

Частное учреждение образования
«Колледж бизнеса и права»

УТВЕРЖДАЮ
Заведующий
методическим кабинетом
_____ Е.В. Фалей
«___» _____ 2017

Специальность: «Программное обеспечение информационных технологий»	Дисциплина: «Базы данных и системы управления базами данных»
Составлена на основании учебной программы, утвержденной директором Колледжа бизнеса и права 30.12.2016	

Лабораторная работа № 4
Инструкционно-технологическая карта

Тема: Создание БД в полнофункциональной СУБД, создание концептуальной схемы базы данных и наполнение таблиц данными

Цель работы: Познакомиться с системой основных компонентов реляционной СУБД. Научиться проектировать базу данных, создавать концептуальную схему базы данных и наполнять таблицы данными

Время выполнения: 2 часа

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретическую часть настоящей инструкционно-технологической карты.
2. Рассмотреть методику создания базы данных в SQL Server, методику создания концептуальной схемы базы данных, методику наполнения таблиц базы данных данными, описанные в разделе «Пример выполнения работы» настоящей инструкционно-технологической карты.
3. Получить у преподавателя индивидуальное задание и выполнить создание базы данных, создание концептуальной схемы базы данных, наполнить таблицы базы данных данными необходимой информацией (не менее 10 записей в родительских таблицах и не менее 20 записей в дочерних таблицах).
4. Ответить на контрольные вопросы.

1. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Прежде чем ознакомиться с методами создания отношений в SQL Server Management Studio нужно изучить типы данных, которые поддерживает SQL Server. SQL Server поддерживает множество типов данных, каждый из которых имеет свое назначение и предназначен для использования в опреде-

ленных случаях для определенных атрибутов. SQL Server не ограничивается стандартными типами данных и предоставляет возможность создавать свои, назначать им имена и использовать в определении отношений. Следует с особой осторожностью подбирать тип данных для атрибута, так как многие типы данных имеют схожую структуру и как следствие могут использоваться не по назначению. Например, если атрибут используется для хранения имен, то подходящим для него типом данных будет строка (последовательность символов). Но SQL Server предлагает несколько форматов строк, и, если нет на то особой надобности, не нужно использовать строки Unicode (2 байта на символ) для хранения имени. Ведь в отношении может содержаться большое количество строк и в результате может появиться ненужная избыточность данных. Такая же ситуация имеет место когда, например, необходимо определить атрибут для хранения значений от 0 до 100 и используется для атрибута 4-х байтовое целое число. Ясно, что для хранения значения 100 хватит и одного байта.

Прежде, чем перейти к обсуждению типов данных, необходимо привести простое правило, используя которое можно правильно подобрать тип данных для столбца:

При выборе типа данных для столбца следует отдавать предпочтение типу, который позволит хранить любые возможные для этого столбца значения и занимать при этом минимальное место на диске.

Типы данных SQL Server подразделяют на семь основных категорий, которые представлены в таблице 1.

Таблица 1 – Категории типов данных

Категория типа данных	Описание
Точный числовой	Хранит точные числовые значения с десятичными точками или без них
Приблизженный числовой	Хранит числовые значения с десятичными точками или без них
Денежный	Хранит числовые значения с десятичными точками; используется специально для денежных значений, максимальная точность составляет четыре знака после запятой
Дата и время	Хранит информацию о дате и времени и выполняет хронологическую проверку, например значение 30 февраля будет забраковано
Символьный	Хранит символьные значения переменной длины
Двоичный	Хранит данные в строго двоичном представлении (0 и 1)
Специальный	Составные типы данных, требующие специальной обработки, например XML-документы или глобально уникальные идентификаторы (GUID)

Точные числовые типы данных

Точные числовые типы данных используют для хранения чисел, как с десятичными точками, так и без них. К таким числам можно применять любые математические операции – никакая специальная обработка для этого не требуется. Объем занимаемой памяти также строго определен, поэтому данные, хранящиеся в этих типах, имеют одинаковые значения как в архитектуре процессора Intel, так и в архитектуре AMD. В таблице 2 приведен список точных числовых типов данных, поддерживаемых в SQL Server.

Таблица 2 – Числовые типы данных

Тип данных	Занимаемая память	Диапазон значений	Назначение
<i>bigint</i>	8 байт	от -2E63 до 2E63 - 1	Хранение очень больших целых чисел — как положительных, так и отрицательных
<i>int</i>	4 байта	от -2E31 до 2E31 - 1	Хранение положительных и отрицательных целых чисел
<i>smallint</i>	2 байта	от -32 768 до 32 767	Хранение положительных и отрицательных целых чисел
<i>tinyint</i>	1 байт	от 0 до 255	Хранение небольшого диапазона положительных целых чисел
<i>decimal(p,s)</i>	от 5 до 17 байт в зависимости от точности	от -10E38 + 1 до 10E38 - 1	Хранение десятичных чисел с максимальным числом разрядов
<i>numeric(p,s)</i>	от 5 до 17 байт в зависимости от точности	от -10E38 + 1 до 10E38 - 1	По своим функциям эквивалентен типу <i>decimal</i> . Типы <i>decimal</i> и <i>numeric</i> взаимозаменяемы

Типы данных *decimal* и *numeric* принимают параметры, позволяющие завершить определение типа данных. Эти параметры задают точность и масштаб типа данных. Так, *decimal(12,4)* определяет десятичное значение, которое может содержать до 12 разрядов, причем четыре из них — после запятой.

Самые распространенные типы данных из этой категории — *int* и *decimal*. Тип данных *decimal* можно использовать и для хранения целочисленных значений, но тогда для каждой строки потребуются дополнительные байты. Поэтому так делать не следует.

Хотя тип данных *int* может хранить как положительные, так и отрицательные числа, отрицательная часть диапазона используется редко. Типы данных *int* нашли широкое, но часто не самое лучшее применение. Если диапазон значений, который нужно хранить в столбце, не превышает 32 767, использование *smallint* вместо *int* позволяет сэкономить два байта для каждой

строки. Если же значения лежат в диапазоне от 0 до 255, тип *tiny int* сэкономит три байта в каждой строке.

Приближенные числовые типы данных

В приближенных числовых типах данных можно хранить десятичные значения. Однако точность данных, хранимых в типах *float* или *real*, ограничивается значением, указанным в определении типа данных. Точность хранения дробной части числа не гарантируется. Например, если для хранения значения 1.00015454 используется тип данных *float(8)*, при запросе столбец гарантированно вернет лишь значение 1.000154. При хранении данных SQL Server округляет все разряды справа. Поэтому вычисления, в которых используют эти типы данных, содержат ошибки округления. Перемещение баз данных с таблицами, содержащими эти типы данных, между платформами Intel и AMD также приведет к ошибкам. В таблице 3 перечислены приближенные числовые типы данных, поддерживаемые в SQL Server.

Таблица 3 – Приближенные числовые типы данных

Тип данных	Занимаемая память	Диапазон значений	Назначение
<i>float(p)</i>	4 или 8 байт	от -2.23E308 до 2.23E308	Хранение больших чисел с плавающей точкой, которые выходят за границы диапазона типа данных <i>decimal</i>
<i>real</i>	4 байта	от -3.4E38 до 3.4E38	Все еще поддерживается, но заменен на <i>float</i> чтобы удовлетворить требованиям стандарта SQL-92

Параметр типа данных *float* позволяет задать гарантированно хранимое число разрядов. Например, в столбце *float(8)* будет храниться в точности семь разрядов, а разряды ниже седьмого приведут к ошибкам округления.

Из-за неточности, присущей этим типам данных, их редко используют. Применять тип *float* следует только тогда, когда хранимые значения выходят за пределы диапазонов точных числовых типов данных.

Денежные типы данных

Денежные типы данных созданы для хранения денежных значений с четырьмя десятичными разрядами после запятой. В таблице 4 представлены денежные типы данных, поддерживаемые в SQL Server.

Таблица 4 – Денежные типы данных

Тип данных	Занимаемая память	Диапазон значений	Назначение
<i>money</i>	8 байт	от -922 337 203 685 477.5808 до 922 337 203 685 477.5807	Хранение больших денежных значений
<i>small-money</i>	4 байта	от -214 748.3648 до 214 748.3647	Хранение небольших денежных значений

Тип данных *smallmoney* достаточно редко встречается в базах данных, хотя он лучше всего подходит для большинства приложений, работающих с товарами и заказами. Гораздо чаще в базах данных некорректно используется тип *money*, что приводит к потере четырех байтов памяти на каждую строку таблицы.

Хотя типы данных *money* и *smallmoney* предназначены для хранения денежных значений, их редко применяют в финансовых приложениях. В этих приложениях нужно выполнять расчеты с точностью до 6, 8 и даже 12 знаков после запятой, поэтому вместо денежных типов используется *decimal*.

Типы данных «Дата и время»

При хранении данных ничто не вызывает таких разногласий, как способ хранения даты и времени. В некоторых приложениях нужно хранить только дату, в других — только время, а в третьих — и то и другое.

Datetime. Этот тип данных способен хранить любую дату и время с 1 января 1753 года до 31 декабря 9999 года. Тем не менее, вместе с датой храниться и время. Если задается только дата, то время принимает по умолчанию значение 12:00:00.

Datetime2. Как и тип данных *datetime*, *datetime2* используется для хранения даты и времени. Но *datetime2* способен хранить более точные значения времени. Формат хранения данных: YYYY-MM-DD hh:mm:ss[.nnnnnnnn].

Smalldatetime. Тип данных *smalldatetime* схож с типом *datetime*, но хранит меньших диапазон дат: с 1 января 1900 года до 6 июня 2079 года. *Smalldatetime* имеет различную с *datetime* двоичную структуру.

Символьные типы данных

Для хранения символьных данных служат специально предназначенные для этого типы. В этих типах один символ занимает в памяти один или два байта в зависимости от используемой кодировки — ANSI или Unicode.

Прежде чем перейти к символьным типам данных, кратко рассмотрим происхождение кодировок ANSI и Unicode. Для работы со всевозможными языками общения компьютерным специалистам был необходим стандартный формат хранения множества различных языковых символов. Поэтому в Американском национальном институте стандартов (ANSI) был разработан стандарт кодировки, в котором для представления диапазона букв требовалось восемь бит. Единственная проблема заключалась в том, что в одной восьми битной кодировке нельзя было определить все существующие символы. Поэтому были созданы десятки наборов символов, содержащих допустимые для данной кодировки символы. Такой подход хорошо работал до тех пор, пока не потребовалось передавать данные между системами, использующими разные наборы символов. Если символа из одной кодировки не было в другой, в процессе преобразования он терялся. Помимо проблем с преобразованием кодировок, восьми битная кодировка была неспособна охватить все символы некоторых языков.

Эти проблемы привели к созданию стандарта Unicode. В этом стандарте для представления каждого символа используются два байта. Это дополнительное пространство позволяет отказаться от наборов символов стандарта ANSI. Теперь все символы могли быть выражены в единой схеме кодировки. А поскольку в Unicode есть только одна схема кодировки, отпадает необходимость преобразования кодировок при передаче данных между системами, в которых используются разные языки. Благодаря этому символьные данные могут легко передаваться между системами без каких-либо ограничений. Единственный недостаток в том, что типы данных Unicode требуют для хранения каждого символа два байта и занимают в два раза больше места, чем их ANSI-аналоги.

Типы данных Unicode предваряются буквой *n*. Например, *nchar* в Unicode является аналогом типа данных *char* из кодировки ANSI. При определении символьного типа данных нужно указать максимальное число байтов, которое можно хранить в столбце. Например, тип *char(10)* служит для хранения не более 10 символов, поскольку для хранения каждого символа требуется один байт, а *nchar(10)* позволяет хранить не более пяти символов — ведь для хранения одного символа Unicode нужно два байта. В таблице 5 перечислены символьные типы данных, поддерживаемые в SQL Server.

Таблица 5 – Символьные типы данных

Тип данных	Занимаемая память	Количество символов	Назначение
<i>char(n)</i>	1-8000 байт	До 8000 символов	Тип данных ANSI фиксированного размера
<i>nchar(n)</i>	2-8000 байт	До 4000 символов	Тип данных Unicode фиксированного размера
<i>varchar(n)</i>	1-8000 байт	До 8000 символов	Тип данных ANSI переменного размера
<i>varchar(max)</i>	До 2 Гб	До 1073741824 символов	Тип данных ANSI переменного размера
<i>nvarchar(n)</i>	2-8000 байт	До 4000 символов	Тип данных Unicode переменного размера
<i>nvarchar(max)</i>	До 2 Гб	До 536 870 912 символов	Тип данных Unicode переменного размера
<i>text</i>	До 2 Гб	До 1073741824 символов	Тип данных ANSI переменного размера
<i>ntext</i>	До 2 Гб	До 536 870 912 символов	Тип данных Unicode переменного размера

Зачем же нужно так много символьных типов данных, на первый взгляд практически идентичных? Отличия между ними едва уловимы, но очень важны. Тип данных *char* как в ANSI, так и в Unicode имеет фиксированный размер. Поэтому ему всегда требуется один и тот же объем памяти

при любом количестве символов, хранящихся в столбце. Например, столбец с типом данных *char(30)* занимает 30 байт памяти независимо от того, хранится ли в нем один символ или тридцать. Неиспользуемое пространство дополняется пробелами до максимально возможного количества символов в столбце. А для хранения каждого символа в столбце с типом данных *varchar(30)* требуется всего один байт.

Типы данных *text* и *ntext* предназначены для хранения больших массивов символьных данных. Все же для столбцов с такими типами данных многие операции запрещены. Например, к ним нельзя применять оператор равенства или операцию объединения. Многие системные функции также не могут использовать эти типы данных.

Из-за этих ограничений в SQL Server появились типы данных *varchar(max)* и *nvarchar(max)*. Они объединяют возможности типов *text/ntext* и *varchar/nvarchar*, могут хранить до 2 Гб данных и не имеют ограничений по использованию с различными операциями и функциями.

Двоичные типы данных

Довольно часто в базах данных нужно хранить двоичные данные. Поэтому SQL Server поддерживает три типа данных, позволяющих хранить в таблице различные объемы двоичных данных. В таблице 6 перечислены двоичные типы данных, поддерживаемые в SQL Server.

Таблица 6 – Двоичные типы данных

Тип данных	Занимаемая память	Назначение
<i>binary(n)</i>	1-8 000 байт	Хранение двоичных данных фиксированного размера
<i>varbinary(n)</i>	1-8 000 байт	Хранение двоичных данных переменного размера
<i>varbinary(max)</i>	До 2 Гб	Хранение двоичных данных переменного размера
<i>image</i>	До 2 Гб	Хранение двоичных данных переменного размера

Типы данных *binary* служат в основном для хранения файлов в SQL Server. Для хранения небольших файлов, например группы файлов размером 4 или 6 Кб содержащих различные данные в двоичном формате, используют типы данных *binary/varbinary*.

Наиболее популярный тип данных в этой категории *image*. У него не совсем удачное имя, ведь он предназначен для хранения не только изображений, например фотографий из отпуска. Помимо фотографий с помощью *image* можно также хранить документы Word, Excel, PDF и Visio. Тип данных *image* подходит для хранения любых файлов, размер которых не превышает 2 Гб. Тип данных *varbinary(max)* позволяет хранить такой же объем данных, как и *image*, и может применяться во всех операциях и функциях, где допустимы типы данных *binary/varbinary*.

Специальные типы данных

Помимо уже перечисленных стандартных типов данных в SQL Server есть семь дополнительных типов данных узкого назначения. Их описание приведено в таблице 7.

Таблица 7 – Специальные типы данных

Тип данных	Назначение
<i>Bit</i>	Хранение 0, 1 или <i>null</i> . Применяется для основных значений флага. TRUE преобразуется в 1, а FALSE — в 0
<i>Timestamp</i>	Автоматически генерируемое значение. В каждой базе данных есть внутренний счетчик относительного времени, не связанного с реальным. В таблице может быть только один столбец с типом данных <i>timestamp</i> , которому присваивается значение временной метки базы данных при каждом добавлении или изменении строки
<i>uniqueidentifier</i>	16-битный GUID, служащий для глобальной идентификации строки в базах данных, экземплярах и серверах
<i>sql_variant</i>	Может изменять тип данных в зависимости от хранящихся данных. Максимальный объем хранимых данных — 8 000 байт
<i>cursor</i>	Встречается в приложениях, работающих с курсорами, и содержит ссылку на курсор, которая применяется для выполнения операций. Этот тип данных нельзя использовать в таблицах
<i>table</i>	Предназначен для хранения набора результатов с целью последующей обработки. Этот тип данных нельзя использовать в столбцах. Он применяется только при объявлении табличных переменных в триггерах, хранимых процедурах и функциях
<i>Xml</i>	Хранение XML-документов размером до 2 Гб. Задаваемые параметры позволяют хранить в столбцах только согласованные документы

Возможность ввода пустых значений

Второй аспект определения атрибута — возможность хранения в нем пустых значений. Для указания отсутствия значения в базах данных служит специальная конструкция *null*, означающая «неизвестный» или «отсутствующий», *null* не является значением и не требует памяти для своего хранения. Понять назначение этой конструкции проще всего на примере.

Допустим, вы проектируете отношение для хранения адресов клиентов вашей компании. Вы выделили три строки адреса для указания улицы и предусмотрели ввод значений страны, штата/провинции, города и почтового

индекса. Итак, вы создали отношение с семью атрибутами. Для указания улицы не всегда нужны все три строки, поэтому одна или две строки для таких адресов будут лишними. Некоторые клиенты проживают в странах, где нет штатов или провинций, поэтому этот атрибут также нужен не для всех клиентов. Кроме того, при вводе адресов пользователи могут и не знать почтовые индексы, и все же им нужно сохранить все известные данные. Возникает противоречие. Если значения в атрибуте нет или оно неизвестно на момент ввода данных, в нем можно указать фиктивное значение. Но это может усугубить ситуацию: ведь в отношение вводятся неверные данные, которыми могут воспользоваться служащие или клиенты. Обычно пользователи просто не заполняют атрибуты, данных для которых нет. Поскольку данные не были указаны явно, они либо неизвестны, либо отсутствуют. В базе данных для обозначения такого состояния неизвестности данных столбец определяется как *null*.

При определении атрибута можно указать возможность ввода пустых значений. Если их вводить нельзя, пользователь должен ввести данные.

Отсутствие чего-либо нельзя сравнить с отсутствием чего-то другого, точно так же одно пустое значение (*null*) несравнимо с другим. Поэтому учтите, что пустые значения нельзя использовать в сравнениях.

Вычисляемые столбцы

В SQL Server можно создавать специальные типы столбцов, которые называются *вычисляемыми* (*computed*) и содержат вычисления, использующие один или несколько других столбцов таблицы.

По умолчанию в вычисляемом столбце хранится определение вычислений, а не физические данные. При получении данных к ним применяется вычисление и возвращаются результаты.

С помощью ключевого слова *PERSISTED* можно задать физическое хранение данных в вычисляемом столбце. При наличии этого ключевого слова вычисление выполняется при добавлении или изменении строки, а его результат физически хранится в таблице.

2. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Создание таблиц. Все таблицы БД находятся в подпапке «Tables» папки «Students» в окне обозревателя объектов, представленного на рисунке 1.

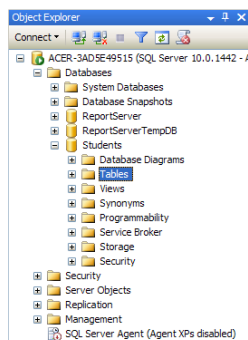


Рисунок 1

Для создания таблицы «Специальности» необходимо щёлкнуть ПКМ по папке «Tables» и в появившемся меню выбрать пункт «New Table». Появится окно создания новой таблицы, представленное на рисунке 2.

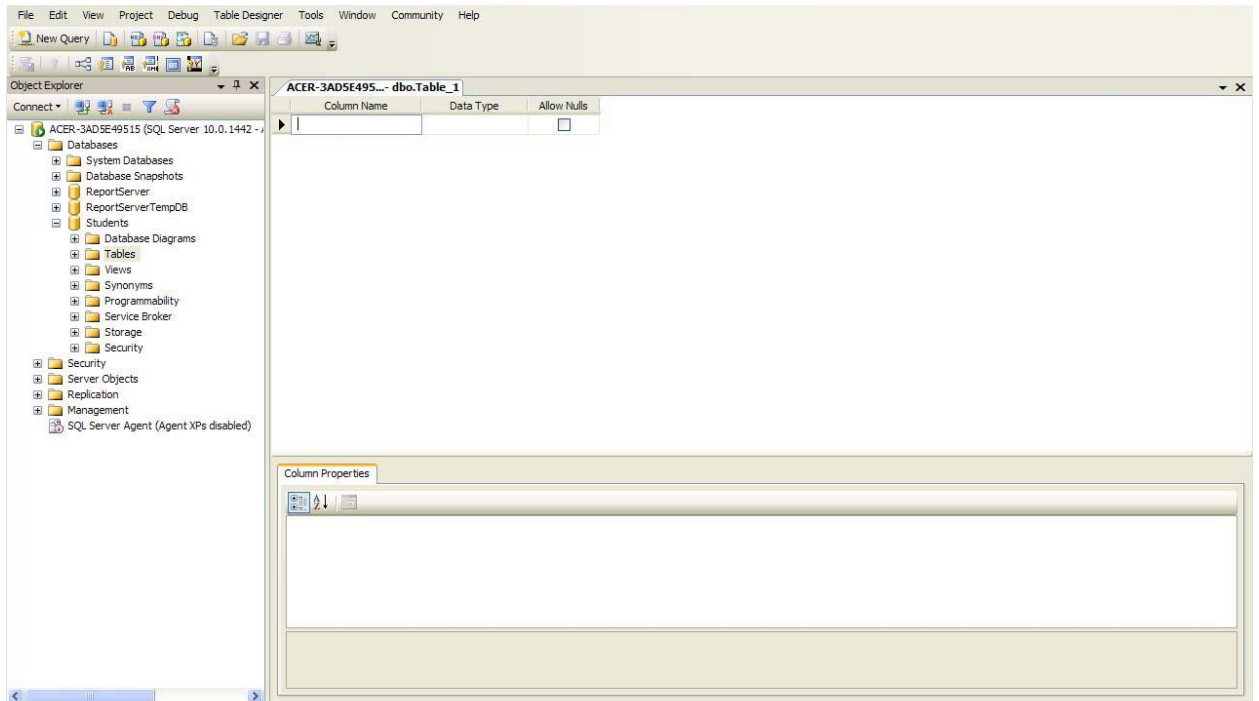


Рисунок 2

В правой части окна расположена таблица определения полей новой таблицы. Данная таблица имеет следующие столбцы:

- **Column Name** – имя поля. Имя поля должно всегда начинаться с буквы и не должно содержать различных специальных символов и знаков препинания. Если имя поля содержит пробелы, то оно автоматически заключается в квадратные скобки.
- **Data Type** – тип данных поля.
- **Allow Nulls** – допуск значения Null. Если эта опция поля включена, то в случае не заполнения поля в него будет автоматически подставлено значение Null. То есть, поле необязательно для заполнения.

Под таблицей определения полей располагается таблица свойств выделенного поля «Column Properties». В данной таблице настраиваются свойства выделенного поля. Некоторые из них будут рассмотрены ниже.

Для создания полей и настройки их свойств в таблице определения полей задайте значения столбцов «Column Name», «Data Type» и «Allow Nulls», как показано на рисунке 3.

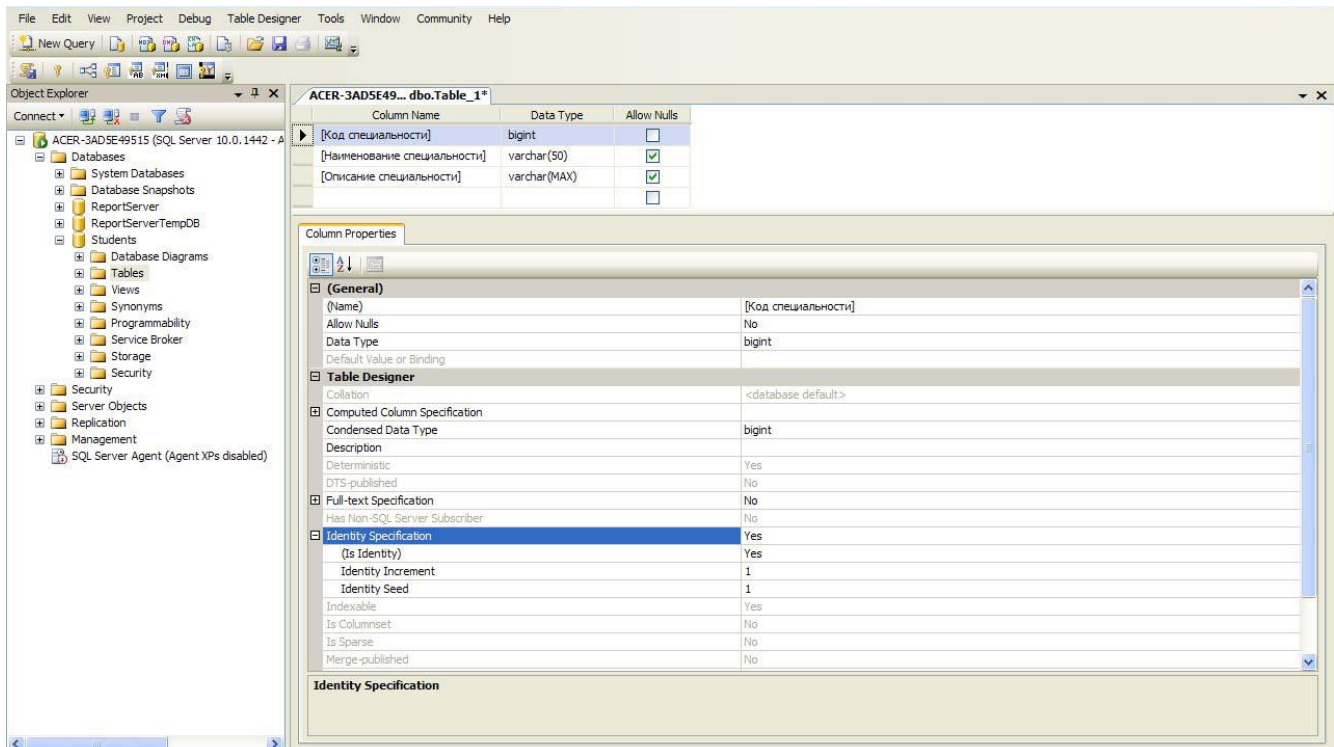



Рисунок 3

Из рисунка следует, что таблица «Специальности» имеет три поля:


- Код специальности – числовое поле для связи с таблицей студенты,
- Наименование специальности – текстовое поле, предназначенное для хранения строк, имеющих длину не более 50 символов.
- Описание специальности - текстовое поле, предназначенное для хранения строк, имеющих неограниченную длину.

Так как, поле «Код специальности» будет являться первичным полем связи в запросе, связывающем таблицы «Студенты» и «Специальности», то необходимо сделать его числовым счётчиком. То есть данное поле должно автоматически заполняться числовыми значениями. Более того, оно должно быть ключевым.

Чтобы сделать поле «Код специальности» счётчиком необходимо выделить поле, просто щёлкнув по нему мышкой в таблице определения полей. В таблице свойств поля отобразятся свойства поля «Код специальности». Необходимо развернуть группу свойств «Identity Specification» (Настройка особенности). Свойство «(Is Identity)» (Особенное) необходимо установить в значение «Yes» (Да). Необходимо задать свойства «Identity Increment» (Увеличение особенности, шаг счётчика) и «Identity Seed» (Начало особенности, начальное значение счётчика) равными 1. Эти настройки показывают, что значение поля «Код специальности» у первой записи в таблице будет равным 1, у второй – 2, у третьей 3 и т.д.

Чтобы сделать поле «Код специальности» ключевым полем выделите поле, а затем на панели инструментов нажмите кнопку с изображением ключа . В таблице определения полей, рядом с полем «Код специальности»

появится изображение ключа, говорящее о том, что поле ключевое. На этом настройку таблицы «Специальности» можно считать завершённой.

Чтобы закрыть окно создания новой таблицы, нажмите кнопку закрытия  в верхнем правом углу окна, над таблицей определения полей. Появится окно с запросом о сохранении таблицы, представленное на рисунке 4.

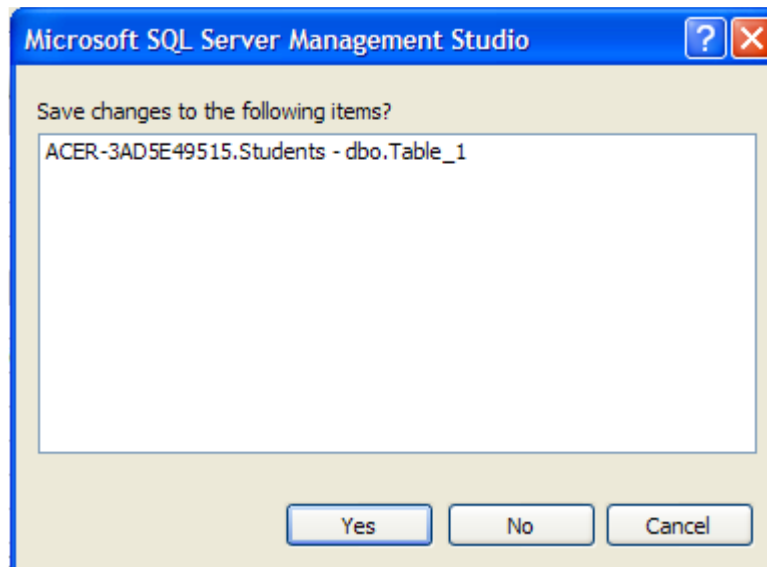


Рисунок 4

В этом окне необходимо нажать «Yes» (Да). Появится окно «Choose Name» (Задайте имя), предназначенное для определения имени новой таблицы, представленное на рисунке 5.

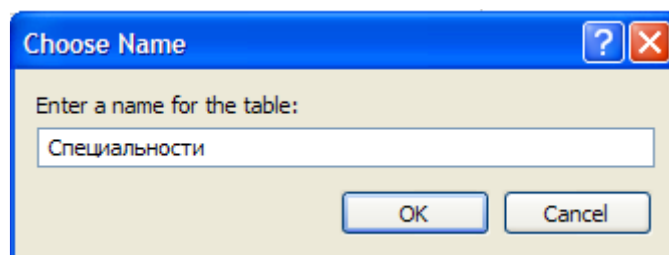


Рисунок 5

В этом окне задайте имя новой таблицы как «Специальности» и нажмите кнопку «Ok». Таблица «Специальности» отобразится в обозревателе объектов в папке «Tables» БД «Students».

В обозревателе объектов таблица «Специальности» отображается как «dbo.Специальности». Префикс «dbo» обозначает, что таблица является объектом БД (Data Base Object). В дальнейшем при работе с объектами БД префикс «dbo» можно опускать.

Далее необходимо создать таблицу «Предметы». Как и в случае с таблицей «Специальности» щёлкните ПКМ по папке «Tables» и в появившемся

меню выберите пункт «New Table». Создайте поля, представленные на рисунке 6.

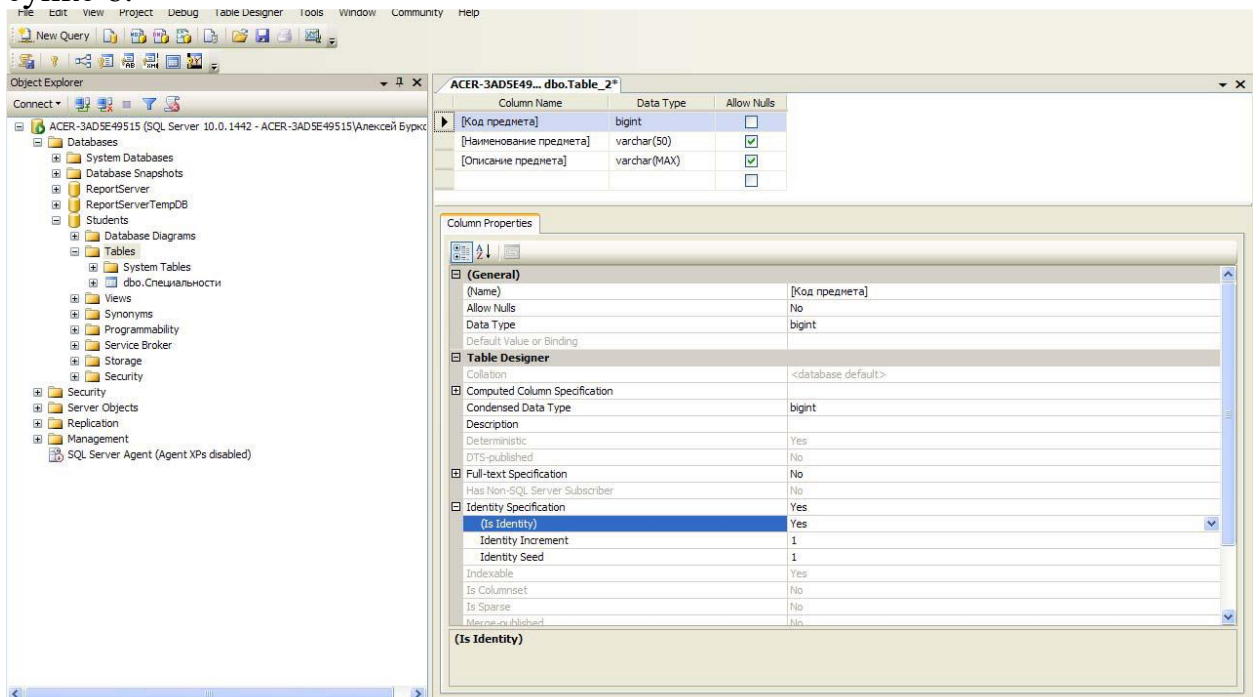


Рисунок 6

Сделайте поле «Код предмета» числовым счётчиком и ключевым полем, как это было сделано в таблице «Специальности». Закройте окно создания новой таблицы. В появившемся окне «Chose Name» задайте имя «Предметы», как показано на рисунке 7.

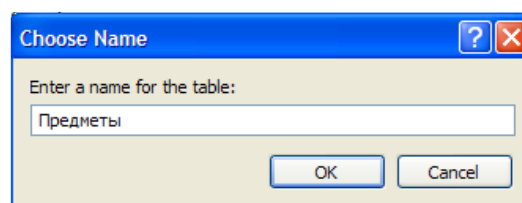


Рисунок 7

Таблица «Предметы» появится в папке «Tables» в обозревателе объектов.

После создания таблицы «Предметы» создайте таблицу «Студенты». Создайте новую таблицу аналогичную таблице, представленной на рисунке 8.

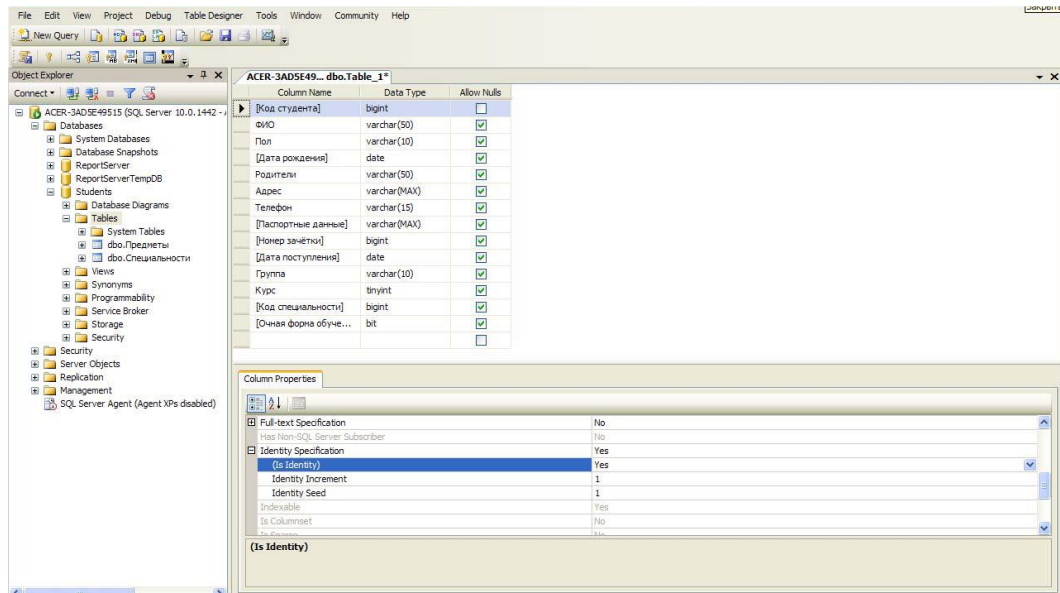


Рисунок 8

Рассматривая поля новой таблицы можно придти к следующим выводам:

- Поле «Код студента» - это первичное поле для связи с таблицей оценки. Следовательно, данное поле необходимо сделать числовым счётчиком и ключевым (см. создание таблицы «Специальности» выше);
- Поля «ФИО», «Пол», «Родители», «Адрес», «Телефон», «Паспортные данные» и «Группа» являются текстовыми полями различной длины (для задания длины выделенного текстового поля необходимо в таблице свойств выделенного поля установить свойство Length равное максимальному количеству знаков текста, вводимого в поле);
- Поля «Дата рождения» и «Дата поступления» предназначены для хранения дат, поэтому они имеют тип данных «date»;
- Поле «Очная форма обучения» является логическим полем. В «Microsoft SQL Server» такие поля должны иметь тип данных «bit»;
- Поля «Номер зачётки» и «Курс» являются целочисленными. Единственным отличием является размер полей. Поле «Номер зачётки» предназначено для хранения целых чисел в диапазоне -263...+263 (тип данных «bigint»). Поле «Курс» предназначено для хранения целых чисел в диапазоне 0...255 (тип данных «tinyint»);
- Поле «Код специальности» - это поле связи с таблицей «Специальности». Однако, данное поле связи является вторичным, поэтому его можно сделать просто целочисленным, то есть, «bigint».

После определения полей таблицы «Студенты», закройте окно создания новой таблицы. В появившемся окне «Chose Name» задайте имя новой таблицы как «Студенты», представленное на рисунке 9.

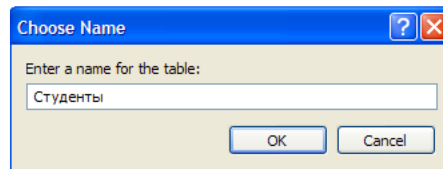


Рисунок 9

Таблица «Студенты» появится в папке «Tables» в обозревателе объектов.

Наконец, создайте таблицу «Оценки». Создайте поля, представленные на рисунке 10.

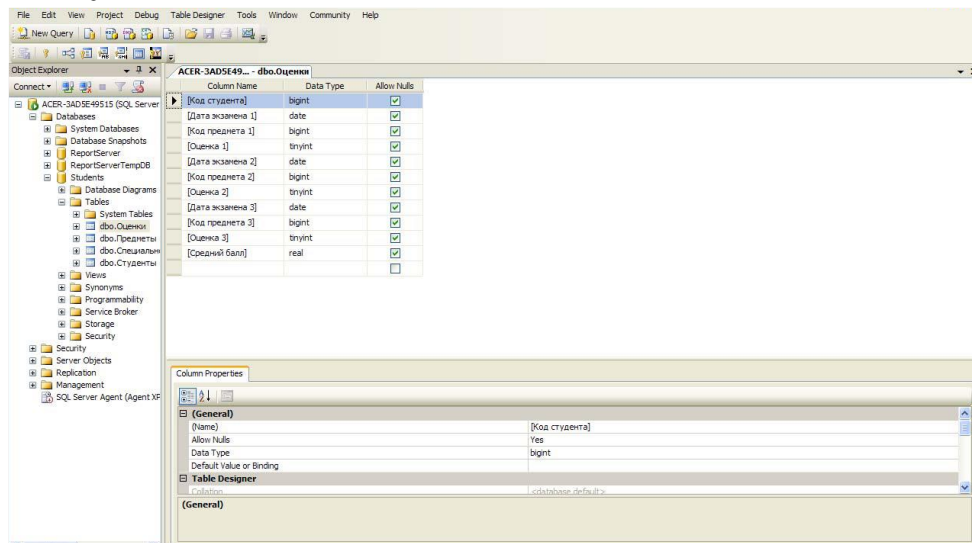


Рисунок 10

Таблица «Оценки» не имеет первичных полей связи. Следовательно, эта таблица не имеет ключевых полей. Поля «Код предмета 1», «Код предмета 2» и «Код предмета 3» являются вторичными полями связи, предназначенными для связи с таблицей «Предметы», поэтому они являются целочисленными (тип данных «bigint»). Поля «Дата экзамена 1», «Дата экзамена 2» и «Дата экзамена 3» предназначены для хранения дат (тип данных «date»). Поля «Оценка 1», «Оценка 2», и «Оценка 3» предназначены для хранения оценок. Задайте тип данных для этого поля «tinyint». Наконец, поле «Средний балл» хранит дробные числа и имеет тип «real». Закройте окно создания новой таблицы, задав имя таблицы как «Оценки», как показано на рисунке 11.

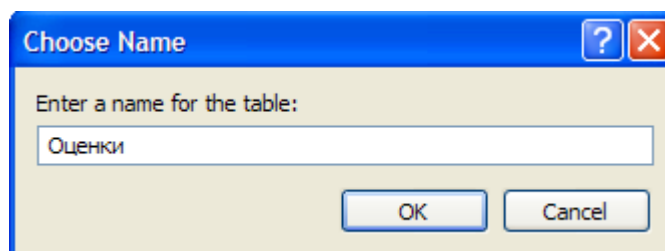


Рисунок 11

На этом мы заканчиваем создание таблиц БД «Students». После создания всех таблиц окно обозревателя объектов будет выглядеть, как показано на рисунке 12.

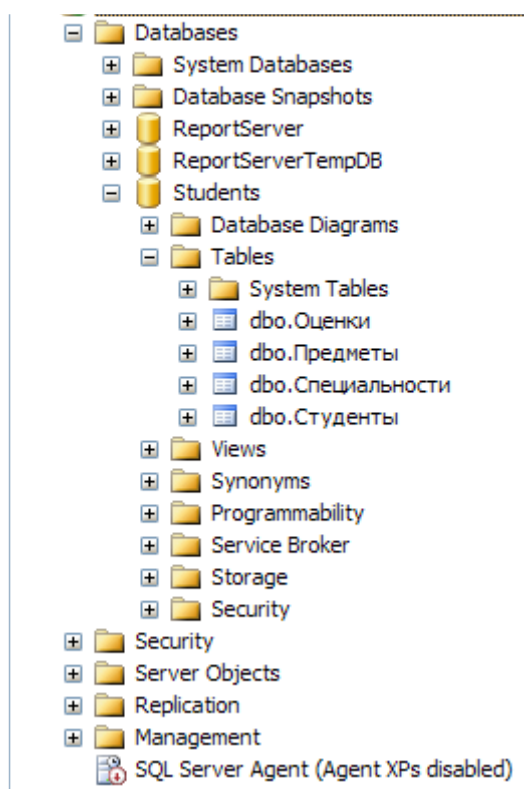


Рисунок 12


2. Операция заполнения таблиц начальными данными. Для начала заполните таблицу «Специальности». Для заполнения этой таблицы в обозревателе объектов щёлкните правой кнопкой мыши по таблице «Специальности» и в появившемся меню выберите пункт «Edit Top 200 Rows» (Редактировать первые 200 записей.). В рабочей области «Microsoft SQL Server Management Studio» проявится окно заполнения таблиц. Заполните таблицу «Специальности», как показано на рисунке 13.

	Код специальности	Наименование специальности	Описание специальности
	1	ММ	Математические методы
	2	ПИ	Прикладная информатика
	3	СТ	Статистика
	4	МО	Менеджмент организаций
►	5	БУ	Бухгалтерский учёт
*	NULL	NULL	NULL

Рисунок 13

Заполнение таблиц происходит полностью аналогично табличному процессору «Microsoft Excel 2000».

Так как поле «Код специальности» является первичным полем связи и ключевым числовым счётчиком, то оно заполняется автоматически (заполнять его не нужно).

Закройте окно заполнения таблицы «Специальность» щелкнув по кнопке закрытия  окна в верхнем правом углу, над таблицей.

После заполнения таблицы «Специальности» заполните таблицу «Предметы».

Откройте её для заполнения как описано выше, и заполните, как показано на рисунке 14.

	Код предмета	Наименование предмета	Описание предмета
	1	Операционные системы	Microsoft Windows Vista
	2	Офисные пакеты	Microsoft Office 2007
	3	Базы данных	Microsoft Access 2007
	4	Языки программирования	Microsoft Visual Studio 2008
►	5	Проектирование информационных систем	Microsoft SQL server 2008
*	NULL	NULL	NULL

Рисунок 14

Закройте окно заполнения таблицы «Предметы» и перейдите к заполнению таблицы «Студенты». Откройте таблицу «Студенты» для заполнения и заполните её как показано на рисунке 15.

ACER-3AD5E49... dbo.Студенты														▼ X
	Код сту...	ФИО	Пол	Дата рож...	Родители	Адрес	Телефон	Паспортные д...	Номер за...	Дата пост...	Группа	Курс	Код спец...	Очная форма...
	1	Иванов А.И.	Мужской	1983-12-12	Отец, Мать	Москва	+74957895674	8567-567543	13245	2007-09-01	ММ11	1	1	True
	2	Петрова И.И.	Женский	1982-11-01	Мать	Москва	+74957889876	4567-765432	34563	2006-08-01	ПИ21	2	2	False
	3	Мухин М.А.	Мужской	1982-05-14	Отец	Самара	+78462875690	5438-098787	56732	2006-07-05	СТ22	2	3	False
	4	Сидорова В.К.	Женский	1981-09-27	Нет	Саратов	+79027868909	1287-987509	27543	2005-06-23	МО31	3	4	True
	5	Кожевников А.А.	Мужской	1981-04-12	Мать	Казань	+79168563476	2312-675468	34217	2005-07-21	БУ33	3	5	True
	6	Пальчикова Н.Е.	Женский	1983-09-02	Отец, Мать	Челябинск	+74569098723	8743-856780	43278	2007-08-01	ММ12	1	1	False
	7	Царегородцев Е.В.	Мужской	1980-02-17	Отец	Самара	+78462234769	6543-834521	43765	2004-07-04	ПИ41	4	2	True
	8	Баранова Г.В.	Женский	1980-07-09	Отец, Мать	Чебоксары	+79027874638	2133-896567	10387	2004-08-09	СТ42	4	3	False
	9	Леухин П.Г.	Мужской	1979-02-26	Нет	Казань	+79067453678	2769-634904	67348	2003-07-23	МО51	5	4	True
►	10	Николаева А.П.	Женский	1979-03-17	Мать	Саратов	+78546456432	3256-490932	45287	2003-06-21	БУ53	5	5	False
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 15

Для заполнения дат в качестве разделителя можно использовать знак «.». Даты можно заполнять в формате «день.месяц.год».

Поле «Код специальности» является вторичным полем связи (для связи с таблицей «Специальности»). Следовательно, значения этого поля необходимо заполнять значениями поля «Код специальности» таблицы «Специаль-

ности». В нашем случае это значения от 1 до 5. Если у Вас коды специальностей в таблице «Специальности» имеют другие значения, то внесите их в таблицу «Студенты».

По окончании заполнения, закройте окно заполнения таблицы «Студенты».

Наконец заполните таблицу «Оценки», как это показано на рисунке 16.

ACER-3AD5E49... - dbo.Оценки											
	Код студента	Дата экзамена 1	Код предмета 1	Оценка 1	Дата экзамена 2	Код предмета 2	Оценка 2	Дата экзамена 3	Код предмета 3	Оценка 3	Средний балл
	1	2008-02-01	1	5	2008-02-09	4	3	2008-02-14	2	4	0
	2	2008-01-30	5	4	2008-02-23	3	5	2008-02-27	1	5	0
	3	2008-01-26	3	5	2008-02-05	1	3	2008-02-15	5	3	0
	4	2007-12-26	2	3	2008-01-05	4	4	2008-01-21	3	4	0
	5	2008-01-12	4	4	2008-01-18	5	4	2008-01-25	1	4	0
	6	2007-12-17	2	4	2007-12-26	4	5	2008-01-05	1	3	0
	7	2008-02-21	5	2	2008-02-25	1	2	2008-02-27	2	4	0
	8	2008-02-03	3	3	2008-02-12	5	3	2008-02-20	4	5	0
	9	2008-01-25	1	5	2008-02-02	3	5	2008-02-14	5	5	0
▶	10	2007-12-28	4	4	2008-01-11	1	4	2008-01-23	2	3	0
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 16

Поля с датами заполняются, как и в таблице «Студенты» (см. выше).

Поля «Код предмета 1», «Код предмета 2» и «Код предмета 3» являются вторичными полями связи с таблицей «Предметы», Поэтому они должны быть заполнены значениями поля «Код предмета из этой таблицы», то есть значениями от 1 до 5.

Закройте окно заполнения таблицы «Оценки». На этом процесс создания и заполнения таблиц БД «Students» заканчивается.

Перенос файла базы данных Microsoft SQL на другой компьютер

В большинстве случаев необходимо разрабатывать приложения, использующие в качестве базы данных Microsoft SQL Server. Наиболее рациональным решением является разработка базы данных в формате Microsoft SQL на рабочем компьютере с установленной локальной версией Microsoft SQL Server. При сдаче проекта заказчику возникает необходимость переноса базы данных с локального компьютера. Для переноса на другой компьютер необходимо скопировать два файла: саму базу данных Students.mdf и файл отчетов о транзакциях Students.ldf. Однако непосредственное копирование данных файлов невозможно, так как данные файлы используются сервером баз данных. Для того чтобы сделать файлы доступными необходимо отсоединить его от сервера: щелкните ПМК в ветке «Базы данных» по названию базы данных Students и выберите Tasks->Detach.... как показано на рисунке 17.

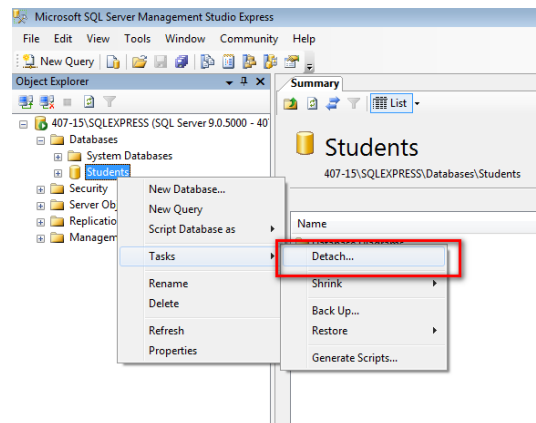


Рисунок 17

Появляется диалоговое окно «Отсоединение базы данных» («Detach»). Подтверждаем отсоединение, нажимая кнопку «ОК», – и база отсоединена. Теперь нужные файлы доступны для копирования.

Для присоединения базы данных на другом компьютере запускаем Microsoft SQL Server Management Studio, выделите ветку «Базы данных» и в контекстном меню выберите «Присоединить» («Attach...»), как показано на рисунке 18.

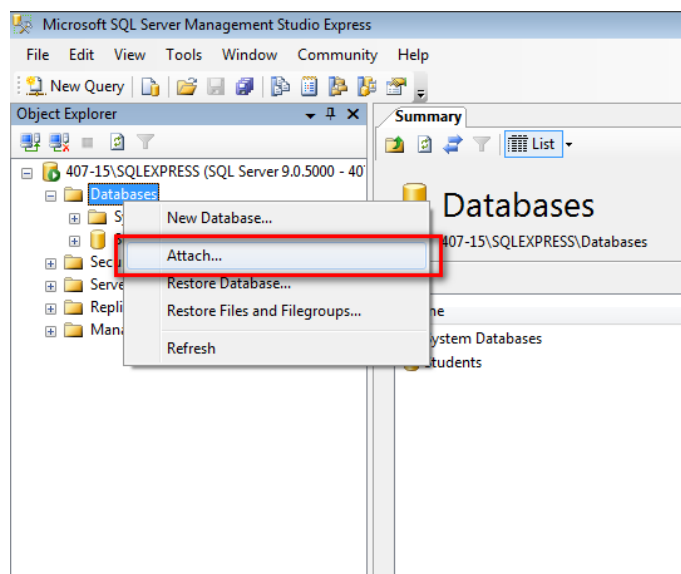
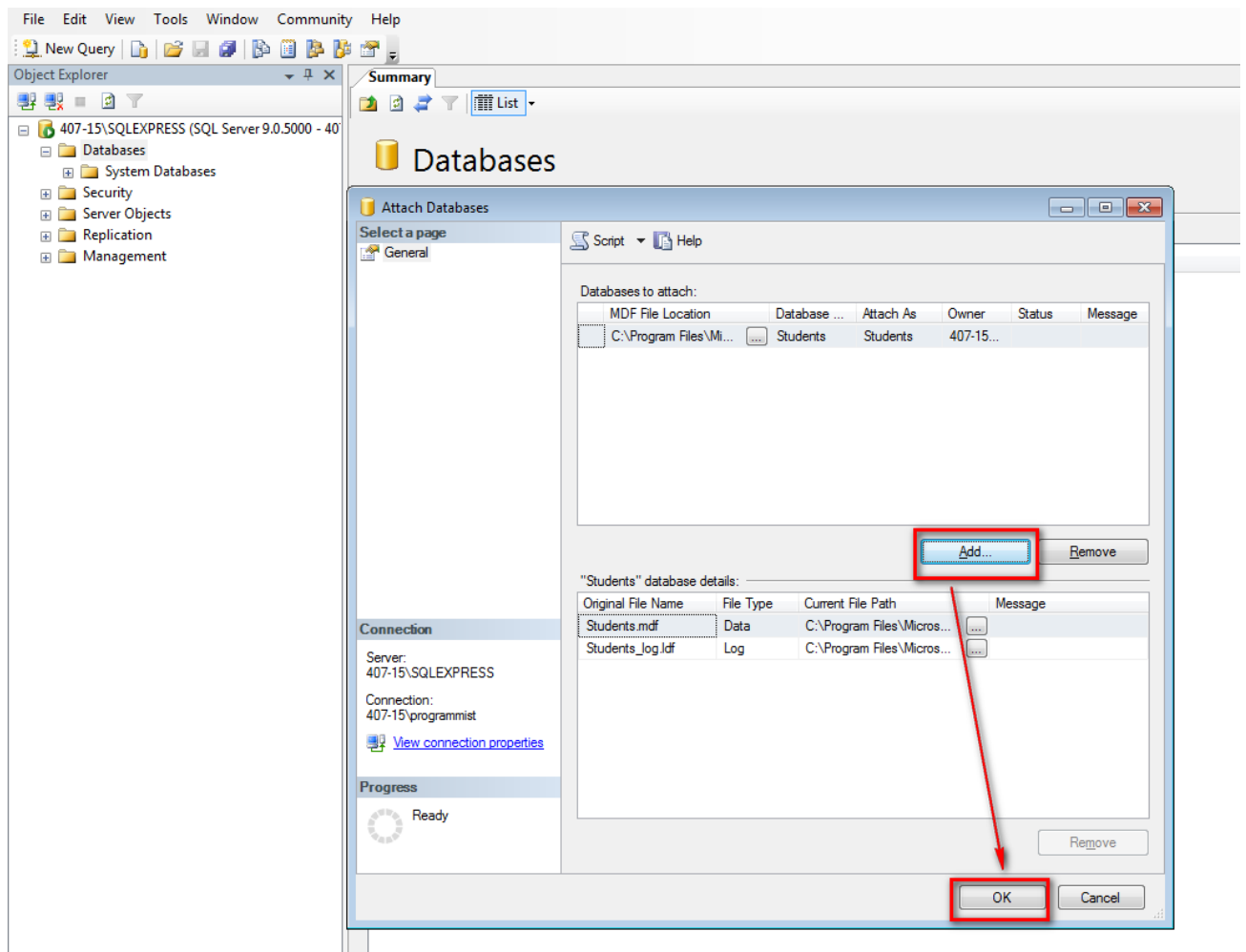


Рисунок 18

Появляется диалоговое окно присоединения базы данных.



В появившемся окне укажите расположение файла базы данных Students.mdf (файл отчетов присоединяется автоматически) и нажимаем кнопку «ОК». Присоединившаяся база данных немедленно отображается в папке «Базы данных».

Следует отметить, что после присоединения БД может потребоваться настройка пользователей БД и прав доступа.

3. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите и опишите основные категории типов данных.
2. Укажите, что задают параметры p и s типов данных $\text{decimal}(p,s)$ и $\text{numeric}(p,s)$.
3. Является ли конструкция `null` значением и требует ли памяти для своего хранения?
4. Опишите, как сделать поле счетчиком.
5. Опишите, как сделать поле ключевым.
6. Опишите процесс заполнения таблиц БД данными.
7. Опишите, как перенести файл базы данных Microsoft SQL на другой компьютер.
8. Перечислите и поясните основные характеристики полей.
9. Опишите, как создать таблицу в «SQL Server Management Studio».
10. Дайте определения понятиям «поле», «запись».

11. Сформулируйте простое правило, используя которое можно правильно подобрать тип данных для столбца.

12. Опишите, как настроить свойства полей таблицы.

4. ДОМАШНЕЕ ЗАДАНИЕ

[1], страницы 64-73

5. ЛИТЕРАТУРА

1. Петкович, Д. Microsoft SQL Server 2012. Руководство для начинающих: пер. с английского / Д. Петкович. – СПб.: БХВ-Петербург, 2013. – 816 с.: ил.

2. Сеть разработчиков Microsoft [Электронный ресурс]. – Режим доступа: <https://msdn.microsoft.com/ru-ru/library>

Преподаватель

С.В. Банцевич

Рассмотрено на заседании цикловой
комиссии программного обеспечения
информационных технологий №10
Протокол № от « » _____ 2017
Председатель ЦК _____ С.В. Банцевич