул.Н-16	650,00р.	40 руб. + 13%	124,5рублей
ЖЛ94	Москва	ул.А-6	400,00р.	40 руб. + 13%	92рублей
ЖЖ21	Пенза	ул.D-18	600,00р.	40 руб. + 13%	118рублей
ЖЖ4	Пенза	ул.L-6	350,00р.	40 руб. + 13%	85,5рублей
ЖЖ16	Пенза	ул.N-5	450,00р.	40 руб. + 13%	98,5рублей
ЖЖ36	Пенза	ул.М-2	375,00р.	40 руб. + 13%	88,75рублей

Пример.

Выдать список сотрудников мужского пола старше 55 лет и женского пола старше 45 лет. В данном примере будем использовать дополнительную функцию Date ( ), которая возвращает текущую дату. Аналог такой функции присутствует в различных реализациях СУБД, но не рассматривается стандартом SQL:

```
Select Имя, Фамилия, Дрожд
From Сотрудники
Where (Пол=’М’ AND (Date()- Дрожд)>55) OR
(Пол=’Ж’ AND (Date()- Дрожд)>45);
```

Использование специальных операторов IN, BETWEEN, LIKE, NULL, NOT

Для определения более сложных выражений в предикатах с одной стороны и упрощения внешнего вида предикатов – с другой в SQL используются различные дополнительные операторы.

Специальные операторы

Оператор, пример	Назначение
<поле> IN (знач.1, знач.2, …) Пример. SELECT * FROM Сотрудники WHERE Фno IN ('Ф3', 'Ф7');	Определяет набор значений (множество), в которое данное значение может или не может быть включено. Обычно проверяется, принадлежит ли значение некоторого поля указанному множеству
<поле> BETWEEN <MinЗнач> AND <MaxЗнач> Пример. SELECT * FROM Сотрудники WHERE ЗП BETWEEN 9000 AND 20000;	Определяет диапазон, значения которого должны увеличиваться от возможного минимального значения (<MinЗнач>) до максимального (<MaxЗнач>) и возвращает истину, если проверяемое поле принадлежит данному диапазону. Значение может быть как числовым, так и символьным
<поле> LIKE <шаблон> Пример. SELECT * FROM Сотрудники WHERE Фамилия LIKE 'Б%';	Проверяет, совпадает ли значение проверяемого поля с шаблоном. В качестве метасимволов шаблонов используются: символ подчеркивания (_), который замещает любой одиночный символ; символ процента (%) замещает последовательность любого количества символов

Замечания:

Замечания:

1. Оператор BETWEEN включает в результат граничные значения, поэтому, для того чтобы исключить граничные значения, можно использовать оператор IN. Например, необходимо выдать список всех сотрудников, зарплата которых больше 9000, но меньше 20000, т.е. граничные значения не включаются в результат. Тогда запрос может быть таким:

```
SELECT * FROM Сотрудники
WHERE (ЗП BETWEEN 9000 AND 20000) AND NOT ЗП IN (9000,20000);
```

Замечания:

2. Если в BETWEEN используются в качестве аргументов строки, то происходит отбор значений по диапазону ASCII-кодов символов. Например, если необходимо выбрать сотрудников, чьи фамилии находятся в алфавитном диапазоне от А до *Л*, то можно использовать такой запрос:

```
SELECT * FROM Сотрудники
WHERE Фамилия BETWEEN  ‘А’ AND ‘Л’;
```

Замечания:

3. Оператор LIKE применим только к символьным данным (типа CHAR).

Замечания:

3. Оператор LIKE применим только к символьным данным (типа CHAR).

4. В некоторых реализациях SQL (например, в СУБД MS-Access) вместо метасимвола ‘%’ используется метасимвол ‘*’.

Часто в таблицах существуют записи, которые не имеют никаких значений для каждого поля, например, потому, что информация не завершена, или потому, что это поле просто не заполнялось. SQL учитывает такой вариант, позволяя вводить значение NULL (ПУСТОЙ) в поле вместо значения.

Когда значение поля равно NULL, это означает, что программа базы данных специально промаркировала это поле как не имеющее никакого значения для этой строки (или записи).

Следует учитывать, что NOT (неверное) равняется «верно», NOT (неизвестное) равняется «неизвестно». Следовательно, выражение типа 'Фамилия = NULL' или 'Фамилия IN (NULL)' будет неизвестно, независимо от значения поля «Фамилия».

Найдем все записи в таблице Съемщик с NULL значениями в столбце «Фамилия»:

```
SELECT * FROM Съемщик WHERE Фамилия IS NULL;
```

Здесь не будет никакого вывода, потому что в таблице нет никаких значений NULL.

Специальные операторы IN, BETWEEN, LIKE, а также NULL могут использоваться совместно с булевым оператором NOT – отрицание, инверсия.

Пример.

Если мы хотим устранить NULL из нашего вывода, мы будем использовать NOT, чтобы изменить на противоположное значение предиката:

```
SELECT * FROM Съемщик WHERE Фамилия IS NOT NULL;
```

Другие примеры использования NOT:

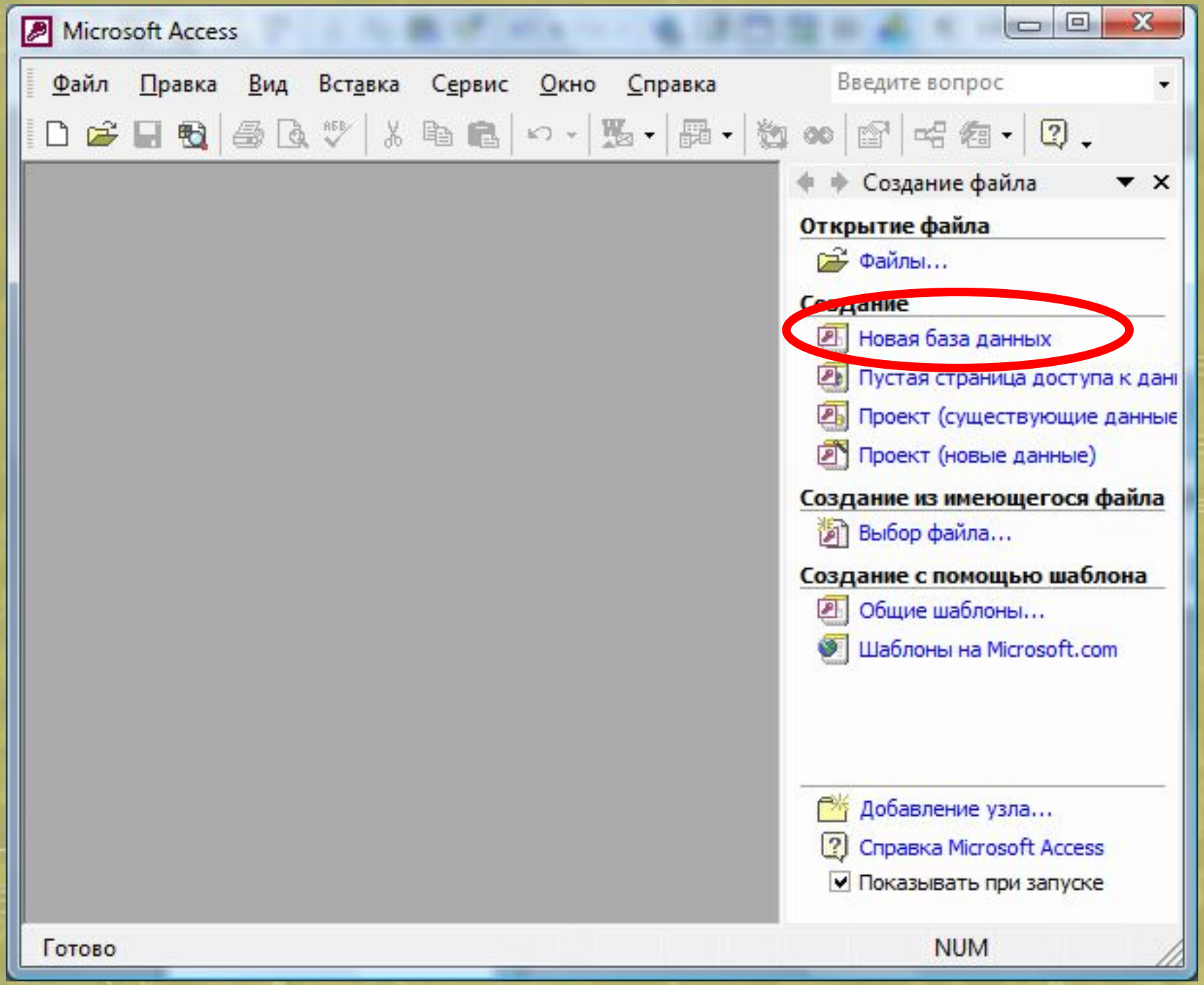
```
SELECT * FROM Жилье
WHERE Город NOT IN ( ' Астрахань ', 'Москва' );

SELECT * FROM Жилье
WHERE NOT Город IN ( 'Астрахань', 'Москва' );
```

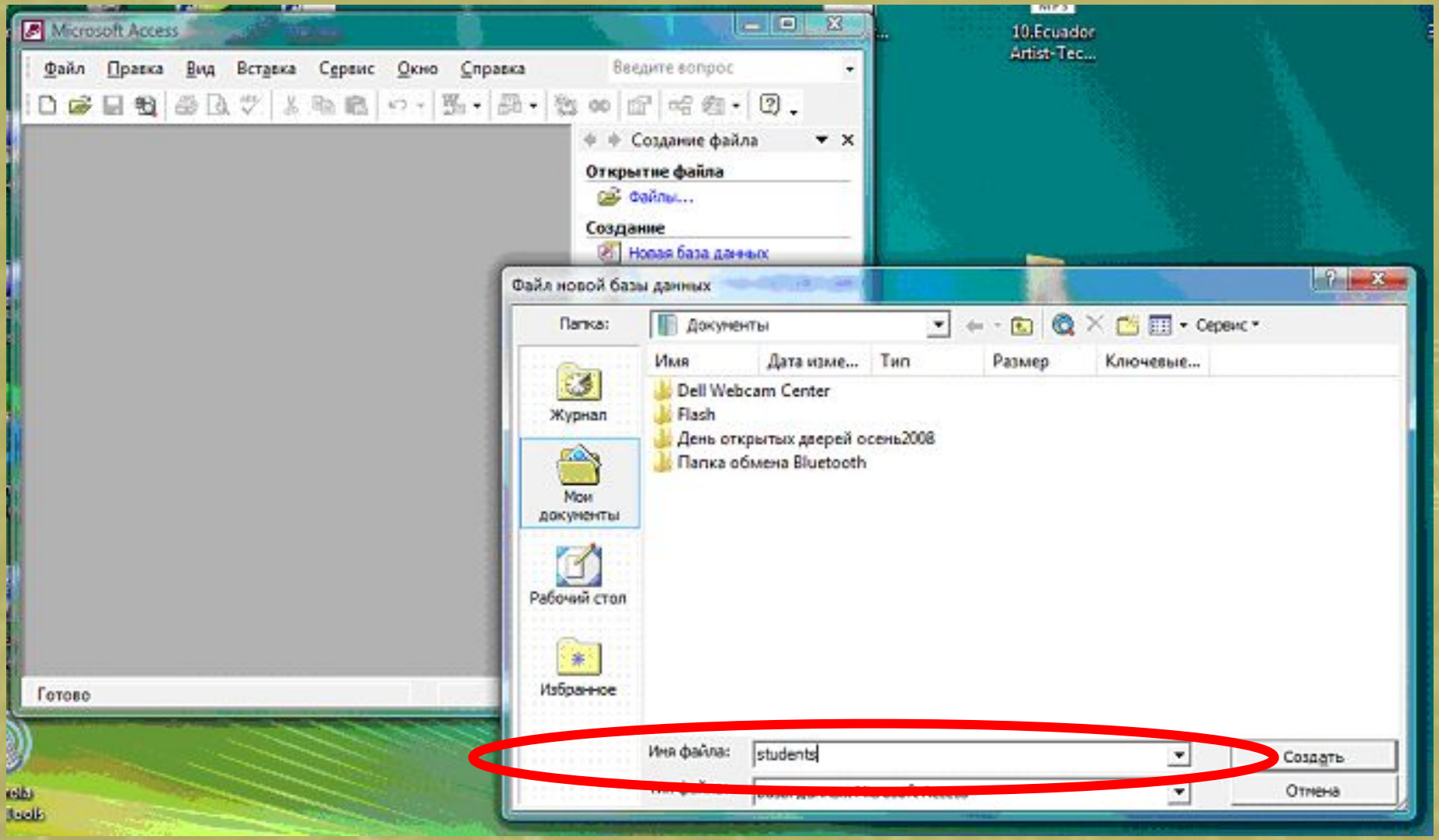
Пример создания простой БД в СУБД MS-ACCESS-2002

Задача.

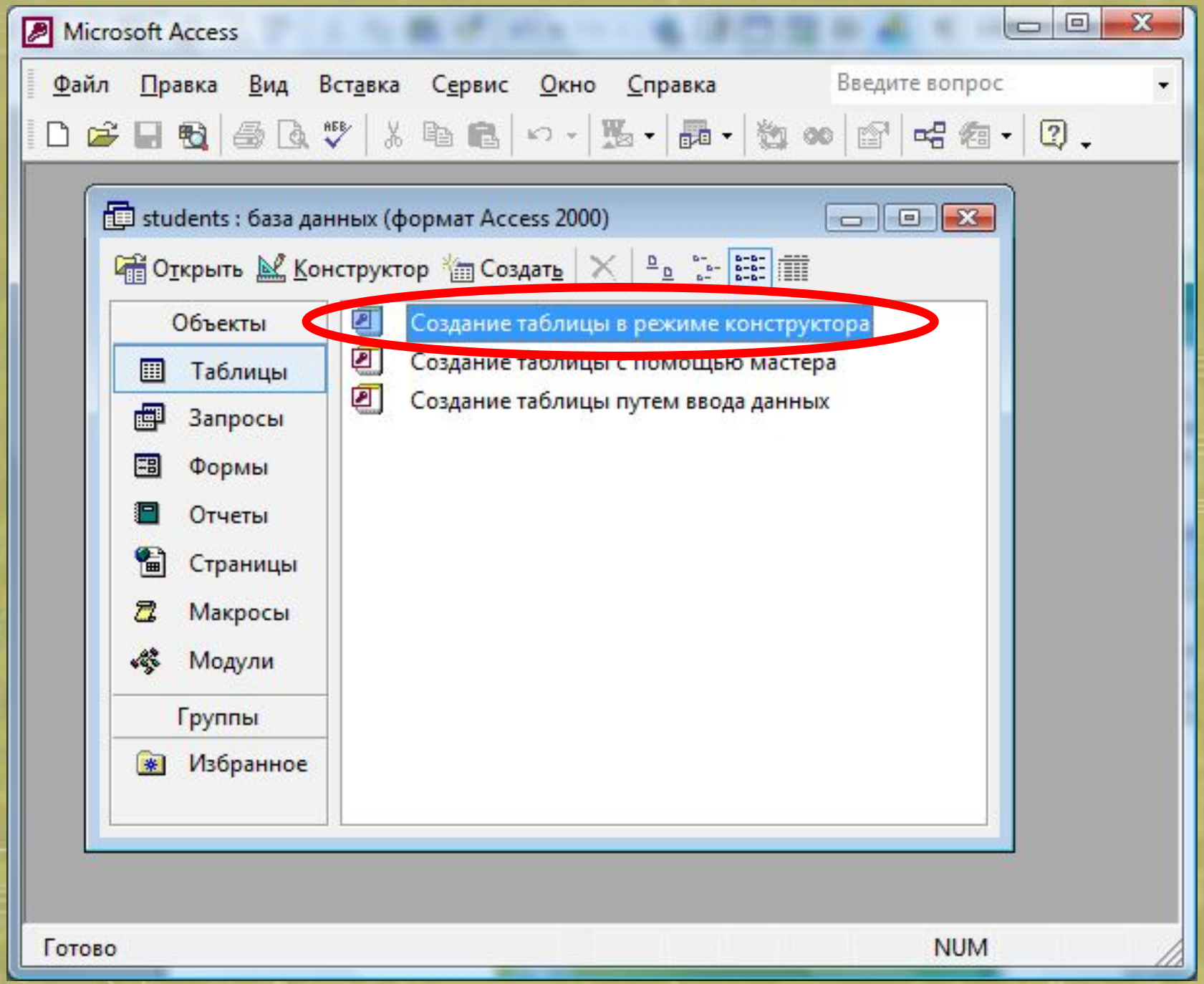
Создадим БД, имеющую 2 таблицы, 1 форму для ввода, 1 запрос и 1 отчет. БД описывает студентов учебного учреждения и их распределение по группам.



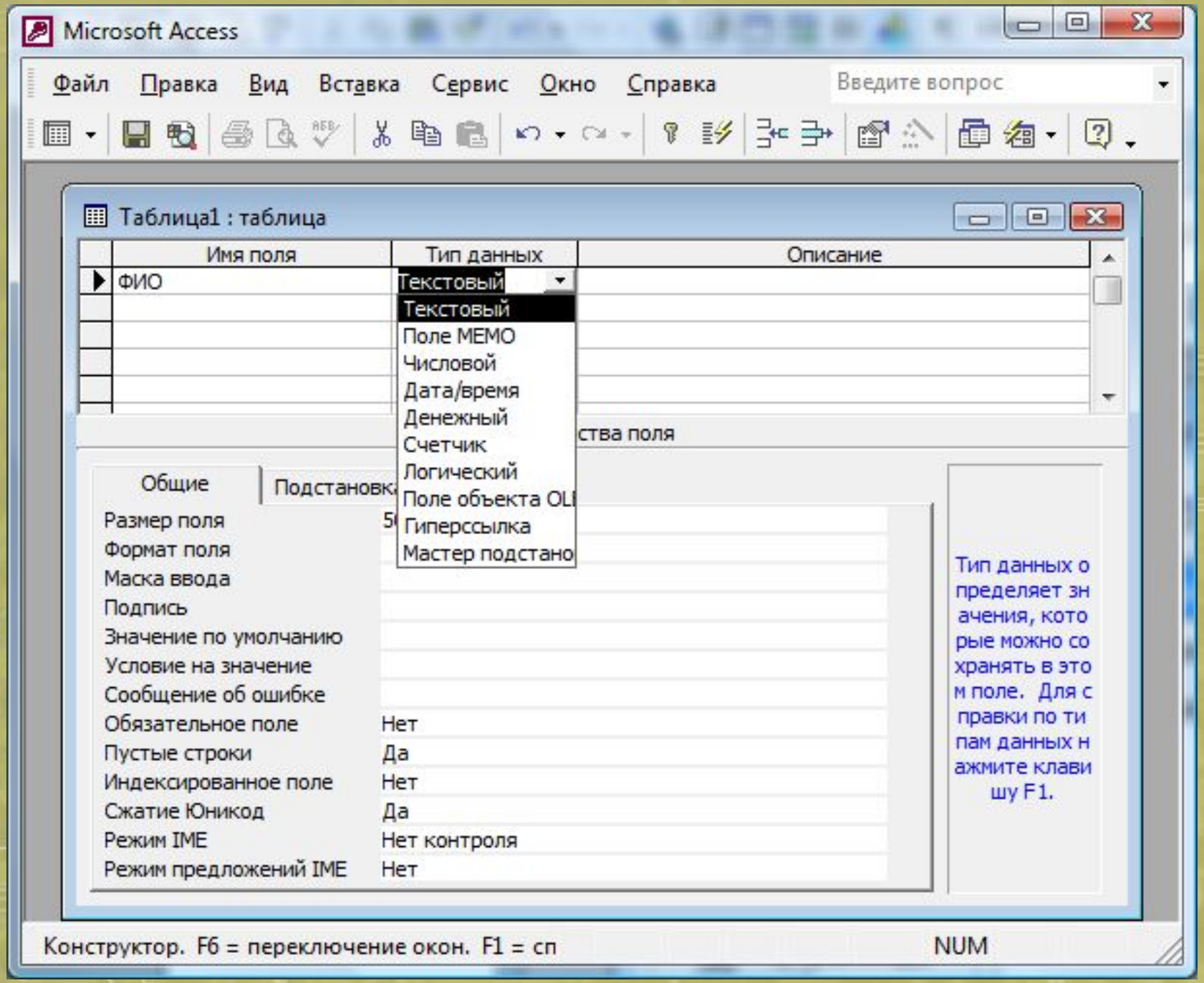
Создание новой БД



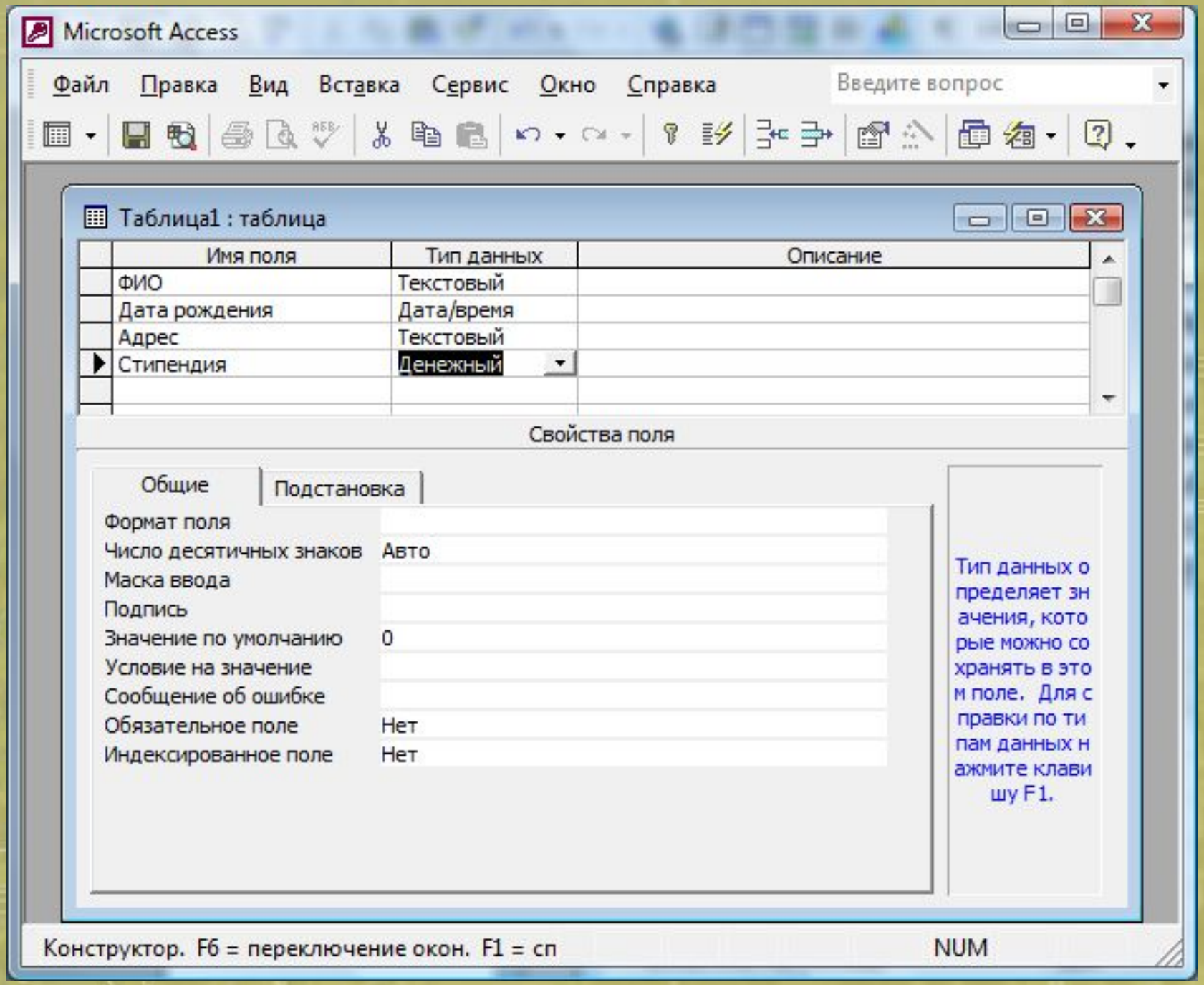
Сохранение БД под новым именем



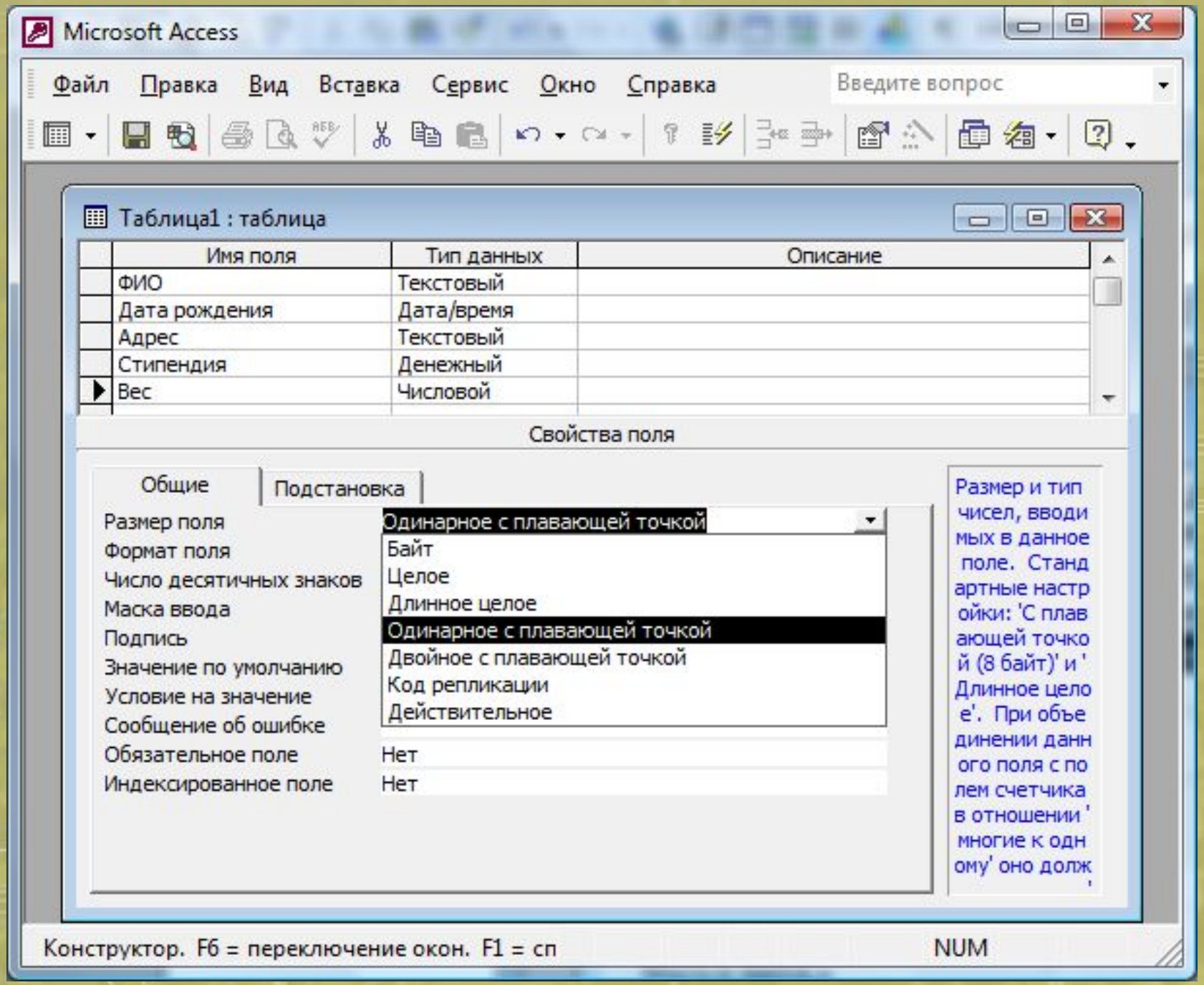
Создание новой таблицы (отношения) в БД



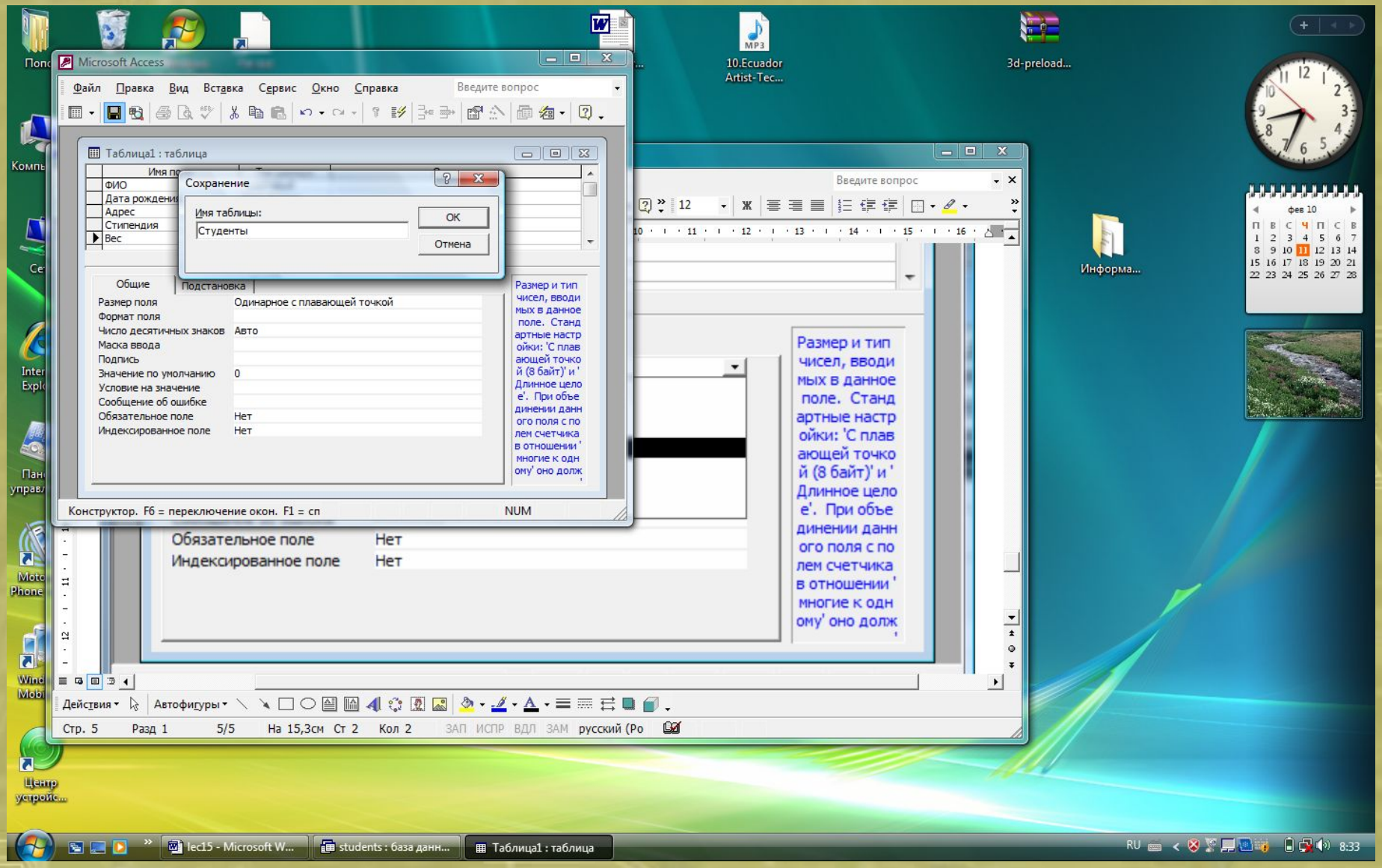
Создание полей в таблице БД



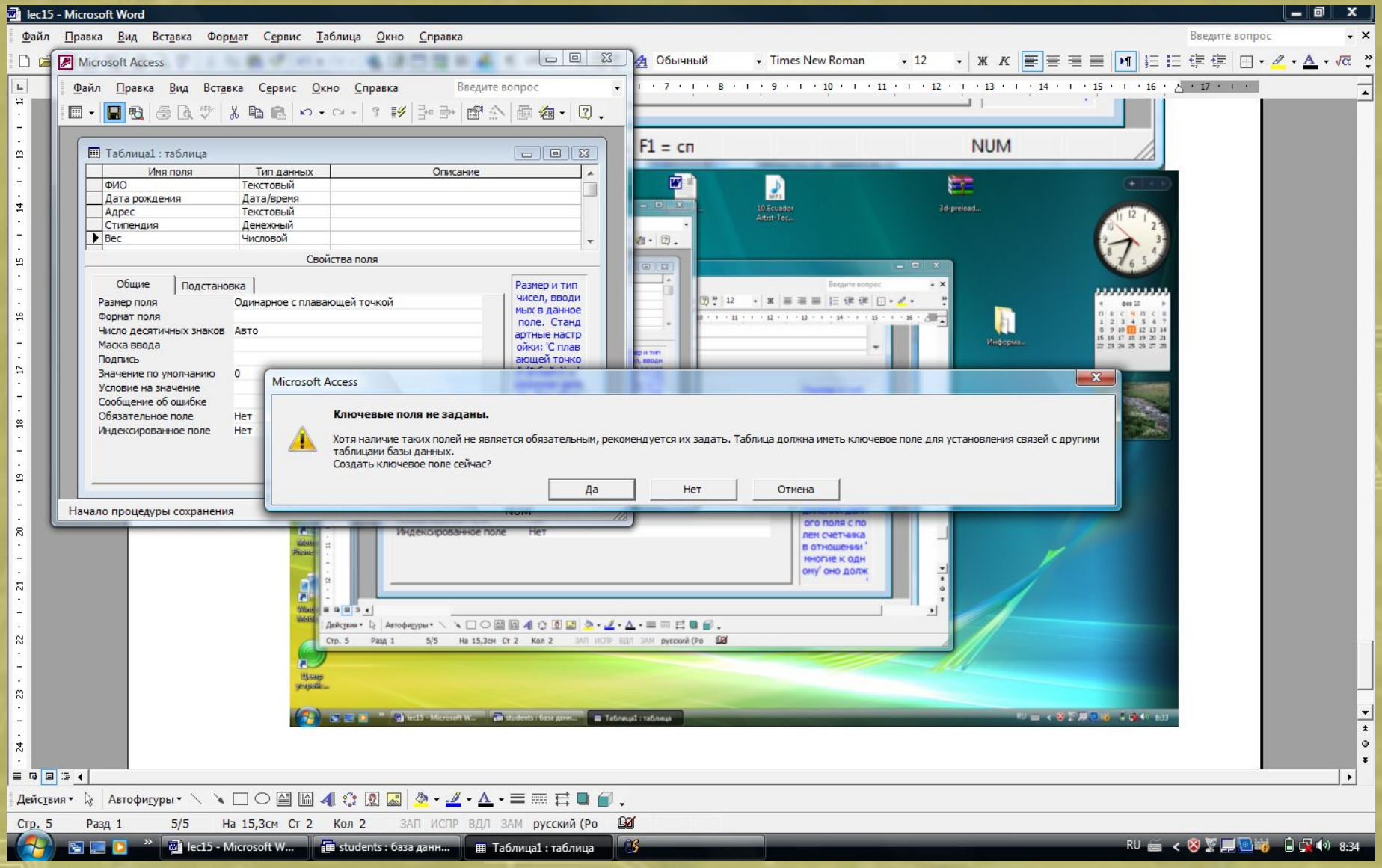
Пример создания поля денежного типа



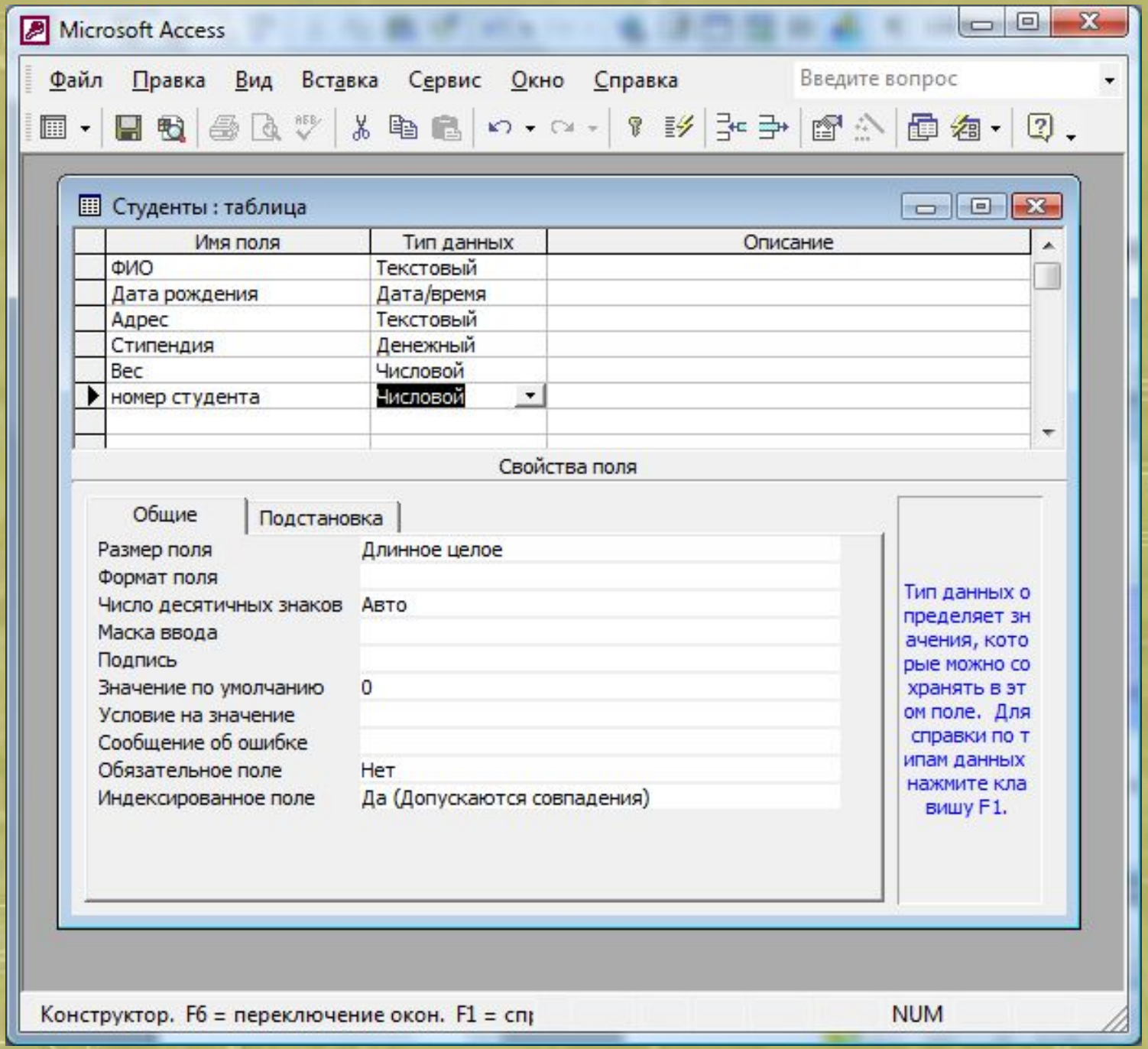
Пример создания числового поля «одинарное с плавающей точкой»



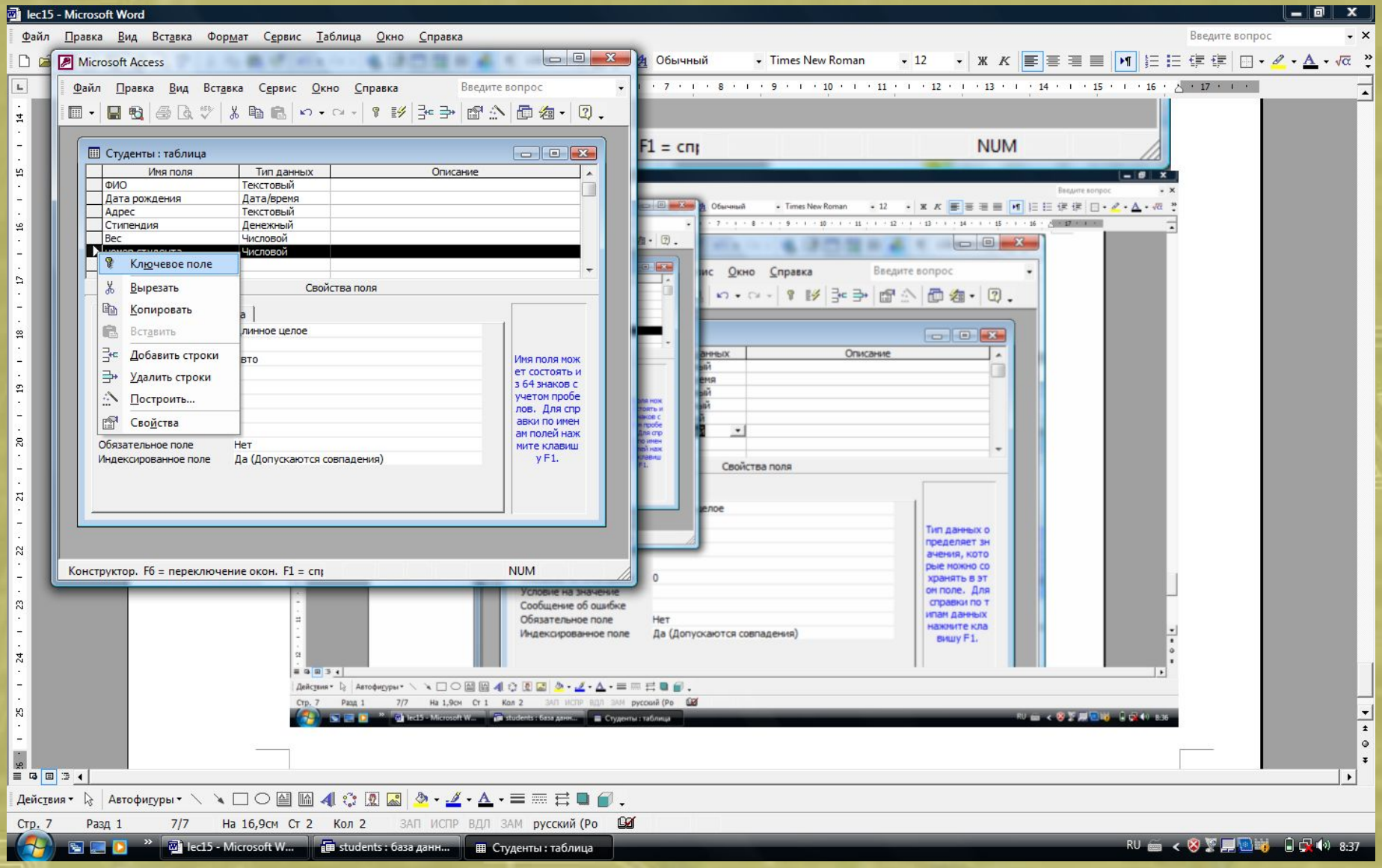
Сохранение таблицы в БД



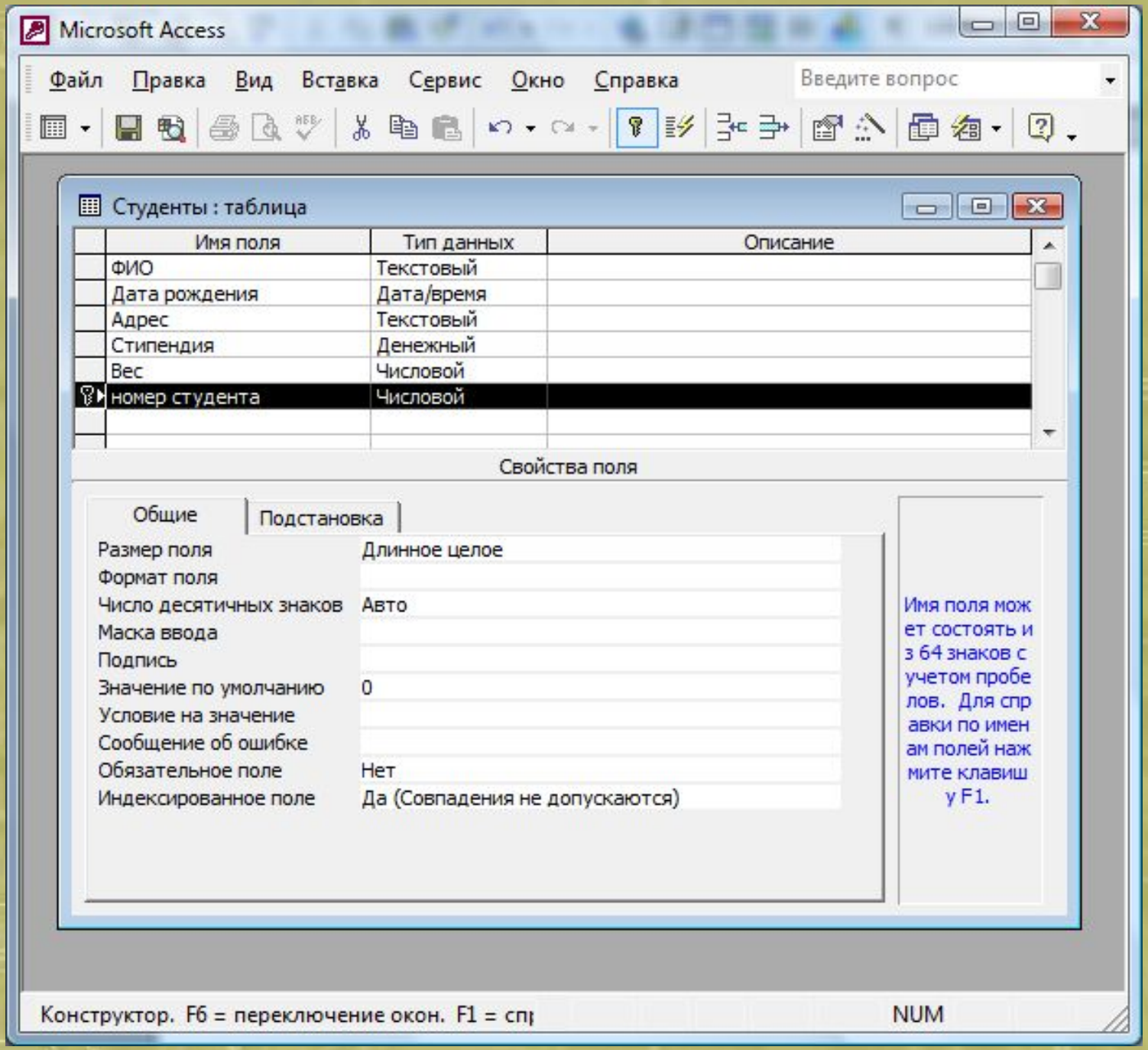
Добавление ключевого поля



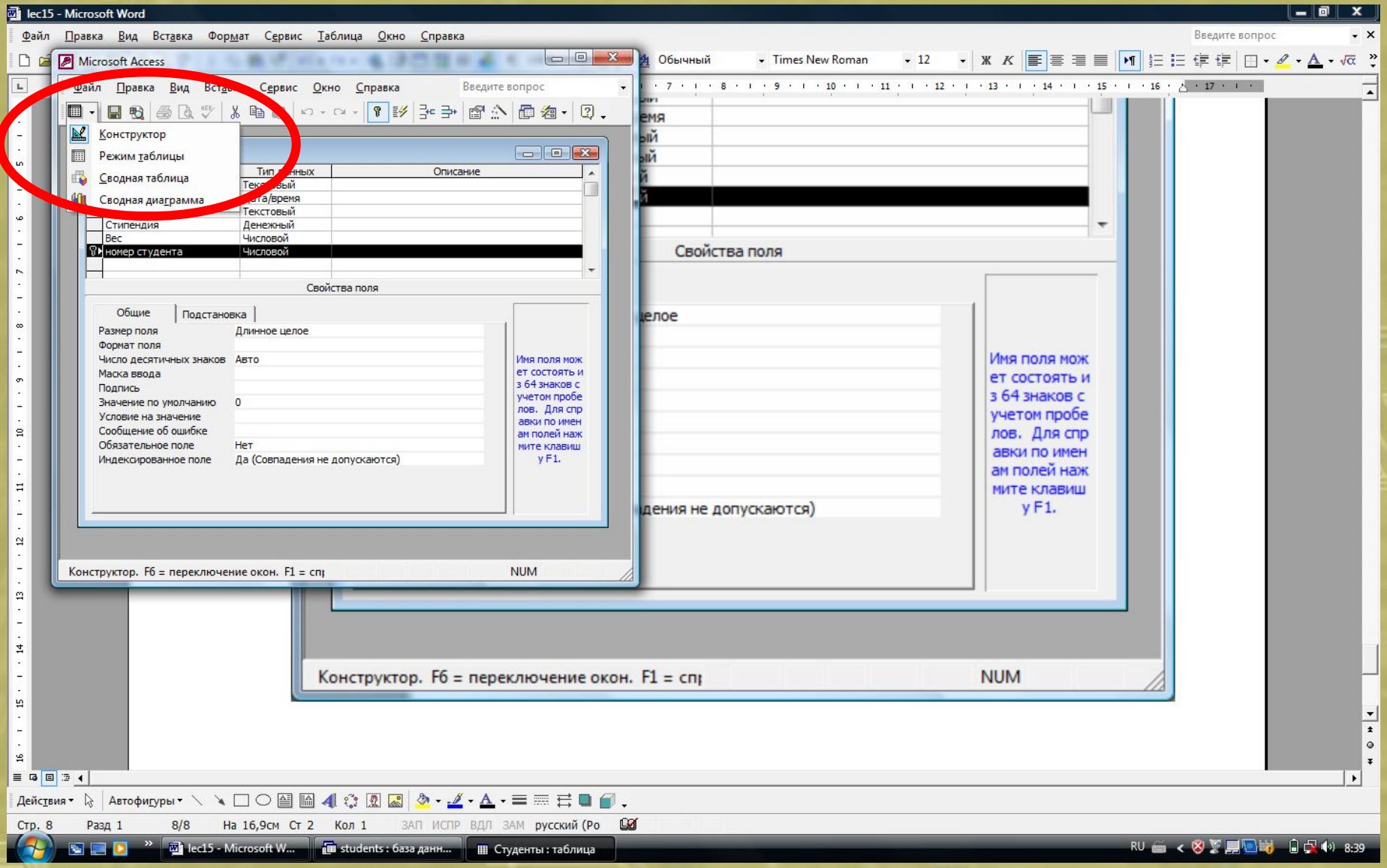
Добавление ключевого поля
«Номер студенческого билета»



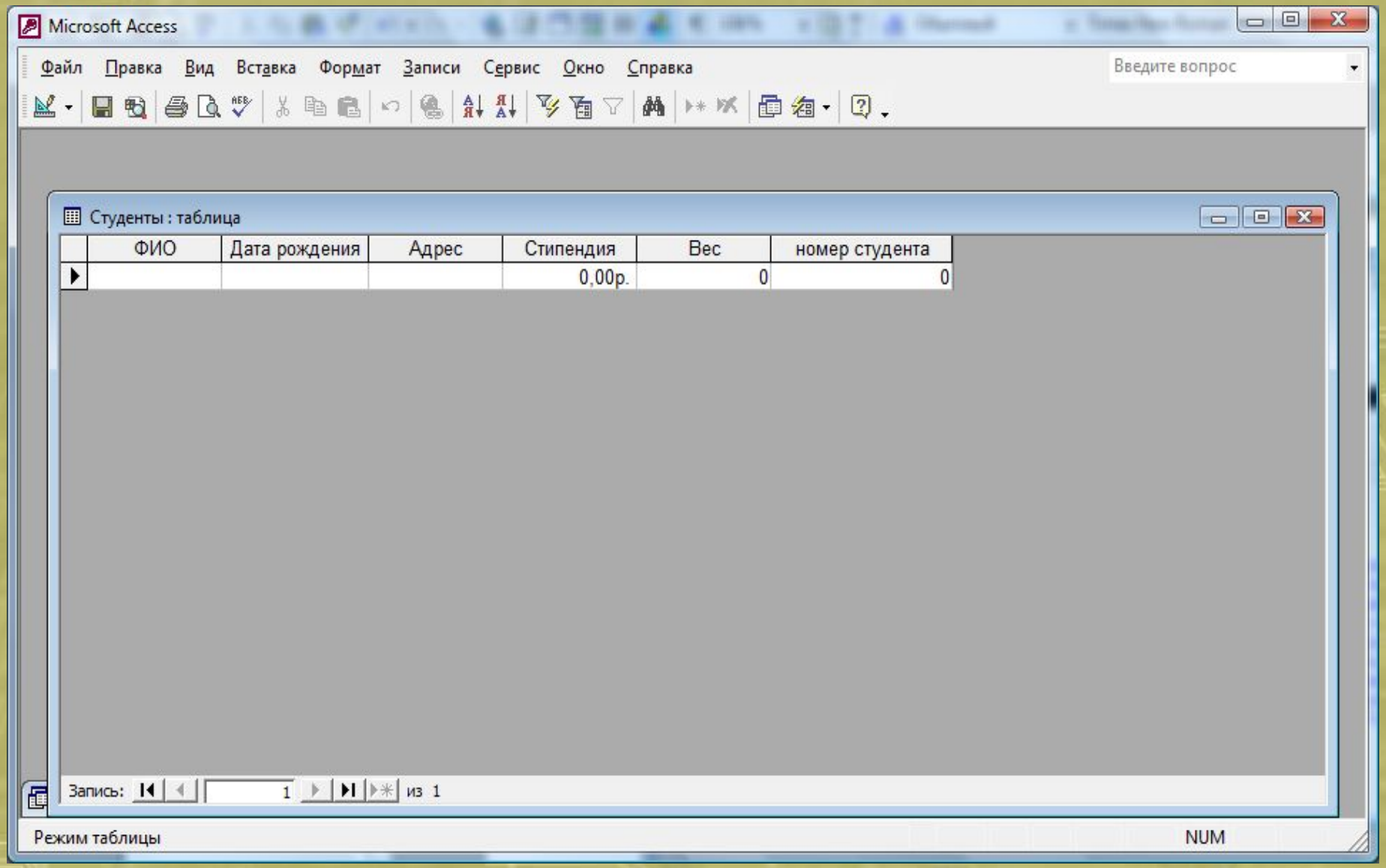
Способ указать ключевое поле



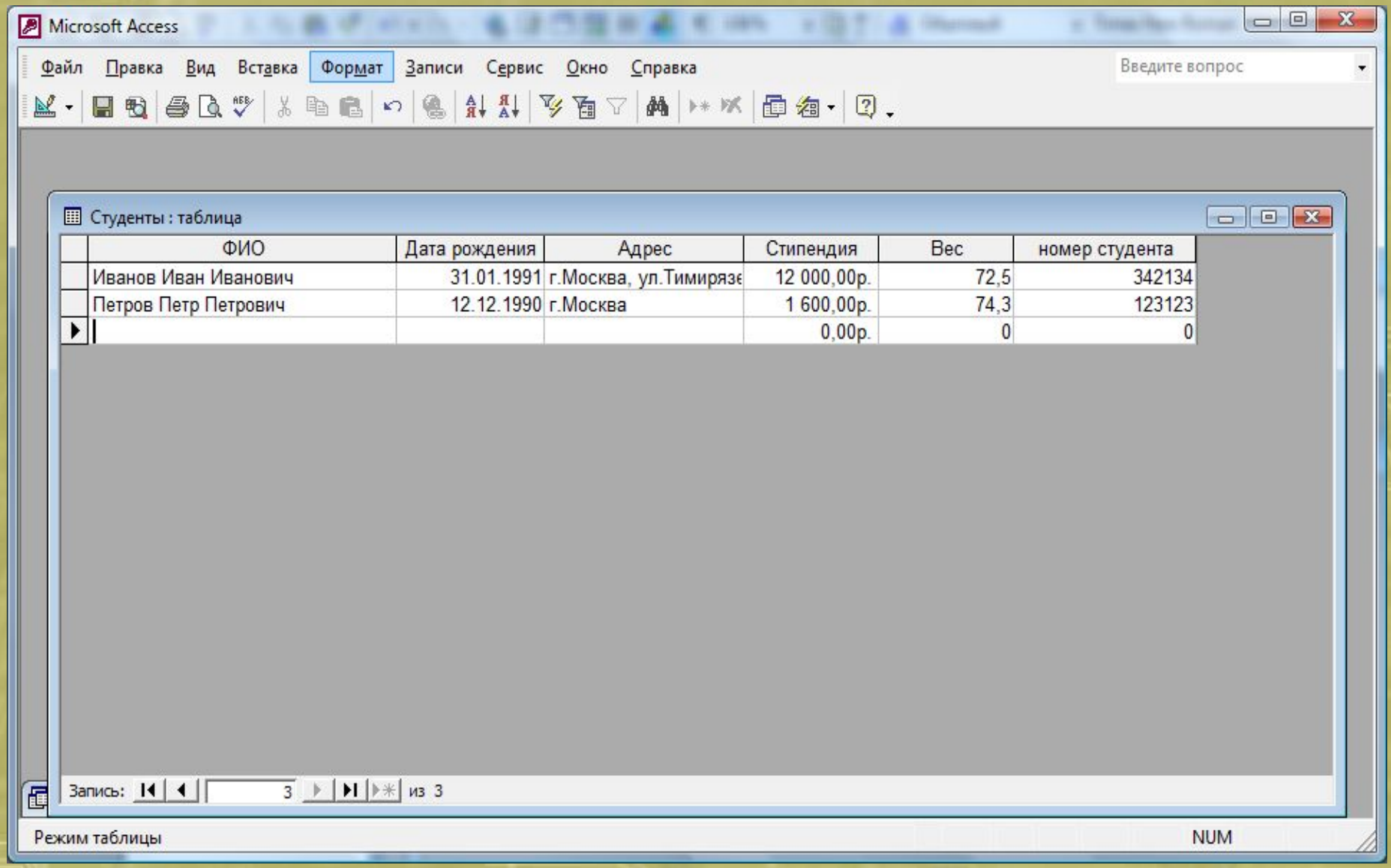
Обозначение ключевого поля в списке полей таблицы



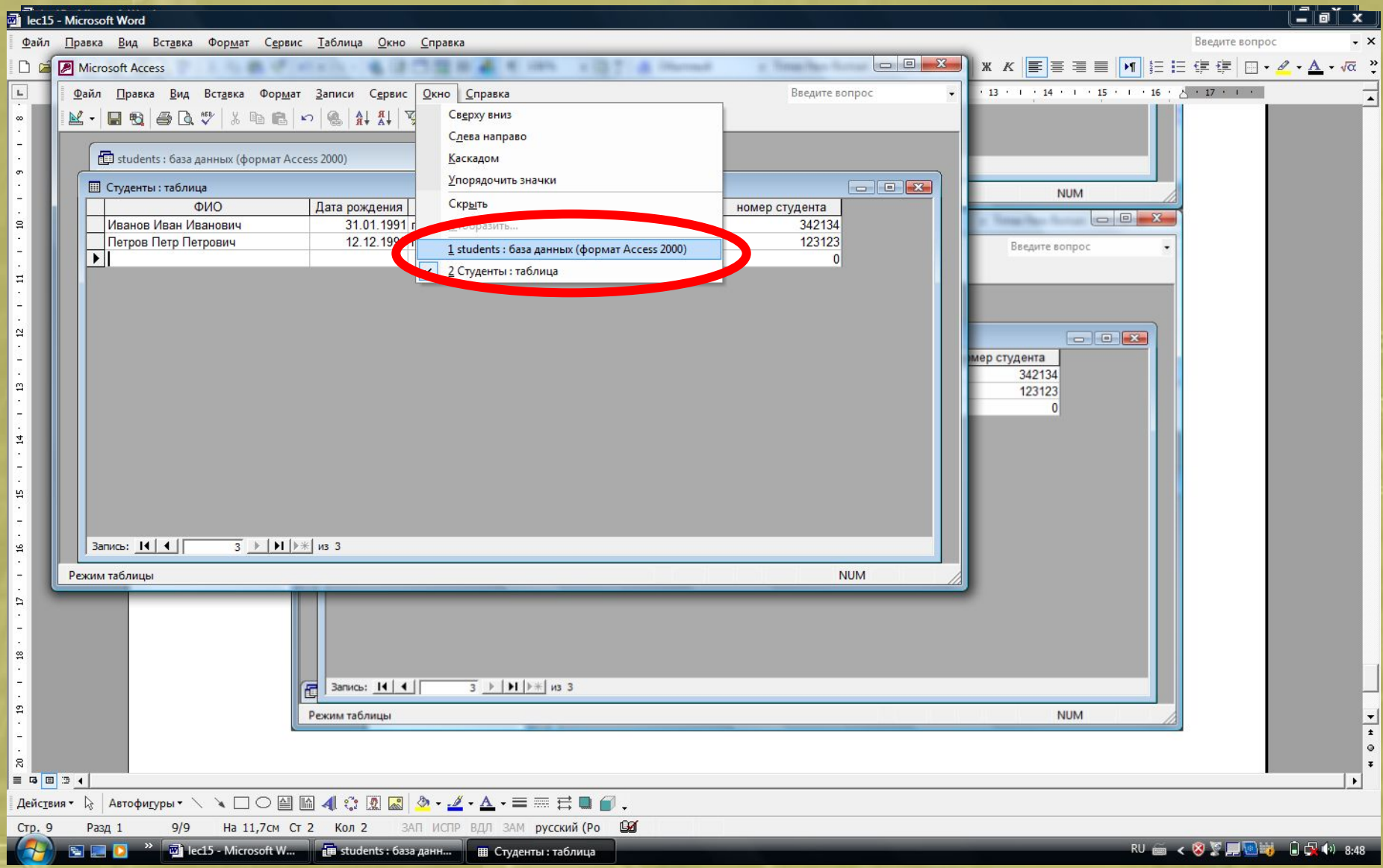
Основные режимы работы с таблицей



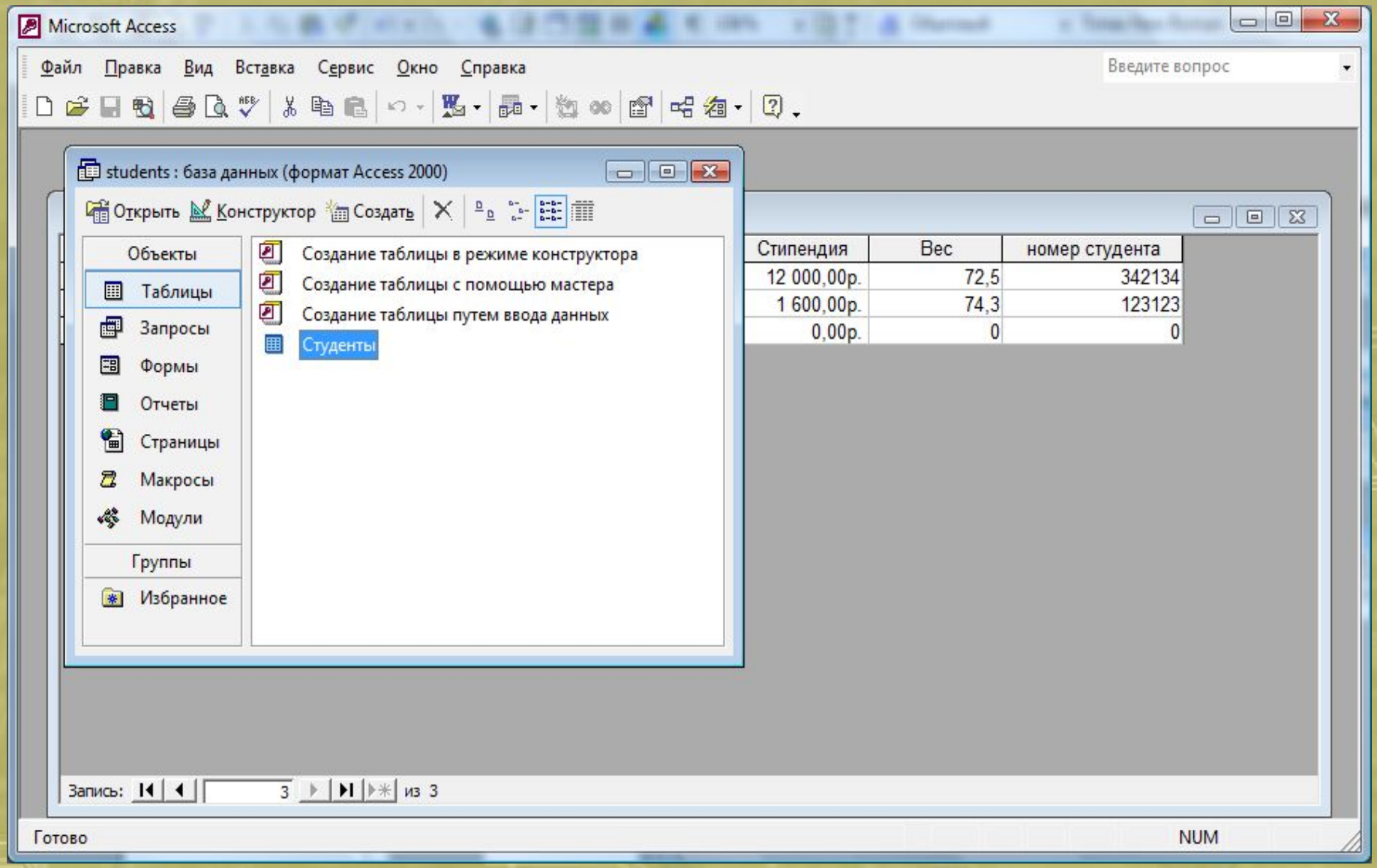
Просмотр в режиме «Таблица» (пустая таблица)



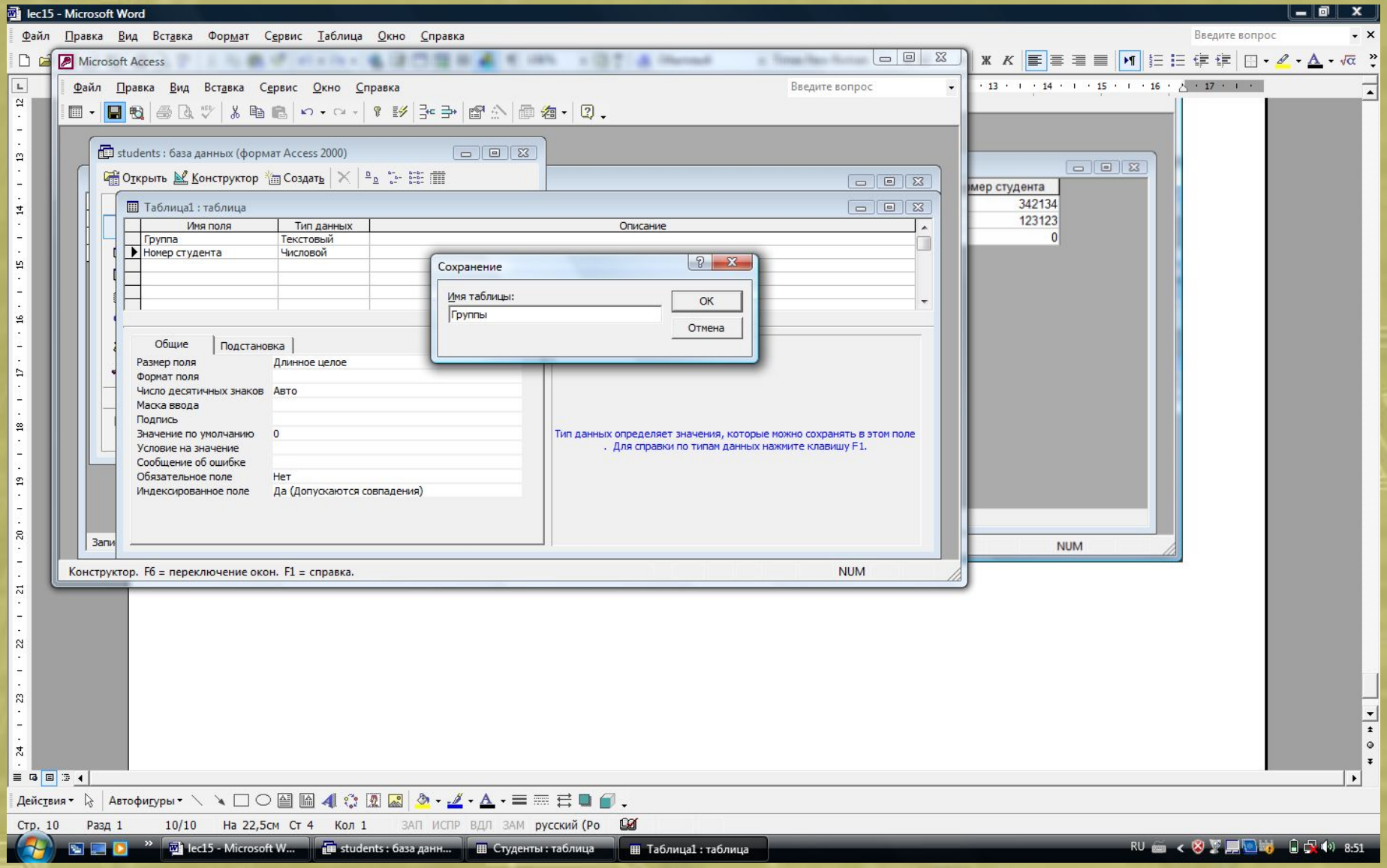
Добавление данных (записей о студентах) в таблицу



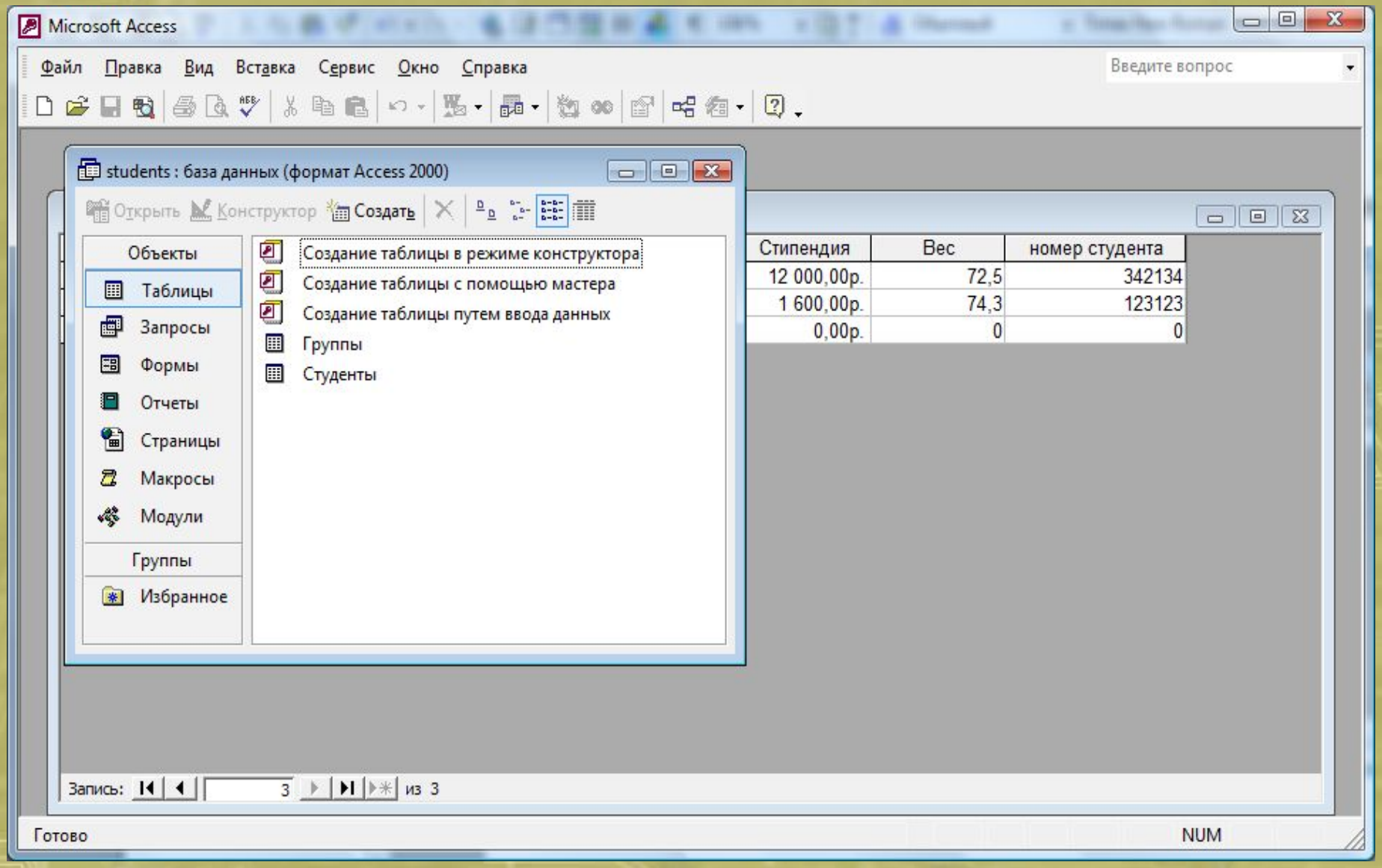
Переход в окно «База данных»



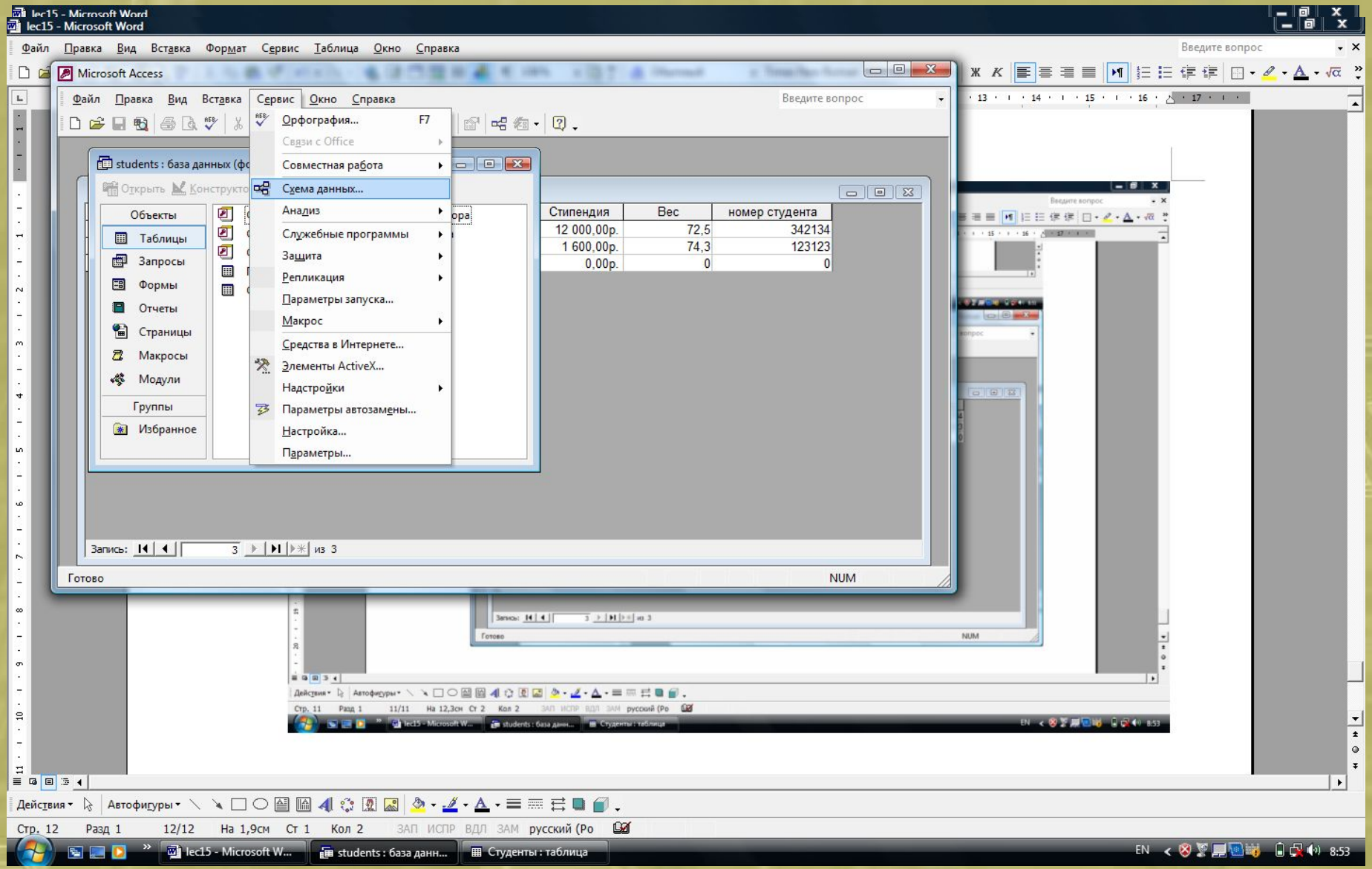
Вид окна «База данных» раздел «Таблицы»



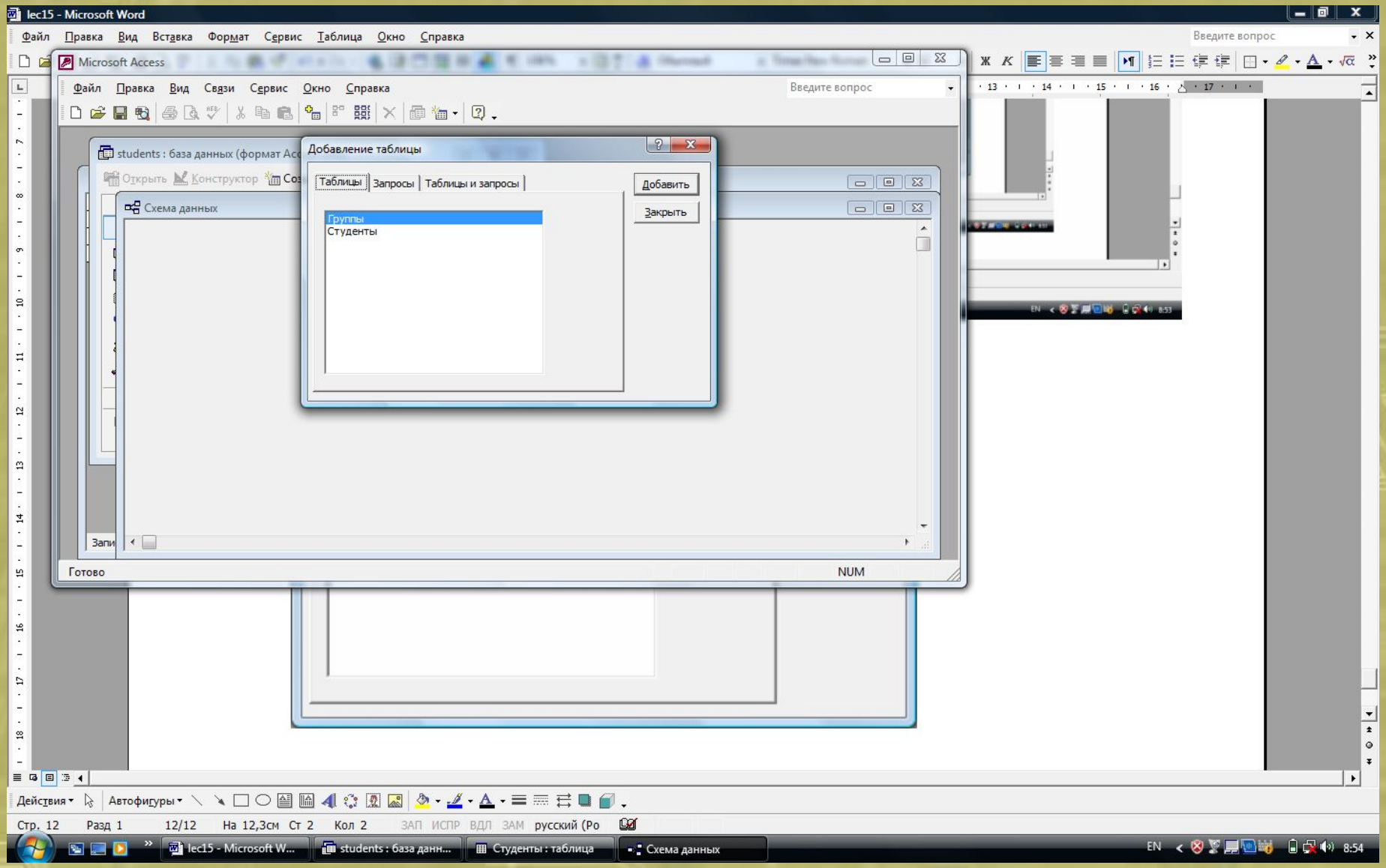
Добавление новой таблицы «Группы» в БД



Вид окна «База данных» раздел «Таблицы»
Создано две таблицы «Студенты» и «Группы»



Добавление связей между отношениями (таблицами) в меню «Сервис/схема данных»



В схеме данных (схеме связи отношений) указываются участвующие в связях таблицы

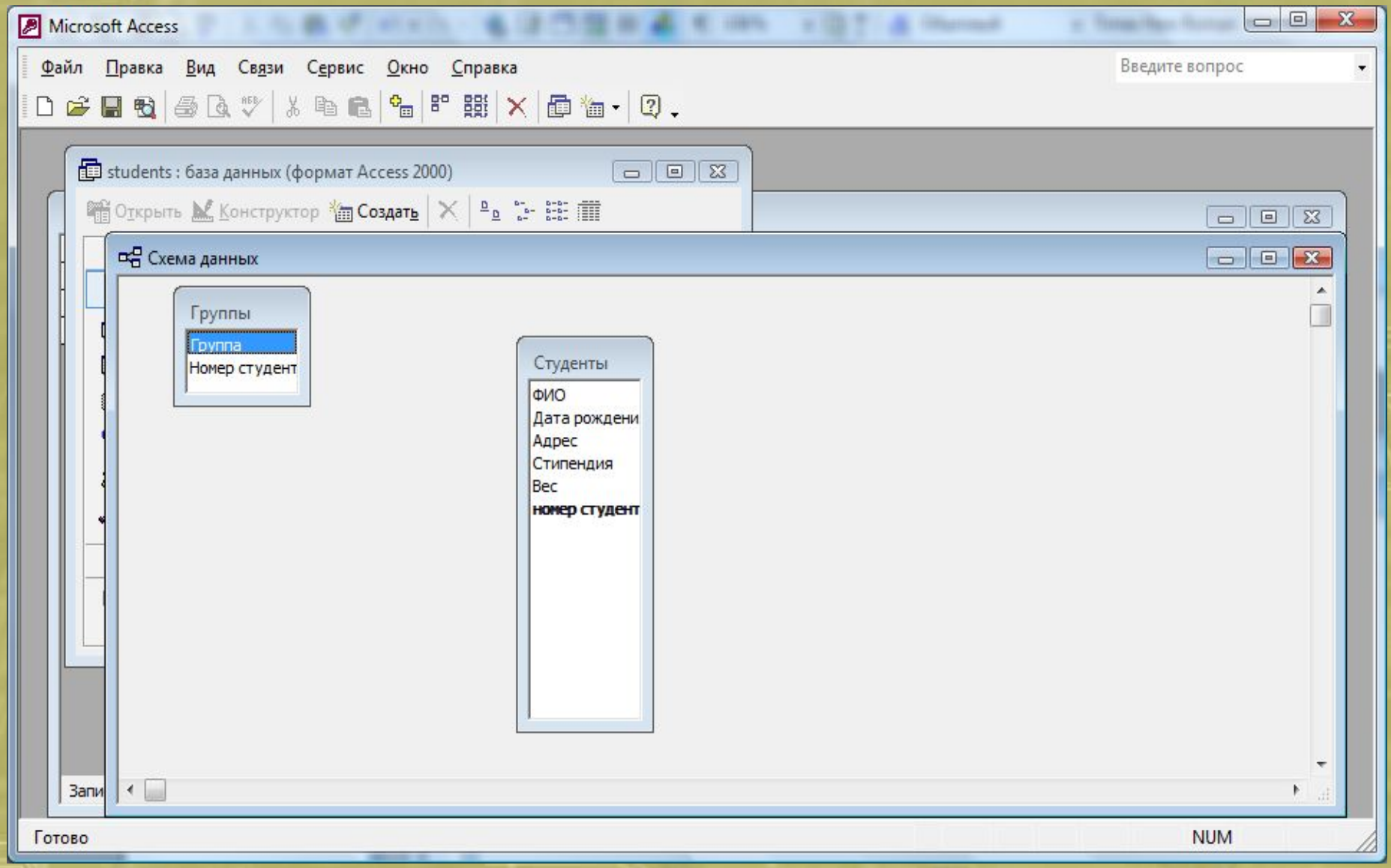
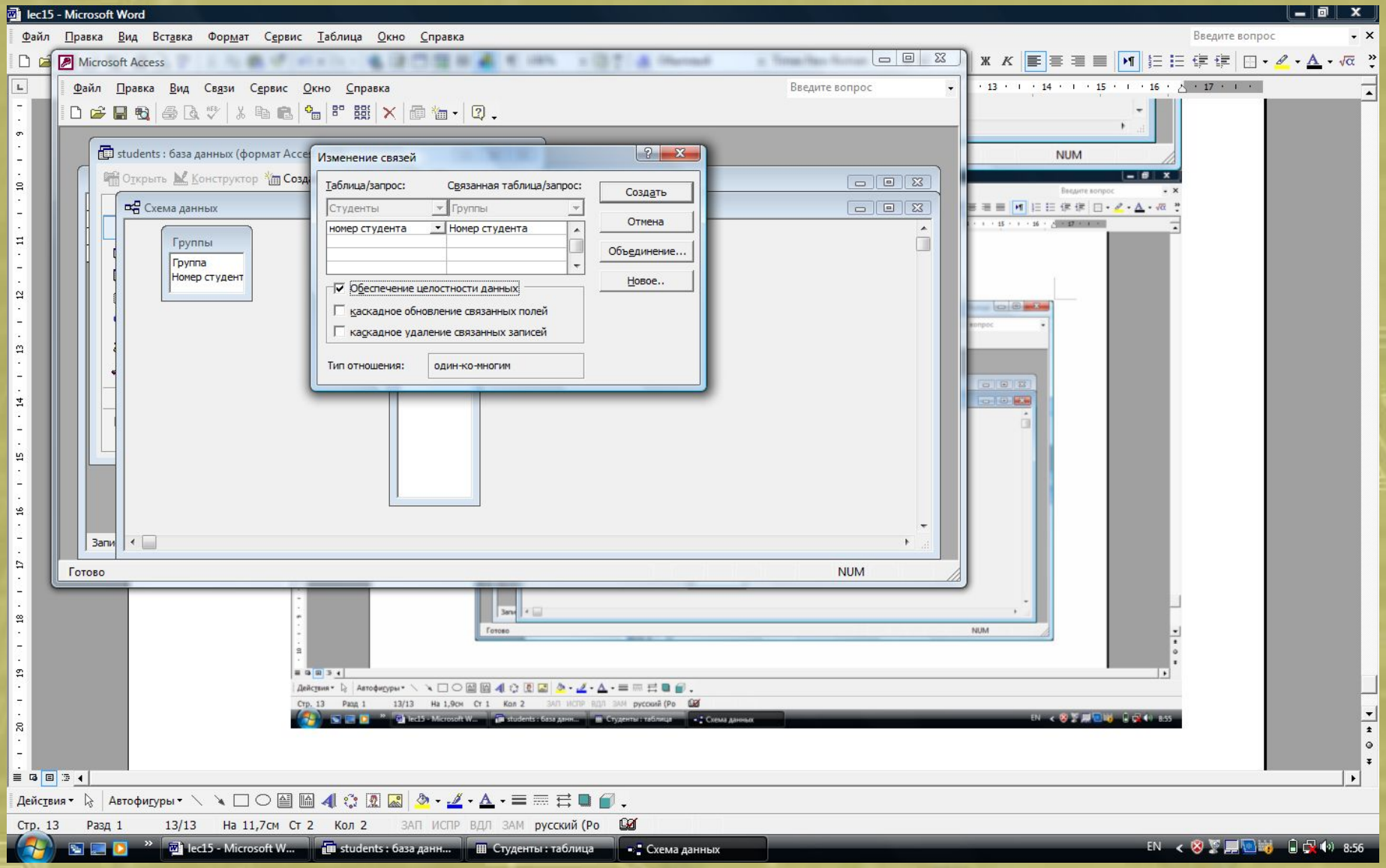
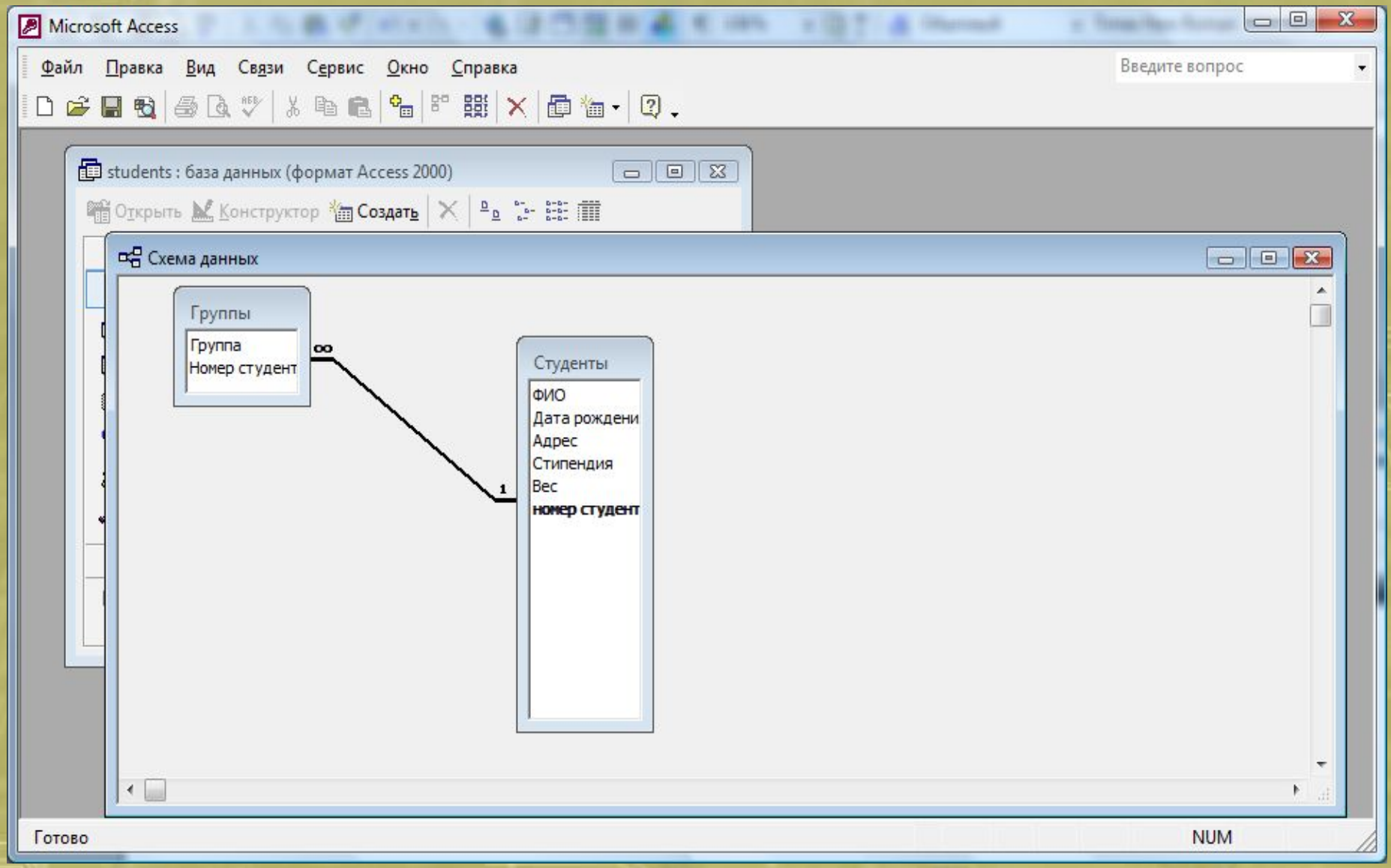


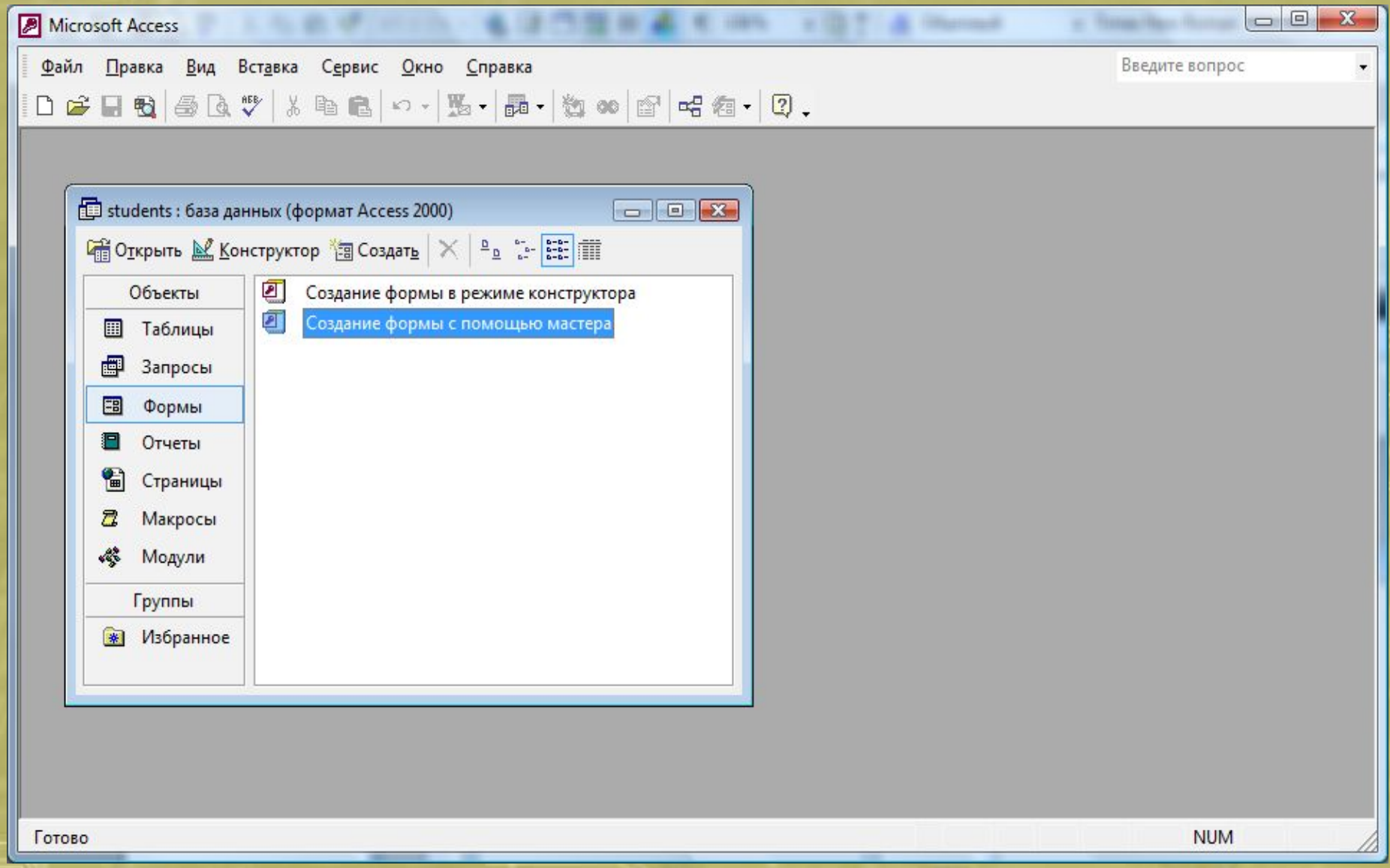
Схема данных. Связи еще не установлены



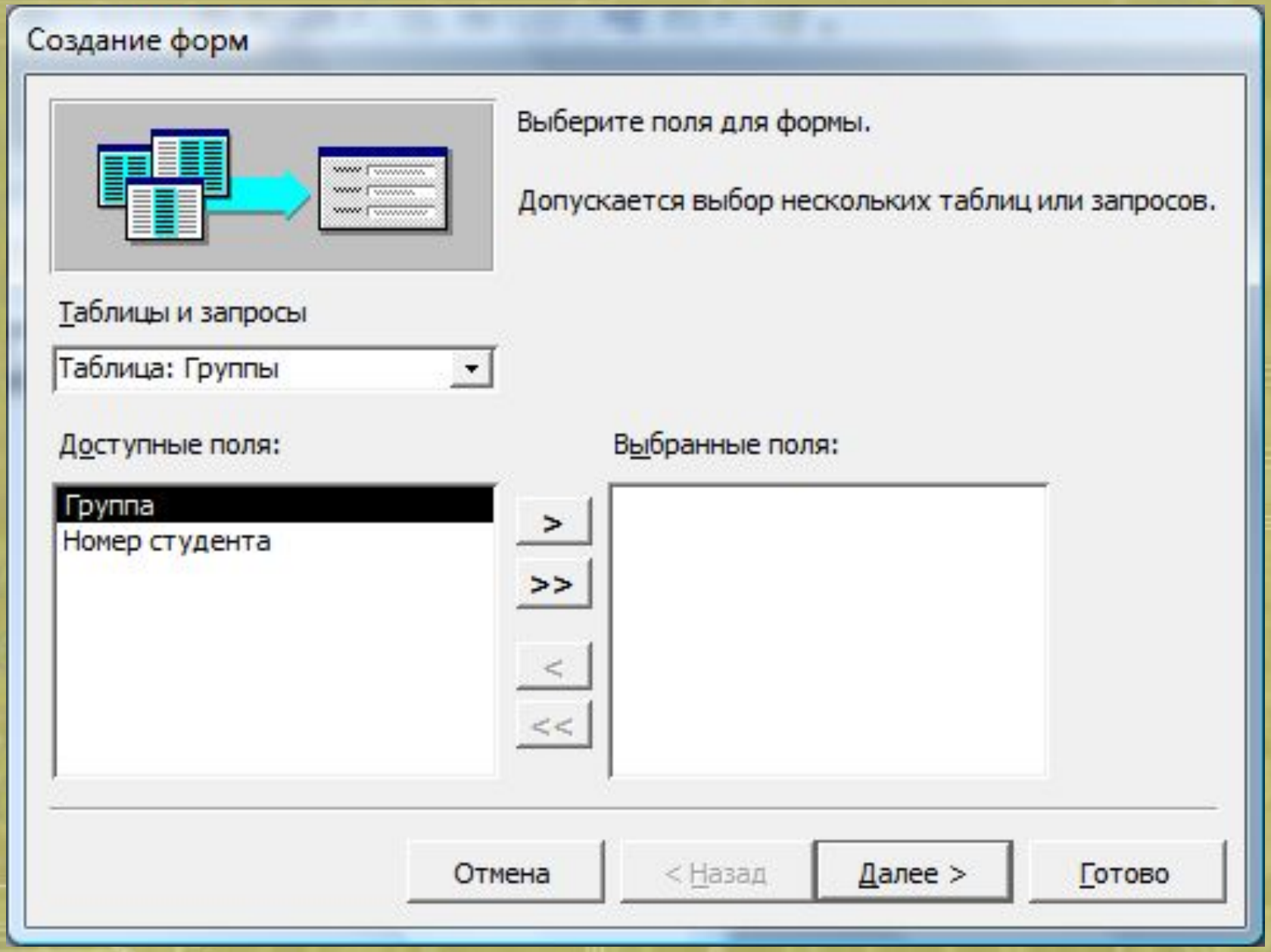
Запрос на тип связи и обеспечение целостности данных



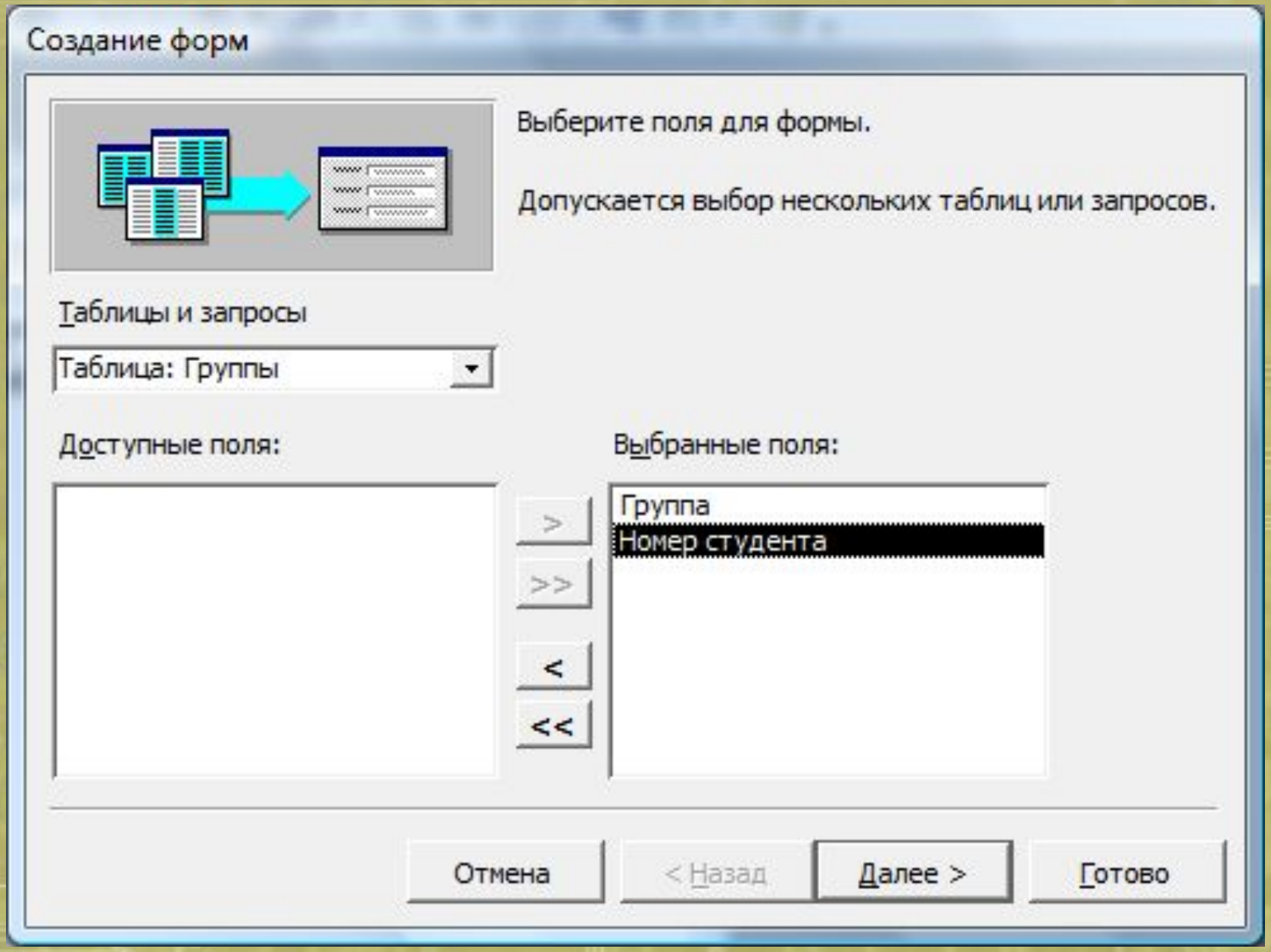
Отображение связи «один ко многим»



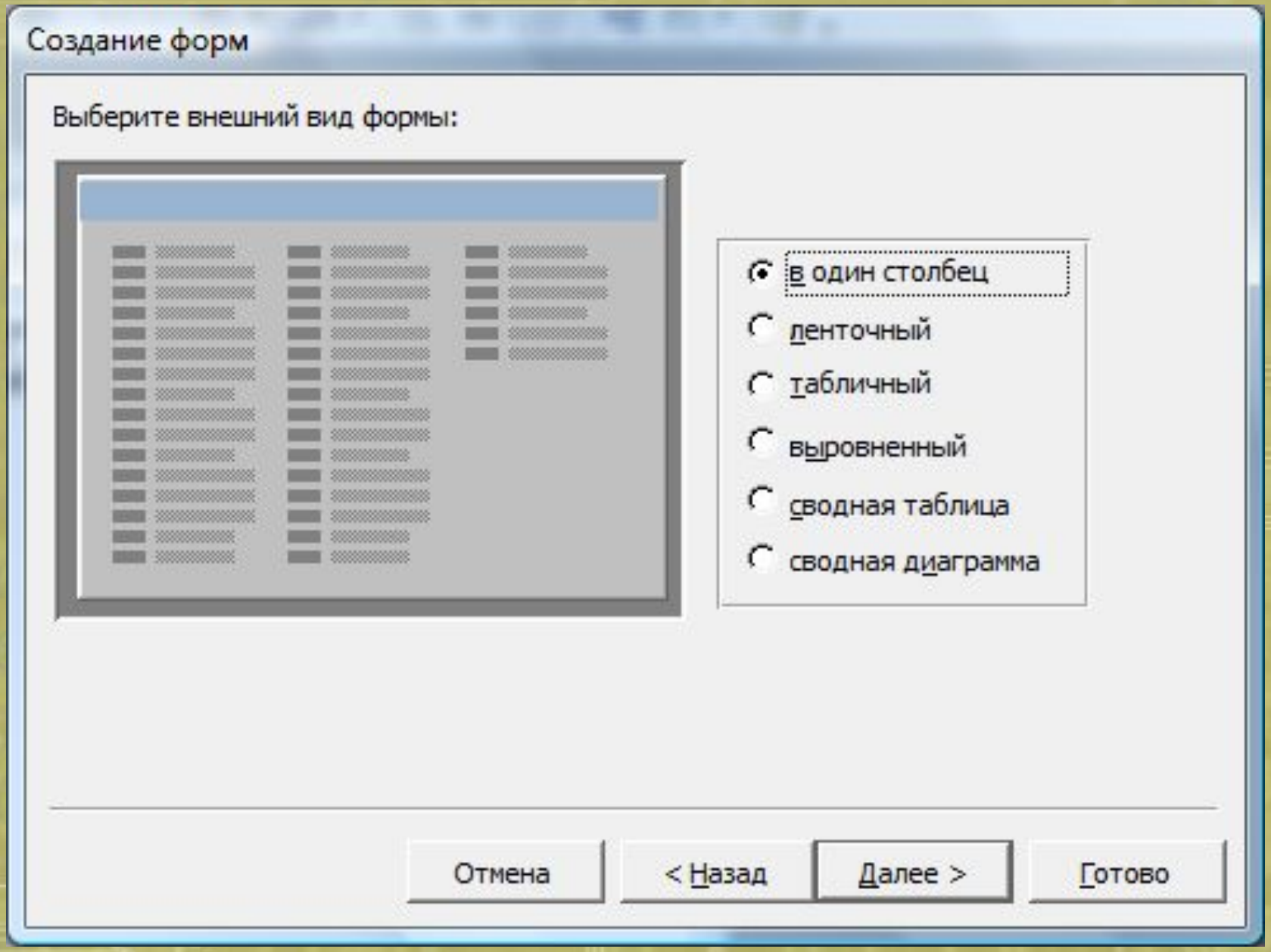
Создание формы



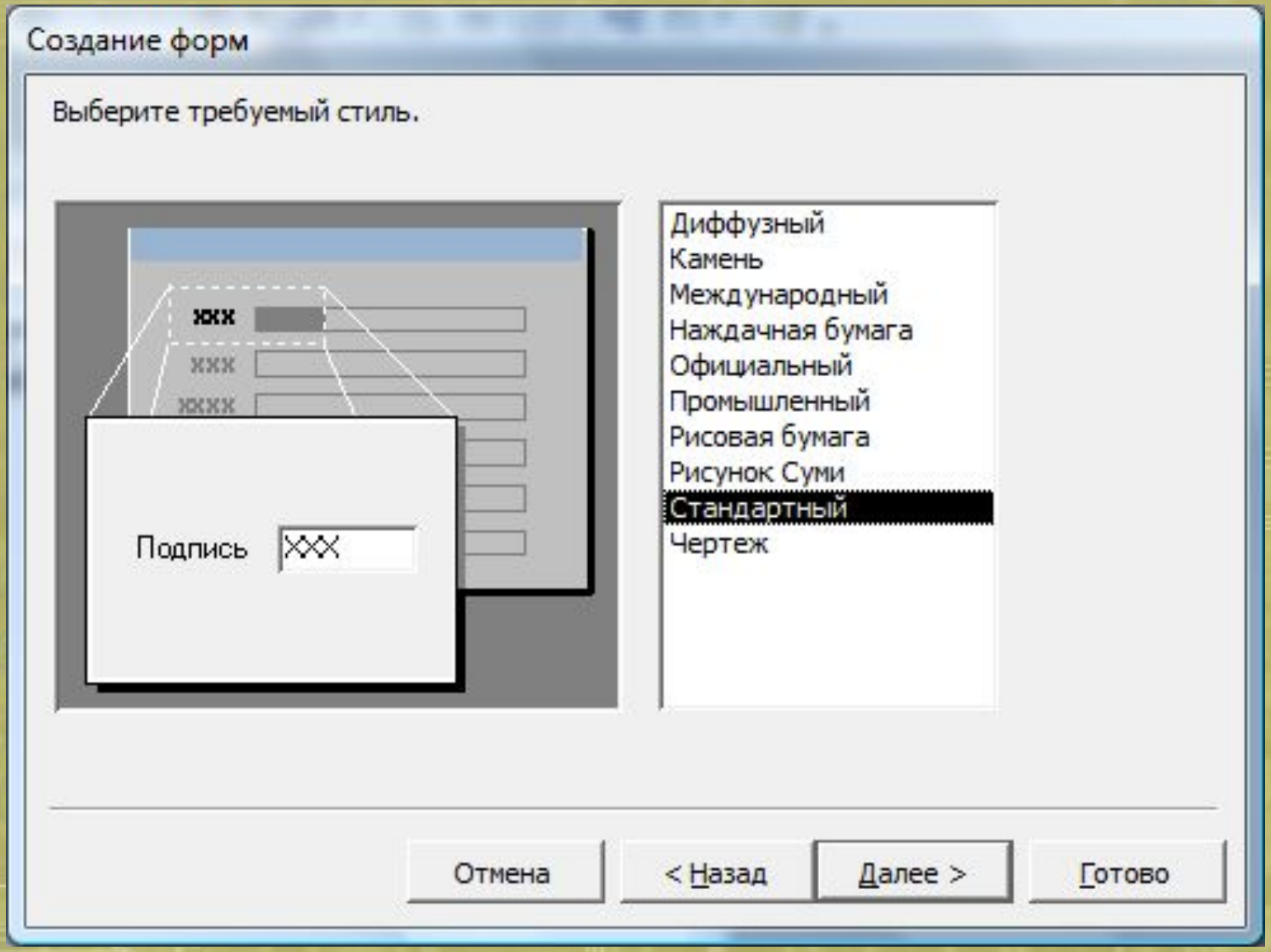
Выбор таблицы для формы и полей



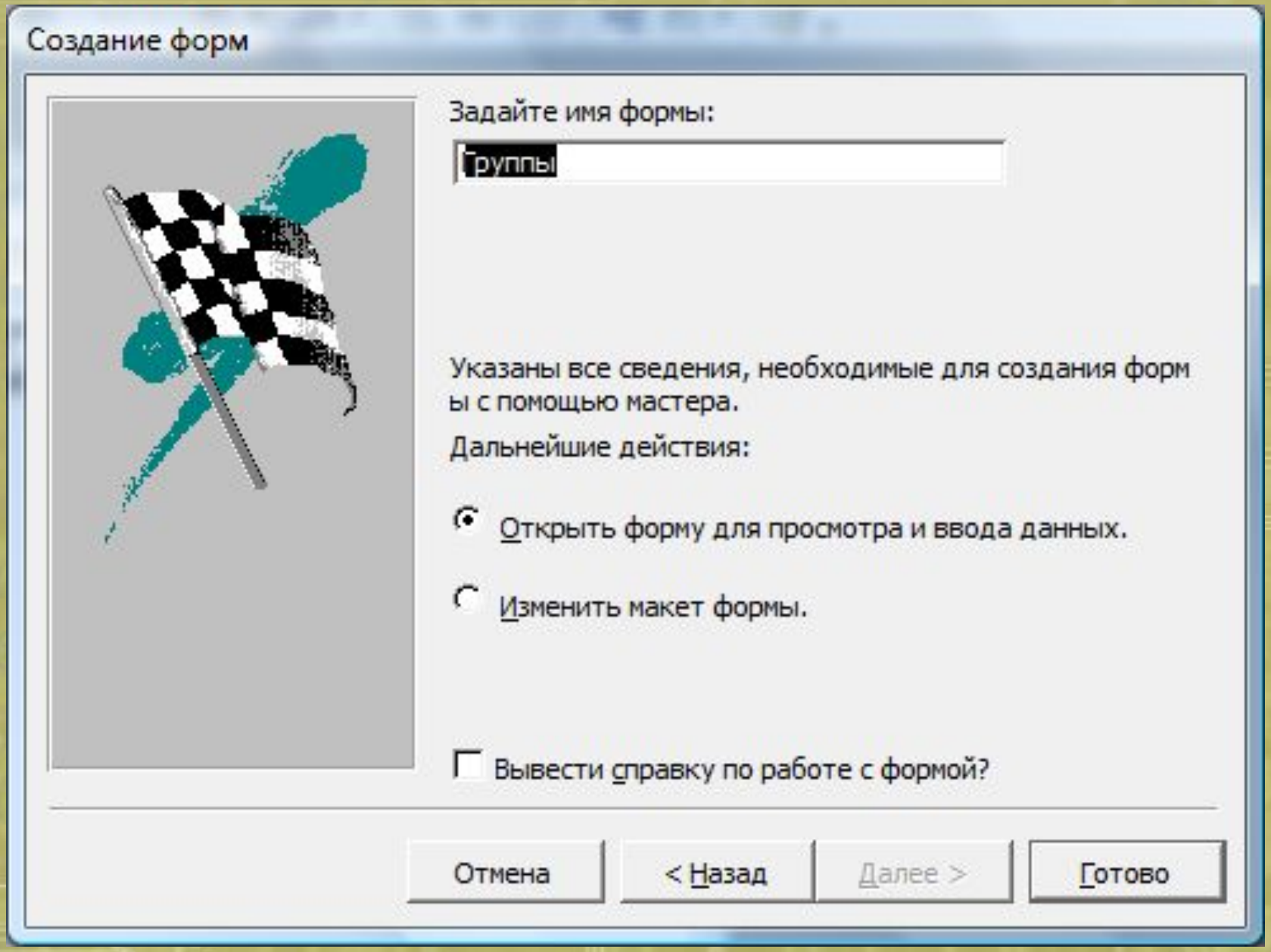
Выбранные поля попали в правую часть экрана



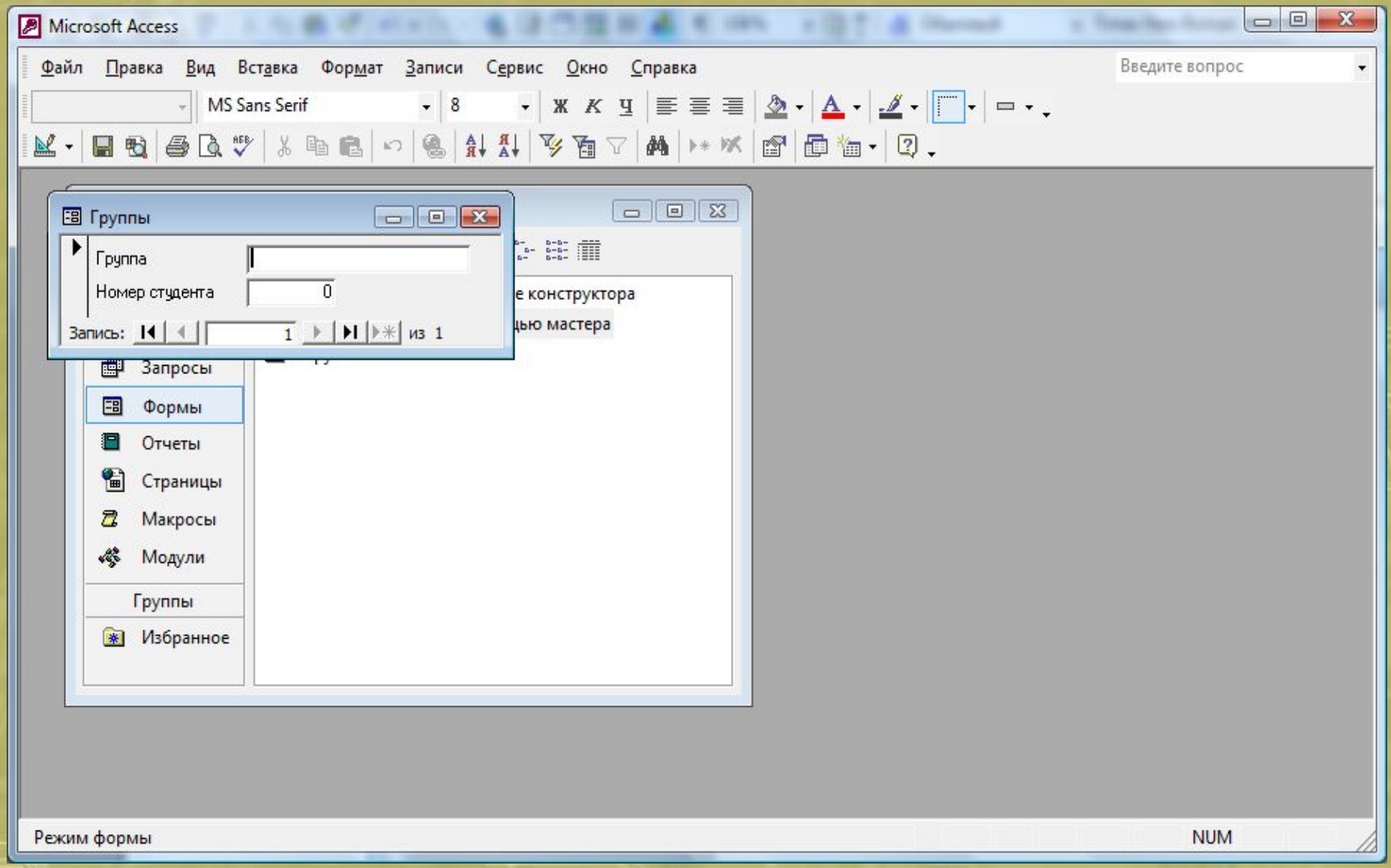
Выбор внешнего вида формы «В один столбец»



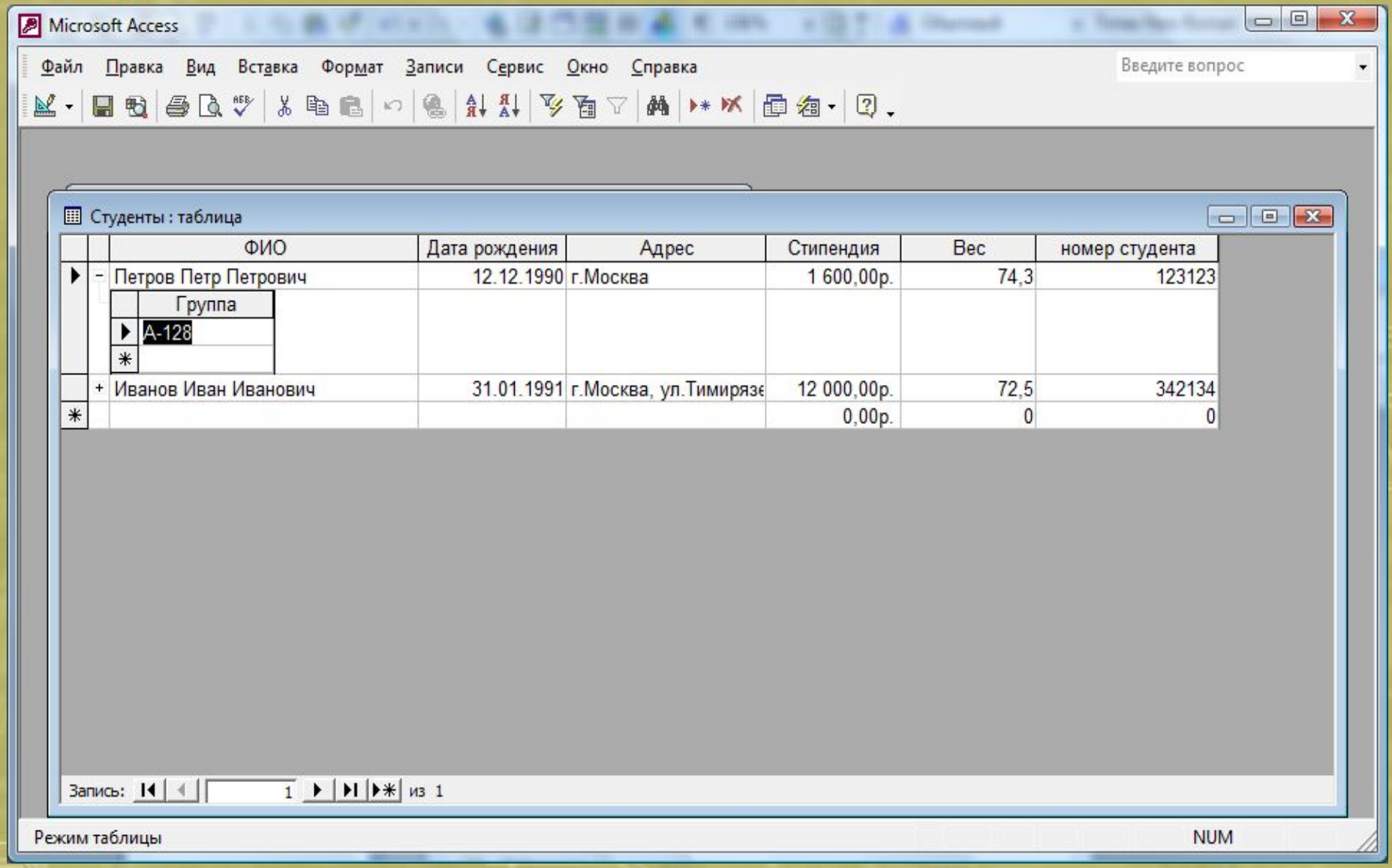
Выбор стиля формы «Стандартный»



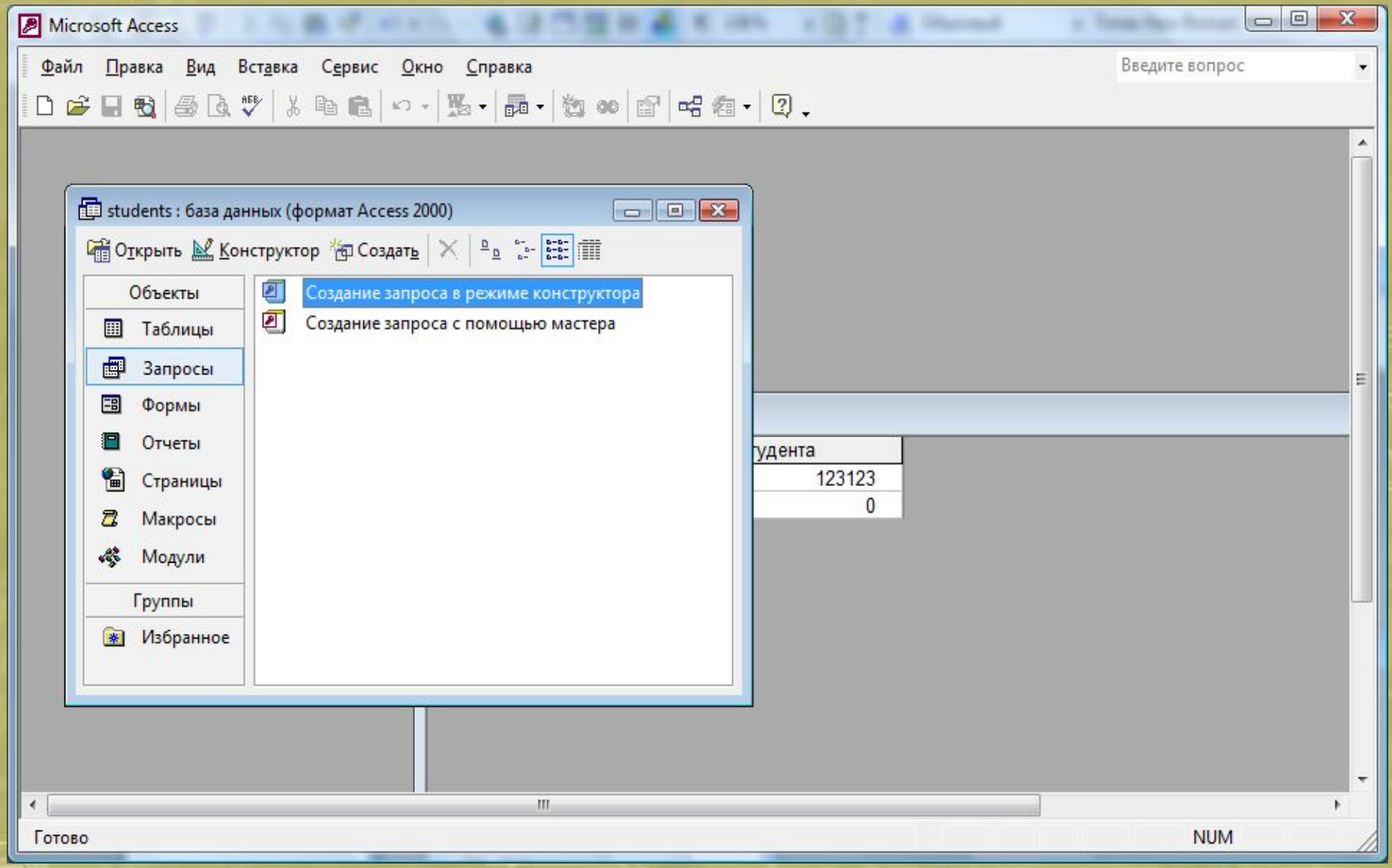
Задание имения формы и ее сохранение



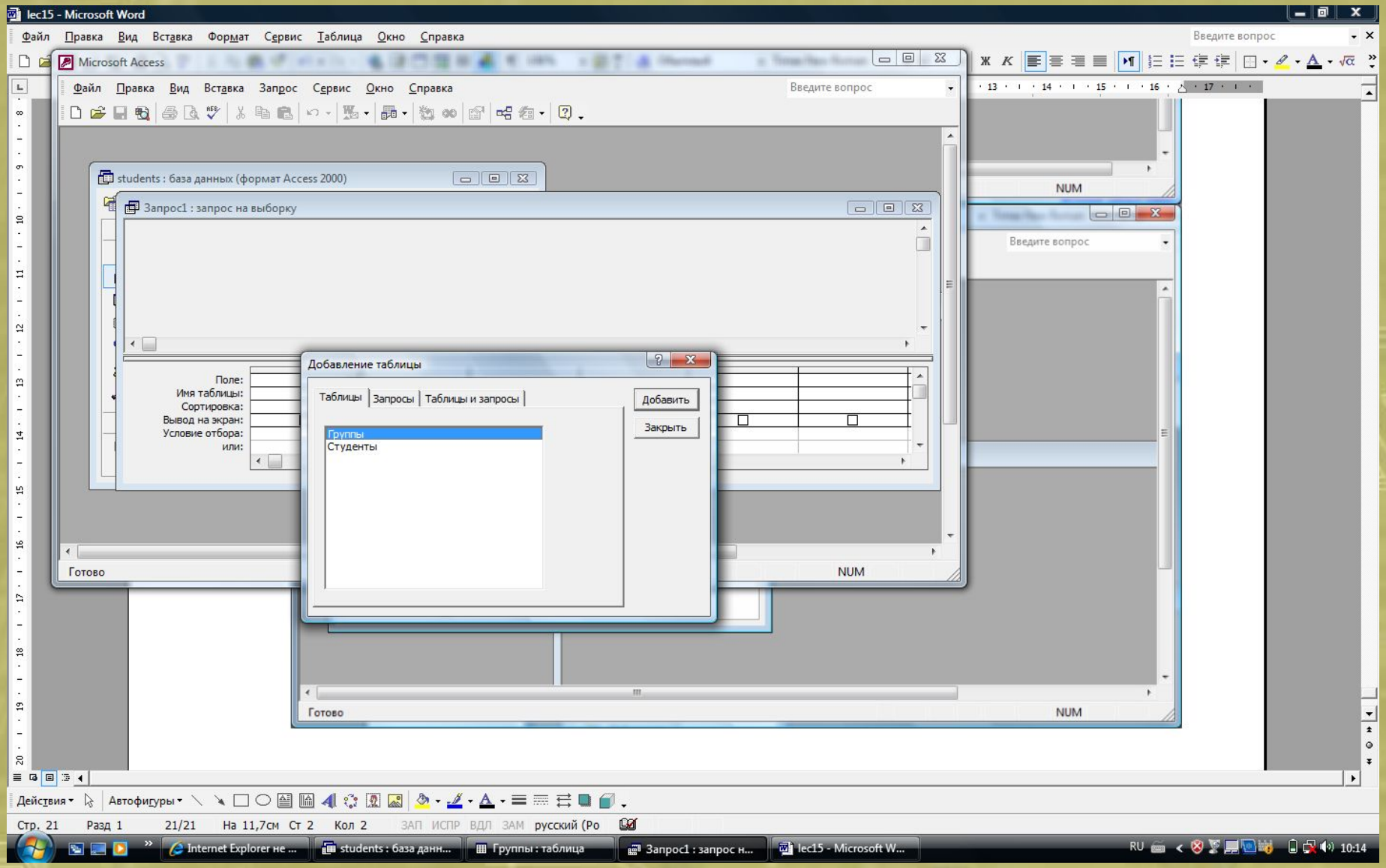
Вид созданной формы «Группы» для ввода данных в таблицу



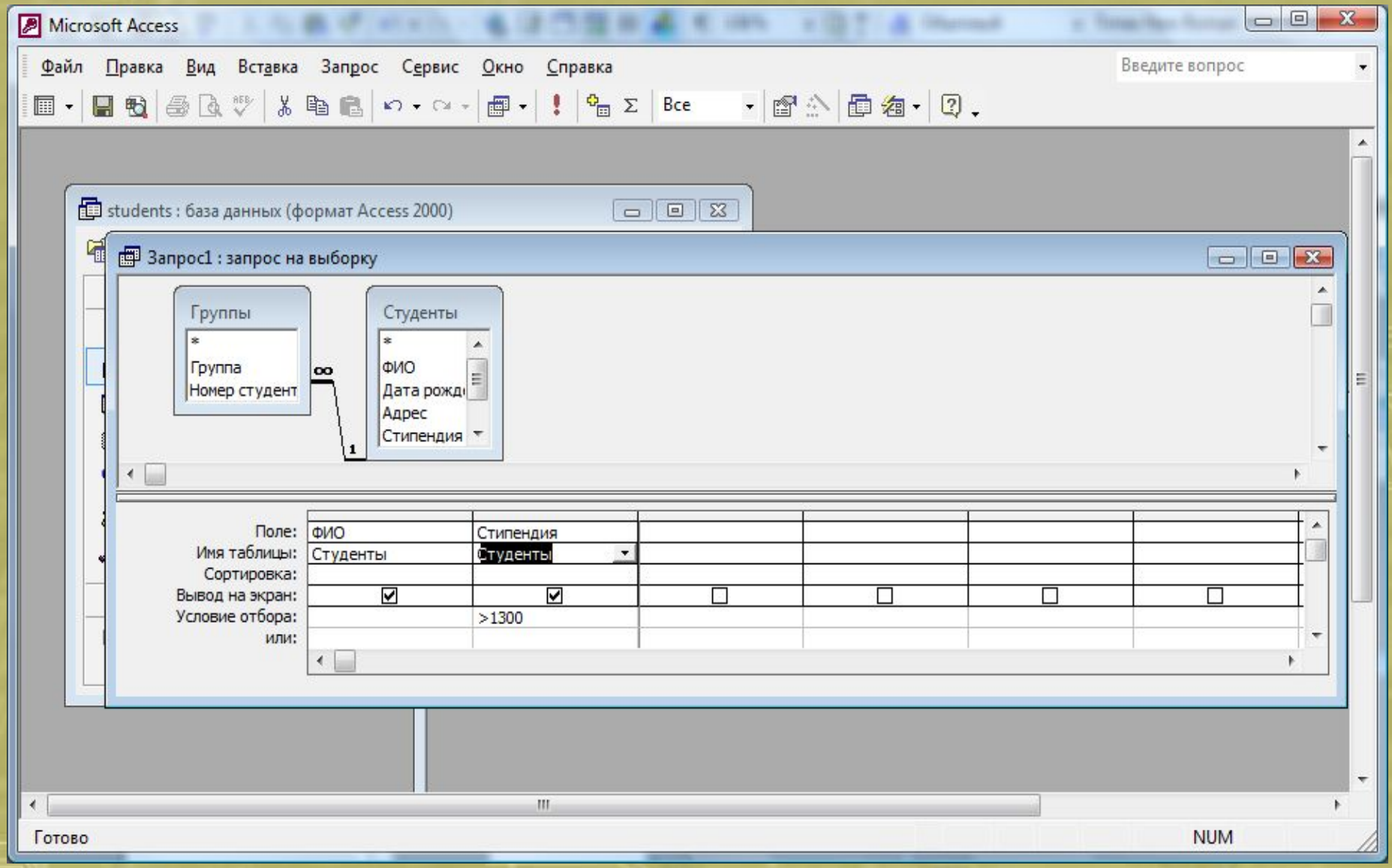
Столбец связи поля «ФИО» с таблицей «Группы»



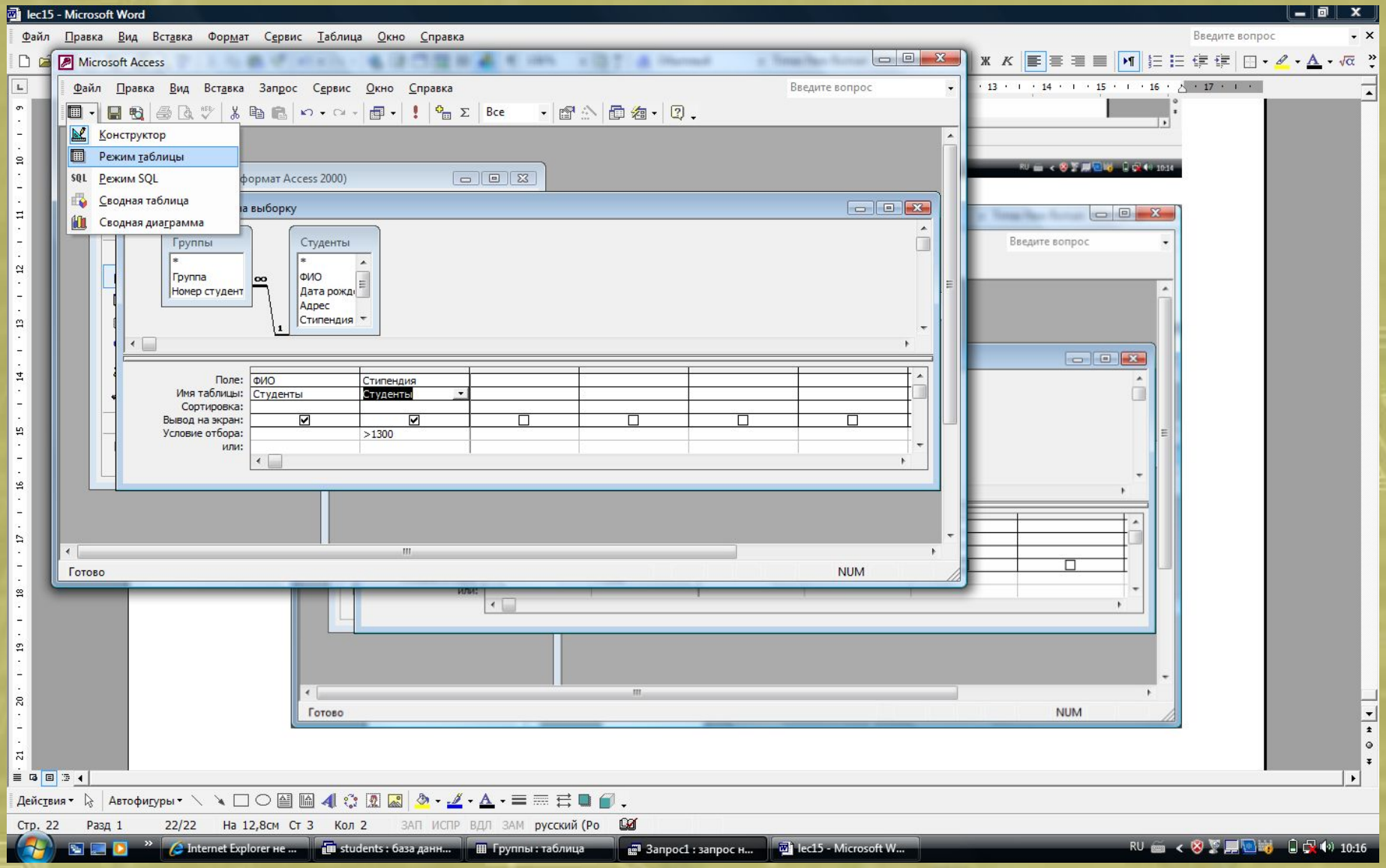
Создание запроса в режиме конструктора



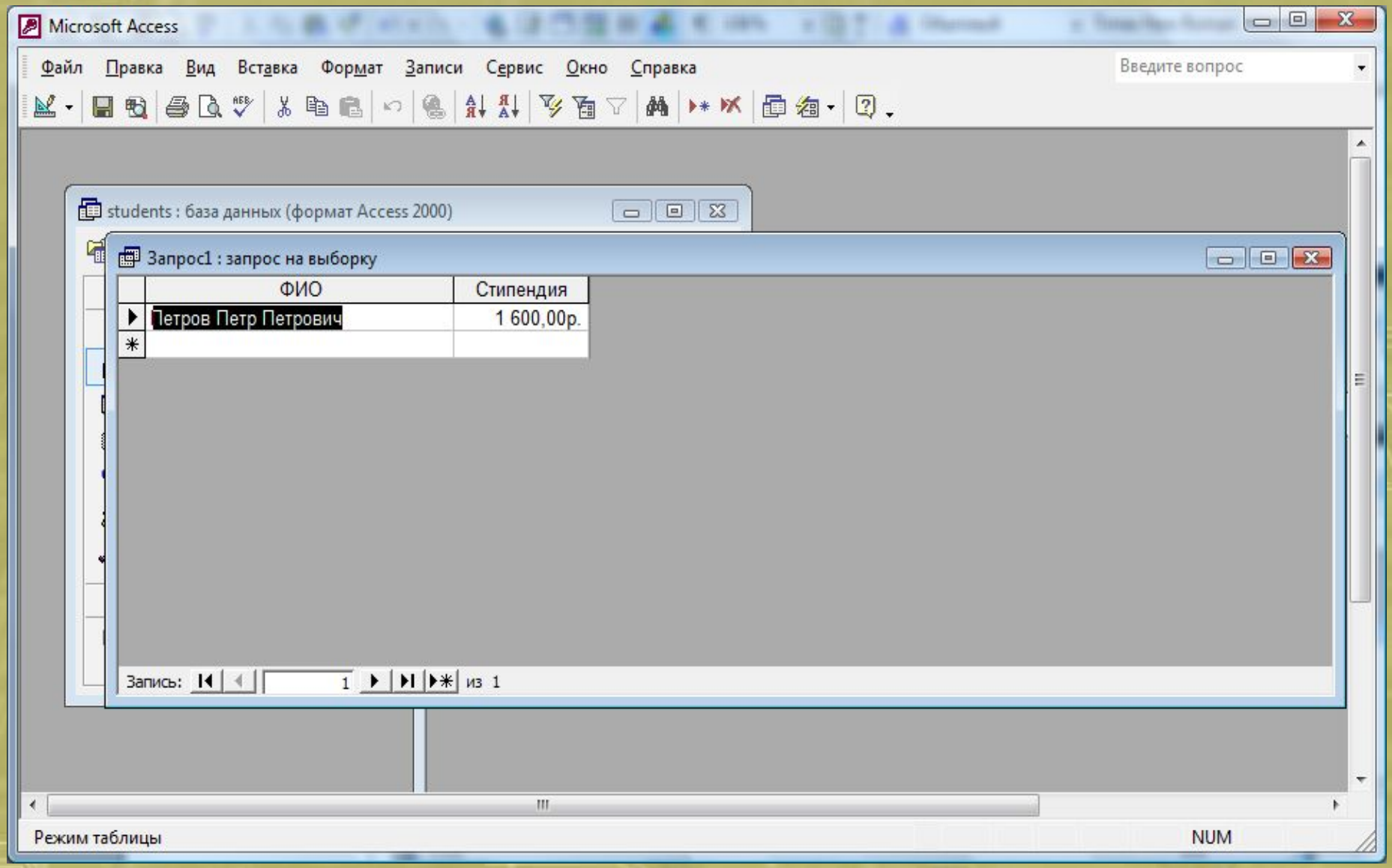
Добавление таблиц, участвующих в запросе



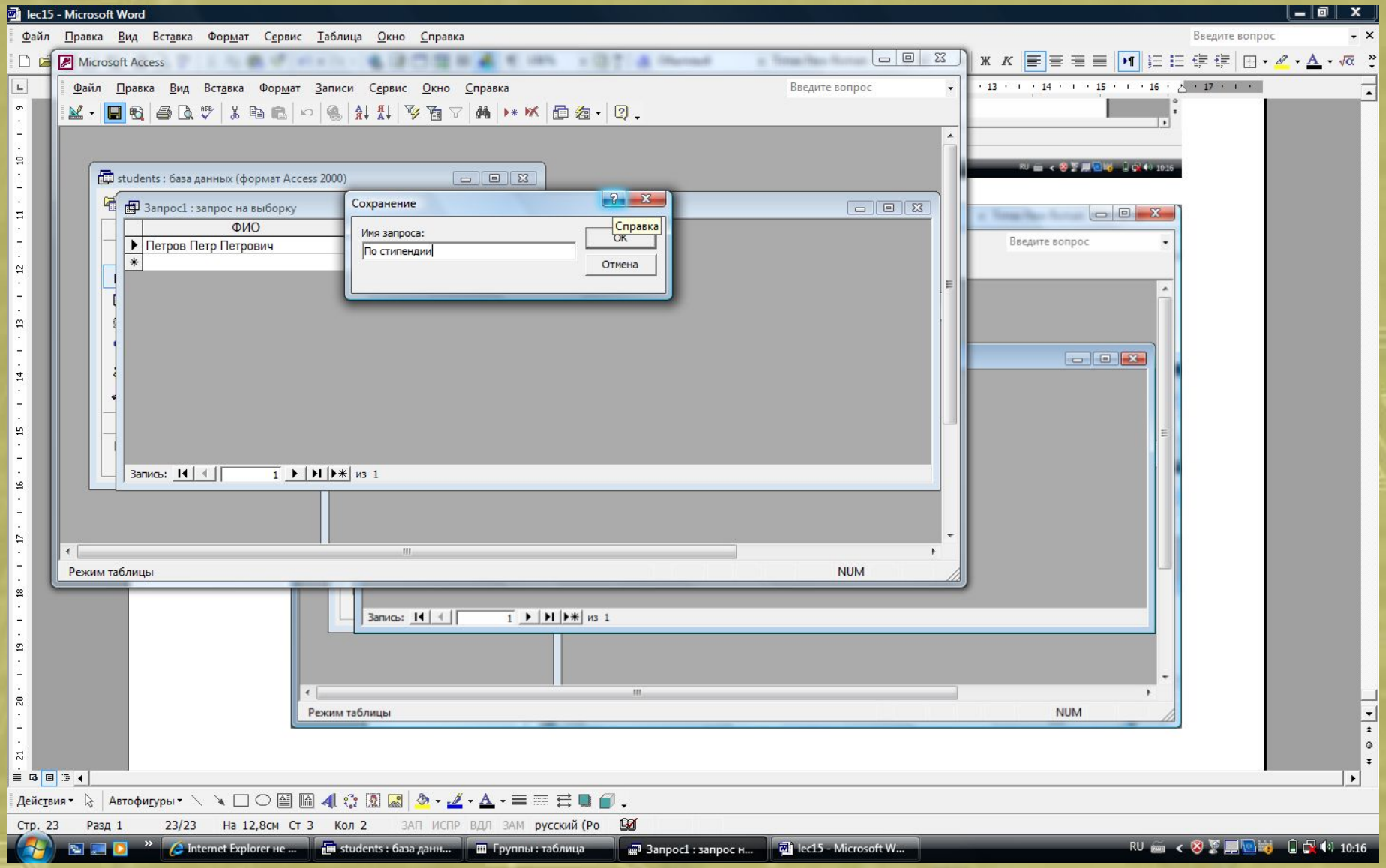
Выбор полей для запроса и установка условий отбора



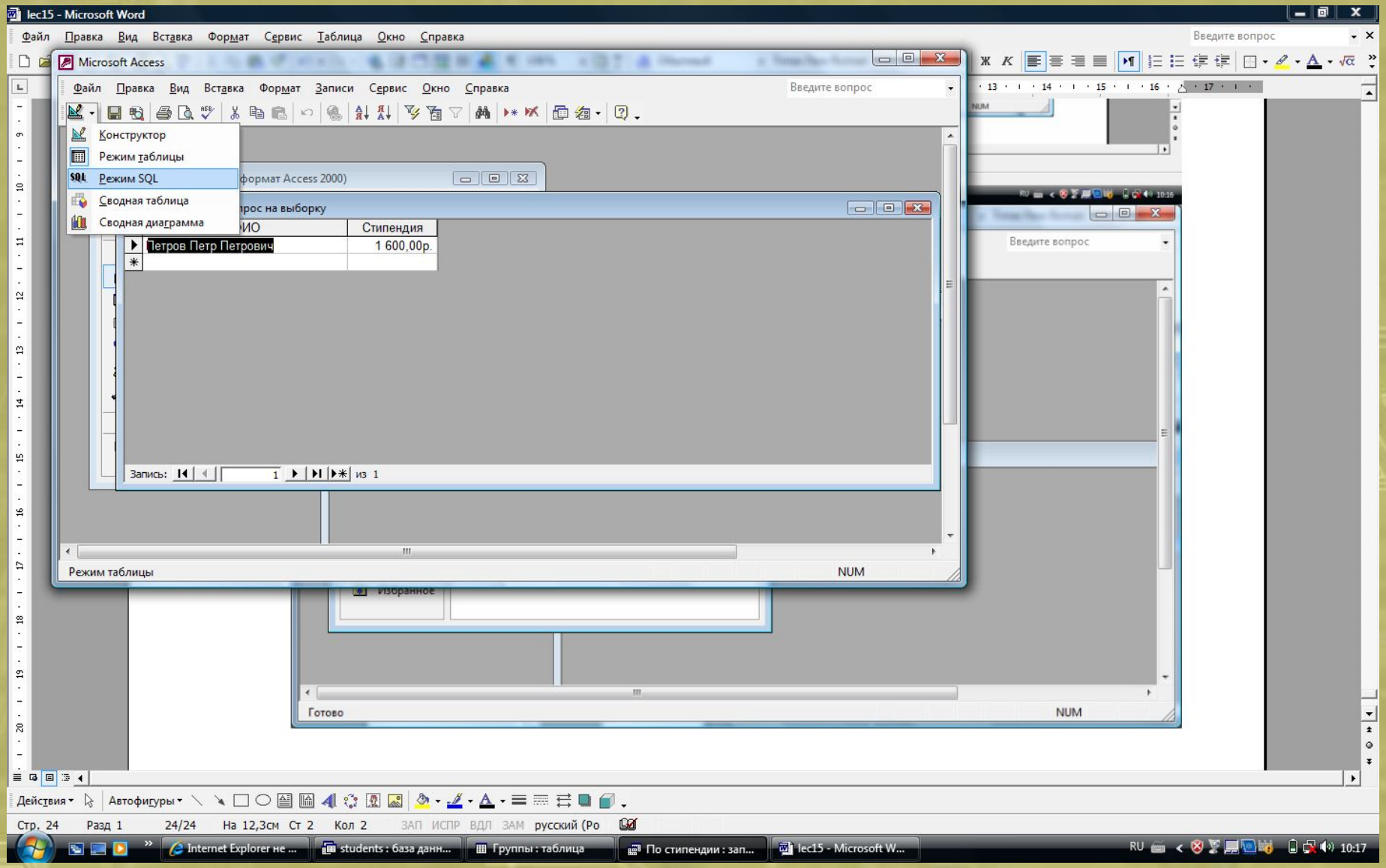
Режимы работы с запросами



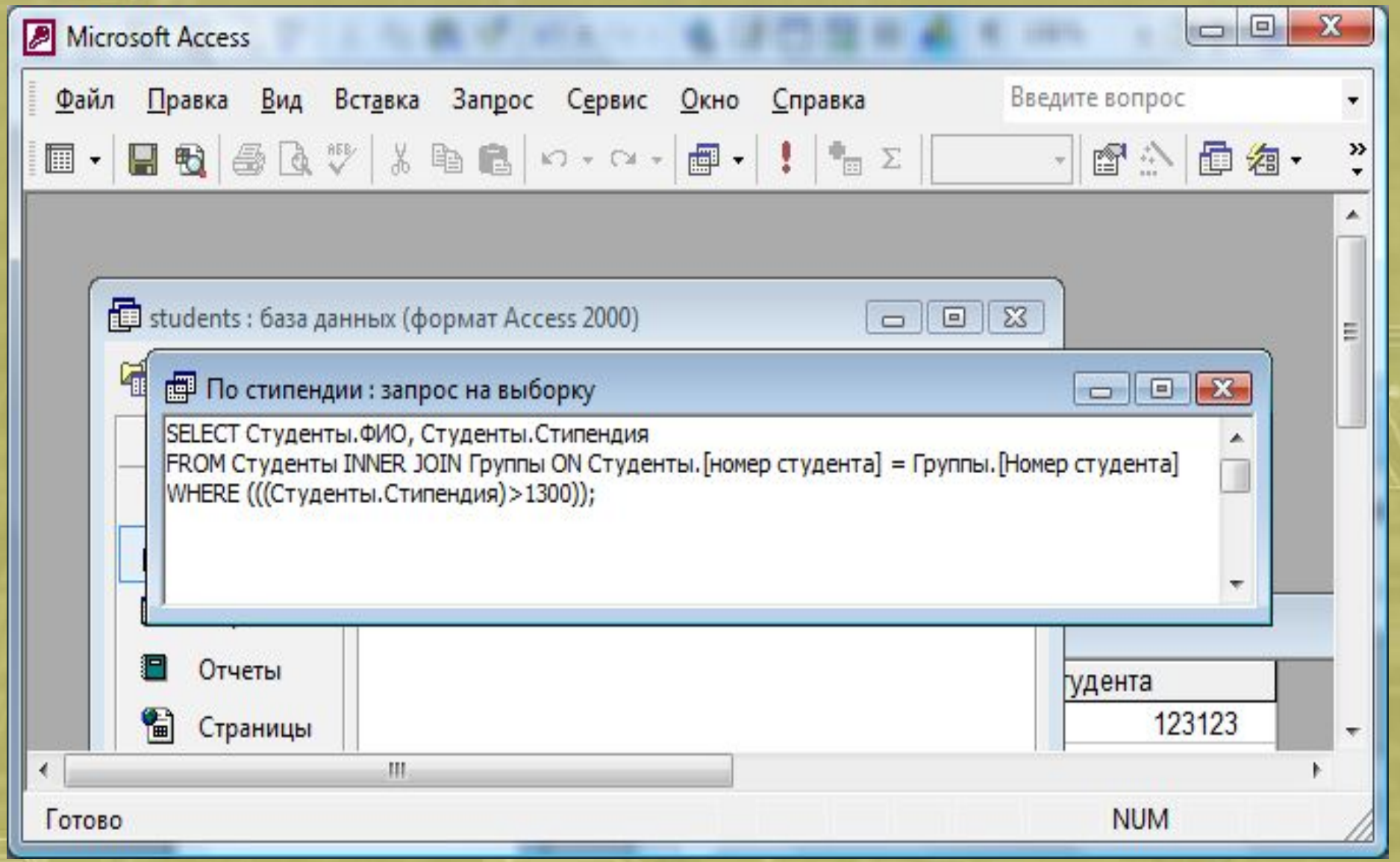
Режим запроса «Таблица» - результат выполнения запроса



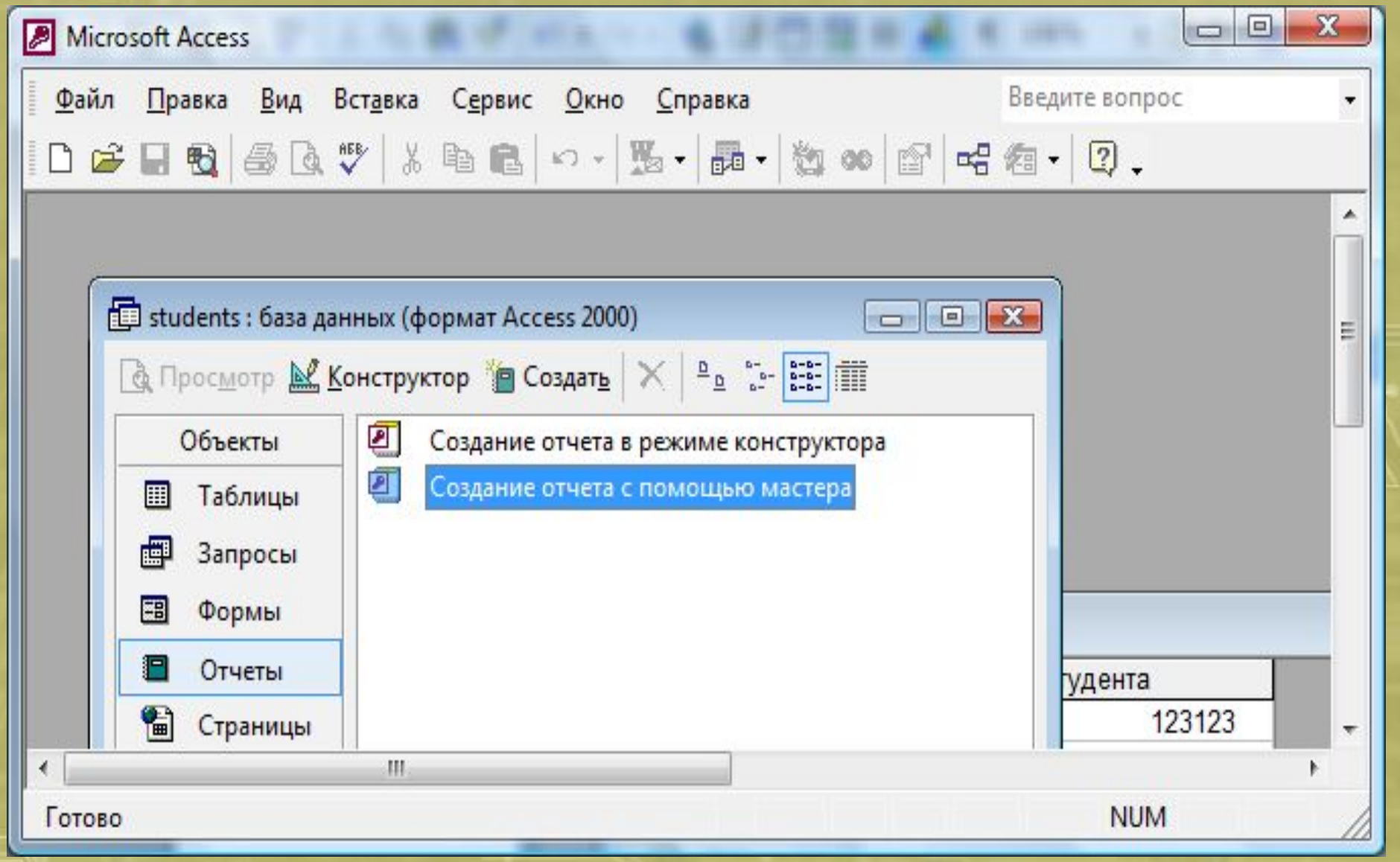
Сохранение запроса под именем «По стипендии»



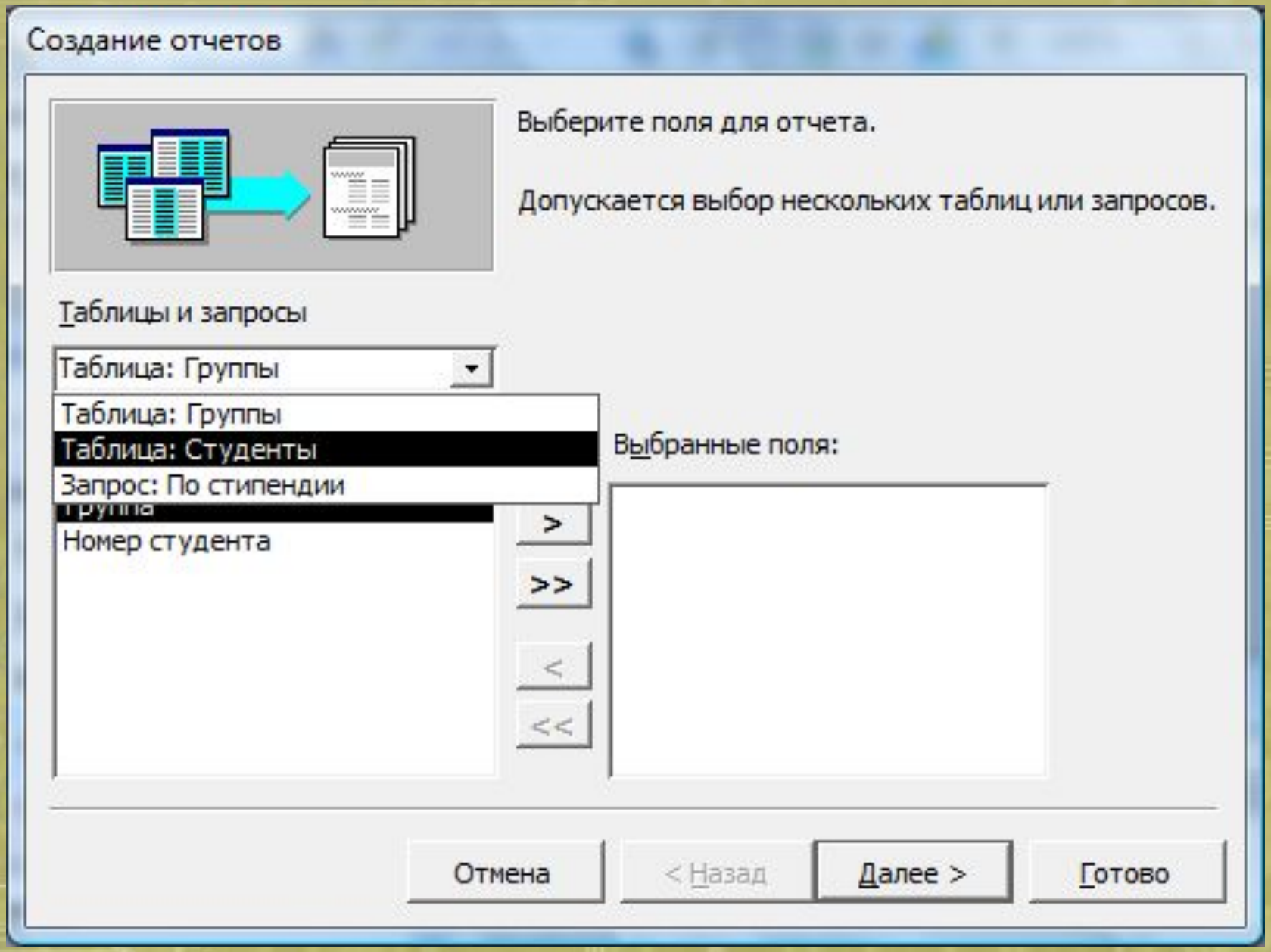
Выбор режима просмотра запроса на языке SQL



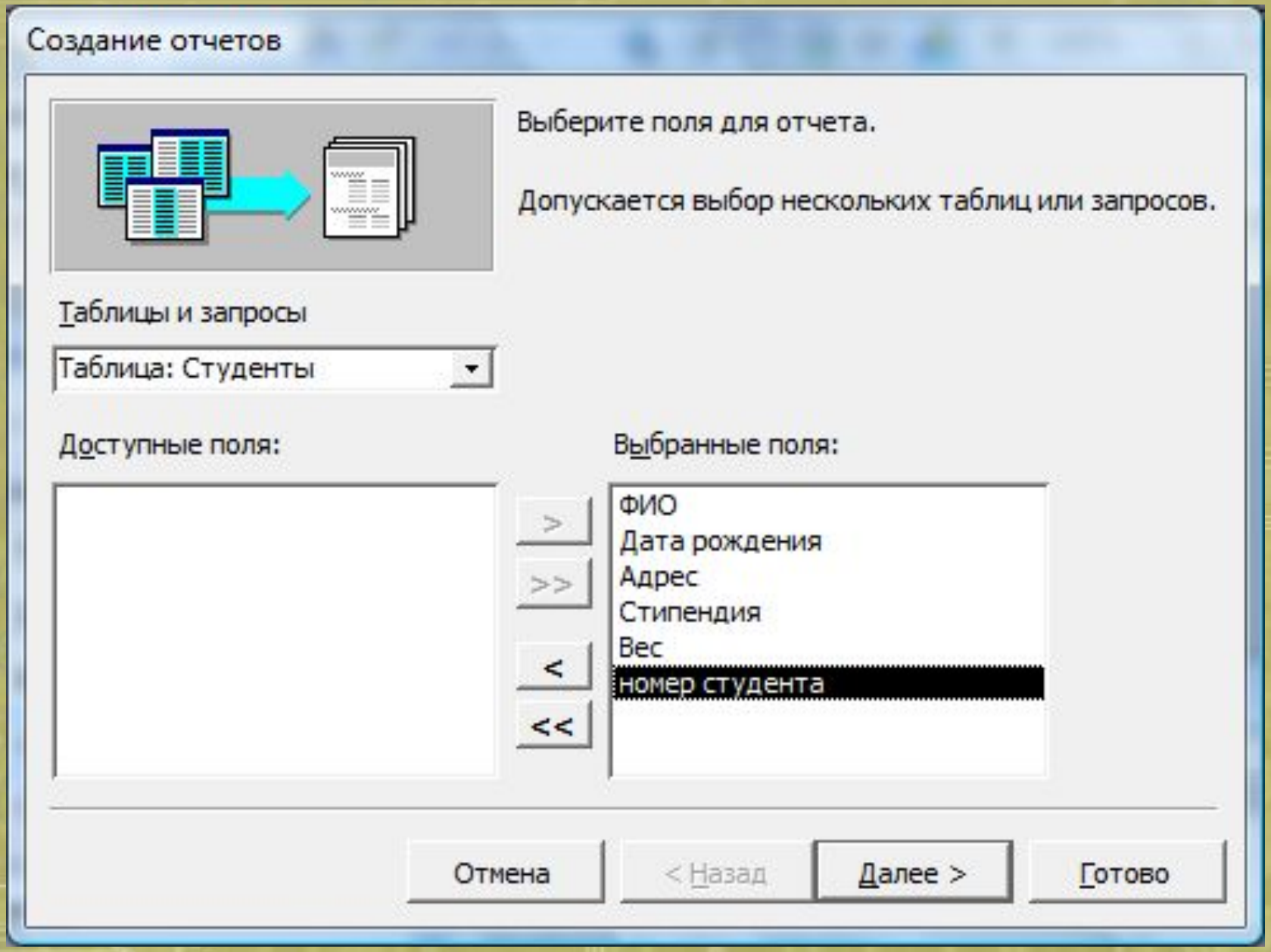
Вид запроса в режиме «SQL» (запрос на языке SQL)



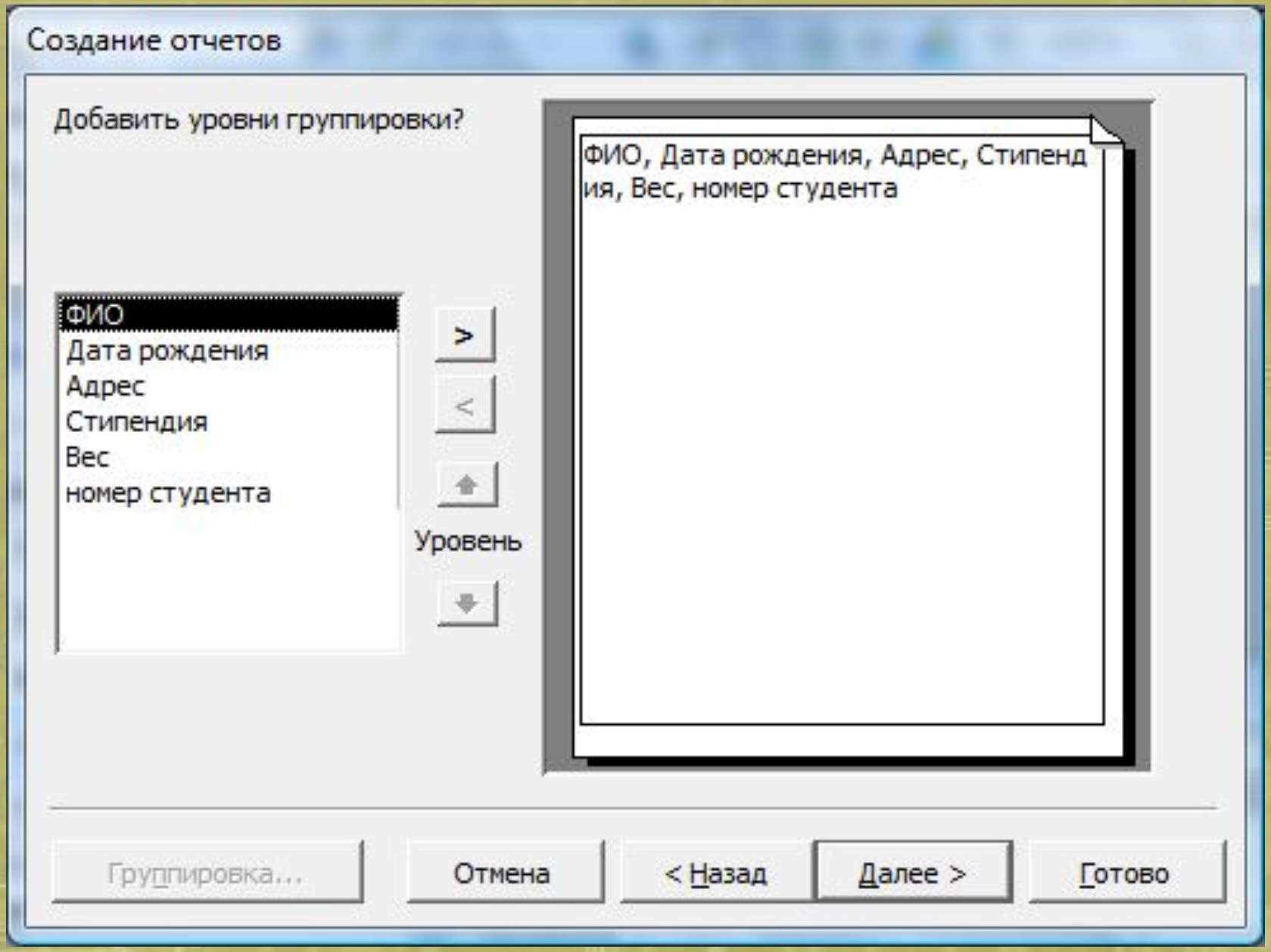
Создание отчета



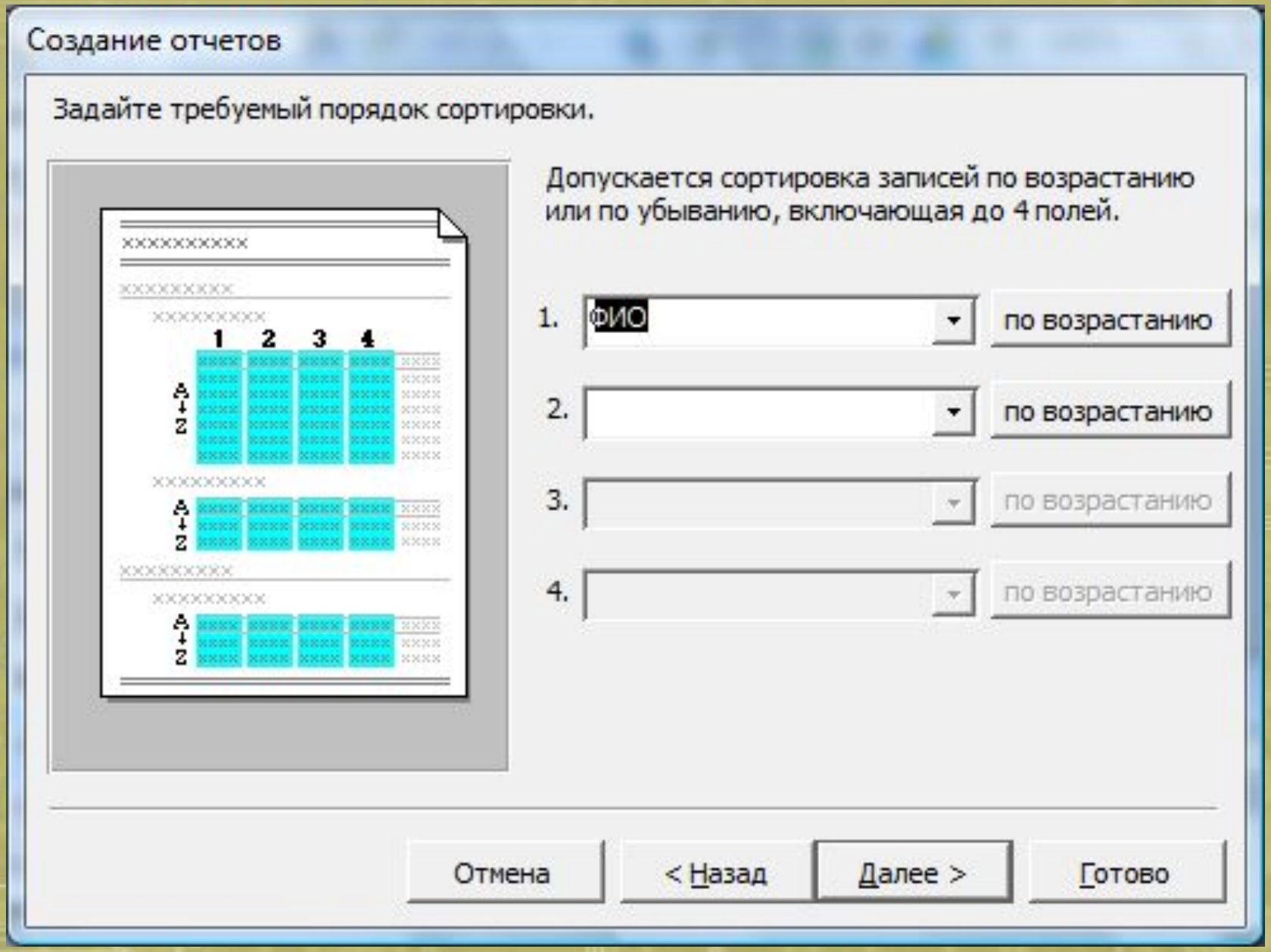
Выбор таблицы и полей для отчета



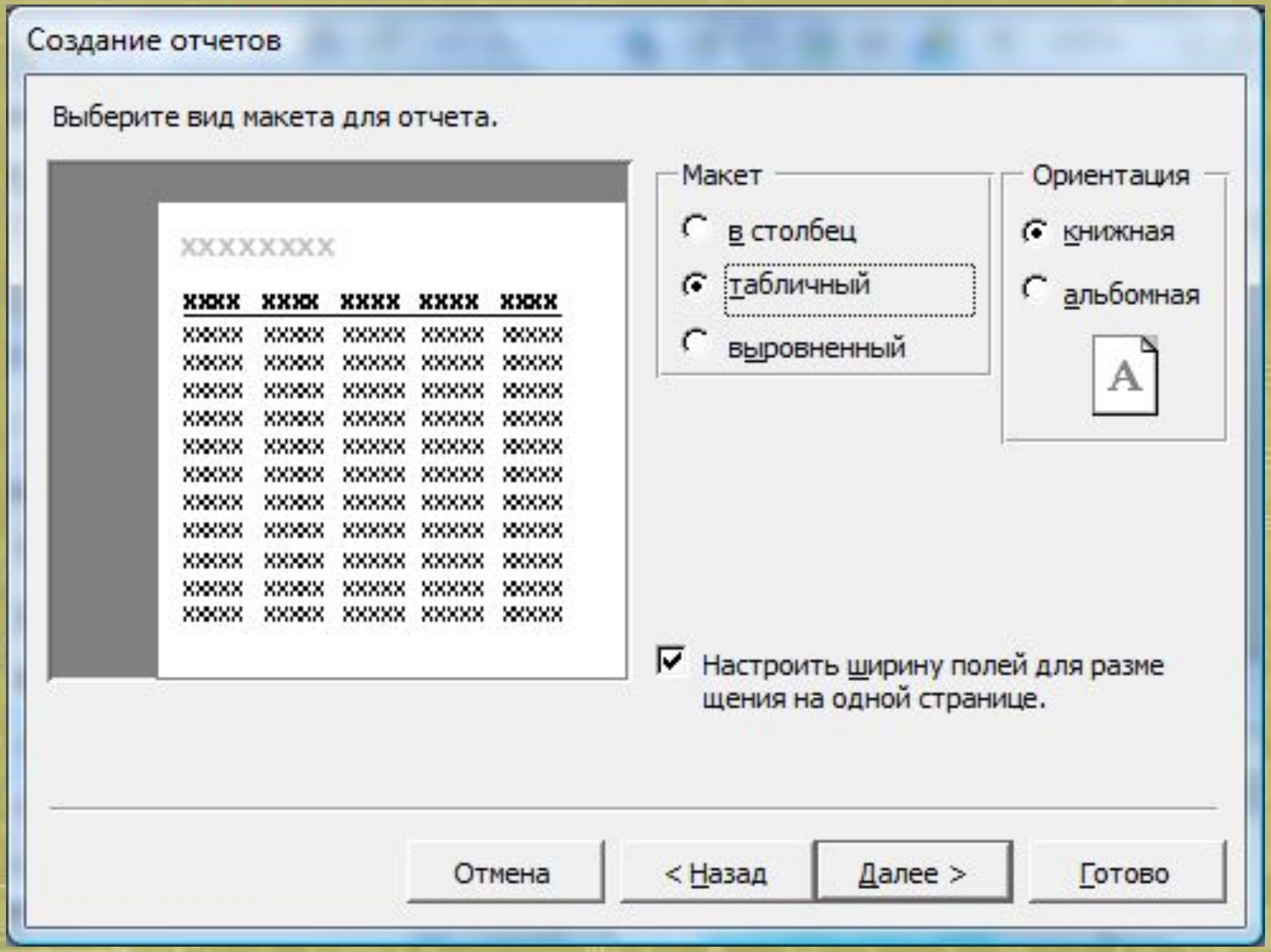
Выбранные поля для отчета в правом столбце



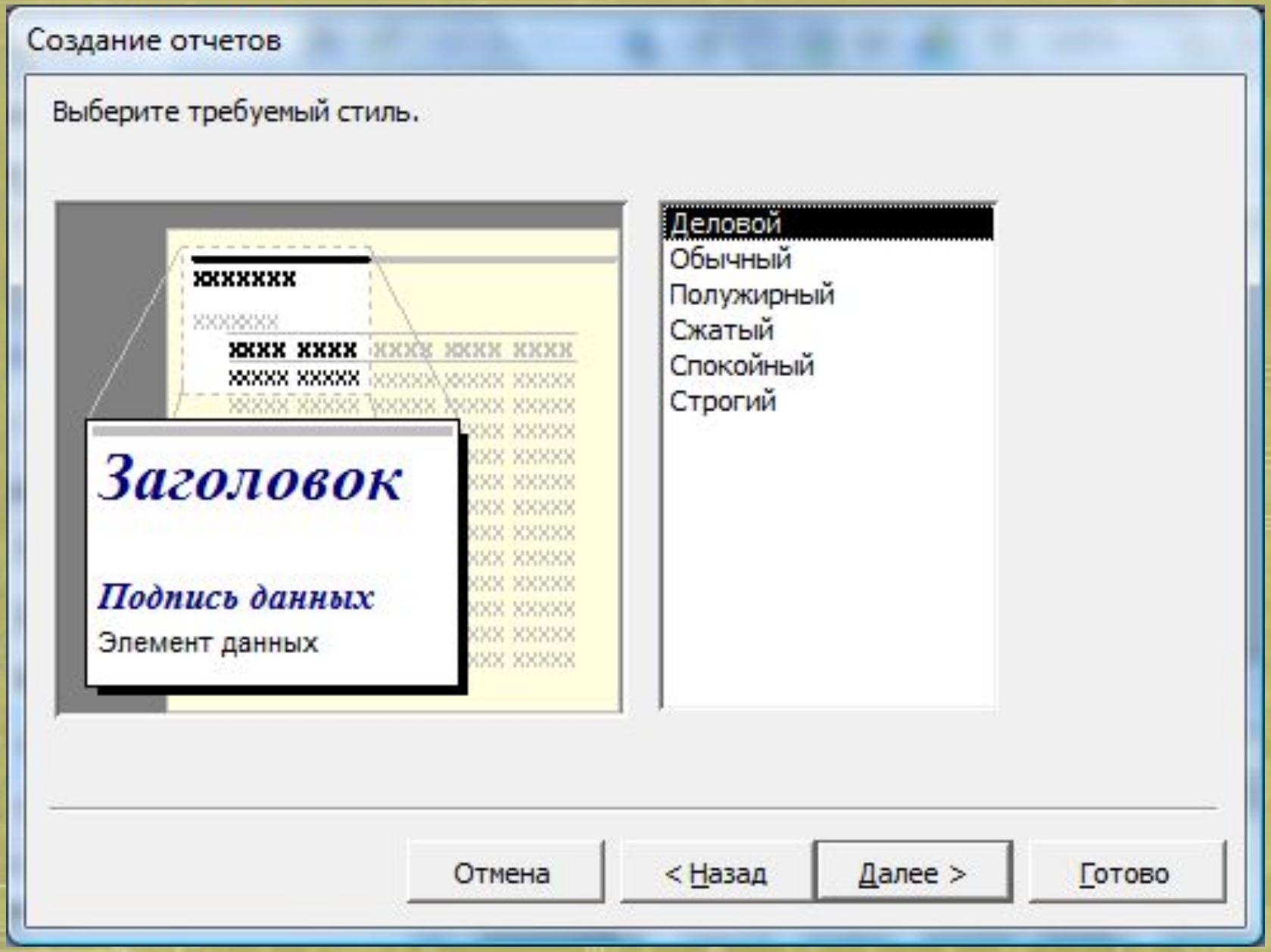
Способы группировки данных в отчете



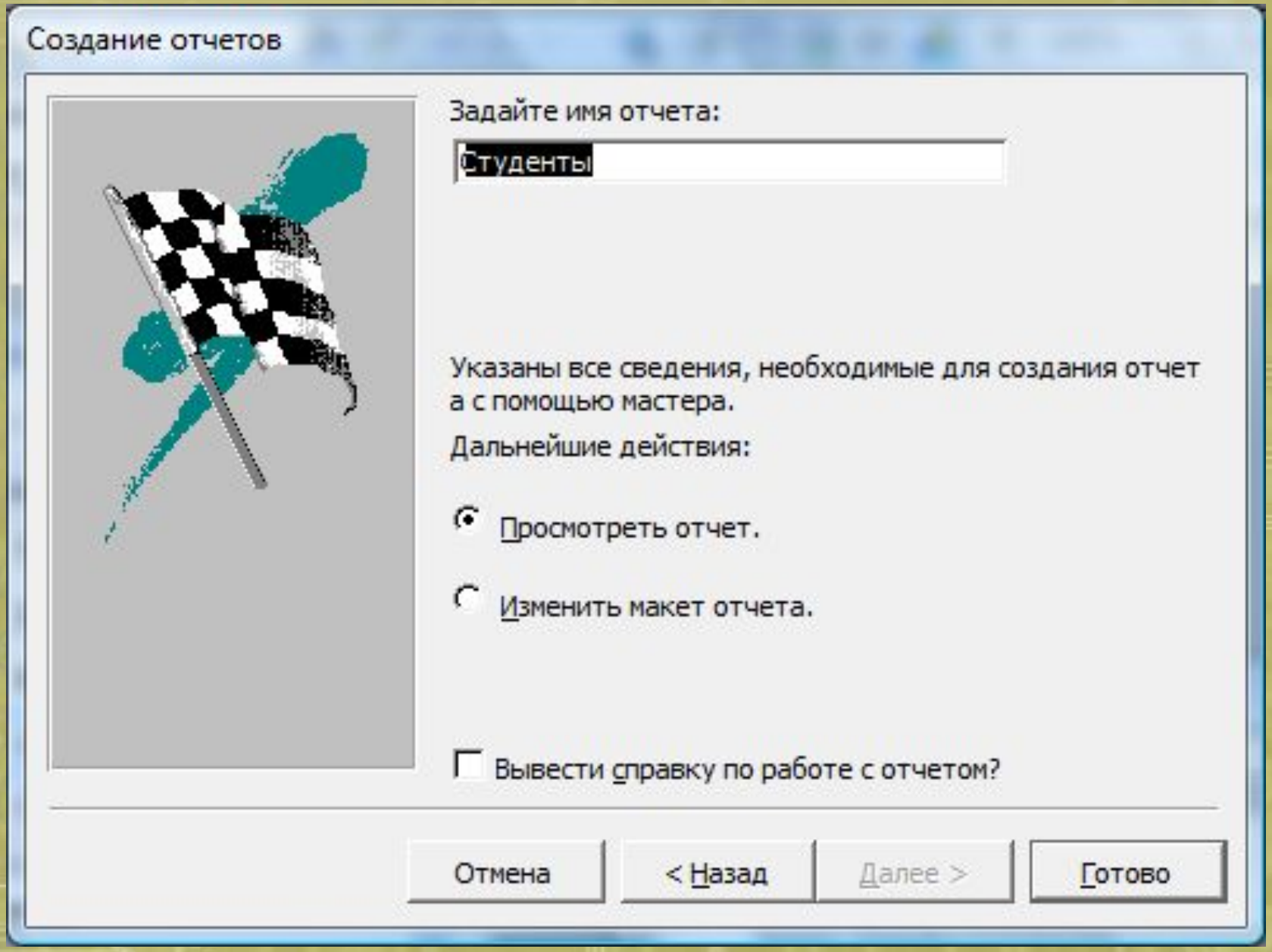
Требуемый порядок сортировки данных в отчете



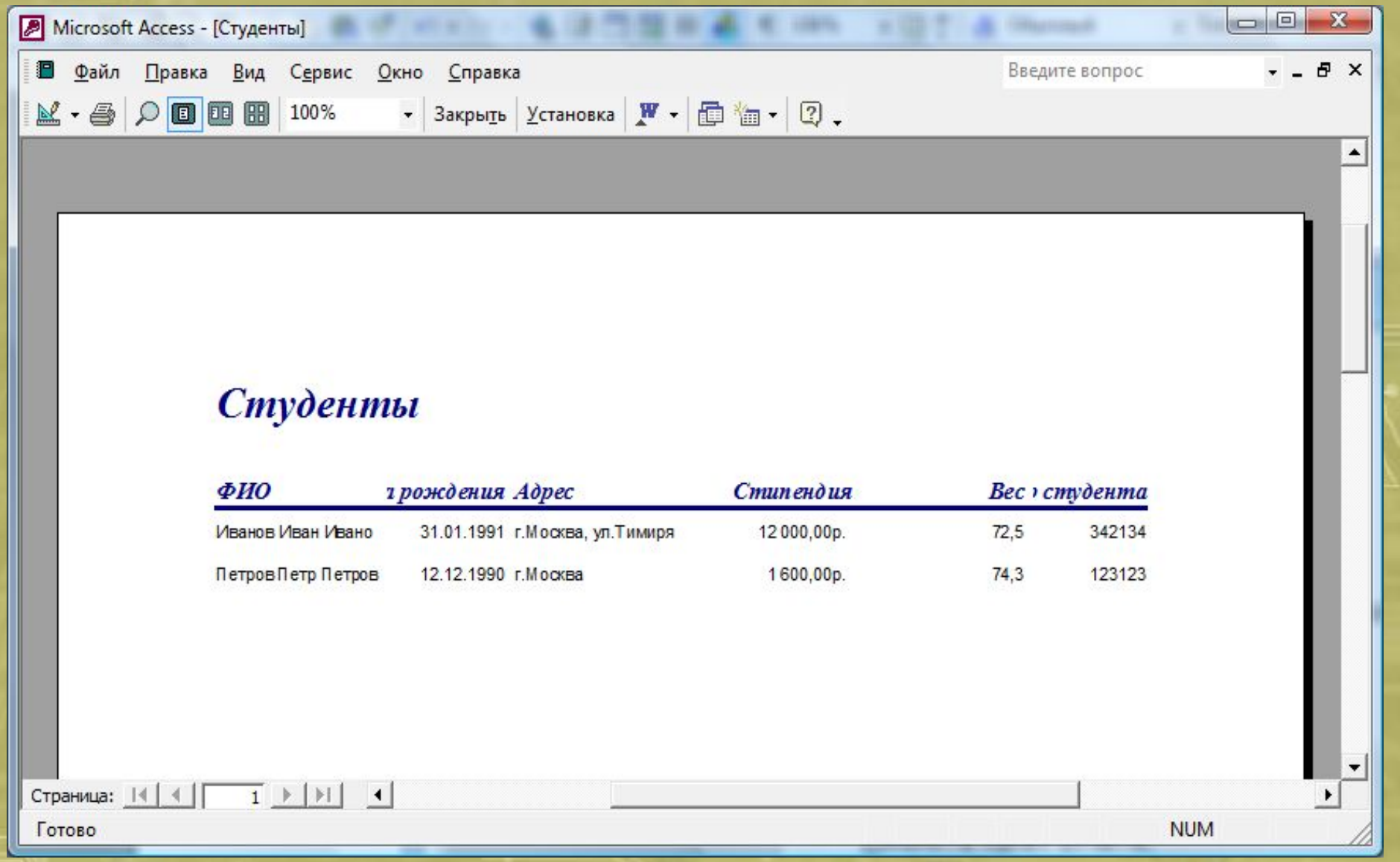
Макет отчета «Табличный»



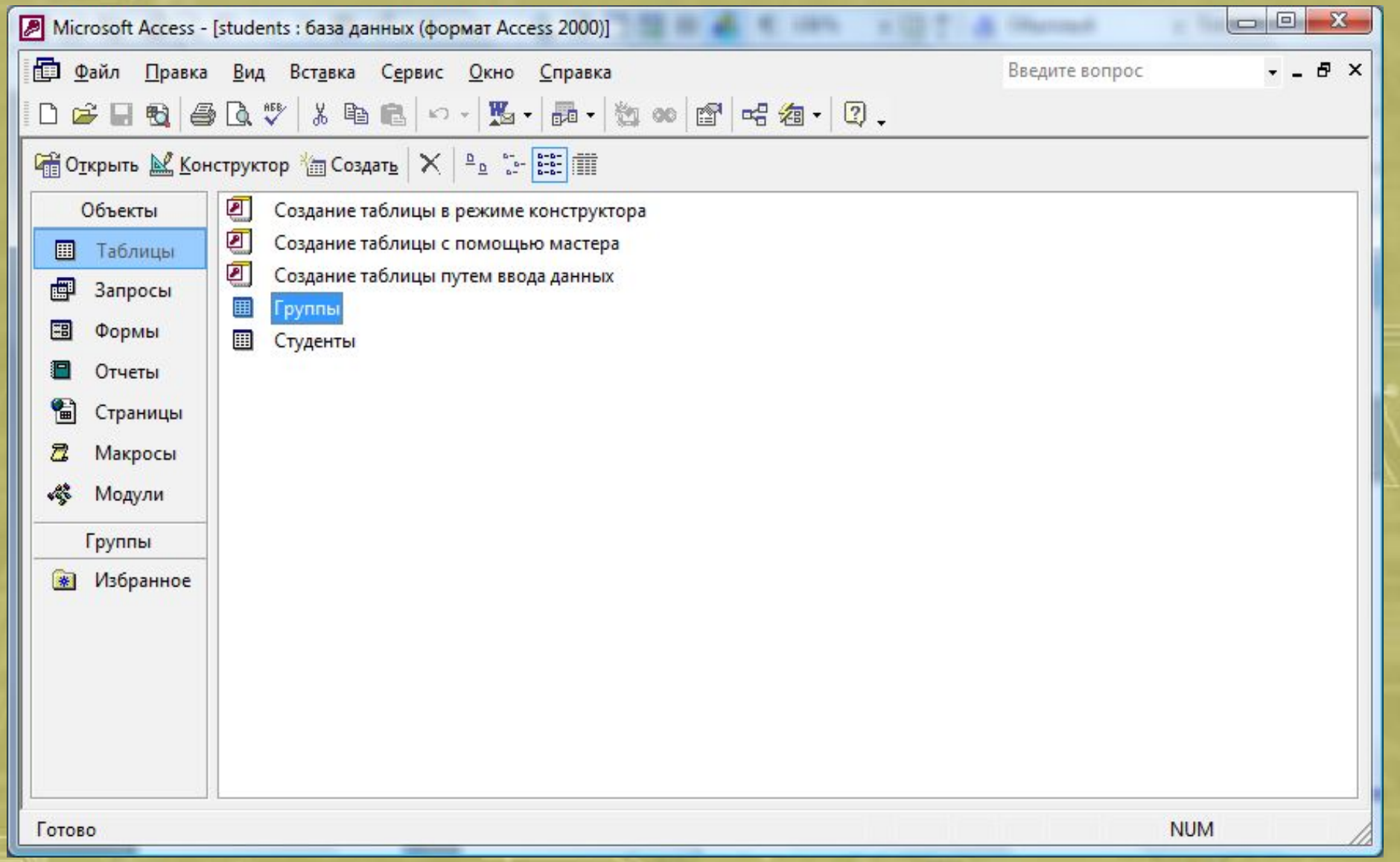
Стиль отчета «Деловой»



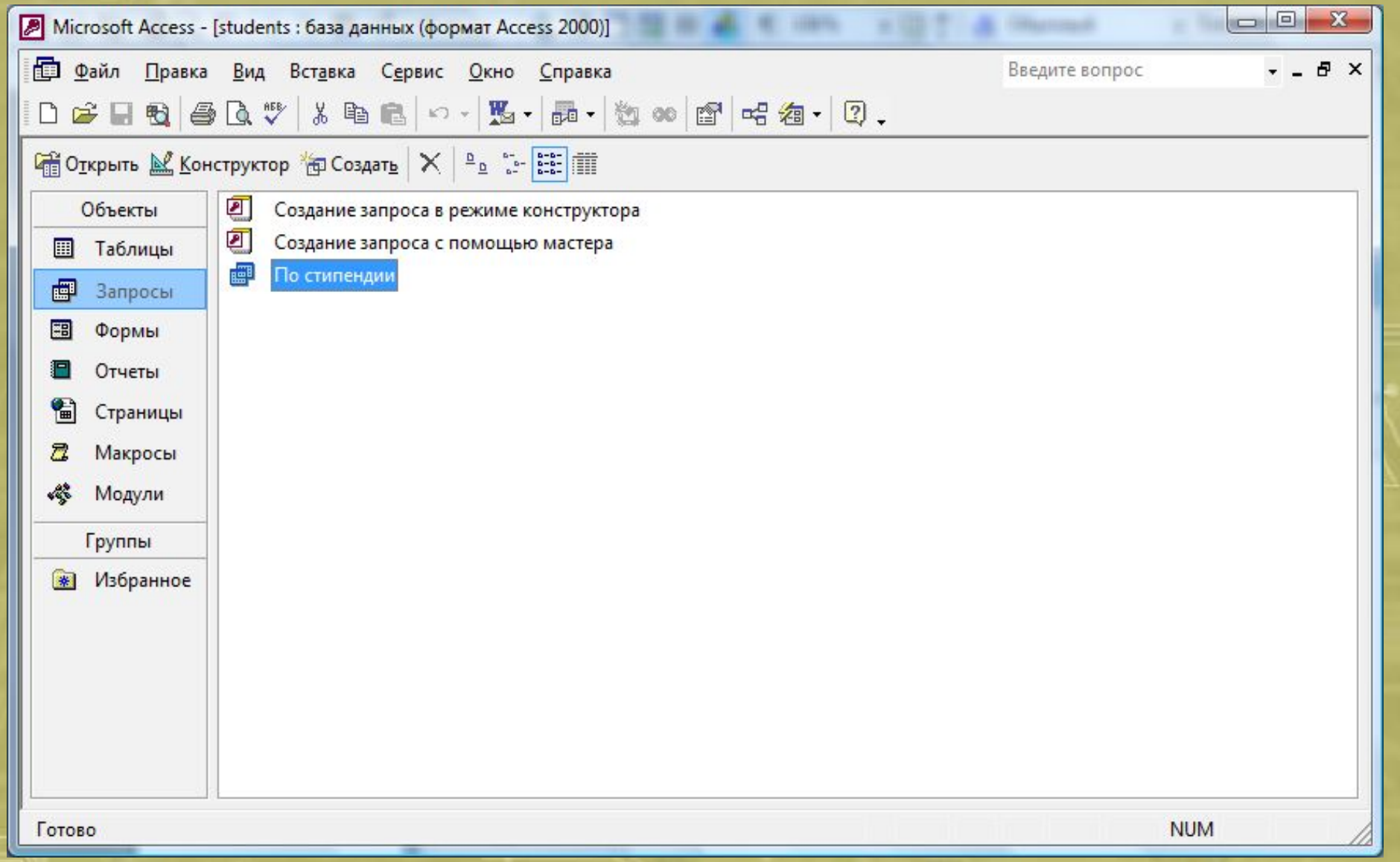
Сохранение отчета под именем «Студенты»



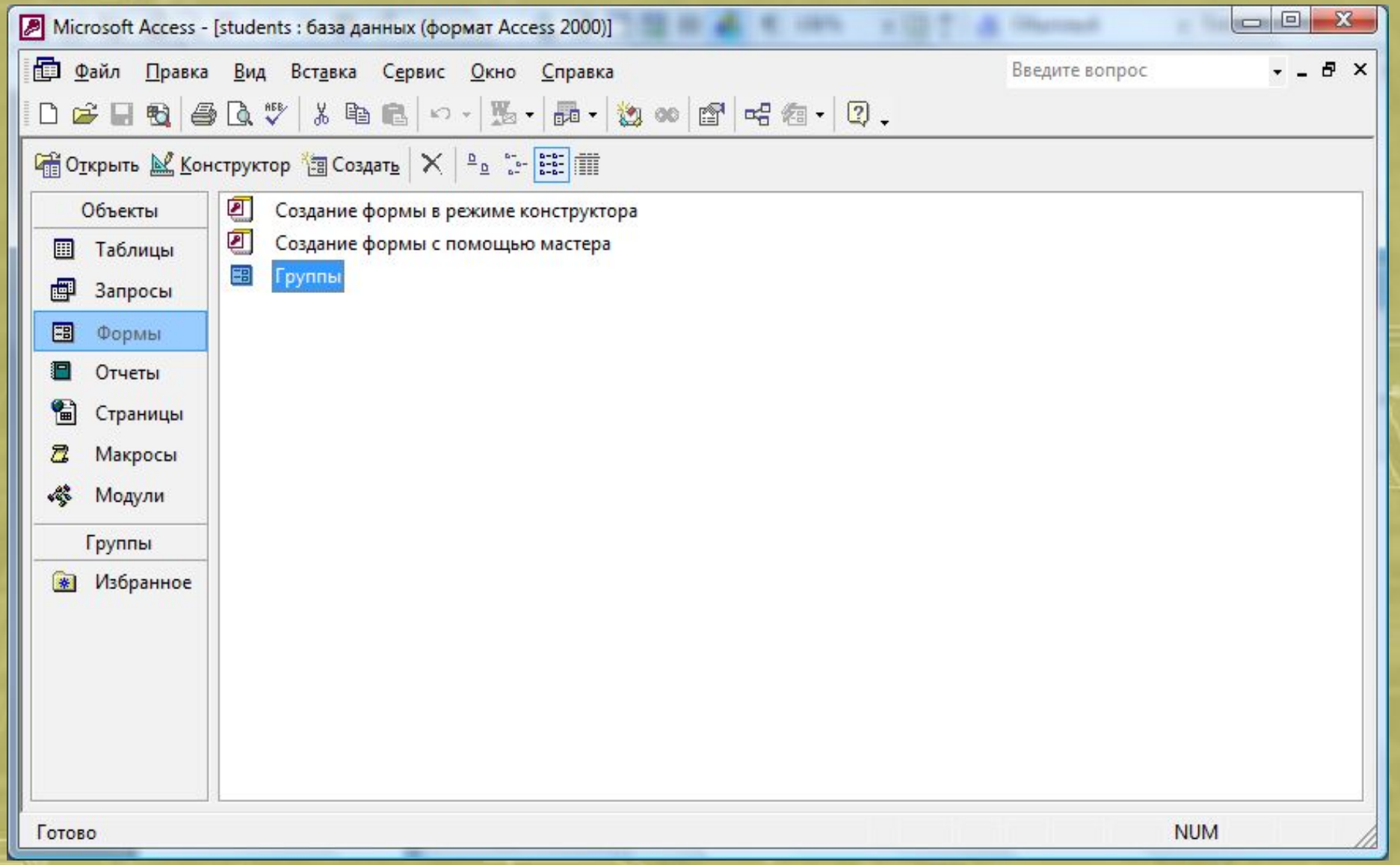
Результат выполнения отчета «Студенты»



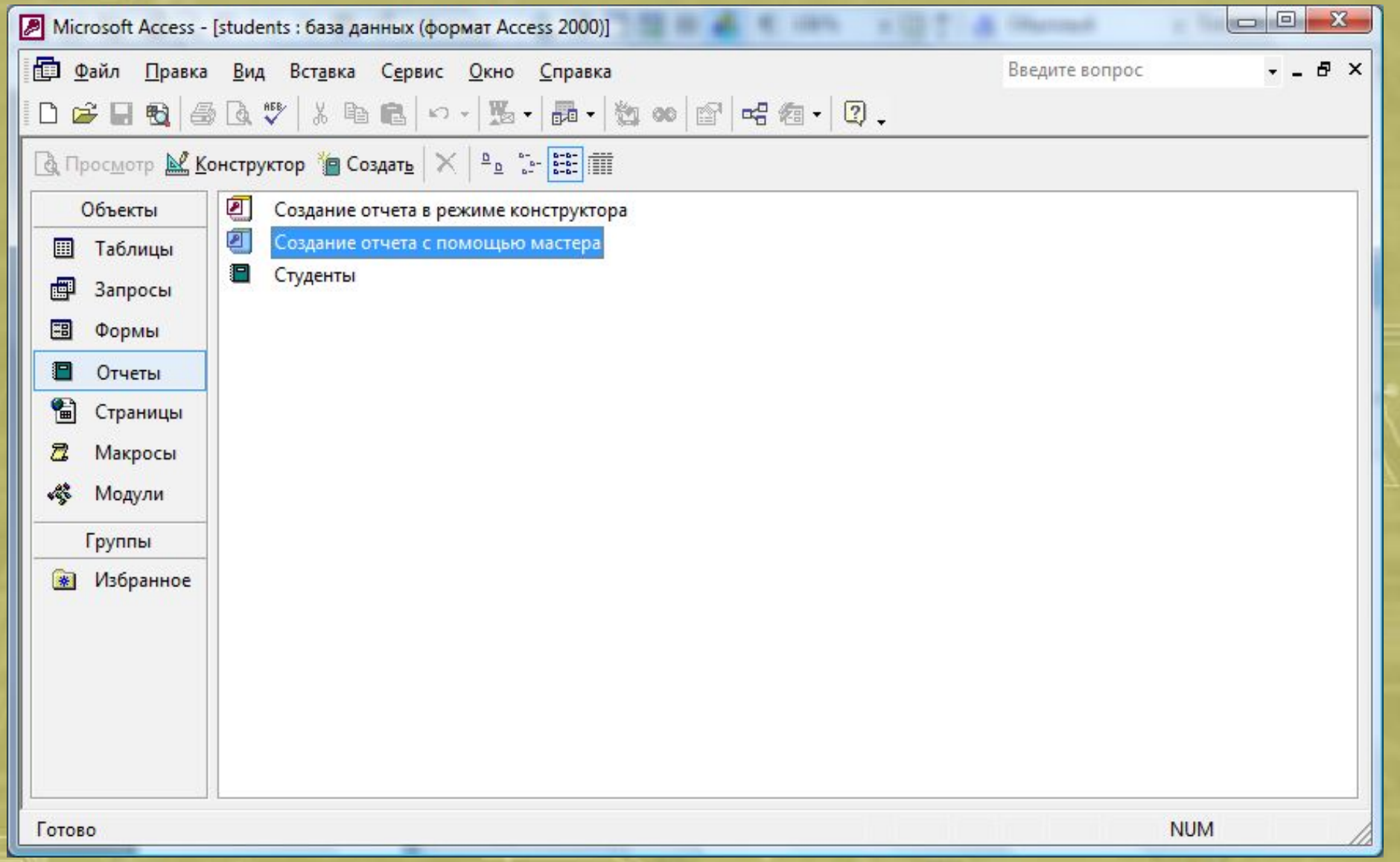
Результат работы. Таблицы: «Группы», «Студенты»



Результат работы. Запрос: «По стипендии»



Результат работы. Форма «Группы»



Результат работы. Отчет «Студенты»

КОНЕЦ

ЛЕКЦИИ