Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Владимирский государственный университет имени Александра
Григорьевича и Николая Григорьевича Столетовых»

А.О. Кучерик, А.Ю. Лексин, Д.Н. Бухаров, В.В. Баринов

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

ПО ДИСЦИПЛИНЕ «ТЕОРИЯ РАЗРАБОТКИ И ЗАЩИТЫ БАЗ ДАННЫХ»

Владимир 2017

УДК 004.65
ББК 75.0

Составители: А.О. Кучерик, А.Ю. Лексин, Д.Н. Бухаров, В.В. Баринов.

Рецензент доктор физико-математических наук, профессор Владимирского государственного университета имени Александра Григорьевича и Николая Григорьевича Столетовых – Л.В. Фуров.

Печатается по решению редакционного совета ВлГУ

Методические указания по выполнению лабораторных работ по дисциплине «ТЕОРИЯ РАЗРАБОТКИ И ЗАЩИТЫ БАЗ ДАННЫХ» / Владим. гос. уни-т имени Александра Григорьевича и Николая Григорьевича Столетовых; А.О. Кучерик, А.Ю. Лексин, Д.Н. Бухаров, В.В. Баринов. – Владимир: Изд-во ВлГУ, 2017. – 104 с.

Рассмотрены основы разделов дисциплины «ТЕОРИЯ РАЗРАБОТКИ И ЗАЩИТЫ БАЗ ДАННЫХ». К лабораторным работам приведена краткая теория, что упрощает их выполнение. Предназначены для проведения практических и лабораторных занятий для специальности 09.02.03 «Программирование в компьютерных системах». Рекомендовано для формирования профессиональных компетенций в соответствии с ФГОС 3-го поколения.

Ил. 52. Табл. 4. Библиогр.: 13 назв.

УДК 004.65
ББК 75.0

Оглавление

Введение

В методических указаниях приведены лабораторные работы по курсу «ТЕОРИЯ РАЗРАБОТКИ И ЗАЩИТЫ БАЗ ДАННЫХ». Лабораторные работы сопровождаются краткой теорией, что делает более удобным их выполнение. Успешное решение практических задач требует владения приемами программирования в среде Visual C#.

Цель выполнения лабораторных работ – изучение основ языка SQL и получение практических навыков работы с реляционными базами данных. Выполнение работ включает создание базы данных, заполнение его данными, формирование запросов на SQL и создание представлений, хранимых процедур, пользовательских функций.

1. Лабораторная работа №1. Проектирование базы данных

1.1. Цель работы

Приобретение навыков анализа предметной области и построения концептуальной модели.

1.2 Содержание работы

- Анализ текстового описания предметной области.
- Построение концептуальной модели.

1.3. Задания к работе

В соответствии с вариантом (см. Приложение):

1.Выделить основные абстракции (сущность, атрибут, связь) в предметной области и определить их параметры.

2.Сформировать максимально полный перечень возможных запросов к базе данных на основе анализа предметной области.

3.Построить концептуальную модель в виде ER-диаграммы.

4.Представить концептуальную модель в терминах реляционной модели.

5.Описать домены (допустимые множества значений, которые могут принимать атрибуты), указывая типы соответствующих данных и их характеристики.

6.Определить ключи и внешние ключи (если они есть).

7.Выписать функциональные зависимости (рассматривая возможные значения полей таблицы)

1.4. Краткая теория

Модель данных в общем понимании является представлением «реального мира» (т.е. реальных объектов и событий, а также семантических (смысловых) связей между ними), однако это некоторая абстракция, в которой остаются только те части реального мира, которые важны для разработчиков конкретной БД, а все второстепенные (малозначимые) детали – игнорируются.

Цель построения модели данных – представление данных пользователя в понятном виде, который можно легко применить при проектировании БД. Модель данных должна точно и недвусмысленно описывать части реального мира в таком виде, который позволяет разработчикам и пользователям (заказчикам) БД обмениваться мнениями при разработке и поддержке БД.

Цель этапа *концептуального проектирования* БД – адекватное отображение предметной области и информационных потребностей пользователей в концептуальной модели данных.

Модель «сущность-связь» (*ER-модель*) представляет собой высокоуровневую концептуальную модель данных с возможностью графического представления информации в виде *ER-диаграмм.*

КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ ДАННЫХ

Можно выделить три основные семантические концепции в ER-модели:

Объекты (типы сущностей).

Типы объектов (типы сущностей) – множество объектов реального мира с одинаковыми свойствами, которые характеризуются независимым существованием и могут быть объектом как с реальным (физическим) существованием (например «работник», «деталь», «поставщик»), так и объектом с абстрактным (концептуальным) существованием (например «рабочий стаж», «осмотр объекта»). Каждый тип объекта идентифицируется уникальным именем и обязательным списком свойств.

Объект (экземпляр типа объекта или сущность) – экземпляр типа сущности, предмет, который может быть четко идентифицирован на основе свойств (так как обладает уникальным набором свойств среди всех объектов одного типа).

Представление объектов на диаграмме: в виде прямоугольника с именем внутри него; сильный тип объекта – прямоугольник с одинарным контуром; слабый тип объекта – прямоугольник с двойным контуром.

Свойства (атрибуты).

Свойства (атрибуты) служат для описания типов объектов или отношений. Значения свойств каждого типа извлекаются из

соответствующего множества значений (в этом множестве определяются все потенциальные значения свойства, различные свойства могут использовать одно множество значений).

Представление свойств на диаграмме: в виде эллипса с уникальным именем (уникальность среди множества атрибутов) внутри него, присоединенных линией к типу объекта; для производных свойств – эллипс окружен пунктирным контуром, для многозначных – двойным; имя свойства, которое является первичным ключом, – подчеркивается.

Отношения (типы связей).

Типы отношений (типы связи) – осмысленная ассоциация (связь) между типами объектов.

Экземпляр отношения (отношение) – ассоциация (связь) между экземплярами объектов, включающая по одному экземпляру объекта с каждой стороны связи.

СТРУКТУРНЫЕ ОГРАНИЧЕНИЯ ER-МОДЕЛИ

Структурные ограничения, накладываемые на участников отношения, являются отражением требований реального мира. Можно выделить такие общие ограничения, как мощность отношения и степень участия объектов в отношении.

Мощность отношения – максимальное количество элементов одного типа объекта, связанных с одним элементом другого типа объекта. Обычно рассматриваются следующие виды связей:

- «*один-к-одному*» – максимальная мощность отношения в обоих направлениях равна одному (обозначается «1»);
- «*один-ко-многим*» – максимальная мощность отношения в одном направлении равна одному, а в другом – многим (обозначается «*»);
- «*многие-ко-многим*» – максимальная мощность отношения в обоих направлениях равна многим. [6]

РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ.

Реляционная модель данных – формальная теория данных, основанная на некоторых положениях математики (теории множеств и предикативной логике).

РЕЛЯЦИОННЫЕ ОБЪЕКТЫ (СТРУКТУРА МОДЕЛИ)

Реляционная модель данных основана на математическом понятии *отношения* (relation), физическим представлением которого является *таблица*. Все данные (описания объектов) в реляционной БД пользователь воспринимает как набор *таблиц* (множество отношений).

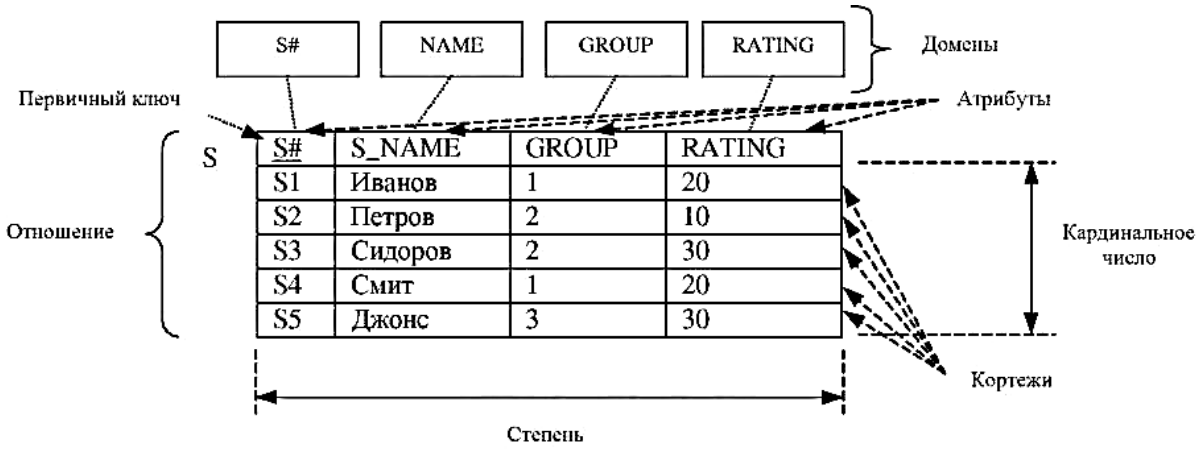


Рис. 1. Отношение в реляционной модели

Краткое описание терминов реляционной модели (рис. 1):

- отношение – плоская таблица;
- кортеж – строка таблицы (не включая заголовок);
- кардинальное число – количество строк таблицы (без заголовка);
- атрибут – столбец таблицы (или поле строки);
- степень – количество столбцов таблицы;
- первичный ключ – уникальный идентификатор для таблицы;
- домен – общая совокупность допустимых значений.

Целостность данных предназначена для сохранения в БД «отражения действительности реального мира», т.е. устранения недопустимых конфигураций (состояний) значений и связей, которые не имеют смысла в реальном мире. Правила целостности данных можно разделить на:

- специфические или корпоративные ограничения целостности - дополнительные ограничения, специфические для конкретных БД;
- общие правила целостности – правила, которые применимы к любой реляционной БД (относятся к потенциальным (первичным) и к внешним ключам).

Атрибуты или множество атрибутов значения, которых уникальным образом идентифицируют экземпляр объекта, называются *первичным ключом***,** т.к. все экземпляры объекта должны быть различны, то каждый объект должен иметь ключ.

Потенциальный ключ – это обобщение понятия первичного ключа. Потенциальные ключи также как и первичный обладают свойством уникальной идентификации кортежа в отношении, но если первичный ключ в отношении должен быть выбран только один, то потенциальных ключей может быть несколько (первичный ключ выбирается из потенциальных).

Внешний ключ – это множество атрибутов объекта; каждому значению внешнего ключа соответствует значение потенциального ключа. Внешние ключи используются для связывания кортежей в реляционных базах данных. [4]

Ссылочная целостность

Правило ссылочной целостности – база данных не должна содержать несогласованных значений внешних ключей (здесь «несогласованное значение внешнего ключа» – это значение внешнего ключа, для которого не существует отвечающего ему значения соответствующего потенциального ключа в соответствующем целевом отношении), т.е. если В ссылается на А, тогда А должно существовать.

NULL-значения

NULL -значения введены для обозначения таких значений атрибутов, которые на настоящий момент неизвестны или неприемлемы для некоторого кортежа. Это не значение по умолчанию, а отсутствие какого-либо значения (например, отсутствующие (на текущий момент) данные об адресе нового студента).

Для каждого атрибута должно быть установлено, может ли он принимать *NULL*-значения или нет, т.к. это влияет на концепции потенциальных и внешних ключей реляционной модели данных:

Получение реляционной модели из er-диаграммы

Алгоритм преобразования ER-диаграммы в реляционную модель (схему) состоит из следующих шагов:

Шаг 1. Каждый объект на ER-диаграмме превращается в таблицу. Имя объекта становится именем таблицы.

Шаг 2. Каждый атрибут объекта становится возможным столбцом с тем же именем; при этом может выбираться более точный формат данных. Столбцы, соответствующие необязательным атрибутам, могут содержать неопределенные значения; столбцы, соответствующие обязательным атрибутам, – не могут.

Шаг 3. Уникальные (ключевые) атрибуты объекта превращаются в первичный ключ таблицы. Если имеется несколько возможных уникальных идентификаторов, то выбирается наиболее подходящий для использования.

Шаг 4. Связи «один-ко-многим» (в том числе и связи «один-к-одному») становятся внешними ключами. Внешний ключ добавляется в виде столбца (столбцов) в таблицу, соответствующую объекту со стороны «многие» связи. Необязательные связи соответствуют столбцам, допускающим неопределенные значения; обязательные связи – столбцам, не допускающим неопределенные значения.

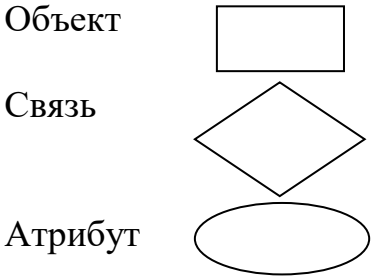
Шаг 5. Связи «многие-ко-многим» реализуются через промежуточную таблицу. Эта таблица будет содержать как минимум

столбцы внешних ключей на соответствующие объекты. Первичный ключ промежуточной таблицы должен включать в себя все внешние ключи на объекты, участвующие в связи.

Шаг 6. Если связь имеет дополнительные атрибуты, то, как и в случае атрибутов объектов, они становятся возможным столбцом таблицы:

- для связей «один-ко-многим» – в таблице со стороны «многие» (вместе с внешним ключом);
- для связей «многие-ко-многим» – в промежуточной таблице (при этом атрибуты, расширяющие комбинацию в связи (например «дата»), также должны войти в состав первичного ключа промежуточной таблицы). [4]

При построении моделей используются следующие геометрические фигуры:



Пример

1. Выделить основные абстракции (сущность, атрибут, связь) в предметной области и определить их параметры.

Определим следующие сущности: **СТУДЕНТ, ЭКЗАМЕН, ОЦЕНКА.** Определим атрибуты сущностей. Пусть для упрощения сущность **СТУДЕНТ** характеризуется только **фамилией**. Фамилию мы и возьмем в качестве атрибута. Так как фамилия может неоднозначно идентифицировать объект, введем дополнительный атрибут **Код студента**, уникальный для каждого студента. Таким образом, сущность СТУДЕНТ характеризуется двумя атрибутами код студента, фамилия. Аналогично определим сущность ЭКЗАМЕН с атрибутами код экзамена, предмет, дата экзамена и сущность ОЦЕНКА с атрибутом значение оценки (оценка). Между этими сущностями существуют следующие связи: студент сдавал экзамен, студент получил оценку, по экзамену получены следующие оценки.

2.Сформировать максимально полный перечень возможных запросов к базе данных на основе анализа предметной области.

По смыслу задачи к базе данных возможны следующие запросы:

Какие оценки получил студент с заданной фамилией (кодом);

Какие студенты получили заданное значение оценки;

Какие экзамены сдал студент с заданной фамилией (кодом);

Какую оценку по конкретному предмету получил студент с заданной фамилией (кодом).

3.Построить концептуальную модель в виде ER-диаграммы.

Нарисуем возможный вариант ER-диаграмм

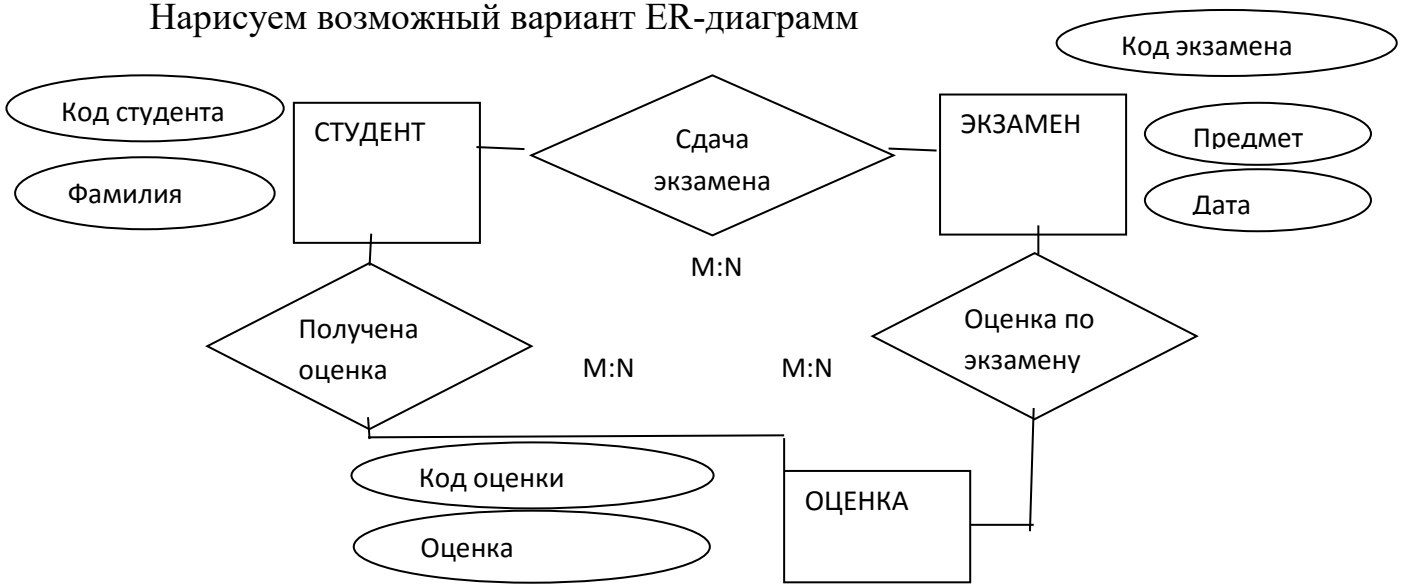


Рис. 2. ER диаграмма

По этой диаграмме можно ответить на все вопросы, кроме последнего. В этом случае предлагается ввести новую агрегированную сущность. Определим эту сущность как ЭКЗАМЕНАЦИОННАЯ ВЕДОМОСТЬ с атрибутами код студента, фамилия, код экзамена, предмет, дата экзамена, оценка. Нарисуем второй вариант ER-диаграммы.

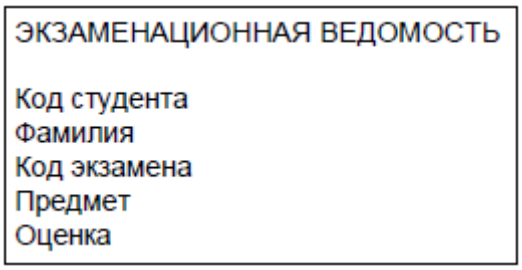


Рис. 3. ER-диаграмма 2

По этой диаграмме можно ответить на все вопросы, но здесь очевидно дублирование информации и возможны аномалии удаления, добавления.

4. Представить концептуальную модель в терминах реляционной модели.

В терминах концептуальной модели эта модель представляется следующей таблицей 1.

Таблица 1

Реляционная модель

Код студента	Фамилия	Код экзамена	Предмет	Дата	Оценка

5. Описать домены (допустимые множества значений, которые могут принимать атрибуты), указывая типы соответствующих данных и их характеристики.

Код студента принимает значения из множества целых чисел, максимальная длина числа 4 знака.

Фамилия принимает символьное значение, максимальная длина 20 символов.

Код экзамена принимает значения из множества целых чисел, максимальная длина числа 4 знака.

Предмет принимает символьное значение, максимальная длина 20 символов.

Дата экзамена принимает значение дата в формате 00.00.00.

Оценка принимает целое значение от 2 до 5.

6.Определить ключи и внешние ключи (если они есть).

Ключом данного отношения является совокупность атрибутов код студента, код экзамена.

7. Выписать функциональные зависимости (рассматривая возможные значения полей таблицы).

Отношение *R*1 представляет объект СТУДЕНТ с атрибутами: код студента (первичный ключ), фамилия.

Отношение *R*3 представляет объект ЭКЗАМЕН с атрибутами: код экзамена (первичный ключ), предмет, дата.

Отношение *R*4 представляет объект ОЦЕНКА с атрибутами: код студента (внешний ключ), код экзамена (внешний ключ), оценка. Первичный ключ здесь составной: код студента, код экзамена.

Для наглядности представим полученную модель в виде ER-диаграммы (рис.4).

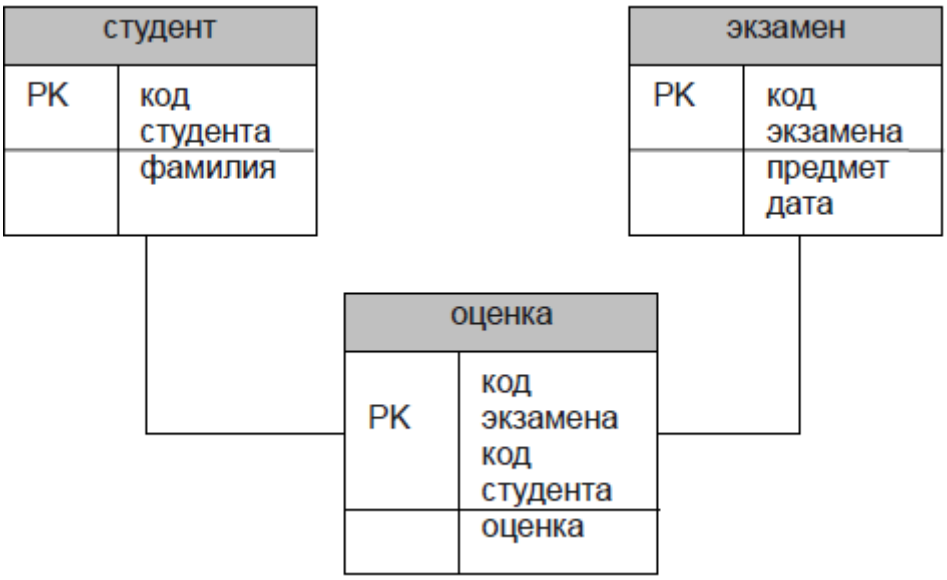


Рис.4. ER диаграмма базы данных

Эта диаграмма и является нужным вариантом. Таким образом, полученная реляционная модель включает три отношения.

ПРИЛОЖЕНИЕ. ПРИМЕРЫ ПРЕДМЕТНЫХ ОБЛАСТЕЙ

Вариант № 1. Страховая компания

Описание предметной области

Вы работаете в страховой компании. Вашей задачей является отслеживание ее финансовой деятельности.

Компания имеет различные филиалы по всей стране. Каждый филиал характеризуется названием, адресом и телефоном. Деятельность компании организована следующим образом: к вам обращаются различные лица с целью заключения договора о страховании. В зависимости от принимаемых на страхование объектов и страхуемых рисков договор заключается по определенному виду страхования (например, страхование автотранспорта от угона, страхование домашнего имущества, добровольное медицинское страхование). При заключении договора вы фиксируете дату заключения, страховую сумму, вид страхования, тарифную ставку и филиал, в котором заключался договор.

Возможный набор сущностей

Договоры (Номер договора, Дата заключения, Страховая сумма, Тарифная ставка, Код филиала, Код вида страхования).

Вид страхования (Код вида страхования, Наименование).

Филиал (Код филиала, Наименование филиала, Адрес, Телефон).

Вариант № 2. Гостиница

Описание предметной области

Вы работаете в гостинице. Вашей задачей является отслеживание финансовой стороны ее работы.

Ваша деятельность организована следующим образом: гостиница предоставляет номера клиентам на определенный срок. Каждый номер характеризуется вместимостью, комфортностью (люкс, полулюкс, обычный) и ценой. Вашими клиентами являются различные лица, о которых вы собираете определенную информацию (фамилия, имя, отчество и некоторый комментарий). Сдача номера клиенту производится при наличии свободных мест в номерах, подходящих клиенту по указанным выше параметрам. При поселении фиксируется дата поселения. При выезде из гостиницы для каждого места запоминается дата освобождения.

Возможный набор сущностей

Клиенты (Код клиента, Фамилия, Имя, Отчество, Паспортные данные, Комментарий).

Номера (Код номера, Номер, Количество человек, Комфортность, Цена).

Поселение (Код поселения, Код клиента, Код номера, Дата поселения, Дата освобождения, Примечание).

Вариант № 3. Ломбард

Описание предметной области

Вы работаете в ломбарде. Вашей задачей является отслеживание финансовой стороны его работы.

Деятельность компании организована следующим образом: к вам обращаются различные лица с целью получения денежных средств под залог определенных товаров. У каждого из приходящих к вам клиентов вы запрашиваете фамилию, имя, отчество и другие паспортные данные. После оценивания стоимости принесенного в качестве залога товара вы определяете сумму, которую готовы выдать на руки клиенту, а также свои комиссионные.

Кроме того, определяете срок возврата денег. Если клиент согласен, то ваши договоренности фиксируются в виде документа, деньги выдаются клиенту, а товар остается у вас. В случае если в указанный срок не происходит возврата денег, товар переходит в вашу собственность.

Возможный набор сущностей

Клиенты (Код клиента, Фамилия, Имя, Отчество, Номер паспорта, Серия паспорта, Дата выдачи паспорта).

Категории товаров (Код категории товаров, Название, Примечание).

Сдача в ломбард (Код, Код категории товаров, Код клиента, Описание товара, Дата сдачи, Дата возврата, Сумма, Комиссионные).

Вариант № 4. Реализация готовой продукции

Описание предметной области

Вы работаете в компании, занимающейся оптово-розничной продажей различных товаров. Вашей задачей является отслеживание финансовой стороны ее работы.

Деятельность компании организована следующим образом: компания торгует товарами из определенного спектра. Каждый из этих товаров характеризуется наименованием, оптовой ценой, розничной ценой и справочной информацией. В вашу компанию обращаются покупатели. Для каждого из них вы запоминаете в базе данных стандартные данные (наименование, адрес, телефон, контактное лицо) и составляете по каждой сделке документ, запоминая наряду с покупателем количество купленного им товара и дату покупки.

Возможный набор сущностей

Товары (Код товара, Наименование, Оптовая цена, Розничная цена, Описание).

Покупатели (Код покупателя, Телефон, Контактное лицо, Адрес).

Сделки (Код сделки, Дата сделки, Код товара, Количество, Код покупателя, Признак оптовой продажи).

Вариант № 5. Ведение заказов

Описание предметной области

Вы работаете в компании, занимающейся оптовой продажей различных товаров. Вашей задачей является отслеживание финансовой стороны ее работы.

Деятельность компании организована следующим образом: компания торгует товарами из определенного спектра. Каждый из этих товаров характеризуется ценой, справочной информацией и признаком наличия или отсутствия доставки. В вашу компанию обращаются заказчики. Для каждого из них вы запоминаете в базе данных стандартные данные (наименование, адрес, телефон, контактное лицо) и составляете по каждой сделке документ, запоминая наряду с заказчиком количество купленного им товара и дату покупки.

Возможный набор сущностей

Товары (Код товара, Цена, Доставка, Описание).

Заказчики (Код заказчика, Наименование, Адрес, Телефон, Контактное лицо).

Заказы (Код заказа, Код заказчика, Код товара, Количество, Дата).

Вариант № 6. Бюро по трудоустройству

Описание предметной области

Вы работаете в бюро по трудоустройству. Вашей задачей является отслеживание финансовой стороны работы компании.

Деятельность бюро организована следующим образом: бюро готово искать работников для различных работодателей и вакансии для ищущих

работу специалистов различного профиля. При обращении к вам клиента-работодателя его стандартные данные (название, вид деятельности, адрес, телефон) фиксируются в базе данных. При обращении к вам клиента-соискателя его стандартные данные (фамилия, имя, отчество, квалификация, профессия, иные данные) также фиксируются в базе данных. По каждому факту удовлетворения интересов обеих сторон составляется документ. В документе указываются соискатель, работодатель, должность и комиссионные (доход бюро).

Возможный набор сущностей

Работодатели (Код работодателя, Название, Вид деятельности, Адрес, Телефон).

Соискатели (Код соискателя, Фамилия, Имя, Отчество, Квалификация, Вид деятельности, Иные данные, Предполагаемый размер заработной платы).

Сделки (Код соискателя, Код работодателя, Должность, Комиссионные).

Вариант № 7. Нотариальная контора

Описание предметной области

Вы работаете в нотариальной конторе. Вашей задачей является отслеживание финансовой стороны работы компании.

Деятельность нотариальной конторы организована следующим образом: фирма готова предоставить клиенту определенный комплекс услуг. Для наведения порядка вы формализовали эти услуги, составив их список с описанием каждой услуги. При обращении к вам клиента его стандартные данные (название, вид деятельности, адрес, телефон) фиксируются в базе данных. По каждому факту оказания услуги клиенту составляется документ. В документе указываются услуга, сумма сделки, комиссионные (доход конторы), описание сделки.

Возможный набор сущностей

Клиенты (Код клиента, Название, Вид деятельности, Адрес, Телефон).

Сделки (Код сделки, Код клиента, Код услуги, Сумма, Комиссионные, Описание).

Услуги (Код услуги, Название, Описание).

Вариант № 8. Фирма по продаже запчастей

Описание предметной области

Вы работаете в фирме, занимающейся продажей запасных частей для автомобилей. Вашей задачей является отслеживание финансовой стороны работы компании.

Основная часть деятельности, находящейся в вашем ведении, связана с работой с поставщиками. Фирма имеет определенный набор поставщиков, по каждому из которых известны название, адрес и телефон.

У этих поставщиков вы приобретаете детали. Каждая деталь наряду с названием характеризуется артикулом и ценой (считаем цену постоянной).

Некоторые из поставщиков могут поставлять одинаковые детали (один и тот же артикул).

Каждый факт покупки запчастей у поставщика фиксируется в базе данных, причем обязательными для запоминания являются дата покупки и количество приобретенных деталей.

Возможный набор сущностей

Поставщики (Код поставщика, Название, Адрес, Телефон).

Детали (Код детали, Название, Артикул, Цена, Примечание).

Поставки (Код поставщика, Код детали, Количество, Дата).

Вариант № 9. Курсы повышения квалификации

Описание предметной области

Вы работаете в учебном заведении и занимаетесь организацией курсов повышения квалификации.

В вашем распоряжении имеются сведения о сформированных группах студентов.

Группы формируются в зависимости от специальности и отделения. В каждую из них включено определенное количество студентов. Проведение занятий обеспечивает штат преподавателей. Для каждого из них у вас в базе данных зарегистрированы стандартные анкетные данные (фамилия, имя, отчество, телефон) и стаж работы. В результате распределения нагрузки вы получаете информацию о том, сколько часов занятий проводит каждый преподаватель с соответствующими группами. Кроме того, хранятся сведения о типе проводимых занятий (лекции, практика), предмете и оплате за 1 час.

Возможный набор сущностей

Группы (Номер группы, Специальность, Отделение, Количество студентов).

Преподаватели (Код преподавателя, Фамилия, Имя, Отчество, Телефон, Стаж).

Нагрузка (Код преподавателя, Номер группы, Количество часов, Предмет, Тип занятия, Оплата).

Вариант № 10. Определение факультативов для студентов

Описание предметной области

Вы работаете в высшем учебном заведении и занимаетесь организацией факультативов. В вашем распоряжении имеются сведения о студентах, включающие стандартные анкетные данные (фамилия, имя, отчество, адрес, телефон). Преподаватели вашей кафедры должны обеспечить проведение факультативных занятий по некоторым предметам. По каждому факультативу установлены определенное количество часов и вид проводимых занятий (лекции, практика, лабораторные работы). В

результате работы со студентами у вас появляется информация о том, на какие факультативы записался каждый из них. Существует некоторый минимальный объем факультативных предметов, которые должен прослушать каждый студент. По окончании семестра вы заносите информацию об оценках, полученных студентами на экзаменах.

Возможный набор сущностей

Студенты (Код студента, Фамилия, Имя, Отчество, Адрес, Телефон).

Предметы (Код предмета, Название, Объем лекций, Объем практик, Объем лабораторных работ).

Учебный план (Код студента, Код предмета, Оценка).

Вариант № 11. Распределение дополнительных обязанностей

Описание предметной области

Вы работаете в коммерческой компании и занимаетесь распределением дополнительных разовых работ. Вашей задачей является отслеживание хода их выполнения.

Компания имеет определенный штат сотрудников, каждый из которых получает определенный оклад. Время от времени возникает потребность в выполнении некоторой дополнительной работы, не входящей в круг основных должностных обязанностей сотрудников. Для наведения порядка в этой сфере деятельности вы проклассифицировали все виды дополнительных работ, определив сумму оплаты по факту их выполнения. При возникновении дополнительной работы определенного вида вы назначаете ответственного, фиксируя дату начала. По факту окончания вы фиксируете дату и выплачиваете дополнительную сумму к зарплате с учетом вашей классификации.

Возможный набор сущностей

Сотрудники (Код сотрудника, Фамилия, Имя, Отчество, Оклад).

Виды работ (Код вида, Описание, Оплата за день).

Работы (Код сотрудника, Код вида, Дата начала, Дата окончания).

Вариант № 12. Техническое обслуживание станков

Описание предметной области

Ваше предприятие занимается ремонтом станков и другого промышленного оборудования. Вашей задачей является отслеживание финансовой стороны деятельности предприятия.

Клиентами вашей компании являются промышленные предприятия, оснащенные различным сложным оборудованием. В случае поломок оборудования они обращаются к вам.

Ремонтные работы в вашей компании организованы следующим образом: все станки проклассифицированы по странам-производителям, годам выпуска и маркам. Все виды ремонта отличаются названием, продолжительностью в днях, стоимостью. Исходя из этих данных, по каждому факту ремонта вы фиксируете вид станка и дату начала ремонта.

Возможный набор сущностей

Виды станков (Код вида станка, Страна, Год выпуска, Марка).

Виды ремонта (Код ремонта, Название, Продолжительность, Стоимость, Примечания).

Ремонт (Код вида станка, Код ремонта, Дата начала, Примечания).

Вариант № 13. Туристическая фирма

Описание предметной области

Вы работаете в туристической компании, продающей путевки клиентам. Вашей задачей является отслеживание финансовой стороны деятельности фирмы.

Работа с клиентами в вашей компании организована следующим образом: у каждого клиента, пришедшего к вам, собираются некоторые стандартные данные – фамилия, имя, отчество, адрес, телефон. После этого сотрудники выясняют у клиента, где он хотел бы отдыхать. При этом ему демонстрируются различные варианты, включающие страну проживания, особенности местного климата, имеющиеся отели разного класса. Наряду с этим обсуждается возможная длительность пребывания и стоимость путевки. В случае если далось договориться и найти для клиента приемлемый вариант, вы регистрируете факт продажи путевки (или путевок, если клиент покупает сразу несколько путевок), фиксируя дату отправления. Иногда вы решаете предоставить клиенту некоторую скидку.

Возможный набор сущностей

Маршруты (Код маршрута, Страна, Климат, Длительность, Отель, Стоимость).

Путевки (Код маршрута, Код клиента, Дата отправления, Количество, Скидка).

Клиенты (Код клиента, Фамилия, Имя, Отчество, Адрес, Телефон).

Вариант № 14. Грузовые перевозки

Описание предметной области

Вы работаете в компании, занимающейся перевозками грузов. Вашей задачей является отслеживание стоимости перевозок с учетом заработной платы водителей.

Компания осуществляет перевозки по различным маршрутам. Для каждого маршрута вы определили некоторое название, вычислили примерное расстояние и установили некоторую оплату для водителя. Информация о водителях включает фамилию, имя, отчество и стаж. Для проведения расчетов вы храните полную информацию о перевозках (маршрут, водитель, даты отправки и прибытия). По факту некоторых перевозок водителям выплачивается премия.

Возможный набор сущностей

Маршруты (Код маршрута, Название, Дальность, Количество дней в пути, Оплата).

Водители (Код водителя, Фамилия, Имя, Отчество, Стаж).

Проделанная работа (Код маршрута, Код водителя, Дата отправки, Дата возвращения, Премия).

Вариант № 15. Учет телефонных переговоров

Описание предметной области

Вы работаете в коммерческой службе телефонной компании. Компания предоставляет абонентам телефонные линии для междугородних переговоров. Вашей задачей является отслеживание стоимости междугородних телефонных переговоров. Абонентами компании являются юридические лица, имеющие телефонную точку, ИНН, расчетный счет в банке. Стоимость переговоров зависит от города, в который осуществляется звонок, и времени суток (день, ночь). Каждый звонок абонента автоматически фиксируется в базе данных. При этом запоминаются город, дата, длительность разговора и время суток.

Возможный набор сущностей

Абоненты (Код абонента, Номер телефона, ИНН, Адрес).

Города (Код города, Название, Тариф дневной, Тариф ночной).

Переговоры (Код переговоров, Код абонента, Код города, Дата, Количество минут, Время суток)

Вариант № 16. Библиотека

Описание предметной области

Вы являетесь руководителем библиотеки. Ваша библиотека решила зарабатывать деньги, выдавая напрокат некоторые книги, имеющиеся в небольшом количестве экземпляров. Вашей задачей является отслеживание финансовых показателей работы.

У каждой книги, выдаваемой в прокат, есть название, автор, жанр. В зависимости от ценности книги вы определили для каждой из них залоговую стоимость (сумма, вносимая клиентом при взятии книги напрокат) и стоимость проката (сумма, которую клиент платит при возврате книги, получая назад залог). В библиотеку обращаются читатели. Все читатели регистрируются в картотеке, которая содержит стандартные анкетные данные (фамилия, имя, отчество, адрес, телефон). Каждый читатель может обращаться в библиотеку несколько раз.

Все обращения читателей фиксируются, при этом по каждому факту выдачи книги запоминаются дата выдачи и ожидаемая дата возврата.

Возможный набор сущностей

Книги (Код книги, Название, Автор, Залоговая стоимость, Стоимость проката, Жанр).

Читатели (Код читателя, Фамилия, Имя, Отчество, Адрес, Телефон).

Выданные книги (Код книги, Код читателя, Дата выдачи, Дата возврата)

Вариант № 17. Прокат автомобилей

Описание предметной области

Вы являетесь руководителем коммерческой службы в фирме, занимающейся прокатом автомобилей. Вашей задачей является отслеживание финансовых показателей работы пункта проката.

В автопарк входит некоторое количество автомобилей различных марок, стоимостей и типов. Каждый автомобиль имеет свою стоимость проката. В пункт проката обращаются клиенты. Все клиенты проходят обязательную регистрацию, при которой о них собирается стандартная информация (фамилия, имя, отчество, адрес, телефон). Каждый клиент может обращаться в пункт проката несколько раз. Все обращения клиентов фиксируются, при этом по каждой сделке запоминаются дата выдачи и ожидаемая дата возврата.

Возможный набор сущностей

Автомобили (Код автомобиля, Марка, Стоимость, Стоимость проката, Тип).

Клиенты (Код клиента, Фамилия, Имя, Отчество, Адрес, Телефон).

Выданные автомобили (Код автомобиля, Код клиента, Дата выдачи, Дата возврата).

Вариант № 18. Инвестирование свободных средств

Описание предметной области

Вы являетесь руководителем аналитического центра инвестиционной компании, занимающейся вложением денежных средств в ценные бумаги.

Ваши клиенты – предприятия, которые доверяют управлять их свободными денежными средствами на определенный период. Вам необходимо выбрать вид ценных бумаг, которые позволят получить прибыль и компании, и клиенту. При работе с клиентом для вас весьма существенной является информация о предприятии – название, вид собственности, адрес и телефон.

Возможный набор сущностей

Ценные бумаги (Код ценной бумаги, Минимальная сумма сделки, Рейтинг, Доходность за прошлый год, Дополнительная информация).

Инвестиции (Код инвестиции, Код ценной бумаги, Код клиента, Котировка, Дата покупки, Дата продажи).

Клиенты (Код клиента, Название, Вид собственности, Адрес, Телефон).

Вариант № 19. Анализ динамики показателей финансовой отчетности различных предприятий

Описание предметной области

Вы являетесь руководителем информационно-аналитического центра крупного холдинга. Вашей задачей является отслеживание динамики показателей для предприятий холдинга.

В структуру холдинга входят несколько предприятий. Каждое предприятие имеет стандартные характеристики (название, реквизиты, телефон, контактное лицо). Работа предприятия может быть оценена следующим образом: в начале каждого отчетного периода на основе финансовой отчетности вычисляется по неким формулам определенный набор показателей. Важность показателей характеризуется некоторыми числовыми константами.

Значение каждого показателя измеряется в некоторой системе единиц.

Возможный набор сущностей

Показатели (Код показателя, Название, Важность, Единица измерения).

Предприятия (Код предприятия, Название, Банковские реквизиты, Телефон, Контактное лицо).

Динамика показателей (Код показателя, Код предприятия

Контрольные вопросы

1. Перечислить и описать основные элементы ER диаграммы.
2. Перечислить и описать основные элементы реляционной модели данных.
3. Как перевести модель базы данных из концептуальной в реляционную?
4. Какие ограничения целостности данных существуют?
5 . Какие структурные ограничения существуют?
6. Какие виды отношений между сущностями существуют?

2. Лабораторная работа №2. Разработка структуры таблиц реляционной базы данных в среде СУБД Microsoft SQL Server

2.1.Цель работы

Ознакомиться со способами создания, модификации, удаления баз данных, таблиц, отношений между таблицами в SQL Server.

2.2 Содержание работы

Реализация операций создания, модификации, удаления баз данных, таблиц, отношений между таблицами:

- а) с помощью графического интерфейса SQL Server Management Studio;
- с помощью Transact - SQL запросов;

2.3. Задания к работе

1. Изучить синтаксис команд CREATE (ALTER, DROP) DATABASE, CREATE (ALTER, DROP) TABLE, ADD (DROP) CONSTRAINT

2. Создать базу данных из лабораторной работы №1. Создать таблицы, ключи и связи. Реализовать задание средствами

а) SQL Server Management Studio с помощью графического интерфейса;

б) с помощью Transact - SQL запросов;

2.4. Краткая теория

Создание БД

Самый простой способ создать базу данных (БД) — воспользоваться графическим интерфейсом SQL Server Management Studio. Сама процедура создания занимает секунды. Нужно щелкнуть правой кнопкой мыши по контейнеру Database в Object Explorer и в контекстном меню выбрать New Database (Новая база). Откроется диалоговое окно New Database, в котором в самом простом случае вам достаточно будет ввести только имя создаваемой базы данных. Для всех остальных параметров будут подставлены значения по умолчанию.

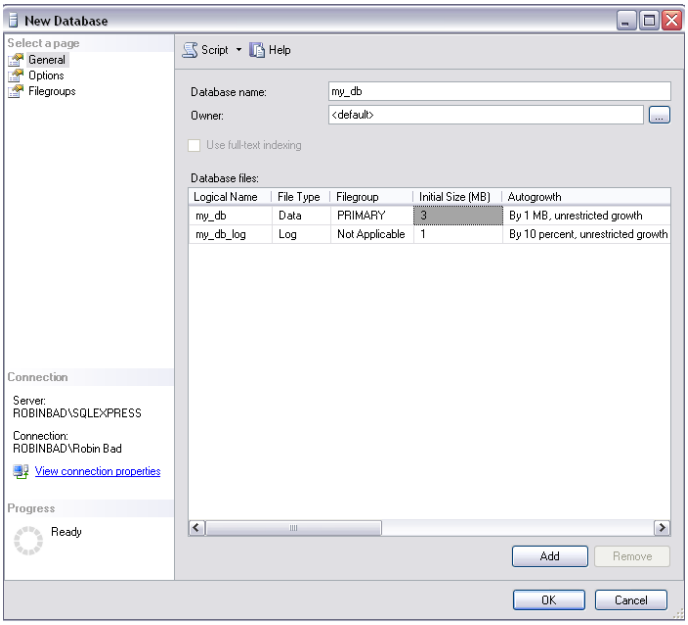


Рис.5. Интерфейс создания БД

В Microsoft SQL Server БД состоит из двух частей:

- файл данных – файл, имеющий расширение mdf и где находятся все таблицы и запросы;

- файл журнала транзакций – файл, имеющий расширение ldf, содержит журнал, где фиксируются все действия с БД. Данный файл предназначен для восстановления БД в случае её выхода из строя.

Очень часто рабочие базы данных создаются при помощи команды Transact-SQL CREATE DATABASE. Обычно эта команда помещается в скрипт, который, помимо создания самой базы данных, выполняет и другие операции, например, настройку параметров базы данных и создание в ней объектов. [4]

Для создания нового файла данных используется команда CREATE DATABASE, которая имеет следующий синтаксис:

CREATE DATABASE <Имя БД>
ON (Name=<Логическое имя>,
FileName=<Имя файла>
[Size=<Нач.размер>,]
[Maxsize=<Макс.размер>,]
[FileGrowth=<Шаг>])
[LOG ON
(Name=<Логическое имя>,
FileName=<Имя файла>
[Size=<Нач.размер>,]
[Maxsize=<Макс.размер >,]
[FileGrowth=<Шаг>])

Здесь:

Имя БД - имя создаваемой БД

Логическое имя - определяет логическое имя файла данных БД, по которому происходит обращение к файлу данных.

Имя файла – определяет полный путь к файлу данных.

Нач. размер – начальный размер файла данных в Мб.

Макс. размер – максимальный размер файла данных в Мб.

Шаг – шаг увеличения файла данных, либо в Мб либо в %.

Параметры в разделе LOG ON аналогичны параметрам в разделе CREATE DATABASE. Однако они определяют параметры журнала транзакций. [9]

Для создания запроса Transact-SQL можно воспользоваться кнопкой New Query панели задач.

Рис. 6. Меню создания нового запроса

В появившемся окне вводится текст запроса:

CREATE DATABASE MY_DATABASE

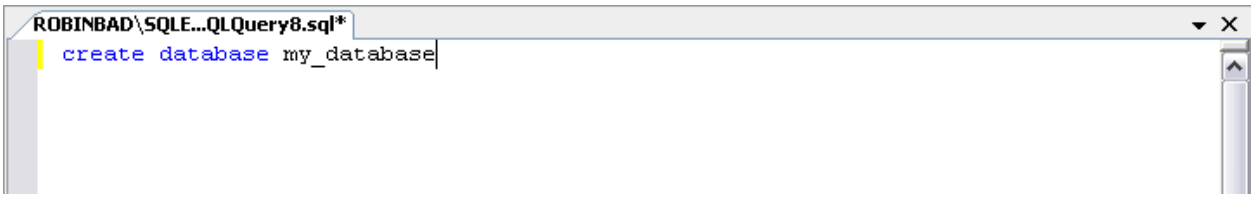


Рис. 7. Окно ввода тела запроса

Затем нужно нажать F5 или кнопку 

на панели задач. Для того, чтобы увидеть созданную БД в Object Explorer, воспользуйтесь командой Refresh (доступной из контекстного меню контейнера Databases).

Создание таблиц

Таблицы, как и базы данных, можно создавать двумя способами (с использованием запросов или графического интерфейса).

1. Выберите из контекстного меню контейнера Table созданной вами таблицы команду New Table. Откроется окно следующего вида:

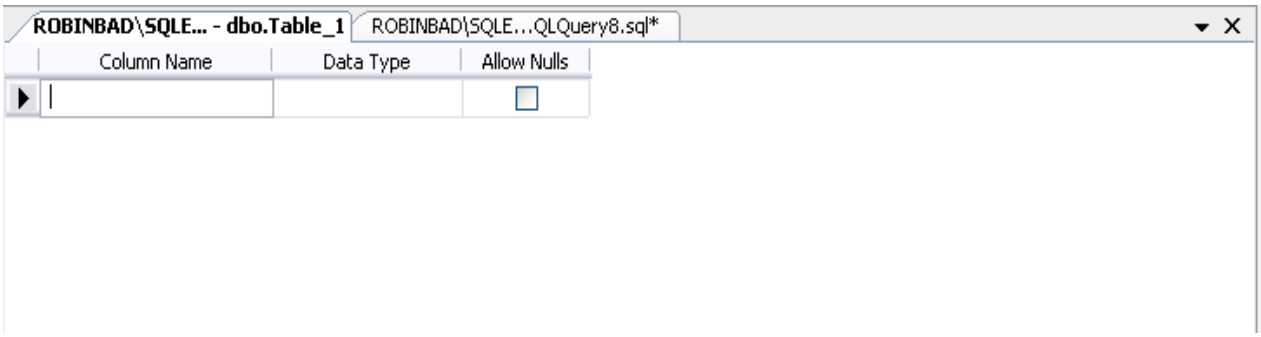


Рис. 8. Окно столбцов таблицы

2. Создайте столбцы со следующими сведениями:

Таблица 2
Столбцы

Имя столбца	Тип данных	Разрешить значения null
CustomerID	nchar(5)	False (флажок не установлен)
CompanyName	nvarchar(40)	False (флажок не установлен)
ContactName	nvarchar (30)	True (флажок установлен)
ContactTitle	nvarchar (30)	True (флажок установлен)
Address	nvarchar (60)	True (флажок установлен)
City	nvarchar (15)	True (флажок установлен)
Region	nvarchar (15)	True (флажок установлен)
PostalCode	nvarchar (10)	True (флажок установлен)
Country	nvarchar (15)	True (флажок установлен)
Phone	nvarchar (24)	True (флажок установлен)
Fax	nvarchar (24)	True (флажок установлен)

Окно таблицы примет следующий вид (рис. 9):

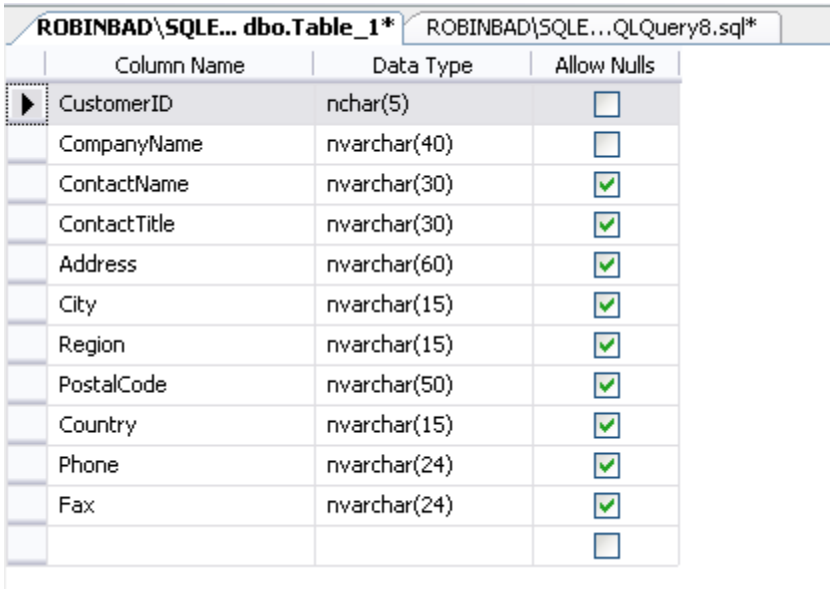


Рис. 9. Результирующий вид таблицы

3. Щелкните правой кнопкой мыши на названии столбца CustomerID и выберите пункт Set Primary Key.

Первичный ключ (primary key) – поле (столбец) или группу полей таблицы базы данных, значение которого (или комбинация значений которых) используется в качестве уникального идентификатора записи (строки) этой таблицы.

4. Выберите пункт Save Table 1 в меню File.

5. Введите Customers в поле для имени таблицы.

Для создания таблицы с помощью запроса предназначена инструкция CREATE TABLE (Transact-SQL).

CREATE TABLE <Имя таблицы>(<Имя поля1> <Тип1> [IDENTITY NULL|NOTNULL],<Имя поля2> <Тип2>, …)

Здесь:

<Имя таблицы> – имя создаваемой таблицы;

<Имя поля> – имена полей таблицы;

<Тип> – типы полей;

<IDENTITY NULL|NOT NULL> - поле счётчик. [10]

6. Выполните нижеприведенный запрос для создания таблицы Orders.

```
CREATE TABLE Orders
(
        OrderID int NOT NULL IDENTITY(1,1)
                PRIMARY KEY,
        CustomerID nchar(5) NULL,
        EmployeeID int NULL,
```

```
    OrderDate datetime NULL,
    RequiredDate datetime NULL,
    ShippedDate datetime NULL,
    ShipVia int NULL,
    Freight money NULL,
    ShipName nvarchar(40) NULL,
    ShipAddress nvarchar(60) NULL,
    ShipCity nvarchar(15) NULL,
    ShipRegion nvarchar(15) NULL,
    ShipPostalCode nvarchar(10) NULL,
    ShipCountry nvarchar(15) NULL
);
```

Свойство IDENTITY создает в таблице столбец идентификаторов. В скобках указывается начальное значение и приращение.

Просмотреть созданную таблицу можно, выбрав из контекстного меню контейнера – названия таблицы пункт Design.

Создание отношений между таблицами

Создание внешнего ключа с помощью диаграммы базы данных

Внешний ключ (FK) – это столбец или сочетание столбцов, которое применяется для принудительного установления связи между данными в двух таблицах. Внешний ключ можно создать, определив ограничение FOREIGN KEY при создании или изменении таблицы.

Если один или несколько столбцов, в которых находится первичный ключ для одной таблицы, упоминается в одном или нескольких столбцах другой таблицы, то в ссылке внешнего ключа создается связь между двумя таблицами. Этот столбец становится внешним ключом во второй таблице. [10]

1. В контекстном меню Database Diagrams выберите пункт New Database Diagram. (рис. 10)

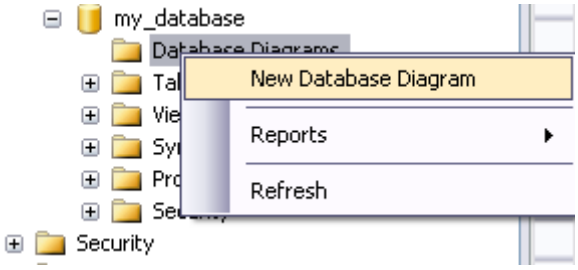


Рис. 10 Новая диаграмма

2. Добавьте на диаграмму таблицы Customers и Orders. (рис.11)

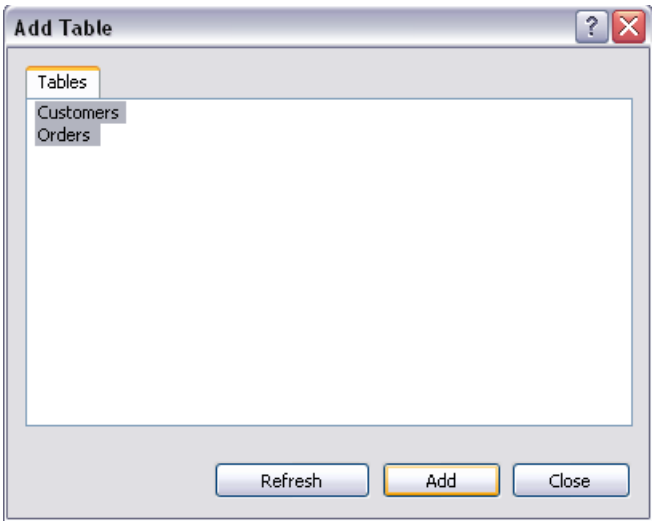


Рис. 11. Добавление таблиц

3. Перетащите столбец CustomerID из таблицы Customers в таблицу Orders. (рис. 12)

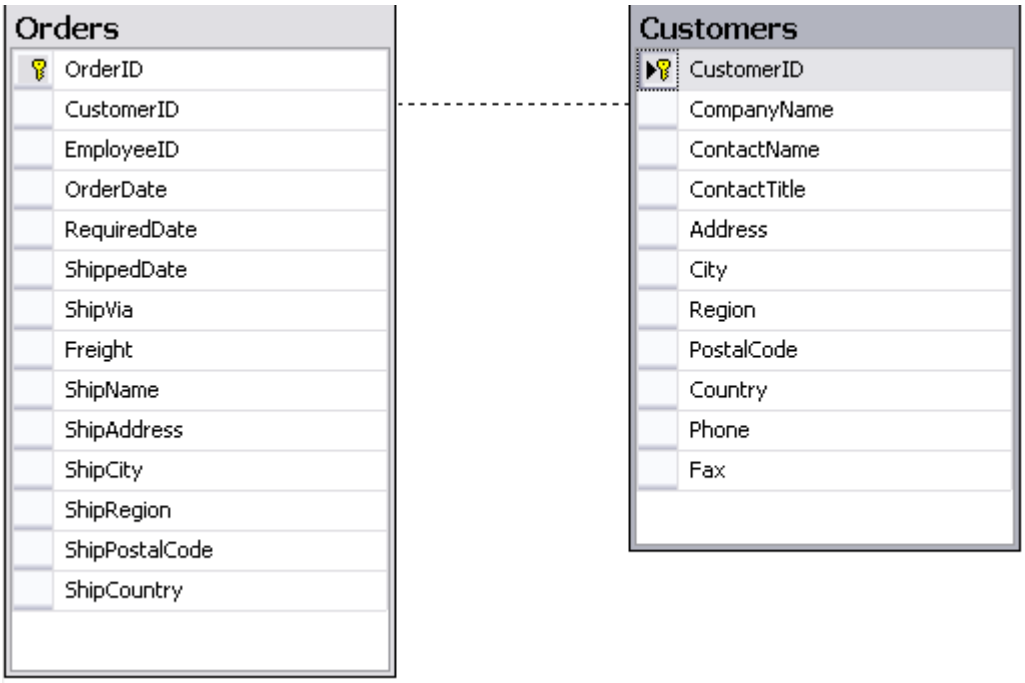


Рис. 12. Установка связи

4. Убедитесь, что таблица Customers является таблицей первичного ключа, а таблица Orders является таблицей внешнего ключа, и что столбец выбран в обеих таблицах CustomerID. (рис. 13)

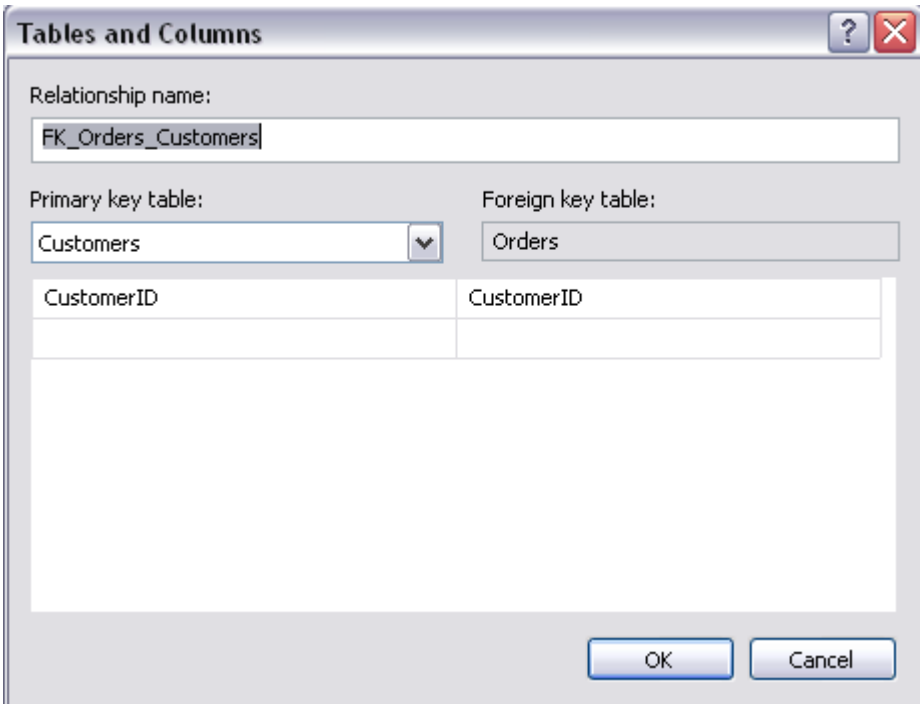


Рис. 13. Создание внешнего ключа

5. Нажмите кнопку OK, чтобы закрыть диалоговое окно Таблицы и столбцы.

6. Для удаления созданного внешнего ключа щелкните правой кнопкой мыши по имени внешнего ключа (my_database -> Tables -> dbo.Orders -> Keys -> FK_Orders_Customers) в окне диаграммы и выберите Delete. (рис. 14)

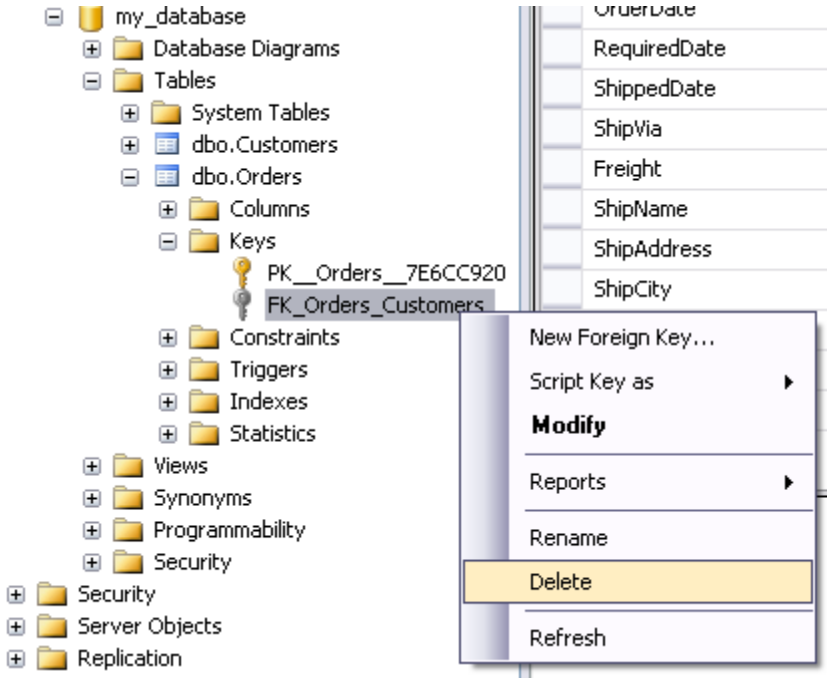


Рис. 14. Создание внешнего ключа

Создание внешнего ключа средствами Transact SQL

1. Для добавления внешнего ключа в ранее созданную таблицу нужно воспользоваться инструкцией модификации таблицы ALTER TABLE:

```
ALTER TABLE Orders
    ADD CONSTRAINT FK_Orders_Customers FOREIGN KEY (CustomerID)
        REFERENCES Customers(CustomerID);
```

2. Для удаления внешнего ключа также используется ALTER TABLE:

```
ALTER TABLE Orders
    DROP CONSTRAINT FK_Orders_Customers;
```

Контрольные вопросы

1. Какими способами можно создать базу данных?
2. Какими способами можно создать таблицы?
3. Какими способами можно задать связи?
4. Для чего используются диаграммы?
5. Какими способами можно изменять таблицы в базе данных
6. Для чего используется свойство IDENTITY?

3. Лабораторная работа №3. Заполнение таблиц данными

3.1. Цель работы

Освоить методы заполнения таблиц в SQL Server.

3.2 Содержание работы

- Заполнение таблиц через Management Tool.
- Заполнение таблиц через запросы.
- Изменение таблиц
- Вывод содержимого таблиц.

3.3. Задание к работе

1. Заполнить базу данных из Лабораторной работы № 2 данными, используя

- Management Tool Б
- запросы SQL

Вывести на экран содержимое таблиц

2. Используя запросы SQL удалить одну запись из таблицы, использовать условие.

Вывести на экран новое содержимое таблиц

3. Используя запросы SQL изменить одно поле в двух записях из двух разных таблиц, использовать условие.

Вывести на экран новое содержимое таблиц

3.4. Краткая теория

Использование Management Tool для заполнения таблиц.

Пусть имеется База данных «Деканат» с таблицами Оценки, Предметы, Специальности. (см рис. 15.)

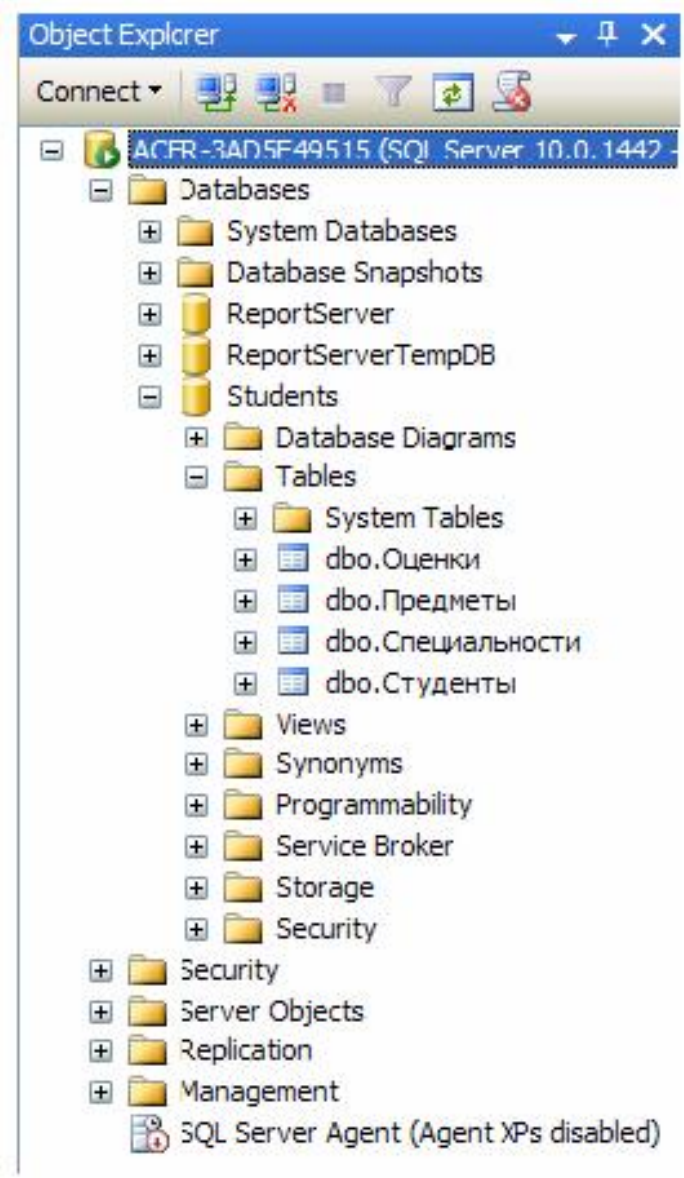


Рис.15. Таблицы базы данных «Деканат»

Рассмотрим операцию заполнения таблиц начальными данными.

Для начала заполним таблицу **"Специальности"**. Для заполнения этой таблицы в обозревателе объектов щелкните 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. Далее заполняем таблицу **"Специальности"**, как показано на рис.16. Так как поле **"Код специальности"** является первичным полем связи и ключевым числовым счетчиком, то оно заполняется автоматически (заполнять его не нужно).

	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

Рис. 16. Заполнение таблицы специальности

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

Вывод на экран содержимого таблиц

В обозревателе объектов щелкните правой кнопкой мыши по таблице и в появившемся меню выберите пункт **"Select Top 200 Rows"** (Выбрать первые 200 записей.). В рабочей области "Microsoft SQL Server Management Studio" проявится окно с содержимом таблицы.

Использование SQL запросов

Заполнение отдельных столбцов и отдельных строк из таблицы

В SQL Server заполнение таблиц производится при помощи следующей команды:

INSERT INTO <Имя таблицы> [(<Список полей>)]

VALUES (<Значения полей>)

где <Имя таблицы> - таблица, куда вводим данные, (<Список полей>) - список полей, куда вводим данные, если не указываем, то подразумевается заполнение всех полей, в списке полей поля указываются через запятую, (<Значения полей>) - значение полей через запятую.

В качестве значений можно указать константу Default, то есть будет поставлено значение по умолчанию, либо можно подставить оператор Select. Здесь он используется как инструмент вычисления формул. [1]

Пример: Добавление записи имеющей следующие значения полей **ФИО = Иванов**, **Адрес = Москва**, **Код специальности = 5** в таблицу **"Студенты"**.

INSERT INTO Студенты (ФИО, Адрес, [Код специальности])

VALUES ('Иванов А.А.', 'Москва', 5)

Удаление отдельных столбцов и отдельных строк из таблицы

Из таблицы можно удалить все строки, либо отдельные записи. Это осуществляется командой

DELETE FROM <Имя таблицы>

[WHERE <Условие>]

где <Условие> - условия, которым удовлетворяют удаляемые записи. Если условия не указаны, то удаляются все строки таблицы. Если условия указаны, то удаляются записи, поля которых соответствуют условию. [1]

Пример: Удалить записи из таблицы **"Студенты"**, у которых поле **Адрес = Москва**.

DELETE FROM Студенты

WHERE Адрес = 'Москва'

Изменение данных в таблице

Для этого используется следующая команда:

UPDATE <Имя таблицы>

SET

<Имя поля1> = <Выражение1>,

[<Имя поля2> = <Выражение2>,]

…

[WHERE <Условие>]

Здесь <Имя поля1>, <Имя поля2> - имена изменяемых полей, <Выражение1>, <Выражение 2> - либо конкретные значения, либо NULL, либо операторы SELECT. Здесь SELECT применяется как функция.

<Условие> - условие, которым должны соответствовать записи, поля которых изменяем. [4]

Пример: В таблице **"Студенты"** у студента Иванова А.А. поменять адрес Москва на Йошкар-Ола, а код специальности вместо 5 поставить 3.

UPDATE Студенты

SET

Адрес = 'Йошкар-Ола',

[Код специальности] = 3

WHERE ФИО = 'Иванов А.А.'

Вывод на экран содержимого таблиц

Для этого используется следующая команда:

SELECT * FROM <имя таблицы>

Контрольные вопросы

1. Каким образом можно заполнять записи в таблицы SQL Server?
2. Как можно вывести на экран содержимое таблиц?
3. Каким образом можно удалять записи из таблицы SQL Server?
4. Каким образом можно изменять записи в таблицах SQL Server?
5. Пусть имеются таблицы Автор и Книги. Их данные агрегируется в таблице Библиотека (поля связи Код автора, Код книги). Таблица Библиотека содержит столбцы Код записи, Код автора, Код книги, Цена. Написать SQL скрипт по обновлению одной записи из таблицы Библиотека, чтобы информация заносилась так в связанные с ней таблицы

4. Лабораторная работа №4. Создание запросов. Представления. Объединения

4.1. Цель работы

Изучить создание запросов и представлений, использование механизма объединений.

4. 2 Содержание работы

- Способы создания запросов, запросов с подзапросами
- Использование представлений
- Использование механизма объединений

4.3. Задание к работе

- Для базы данных согласно варианту задания (из Лабораторной работы №1) написать следующие SQL запросы из Приложения 2
- Реализовать запросы из лабораторной работы 1 по варианту, оформить два из них в виде представлений.

4.4. Краткая теория

Создание запросов

Запросы предназначены для связи одной или нескольких таблиц, также они могут осуществлять отбор отдельных полей из таблицы и производить фильтрацию данных согласно условию, наложенному на одно или несколько полей, такие запросы называют фильтрами.

Для реализации запросов используют специальный язык запросов SQL (Structured Query Language). [4]

Чтобы создать запрос необходимо сделать активной БД для которой создается запрос, затем в рабочей области редактора запросов создать запрос с помощью команды SELECT, имеющей следующий синтаксис:

SELECT [ALL|DISTINCT]
[TOP|PERCENT n]
<Список полей>
[INTO <Имя новой таблицы>]
[FROM <Имя таблицы >]
[WHERE <Условие>]
[GROUP BY <Поле>]
[ORDER BY <Поле > [ASC|DESC]]
[COMPUTE AVG|COUNT|MAX|MIN|SUM(<Выражение>)]

Здесь параметры ALL|DISTINCT показывают, какие записи обрабатываются: ALL обрабатывает все записи, DISTINCT только уникальные, удаляются повторения записей.

TOP n определяет какое количество записей обрабатывают, если указан PERCENT, то n указывает процент от общего числа записей. <Список полей> - здесь указываются отображаемые поля из таблиц через запятую.

Раздел INTO. Если присутствует этот раздел, то на основе результатов запроса создается новая таблица.

Раздел FROM. Здесь указываются таблицы и запросы, через запятую, которые участвуют в новом запросе.

Раздел WHERE. Данный раздел используют для создания простых запросов, в этом случае в качестве условия указываем связываемые поля, либо этот раздел используют для создания фильтров, здесь указывают условия отбора. В условиях отбора мы можем использовать стандартные логические операторы NOT, OR, AND.

Раздел GROUP BY - определяет поле для группировки записей в запросе.

Раздел ORDER BY - определяет поле для сортировки записей в запросе. Если указан параметр ASC, то будет производиться сортировка по возрастанию, если DESC – по убыванию. По умолчанию используется сортировка по возрастанию.

Раздел COMPUTE позволяет в конце результатов выполнения запроса вывести некоторые итоговые вычисления по запросу. Возможны следующие виды вычислений: AVG – средняя параметра; COUNT – количество значений параметра не равных NULL; MAX и MIN – максимальные и минимальные значения параметра; SUM - сумма всех значений параметра, где <Выражение> – сам параметр. В качестве

параметра обычно выступают какие-либо поля таблиц, участвующих в запросе. [12]

Пример: Данный запрос связывает две таблицы **Сотрудники** и **Должности** по полям **Код**. При своем выполнении он отображает первые 20 процентов сотрудников из обеих таблиц. Из таблицы сотрудники отображаются все поля, а из таблицы Должности только поле должность. В конце результатов выводится количество отображенных сотрудников.

```
SELECT TOP 20 PERCENT *. Сотрудники, Должность.Должности
FROM Сотрудники, Должности
WHERE Код.Сотрудники = Код.Должности
COMPUTE COUNT (ФИО.Сотрудники)
```

Пример: Данный запрос из таблицы **Операции** выводит все записи, значение поля **Месяц** у которых равняется "Май". Данные в результате группируются по полю операция и сортируются по сумме операции. В конце результатов запроса отображается общая сумма отобранных операций за май. Результаты данного запроса сохраняются в таблице **"Сделки за май"**.

```
SELECT ALL Операция, Сумма
INTO [Сделки за Май]
FROM Операции
WHERE Месяц = 'Май'
GROUP BY Операция
ORDER BY Сумма
COMPUTE SUM (Сумма)
```

Выполнение вычислений при помощи оператора SELECT. Встроенные функции

Кроме связывания таблиц и отбора данных оператор SELECT может использоваться для вычислений. В этом случае он имеет синтаксис:

SELECT <Выражение>

где <выражение> – какое-то математическое выражение или функция. Выражение имеет стандартный вид, оно может включать в себя встроенные функции сервера.

В SQL Server существуют следующие встроенные функции, разбитые на группы.

Математические функции

Замечание: В качестве параметров функции будем указывать соответствующий им тип данных.

Например,

FLOOR(Numeric) – наибольшее целое меньшее или равное выражению **numeric**;

- POWER (Numeric,y) – возводит выражение **Numeric** в степень **y** ;

- RAND () – генерирует случайное число типа данных **Float**, расположенное между нулем и единицей;
- ROUND (Numeric, Длина) – округляет выражение **Numeric** до заданной **Длины** (количество знаков после запятой);
- SIGN (Numeric) – выводит знак числа +/- или ноль;
- SQRT (Float) – вычисляет квадратный корень числа **Float**.

Примеры использования математических функций:

- SELECT SQRT (16) результат 4
- SELECT ROUND (125.85,0) результат 126
- SELECT POWER (2,4) результат 16

Строковые функции

Строковые функции позволяют производить операции с одной или несколькими строками.

Например:

- 'Строка1'+ 'Строка2' присоединяет **Строку1** к **Строке2**;
- REPLACE (Строка1, Строка2, Строка3) – меняет в **Строке1** все элементы **Строка2** на элементы **Строка3** ;
- STR(Float) – переводит число **Float** в строку;
- SUBSTRING(Выражение, Начало, Длина) – из **Выражения** выводится строка заданной **Длины** начиная с позиции **Начало** ;
- LOWER(Char) – переводит строку **Char** в маленькие буквы;
- UPPER(Char) – переводит строку **Char** в заглавные буквы.

Примеры применения строковых функций:

- SELECT LOWER('ABC') результат abc.

Замечание. Во всех строковых функциях значения выражения типа Char заключаются в одинарные кавычки.

Функции дат

Замечание: в некоторых функциях дат используется так называемая часть дат, которая кодируется специальными символами. Например,

dd – число дат (от 1 до 31);**dy** – день года (число от 1 до 366); **hh** – значение часа (0-23); **ms** – значение милисекунд (от 0 до 999); **mi** – значение минут (0-59); **qq** – значение (1-4); **mm** – значение месяцев (1-12); **ss** –- значение секунд (0-59); **wk** – значение номеров недель в году; **dw** – значение дней недели, неделя начинается с воскресенья (1-7).; **yy** – значение лет (1753 -999)

Функции дат предназначены для работы с датами или времени. Существуют несколько следующие функции дат:

- DATEADD(часть, число, date) – добавляет к дате **date** часть даты увеличенное на число;
- DAY(date) – выводит количество дней в заданной дате;
- MONTH (date) – выводит количество месяцев в заданной дате;
- YEAR(date) – выводит количество лет в заданной дате;

- GETDATE() – выводит текущую дату установленную на компьютере;

Замечание: Даты выводятся в Американском формате: месяц/день/год.

Примеры функции работ с датами:

- SELECT DATEADD(dd,5,11/20/07) результат Nov/25/2007.

Замечание: В выражениях оператора SELECT можно использовать операции сравнения. В результате будет либо истина TRUE, либо ложь FALSE. Можно использовать следующие операторы: **=, <, >, >=, <=, <>, !<(не меньше), !>(не больше), !=(не равно)**. Приоритет операции задается круглыми скобками.

Системные функции

Системные функции предназначены для получения информации о базе данных и ее содержимом. В SQL сервере существуют следующие системные функции, например

- ISNUMERIC(выражение) – выводит единицу, если выражение является числовым и ноль, если не числовым;
- NULIFF(выражение1, выражение2) – выводит NULL если **выражение1** равно **выражению 2**.

Агрегатные функции

Агрегатные функции – позволяют вычислять итоговые значения по полям таблицы.

- AVG(поле) – выводит среднее значение поля;
- COUNT(*) – выводит количество записей в таблице;
- COUNT(поле) – выводит количество всех значений поля;
- MAX(поле) – выводит максимальное значение поля;
- MIN(поле) – выводит минимальное значение поля;
- SUM(поле) – суммирует все значения поля;
- TOP n [Percent] – выводит n первых записей из таблицы, либо n% записей из таблицы;

Примеры использования агрегатных функций:

- SELECT AVG(возраст) FROM Студенты – выводит средний возраст студента из таблицы **"Студенты"**.
- SELECT COUNT(ФИО) FROM Студенты – выводит количество различных **ФИО** из таблицы **"Студенты"**.
- SELECT Top 100 * FROM Студенты – выводит первые 100 студентов из таблицы **"Студенты"**.[4]

Представления

Представление (VIEW) – это виртуальная таблица, созданная на основе запроса к обычным таблицам. Представление реализовано как запрос, хранящийся на сервере и выполняющийся всякий раз, когда происходит обращение к представлению.

Синтаксис:

CREATE VIEW viewname [(view_column [, view_column...])]

AS <SELECT> [WITH CHECK OPTION];

Здесь viewname - имя представления, которое должно быть уникальным в пределах базы данных, далее идет группа не всегда обязательных наименований полей, входящих в представление: [(view_column [, view_column...])]. Обязательно необходимо определить предложение <SELECT>, которое выбирает данные, включаемые в представление.

Чтобы изменить какое-либо представление, придется его пересоздать, т. е. удалить и создать заново. При удалении представления необходимо также удалить все зависимые от него объекты – триггеры, хранимые процедуры и другие представления. [13]

Чтобы удалить представление, необходимо воспользоваться следующей командой:

DROP VIEW viewname;

Примеры представлений

Вот пример простого представления:

CREATE VIEW MyView AS

SELECT NAME, PRICE_1

FROM Table_example;

В данном случае представление будет состоять из двух полей – NAME и PRICE_1, которые будут выбираться из таблицы Table_example без всяких условий, т. е. число записей в представлении MyView будет равно числу записей в Table_example.

Однако представления не всегда являются такими простыми. Они могут основываться на данных из нескольких таблиц и даже на основе других представлений. Также представления могут содержать данные, получаемые на основе различных выражений – в том числе на основе агрегатных функций.

Таким образом, легко создавать представления, которые исполняют роль постоянно обновляемых поставщиков данных, отбирая их из базы данных по определенным условиям. [1]

Объединения JOIN

Объединения JOIN – это объединение двух или более объектов базы данных по средствам определенного ключа или ключей или в случае cross join и вовсе без ключа. Под объектами здесь подразумевается различные таблицы, представления (views), табличные функции или просто подзапросы sql, т.е. все, что возвращает табличные данные.

Объединение SQL Left и Right join

Left join – это объединение данных по левому ключу, т.е. допустим, мы объединяем две таблицы по left join, и это значит что все данные из

второй таблицы подтянутся к первой, а в случае отсутствия ключа выведется NULL значения, другими словами выведутся все данные из левой таблицы и все данные по ключу из правой таблицы.

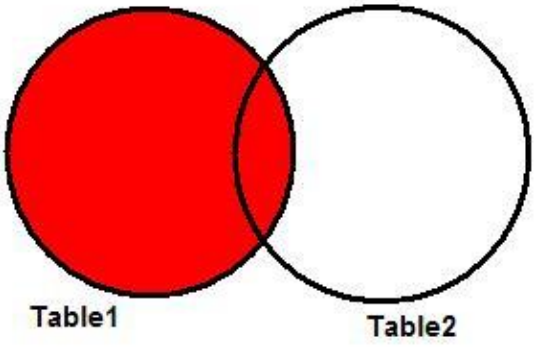


Рис.17. LEFT JOIN.

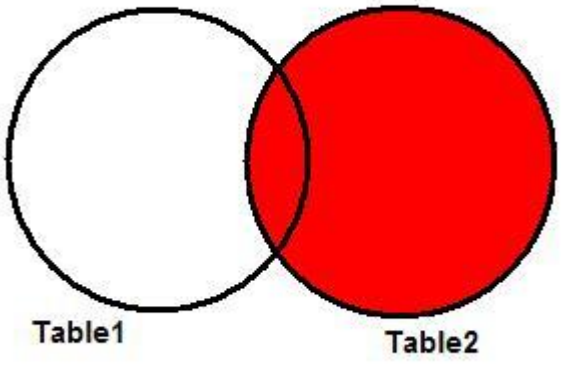


Рис.18. RIGHT JOIN

Right join – это такое же объединение как и Left join только будут выводиться все данные из правой таблицы и только те данные из левой таблицы в которых есть ключ объединения.

Объединение SQL Inner join

Inner join – это объединение когда выводятся все записи из одной таблицы и все соответствующие записи из другой таблице, а те записи которых нет в одной или в другой таблице выводиться не будут, т.е. только те записи которые соответствуют ключу.

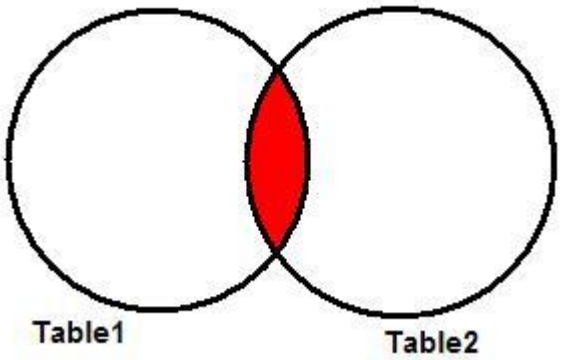


Рис. 19. INNER JOIN

Объединение SQL Cross join

Cross join – это объединение SQL по которым каждая строка одной таблицы объединяется с каждой строкой другой таблицы. [9]

Приложение. Варианты заданий.

Вариант 1

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос "Филиалы по заданному адресу", который выдает список филиалов, находящихся в заданном городе;

c. Запрос "Алфавитный список", который выдает список видов страхования в указанном буквенном диапазоне. Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все договоры для заданного филиала.

e. Запрос "Вид страхования, преобразованный к верхнему регистру", который выдает список всех Филиалов, преобразованных к верхнему регистру.

f. Запрос "Филиалы и вид страхования", который выдает список всех филиалов и видов страхований, предлагаемых у них.

g. Запрос с расчетами – найти максимальную страховую сумму.

p. Запрос с групповой операцией – найти среднюю страховую сумму за месяц.

Вариант 2

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос "Номера по заданному адресу", который выдает список номеров, с заданной ценой;

c. Запрос "Алфавитный список", который выдает список фамилий в указанном буквенном диапазоне.

Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все номера на заданное количество человек.

e. Запрос Фамилия, преобразованная к верхнему регистру", который выдает список всех фамилий, преобразованных к верхнему регистру.

f. Запрос "Клиенты и номера", который выдает список клиентов, проживающих в номерах.

g. Запрос с расчетами – найти самый дорогой номер.

p. Запрос с групповой операцией – найти количество клиентов гостиницы за месяц.

Вариант 3

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос "Категории товаров для клиентов", который выдает список категорий товаров, для заданных клиентов;

c. Запрос "Алфавитный список", который выдает список фамилий в указанном буквенном диапазоне.

Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все товары со стоимостью меньше заданной.

e. Запрос Фамилия, преобразованная к верхнему регистру", который выдает список всех фамилий, преобразованных к верхнему регистру.

f. Запрос "Клиенты и товары", который выдает список клиентов, сдавших товары.

g. Запрос с расчетами – найти самый дорогой товар.

p. Запрос с групповой операцией – найти среднюю сумму ломбарда за месяц.

Вариант 4

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос "Покупатели по заданному адресу", который выдает список покупателей, с заданным городом;

c. Запрос "Алфавитный список", который выдает список контактных лиц в указанном буквенном диапазоне.

Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все оптовые сделки.

e. Запрос Контактное лицо, преобразованная к верхнему регистру", который выдает список всех контактных лиц, преобразованных к верхнему регистру.

f. Запрос "Покупатели и сделки", который выдает список покупателей, купивших заданный товар.

g. Запрос с расчетами – найти самый дорогой товар.

p. Запрос с групповой операцией – найти количество сделок за месяц.

Вариант 5

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос "Заказчики по заданному адресу", который выдает список заказчиков, с заданным городом;

c. Запрос "Алфавитный список", который выдает список контактных лиц в указанном буквенном диапазоне.

Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все заказы заданного заказчика.

e. Запрос Контактное лицо, преобразованная к верхнему регистру", который выдает список всех контактных лиц, преобразованных к верхнему регистру.

f. Запрос "Заказчики и заказы ", который выдает список заказчиков, заказавших заданный товар.

g. Запрос с расчетами – найти самый дорогой товар.

p. Запрос с групповой операцией – найти количество заказов за месяц.

Вариант 6

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос "Работодатели по заданному адресу", который выдает список покупателей, с заданным городом;

c. Запрос "Алфавитный список", который выдает список фамилий соискателей в указанном буквенном диапазоне.

Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все сделки для заданной должности.

e. Запрос Фамилия, преобразованная к верхнему регистру", который выдает список всех фамилий, преобразованных к верхнему регистру.

f. Запрос " Соискатели и сделки", который выдает список соискателей, рассмотревших данного работодателя.

g. Запрос с расчетами – найти самого высокооплачиваемого соискателя.

p. Запрос с групповой операцией –среднюю комиссионную.

Вариант 7

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить

повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос " Клиенты по заданному адресу", который выдает список Клиентов, с заданным городом;

c. Запрос "Алфавитный список", который выдает список Названий Клиентов в указанном буквенном диапазоне.

Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все сделки для заданного Клиента.

e. Запрос Название, преобразованное к верхнему регистру", который выдает список всех Названий Клиентов, преобразованных к верхнему регистру.

f. Запрос " Клиенты и сделки", который выдает список клиентов, участвующих в заданных сделках.

g. Запрос с расчетами – найти самую высокооплачиваемую сделку.

p. Запрос с групповой операцией – среднюю комиссионную.

Вариант 8

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос " поставщики по заданному адресу", который выдает список поставщиков, с заданным городом;

c. Запрос "Алфавитный список", который выдает список Названий деталей в указанном буквенном диапазоне.

Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все детали для заданной поставки.

e. Запрос Название, преобразованное к верхнему регистру", который выдает список всех Названий Клиентов, преобразованных к верхнему регистру.

f. Запрос " Детали и поставки", который выдает список артикулов деталей, участвующих в заданных поставках.

g. Запрос с расчетами – найти самую высокооплачиваемую деталь.

p. Запрос с групповой операцией – найти среднюю количество деталей в поставке.

Вариант 9

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос "Преподаватели со стажем ", который выдает список преподавателей с заданным стажем

c. Запрос "Алфавитный список", который выдает список фамилий в указанном буквенном диапазоне. Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все группы заданного отделения.

e. Запрос Фамилия, преобразованная к верхнему регистру", который выдает список всех фамилий, преобразованных к верхнему регистру.

f. Запрос " Преподаватели и предметы", который выдает список предметов, которые ведет преподаватель

g. Запрос с расчетами – найти самого высокооплачиваемого преподавателя.

p. Запрос с групповой операцией – найти среднее количество студентов в группе.

Вариант 10

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос "Студенты и предметы ", который выдает список студентов, изучающих заданный предмет

c. Запрос "Алфавитный список", который выдает список фамилий в указанном буквенном диапазоне. Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все студенты из заданного города.

e. Запрос Фамилия, преобразованная к верхнему регистру", который выдает список всех фамилий, преобразованных к верхнему регистру.

f. Запрос " Оценки и предметы", который выдает список оценок, которые были получены по предмету

g. Запрос с расчетами – найти самый длительный предмет.

p. Запрос с групповой операцией – найти среднюю оценку по предмету.

Вариант 11

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос "Сотрудники и виды работ ", который выдает список сотрудников, выполняющих заданный вид работ

c. Запрос "Алфавитный список", который выдает список фамилий в указанном буквенном диапазоне. Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все сотрудники с заданным окладом.

e. Запрос Фамилия, преобразованная к верхнему регистру", который выдает список всех фамилий, преобразованных к верхнему регистру.

f. Запрос " Виды работ", который выдает список работ, которые были выполнены за месяц.

g. Запрос с расчетами – найти самую высокооплачиваемую работу.

p. Запрос с групповой операцией – найти средний оклад сотрудников.

Вариант 12

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос " Виды ремонта ", который выдает список видов ремонта для заданного станка

c. Запрос "Алфавитный список", который выдает список марок станков в указанном буквенном диапазоне. Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все станки с заданным годом выпуска.

e. Запрос Название ремонта, преобразованное к верхнему регистру", который выдает список всех названий ремонта, преобразованных к верхнему регистру.

f. Запрос " Ремонт", который выдает список ремонтных работ, которые были выполнены к текущей дате.

g. Запрос с расчетами – найти самый дорогой ремонт.

p. Запрос с групповой операцией – найти среднюю продолжительность ремонта.

Вариант 13

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос "Клиенты и маршруты ", который выдает список клиентов, выбравших заданный маршрут

c. Запрос "Алфавитный список", который выдает список фамилий в указанном буквенном диапазоне. Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все маршруты для заданной страны.

e. Запрос Фамилия, преобразованная к верхнему регистру", который выдает список всех фамилий, преобразованных к верхнему регистру.

f. Запрос " Путевки и клиенты", который выдает список путевок, которые были куплены данным клиентом

g. Запрос с расчетами – найти самый длительный маршрут.

p. Запрос с групповой операцией – найти среднюю стоимость маршрута.

Вариант 14

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос "Водители и маршруты ", который выдает список водителей, на заданном маршруте

c. Запрос "Алфавитный список", который выдает список фамилий в указанном буквенном диапазоне. Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все маршруты для заданного водителя.

e. Запрос Фамилия, преобразованная к верхнему регистру", который выдает список всех фамилий, преобразованных к верхнему регистру.

f. Запрос " Работа и водители", который выдает список работ, которые были куплены данным водителем

g. Запрос с расчетами – найти самый длительный маршрут.

p. Запрос с групповой операцией – найти среднюю дальность маршрута.

Вариант 15

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос "Абоненты и города", который выдает список абонентов, из заданного города

c. Запрос "Алфавитный список", который выдает список городов в указанном буквенном диапазоне. Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все переговоры для заданного абонента.

e. Запрос Название, преобразованное к верхнему регистру", который выдает список всех название городов, преобразованных к верхнему регистру.

f. Запрос " Переговоры и абоненты", который выдает список переговоров, которые были совершены данным абонентом

g. Запрос с расчетами – найти самый длительный переговор.

p. Запрос с групповой операцией – найти среднюю дальность переговора.

Вариант 16

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос "Книги и читатели ", который выдает список читателей, взявших данную книгу

c. Запрос "Алфавитный список", который выдает список фамилий в указанном буквенном диапазоне. Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все книги заданного жанра.

e. Запрос Фамилия, преобразованная к верхнему регистру", который выдает список всех фамилий, преобразованных к верхнему регистру.

f. Запрос " Выданные книги ", который выдает список книг, которые были выданы к данной дате данным водителем

g. Запрос с расчетами – найти самую дорогую книгу.

p. Запрос с групповой операцией – найти среднюю стоимость проката.

Вариант 17

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос "Автомобили и клиенты ", который выдает список клиентов, взявших на прокат данный автомобиль

c. Запрос "Алфавитный список", который выдает список фамилий в указанном буквенном диапазоне. Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все автомобили заданной марки.

e. Запрос Фамилия, преобразованная к верхнему регистру", который выдает список всех фамилий, преобразованных к верхнему регистру.

f. Запрос " Выданные книги", который выдает список книг, которые были выданы к данной дате данным водителем

g. Запрос с расчетами – найти самую дорогую книгу.

p. Запрос с групповой операцией – найти среднюю стоимость проката.

Вариант 18

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить

повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос "Ценные бумаги и клиенты ", который выдает список клиентов, купивших данные ценные бумаги

c. Запрос "Алфавитный список", который выдает список названий в указанном буквенном диапазоне. Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все ценные бумаги с заданным рейтингом.

e. Запрос Название, преобразованная к верхнему регистру", который выдает список всех названий, преобразованных к верхнему регистру.

f. Запрос " Инвестиции ", который выдает список инвестиций, которые были куплены к данной дате данным клиентом

g. Запрос с расчетами – найти самую доходную ценную бумагу.

p. Запрос с групповой операцией – найти среднюю котировку.

Вариант 19

Показатели (Код показателя, Название, Важность, Единица измерения).

Предприятия (Код предприятия, Название, Банковские реквизиты, Телефон,

Контактное лицо).

Динамика показателей (Код показателя, Код предприятия, Дата, Значение).

a. Запрос «Исходные данные» – выдает все данные из всех таблиц, представляя их в удобной для восприятия форме при этом исключить повторение первичных ключевых полей из главных таблиц; (использовать объединения)

b. Запрос "Предприятия и показатели ", который выдает список предприятий, с заданным показателем

c. Запрос "Алфавитный список", который выдает список названий в указанном буквенном диапазоне. Начальную и конечную буквы диапазона выбрать самостоятельно. Отсортировать список в алфавитном порядке.

d. Запрос "Список с условием": все показатели с заданной важностью.

e. Запрос «Название, преобразованная к верхнему регистру", который выдает список всех названий, преобразованных к верхнему регистру.

f. Запрос " Динамика показателей ", который выдает список значений динамики показателей для данного предприятия к данной дате.

g. Запрос с расчетами – найти самый важный показатель.

p. Запрос с групповой операцией – найти среднее значение динамики показателей.

Контрольные вопросы

1. Каким образом создаются запросы в SQL?
2. Для чего используются представления и как они создаются в SQL?
3. Как реализуются объединения в SQL?
4. Как пользоваться встроенными функциями SQL?
5. Пусть имеются таблицы Автор и Книги. Их данные агрегируется в таблице Библиотека (поля связи Код автора, Код книги). Таблица Библиотека содержит столбцы Код записи, Код автора, Код книги, Цена. Написать SQL запрос, выводящий все книги заданного автора.
6. Написать запрос, создающий представление, которое выводит выборку содержимого первых 5 строк таблицы

5. Лабораторная работа №5 Хранимые процедуры

5.1. Цель работы

Изучение механизма динамических запросов с использованием хранимых процедур и функций

5.2. Содержание работы

1. Изучить процесс создания динамических запросов при помощи хранимых процедур
2. Освоить применение пользовательских функций

5.3. Задание к работе

Для базы данных согласно варианту задания (из Лабораторной работы №2) написать:

Вариант 1

- хранимую процедуру «**Печать договора**»
- пользовательскую функцию «**Вычисление страховой суммы**»

Вариант 2

- хранимую процедуру «**Печать бланка поселения**»
- пользовательскую функцию «**Вычисление цены номера**»

Вариант 3

- хранимую процедуру «**Печать бланка сдача в ломбард**»
- пользовательскую функцию «**Вычисление комиссионных**»

Вариант 4

- хранимую процедуру «**Печать бланка сделки**»

- пользовательскую функцию «**Вычисление цены всего товара в сделке**»

Вариант 5

- хранимую процедуру «**Печать бланка Заказа**»
- пользовательскую функцию «**Вычисление цены всего товара в заказе**»

Вариант 6

- хранимую процедуру «**Печать бланка сделки**»
- пользовательскую функцию «**Вычисление комиссионных**»

Вариант 7

- хранимую процедуру «**Печать бланка сделки**»
- пользовательскую функцию «**Вычисление комиссионных**»

Вариант 8

- хранимую процедуру «**Печать бланка поставки**»
- пользовательскую функцию «**Вычисление стоимости поставки**»

Вариант 9

- хранимую процедуру «**Печать бланка нагрузка**»
- пользовательскую функцию «**Вычисление оплаты**»

Вариант 10

- хранимую процедуру «**Печать бланка Учебный план**»
- пользовательскую функцию «**Вычисление средней оценки студента**»

Вариант 11

- хранимую процедуру «**Печать бланка Работы**»
- пользовательскую функцию «**Вычисление зарплаты сотрудника**»

Вариант 12

- хранимую процедуру «**Печать бланка Ремонт**»
- пользовательскую функцию «**Вычисление Стоимости Ремонта**»

Вариант 13

- хранимую процедуру «**Печать бланка Путевки**»
- пользовательскую функцию «**Вычисление величины скидки**»

Вариант 14

- хранимую процедуру «**Печать бланка Проделанная работа**»
- пользовательскую функцию «**Вычисление премии**»

Вариант 15

- хранимую процедуру «**Печать бланка Переговоры**»
- пользовательскую функцию «**Вычисление стоимости переговоров**»

Вариант 16

- хранимую процедуру «**Печать бланка Выданные книги**»
- пользовательскую функцию «**Вычисление стоимости проката**»

Вариант 17

- хранимую процедуру «**Печать бланка Выданные автомобили**»
- пользовательскую функцию «**Вычисление стоимости проката**»

Вариант 18

- хранимую процедуру «**Печать бланка Инвестиции**»
- пользовательскую функцию «**Вычисление котировки**»

Вариант 19

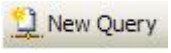
- хранимую процедуру «**Печать бланка Динамика показателей**»
- пользовательскую функцию «**Вычисление значения Динамики показателя**»

5.4. Краткая теория

Хранимая процедура – SQL запрос, который имеет параметры, то есть он выполняется как обычная процедура (мы задаем ее имя и передаем в хранимую процедуру значение параметров.) В зависимости от значения параметров хранимой процедуры мы получаем тот или иной результат запроса. [13]

Замечание. В SQL сервере хранимые процедуры реализуют динамические запросы, выполняемые на стороне сервера.

Рассмотрим создание хранимых процедур при помощи команд языка SQL. Чтобы отобразить хранимые процедуры рабочей БД панели **"Object Explorer"** нужно выделить пункт **"Stored Procedures"**. Чтобы создать новую процедуру при помощи команд языка SQL нужно щелкнуть ЛКМ по кнопке



на панели инструментов. В рабочей области окна сервера появится вкладка **SQLQuery1.sql**, где нужно набрать код новой процедуры, который имеет следующий синтаксис:

CREATE PROCEDURE <Имя процедуры>
[@<Параметр1> <Тип1>[=<Значение1>],
@<Параметр2> <Тип2>[=<Значение2>], …]
[WITH ENCRYPTION]
AS <Команды SQL>

Здесь:

- Имя процедуры - имя создаваемой хранимой процедуры.
- Параметр1, Параметр2, … - параметры передаваемые в процедуру.
- Значение1, Значение2, … - значения параметров по умолчанию.
- Тип1, Тип2, … - типы данных параметров.
- WITH ENCRYPTION - включает шифрование данных.
- Команды SQL - SQL запрос, который выполняется при запуске процедур. [2]

Замечание: После создания процедура помещается в раздел **Stored Procedures** текущей БД на панели **"Object Explorer"**. Если дважды щелкнуть по процедуре ЛКМ, то она откроется для редактирования на вкладке **"SQLQuery"**.

Чтобы посмотреть информацию о хранимой процедуре необходимо выполнить команду:

EXEC SP_HELPTEXT <Имя процедуры>

Хранимые процедуры могут быть запущены следующей командой

EXEC <Имя процедуры> [<Параметр1>, <Параметр2>, …]

Здесь:

- <Имя процедуры> - имя выполняемой процедуры.
- Параметр1, Параметр2, … - значение параметров. [7]

Пример: Создание хранимой процедуры, который выводит имя студентов, у которых средний балл больше заданной величины:

```
CREATE PROCEDURE СрБАЛЛ
@X Real
AS
SELECT *
FROM Студенты
WHERE
(Оценка1+ Оценка2+ Оценка3)/3>@X
```

Команда вызова приведенной выше процедуры выглядит следующим образом:

EXEC СрБАЛЛ 4

Команда выводит всех студентов, у которых средний балл больше 4.

Пользовательские функции

Пользовательские функции очень похожи на хранимые процедуры. Так же в них можно передавать параметры и они выполняют некоторые действия, однако их главным отличием от хранимых процедур является то, что они выводят (возвращают) какой-то результат. Более того, они вызываются только при помощи оператора SELECT, аналогично встроенным функциям. Все пользовательские функции делятся на 2 вида:

1. Скалярные функции – функции, которые возвращают число или текст, то есть одно или несколько значений;

2. Табличные функции – функции, которые выводят результат в виде таблицы. [1]

Для создания новой пользовательской функции используется команда CREATE FUNCTION имеющая следующий синтаксис:

```
CREATE FUNCTION <Имя функции>
([@<Параметр1> <Тип1>[=<Значение1>],
 @<Параметр2> <Тип2>[=<Значение2>], . . .])
RETURNS <Тип>/TABLE
AS
RETURN([<Команды SQL>])
```

Здесь:

- Имя функции – имя создаваемой пользовательской функции.
- Параметр1, Параметр2, .. – параметры передаваемые в функцию.
- Значение1, Значение2, … – значения параметров по умолчанию.
- Тип1, Тип2, . .. – типы данных параметров.

После служебного слова RETURNS в скалярных функциях ставится тип данных результата, который возвращает скалярная функция, либо ставится служебное слово TABLE в табличных функциях. [6]

После служебного слова RETURN ставится SQL команда самой функции.

Замечание: После служебного слова RETURN может быть несколько команд, которые располагаются между словами BEGIN и END. В этом случае служебное слово RETURN не ставится.

Замечание: Тип данных параметра должен совпадать с типом данных выражения, в котором он используется.

Замечание: Если используются несколько SQL команд и BEGIN и END, то перед END нужно ставить команду RETURN <результат функции>.

Пример (скалярная пользовательская функция): Функция для вычисления среднего 3 чисел:

```
CREATE FUNCTION Среднее
(@X1 Int,@X2 Int,@X3 Int)
RETURNS Real
AS
BEGIN
DECLARE @Res Real
SET @Res =(@X1+@X2+@X3)/3
RETURN @Res
```

END

Замечание: Команда DECLARE создает переменную Res для хранения дробных чисел (тип данных **Real**). [4]

Представленная выше пользовательская функция реализована при помощи нескольких команд SQL, но ее можно реализовать при помощи одной функции следующим образом:

```
CREATE FUNCTION Среднее
(@X1 Int,@X2 Int,@X3 Int)
RETURNS Real
AS
RETURN (SELECT (@X1+@X2+@X3)/3)
```

Созданная функция, вычисляющая среднее 6, 3 и 3, запускается следующим образом:

```
SELECT Среднее (6, 3, 3)
```

Результат будет 4.

Пример (табличная пользовательская функция): Из таблицы **Студенты** выводятся поля **ФИО**, **дата рождения** и **столбец возраст**, который вычисляется как разница дат в годах, между датой рождения и текущей датой (параметр CurDate).

```
CREATE FUNCTION Возраст
(@CurDate Date)
RETURNS TABLE
AS
RETURN (SELECT ФИО, [Дата рождения], Возраст = DATEDIFF (yy,[Дата рождения], @CurDate)
FROM Студенты)
```

Данная функция вызывается следующим образом:

```
SELECT * FROM Возраст ('12/17/2007')
```

В результате отобразятся студенты с их возрастом на 17 декабря 2007 года.

Контрольные вопросы

1. Как задается хранимая процедура?
2. Как задается пользовательская функция?
3. Каким образом объявляется переменная в SQL?
4. Каким образом можно возвратить результаты работы процедуры?
5. Каким образом можно возвратить результаты работы функции?

6. Лабораторная работа №6. Целостность данных

6.1. Цель работы

Изучение механизмов обеспечения целостности данных SQL

6.2. Содержание работы

1. Изучить порядок обеспечения целостности данных
2. Изучить порядок создания и работы триггеров.

6.3. Задание к работе

Для базы данных из лабораторной работы 2

1. Добавить ограничение UNIQUE, ограничение CHECK , DEFAULT
2. Создать триггеры индикаторы:

Добавление, выводящий на экран сообщение **"Запись добавлена"** при добавлении новой записи в таблицу

Изменение, выводящий на экран с сообщение **"Запись изменена"** при изменении записи в таблице

Удаление, выводящий на экран с сообщение **"Запись удалена"** при удалении записи из таблицы

3. Создать триггер по контролю целостности данных при удалении, когда данные удаляются сразу из двух связанных таблиц.
4. Проверить и продемонстрировать работоспособность созданных триггеров.

6.4. Краткая теория

При работе БД должна обеспечиваться целостность данных. Под целостностью данных понимают обеспечения целостности связей между записями в таблицах при удалении записей из первичных таблиц. То есть, при удалении записей из первичных таблиц автоматически должны удаляться связанные с ними записи из вторичных таблиц.

В SQL Server существуют следующие механизмы обеспечения целостности данных в таблице: Механизмы декларативной целостности и Механизмы процедурной целостности. [10]

Механизмы декларативной целостности

- *NULL / NOT NULL* ограничение – задаётся на уровне какого-то столбца и определяет, может ли хранится значение NULL в колонке.
- *UNIQUE* ограничение – позволяет обеспечить уникальность значений в одном или нескольких столбцах.
- *PRIMARY KEY* ограничение – практически тоже самое, что и UNIQUE ограничение, но в отличие от него, PRIMARY KEY не позволяет хранить NULL.

- *CHECK* ограничение – позволяет задать некое логическое условие, которое должно быть истинным (TRUE) при вставке или обновлении данных в таблице. Может быть задано как на уровне одного столбца, так и на уровне таблицы.
- *RULE* – создает объект, называемый правилом. Будучи привязанным к столбцу, имеющему псевдоним типа данных, правило определяет *значения, которые могут быть вставлены в этот столбец.*
- *FOREIGN KEY* ограничение – позволяет обеспечить ссылочную связность двух таблиц. При вставке значения в колонку (или колонки) с FOREIGN KEY ограничением, будет производится проверка на наличие такого же значения в таблице, на которую указывает FOREIGN KEY. Если значения нет, то обновление или вставка строки завершается с ошибкой. Исключением может быть только значение NULL, если на колонке не задано ограничение NOT NULL. Кроме того, ссылаться можно только на колонку с уникальными значениями, т.е. с UNIQUE или PRIMARY KEY ограничением. Так же можно задать поведение, на случай обновления или удаления строки, в «отцовской» таблице:
  - *NO ACTION* – отцовскую таблицу запрещено менять
  - *CASCADE* – подчинённые строки будут обновлены или удалены, в зависимости от выполняемого действием над отцовской таблицей
  - *SET NULL* – значение в подчинённой таблице будет установлено в NULL
  - *SET DEFAULT* – значение в подчинённой таблице будет установлено в значение по умолчанию. [2]

Добавление нового CHECK CONSTRAINT

```
ALTER TABLE table_name
    [ WITH { CHECK | NOCHECK } ]
        ADD CONSTRAINT constraint_name CHECK
            [ NOT FOR REPLICATION ]
                ( logical_expression )
```

Опциональные секции:

1. *[WITH { CHECK | NOCHECK }]* – в случае отсутствия применяется значение WITH CHECK
2. *[NOT FOR REPLICATION]* – если конструкция указана, то ограничение не проверяется при вставке или обновлении данных в момент репликации; если конструкция пропущена –ограничение проверяется.

Примеры команд будут приведены для простейшей таблицы Employees, которая выглядит следующим образом:

Id	Name	Age
1	John	35

```
--> Check Existing Data: "Yes" (Default: WITH CHECK)
--> Enforce For Replication: "Yes" ( Default: ommited )
ALTER TABLE dbo.Employees
        ADD CONSTRAINT CK_Age_1 CHECK       --> Name: "CK_Age_1"
                (Age < 100)                 --> Expression: Age < 100


ALTER TABLE dbo.Employees
    WITH NOCHECK                            --> Check Existing Data: "No"
        ADD CONSTRAINT CK_Age_2 CHECK       --> Name: "CK_Age_2"
            NOT FOR REPLICATION             --> Enforce For Replication: "No"
                (Age >= 0)                  --> Expression: Age >= 0
```

Изменение существующего CHECK CONSTRAINT

Для обновления существующего проверочного ограничения используется конструкция ALTER TABLE. Для изменения доступны только следующие свойства:

- Check Existing Data On Creation Or Re-Enabling
- Enforce For INSERTs And UPDATEs

```
ALTER TABLE table_name
    [ WITH { CHECK | NOCHECK } ]
        { CHECK | NOCHECK }
            CONSTRAINT { ALL | constraint_name [ ,...n ] }
```

Опциональные секции:

1. *[WITH { CHECK | NOCHECK }]* – в случае отсутствия применяется значение WITH NOCHECK
2. *[, ...n]* – позволяет задать имя более чем одного ограничения, к которым будут применены изменения; использование слова ALL изменения применятся ко всем проверочным ограничениям на таблице. [1]

```
-------------------------------------------------------------------------------
-- Enable CK_Age_1 (existing data validation is not performed)
ALTER TABLE dbo.Employees
                                    --> Check Existing Data: "No"
        CHECK                       --> Enforce for INSERTs And UPDATEs: "Yes"
            CONSTRAINT CK_Age_1
-------------------------------------------------------------------------------
-- Enable with explicit validation on existing data
ALTER TABLE dbo.Employees
     WITH CHECK                     --> Check Existing Data: "Yes"
        CHECK                       --> Enforce for INSERTs And UPDATEs: "Yes"
            CONSTRAINT CK_Age_1, CK_Age_2
-------------------------------------------------------------------------------
-- Disable all check constraints
ALTER TABLE dbo.Employees
        NOCHECK                     --> Enforce for INSERTs And UPDATEs: "No"
            CONSTRAINT ALL
-------------------------------------------------------------------------------
```

Удаление существующего CHECK CONSTRAINT

Команда очень проста и не требует дополнительных объяснений. Ещё шаблон:

```
ALTER TABLE table_name
    DROP [ CONSTRAINT ] constraint_name
```

Ограничение по умолчанию (DEFAULT)

Дополнительным механизмом использования значений по умолчанию являются объекты базы данных, созданные командой:

CREATE DEFAULT имя_умолчания AS константа

Умолчание связывается с тем или иным столбцом какой-либо таблицы с помощью процедуры:

sp_bindefault [@defname=] 'default',
[@objname=] 'object_name'
[,[@futureonly=] 'futureonly_flag'],
где
'object_name'
может быть представлен как
'имя_таблицы.имя_столбца'

Удаление ограничения по умолчанию выполняется командой

DROP DEFAULT {имя_умолчания} [,...n]

если предварительно это ограничение было удалено из всех таблиц процедурой

sp_unbindefault [@objname=] 'object_name'
[,[@futureonly=] 'futureonly_flag'] [1]

Примеры

Пусть создана таблица без ограничений:

CREATE TABLE Товар
(КодТовара INT, Название VARCHAR(20), Тип VARCHAR(20),
Дата DATETIME, Цена MONEY, Остаток INT)

Рассмотрим примеры внесения в таблицу всевозможных ограничений.

Пример Поле КодТовара необходимо сделать первичным ключом. Выполнение следующей команды будет отвергнуто, поскольку поле КодТовара допускает внесение значений NULL.

ALTER TABLE Товар ADD CONSTRAINT pk1
PRIMARY KEY(КодТовара)

Сначала нужно изменить объявление столбца КодТовара, запретив внесение значений NULL:

ALTER TABLE Товар
ALTER COLUMN КодТовара INT NOT NULL

И только потом создать ограничение первичного ключа:

```
ALTER TABLE Товар ADD CONSTRAINT pk1
PRIMARY KEY(КодТовара)
```

Пример Изменить столбец, добавив ограничение NOT NULL.

```
ALTER TABLE Товар ALTER COLUMN
   Название VARCHAR(40) NOT NULL
```

Пример Добавить ограничение уникальности значения.

```
ALTER TABLE Товар ADD CONSTRAINT
   u1 UNIQUE(Название)
```

Пример Создать умолчание и добавить умолчание столбцу.

```
CREATE DEFAULT df1 AS 0
sp_bindefault 'df1', 'Товар.Остаток'
CREATE DEFAULT df2 AS GETDATE()
sp_bindefault 'df2', 'Товар.Дата'
```

Пример. Создать правило и добавить правило столбцу.

```
CREATE RULE r1 AS @m IN
   ('мебель','бытовая химия','косметика')
sp_bindrule 'r1','Товар.Тип'
```

Механизмы процедурной целостности

Для поддержания согласованности данных могут использоваться декларативные и процедурные механизмы. Распространённым методом обеспечения процедурной целостности являются триггеры.

Диаграммы – это компоненты БД, которые блокируют удаление записей из первичных таблиц если существуют связанные с ними записи во вторичных таблицах. Следовательно, диаграммы предотвращают нарушение целостности данных. В SQL Server диаграммы создаются при помощи мастера диаграмм.

Триггеры – это аналог процедур обработчиков событий. То есть они выполняют команды SQL если происходят какие-либо действия с таблицей (Например: добавление, изменение или удаление записей). При помощи триггеров можно организовать автоматическое удаление записей из вторичной таблицы при удалении связанной с ними записи из первичной таблицы. [1]

Рассмотрим создание триггеров при помощи языка SQL.

Создание триггеров

Триггеры представляют собой особый вид хранимых процедур, привязанных к таблицам и представлениям, которые вызываются в результате исполнения операторов INSERT, UPDATE или DELETE языка Transact-SQL. Сами триггеры создаются с помощью Transact-SQL.

Один и тот же триггер способен реагировать как только на одно событие INSERT, UPDATE или DELETE, так и одновременно на несколько.

Триггер не может быть установлен для событий временной или системной таблицы. Кроме того, триггеры INSTEAD OF DELETE и INSTEAD OF UPDATE нельзя определять в таблицах, в которых определены ограничения каскадной ссылочной целостности ON DELETE и/или ON UPDATE.

Для обнаружения модификации данных в триггерах используются две псевдотаблицы: Inserted и Deleted. При этом в момент срабатывания триггера новые или изменённые строки попадают в таблицу Inserted, а удалённые в таблицу Deleted. В теле триггера на языке Transact-SQL выполняются запросы к этим таблицам для анализа данных и принятия решения. Таблицы Inserted и Deleted имеют точно такую же структуру как и таблица/представление, к которым привязан триггер, и хранятся в памяти. [13]

Существует два типа триггеров:

INSTEAD OF – выполняется вместо вызвавшей операции (заменяет операцию);

AFTER – выполняется после события, вызвавшего срабатывание (этот класс используется по умолчанию).

Допускается привязывать к таблице триггеры сразу обоих классов. При этом первым срабатывает INSTEAD OF, а вторым – AFTER.

Глубина при вложенных и рекурсивных вызовах триггеров не может превышать 32. [8]

Создание триггера осуществляется с помощью команды CREATE TRIGGER:

Для создания триггера на вкладке нового запроса необходимо набрать команду CREATE TRIGGER, имеющую следующий синтаксис:

CREATE TRIGGER <Имя триггера>
ON <Имя таблицы>
FOR <INSERT|UPDATE|DELETE>
[WITH ENCRYPTION]
AS <Команды SQL>

Здесь:

- Имя триггера – это имя создаваемого триггера.
- Имя таблицы – имя таблицы, для которой создаётся триггер.
- Если используется параметр AFTER, то триггер выполняется после события, а если параметр INSTEAD OF, то выполняется вместо события.
- Параметры INSERT, UPDATE и DELETE определяют событие, при котором (или вместо которого) выполняется триггер.
- Параметр WITH ENCRYPTION – предназначен для включения шифрования данных при выполнении триггера.

- Команды SQL – это SQL команды, выполняемые при активизации триггера. [4]

Рассмотрим примеры создания различных триггеров для таблицы **"Студенты"**.

Пример: Создаёт триггер **"Добавление"**, выводящий на экран сообщение **"Запись добавлена"** при добавлении новой записи в таблицу **"Студенты"**

```
CREATE TRIGGER Добавление
ON Студенты
FOR AFTER INSERT
AS PRINT 'Запись добавлена'
```

Пример: Создаёт триггер **"Изменение"**, выводящий на экран с сообщение **"Запись изменена"** при изменении записи в таблице **"Студенты"**

```
CREATE TRIGGER Изменение
ON Студенты
FOR AFTER UPDATE
AS PRINT 'Запись изменена'
```

Пример: Создаёт триггер **"Удаление"**, выводящий на экран с сообщение **"Запись удалена"** при удалении записи из таблицы **"Студенты"**

```
CREATE TRIGGER Удаление
ON Студенты
FOR AFTER DELETE
AS PRINT 'Запись удалена'
```

Пример: В данном примере вместо удаления студента из таблицы **"Студенты"** выполняется код между BEGIN и END. Он состоит из двух команд DELETE. Первая команда удаляет все записи из таблицы **"Оценки"**, которые связаны с записями из таблицы **"Студенты"**. То есть у которых **Оценки.[Код студента]** равен коду удаляемого студента. Затем из таблицы **"Студенты"** удаляется сам студент.

```
CREATE TRIGGER УдалениеСтудента
ON Студенты
INSTEAD OF DELETE
AS
BEGIN
DELETE Оценки
FROM deleted
WHERE deleted.[Код студента]=Оценки.[Код студента]
DELETE Студенты
FROM deleted
WHERE deleted.[Код студента]=Студенты.[Код студента]
```

END

Замечание: Здесь удаляемая запись обозначается служебным словом deleted. [4]

Замечание: Для обеспечения целостности данных триггеры используют обычно вместе с диаграммами, но мы можем применять такие триггеры и без диаграмм, однако мы не можем применять диаграммы без триггеров.

Контрольные вопросы

1. Какие механизмы обеспечения декларативной целостности данных существуют?
2. Как добавить ограничение?
3. Какие механизмы обеспечения процедурной целостности данных существуют?
4. Что такое триггер и для чего его необходимо использовать?
5. К каким событиям может быть применен триггер?
6. Для чего необходимы диаграммы?

7. Лабораторная работа №7 Безопасность и обслуживание баз данных

7.1. Цель работы

Освоить методы переноса и генерации скриптов базы данных

7.2. Содержание работы

- Механизм переноса базы данных
- Резервное копирование
- Генерация скриптов базы данных

7.3. Задание к работе

Для Базы данных из лабораторной работы 2 реализовать перенос базы данных через механизм резервного копирования, сгенерировать скрипт создания базы данных.

7.4. Краткая теория

Перенос файла БД 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 нам потребуется скопировать два файла - саму базу данных BDTur_firmSQL.mdf и файл отчетов о транзакциях BDTur_firmSQL.ldf. Однако непосредственное копирование данных файлов невозможно, так как данные файлы используются сервером баз данных. Для того чтобы сделать файлы доступными для копирования, базу данных необходимо отсоединить от сервера (рис. 20).

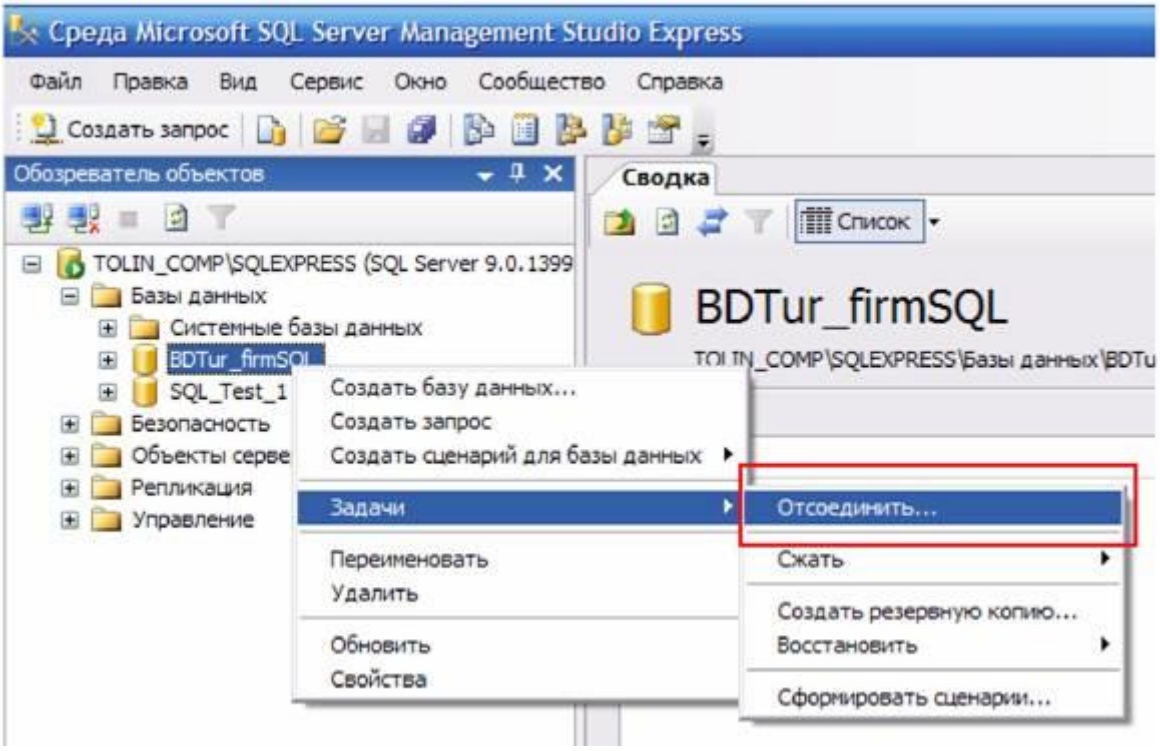


Рис. 20. Отсоединение от сервера

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

Для присоединения базы данных на другом компьютере запускаем SQL Management Studio, выделяем ветку «Базы данных» и в контекстном меню выбираем «Присоединить» (рис.21).

В появившемся окне указываем расположение файла базы данных BDTur_firmSQL.mdf – файл отчетов присоединится автоматически - и нажимаем «ОК». Присоединившаяся база данных немедленно отображается в папке «Базы данных». Следует отметить, что после присоединения БД может потребоваться настройка пользователей БД и прав доступа. [6]

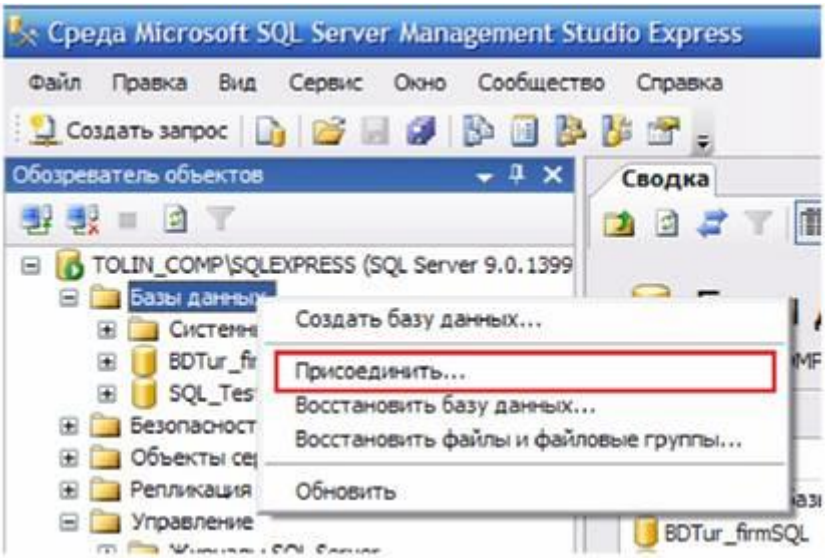


Рис. 21. Присоединение базы данных

Перенос БД с одного сервера БД на другой через создание резервной копии

Создание резервной копии БД

1. В контекстном меню вашей БД выбрать Tasks/ Back Up

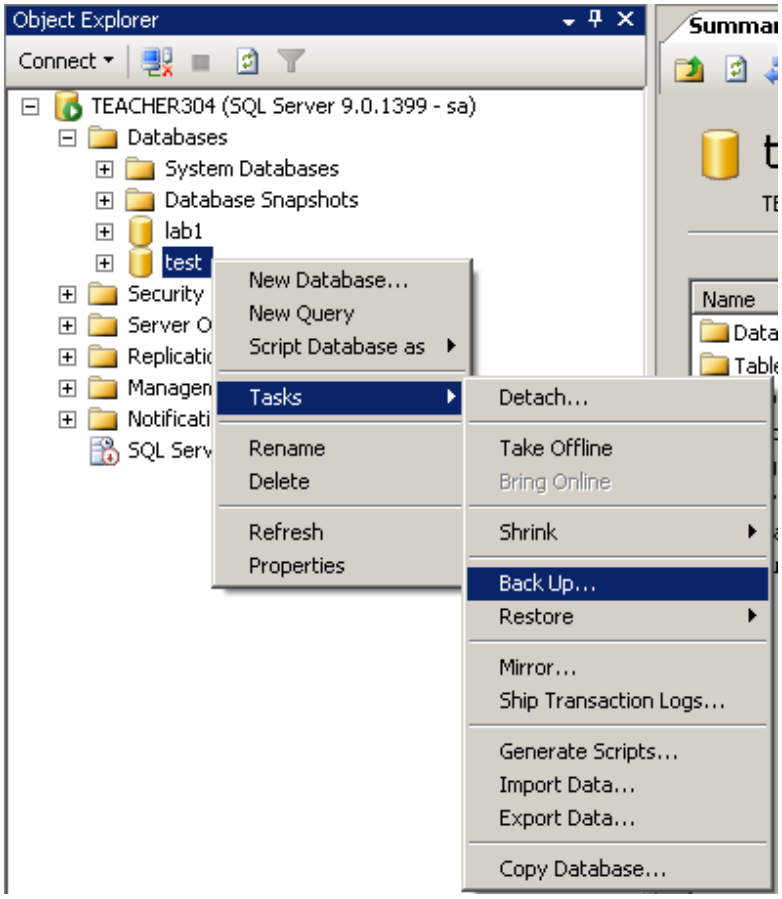


Рис. 22. Back up

2. В появившемся окне **Back Up Database** в разделе **Destination** кнопкой Remove удалить все пути к резервной копии.

3. В окне **Back Up Database** Кнопкой **Add** вызвать проводник, в котором указать путь и название файла, в который будет помещена резервная копия.

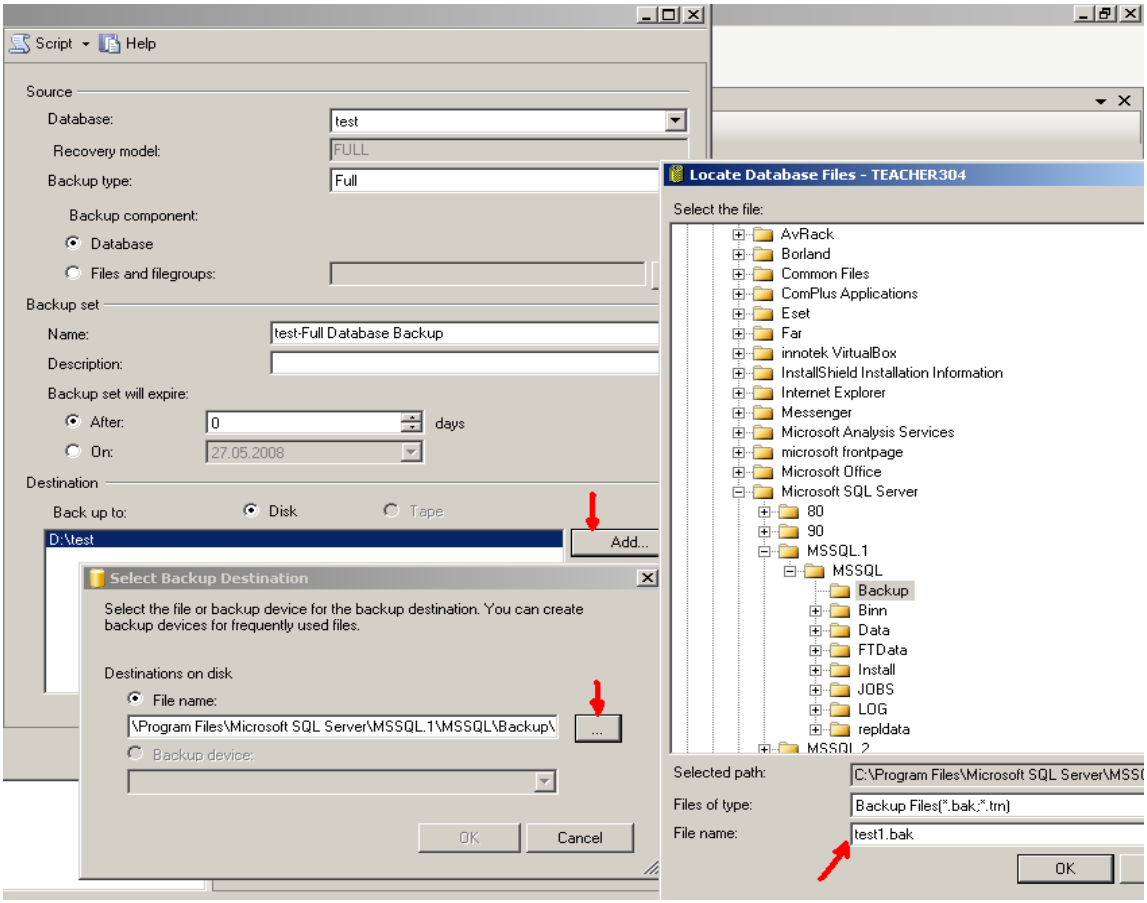


Рис. 23. Сохранение резервной копии -1

4. После выбора пути и файла для резервной копии в окне **Back Up Database** нажатием на **OK** запускаем процесс создания резервной копии. В случае успешной работы появится сообщение:

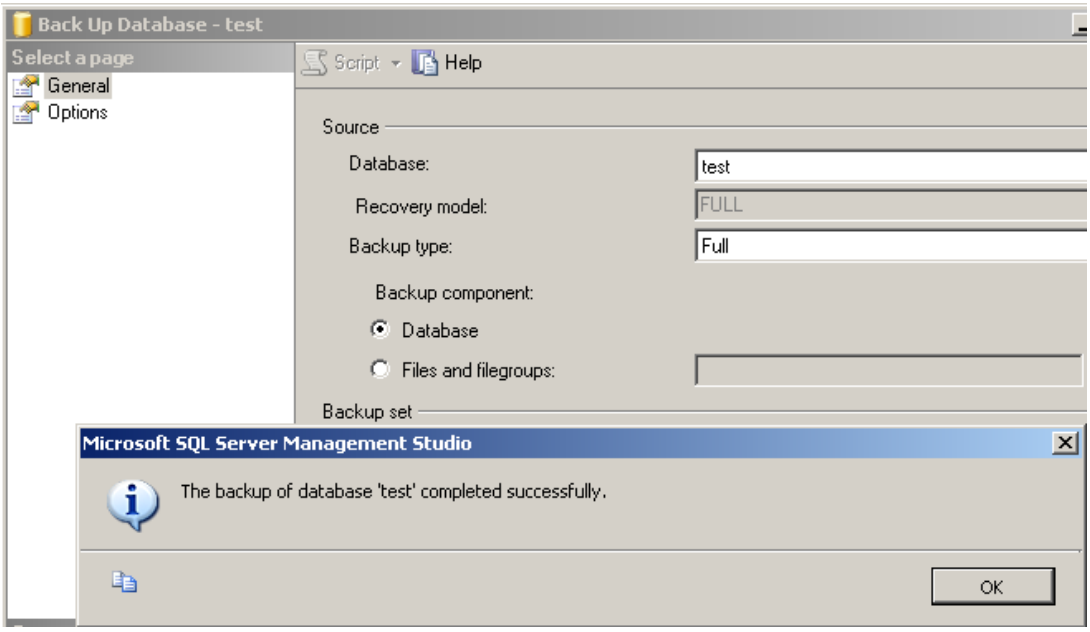


Рис. 24. Сохранение резервной копии -2

Восстановление из резервной копии БД

1. В разделе **Databases** в контекстном меню выбираем **Restore Database…**

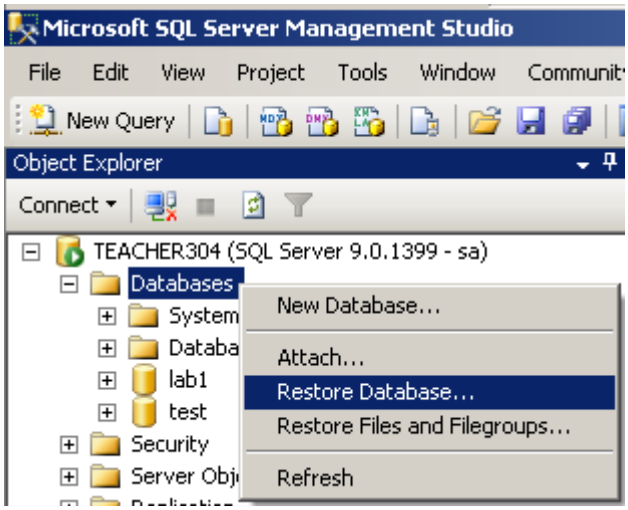


Рис. 25. Восстановление БД

2. В окне **Restore Database** в разделе **To Database** написать имя новой БД, в которую будет помещен результат, способ восстановления выбирается **From Device**, в следующем окне **Specify Backup** кнопкой **Add** вызываем проводник, в котором указываем путь и имя файла с резервной копией.

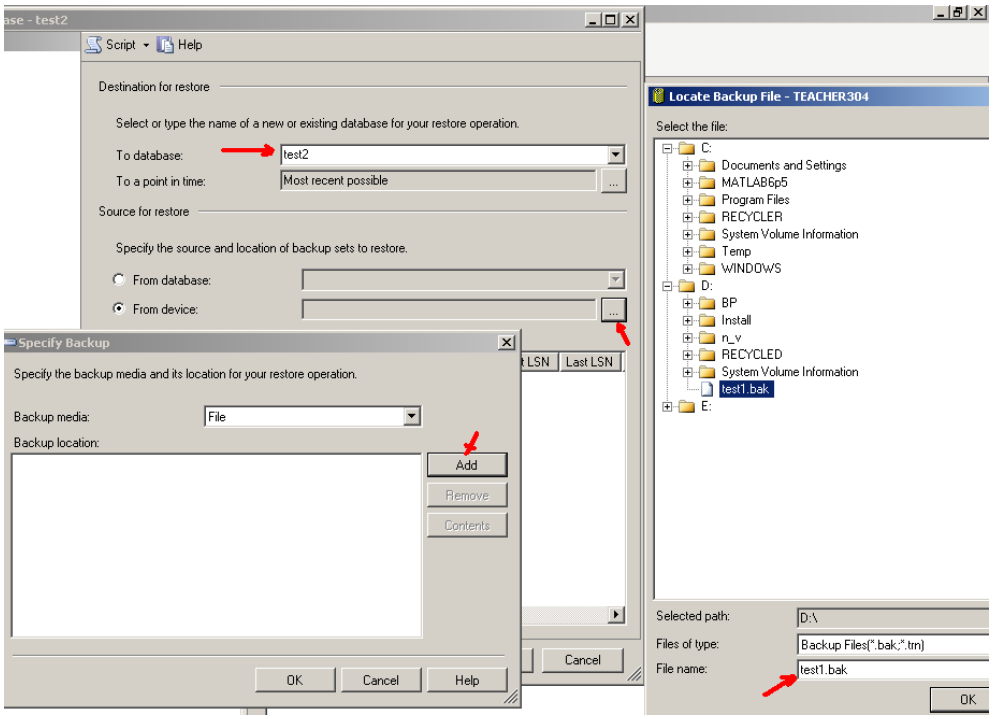


Рис. 26. Восстановление БД – имя новой БД

3. После возврата в окно **Restore Database** в списке **Select the backup sets to restore** отмечаем выбранную резервную копию.

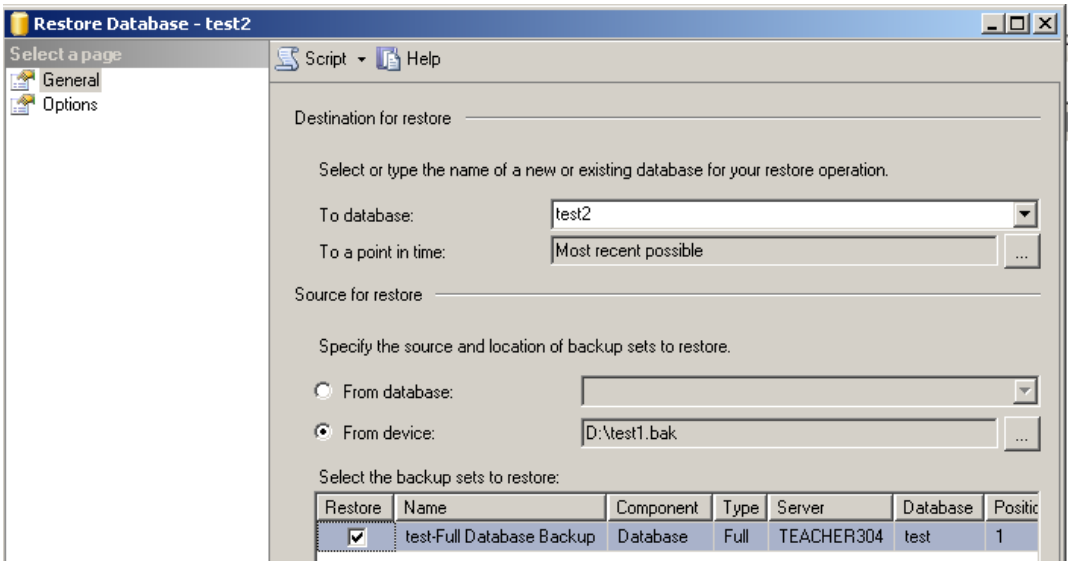


Рис. 27. Восстановление БД – выбор резервной копии

5. Запускаем процесс восстановления. В случае успешного выполнения получим:

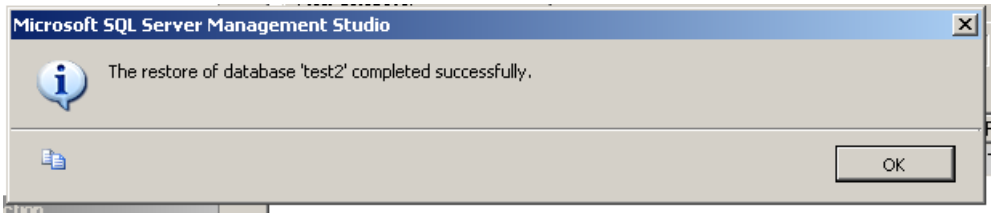


Рис. 28. Восстановление БД – завершение

Создание скрипта для копирования базы данных

Это краткое описание того, как с генерировать скрипт создания схемы базы (включая хранимые процедуры, функции, триггеры, зависимости и проекции). Для этого понадобится SQL Server Management Studio, которая входит в состав поставки базы.

Выбираем *Tasks->Generate Scripts* из контекстного меню базы.

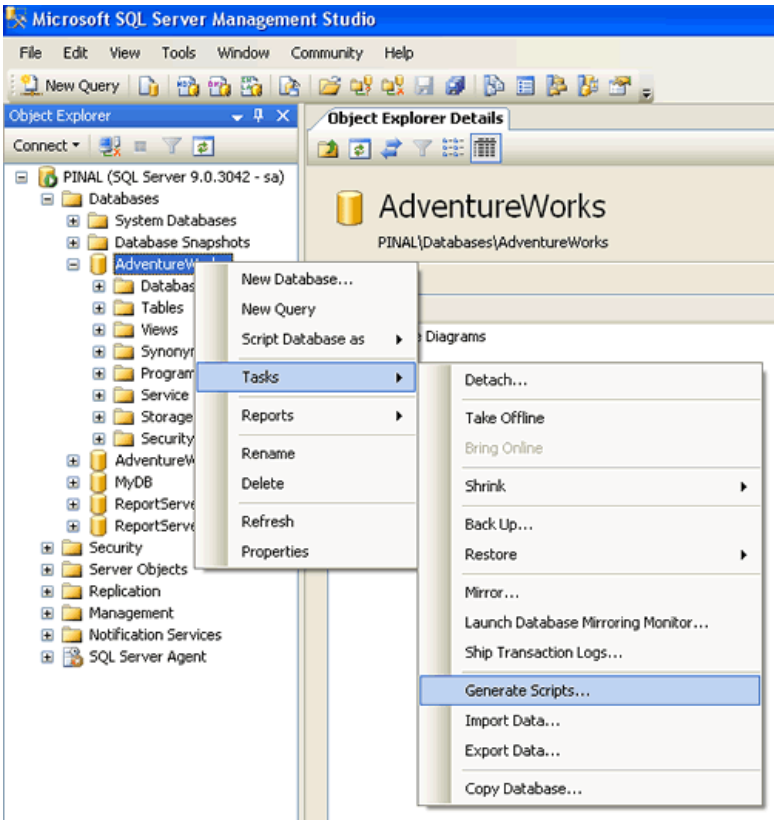


Рис. 29. Генерация скрипта БД

Выбираем базу, которой нужно сделать генерацию. Галочка внизу обозначает, что мы хотим экспортировать все объекты, включая триггеры, хранимые процедуры и т.д.

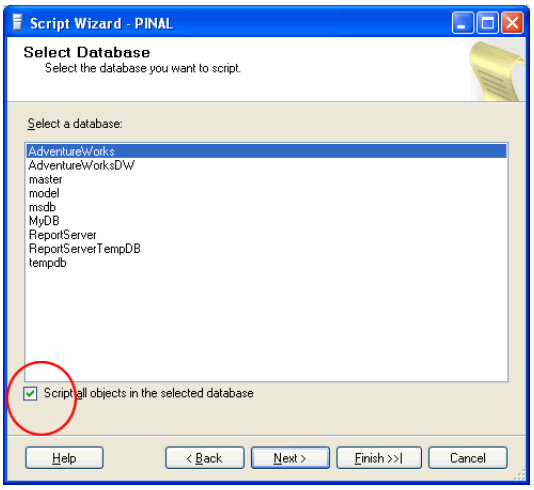


Рис. 30 Выбор базы для генерации скрипта

Далее - Выбор опций базы

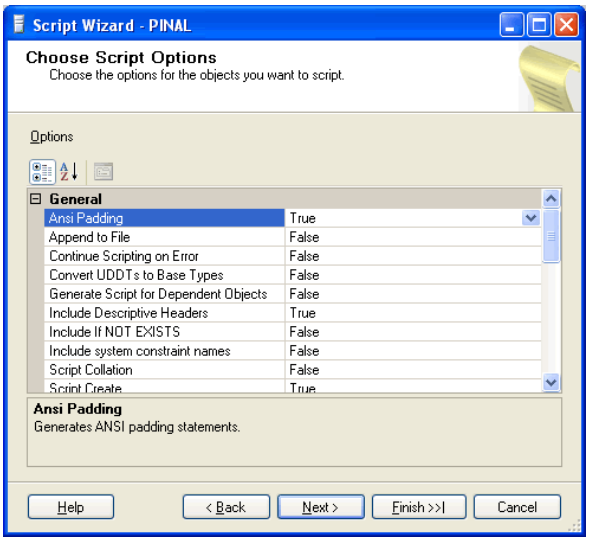


Рис. 31. Выбор опций базы

Выбираем просто в новое окно запроса.

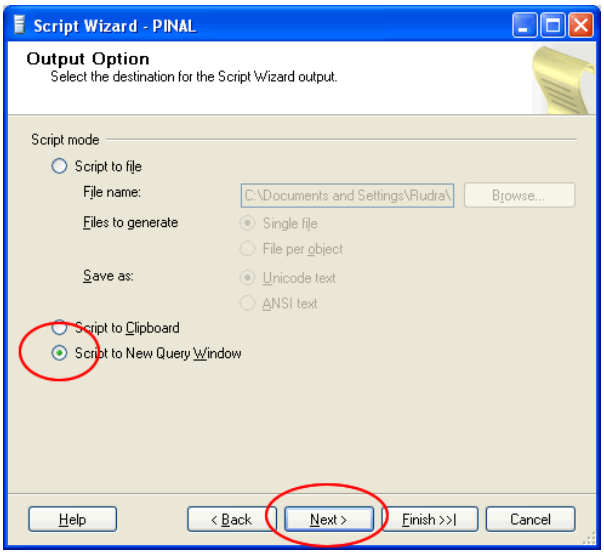


Рис. 32. Выбор места для записи результата

Нажимаем *Finish* для начала генерации.

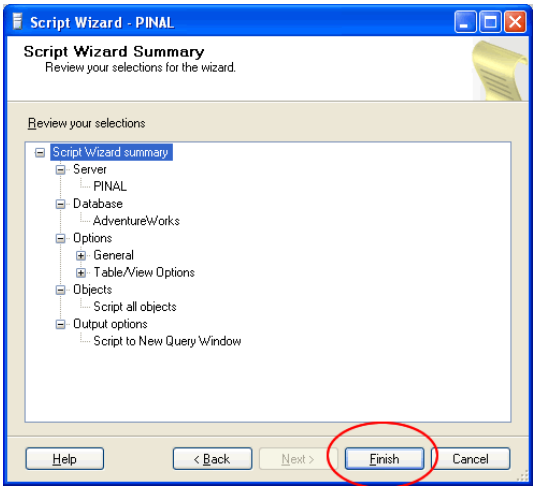


Рис. 33. Начало генерации

Процесс создания скрипта

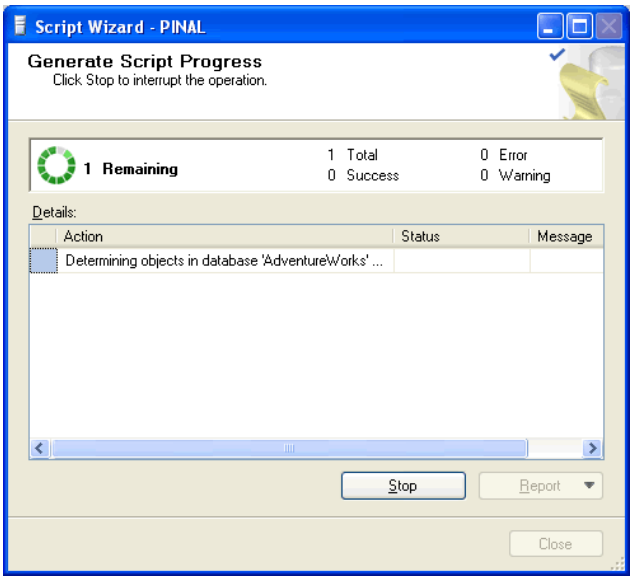


Рис. 34. Процесс создания скрипта

Примерно так будет выглядеть завершительная ступень генерации.

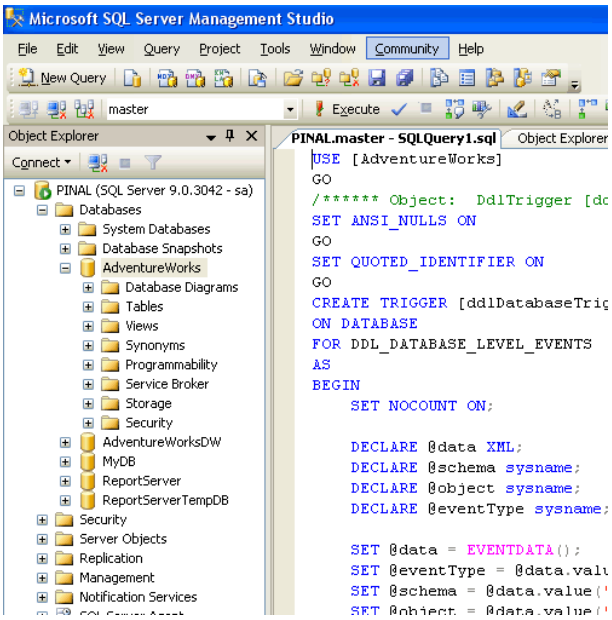


Рис. 35. Скрипт, открытый в новом окне

Контрольные вопросы

1. Какие действия необходимо выполнить при переносе БД?
2. Как создавать резервные копии БД?
3. Как можно развернуть БД из резервной копии?
4. Как возможно сгенерировать скрипт уже созданной БД?
5. Каким образом возможно сгенерировать скрипт для пустой БД, для ее содержимого, триггеров и хранимых процедур?

8. Лабораторная работа №8. Управление ролями и разрешениями в ms sql Server

8.1. Цель работы

Приобрести знания и сформировать навыки в применении инструкции GRANT для предоставления прав доступа к объектам баз данных; в применении инструкции REVOKE для отмены прав доступа к объектам баз данных и выполнение хранимых процедур баз данных; в использовании инструкции DENY для запрета предоставления прав доступа к объектам баз данных.

8.2. Содержание работы

- Создание ролей и пользователей средствами Management Tool
- Создание ролей и пользователей средствами запросов SQL

8.3. Задание к лабораторной работе

По индивидуальному варианту базы данных, которая выполнена в 1 лабораторной работе, определить 2-3 должностных лица, которые могут работать с таблицами БД. Для каждого должностного лица определить набор привилегий, которыми он может пользоваться.

В утилите **SQL Server Management Studio и программно через запросы SQL** создать под каждое должностное лицо соответствующую роль, наделить эту роль определенными привилегиями. Далее создать по одному пользователю на каждую должность и присвоить им соответствующие роли.

8.4. Краткая теория

Добавление пользователей базы данных

Пользователь может войти в систему баз данных, используя учетную запись пользователя Windows или регистрационное имя входа в SQL Server. Для последующего доступа и работы с определенной базой данных пользователь также должен иметь учетную запись пользователя базы данных. Для работы с каждой отдельной базой данных требуется иметь учетную запись пользователя именно для этой базы данных. Учетную запись пользователя базы данных можно сопоставить с существующей учетной записью пользователя Windows, группой Windows (в которой пользователь имеет членство), регистрационным именем или ролью.

Управлять пользователями баз данных можно с помощью среды Management Studio или инструкций языка Transact-SQL. Оба эти способа рассматриваются в следующих подразделах. [4]

Управление пользователями базы данных с помощью среды Management Studio

Чтобы добавить пользователя базы данных с помощью среды Management Studio, разверните узел сервера в окне Object Explorer и в нем папку "Databases", в этой папке разверните узел требуемой базы данных, а в ней папку "Security". Щелкните правой кнопкой мыши папку "Users" и в контекстном меню выберите пункт New User. Откроется диалоговое окно Database User - New, в котором следует ввести имя пользователя User name и выбрать соответствующее регистрационное имя Login name:

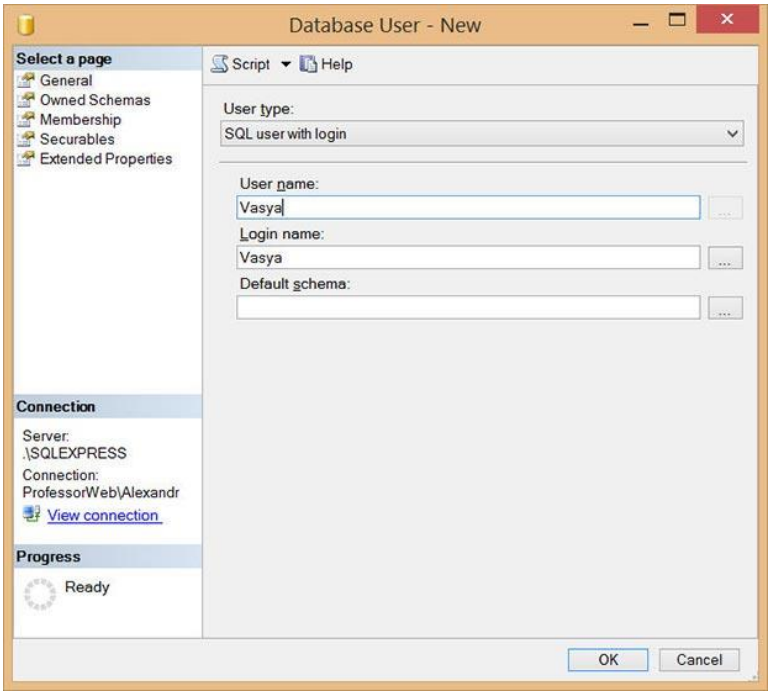


Рис. 36. Добавление пользователя

Управление безопасностью базы данных посредством инструкций языка Transact-SQL

Для добавления пользователя в текущую базу данных используется **инструкция CREATE USER**. Синтаксис этой инструкции выглядит таким образом:

```
CREATE USER user_name
  [FOR {LOGIN login |CERTIFICATE cert_name | ASYMMETRIC KEY key_name}]
  [WITH DEFAULT_SCHEMA = schema_name]
```

Параметр user_name определяет имя, по которому пользователь идентифицируется в базе данных, а в параметре login указывается

регистрационное имя, для которого создается данный пользователь. В параметрах cert_name и key_name указываются соответствующий сертификат и асимметричный ключ соответственно. Наконец, в параметре **WITH DEFAULT_SCHEMA** указывается первая схема, с которой сервер базы данных будет начинать поиск для разрешения имен объектов для данного пользователя базы данных.

С помощью инструкции **ALTER USER** можно изменить имя пользователя базы данных, изменить схему пользователя по умолчанию или переопределить пользователя с другим регистрационным именем. Подобно инструкции CREATE USER, пользователю можно присвоить схему по умолчанию прежде, чем она создана.

Для удаления пользователя из текущей базы данных применяется инструкция **DROP USER**. Пользователя, который является владельцем защищаемых объектов (объектов базы данных), удалить нельзя. [4]

Роли

Когда нескольким пользователям требуется выполнять подобные действия в определенной базе данных и при этом они не являются членами соответствующей группы Windows, то можно воспользоваться ролью базы данных, задающей группу пользователей базы данных, которые могут иметь доступ к одним и тем же объектам базы данных.

Членами роли базы данных могут быть любые из следующих:

- группы и учетные записи Windows;
- регистрационные имена входа в SQL Server;
- другие роли.

Архитектура безопасности компонента Database Engine включает несколько "системных" ролей, которые имеют специальные явные разрешения. Кроме ролей, определяемых пользователями, существует два типа предопределенных ролей: фиксированные серверные роли и фиксированные роли базы данных. [4]

Фиксированные серверные роли

Фиксированные серверные роли определяются на уровне сервера и поэтому находятся вне баз данных, принадлежащих серверу баз данных. В таблице 3. ниже приводится список фиксированных серверных ролей и краткое описание действий, которые могут выполнять члены этих ролей:

Таблица 3

Серверные роли

Фиксированная серверная роль	**Описание**
sysadmin	Выполняет любые действия в системе баз данных
serveradmin	Конфигурирует параметры сервера
setupadmin	Устанавливает репликацию и управляет расширенными процедурами

securityadmin	Управляет регистрационными именами и разрешениями для инструкции CREATE DATABASE и чтением журналов логов
processadmin	Управляет системными процессами
dbcreator	Создает и модифицирует базы данных
diskadmin	Управляет файлами на диске

Членов фиксированной серверной роли можно добавлять и удалять двумя способами:

- используя среду Management Studio;
- используя инструкции языка Transact-SQL.

Чтобы добавить регистрационное имя в члены фиксированной серверной роли посредством среды Management Studio, разверните в обозревателе объектов узел сервера, в нем папку "Security", а в ней разверните папку "Logins". Выберите имя пользователя, для которого нужно изменить роль, щелкните правой кнопкой мыши и выберите в контекстном меню Properties. В открывшемся окне Login Properties перейдите на вкладку Server Role, где можно задавать или удалять пользователя в фиксированной роли:

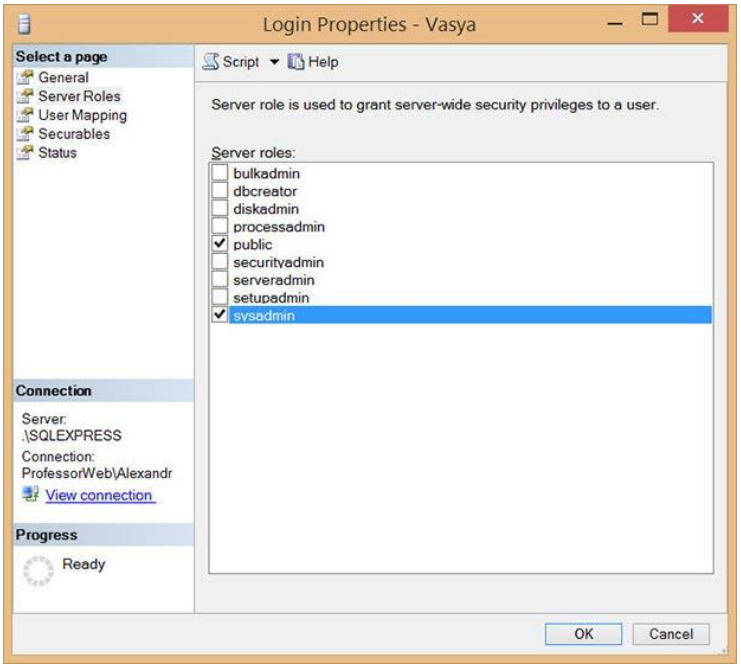


Рис. 37. Свойства серверной роли

Для добавления и удаления членов в фиксированные серверные роли используются инструкции языка Transact-SQL **CREATE SERVER ROLE** и **DROP SERVER ROLE** соответственно. А для изменения членства в серверной роли используется инструкция **ALTER SERVER ROLE**.

Фиксированные серверные роли нельзя добавлять, удалять или переименовывать. Кроме этого, только члены фиксированных серверных ролей могут выполнять системные процедуры для добавления или удаления регистрационного имени в роли. [9]

Фиксированные роли базы данных

Фиксированные роли базы данных определяются на уровне базы данных и поэтому существуют в каждой базе данных, принадлежащей серверу баз данных. В таблице 4 ниже приводится список фиксированных ролей базы данных и краткое описание действий, которые могут выполнять члены этих ролей.

Таблица 4

Роли БД

Фиксированные роли базы данных	
Фиксированная роль базы данных	**Описание**
db_owner	Пользователи, которые могут выполнять почти все действия в базе данных
db_accessadmin	Пользователи, которые могут добавлять и удалять пользователей
db_datareader	Пользователи, которые могут просматривать данные во всех таблицах пользователей базы данных
db_datawriter	Пользователи, которые могут добавлять, изменять или удалять данные во всех пользовательских таблицах базы данных
db_ddladmin	Пользователи, которые могут выполнять инструкции DDL в базе данных
db_securityadmin	Пользователи, которые могут управлять всеми действиями в базе данных, связанными разрешениями безопасности
db_backupoperator	Пользователи, которые могут выполнять резервное копирование базы данных
db_denydatareader	Пользователи, которые не могут просматривать любые данные в базе данных
db_denydatawriter	Пользователи, которые не могут изменять никакие данные в базе данных

Фиксированная роль базы данных public

Кроме перечисленных в таблице фиксированных ролей базы данных, существует специальная фиксированная роль базы данных public. Фиксированная роль базы данных public является специальной ролью, членом которой являются все законные пользователи базы данных. Она

охватывает все разрешения по умолчанию для пользователей базы данных. Это позволяет предоставить всем пользователям, которые не имеют должных разрешений, набор разрешений (обычно ограниченный). Роль public предоставляет все разрешения по умолчанию для пользователей базы данных и не может быть удалена. Пользователям, группам или ролям нельзя присвоить членство в этой роли, поскольку они имеют его по умолчанию.

По умолчанию роль public разрешает пользователям выполнять следующие действия:

- просматривать системные таблицы и отображать информацию из системной базы данных master, используя определенные системные процедуры;
- выполнять инструкции, для которых не требуются разрешения, например, PRINT.

Присвоения пользователю членства в фиксированной роли базы данных

Чтобы присвоить пользователю базы данных членство в фиксированной роли базы данных с помощью среды Management Studio, разверните сервер и папку "Databases", а в ней базу данных, затем разверните папку "Security", "Roles" и папку "Databases Roles". Щелкните правой кнопкой мыши роль, в которую требуется добавить пользователя, и в контекстном меню выберите пункт Properties. В диалоговом окне свойств роли базы данных нажмите кнопку Add и выберите пользователей, которым нужно присвоить членство в этой роли. Теперь этот пользователь является членом данной роли базы данных и наследует все параметры доступа, предоставленные этой роли. [13]

Управление определяемыми пользователем ролями базы данных с помощью среды Management Studio

Чтобы создать определяемую пользователем роль базы данных с помощью среды Management Studio, разверните узел сервера, папку "Databases", требуемую базу данных, папку "Security". Щелкните правой кнопкой папку "Roles", в появившемся контекстном меню выберите пункт New, а во вложенном меню выберите пункт New Database Role. В открывшемся диалоговом окне Database Role - New введите в соответствующее поле имя новой роли:

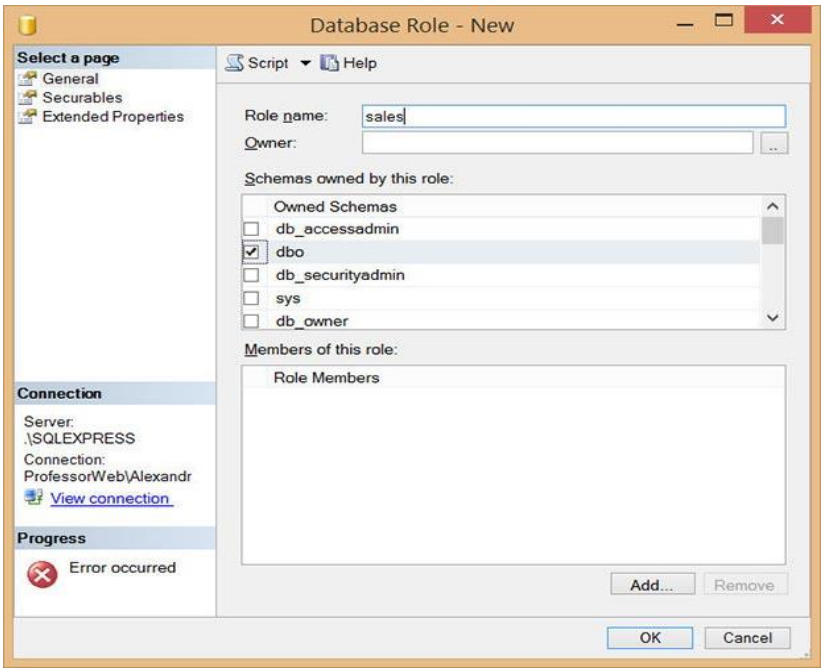


Рис. 38. Добавление роли БД

Нажмите кнопку Add, чтобы добавить членов в новую роль. Выберите требуемых членов (пользователей и/или другие роли) новой роли базы данных и нажмите кнопку OK.

Управление определяемыми пользователем ролями базы данных с помощью инструкций Transact-SQL

Для создания новой определяемой пользователем роли базы данных в текущей базе данных применяется **инструкция CREATE ROLE**. Синтаксис этой инструкции выглядит таким образом:

CREATE ROLE role_name [AUTHORIZATION owner_name]

В параметре role_name инструкции указывается имя создаваемой определяемой пользователем роли, а в параметре owner_name – пользователь базы данных или роль, которая будет владельцем новой роли. (Если пользователь не указан, владельцем роли будет пользователь, исполняющий инструкцию CREATE ROLE.)

Для изменения имени определяемой пользователем роли базы данных применяется **инструкция ALTER ROLE**, а для удаления роли из базы данных – **инструкция DROP ROLE**. Роль, которая является владельцем защищаемых объектов (т.е. объектов базы данных), удалить нельзя. Чтобы удалить такую роль, сначала нужно изменить владельца этих объектов. [8]

В примере ниже показано создание определяемой пользователем роли базы данных и добавление в нее членов:

```
USE SampleDb;

CREATE ROLE marketing AUTHORIZATION Vasya;
```

```
GO
ALTER ROLE marketing ADD MEMBER 'Vasya';
ALTER ROLE marketing ADD MEMBER 'ProfessorWeb\Alexandr';
```

В этом примере сначала создается определяемая пользователем роль базы данных marketing, а затем, *предложением ADD MEMBER* инструкции ALTER ROLE, в нее добавляются два члена – Vasya и ProfessorWeb\Alexandr.

Авторизация пользователей

Выполнять инструкции или осуществлять операции с объектами баз данных могут только авторизованные пользователи. Попытка выполнения любой из этих задач неавторизованным пользователем будет неудачной. Для выполнения задач, связанных с авторизацией, используются следующие три инструкции языка Transact-SQL: GRANT, DENY и REVOKE. [8]

Инструкция GRANT

Инструкция GRANT предоставляет разрешения принципалам на защищаемые объекты. Эта инструкция имеет следующий синтаксис:

```
GRANT {ALL [PRIVILEGES]} | permission_list
   [ON [class::] securable] TO principal_list [WITH GRANT OPTION]
   [AS principal ]
```

Предложение ALL означает, что указанному принципалу предоставляются все применимые к указанному защищаемому объекту разрешения. В параметре permission_list указываются разрешаемые инструкции или объекты (разделенные запятыми), а в параметре class – класс или имя защищаемого объекта, для которого предоставляются разрешения. Предложение on securable указывает защищаемый объект, на который предоставляется разрешение. В параметре prinicpal_list перечисляются все учетные записи (разделенные запятыми), которым предоставляются разрешения. Параметр principal и составляющие списка principal_list могут быть учетной записью пользователя Windows, регистрационным именем или учетной записью пользователя, сопоставленной с сертификатом, регистрационным именем, сопоставленным с асимметричным ключом, пользователем базы данных, ролью базы данных или ролью приложения.

В таблице 5 ниже приводятся разрешения и защищаемые объекты, к которым они применяются, а также краткое описание предоставляемых каждым разрешением возможностей:

Таблица 5

Разрешения и соответствующие защищаемые объекты

Разрешения и соответствующие защищаемые объекты		
Разрешение	**Применение**	**Описание**

SELECT	Таблицы и их столбцы, синонимы, представления и их столбцы, возвращающие табличные значения функции	Предоставляет возможность выборки (чтения) строк. Это разрешение можно ограничить одним или несколькими столбцами, перечислив требуемые столбцы. (Если список столбцов отсутствует, то разрешение применимо ко всем столбцам таблицы)
INSERT	Таблицы и их столбцы, синонимы, представления и их столбцы	Предоставляет возможность вставлять столбцы
UPDATE	Таблицы и их столбцы, синонимы, представления и их столбцы	Предоставляет возможность изменять значения столбцов. Это разрешение можно ограничить одним или несколькими столбцами, перечислив требуемые столбцы. (Если список столбцов отсутствует, то разрешение применимо ко всем столбцам таблицы)
DELETE	Таблицы и их столбцы, синонимы, представления и их столбцы	Предоставляет возможность удалять столбцы
REFERENCES	Определяемые пользователем функции (SQL и среды CLR), таблицы и их столбцы, синонимы, представления и их столбцы	Предоставляет возможность обращаться к столбцам внешнего ключа в родительской таблице, когда пользователь не имеет разрешения SELECT для этой таблицы
EXECUTE	Хранимые процедуры (SQL и среды CLR), определяемые пользователем функции (SQL и среды CLR), синонимы	Предоставляет возможность выполнять указанную хранимую процедуру или определенную пользователем функцию
CONTROL	Хранимые процедуры (SQL и среды CLR), определяемые пользователем функции	Предоставляет возможности, подобные возможностям владельца; получатель имеет практически все разрешения,

	(SQL и среды CLR), синонимы	определенные для защищаемого объекта. Принципал, которому было предоставлено разрешение CONTROL, также имеет возможность предоставлять разрешения на данный защищаемый объект. Разрешение CONTROL на определенной области видимости неявно включает разрешение CONTROL для всех защищаемых объектов в этой области видимости
ALTER	Хранимые процедуры (SQL и среды CLR), определяемые пользователем функции (SQL и среды CLR), таблицы, представления	Предоставляет возможность изменять свойства (за исключением владения) защищаемых объектов. Когда это право предоставляется применимо к области, оно также предоставляет права на выполнение инструкций ALTER, CREATE и DROP на любых защищаемых объектах в данной области
TAKE OWNERSHIP	Хранимые процедуры (SQL и среды CLR), определяемые пользователем функции (SQL и среды CLR), таблицы, представления, синонимы	Предоставляет возможность становиться владельцем защищаемого объекта, для которого оно применяется
VIEW DEFINITION	Хранимые процедуры (SQL и среды CLR), определяемые пользователем функции (SQL и среды CLR), таблицы, представления, синонимы	Предоставляет получателю возможность просматривать метаданные защищаемого объекта
CREATE (безопасность сервера)	Нет данных	Предоставляет возможность создавать защищаемые объекты сервера

CREATE (безопасность базы данных)	Нет данных	Предоставляет возможность создавать защищаемые объекты базы данных

В таблице перечислены только наиболее важные разрешения. Так как модель безопасности компонента Database Engine является иерархической, то она содержит многие гранулярные разрешения, которые не отображены в списке. Описание этих разрешений смотрите в электронной документации. [8]

Применение инструкции GRANT показано в примерах ниже. Для начала, в следующем примере показано использование разрешения CREATE:

```
USE SampleDb;

GRANT CREATE TABLE, CREATE PROCEDURE
   TO Vasya, [ProfessorWeb\Alexandr];
```

В этом примере пользователям Vasya и [ProfessorWeb\Alexandr] дается право на выполнение инструкций языка Transact-SQL CREATE TABLE и CREATE PROCEDURE. (Как можно видеть в этом примере, инструкция GRANT для разрешения CREATE не включает параметр ON.)

В примере ниже пользователю Vasya предоставляется возможность для создания определяемых пользователем функций в базе данных SampleDb:

```
USE SampleDb;

GRANT CREATE FUNCTION TO Vasya;
```

Далее показано использование разрешения SELECT в инструкции GRANT:

```
USE SampleDb;

GRANT SELECT ON Employee
   TO Vasya;
```

Здесь пользователь Vasya получает разрешение на чтение строк из таблицы Employee. Когда разрешение дается учетной записи пользователя Windows или регистрационному имени, это разрешение распространяется только на данную учетную запись (регистрационное имя). С другой стороны, разрешение, предоставленное группе или роли, распространяется на всех пользователей данной группы или роли.

В примере ниже показано использование разрешения UPDATE в инструкции GRANT:

```
USE SampleDb;
```

```
GRANT UPDATE ON Works_on (EmpId, EnterDate)
   TO Vasya;
```

После выполнения инструкции GRANT в этом примере ниже, пользователь Vasya может модифицировать значения столбцов Id и EnterDate таблицы Works_on.

Инструкция DENY

Инструкция DENY запрещает пользователю выполнять указанные действия на указанных объектах. Иными словами, эта инструкция удаляет существующие разрешения для учетной записи пользователя, а также предотвращает получение разрешений пользователем посредством его членства в группе или роли, которое он может получить в будущем. Эта инструкция имеет следующий синтаксис:

```
DENY {ALL [PRIVILEGES] } | permission_list
   [ON [class::] securable] TO principal_list
   [CASCADE] [ AS principal ]
```

Все параметры инструкции DENY имеют точно такое же логическое значение, как и одноименные параметры инструкции GRANT. Инструкция DENY имеет дополнительный *параметр CASCADE*, в котором указывается, что разрешения, запрещенные пользователю A, будут также запрещены пользователям, которым он их предоставил. Если в инструкции DENY параметр CASCADE опущен, и при этом ранее были предоставлены разрешения для соответствующего объекта с использованием предложения WITH GRANT OPTION, исполнение инструкции DENY завершается ошибкой.

Инструкция DENY блокирует разрешения, полученные пользователем, группой или ролью посредством их членства в группе или роли. Это означает, что если член группы, которому запрещено разрешение, предоставленное для группы, то этот пользователь будет единственным из группы, кто не сможет использовать это разрешение. С другой стороны, если разрешение запрещено для всей группы, все члены этой группы не смогут пользоваться этим разрешением. [7]

Инструкцию GRANT можно рассматривать как положительную авторизацию пользователя, а инструкцию DENY – как отрицательную. Обычно инструкция DENY используется для запрещения разрешений, уже предоставленных для группы (или роли), отдельным членам этой группы.

Использование инструкции DENY показано в примерах ниже. В следующем примере мы запрещаем пользователю Vasya два предоставленных ранее разрешения:

```
USE SampleDb;

DENY CREATE TABLE, CREATE PROCEDURE
   TO Vasya;
```

Инструкция DENY в примере отменяет для пользователя Vasya ранее предоставленные ему разрешения на создание таблиц и процедур. В примере ниже показана негативная авторизация для некоторых пользователей базы данных SampleDb:

```
USE SampleDb;

GRANT SELECT ON Project TO PUBLIC;
DENY SELECT ON Project TO Vasya;
```

Вначале предоставляется разрешение на выборку всех строк из таблицы Project всем пользователям базы данных SampleDb. После этого это разрешение отменяется для пользователя Vasya.

Инструкция REVOKE

Инструкция REVOKE удаляет предоставленное или запрещенное ранее разрешение. Эта инструкция имеет следующий синтаксис:

```
REVOKE [GRANT OPTION FOR]
   { [ALL [PRIVILEGES] ] | permission_list ]}
   [ON [class:: ] securable ]
   FROM principal_list [CASCADE] [ AS principal ]
```

Единственным новым параметром инструкции REVOKE является *параметр GRANT OPTION FOR*. Все другие параметры этой инструкции имеют точно такое же логическое значение, как и одноименные параметры инструкций GRANT или DENY. Параметр GRANT OPTION FOR используется для отмены эффекта предложения WITH GRANT OPTION в соответствующей инструкции GRANT. Это означает, что пользователь все еще будет иметь предоставленные ранее разрешения, но больше не сможет предоставлять их другим пользователям.

Инструкция REVOKE отменяет как "позитивные" разрешения, предоставленные инструкцией GRANT, так и "негативные" разрешения, предоставленные инструкцией DENY. Таким образом, его функцией является нейтрализация указанных ранее разрешений (позитивных и негативных).

Использование инструкции REVOKE показано в примере ниже:

```
USE SampleDb;

REVOKE SELECT ON Project FROM public;
```

Инструкция REVOKE в этом примере отменяет предоставленное разрешение выборки данных для роли public. При этом существующее запрещение разрешения этой инструкции для пользователя Vasya не удаляется, поскольку разрешения, предоставленные или запрещенные явно членам группы (или роли), не затрагиваются удалением этих разрешений (положительных или отрицательных) для данной группы. [6]

Управление разрешениями с помощью среды Management Studio

Пользователи базы данных могут выполнять действия, на которые им были предоставлены разрешения. Разрешения на определенные действия установлены в значение g (от GRANT) в столбце state в представлении просмотра каталога sys.database_permissions. Негативная запись в таблице не дает возможность пользователям выполнять деятельность. Значение d (от DENY) в этом столбце state аннулирует разрешение, предоставленное пользователю явно или неявно посредством предоставления его роли, к которой он принадлежит. Таким образом, пользователь не может выполнять данное действие в любом случае. И последнее возможное значение r (от REVOKE) в столбце state означает, что пользователь не имеет никаких явных разрешений, но может выполнять действие, если роли, к которой он принадлежит, предоставлено соответствующее разрешение. [5]

Для управления разрешениями с помощью среды Management Studio разверните сервер, а затем папку "Databases". Щелкните правой кнопкой мыши требуемую базу данных и в контекстном меню выберите пункт Properties. В открывшемся диалоговом окне свойств базы данных Database Properties - SampleDb выберите страницу Permissions, после чего нажмите кнопку Search, чтобы выбрать пользователей, которым предоставить или запретить разрешения. В открывшемся диалоговом окне нажмите кнопку Browse, в диалоговом окне Browse for Objects выберите требуемых пользователей или роли (например, guest и public) и нажмите кнопки ОК соответствующих окон, чтобы добавить выбранных пользователей (роли).

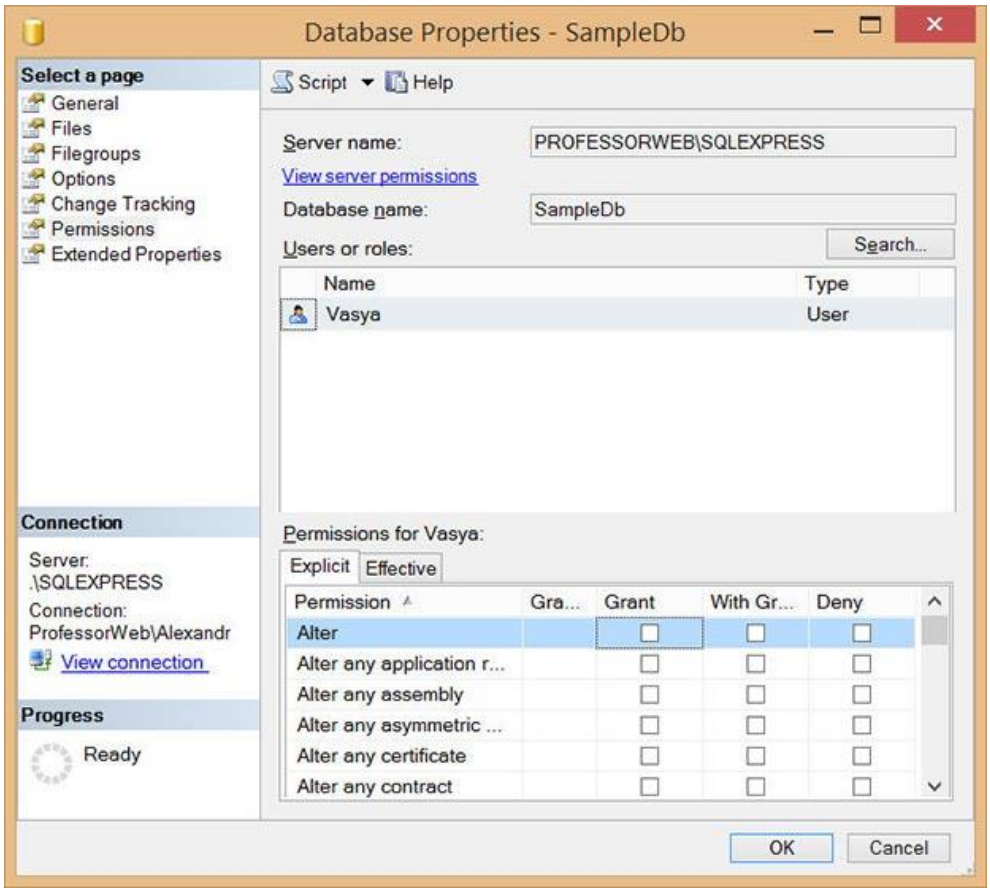


Рис. 39. Управление разрешениями

Выбранные таким образом пользователи будут отображены в окне свойств базы данных в поле Users or roles (Пользователи и роли).

Теперь для выбранных пользователей можно разрешить или запретить выполнение определенных действий. В частности, для предоставления разрешения, для требуемого действия установите флажок в столбце Grant, а для запрещения действия установите для него флажок в столбце Deny. Установленный флажок в столбце With Grant означает, что получатель разрешения также имеет право предоставлять его другим пользователям. Отсутствие установленных флажков в столбце разрешений или запрещений означает, что для данного пользователя нет никаких явных разрешений или запрещений на выполнение соответствующих действий. [12]

Контрольные вопросы

1. Какие фиксированные серверные роли существуют?
2. Как создать роль БД?
3. Как осуществляется управление авторизацией?
4. Как осуществляется управление разрешениями?
5. Пояснить команды GRANT, REVOKE, DENY.

9. Лабораторная работа №9. СОЗДАНИЕ КЛИЕНТСКОЙ ЧАСТИ ПРИЛОЖЕНИЯ ДЛЯ БД.

9.1. Цель работы

Научиться создавать клиентское приложение для работы с базой данных с применением встроенных инструментов на Visual C#.

9.2. Содержание работы

- Подключение визуальной формы к БД
- Привязка элементов формы к sql запросами, хранимым процедурам и функциям

9.3. Задание к работе

Для базы данных из работы №2 создать клиентское приложение, которое реализует добавление, удаление и обновление информации из БД. Также форму по работе с хранимыми процедурами и функциями: добавить на форму кнопки, реализующие хранимые процедуры и пользовательские функции из работы 5, а так же поля ввода для входных параметров и поля для вывода результатов.

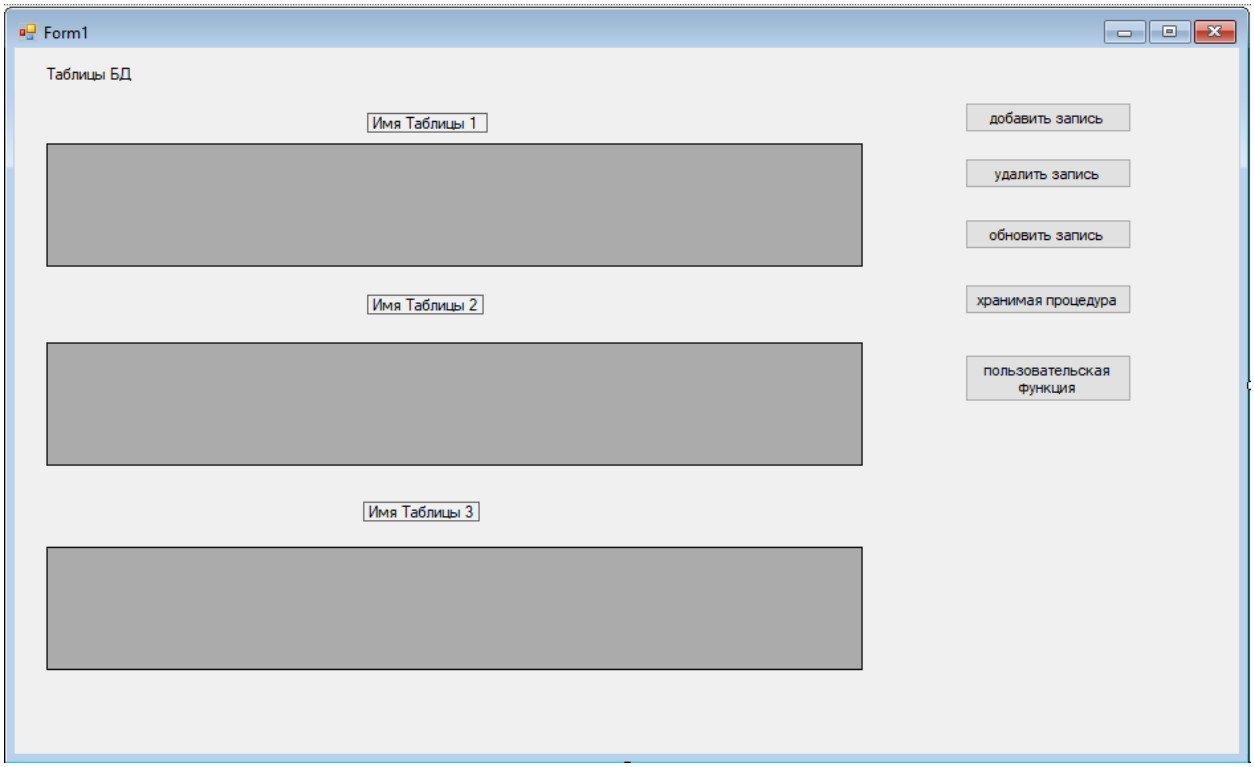


Рис. 40. Интерфейс приложения

9.4. Краткая Теория

Подключение к Базе данных

Рассмотрим как подключать базу данных sql server к программе на C# через Visual Studio.

Создаем новый проект Visual C# – приложение для Windows.

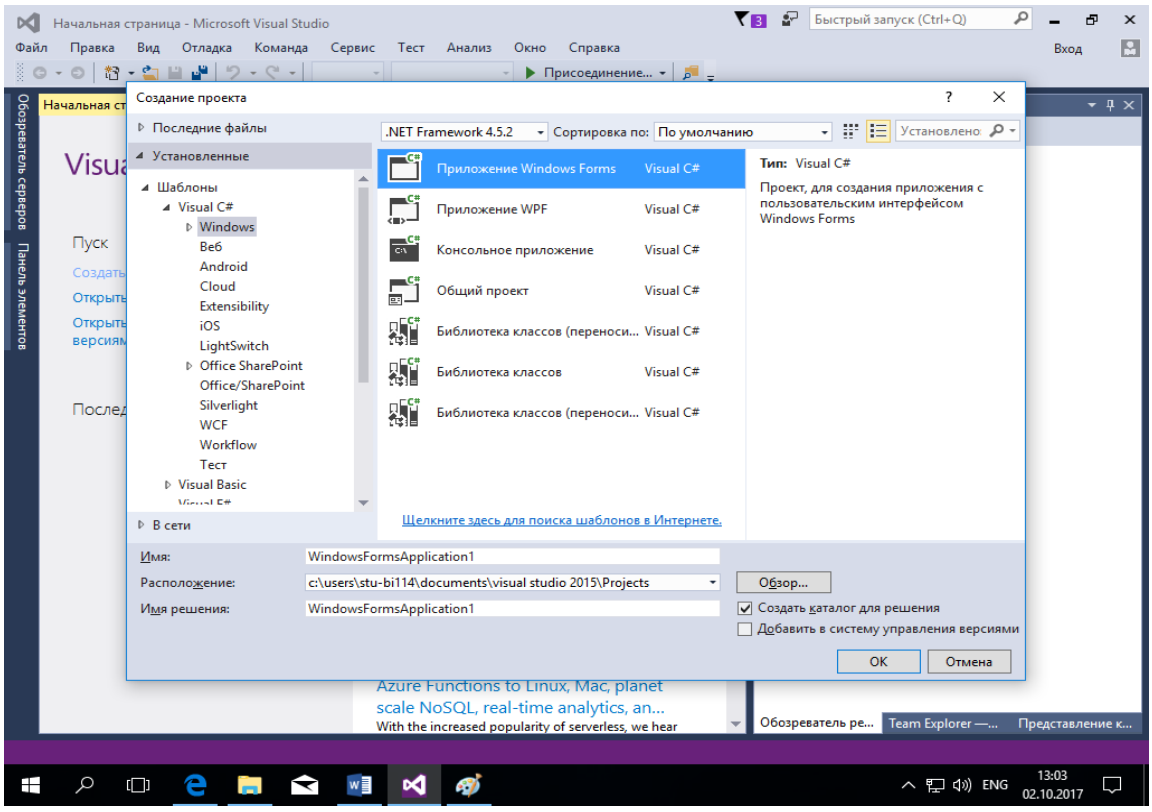


Рис. 41. Новый проект

1. В проекте выбираем меню Tools => Connect to DataBase (Сервис- Подключить к Базе данных).

Выбираем источник данных (DataSet): Microsoft SQL Server.

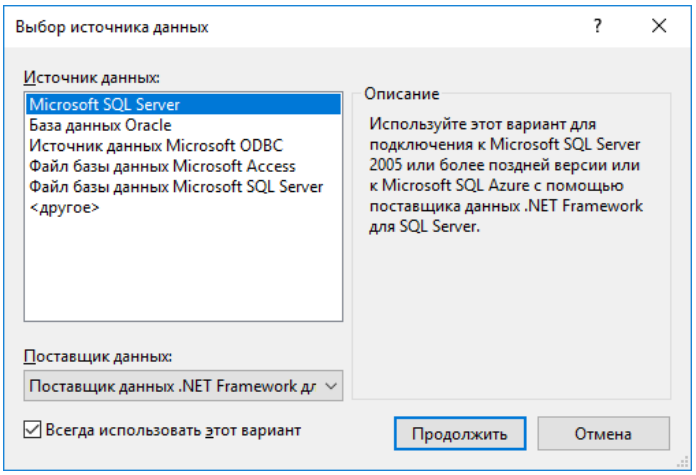


Рис. 42. Выбор источника данных

В открывшемся окне в поле Источник данных (Data Source) ставим Microsoft SQL Server, в поле Server Name –amd-sql\stufent, далее в поле Select or enter DB name выберите имя БД, к которой будем подключаться, и нажмите OK.

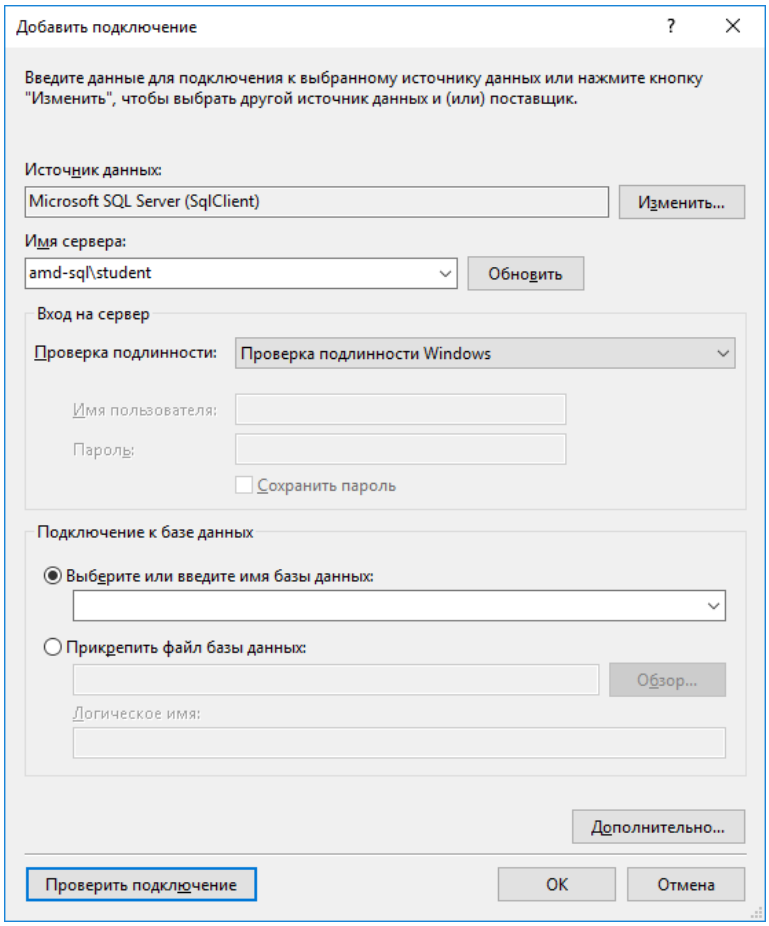


Рис. 43. Источник данных

Теперь открыв окно Обозреватель серверов (Server explorer) можно увидеть подключенную БД. Нажав на нее, в окне свойств копируем Connection String, она еще пригодится.

2. В конструкторе создаем новую форму, на нее добавляем компонент DataGridView:

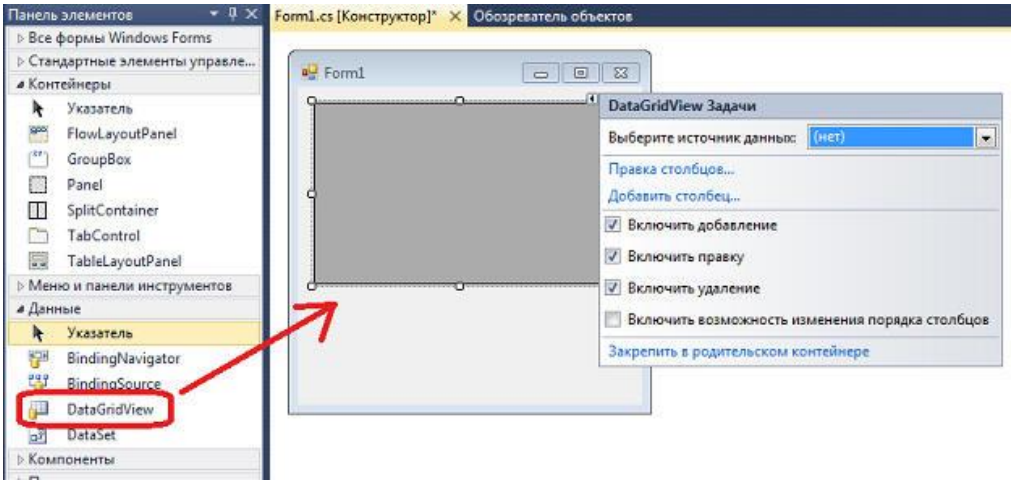


Рис. 44. Добавление формы данных

3. Во вкладке Свойства находим Данные (Data) – DataSet - выберем Добавить источник данных проекта(Add New Data Source). В появившемся окне выберем База данных(DataBase) .

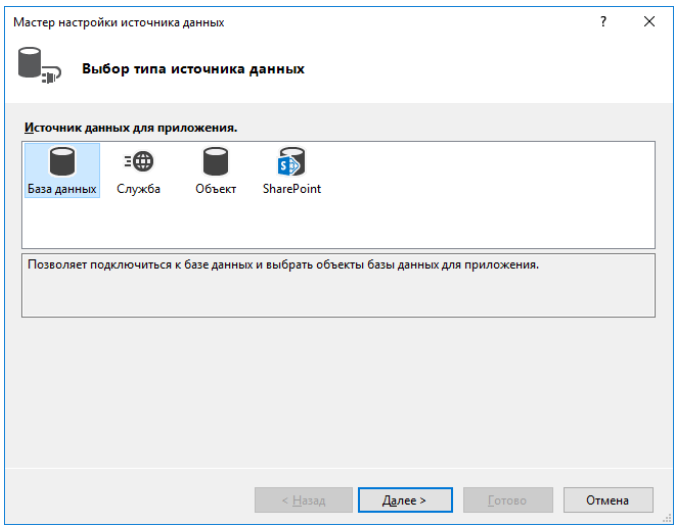


Рис. 45. Добавление базы данных

Нажмем Далее(Next).

Выбираем: "Набор данных" и жмем кнопочку "Далее":

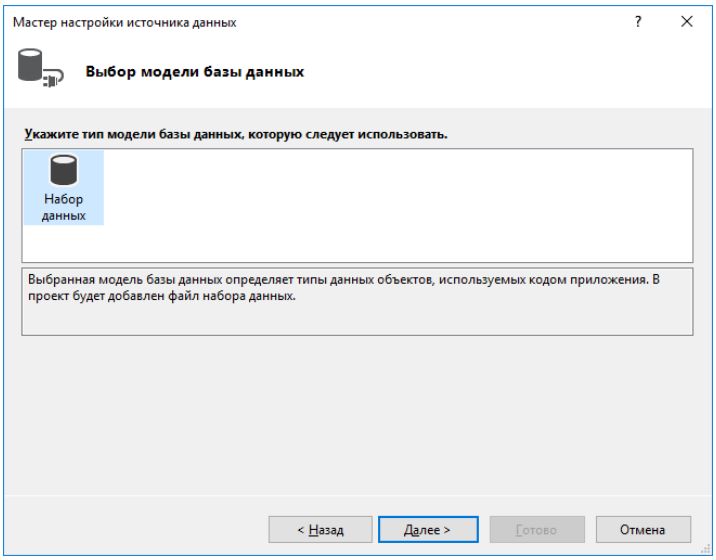


Рис. 46. Добавление набора данных

Выбираем нашу БД, жмем Next.

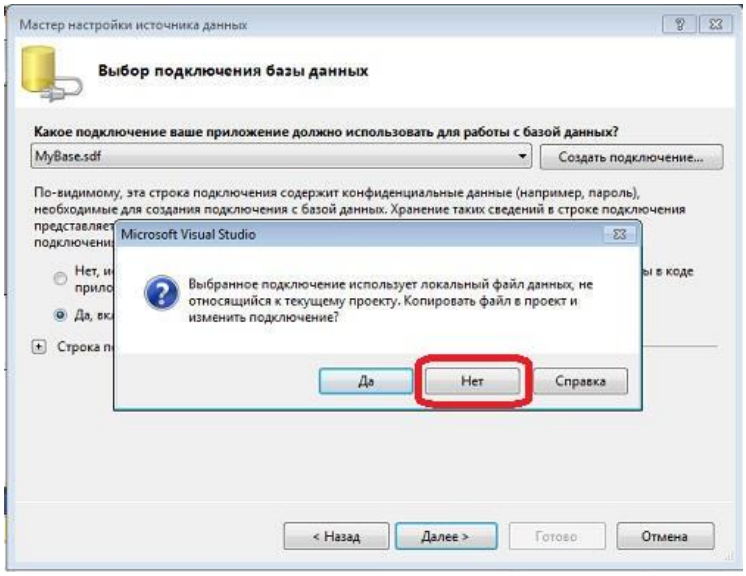


Рис. 47. Добавление файла БД

В следующем диалоге опять жмем "Далее":

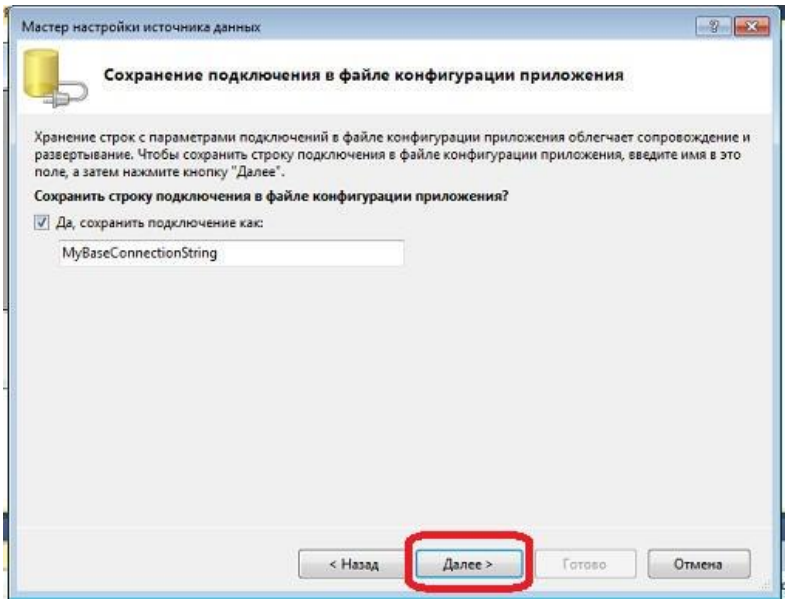


Рис. 48. Сохранение подключения

После чего у нас появиться список данных, доступных для выбора:

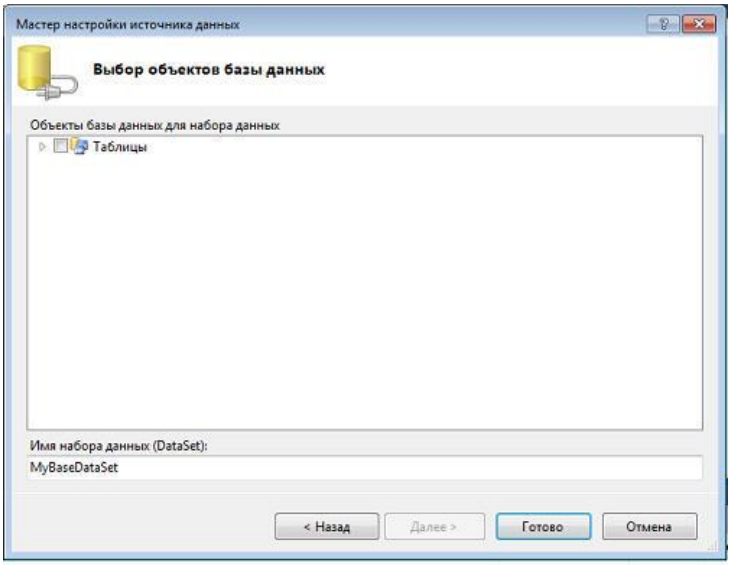


Рис. 49. Выбор таблицы БД

Выберем созданную нами таблицу:

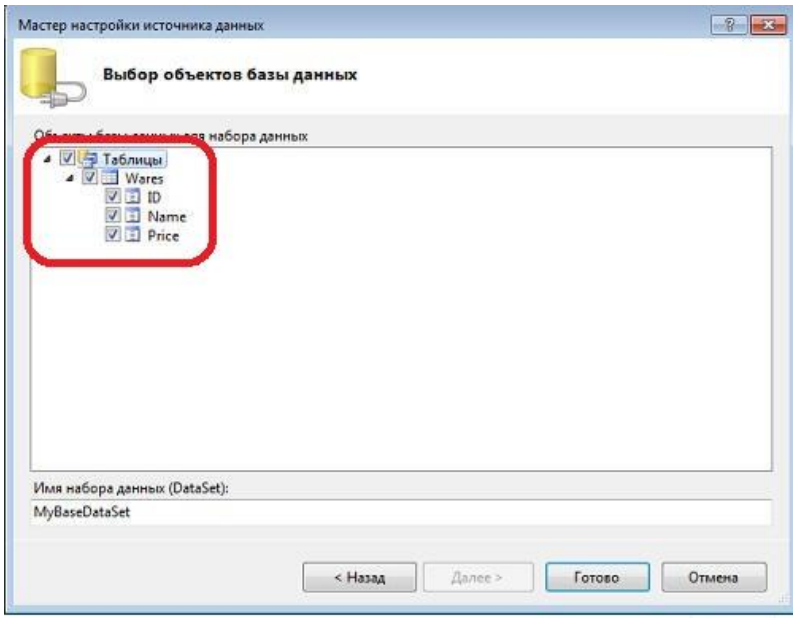


Рис. 50. Выбор столбцов таблицы

После нажатия кнопочки "Готово" у нас в компоненте DataGridView отобразятся столбцы выбранной таблицы:

Рис. 51. Связка с DataGridView

Теперь мы можем запустить программу и даже редактировать данные в таблице базы данных.

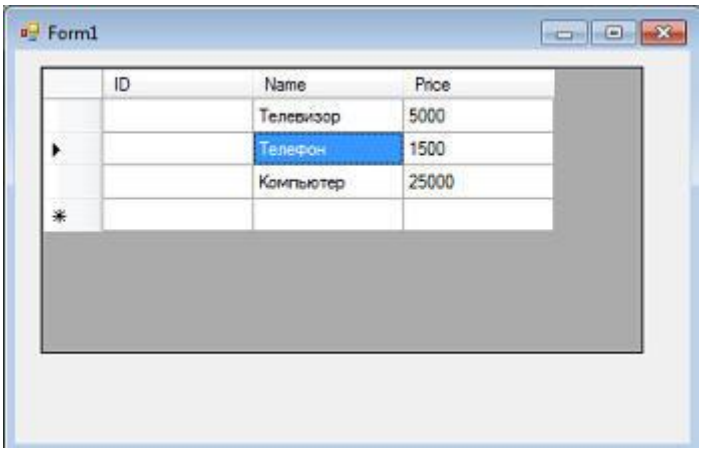


Рис. 52. Итоговое приложение

ADO.NET

Платформа .NET определяет ряд пространств имен, которые позволяют непосредственно взаимодействовать с локальными и удаленными базами данных. Вместе эти пространства имен известны как ADO.NET. Рассмотрим два уровня работы с ADO.NET: ***подключенный уровень (connected layer)*** - позволяет взаимодействовать с базой данных с помощью объектов подключения, чтения данных и команд конкретного поставщика данных; ***отключенный уровень (disconnected layer)*** - позволяет смоделировать в памяти данные из базы данных с помощью многочисленных классов из пространства имен System.Data (DataSet, DataTable и т.д.)

ADO.NET поставляется с тремя пространствами имен клиента базы данных: одно для **SQL Server**, другое для источников данных **Open Database Connectivity (ODBC)** и третье для любой базы данных, доступной через **OLE DB**. [7]

Поставщики данных в ADO.NET

Поставщик данных (data provider) — это набор классов ADO.NET, которые позволяют получать доступ к определенной базе данных, выполнять команды SQL и извлекать данные. По сути, поставщик данных — это мост между вашим приложением и источником данных.

Таблица 6

Основные объекты поставщиков данных ADO.NET

Тип объекта	Базовый класс	Соответствующие интерфейсы	Назначение
Connection	DbConnection	IDbConnection	Позволяет подключаться к хранилищу данных и отключаться от него. Кроме того, объекты подключения

			обеспечивают доступ к соответствующим объектам транзакций
Command	DbCommand	IDbCommand	Представляет SQL-запрос или хранимую процедуру. Кроме того, объекты команд предоставляют доступ к объекту чтения данных конкретного поставщика данных
DataReader	DbDataReader	IDataReader, IDataRecord	Предоставляет доступ к данным только для чтения в прямом направлении с помощью курсора на стороне сервера
DataAdapter	DbDataAdapter	IDataAdapter, IDbDataAdapter	Пересылает наборы данных из хранилища данных к вызывающему процессу и обратно. Адаптеры данных содержат подключение и набор из четырех внутренних объектов команд для выборки, вставки, изменения и удаления информации в хранилище данных
Parameter	DbParameter	IDataParameter, IDbDataParameter	Представляет именованный параметр в параметризованном запросе
Transaction	DbTransaction	IDbTransaction	Инкапсулирует транзакцию в базе данных

Фундаментальные классы ADO.NET

Классы ADO.NET группируются в несколько пространств имен. Каждый поставщик имеет свое собственное пространство имен, а обобщенные классы вроде DataSet находятся в пространстве имен

System.Data. Ниже описаны наиболее важные пространства имен для базовой поддержки ADO.NET:

System.Data

Содержит ключевые классы контейнеров данных, которые моделируют столбцы, отношения, таблицы, наборы данных, строки, представления и ограничения. Дополнительно содержит ключевые интерфейсы, которые реализованы объектами данных, основанными на соединениях.

System.Data.Common

Содержит базовые, наиболее абстрактные классы, которые реализуют некоторые из интерфейсов из System.Data и определяют ядро функциональности ADO.NET. Поставщики данных наследуются от этих классов (DbConnection, DbCommand и т.п.), создавая собственные специализированные версии.

System.Data.OleDb

Содержит классы, используемые для подключения к поставщику OLE DB, включая OleDbCommand, OleDbConnection и OleDbDataAdapter. Эти классы поддерживают большинство поставщиков OLE DB, но не те, что требуют интерфейсов OLE DB версии 2.5

System.Data.SqlClient

Содержит классы, используемые для подключения к базе данных Microsoft SQL Server, в том числе SqlDbCommand, SqlDbConnection и SqlDbDataAdapter. Эти классы оптимизированы для использования интерфейса TDS к SQL Server

System.Data.OracleClient

Содержит классы, необходимые для подключения к базе данных Oracle (версии 8.1.7 и выше), в том числе OracleCommand, OracleConnection и OracleDataAdapter. Эти классы используют оптимизированный интерфейс OCI (Oracle Call Interface – Интерфейс вызовов Oracle)

System.Data.Odbc

Содержит классы, необходимые для подключения к большинству драйверов ODBC, такие как OdbcCommand, OdbcConnection, OdbcDataReader и OdbcDataAdapter. Драйверы ODBC поставляются для всех видов источников данных и конфигурируются через значок Data Sources (Источники данных) панели управления

System.Data.SqlTypes

Содержит структуры, соответствующие встроенным типам данных SQL Server. Эти классы не являются необходимыми, но предоставляют альтернативу применению стандартных типов данных .NET, требующих автоматического преобразования. [3]

Использование SQL команд

Для того, чтобы команда исполнилась, нужно ввести текст команды, затем определить ее тип и привязать к соединению. Рассмотрим какие бывают типы команд:

Text – выполнится текст SQL оператора

StoredProcedure – выполнится хранимая процедура

TableDirect – вернет все записи в таблице

Для того, чтобы установить тип процедуры нужно подключить пространство имен: `using System.Data;`

Теперь рассмотрим пример вызова текстовой sql команды:

```
SqlConnection con = new SqlConnection(connectionString);
SqlCommand command = new SqlCommand();
command.Connection = con;//устанавливаем соединение
command.CommandType = CommandType.Text;//тип команды
    .CommandText = "SELECT * FROM Student";//текст команды
```

Но для более короткой записи можно использовать конструктор SqlCommand, так как он по умолчанию имеет тип Text.

```
SqlCommand command = new SqlCommand("SELECT * FROM
Student",con);
```

Если вы хотите сократить запись команды, которая вызывает хранимую процедуру, то можете использовать следующий код:

```
SqlCommand command = new SqlCommand("GetAllStudents",con);
command.CommandType = CommandType.StoredProcedure;
```

Этот код вызывает хранимую процедуру GetAllStudents. [7]

В приведенном выше коде, команда создается, но не выполняется. Существует три способа как вызвать SQL оператор:

ExecuteNonQuery() – используется для вызова команд удаления, вставки и обновления записей, возвращает количество обработанных записей

ExecuteScalar() – выполняет SQL оператор и возвращает значение первого поля первой строки

ExecuteReader() – выполняет запрос SELECT и возвращает объект DataReader, который содержит результат выполнения команды

Теперь необходимо ознакомиться к контейнером данных DataReader.

Рассмотрим его методы:

Read() – метод считывает следующую строку. Возвращает true, если существует следующая строка, и false в противном случае. Необходимо вызвать его перед началом чтения данных

GetValue() – возвращает значение указанного элемента в строке. Имеется возможность обращаться к элементу как по номеру, так и по имени.GetValue(«ID»); и GetValue(0) эквивалентные вызовы в нашей таблице.

GetValues() – сохраняет значения строки в массив

NextResult() – возвращает следующий набор данных, если в сгенерированном DataReader более одного набора

Close () – закрывает модуль чтения. [3]

Пример использования SqlCommand для возврата количества строк в таблице:

```
string connStr = "Data Source=MyServer;Initial
Catalog=MyDb;Integrated Security=SSPI";

using (SqlConnection conn = new SqlConnection(connStr))
{
  SqlCommand cmd = new SqlCommand("SELECT COUNT(*) FROM
MyTable", conn);
  conn.Open();
  int numRows = (int) cmd.ExecuteScalar();

  Console.WriteLine("Количество строк в MyTable: " + numRows);

  conn.Close();
}
```

Пример использования SqlCommand для обновления таблицы:

```
string connStr = "Data Source=MyServer;Initial
Catalog=MyDb;Integrated Security=SSPI";

using (SqlConnection conn = new SqlConnection(connStr))
{
  SqlCommand cmd = new SqlCommand("UPDATE MyTable SET Price =
Price * 2", conn);
  conn.Open();
  int numRows = cmd.ExecuteNonQuery();
```

```
  Console.WriteLine("Количество обновлённых строк: " + numRows);

  conn.Close();
}
```

Для возврата результата может быть использован класс SqlDataReader (*System.Data.SqlClient*.SqlDataReader).

Этот класс предоставляет возможность чтения вперёд результирующего набора строк с возможностью обратиться к каждой строке. Класс использует понятие текущей записи. Для доступа к значениям полей текущей записи могут использоваться индексаторы по номеру поля (начиная с нуля) или по имени (не рекомендуется), а также набор методов Get[DataType], которые возвращают преобразованные к DataType значения полей. В момент чтения подключение должно быть активно. При необходимости последующего использования подключения вызывается SqlDataReader.Close(), если подключение не нужно, то SqlConnection.Dispose(). [6]

Пример использования SqlDataReader для чтения результирующего набора данных:

```
    string queryStr = "SELECT ID, S, N FROM MyTable";
string connStr = "Data Source=MyServer;Initial Catalog=MyDb;Integrated
Security=SSPI";

using (SqlConnection conn = new SqlConnection(connStr))
{
  SqlCommand cmd = new SqlCommand(queryStr, conn);
  conn.Open();

  SqlDataReader r = cmd.ExecuteReader();

  // Чтение данных
  while (r.Read())
  {
    Console.WriteLine(
      // форматированный вывод
      String.Format("ID:{0}, S:{1}, N:{2}",
      // доступ к значению по индексатору с номером поля
      r[0],
      // доступ к значению по индексатору с именем поля
      r["S"],
      // доступ к значению с использованием методов Get
      r.GetString(2) //
      ));
```

```
  }

  // Вызов Close() после завершения чтения
  r.Close();
}
```

Пример привязки хранимой процедуры к кнопке

В sql server есть БД students: таблица Stud с полями: id(счетчик), fio, age), а также хранимая процедура по имени inesrt_d. Тело хранимой процедуры:

```
CREATE PROCEDURE inesrt_d
 @nm nvarchar(50),
 @ag INT
 AS
 BEGIN TRANSACTION
 INSERT INTO Stud (fio,age) VALUES
(@nm,@ag);
 COMMIT TRANSACTION;
```

Имеется форма(C#), на ней textbox1 для fio ,textbox2 для age и кнопка button1.

Осуществлена привязка к форме бд students с помощью визуального подключения visual studio 2010 (Add New Data Source...), т.е. реализована работа с бд в автономном уровне.

Необходимо привязать к button1 хранимую процедуру для ее вызова, привязать к текстбоксам входные параметры хранимой процедуры.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace WindowsFormsApplication4
{
   public partial class Form1 : Form
   {
      public Form1()
      {
```

```
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Insert(textBox1.Text,Int32.Parse(textBox2.Text));
        }

        private bool Insert(string fio, int age)
        {
            bool result = false;
                using (SqlConnection con = new SqlConnection(/* zapishi
connection string*/))
            {
                SqlCommand cmd = new SqlCommand();
                cmd.Connection = con;
                cmd.CommandText = "inesrt_d";
                cmd.CommandType = CommandType.StoredProcedure;
                cmd.Parameters.Clear();
                cmd.Parameters.AddWithValue("pfio", fio);
                cmd.Parameters.AddWithValue("pAge", age);
                try
                {
                    con.Open();
                    if (cmd.ExecuteNonQuery() == 1)
                    {
                        result = true;
                    }
                }
                catch { }
                return result;
            }
        }
    }
}
```

Контрольные вопросы

1. Как устанавливается соединение приложения с базой данных?
2. Какие поставщики данных в ADO.NET используются для чтения и возврата данных из базы данных?

3. Как добавить SQL в обработчик для какой-либо формы приложения?

4. Как связать хранимую процедуру или функцию из базы данных с обработчиком для какой-либо формы приложения?

5. Как можно передавать и возвращать параметры в пользовательские функции или хранимые процедуры для визуального приложения?

Библиографический список

1. А.В. Бурков Проектирование информационных систем в Microsoft SQL Server 2008 и Visual Studio 2008. Режим доступа: http://window.edu.ru/resource/403/61403/files/burkov-pract.pdf
2. Глушаков С. В. Базы данных: [учебное издание] / С. В. Глушаков, Д. В. Ломотько. – Харьков- Москва: Фолио, АСТ, 2002. 504 с.
3. Работа с базами данных на языке C#. Технология ADO.NET: Учебное пособие / сост. О. Н. Евсеева, А. Б. Шамшев. - Ульяновск: УлГТУ, 2009. – 170 с.
4. Интуит. Ру. (коллектив авторов) Разработка и защита баз данных в Microsoft SQL Server 2005 (2-е изд.) М.: НОУ "Интуит", 2016, 147 с.
5. Туманов В.Е. Основы проектирования реляционных баз данных. – Интуит, 2016, 504 с.
6. Семенова И.И. Разработка клиент-серверных приложений в Microsoft SQL Server 2005 и Microsoft Visual C# 2005 Express Edition: Учебно-методическое пособие. – Омск: Изд-во СибАДИ, 2010. – 65 с.
7. В.М. Снетков Разработка приложений на C# в среде Visual Studio 2005 Режим доступа: http://www.intuit.ru/studies/courses/592/448/info
8. И.П. Карпова Изучение основ языка SQL: Методические указания к лабораторным работам по курсу "Базы данных" / Московский институт электроники и математики НИУ ВШЭ; Сост.: И.П. Карпова. – М., 2012. – 39 с.
9. Клайн К., Клайн Д., Хант Б. SQL справочник. – Пер. с англ. – 3-е издание. – СПб.: Символ, 2010. – 652 с.
10. Фиайли К. SQL: Пер. с англ. – М.: ДМК Пресс, 2003. – 456 с.: ил. (Серия «Quick Start»).
11. Филипп Андон, Валерий Резниченко Язык запросов SQL. Учебный курс Питер, BHV – Киев**,** 2006**,** 416 с.
12. Михаил Фленов Transact-SQL. – Издательство: БХВ-Петербург, 2006 576 с.
13. Швецов В.И., Визгунов А.Н., Мееров И.Б. Базы данных. учеб. пособие. – Нижний Новгород: Изд-во ННГУ, 2004. 217 с.