Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет им. Г.И. Носова»

Многопрофильный колледж

УТВЕРЖДАЮ
Директор
_____ / С.А. Махновский
«09» февраля 2022 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

ПМ.11 Разработка, администрирование и защита баз данных
МДК.11.01 «Технология разработки и защиты баз данных»

для обучающихся специальности

09.02.07 Информационные системы и программирование
Квалификация: Программист

Магнитогорск, 2022

ОДОБРЕНО:

Предметно-цикловой комиссией «Информатики и вычислительной техники»
Председатель И.Г.Зорина
Протокол № 5 от19.01.2022

Методической комиссией МпК
Протокол №4 от «09» февраля 2022 г.

Разработчики:
преподаватель ФГБОУ ВО «МГТУ им. Г.И. Носова» Многопрофильный колледж
Н.В. Кучерова
преподаватель ФГБОУ ВО «МГТУ им. Г.И. Носова» Многопрофильный колледж
И.Г. Зорина

Методические указания по выполнению практических и лабораторных работ разработаны на основе рабочей программы ПМ.11 Разработка, администрирование и защита баз данных, МДК11.1 Технология разработки и защиты баз данных.

Содержание практических и лабораторных работ ориентировано на формирование общих и профессиональных компетенций по программе подготовки специалистов среднего звена по специальности 09.02.07 Информационные системы и программирование.

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

Состав и содержание практических и лабораторных занятий направлены на реализацию Федерального государственного образовательного стандарта среднего профессионального образования.

Ведущей дидактической целью практических занятий является формирование профессиональных практических умений (умений выполнять определенные действия, операции, необходимые в последующем в профессиональной деятельности).

Ведущей дидактической целью лабораторных занятий является экспериментальное подтверждение и проверка существенных теоретических положений (законов, зависимостей).

В соответствии с рабочей программой ПМ.11 Разработка, администрирование и защита баз данных, МДК.11.1 Технология разработки и защиты баз данных, предусмотрено проведение практических и лабораторных занятий. В рамках практического/лабораторного занятия обучающиеся могут выполнять одну или несколько практических/лабораторных работ.

В результате их выполнения, обучающийся должен:

уметь:

- У1. работать с современными case-средствами проектирования баз данных;
- У2. проектировать логическую и физическую схемы базы данных;
- У3. создавать хранимые процедуры и триггеры на базах данных;
- У4. применять стандартные методы для защиты объектов базы данных;
- У5. выполнять стандартные процедуры резервного копирования и мониторинга выполнения этой процедуры;
- У6. выполнять процедуру восстановления базы данных и вести мониторинг выполнения этой процедуры;
- У7. обеспечивать информационную безопасность на уровне базы данных;
- У8. собирать, обрабатывать и анализировать информацию на предпроектной стадии;
- У9. создавать объекты баз данных в современных СУБД;
- У10. выполнять установку и настройку программного обеспечения для обеспечения работы пользователя с базой данных.

Содержание практических и лабораторных занятий ориентировано на формирование общих компетенций по профессиональному модулю программы подготовки специалистов среднего звена по специальности и овладению ***профессиональными компетенциями***:

ПК 11.1 Осуществлять сбор, обработку и анализ информации для проектирования баз данных.

ПК 11.2 Проектировать базу данных на основе анализа предметной области.

ПК 11.3 Разрабатывать объекты базы данных в соответствии с результатами анализа предметной области.

ПК 11.4 Реализовывать базу данных в конкретной системе управления базами данных.

ПК 11.5 Администрировать базы данных.

ПК 11.6 Защищать информацию в базе данных с использованием технологии защиты информации.

А также формированию ***общих компетенций***:

ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

ОК 02. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.

ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие.

ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.

ОК 06. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей, применять стандарты антикоррупционного поведения.

ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях.

ОК 08. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности.

ОК 09. Использовать информационные технологии в профессиональной деятельности.

ОК 10. Пользоваться профессиональной документацией на государственном и иностранном языках.

ОК 11. Использовать знаний по финансовой грамотности, планировать предпринимательскую деятельность в профессиональной сфере.

Выполнение обучающимися практических и лабораторных работ по ПМ.11 Разработка, администрирование и защита баз данных, МДК.11.1 Технология разработки и защиты баз данных, направлено на:

- обобщение, систематизацию, углубление, закрепление, развитие и детализацию полученных теоретических знаний по конкретным темам профессионального модуля;
- формирование умений применять полученные знания на практике, реализацию единства интеллектуальной и практической деятельности;
- формирование и развитие умений: наблюдать, сравнивать, сопоставлять, анализировать, делать выводы и обобщения, самостоятельно вести исследования, оформлять результаты в виде таблиц, схем, графиков;
- развитие интеллектуальных умений у будущих специалистов: аналитических, проектировочных, конструктивных и др.;
- выработку при решении поставленных задач профессионально значимых качеств, таких как самостоятельность, ответственность, точность, творческая инициатива.

Практические и лабораторные занятия проводятся после соответствующей темы, которая обеспечивает наличие знаний, необходимых для ее выполнения.

2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Тема 1.1 Основы хранения и обработки данных. Проектирование базы данных

Лабораторная работа № 1
Сбор и анализ информации

Цель: получение практических навыков анализа предметной области

Выполнив работу, Вы будете:
уметь:
- выполнять анализ предметной области.

Материальное обеспечение:
Методические указания для выполнения практических работ, вариант задания

Задание:
Выполнить сбор и анализ информации по заданной предметной области.

Порядок выполнения работы:
1. Собрать информацию по предметной области, выявить документы-источники данных для создания базы данных;
2. Проанализировать предметную область;
3. Определить сущности;
4. Определить атрибуты для каждой сущности;
5. Определить связи между сущностями.

Форма представления результата:
Оформленная схема базы данных.

Критерии оценки:
Оценка «отлично» ставится, если задание выполнено верно.
Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.
Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.
Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.1 Основы хранения и обработки данных. Проектирование базы данных

Лабораторная работа № 2
Проектирование реляционной схемы базы данных в среде СУБД

Цель: получение практических навыков проектирования базы данных

Выполнив работу, Вы будете:
уметь:
- проектировать базу данных;
- работать с современными case-средствами проектирования баз данных.

Материальное обеспечение:
Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MySQL Workbench.

Задание:
В соответствии со своим вариантом проанализируйте предметную область решаемой задачи и разработайте логическую структуру соответствующей базы данных.

Варианты заданий:

Вариант	Наименование базы данных, перечень таблиц и их показателей
1	*БД «Лицензионное программное обеспечение».* Таблица «Лицензии»: номер лицензии; название; код CD-диска; код владельца. Таблица «CD-диски»: код CD-диска; дата выпуска; вид программного обеспечения; общий объем файлов, кбайт; пояснения о назначении и свойствах программного обеспечения. Таблица «Владельцы»: код владельца; владелец; город; адрес; телефон
2	*БД «Студенты МпК».* Таблица «Группы»: отделение; группа; Ф.И.О. кл. руководителя. Таблица «Студенты»: группа; шифр студента; Ф.И.О.; адрес; телефон; хобби. Таблица «Дисциплины»: шифр дисциплины; наименование дисциплины. Таблица «Успеваемость»: дата; шифр дисциплины; шифр студента; оценка; отметка о пропуске занятия
3	*БД «Гостиница».* Таблица «Номерной фонд»: категория номера (люкс, одноместный первой категории, двухместный первой категории и др.); номер помещения; место (А, Б, … – в зависимости от количества мест в номере); стоимость проживания за сутки. Таблица «Проживание»: дата заезда; дата выезда; номер помещения; место; Ф.И.О.; паспортные данные. Таблица «Бронирование»: дата заявки; код брони; категория номера; количество человек; дата заезда; срок пребывания
4	*БД «Товарооборот».* Таблица «Поставщики»: код поставщика; поставщик; адрес; телефон. Таблица «Товары»: код товара; товар; единица измерения; цена. Таблица «Поступление товаров»: дата поступления; код поставщика; код товара; количество. Таблица «Продажа»: дата продажи; код товара; количество
5	*БД «Промышленность региона».* Таблица «Предприятия»: код предприятия; предприятие; форма собственности; адрес; основной вид продукции. Таблица «Виды налогов»: код налога; вид налога (на прибыль, на имущество, НДС и др.); ставка, %. Таблица «Налоговые платежи»: код предприятия; код налога; дата платежа; сумма платежа. Таблица «Прибыль»: год; месяц; код предприятия; прибыль
6	*БД «Пассажирские поезда».* Таблица «Поезда»: категория поезда (скорый, пассажирский, пригородный); номер поезда; название поезда (для фирменного

	поезда). Таблица «Составы вагонов»: номер поезда; код состава; общее количество вагонов; схема формирования (например, 3К + 8П – три купейных и восемь плацкартных вагонов). Таблица «Расписание»: номер поезда; время прибытия; время отправления; режим движения (дни следования); станция назначения. Таблица «Перевозки»: дата отправления; номер поезда; код состава; количество пассажиров; прибыль за поездку
7	*БД «Автопарк».* Таблица «Типы автобусов»: код автобуса; марка автобуса; количество мест. Таблица «Парк»: код автобуса; гаражный номер; государственный номер; год выпуска. Таблица «Водители»: табельный номер водителя; Ф.И.О.; дата рождения; оклад; номер маршрута. Таблица «Перевозки»: дата; код автобуса; номер маршрута; табельный номер водителя; время выхода автобуса на маршрут; время прибытия автобуса с маршрута; причина схода автобуса с маршрута; количество проданных билетов
8	*БД «Агентство недвижимости».* Таблица «Риэлторы»: код риэлтора; Ф.И.О.; телефон. Таблица «Недвижимость»: номер объекта; адрес; тип дома; общая площадь; количество комнат; наличие балкона; наличие телефона; стоимость. Таблица «Сделки»: дата; регистрационный номер договора; код риэлтора; номер объекта
9	*БД «Авиаперевозки».* Таблица «Авиапарк»: код модели самолета; модель самолета; количество мест. Таблица «Рейсы и тарифы»: рейс; аэропорт отправления; аэропорт назначения; код класса; вид класса (бизнес-класс, эконом-класс); стоимость билета. Таблица «Перевозки»: рейс; дата вылета; код модели самолета; количество пассажиров; доход за рейс

Краткие теоретические сведения:

Проектирование базы данных заключается в ее многоступенчатом описании с различной степенью детализации и формализации, в ходе которого производится уточнение и оптимизация структуры базы данных. Проектирование начинается с описания предметной области и задач информационной системы, идет к более абстрактному уровню логического описания данных и далее – к схеме физической (внутренней) модели базы данных. Трем основным уровням моделирования системы – **концептуальному, логическому и физическому** соответствуют три последовательных этапа детализации описания объектов базы данных и их взаимосвязей.

На **концептуальном уровне** проектирования производиться смысловое описание информации предметной области, определяются ее границы, производиться абстрагирование от несущественных деталей. В результате определяются моделируемые объекты, и их свойства, и связи. Выполняется структуризация знаний о предметной области, стандартизируется терминология. Затем строится концептуальная модель, описываемая на естественном языке. Для описания свойств и связей объектов применяют различные диаграммы.

На следующем шаге принимается решение о том, в какой конкретно СУБД будет реализована база данных. **Выбор СУБД** является сложной задачей и должен основывается на потребностях с точки зрения информационной системы и пользователей. Определяющими здесь являются вид программного продукта и категория пользователей (или профессиональные программисты, или конечные пользователи, или то и другое).

Другими показателями, влияющими на выбор СУБД, являются:

- Удобство и простота использования;
- Качество средств разработки, защиты и контроля базы данных;
- Уровень коммуникационных средств (в случае применения ее в сетях);
- Фирма-разработчик;
- Стоимость.

Каждая конкретная СУБД работает с определенной моделью данных. Под моделью данных понимается способ их взаимосвязи: в виде иерархического дерева, сложной сетевой структуры или связанных таблиц. В настоящее время большинство СУБД использует табличную модель данных, называемую *реляционной*.

На **логическом уровне** производиться отображение данных концептуальной модели в логическую модель в рамках той структуры данных, которая поддерживается выбранной СУБД. Логическая модель не зависит от конкретной СУБД и может быть реализована на любой СУБД реляционного типа.

На **физическом уровне** производиться выбор рациональной структуры хранения данных и методов доступа к ним, которые обеспечивает выбранная СУБД. На этом уровне решаются вопросы эффективного выполнения запросов к БД, для чего строятся дополнительные структуры, например индексы. В физической модели содержится информация обо всех объектах базы данных (таблицах, индексах, процедурах и др.) и используемых типах данных. Физическая модель *зависит* от конкретной СУБД. Одной и той же логической модели может соответствовать несколько разных физических моделей. Физическое проектирование является начальным этапом реализации базы данных.

Порядок выполнения работы:

1. Выполнить анализ предметной области.
2. Выполнить анализ данных.
3. Определить набор атрибутов для данной предметной области.
4. Определить набор таблиц.
   При проектировании таблиц рекомендуется руководствоваться следующими основными принципами:
   - каждая таблица должна содержать данные только на одну тему;
   - данные не должны дублироваться.
5. Создать словарь имен.
6. Определить состав и типы полей.
7. Создать связи между таблицами.
8. Создать схему базы данных в MySQL Workbench.

Форма представления результата:

Оформленная схема базы данных.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.1 Основы хранения и обработки данных. Проектирование базы данных

Лабораторная работа № 3
Приведение базы данных к нормальной форме 3НФ

Цель: получение практических навыков по нормализации базы данных

Выполнив работу, Вы будете:
уметь:
- проектировать логическую и физическую схемы базы данных;
- работать с современными case-средствами проектирования баз данных.

Материальное обеспечение:
Методические указания для выполнения лабораторных работ, вариант задания, компьютер, программное обеспечение: MySQL Workbench.

Задание:
В соответствии со своим вариантом задания, выданным на предыдущем занятии привести базу данных к 3НФ. Реализовать схему базы данных в среде MySQL Workbench.

Краткие теоретические сведения:

Нормализация баз данных
Нормальные формы – это рекомендации по проектированию баз данных. Рекомендуется нормализовать базу данных в некоторой степени потому, что этот процесс имеет ряд существенных преимуществ с точки зрения эффективности и удобства обращения с базой данных.

- В нормализованной структуре базы данных можно производить сложные выборки данных относительно простыми SQL-запросами.
- Целостность данных. Нормализованная база данных позволяет надежно хранить данные.
- Нормализация предотвращает появление избыточности хранимых данных. Данные всегда хранятся только в одном месте, что делает легким процесс вставки, обновления и удаления данных. Есть исключение из этого правила. Ключи, сами по себе, хранятся в нескольких местах потому, что они копируются как внешние ключи в другие таблицы.
- Масштабируемость – это возможность системы справляться с будущим ростом. Для базы данных это значит, что она должна быть способна работать быстро, когда число пользователей и объемы данных возрастают. Масштабируемость – это очень важная характеристика любой модели базы данных и для РСУБД.

1 Первая нормальная форма (1НФ)
Первая нормальная форма гласит, что таблица базы данных – это представление сущности системы, которая создается. Примеры сущностей: заказы, клиенты, заказ билетов, отель, товар и т.д. Каждая запись в базе данных представляет один экземпляр сущности. Например, в таблице клиентов каждая запись представляет одного клиента.

Первичный ключ
Правило: каждая таблица имеет первичный ключ, состоящий из наименьшего возможного количества полей.

Первичный ключ может состоять из нескольких полей. К примеру, можно выбрать имя и фамилию в качестве первичного ключа (и надеяться, что эта комбинация будет уникальной всегда). Будет намного более хорошим выбором номер социального страхования в качестве

первичного ключа, т.к. это единственное поле, которое уникальным образом идентифицирует человека.

Еще лучше, когда нет очевидного кандидата на звание первичного ключа, создается суррогатный первичный ключ в виде числового автоинкрементного поля.

Атомарность

Правило: поля не имеют дубликатов в каждой записи и каждое поле содержит только одно значение.

Например, сайт коллекционеров автомобилей, на котором каждый коллекционер может зарегистрировать его автомобили. Таблица ниже хранит информацию о зарегистрированных автомобилях.

Таблица 1 - Горизонтальное дублирование данных – плохая практика.

id	first_name	last_name	car1	car2	car3	car4	car5
1	paul	johnson	mitsubishi	(NULL)	(NULL)	paul	johnson
2	frank	black	subaru imp	daihatsu	(NULL)	(NULL)	(NULL)
3	robert	smith	Mercedes S	Ferrari f	Maserati	Toyota Pr	Spyker C8
4	john	bonham	Mazda 626	(NULL)	(NULL)	(NULL)	(NULL)

С таким вариантом проектирования можно сохранить только пять автомобилей и если их менее 5, то тратится впустую свободное место в базе данных на хранение пустых ячеек.

Другим примером плохой практики при проектировании является хранение множественных значений в ячейке.

Таблица 2 - Множественные значения в одной ячейке.

id	first_name	last_name	cars
1	john	bonham	Mazda 626
2	robert	smith	Mercedes SL450,Ferrari f40, Maserati...
3	frank	black	subaru impreza, daihatsu cuore
4	paul	johnson	mitsubishi lancer

Верным решением в данном случае будет выделение автомобилей в отдельную таблицу и использование внешнего ключа, который ссылается на эту таблицу.

Порядок записей не должен иметь значение

Правило: порядок записей таблицы не должен иметь значения.

Разработчик может быть склонен использовать порядок записей в таблице клиентов для определения того, какой из клиентов зарегистрировался первым. Для этих целей лучше создать поля даты и времени регистрации клиентов. Порядок записей будет неизбежно меняться, когда клиенты будут удаляться, изменяться или добавляться. Вот почему никогда не следует полагаться на порядок записей в таблице.

2 Вторая нормальная форма

Для того, чтобы база данных была нормализована согласно второй нормальной форме, она должна быть нормализована согласно первой нормальной форме. Вторая нормальная форма связана с избыточностью данных.

Избыточность данных

Правило: поля с не первичным ключом не должны быть зависимы от первичного ключа.

Может звучать немного заумно. А означает это то, что можно хранить в таблице только данные, которые напрямую связаны с ней и не имеют отношения к другой сущности. Следование второй нормальной форме – это вопрос нахождения данных, которые часто дублируются в записях таблицы и которые могут принадлежать другой сущности.

Таблица 3 - Дублирование данных среди записей в поле store.

car_id	brand	type	color	store	price
1	Maserati	Quattropor	black	Amsterdam South	203000
2	Lada	1118	yellow	Amsterdam South	150
3	Volkswagen	Golf	green	Amsterdam North	14800
4	Volkswagen	Polo	black	Amsterdam West	10200
5	Jaguar	e type	green	The Hague	82399
6	Jaguar	e type	blue	The Hague	22374

Таблица выше может принадлежать компании, которая продает автомобили и имеет несколько магазинов в Нидерландах.

Если посмотреть на эту таблицу, то можно увидеть множественные примеры дублирования данных среди записей. Поле brand могло бы быть выделено в отдельную таблицу. Также, как и поле type (модель), которое также могло бы быть выделено в отдельную таблицу, которая бы имела связь многие-к-одному с таблицей brand потому, что у бренда могут быть разные модели.

Колонка store содержит наименование магазина, в котором в настоящее время находится машина. Store – это очевидный пример избыточности данных и хороший кандидат для отдельной сущности, которая должна быть связана с таблицей автомобилей связью по внешнему ключу.

Ниже пример того, как бы можно смоделировать базу данных для автомобилей, избегая избыточности данных.

car_id	type	color	store	price
1	5	black	1	203000
2	2	yellow	1	150
3	3	green	3	14800
4	4	black	2	10200
5	1	green	4	82399
6	1	blue	4	22374

type_id	brand_id	name
1	3	e type
2	2	1118
3	4	Golf
4	4	Polo
5	1	Quattroporte s

brand_id	name	country_of_origin
1	Maserati	Italy
2	Lada	Russia
3	Jaguar	United Kingdom
4	Volkswagen	Germany

store_id	name	street	hou...	zip...	phone
1	Amsterdam South	Churchil	14	1079HA	020373
2	Amsterdam West	Mercator	27	1056 RT	020838
3	Amsterdam North	Buikslot	76	1031 AB	020387
4	The Hague	Neherstr	82	2491JJ	070387

В примере выше таблица car имеет внешний ключ – ссылку на таблицы type и store. Столбец brand исчез потому, что на бренд есть неявная ссылка через таблицу type. Когда есть ссылка на type, есть ссылка и на brand, т.к. type принадлежит brand.

Избыточность данных была существенным образом устранена из модели базы данных. Но это не окончательный вариант. В поле country_of_origin в таблице brand пока дубликатов нет потому, что есть только четыре бренда из разных стран. Внимательный разработчик базы данных должен выделить названия стран в отдельную таблицу country.

И даже сейчас нельзя удовлетвориться результатом потому, что могжно бы выделить поле color в отдельную таблицу.

Если планируется хранить огромное количество единиц автомобилей в системе, и разработчик хочет иметь возможность производить поиск по цвету (color), то было бы мудрым решением выделить цвета в отдельную таблицу так, чтобы они не дублировались.

Существует другой случай, когда можно захотеть выделить цвета в отдельную таблицу. Если необходимо позволить работникам компании вносить данные о новых автомобилях, чтобы они имели возможно выбирать цвет машины из заранее заданного списка. В этом случае нужно хранить все возможные цвета в базе данных. Даже если еще нет машин с таким цветом, чтобы эти цвета присутствовали в базе данных, и работники могли их выбирать. Это определенно тот случай, когда нужно выделить цвета в отдельную таблицу.

3 Третья нормальная форма

Третья нормальная форма связана с транзитивными зависимостями. Транзитивные зависимости между полями базы данных существуют тогда, когда значения не ключевых полей зависят от значений других не ключевых полей. Чтобы база данных была в третьей нормальной форме, она должна быть во второй нормальной форме.

Транзитивные зависимости

Правило: не может быть транзитивных зависимостей между полями в таблице.

Таблица клиентов (мои клиенты – игроки немецкой и французской футбольной команды) ниже содержит транзитивные зависимости.

client_id	first_name	last_name	province	city	postal_code
23	Khalid	Boulahrouz	Noord-Holland	Alkmaar	1825HH
24	ZinÉdine	Zidane	Noord-Holland	Langedijk	1834DK
25	Ruud	van Nistelrooy	Noord-Holland	Schermer	1844JJ
19	Phillip	Cocu	Noord-Holland	Heilo	1850WI

В этой таблице не все поля зависят исключительно от первичного ключа. Существует отдельная связь между полем postal_code и полями города (city) и провинции (province). В Нидерландах оба значение: город и провинция – определяются почтовым кодом, индексом. Таким образом, нет необходимости хранить город и провинцию в клиентской таблице. Если знать почтовый код, то можно знать город и провинцию.

Такую транзитивную зависимость следует избегать, чтобы модель базы данных была в третьей нормальной форме.

В данном случае устранение транзитивной зависимости из таблицы может быть достигнуто путем удаления полей города и провинции из таблицы и хранение их в отдельной таблице, содержащей почтовый код (первичный ключ), имя провинции и имя города.

Получение комбинации почтовый код-город-провинция для целой страны может быть весьма нетривиальным занятием.

Другим примером для применения третьей нормальной формы может служить (слишком) простой пример таблицы заказов интернет-магазина ниже.

order_number	total_ex_vat	total_inc_vat
23	13,44	16,00
24	34,70	41,30
25	543,44	645,00
19	34,50	41,06

НДС – это процент, который добавляется к цене продукта (19% в данной таблице). Это означает, что значение total_ex_vat может быть вычислено из значения total_inc_vat и vice versa. Вы должны хранить в таблице одно из этих значений, но не оба сразу. Вы должны возложить задачу вычисления total_inc_vat из total_ex_vat или наоборот на программу, которая использует базу данных.

Третья нормальная форма гласит, что нельзя хранить данные в таблице, которые могут быть получены из других (не ключевых) полей таблицы.

Порядок выполнения работы:

Используя метод нормальных форм спроектировать базу данных. Процесс проектирования должен быть представлен в форме отчета, показан процесс формирования таблиц под выделенные сущности и их последовательная нормализация методом нормальных форм.

Форма представления результата:

- представление отчета на образовательном портале (в соответствующем курсе);
- обсуждение составленного отчета, оценка.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.2 Разработка и администрирование базы данных

Лабораторная работа № 4
Установка и настройка SQL-сервера

Цель: получение практических навыков по освоению операций установки и настройки SQL-сервера.

Выполнив работу, Вы будете:
уметь:
- устанавливать Microsoft SQL Server 2017 Express;
- настраивать Microsoft SQL Server 2017 Express.

Материальное обеспечение:
Методические указания для выполнения лабораторных работ, компьютер, программное обеспечение: MS SQLServer 2017 Express.

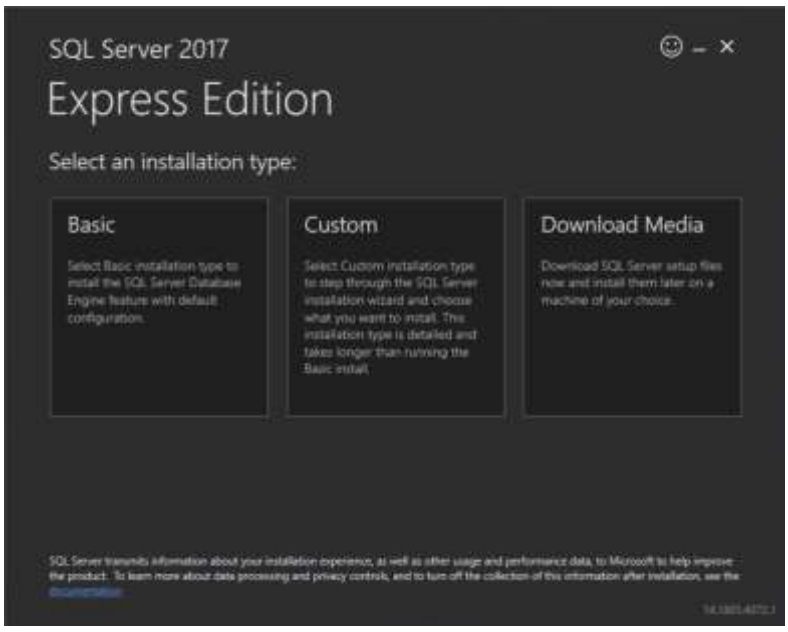
Задание:
Установить и настроить MS SQL Server Express.

Порядок выполнения работы:

Ограничения выпуска SQL Server 2017 Express
Данный выпуск является бесплатным и подходит для коммерческого использования, но имеет ряд ограничений:
- Максимальное количество ядер процессора: 4;
- Максимальный размер базы данных: 10 ГБ;

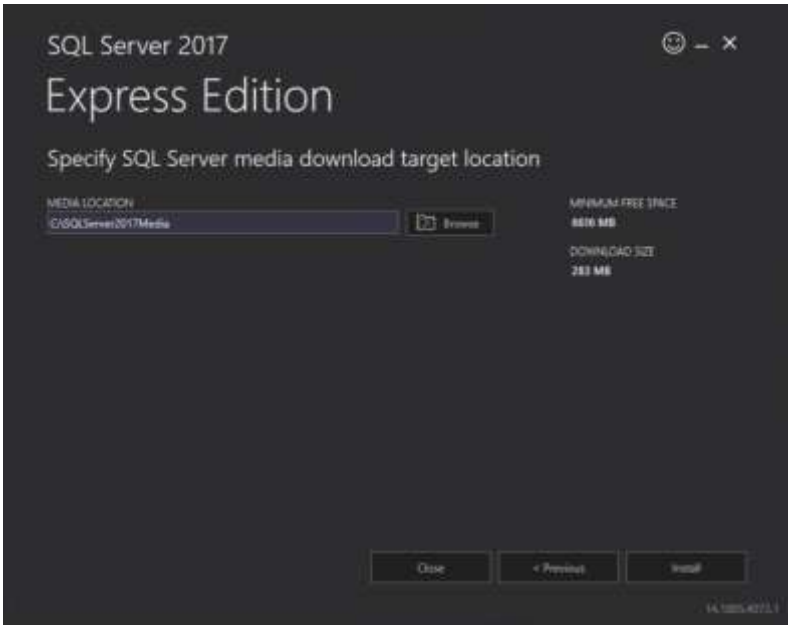
Требования к операционной системе
SQL Server 2017 Express доступен для установки на следующих операционных системах:
- Windows 8/8.1/10 и новее;
- Windows Server 2012, 2012 R2, 2016, 2019 и новее.

Процедура установки
Чтобы установить SQL Server 2017 Express, перейдите на официальный сайт, выберите язык установки и нажмите **Download**:

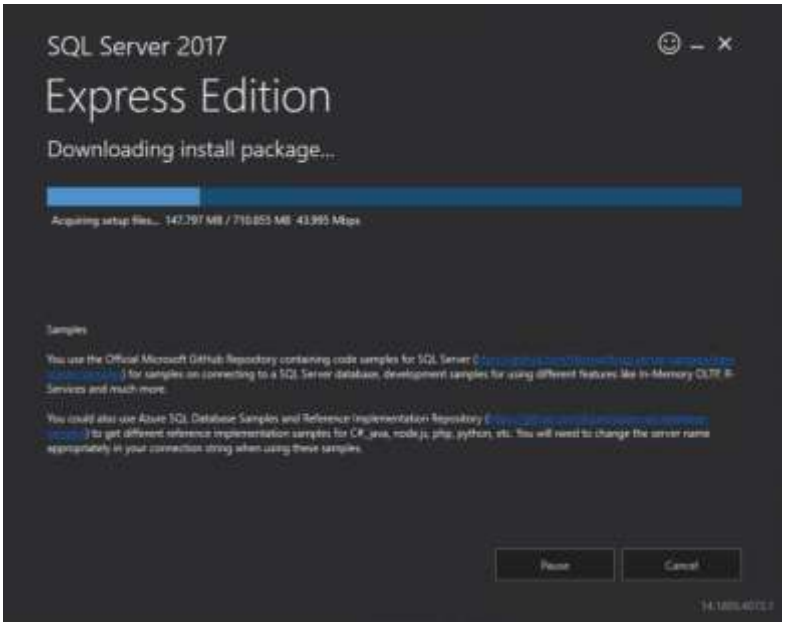


После запуска скачанного файла установщик откроет специальное окно, в котором предложит несколько вариантов установки на выбор. Для контроля за параметрами установки выберите **Custom**:

Далее выберите место, куда будут скачаны установочные файлы, при необходимости поменяйте его на нужное и нажмите **Install:**

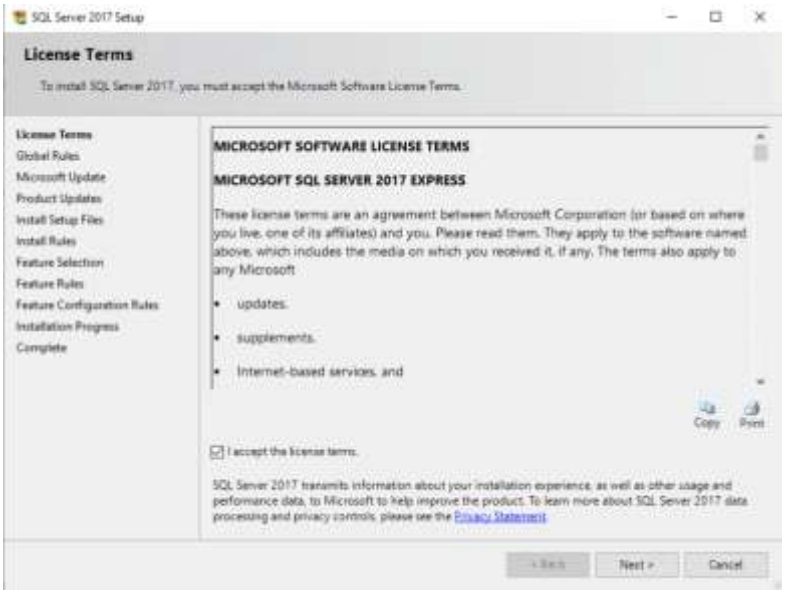


Дождитесь завершения процесса скачивания установочных файлов:

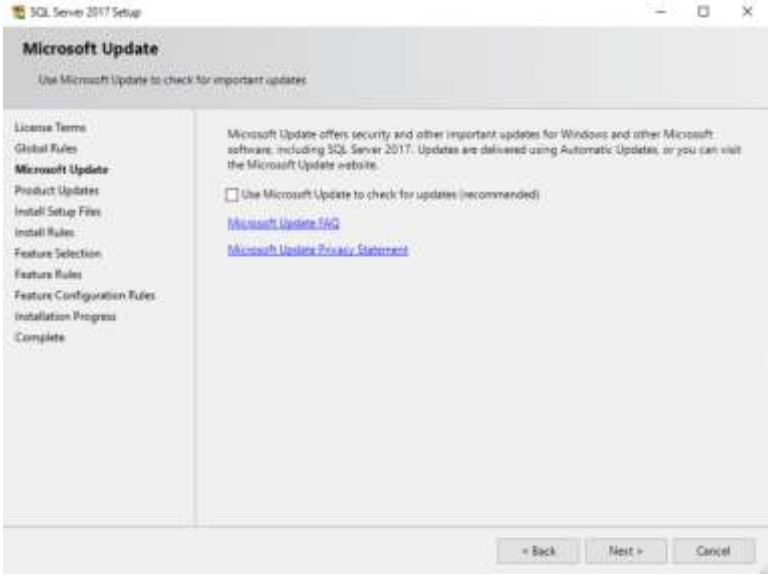


Далее выберите первый пункт установки **New SQL Server stand-alone installation**:

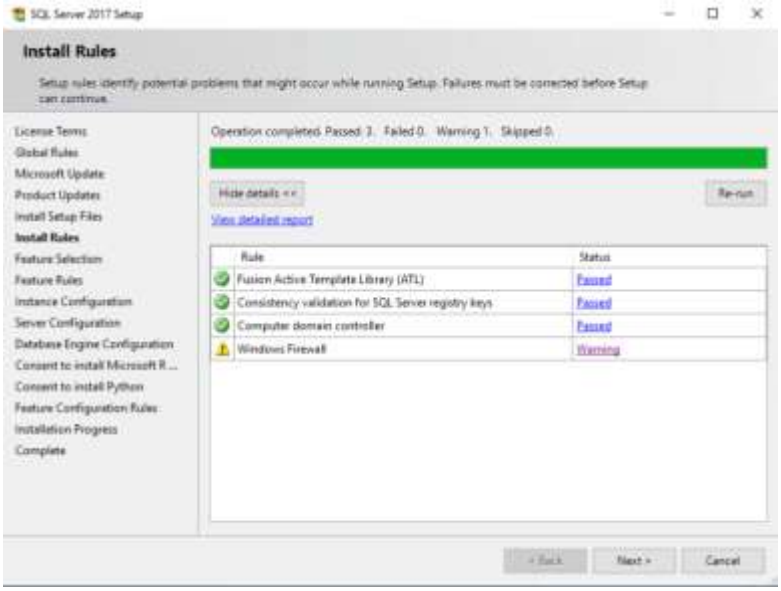
Ознакомьтесь с условиями лицензионного соглашения и нажмите **Next**:



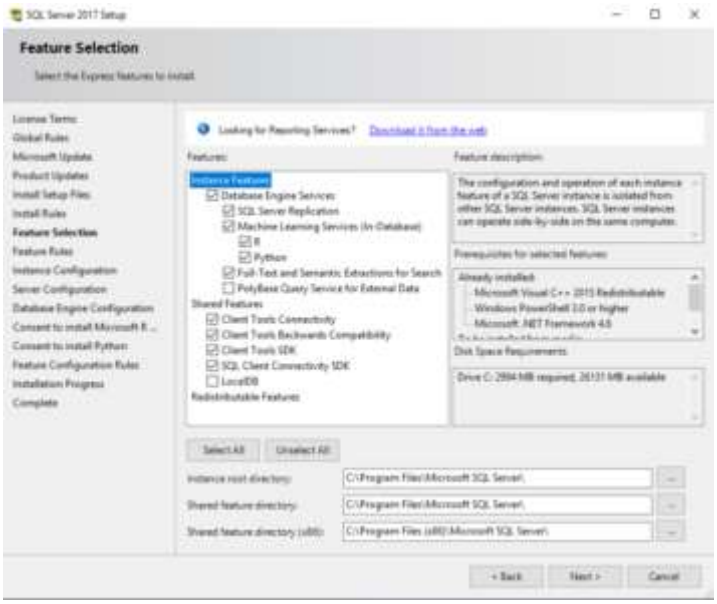
При необходимости получать обновления из Windows Update поставьте галочку и нажмите **Next**:

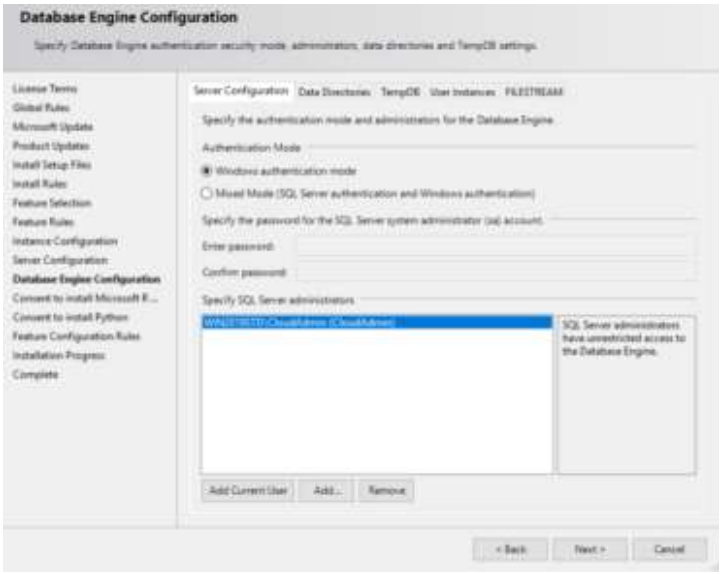


Ознакомьтесь с предупреждением о возможных проблемах, а при их отсутствии нажмите **Next:**

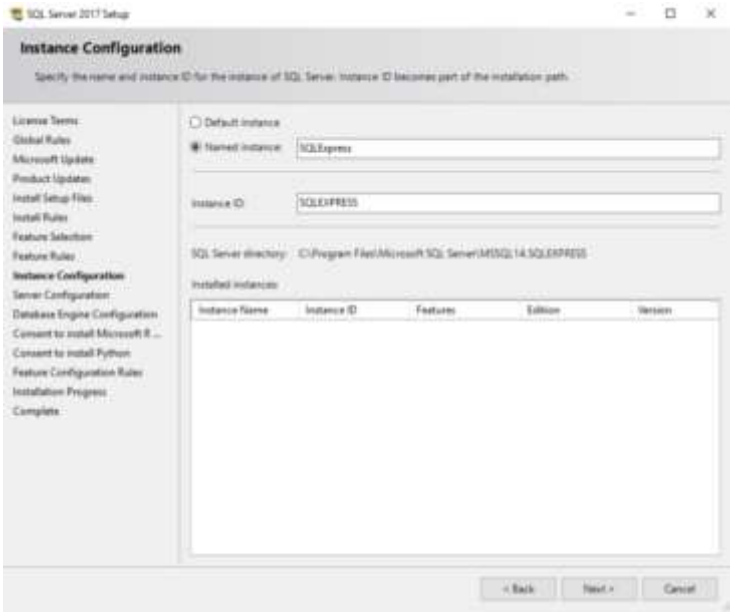


Перейдите на экран выбора функционала, где можно, при необходимости, выбрать или убрать дополнительные возможности для сервера баз данных, оставьте без изменений и нажмите **Next**:

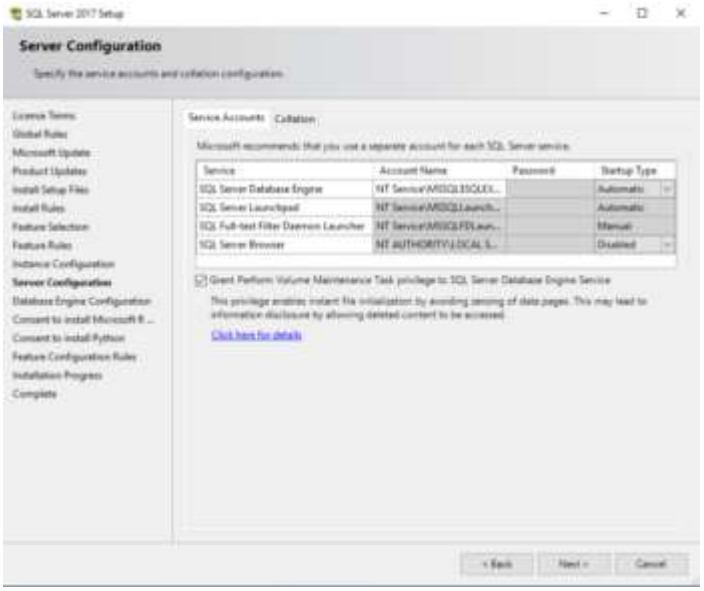




Выберите имя и идентификатор сервера. Идентификатор сервера будет включен в путь установки. Оставьте по умолчанию и нажмите **Next**:

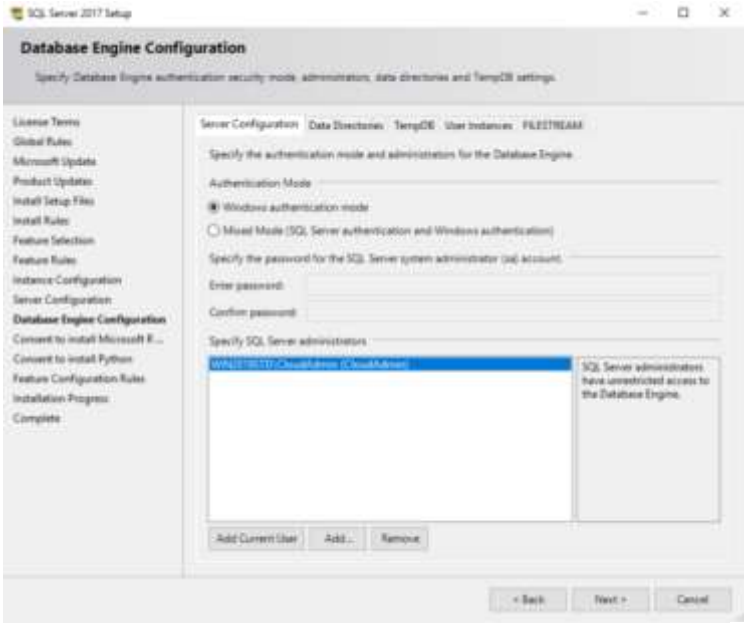


На следующем экране можно указать сервисные аккаунты, отличные от стандартных, и предоставить право на выполнение задач обслуживания тома службе ядра СУБД SQL Server, что повысит скорость инициализации файлов, но СУБД может получить доступ к удаленному контенту. На вкладке **Collation** можно изменить параметры сортировки движка базы данных. На указанном примере мы предоставим привилегии, оставим по умолчанию параметры сортировки и нажмем **Next**:

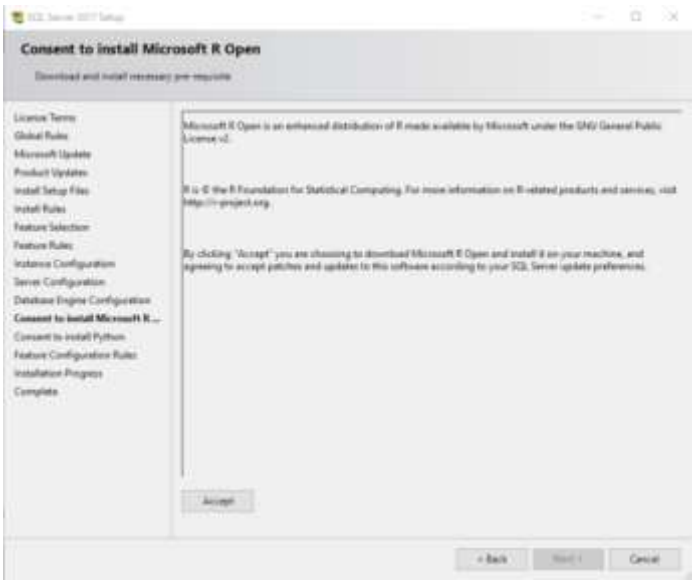


На следующем этапе установки необходимо настроить конфигурацию ядра базы данных. Для этого предусмотрены следующие вкладки:

- **Server Configuration** — указывается способ авторизации в базу данных: средствами Windows или смешанный режим, включающий в себя авторизацию Windows и собственную авторизацию SQL Server. При выборе второго варианта следует указать пароль администратора SQL Server;
- **Data Directories** — указывается расположение исполняемых файлов SQL Server и данных;
- **TempDB** — параметры TempDB, используемой внутренними ресурсами SQL Server, временными объектами пользователей и хранилищем версий;
- **User instances** — позволяет дать права пользователям, не имеющим прав администратора, запускать отдельные экземпляры баз данных;
- **FILESTREAM** — включается при необходимости использовать оптимизированные для памяти (Memory Optimized) таблицы.

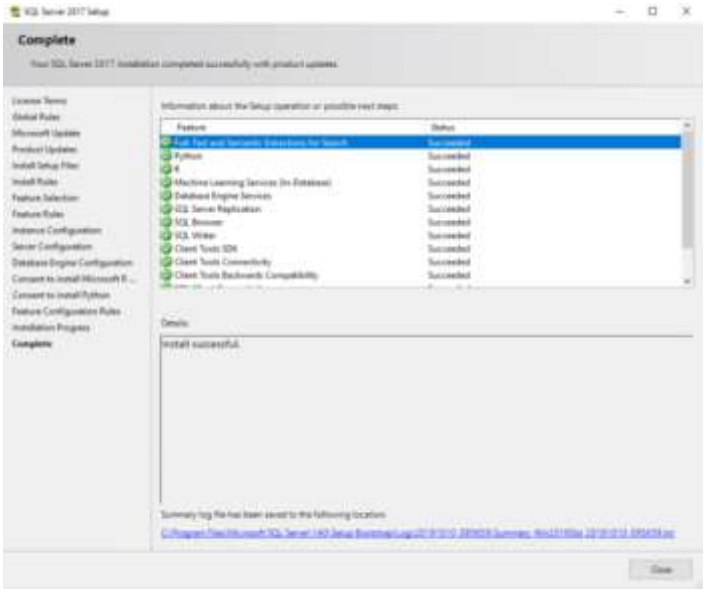


Так как при установке по умолчанию был выбран пункт Microsoft R (Machine Learning Services) и Python, следует согласиться с условиями его использования на этом и следующем этапе, последовательно нажав **Accept** и **Next**:



Запустится процесс установки, после чего появится окно о завершении

работ установщика SQL Server 2017. Нажмите **Close**:



На этом установка SQL Server 2017 Express завершена.

Форма представления результата:
Отчет по выполненной лабораторной работе

Критерии оценки:
Оценка «отлично» ставится, если задание выполнено верно.
Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.
Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.
Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.2 Разработка и администрирование базы данных

Лабораторная работа № 5
Создание базы данных в среде разработки

Цель: 1. получение практических навыков по освоению операций создания таблиц; 2. Получение практических навыков по освоению операций подстановок.

Выполнив работу, Вы будете:

уметь:

- устанавливать связи между таблицами;
- обеспечивать целостность данных;
- создавать поля со списками;
- создавать объекты баз данных в современных системах управления базами данных и управлять доступом к этим объектам.

Материальное обеспечение:

Методические указания для выполнения лабораторных работ, вариант задания, компьютер, программное обеспечение: MS SQLServer, MySQL Workbench.

Задание:

Создать базу данных по учету успеваемости студентов в СУБД MS SQLServer.

Краткие теоретические сведения:

Создание базы данных в среде MS SQL Server

Процесс создания *базы данных* в системе SQL-сервера состоит из двух этапов: сначала организуется сама *база данных*, а затем принадлежащий ей *журнал транзакций*. Информация размещается в соответствующих файлах, имеющих расширения *.mdf (для *базы данных*) и *.ldf. (для *журнала транзакций*). В файле *базы данных* записываются сведения об основных объектах (*таблицах*, *индексах*, просмотрах и т.д.), а в файле *журнала транзакций* – о процессе работы с транзакциями (контроль целостности данных, состояния *базы данных* до и после выполнения транзакций).

Создание *базы данных* в системе SQL-сервер осуществляется командой CREATE DATABASE. Следует отметить, что процедура создания *базы данных* в SQL-сервере требует наличия прав администратора сервера.

```
<определение_базы_данных> ::=
   CREATE DATABASE имя_базы_данных
   [ON [PRIMARY]
   [ <определение_файла>  [,...n] ]
   [,<определение_группы> [,...n] ] ]
   [ LOG ON {<определение_файла>[,...n] } ]
   [ FOR LOAD | FOR ATTACH ]
```

При выборе имени *базы данных* следует руководствоваться общими правилами именования объектов. Если имя *базы данных* содержит пробелы или любые другие недопустимые символы, оно заключается в ограничители (двойные кавычки или квадратные скобки). Имя *базы данных* должно быть уникальным в пределах сервера и не может превышать 128 символов.

При создании и изменении *базы данных* можно указать имя файла, который будет для нее создан, изменить имя, путь и исходный размер этого файла. Если в процессе использования *базы данных* планируется ее размещение на нескольких дисках, то можно создать так называемые *вторичные файлы базы данных* с расширением *.ndf. В этом случае

основная информация о *базе данных* располагается в *первичном* (PRIMARY) файле, а при нехватке для него свободного места добавляемая информация будет размещаться во *вторичном файле*. Подход, используемый в SQL-сервере, позволяет распределять содержимое *базы данных* по нескольким дисковым томам.

Параметр ON определяет список файлов на диске для размещения информации, хранящейся в *базе данных*.

Параметр PRIMARY определяет *первичный файл*. Если он опущен, то *первичным* является первый файл в списке.

Параметр LOG ON определяет список файлов на диске для размещения *журнала транзакций*. Имя файла для *журнала транзакций* генерируется на основе имени *базы данных*, и в конце к нему добавляются символы _log.

При создании *базы данных* можно определить набор файлов, из которых она будет состоять. Файл определяется с помощью следующей конструкции:

```
<определение_файла>::=
    ([ NAME=логическое_имя_файла,]
   FILENAME='физическое_имя_файла'
   [,SIZE=размер_файла ]
   [,MAXSIZE={max_размер_файла |UNLIMITED } ]
   [, FILEGROWTH=величина_прироста ] )[,...n]
```

Здесь ***логическое имя файла*** – это имя файла, под которым он будет опознаваться при выполнении различных SQL-команд.

Физическое имя файла предназначено для указания полного пути и названия соответствующего физического файла, который будет создан на жестком диске. Это имя останется за файлом на уровне операционной системы.

Параметр SIZE определяет первоначальный размер файла; минимальный размер параметра – 512 Кб, если он не указан, по умолчанию принимается 1 Мб.

Параметр MAXSIZE определяет максимальный размер файла *базы данных*. При значении параметра UNLIMITED максимальный размер *базы данных* ограничивается свободным местом на диске.

При создании *базы данных* можно разрешить или запретить автоматический рост ее размера (это определяется параметром FILEGROWTH) и указать приращение с помощью абсолютной величины в Мб или процентным соотношением.

Дополнительные файлы могут быть включены в группу:

```
<определение_группы>::=FILEGROUP имя_группы_файлов
<определение_файла>[,...n]
```

Порядок выполнения работы:

Создание любой *базы данных* начинается с создания файла данных. Рассмотрим этот процесс на примере создания простой *базы данных* по учету успеваемости студентов.

Для начала необходимо запустить среду разработки "*SQL Server Managemen* Studio". Для этого выбираем *пункт* **"Все программы\Microsoft SQL Server 2008\SQL Server Management Studio"**

После запуска среды разработки появится окно подключения к серверу **"Соединение с сервером"** (*Connect to Server*).

В этом окне необходимо нажать кнопку **"Соединить"** (*Connect*)

Создание файла данных.

Для этого в обозревателе объектов щелкните **ПКМ** на папке **"Базы данных"** (*Databases*) и в появившемся меню выберите пункт **"Создать базу данных"** (*New Database*). Появится окно настроек параметров файла данных новой базы данных **"Создать базу данных"**. В левой части окна настроек имеется список **"Выбор страницы"** (*Select a page*). Этот список позволяет переключаться между группами настроек.

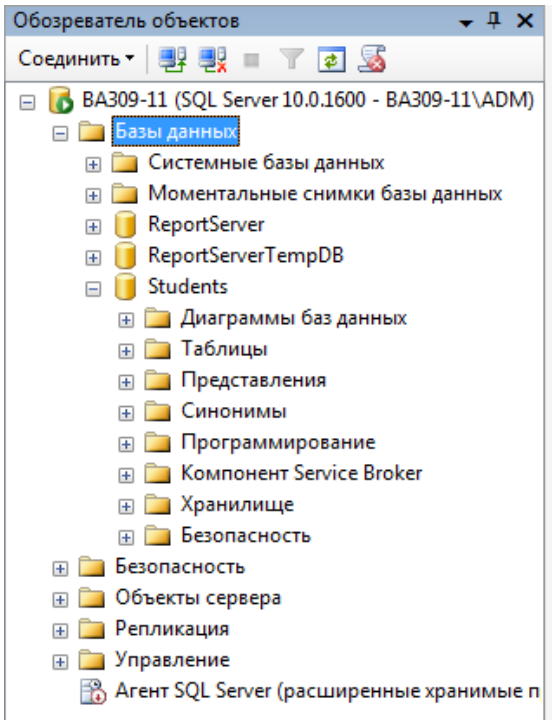
Настроим основные настройки **"Общие"**. Для выбора основных настроек нужно просто щелкнуть мышью по пункту **" Общие "** в списке **"Выбор страницы"**. В правой части окна **" Создать базу данных "** появятся основные настройки.

В верхней части окна расположено два параметра: **"Имя базы данных"** и **"Владелец"**. Задайте параметр **"Имя базы данных"** равным **"Students"**. Параметр **"Владелец"** оставьте без изменений.

В нашем случае мы оставим все основные настройки без изменений.

Для принятия всех настроек и создание файла данных и журнала транзакций нашей базы данных в окне **"Создание базы данных"** нажмем кнопку **"Ok"**.

Произойдет возврат в окно среды разработки **"SQL Server Management Studio"**. На панели обозревателя объектов в папке **"Базы данных"** появится новая база данных **"Students"**.



Для переименования БД необходимо в обозревателе объектов щелкнуть по ней **ПКМ** и в появившемся меню выбрать пункт **"Rename"**. Для удаления в это же меню выбираем пункт **"Delete"**, для обновления – пункт **"Refresh"**, а для изменения свойств, описанных выше – пункт **"Properties"**.

Все таблицы нашей базы данных находятся в подпапке **"Таблицы"** папки **"Students"** в окне обозревателя объектов.

Создание таблиц.

Создадим таблицу **"Специальности"**. Для этого щелкните **ПКМ** по папке **" Таблицы "** и в появившемся меню выберите пунк **"Создать таблицу…"**. Появится окно создания новой таблицы.

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

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

Под таблицей определения полей располагается таблица свойств выделенного поля **"Column Properties"**. В данной таблице настраиваются свойства выделенного поля.

Перейдем к созданию полей и настройке их свойств. В таблице определения полей задайте значения столбцов **"Column Name"**, **"Data Type"** и **"Allow Nulls"**.

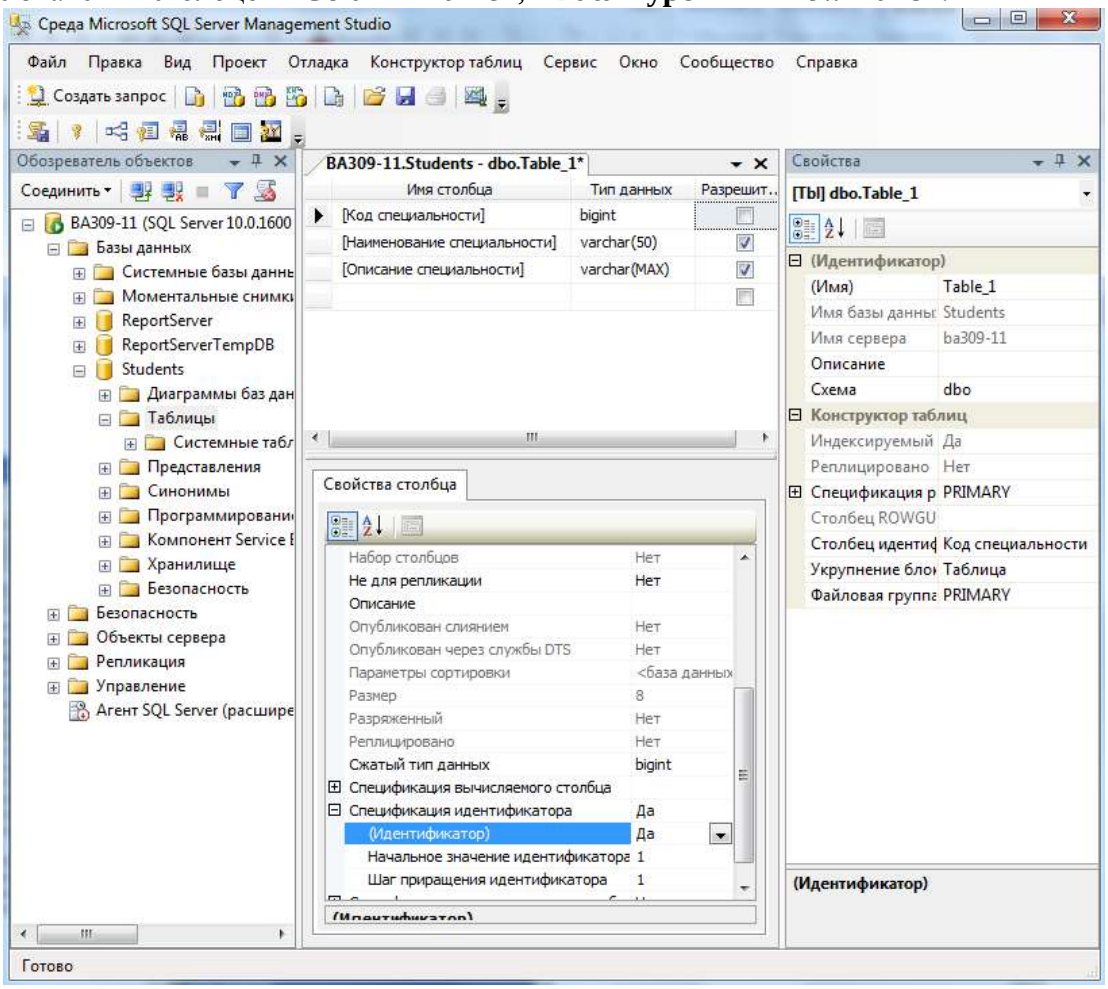


Таблица **"Специальности"** имеет три поля:

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

Так как, поле **"Код специальности"** будет являться первичным полем связи в запросе, связывающем таблицы **"Студенты"** и **"Специальности"**. То мы должны сделать его числовым счетчиком. То есть данное поле должно автоматически заполняться числовыми значениями. Более того, оно должно быть ключевым.

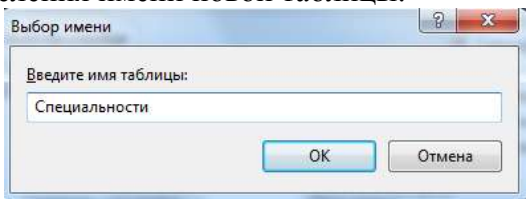
Сделаем поле **"Код специальности"** счетчиком. Для этого выделите поле, просто щелкнув по нему мышкой в таблице определения полей. В таблице свойств поля отобразятся свойства поля **"Код специальности"**. Разверните группу свойств **"Identity Specification"** (Спецификация идентификатора). Свойство **"(Is Identity)"** (Идентификатор) установите в значение **"Yes"** (Да). Задайте свойства **"Identity Increment"** (Шаг приращения идентификатора) и **"Identity Seed"** (Начальное значение идентификатора) равными 1. Эти настройки показывают, что значение поля **"Код специальности"** у первой записи в таблице будет равным 1, у второй - 2, у третьей 3 и т.д.

Теперь сделаем поле **"Код специальности"** ключевым полем. Выделите поле, а затем на панели инструментов нажмите кнопку с изображением ключа.

В таблице определения полей, рядом с полем **"Код специальности"** появиться изображение ключа, говорящее о том, что поле ключевое.

На этом настройку таблицы **"Специальности"** можно считать завершенной. Закройте окно создания новой таблицы, нажав кнопку закрытия в верхнем правом углу окна, над таблицей определения полей. Появиться окно с запросом о сохранении таблицы.

В этом окне необходимо нажать **"Да"**. Появиться окно **"Выбор имени"**, предназначенное для определения имени новой таблицы.



В этом окне задайте имя новой таблицы как **"Специальности"** и нажмите кнопку **"Ok"**. Таблица **"Специальности"** отобразиться в обозревателе объектов в папке **"Таблицы"** базы данных **"Students"**.

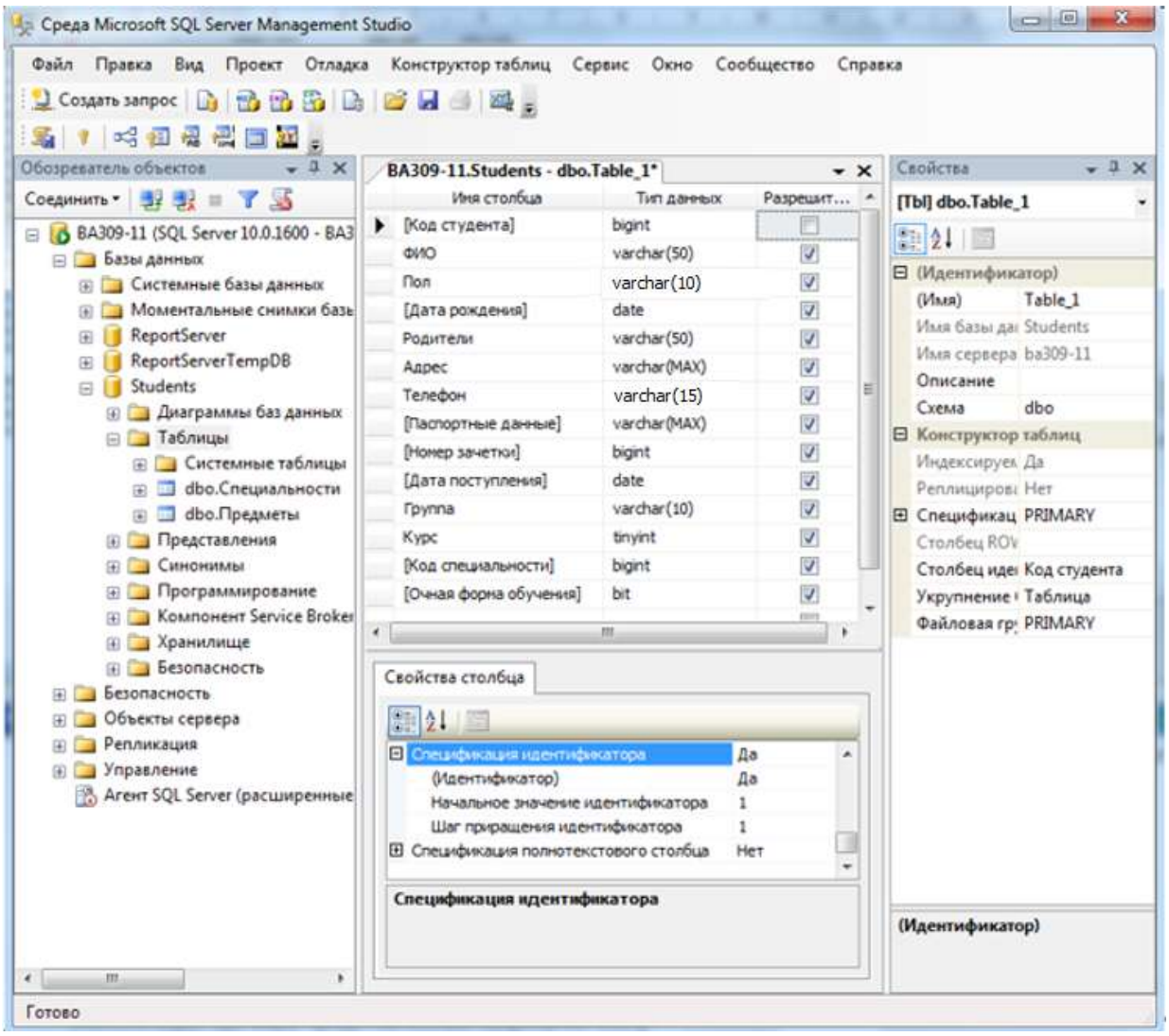
В обозревателе объектов таблица **"Специальности"** отображается как **"dbo.Специальности"**. Префикс "dbo" обозначает, что таблица является объектом БД (Data Base Object). В дальнейшем при работе с объектами БД префикс "dbo" можно опускать.

Теперь перейдем к созданию таблицы **"Предметы"**. Аналогично таблице **"Специальности"** создайте поля.

Сделайте поле **"Код предмета"** числовым счетчиком и ключевым полем, как это было сделано в таблице **"Специальности"**. Закройте окно создания новой таблицы, задайте имя **"Предметы"**.

Таблица **"Предметы"** появится в папке **"Таблицы"** в обозревателе объектов.

После создания таблицы **"Предметы"** создайте таблицу **"Студенты"**. Создайте новую таблицу аналогичную таблице, представленной на рисунке.

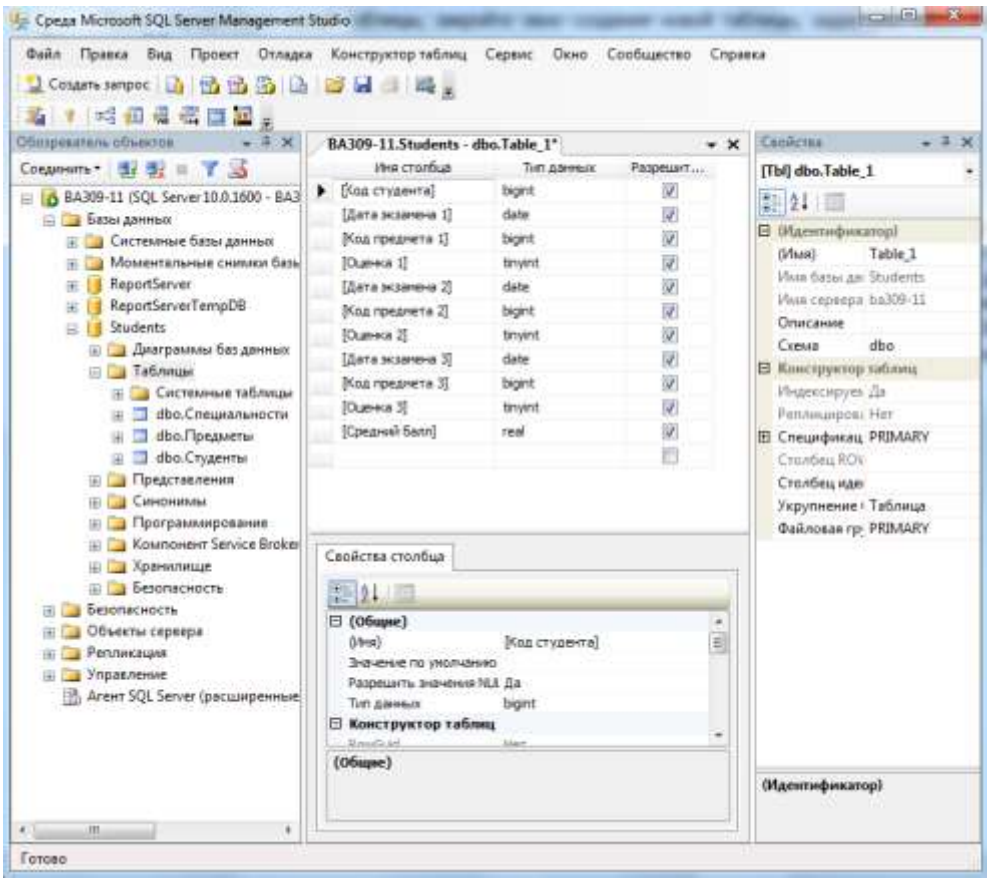


Рассматривая поля новой таблицы можно прийти к следующим выводам:

- Поле **"Код студента"** - это первичное поле для связи с таблицей оценки. Следовательно, данное поле необходимо сделать числовым счетчиком и ключевым;
- Поля **"ФИО"**, **"Пол"**, **"Родители"**, **"Адрес"**, **"Телефон"**, **"Паспортные данные"** и **"Группа"** являются текстовыми полями различной длины (для задания длины выделенного текстового поля необходимо в таблице свойств выделенного поля установить свойство **Length** равное максимальному количеству знаков текста, вводимого в поле);
- Поля **"Дата рождения"** и **"Дата поступления"** предназначены для хранения дат. Поэтому они имеют тип данных "date";
- Поле **"Очная форма обучения"** является логическим полем. В "Microsoft SQL Server 2008" такие поля должны иметь тип данных "bit";
- Поля **"Номер зачетки"** и **"Курс"** являются целочисленными. Единственным отличием является размер полей. Поле **"Номер зачетки"** предназначено для хранения целых чисел в диапазоне $-2^{63}…+2^{63}$ (тип данных "bigint"). Поле **"Курс "** предназначено для хранения целых чисел в диапазоне 0…255 (тип данных "tinyint");
- Поле **"Код специальности"** - это поле связи с таблицей **"Специальности"**. Однако, данное поле связи является вторичным, поэтому его можно сделать просто целочисленным, то есть, "bigint".

После определения полей таблицы, закройте окно создания новой таблицы, задав имя новой таблицы **"Студенты"**.

Таблица **"Студенты"** появится в папке **"Таблицы"** в обозревателе объектов.



Наконец, создадим таблицу **"Оценки".** Создайте поля, представленные на рисунке.

Таблица **"Оценки"** не имеет первичных полей связи. Следовательно, эта таблица не имеет ключевых полей. Поля **"Код предмета 1"**, **"Код предмета 2"** и **"Код предмета 3"** являются вторичными полями связи, предназначенными для связи с таблицей **"Предметы"**, поэтому они являются целочисленными *(*тип данных "bigint"). Поля **"Дата экзамена 1"**, **"Дата экзамена 2"** и **"Дата экзамена 3"** предназначены для хранения дат (тип данных "date"). Поля **"Оценка 1"**, **"Оценка 2"**, **Оценка 3"** предназначены для хранения оценок. Задайте тип данных для этого поля "tinyint". Наконец, поле **"Средний балл"** хранит дробные числа и имеет тип "real".

Закройте окно создания новой таблицы, задав имя таблицы как "Оценки".

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.2 Разработка и администрирование базы данных

Лабораторная работа № 6
Импорт данных пользователя в базу данных

Цель: получение практических навыков импортирования данных в базу данных

Выполнив работу, Вы будете:
уметь:
- создавать объекты базы данных;
- импортировать данные в базу данных.

Материальное обеспечение:
Методические указания для выполнения лабораторных работ, вариант задания, компьютер, программное обеспечение: MS SQL Server, MySQL Workbench .

Задание:
Выполнить импорт данных в базу данных.

Краткие теоретические сведения:
Если требуется добавить в таблицу большой массив данных, удобно использовать для этого команду загрузки данных из файла. Загрузка из файла выполняется программой MySQL значительно быстрее, чем вставка строк с помощью команды INSERT.

Например, чтобы загрузить данные в таблицу Customers, выполните следующие действия.

1. Запустите стандартную программу Windows Блокнот (Пуск → Все программы → Стандартные → Блокнот).

2. В окне программы Блокнот введите данные, используя для отделения значений друг от друга клавишу Tab, а для перехода на следующую строку – клавишу Enter.

Вместо отсутствующего значения необходимо при заполнении файла ввести символы «\N». Тогда в базу данных будет загружено неопределенное значение (NULL).

3. Для сохранения файла с данными нажмите комбинацию клавиш Ctrl+S. В стандартном окне Windows *Сохранить как* выберите папку, в которую нужно поместить файл (например, C: \data). Введите имя файла (например, Customers. txt) и нажмите кнопку Сохранить.

4. Для загрузки данных из созданного файла выполните команду

```
LOAD DATA LOCAL INFILE 'C:/data/Customers.txt'
INTO TABLE Customers
CHARACTER SET utf8;
```

Обратите внимание, что в пути к файлу необходимо использовать прямую косую черту, а не обратную.

Порядок выполнения работы:
1. Создайте базу данных на сервере MySQL.
2. Сценарий SQL предоставлен так, чтобы создать большинство таблиц и вставки данных в них. Все, что нужно сделать, это импортировать сценарий SQL в вашу базу данных. database.sql.
3. Таблица Сотрудники (персонал, должности) не включены в этот сценарий SQL. Обратитесь к диаграмме базы данных (ERD) и словарю данных.
4. Создайте таблицы сотрудников (персонал, положение и расписаний) согласно спецификации.

5. Все данные сотрудников представлены в файле employee-import.
6. Эти данные не отформатированы для импортирования непосредственно в базу данных, необходимо отформатировать данные и загрузить их в таблицы, которые вы только что создали.
7. В поле " Full Name" в формате "Имя Фамилия" используются разные символы разделителя.
8. Убедитесь, что адреса электронной почты в правильном формате

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.2 Разработка и администрирование базы данных

Лабораторная работа № 7
Экспорт данных базы в документы пользователя

Цель: получение практических навыков по экспорту данных базы данных в документы пользователя

Выполнив работу, Вы будете:

уметь:

- выполнять экспорт данных базы.

Материальное обеспечение:

Методические указания для выполнения лабораторных работ, вариант задания, компьютер, программное обеспечение: MS SQLServer.

Задание:

Выгрузить данные из всех таблиц базы данных своего варианта в документы.

Краткие теоретические сведения:

Чтобы результат запроса был сохранен в файл, необходимо добавить в команду SELECT выражение:

INTO OUTFILE 'Путь и имя файла'

В этой команде нужно указать полный путь к файлу, в который будут выгружены данные (этот файл должен быть новым, не существующим на момент выгрузки). При задании пути к файлу необходимо использовать прямую косую черту вместо принятой в Windows обратной косой черты. Указанный файл создается на компьютере, на котором работает сервер MySQL. Данные выгружаются в той кодировке, в которой они хранятся в базе данных.

Команды SELECT… INTO OUTFILE и LOAD DATA можно использовать для резервного копирования таблиц или для переноса данных на другой сервер MySQL.

Например, данные из таблицы Customers (Клиенты), сохраненные в файл с помощью команды SELECT

SELECT * from Customers INTO OUTFILE 'C:/data/Customers.txt';

можно загрузить в таблицу Customers_copy (имеющую такую же структуру, что и таблица Customers) с помощью команды

LOAD DATA INFILE 'C:/data/Customers.txt'
INTO TABLE Customers_copy;

Порядок выполнения работы:

Выгрузить данные из всех таблиц базы данных своего варианта в документы Word и Excel. Создать копии таблиц и выполнить загрузку данных из файла в базу данных.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.2 Разработка и администрирование базы данных

Лабораторная работа № 8
Создание представлений

Цель: получение практических навыков разработки представлений на языке SQL.

Выполнив работу, Вы будете:
уметь:
- создавать представления с помощью SQL.

Материальное обеспечение:
Методические указания для выполнения лабораторных работ, вариант задания, компьютер, программное обеспечение: MS SQLServer.

Задание:
Разработать представления для базы данных на языке SQL.

Краткие теоретические сведения:
Представления, или *просмотры* (VIEW), представляют собой временные, производные (иначе - виртуальные) таблицы и являются объектами базы данных, информация в которых не хранится постоянно, как в базовых таблицах, а формируется динамически при обращении к ним. Обычные таблицы относятся к базовым, т.е. содержащим данные и постоянно находящимся на устройстве хранения информации. *Представление* не может существовать само по себе, а определяется только в терминах одной или нескольких таблиц. Применение *представлений* позволяет разработчику базы данных обеспечить каждому пользователю или группе пользователей наиболее подходящие способы работы с данными, что решает проблему простоты их использования и безопасности. Содержимое *представлений* выбирается из других таблиц с помощью выполнения запроса, причем при изменении значений в таблицах данные в *представлении* автоматически меняются. *Представление* - это фактически тот же запрос, который выполняется всякий раз при участии в какой-либо команде. Результат выполнения этого запроса в каждый момент времени становится содержанием *представления*. У пользователя создается впечатление, что он работает с настоящей, реально существующей таблицей.
У СУБД есть две возможности *реализации представлений*. Если его определение простое, то система формирует каждую запись *представления* по мере необходимости, постепенно считывая исходные данные из базовых таблиц. В случае сложного определения СУБД приходится сначала выполнить такую операцию, как материализация *представления*, т.е. сохранить информацию, из которой состоит *представление*, во временной таблице. Затем система приступает к выполнению пользовательской команды и формированию ее результатов, после чего временная таблица удаляется.

Представление - это предопределенный запрос, хранящийся в базе данных, который выглядит подобно обычной таблице и не требует для своего хранения дисковой памяти. Для хранения *представления* используется только оперативная память. В отличие от других объектов базы данных *представление* не занимает дисковой памяти за исключением памяти, необходимой для хранения определения самого *представления*.
Создания и изменения *представлений* в стандарте языка и реализации в MySQL совпадают и представлены следующей командой:

```
{ CREATE| ALTER} VIEW имя_просмотра
[(имя_столбца [,...n])]
[WITH ENCRYPTION]
```

AS SELECT_оператор
[WITH CHECK OPTION]

Рассмотрим назначение основных параметров.

По умолчанию имена столбцов в *представлении* соответствуют именам столбцов в исходных таблицах. Явное указание имени столбца требуется для вычисляемых столбцов или при объединении нескольких таблиц, имеющих столбцы с одинаковыми именами. Имена столбцов перечисляются через запятую, в соответствии с порядком их следования в *представлении*.

Параметр WITH ENCRYPTION предписывает серверу шифровать SQL-код запроса, что гарантирует невозможность его несанкционированного просмотра и использования. Если при определении *представления* необходимо скрыть имена исходных таблиц и столбцов, а также алгоритм объединения данных, необходимо применить этот аргумент.

Параметр WITH CHECK OPTION предписывает серверу исполнять проверку изменений, производимых через *представление*, на соответствие критериям, определенным в операторе SELECT. Это означает, что не допускается выполнение изменений, которые приведут к исчезновению строки из *представления*. Такое случается, если для *представления* установлен горизонтальный фильтр и изменение данных приводит к несоответствию строки установленным фильтрам. Использование аргумента WITH CHECK OPTION гарантирует, что сделанные изменения будут отображены в *представлении*. Если пользователь пытается выполнить изменения, приводящие к исключению строки из *представления*, при заданном аргументе WITH CHECK OPTION сервер выдаст сообщение об ошибке и все изменения будут отклонены.

Порядок выполнения работы:

Варианты заданий по созданию представлений

Вариант	**Содержание запроса**
1	1. CD-диски с общим объемом файлов более 900 кбайт. 2. CD-диски с названием «Access», выпущенные с 2017-го по 2018-й гг. 3. Владельцы CD-дисков с прикладным программным обеспечением.
2	1. Студенты, имеющие оценку «2» по химии. 2. Студенты, имеющие пропуски занятий по математике. 3. Успеваемость студента Р. Л. Ершова по математике и физике.
3	1. Постояльцы, проживающие в гостиничных номерах 5 и 40. 2. Постояльцы, которые забронировали гостиничный номер категории «люкс» и проживали в нем в июле 2017г. 3. Постояльцы, проживающие в номерах со стоимостью места менее 1500 р.
4	1. Товары, полученные от поставщика – завода «Монолит». 2. Товары, проданные летом 2017г. в количестве от 1000 до 3000 шт. 3. Товары с наименованием «Миксер», поступившие до 01.02.2017.
5	1. Предприятия, имеющие форму собственности «ОАО». 2. Предприятия, выпускающие электротехнику, прибыль которых составляет от 500 тыс. до 1 млн $. 3. Предприятия, заплатившие налоги в IV квартале 2017г.
6	1. Поездки в Москву летом 2017 г. 2. Поездки пассажирского поезда № 5. 3. Поездки в Новосибирск поездов с количеством вагонов более 10.
7	1. Водители, имеющие оклад от 15000 до 17000 р. 2. Автобусы, работавшие на маршруте № 79 в сентябре 2017 г. 3. Водители, работающие на автобусах марки «Икарус» на маршрутах № 90 и 104.
8	1. Риэлторы, совершившие обмен трехкомнатных квартир в мае 2017г.

	2. Сделки, совершенные риэлторами в июне 2017г. 3. Список однокомнатных квартир с телефоном и балконом, общий метраж которых не менее 42 кв. м.
9	1. Рейсы, выполненные из Москвы в Париж весной 2017г. 2. Рейсы, выполненные самолетом Ту-154. 3. Рейсы в Минск с доходом более 800 тыс. р.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.2 Разработка и администрирование базы данных

Лабораторная работа № 9
Создание хранимых процедур

Цель: получение практических навыков разработки хранимых процедур на языке SQL.

Выполнив работу, Вы будете:
уметь:
- создавать хранимые процедуры без параметров;
- создавать хранимые процедуры с входными параметрами;
- создавать хранимые процедуры с выходными параметрами.

Материальное обеспечение:
Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MS SQLServer.

Задание 1:
Разработать хранимые процедуры для базы данных на языке SQL.

Краткие теоретические сведения:
Хранимые процедуры представляют собой группы связанных между собой операторов SQL, применение которых делает работу программиста более легкой и гибкой, поскольку выполнить *хранимую процедуру* часто оказывается гораздо проще, чем последовательность отдельных операторов SQL. Хранимые процедуры представляют собой набор команд, состоящий из одного или нескольких операторов SQL или функций и сохраняемый в базе данных в откомпилированном виде. *Выполнение* в базе данных *хранимых процедур* вместо отдельных операторов SQL дает пользователю следующие преимущества:

- необходимые операторы уже содержатся в базе данных;
- все они прошли этап *синтаксического анализа* и находятся в исполняемом формате; перед *выполнением хранимой процедуры* MySQL генерирует для нее *план исполнения*, выполняет ее оптимизацию и компиляцию;
- *хранимые процедуры* поддерживают *модульное программирование*, так как позволяют разбивать большие задачи на самостоятельные, более мелкие и удобные в управлении части;
- *хранимые процедуры* могут вызывать другие *хранимые процедуры* и функции;
- *хранимые процедуры* могут быть вызваны из прикладных программ других типов;
- как правило, *хранимые процедуры* выполняются быстрее, чем последовательность отдельных операторов;
- *хранимые процедуры* проще использовать: они могут состоять из десятков и сотен команд, но для их запуска достаточно указать всего лишь имя нужной *хранимой процедуры*. Это позволяет уменьшить размер запроса, посылаемого от клиента на сервер, а значит, и нагрузку на сеть.

Создание новой и изменение имеющейся хранимой процедуры осуществляется с помощью следующей команды:

```
<определение_процедуры>::=
{CREATE | ALTER } PROC[EDURE] имя_процедуры
    [;номер]
[{@имя_параметра тип_данных } [VARYING ]
   [=default][OUTPUT] ][,...n]
[WITH { RECOMPILE | ENCRYPTION | RECOMPILE,
```

```
   ENCRYPTION }]
[FOR REPLICATION]
AS
   sql_оператор [...n]
```

Для *выполнения хранимой процедуры* используется команда:

```
[[ EXEC [ UTE] имя_процедуры [;номер]
[[@имя_параметра=]{значение | @имя_переменной}
   [OUTPUT ]|[DEFAULT ]][,...n]
```

Если вызов *хранимой процедуры* не является единственной командой в пакете, то присутствие команды EXECUTE обязательно. Более того, эта команда требуется для вызова процедуры из тела другой процедуры или триггера.

Использование ключевого слова OUTPUT при вызове процедуры разрешается только для параметров, которые были объявлены при создании процедуры с ключевым словом OUTPUT.

Когда же при вызове процедуры для параметра указывается ключевое слово DEFAULT, то будет использовано значение по умолчанию. Естественно, указанное слово DEFAULT разрешается только для тех параметров, для которых определено значение по умолчанию.

Из синтаксиса команды EXECUTE видно, что имена параметров могут быть опущены при вызове процедуры. Однако в этом случае пользователь должен указывать значения для параметров в том же порядке, в каком они перечислялись при создании процедуры. Присвоить параметру значение по умолчанию, просто пропустив его при перечислении нельзя. Если же требуется опустить параметры, для которых определено значение по умолчанию, достаточно явного указания имен параметров при вызове хранимой процедуры. Более того, таким способом можно перечислять параметры и их значения в произвольном порядке.

Порядок выполнения работы:

Варианты заданий по созданию хранимых процедур с вычисляемым полем

Вариант	Имя таблицы	Вычисляемое поле
1	CD-диски	Объем CD-диска в мегабайтах
2	Успеваемость	Индивидуальный код студента, представляющий собой сумму шифра студента и шифра дисциплины
3	Проживание	Сумма оплаты за проживание постояльца гостиницы
4	Продажа	Доход от продажи товара
5	Предприятия	Прибыль предприятия, рассчитанная в евро
6	Расписание	Прибыль за поездку, рассчитанная в долларах
7	Парк	Длительность маршрута
8	Недвижимость	Стоимость 1 кв. м общей площади
9	Перевозки	Количество свободных мест на рейсе

Варианты заданий по созданию хранимых процедур с групповыми операциями

Вариант	Имя таблицы	Итоговый показатель для расчета
1	Владельцы	Количество CD-дисков по каждому виду программного обеспечения
2	Студенты	Количество пропусков занятий по каждой дисциплине
3	Номерной фонд	Количество проживающих в гостиничных номерах каждой категории
4	Товары	Количество проданного товара по каждому наименованию
5	Предприятия	Сумма уплаченных налогов каждым предприятием

6	Поезда	Количество пассажиров, перевезенных каждым поездом
7	Парк	Количество автобусов по каждому маршруту
8	Сделки	Количество сделок, совершенных каждым риэлтором
9	Рейсы	Количество пассажиров, перевезенных каждым самолетом

Варианты заданий по созданию хранимых процедур с параметрами

Вариант	Условие проецедуры с параметром
1	Названия CD-дисков, принадлежащие владельцу *N*
2	Отметки о пропусках занятий студентом *N*
3	Постояльцы, проживающие в гостиничном номере категории *N*
4	Поставки товара поставщиком *N*
5	Продукция, выпускаемая предприятием *N*
6	Расписание движения пассажирского поезда *N*
7	Водители, работающие на автобусах по маршруту *N*
8	Сделки, совершенные риэлтором *N*
9	Рейсы, выполненные самолетами модели *N*

Задание 2:

Разработать хранимые процедуры для базы данных Students в СУБД MS SQLServer.

Порядок выполнения работы:

Для работы с хранимыми процедурами в обозревателе объектов необходимо выделить папку **"Программирование/Хранимые процедуры"** базы данных **"Students"**.

Создадим процедуру, вычисляющую среднее трех чисел. Для создания новой хранимой процедуры щелкните **ПКМ** по папке **" Хранимые процедуры "** и в появившемся меню выберите пункт **"Создать хранимую процедуру"**. Появиться окно кода новой хранимой процедуры.

Хранимая процедура имеет следующую структуру:

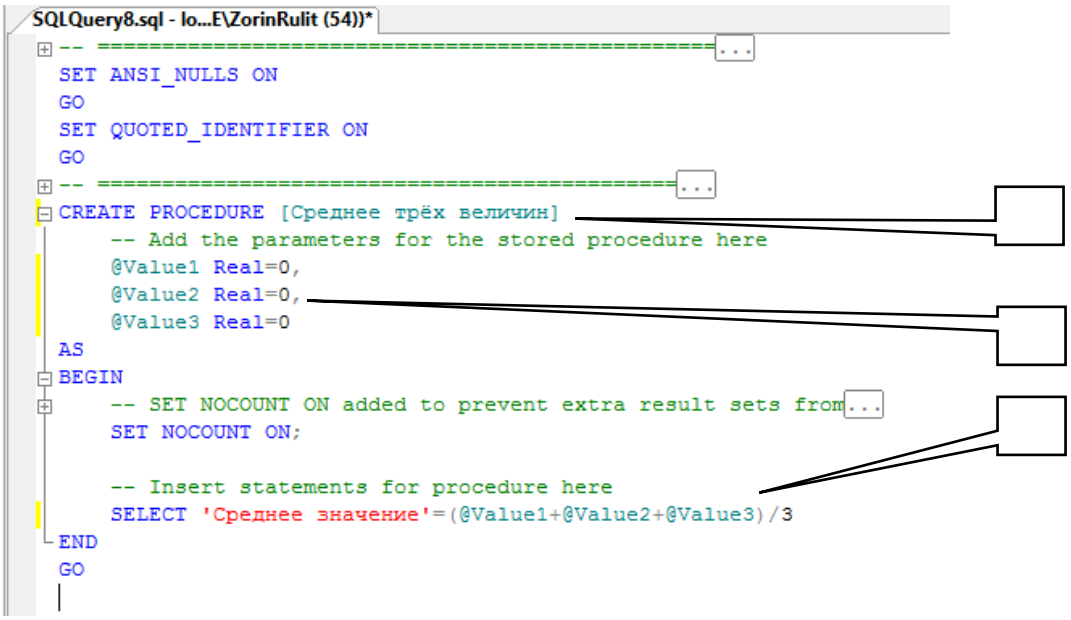
1. Область настройки параметров синтаксиса процедуры. Позволяет настраивать некоторые синтаксические правила, используемые при наборе кода процедуры. В нашем случае это:
   - SET ANSI_NULLS ON - включает использование значений NULL (Пусто) в кодировке ANSI,

- SET QUOTED_IDENTIFIER ON - включает возможность использования двойных кавычек для определения идентификаторов;

2. Область определения имени процедуры (**Procedure_Name**) и параметров, передаваемых в процедуру (**@Param1, @Param2**). Определение параметров имеет следующий синтаксис:
   @<Имя параметра> <Тип данных> = <Значение по умолчанию>
   Параметры разделяются между собой запятыми;
3. Начало тела процедуры, обозначается служебным словом "BEGIN";
4. Тело процедуры, содержит команды языка программирования запросов T-SQL;
5. Конец тела процедуры, обозначается служебным словом "END".

В коде зеленым цветом выделяются комментарии. Они не обрабатываются сервером и выполняют функцию пояснений к коду. Строки комментариев начинаются с подстроки "--". Далее в коде, мы не будем отображать комментарии, они будут свернуты. Слева от раздела с комментариями будет стоять знак "+", щелкнув по которому можно развернуть комментарий.

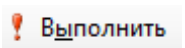
Наберем код процедуры, вычисляющей среднее трех чисел, как это показано на рисунке 4.2.



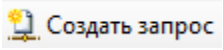
Рассмотрим код данной процедуры более подробно:

1. *CREATE PROCEDURE* [Среднее трех величин] - определяет имя создаваемой процедуры как "Среднее трех величин";
2. @Value1 Real = 0, @Value2 Real = 0, @Value3 Real = 0 - определяют три параметра процедуры Value1, Value2 и Value3. Данным параметрам можно присвоить дробные числа (Тип данных Real), значения по умолчанию равны 0;
3. SELECT 'Среднее значение'=(@Value1+@Value2+@Value3)/3 - вычисляет среднее и выводит результат с подписью "Среднее значение".

Остальные фрагменты кода рассмотрены ранее.

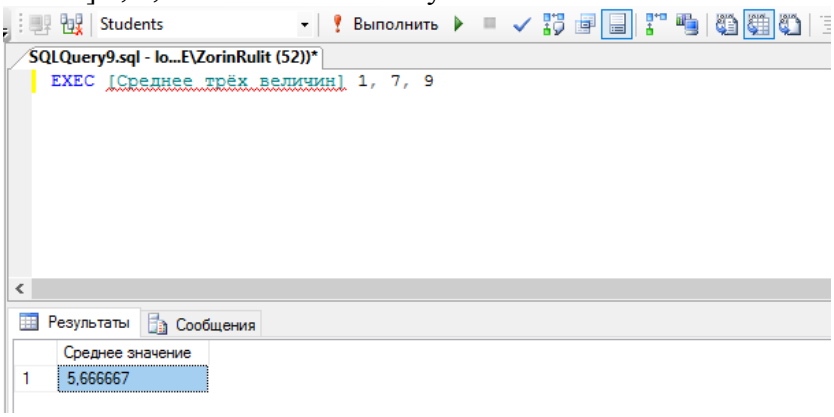
Для *создания процедуры*, выполним вышеописанный код, нажав кнопку 

на панели инструментов. В нижней части окна с кодом появиться сообщение **"Выполнение команд успешно завершено."**. Закройте окно с кодом, щелкнув мышью по кнопке закрытия.

Проверим работоспособность созданной хранимой процедуры. Для запуска хранимой процедуры необходимо создать новый пустой запрос, нажав на кнопку 

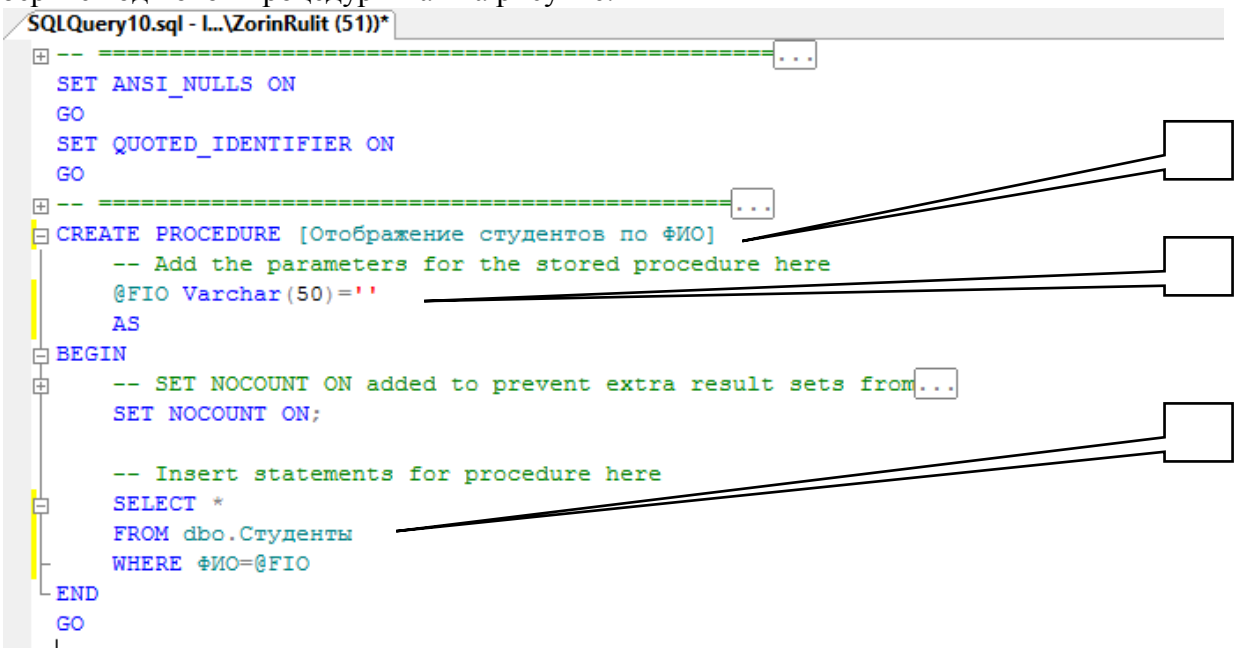
на

панели инструментов. В появившемся окне с пустым запросом наберите команду EXEC [Среднее трех величин] 1, 7, 9 и нажмите кнопку Выполнить на панели инструментов.



В нижней части окна с кодом появится результат выполнения новой хранимой процедуры: **Среднее значение 5,66667**.

Теперь создадим хранимую процедуру для отбора студентов из таблицы студенты по их "ФИО". Для этого создайте новую хранимую процедуру, как это описано выше, и наберите код новой процедуры как на рисунке.

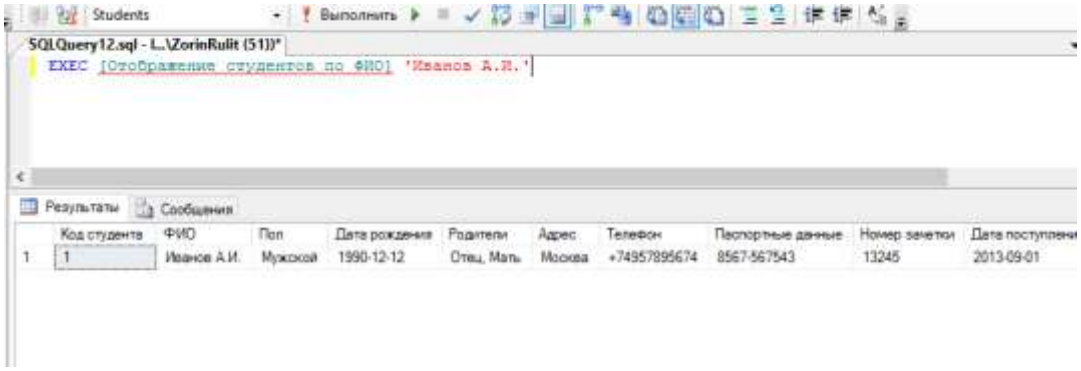


Рассмотрим код процедуры **"Отображение студентов по ФИО"** более подробно:

1. *CREATE PROCEDURE* [Отображение студентов по ФИО] - определяет имя создаваемой процедуры как "Отображение студентов по ФИО";
2. @FIO Varchar(50)='' - определяют единственный параметр процедуры **FIO**. Параметру можно присвоить текстовые строки переменной длины, длинной до 50 символов (Тип данных Varchar(50)), значения по умолчанию равны пустой строке;
3. SELECT * FROM dbo.Студенты WHERE ФИО=@FIO - отобразить все поля (*) из таблицы студенты (dbo.Студенты), где значение поля ФИО равно значению параметра **FIO (ФИО=@FIO)**.

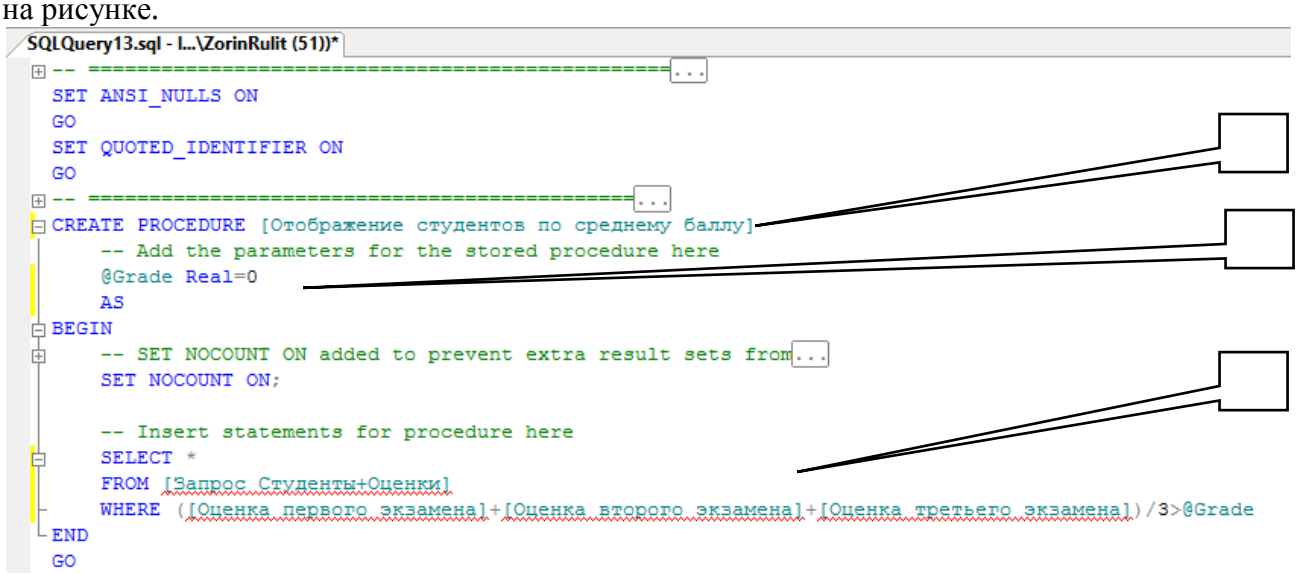
Выполним вышеописанный код.

Проверим работоспособность созданной хранимой процедуры. Создайте новый пустой запрос. В появившемся окне с пустым запросом наберите команду EXEC [Отображение студентов по ФИО] 'Иванов А.И.' и нажмите кнопку Выполнить на панели инструментов.



В нижней части окна с кодом появиться результат выполнения хранимой процедуры **"Отображение студентов по ФИО"**.

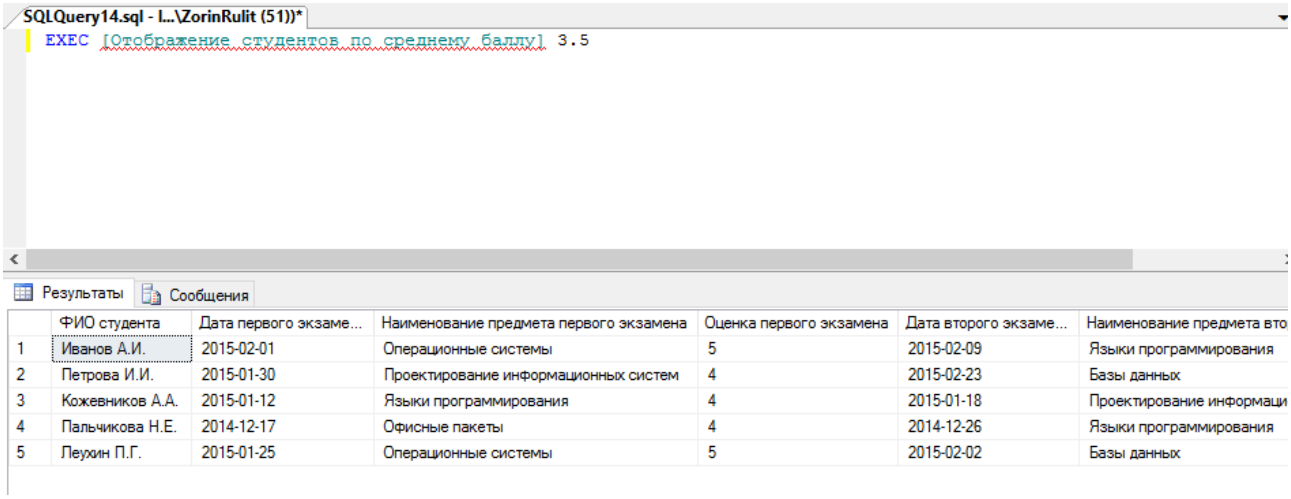
Теперь перейдем к следующей задаче - отобразить студентов, у которых средний балл выше заданного. Создайте новую хранимую процедуру и наберите код новой процедуры как на рисунке.



Рассмотрим код процедуры **"Отображение студентов по среднему баллу"** более подробно:

1. CREATE PROCEDURE [Отображение студентов по среднему баллу] - определяет имя создаваемой процедуры как "Отображение студентов по среднему баллу";
2. *@Grade* Real=0 - определяют параметр процедуры **Grade**. Параметру можно присвоить дробные числа (Тип данных Real), значения по умолчанию равны 0;
3. SELECT * FROM [Запрос Студенты+Оценки] WHERE ([Оценка первого экзамена]+[Оценка второго экзамена]+[Оценка третьего экзамена])/3>*@Grade* - отобразить все поля (*) из запроса **"Запрос Студенты+Оценки"** (Запрос Студенты+Оценки), где средний балл больше чем значение параметра **Grade (([Оценка первого экзамена]+[Оценка второго экзамена]+[Оценка третьего экзамена])/3>@Grade)**.

Выполним вышеописанный код и закроем окно с кодом, как описано выше. Проверим, как работает *запрос*, описанный выше. Для этого, создайте новый *запрос* и в нем наберите команду EXEC [Отображение студентов по среднему баллу] 3.5 и выполните ее.

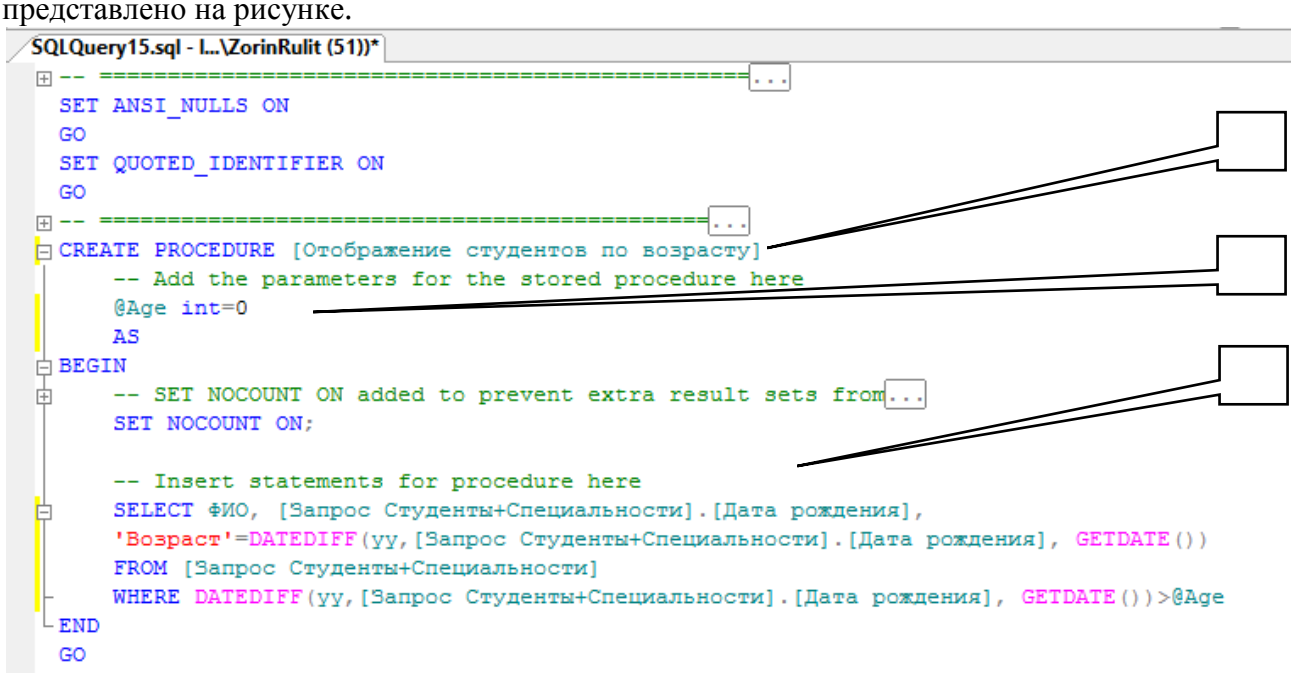


	ФИО студента	Дата первого экзаме...	Наименование предмета первого экзамена	Оценка первого экзамена	Дата второго экзаме...	Наименование предмета вто
1	Иванов А.И.	2015-02-01	Операционные системы	5	2015-02-09	Языки программирования
2	Петрова И.И.	2015-01-30	Проектирование информационных систем	4	2015-02-23	Базы данных
3	Кожевников А.А.	2015-01-12	Языки программирования	4	2015-01-18	Проектирование информаци
4	Пальчикова Н.Е.	2014-12-17	Офисные пакеты	4	2014-12-26	Языки программирования
5	Леухин П.Г.	2015-01-25	Операционные системы	5	2015-02-02	Базы данных

В нижней части окна с кодом появиться результат выполнения хранимой процедуры **"Отображение студентов по среднему баллу"**.

В заключение решим более сложную задачу – отображение студентов старше заданного возраста. Причем возраст будет автоматически вычисляться в зависимости от даты рождения.

Создадим новую хранимую процедуру и наберем код новой процедуры как представлено на рисунке.



Рассмотрим код создаваемой процедуры **"Отображение студентов по возрасту"** более подробно (рис. 4.8):

1. CREATE PROCEDURE [Отображение студентов по возрасту] - определяет имя создаваемой процедуры как "Отображение студентов по возрасту";
2. @Age int=0 - определяют параметр процедуры **Age**. Параметру можно присвоить целые числа (Тип данных int), значения по умолчанию равны 0;
3. ФИО, [Запрос Студенты+Специальности].[Дата рождения], 'Возраст'=DATEDIFF(yy,[Запрос Студенты+Специальности].[Дата рождения], GETDATE()) - отображает из запроса **"Запроса Студенты+Специальности"** (FROM [Запрос Студенты+Специальности]) поля **"ФИО"** (ФИО) и **"Дата рождения"** ([Запрос Студенты+Специальности].[Дата рождения]), а также отображает возраст студента ('Возраст') в годах (yy), вычисленный исходя из его даты рождения и текущей даты(DATEDIFF(yy,[Запрос Студенты+Специальности].[Дата рождения], GETDATE())).

Более того, выводятся студенты возраст которых больше определенного в параметре **"Age"** (DATEDIFF(yy,[Запрос Студенты+Специальности].[Дата рождения], GETDATE())>@Age).

Встроенная функция **DATEDIFF** вычисляющая количество периодов между двумя датами, имеет следующий синтаксис: DATEDIFF(<период>,<начальная дата>, <конечная дата>)

Выполним код запроса **"Отображение студентов по возрасту"**, а затем закроем окно с кодом, как описано выше. Проверим, как работает запрос. Для этого, создадим новый запрос и в нем наберем команду EXEC [Отображение студентов по возрасту] 26 и выполните ее.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.2 Разработка и администрирование базы данных

Лабораторная работа № 10
Создание триггеров

Цель: получение практических навыков разработки триггеров.

Выполнив работу, Вы будете:
уметь:
- создавать триггеры.

Материальное обеспечение:
Методические указания для выполнения лабораторных работ, вариант задания, компьютер, программное обеспечение: MS SQLServer.

Задание:
Создать триггеры для базы данных в СУБД MS SQLServer.

Краткие теоретические сведения:
Триггеры являются одной из разновидностей хранимых процедур. Их исполнение происходит при выполнении для таблицы какого-либо оператора языка манипулирования данными (DML). *Триггеры* используются для проверки целостности данных, а также для отката транзакций.

Триггер – это откомпилированная SQL-процедура, исполнение которой обусловлено наступлением определенных событий внутри реляционной базы данных. Применение *триггеров* большей частью весьма удобно для пользователей базы данных. И все же их использование часто связано с дополнительными затратами ресурсов на операции ввода/вывода. В том случае, когда тех же результатов (с гораздо меньшими непроизводительными затратами ресурсов) можно добиться с помощью хранимых процедур или прикладных программ, применение *триггеров* нецелесообразно.

Триггеры – особый инструмент SQL-сервера, используемый для поддержания целостности данных в базе данных. С помощью ограничений целостности, правил и значений по умолчанию не всегда можно добиться нужного уровня функциональности. Часто требуется реализовать сложные алгоритмы проверки данных, гарантирующие их достоверность и реальность. Кроме того, иногда необходимо отслеживать изменения значений таблицы, чтобы нужным образом изменить связанные данные. *Триггеры* можно рассматривать как своего рода фильтры, вступающие в действие после выполнения всех операций в соответствии с правилами, стандартными значениями и т.д.

Триггер представляет собой специальный тип хранимых процедур, запускаемых сервером автоматически при попытке изменения данных в таблицах, с которыми *триггеры* связаны. Каждый *триггер* привязывается к конкретной таблице. Все производимые им модификации данных рассматриваются как одна транзакция. В случае обнаружения ошибки или нарушения целостности данных происходит откат этой транзакции. Тем самым внесение изменений запрещается. Отменяются также все изменения, уже сделанные *триггером*.

Создает *триггер* только владелец базы данных. Это ограничение позволяет избежать случайного изменения структуры таблиц, способов связи с ними других объектов и т.п.

Триггер представляет собой весьма полезное и в то же время опасное средство. Так, при неправильной логике его работы можно легко уничтожить целую базу данных, поэтому *триггеры* необходимо очень тщательно отлаживать.

В отличие от обычной подпрограммы, *триггер* выполняется неявно в каждом случае возникновения *триггерного события*, к тому же он не имеет аргументов. Приведение его в действие иногда называют запуском *триггера*.

Основной формат команды `CREATE TRIGGER` показан ниже:

```
<Определение_триггера>::=
  CREATE TRIGGER имя_триггера
  BEFORE | AFTER <триггерное_событие>
  ON <имя_таблицы>
  [REFERENCING
    <список_старых_или_новых_псевдонимов>]
  [FOR EACH { ROW | STATEMENT}]
  [WHEN(условие_триггера)]
  <тело_триггера>
```

Порядок выполнения работы:

В БД "Microsoft SQL Server" все триггеры создаются отдельно для каждой таблицы и располагаются в обозревателе объектов в папке **"Триггеры"**. В нашем случае, папка **"Триггеры"** входит в состав таблицы **"Студенты"**.

Для начала создадим триггер, выводящий сообщение **"Запись добавлена"** при добавлении записи в таблицу **"Студенты"**. Создадим новый триггер, щелкнув **ПКМ** по папке **"Триггеры"** в таблице **"Студенты"** и в появившемся меню выбрав пункт **"Создать триггер"**. Появится следующее окно с новым триггером:

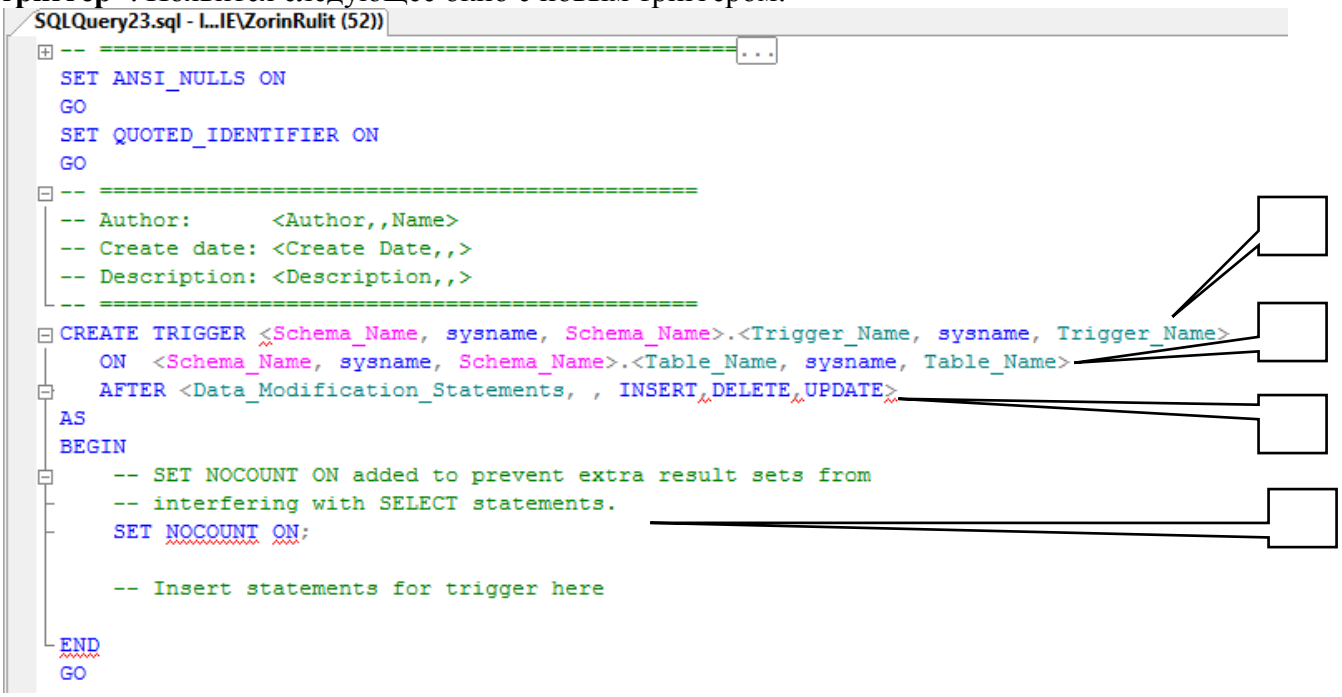


Рисунок 6.2

Рассмотрим структуру триггеров:

1. Область определения имени функции (**Trigger_Name**);
2. Область, показывающая для какой таблицы, создается триггер (**Table_Name**);
3. Область показывающая, когда выполнять триггер (**INSERT** - при создании записи в таблице, **DELETE** - при удалении и **UPDATE** - при изменении) и как его выполнять (**AFTER** - после выполнения операции, **INSTEAD OF** - вместо выполнения операции);
4. Тело триггера, содержит команды языка программирования запросов T-SQL.

В окне нового триггера наберите код как показано на рисунке.

```
SQLQuery23.sql - l...\ZorinRulit (52))*
-- ============================================...
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================...
CREATE TRIGGER [Индикатор добавления]
   ON  dbo.Студенты
   AFTER INSERT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;
    -- Insert statements for trigger here
    PRINT 'Запись добавлена'
END
GO
```

Из рисунка видно, что создаваемый триггер **"Индикатор добавления"** выполняется после добавления записи (**AFTER INSERT**) в таблицу "Студенты" (**ON dbo.Студенты**). После добавления записи триггер выведет на экран сообщение "Запись добавлена" (**PRINT 'Запись добавлена'**). Выполните набранный код, нажав кнопку Выполнить на панели инструментов. В нижней части окна с кодом появится сообщение **"Выполнение команд успешно завершено."**.

```
SQLQuery24.sql - l...\ZorinRulit (56))*
INSERT INTO dbo.Студенты
VALUES (
        'Татаринов А.В.'
        ,'Мужской'
        ,'05/07/1988'
        ,'Отец и Мать'
        ,'Казань'
        ,'+79342213455'
        ,'3456-465375'
        ,74378
        ,'05/07/2013'
        ,'ПИ22'
        ,2
        ,5
        ,1
        )
```

```
Сообщения
Запись добавлена

(строк обработано: 1)
```

Проверим, как работает новый триггер. Создайте новый пустой запрос и в нем наберите следующую команду для добавления новой записи в таблицу **"Студенты"**:

Выполните набранную команду, нажав кнопку Выполнить на панели инструментов. В таблицу будет добавлена новая запись, и триггер выведет сообщение **"Запись добавлена"**.

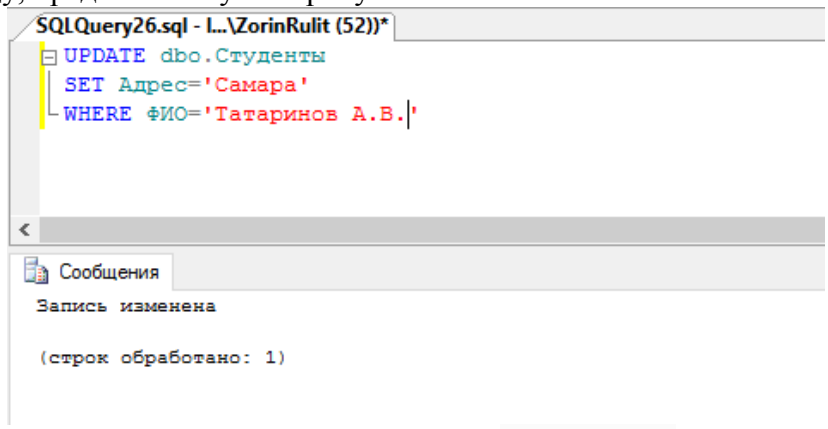
Теперь создадим триггер отображающий сообщение **"Запись изменена"**. Создайте новый триггер, как в предыдущем случае. В окне нового триггера наберите следующий код:

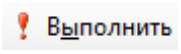
```
SQLQuery25.sql - l...\ZorinRulit (52))*
-- ============================================...
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================...
CREATE TRIGGER [Индикатор изменения]
   ON  dbo.Студенты
   AFTER UPDATE
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;
    -- Insert statements for trigger here
    PRINT 'Запись изменена'
END
GO
```

Из рисунка видно, что новый триггер **"Индикатор изменения"** выполняется после изменения записи (**AFTER UPDATE**) в таблице "Студенты" (**ON dbo.Студенты**). После

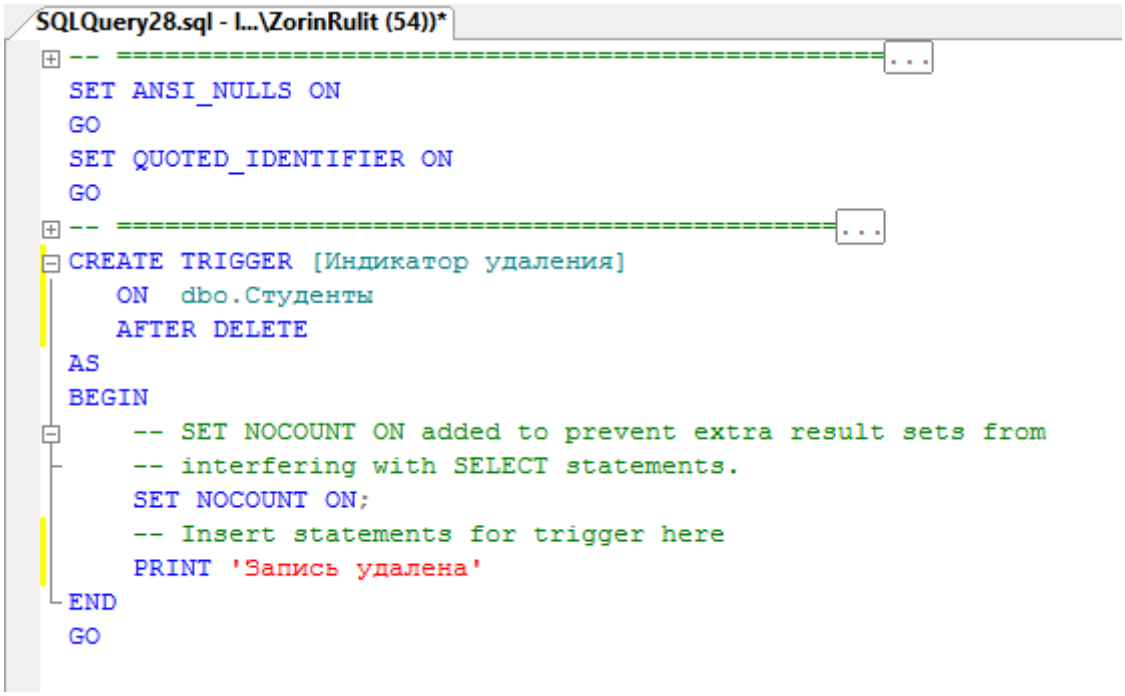
изменения записи триггер выведет на экран сообщение "Запись изменена" (**PRINT 'Запись изменена'**). Выполните набранный код. В нижней части окна с кодом появится сообщение **"Выполнение команд успешно завершено."**.

Проверим работоспособность созданного триггера. Создайте новый *запрос* и в нем наберите команду, представленную на рисунке 6.6.



Выполните набранную команду, нажав кнопку  на панели инструментов. В таблице будет изменена одна запись, и триггер выведет сообщение **"Запись изменена"**.

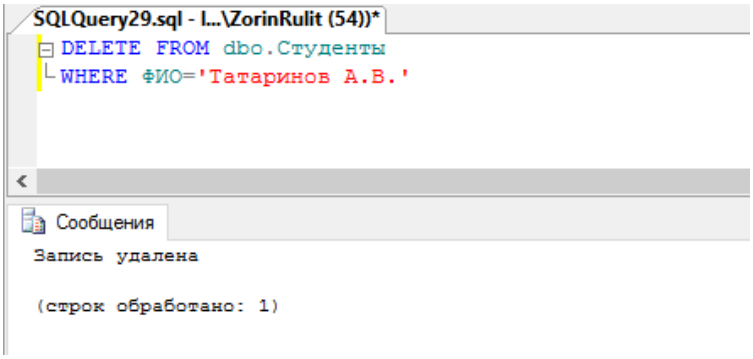
Для полноты картины создадим триггер, выводящий сообщение при удалении записи из таблицы **"Студенты"**. Создайте новый триггер и в нем наберите код, показанный на рисунке.



Создаваемый триггер **"Индикатор удаления"** выполняется после удаления записи (**AFTER DELETE**) из таблицы студенты (**ON dbo.Студенты**). После удаления записи триггер выводит сообщение "Запись удалена" (**PRINT 'Запись удалена'**).

Выполните код. В нижней части окна с кодом появится сообщение **"Выполнение команд успешно завершено."**.

Проверим работу триггера **"Индикатор удаления"** удалив созданную ранее запись из таблицы **"Студенты"**. Для этого создайте новый запрос и в нем наберите следующую команду:



Выполните вышеприведенную команду. После удаления записи триггер **"Индикатор удаления"** отобразит сообщение **"Запись удалена"**.

В заключение рассмотрим пример применения триггеров для обеспечения целостности данных. Создадим триггер **"Удаление студента"**, который при удалении записи из таблицы студенты сначала удаляет все связанные с ней записи из таблицы **"Оценки"**, а затем удаляет саму запись из таблицы **"Студенты"**, тем самым обеспечивается целостность данных.

Создайте новый триггер и в нем наберите следующий код:

```
SQLQuery30.sql - l...\ZorinRulit (51))*
-- ============================================...
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================...
CREATE TRIGGER [Удаление студента]
   ON  dbo.Студенты
   INSTEAD OF DELETE
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;
    -- Insert statements for trigger here
    DELETE dbo.Оценки
    FROM Deleted
    WHERE Deleted.[Код студента]=Оценки.[Код студента]
    DELETE dbo.Студенты
    FROM Deleted
    WHERE Deleted.[Код студента]=Студенты.[Код студента]
END
GO
```

Создаваемый триггер **"Удаление студента"** выполняется вместо удаления записи (**INSTEAD OF DELETE**) из таблицы "Студенты" (**ON dbo.Студенты**).

Замечание: При срабатывании триггера вместо удаления записи создается временная константа **Deleted**, содержащая имя таблицы из которой должно было быть произведено удаление.

После срабатывания триггера из таблицы **"Оценки"** удаляется запись, у которой значение поля **"Код студента"** равно значению такого же поля у удаляемой записи из таблицы **"Студенты"**. Эту операцию выполняют следующие команды:

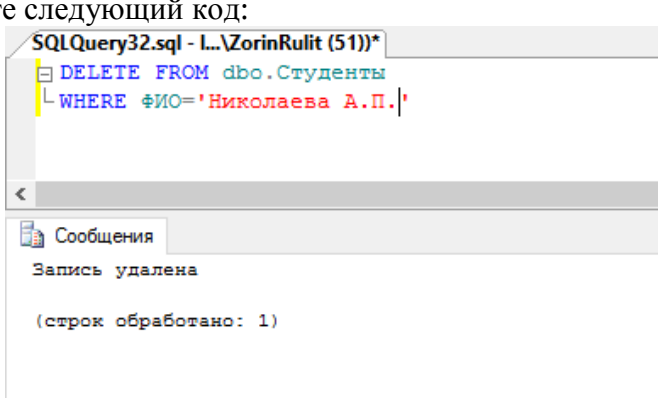
DELETE dbo.Оценки FROM Deleted
WHERE Deleted.[Код студента] = Оценки.[Код студента]

Затем удаляется запись из таблицы **"Студенты"**, которую удаляли до срабатывания триггера. Удаление выполняется следующими командами:

DELETE dbo.Студенты
FROM Deleted
WHERE Deleted.[Код студента] = Студенты.[Код студента]

Выполните код. В нижней части окна с кодом появиться сообщение **"Выполнение команд успешно завершено."**.

Проверим, как работает триггер **"Удаление студента"**. Для этого создайте новый запрос и в нем наберите следующий код:



При срабатывании триггера сначала из таблицы **"Оценки"** удалятся все связанные с удаляемой записью записи, а затем удаляется сама удаляемая запись из таблицы **"Студенты"**, при этом сохраняется целостность данных.

Хотелось бы заметить, что без использования триггера **"Удаление студента"** нам бы не удалось удалить запись из таблицы **"Студенты"**. Команда удаления была бы заблокирована диаграммой **"Диаграмма БД Студенты"** во избежание нарушения целостности данных.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.2 Разработка и администрирование базы данных

Лабораторная работа № 11
Транзакции и блокировки

Цель: получение практических навыков работы с транзакциями и блокировками.

Выполнив работу, Вы будете:
уметь:
- осуществлять блокировку и разблокировку таблиц;
- работать с транзакциями.

Материальное обеспечение:
Методические указания для выполнения лабораторных работ, вариант задания, компьютер, программное обеспечение: MS SQLServer.

Задание:
1. Выполнить блокировку таблиц на чтение и запись.
2. Выполнить транзакции для пользователей.
3. Выполнить анализ по результатам выполнения запросов.

Краткие теоретические сведения:
Концепция *транзакций* – неотъемлемая часть любой клиент-серверной базы данных.

Под ***транзакцией*** понимается *неделимая* с точки зрения воздействия на БД последовательность операторов манипулирования данными (чтения, удаления, вставки, модификации), приводящая к одному из двух возможных результатов: либо последовательность выполняется, если все операторы правильные, либо вся *транзакция* откатывается, если хотя бы один оператор не может быть успешно выполнен. Обработка *транзакций* гарантирует целостность информации в базе данных. Таким образом, *транзакция* переводит базу данных из одного целостного состояния в другое.

Поддержание механизма *транзакций* – показатель уровня развитости СУБД. Корректное поддержание *транзакций* одновременно является основой обеспечения целостности БД. *Транзакции* также составляют основу *изолированности* в многопользовательских системах, где с одной БД параллельно могут работать несколько пользователей или прикладных программ. Одна из основных задач СУБД – обеспечение *изолированности*, т.е. создание такого режима функционирования, при котором каждому пользователю, казалось бы, что БД доступна только ему. Такую задачу СУБД принято называть параллелизмом *транзакций*.

Большинство выполняемых действий производится в теле *транзакций*. По умолчанию каждая команда выполняется как самостоятельная *транзакция*. При необходимости пользователь может явно указать ее *начало* и *конец*, чтобы иметь возможность включить в нее несколько команд.

При выполнении *транзакции* система управления базами данных должна придерживаться определенных правил обработки набора команд, входящих в *транзакцию*. В частности, разработано четыре правила, известные как требования ACID, они гарантируют правильность и надежность работы системы.

ACID-свойства транзакций

Характеристики *транзакций* описываются в терминах ACID (Atomicity, Consistency, Isolation, Durability – *неделимость*, *согласованность*, *изолированность*, *устойчивость*).

- *Транзакция* ***неделима*** в том смысле, что представляет собой единое целое. Все ее компоненты либо имеют место, либо нет. Не бывает частичной *транзакции*. Если может быть выполнена лишь часть *транзакции*, она отклоняется.
- *Транзакция* является ***согласованной***, потому что не нарушает бизнес-логику и отношения между элементами данных. Это *свойство* очень важно при разработке клиент-серверных систем, поскольку в хранилище данных поступает большое количество *транзакций* от разных систем и объектов. Если хотя бы одна из них нарушит целостность данных, то все остальные могут выдать неверные результаты.
- *Транзакция* всегда ***изолированна***, поскольку ее результаты самодостаточны. Они не зависят от предыдущих или последующих *транзакций* – это *свойство* называется *сериализуемостью* и означает, что *транзакции* в последовательности независимы.
- *Транзакция* ***устойчива***. После своего завершения она сохраняется в системе, которую ничто не может вернуть в исходное (до *начала транзакции*) состояние, т.е. происходит фиксация *транзакции*, означающая, что ее действие постоянно даже при сбое системы. При этом подразумевается некая форма хранения информации в постоянной памяти как часть *транзакции*.

Указанные выше правила выполняет сервер. Программист лишь выбирает нужный *уровень изоляции*, заботится о соблюдении логической целостности данных и бизнес-правил. На него возлагаются обязанности по созданию эффективных и логически верных алгоритмов обработки данных. Он решает, какие команды должны выполняться как одна *транзакция*, а какие могут быть разбиты на несколько последовательно выполняемых *транзакций*. Следует по возможности использовать небольшие *транзакции*, т.е. включающие как можно меньше команд и изменяющие минимум данных. Соблюдение этого требования позволит наиболее эффективным образом обеспечить одновременную работу с данными множества пользователей.

Блокировки

Повышение эффективности работы при использовании небольших *транзакций* связано с тем, что при выполнении *транзакции* сервер накладывает на данные *блокировки*.

Блокировкой называется временное ограничение на выполнение некоторых операций обработки данных. *Блокировка* может быть наложена как на отдельную строку таблицы, так и на всю базу данных. ***Управлением блокировками*** на сервере занимается менеджер блокировок, контролирующий их применение и разрешение конфликтов. *Транзакции* и *блокировки* тесно связаны друг с другом. *Транзакции* накладывают *блокировки* на данные, чтобы обеспечить выполнение требований ACID. Без использования блокировок несколько *транзакций* могли бы изменять одни и те же данные.

Блокировка представляет собой метод *управления параллельными процессами*, при котором объект БД не может быть модифицирован без ведома *транзакции*, т.е. происходит блокирование доступа к объекту со стороны других *транзакций*, чем исключается непредсказуемое изменение объекта. Различают два вида *блокировки*:

- *блокировка* записи – *транзакция* блокирует строки в таблицах таким образом, что запрос другой *транзакции* к этим строкам будет *отменен*;
- *блокировка* чтения – *транзакция* блокирует строки так, что запрос со стороны другой *транзакции* на *блокировку* записи этих строк будет отвергнут, а на *блокировку* чтения – принят.

Порядок выполнения работы:

1. Зайдите на сервер MS SQLServer под пользователем user1 и во втором сеансе связи под пользователем user2.
2. В базе данных sales создайте таблицу product:

```
create table product (
  id serial,
  name_prod varchar(100) not null,
  description text,
  price decimal(10,2) not null,
  qty int unsigned,
  primary key(id))
  Engine InnoDB character set utf8;
```

3. Заполните таблицу значениями в соответствии с заданными.

id	pname	description	price	qty
1	Утюг BOSH	Паровой удар, мощность 1000Вт	3500.00	6
2	Холодильник INDESIT	Три камеры	15600.00	14
4	Стиральная машина BOSH	Загрузка вертикальная	6556.99	55
5	Стиральная машина Indesit	Загрузка горизонтальная	18000.00	4

4. Выполните действия каждым пользователем, в соответствии с таблицей. После каждого действия проанализировать результат выполнения запросов.

Пользователь user1 **Конкурирующая *транзакция***	**Пользователь user2** **Текущая *транзакция***
USE sales START TRANSACTION trA	USE sales START TRANSACTION trB
1. SELECT * FROM product	2. SELECT * FROM product
3. UPDATE product SET qty=qty+10 WHERE id=4	4. SELECT * FROM product (выполнить анализ)
5. DELETE FROM product WHERE id =4	6. SELECT * FROM product (выполнить анализ)
7. INSERT INTO product (name_pr, description, price, qty) VALUES ('Утюг паровой', 'BOSH', 5500.00, 23)	8. SELECT * FROM product (выполнить анализ) 9. UPDATE product SET qty=qty+10 WHERE id=4 (выполнить анализ)
10. DELETE FROM product WHERE id =4 (выполнить анализ)	11. INSERT INTO product (name_pr, description, price, qty) VALUES ('Пароварка', 'BOSH', 3700.00, 14) (выполнить анализ)
12. ROLLBACK TRANSACTION trA	13. COMMIT TRANSACTION trB

14. LOCK TABLES product READ 16. SELECT * FROM product (выполнить анализ) 18. UPDATE product SET qty=qty+20 WHERE id=4 (выполнить анализ) 19. UNLOCK TABLES 21. LOCK TABLES product WRITE 22. SELECT * FROM product (выполнить анализ) 23. UNLOCK TABLES	15. SELECT * FROM product (выполнить анализ) 17. UPDATE product SET qty=qty+20 WHERE id=4 (выполнить анализ) 20. (выполнить анализ) 22. SELECT * FROM product (выполнить анализ) 24. (выполнить анализ)

5. Составить отчет

Анализ выполнения запросов при блокировках и транзакциях

№	Пользователь user1	Пользователь user2

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.2 Разработка и администрирование базы данных

Лабораторная работа № 12
Выполнение настроек для автоматизации обслуживания базы данных

Цель: получение практических навыков по освоению настроек для автоматизации обслуживания базы данных

Выполнив работу, Вы будете:

уметь:

- выполнять автоматизацию процесса выполнения скриптов.

Материальное обеспечение:

Методические указания для выполнения лабораторных работ, вариант задания, компьютер, программное обеспечение: MS SQLServer.

Задание:

Выполнить настройки для автоматизации процесса выполнения скриптов.

Краткие теоретические сведения:

При постоянном использовании базы данных увеличиваются объемы данных, усложняется функционал, количество пользователей возрастает. Чтобы сохранять производительность базы данных в хорошем состоянии, необходимо постоянно выполнять работы по обслуживанию базы данных и серверов базы данны.

Основными действиями по обслуживанию базы данныявляются следующие операции:

1) обслуживание Индексов;
2) обновление Статистики;
3) чистка процедурного Кэша.

Процесс обслуживания баз данных будет выполняться с помощью скриптов написанных на языке T-SQL. Язык T-SQL является ключом к использованию MS SQL Server. Все приложения, взаимодействующие с экземпляром MS SQL Server, независимо от их реализации и пользовательского интерфейса, отправляют серверу инструкции Transact-SQL.

Разработка скрипта для резервного копирования баз данных

Одной из важнейших задач для обеспечения защищенности и доступности баз данных организации является резервное копирование.

```
DECLARE @pathName NVARCHAR(512)
--создаем переменную с символьным типом данных
SET @pathName = 'D:\BackUp\stud_' + CONVERT(varchar(8),GETDATE(),112) + '.bak'
/*задаем переменной значение путь сохранения резервных копий и имя файла
с текущей датой*/
BACKUP DATABASE [students] TO DISK = @pathName
-- выбираем базу данных и указываем переменную со значением пути и имени
WITH NOFORMAT, NOINIT, NAME = N'backup_db_stud', SKIP
/*указываем имя создаваемого резервного набора и
отключаем срок хранения*/
GO
```

Этот скрипт создает резервную копию с именем файла stud_YYYYDDMM.bak, где YYYYDDMM – это текущая дата. Дата в имени файла позволит нам создавать каждый день резервное копирование в новом файле.

Разработка скрипта для обслуживания индексов баз данных

Индексы позволяют быстро получить доступ к нужным данным без поиска по всей базе данных. В SQL Server индексы создаются для таблиц, представлений и столбцов. Создав индекс для таблицы, вы ускоряете поиск данных в таблице.

Со временем без обслуживания индексов накапливаются следующие проблемы:

- индексы становятся сильно фрагментированными и перемешанными;
- часть данных строк, когда-то участвовавших, в индексе уже удалена, и из-за этого индекс занимает на диске больше места и требует при выполнении запросов больше операций ввода-вывода.

Перестроение и дефрагментация индексов используется с целью повышения их производительности. При этом оптимизируется распределение данных и свободного места на страницах индекса, что также повышает его эффективность.

Скрипт автоматически реорганизует или перестраивает все секции индексов в базе данных со средней степенью фрагментации более 10 процентов. На рисунке 1 представлена подготовительная часть скрипта. На рисунке 2 представлена реорганизация и перестроение индексов.

```
use students;              --выбор базы
go
set nocount on;
declare @objectid int;     --создание локальных переменных
declare @indexid int;
declare @partitioncount bigint;
declare @shemaname nvarchar(130);
declare @objectname nvarchar(130);
declare @indexname nvarchar(130);
declare @partitionnum bigint;
declare @partitions bigint;
declare @frag float;
declare @command nvarchar(4000);
print ' ';
print 'begin: '+convert(char,getdate(),120);
select
OBJECT_ID as objectid,         --присвоение псевдонимов
index_id as indexid,
PARTITION_number as partitionnum,
avg_fragmentation_in_percent as frag
into #work_to_do
/*создание временной таблицы для добавления в нее результатов запроса*/
from sys.dm_db_index_physical_stats(db_id(),null, null, null, 'limited')
where avg_fragmentation_in_percent > 10.0 and index_id > 0;
/*получили данные о таблицах и индексах из sys.dm_db_index_physical_stats
и сконвертировали полученные ID в имена объектов*/
declare partitions cursor for select * from #work_to_do;
--создание курсора для определения списка
open partitions;              --открытие курсора
```

```
while (1=1)
begin;
 fetch next
 from partitions
 into @objectid, @indexid, @partitionnum, @frag;
 if @@FETCH_STATUS < 0 break;
 select @objectname=QUOTENAME(o.name), @shemaname=QUOTENAME(s.name)
 from sys.objects as o
 join sys.schemas as s on s.schema_id=o.schema_id
 where o.object_id=@objectid;
 select @indexname=QUOTENAME(name)
 from sys.indexes
 where object_id=@objectid and index_id=@indexid;
 select @partitioncount=COUNT(*)
 from sys.partitions
 where object_id=@objectid and index_id=@indexid;
 if @frag<30   --если общая фрагментация индекса меньше 30, начинается реорганизация
 set @command='alert index' + @indexname + 'on' + @shemaname + '.'+@objectname+'reorganize';
  if @frag>=30   --если общая фрагментация индекса больше 30, нечинается перестроение
 set @command='alert index' + @indexname + 'on' + @shemaname + '.'+@objectname+'rebuld with (fillfactor=90)';
 if @partitioncount>1     --если есть несколько секций
 set @command=@command+'partitions='+cast(@partitionnum as nvarchar(10));
 print rtrim(convert(char,getdate(),108))+' : '+@command;   --вывод выполняемого запроса
 exec (@command);
 end;

 close partitions;    --закрытие курсора из памяти
 deallocate partitions;    --удаление курсора из памяти

 drop table #work_to_do;   --удаление временной таблицы
 print 'end: '+convert(char, getdate(),120);
```

Скрипт для обновления статистики

Статистика – очень важный механизм в MS SQL Server. Для эффективного выполнения запросов необходимо постоянно поддерживать статистику для каждой из баз данных в актуальном состоянии. Чтобы минимизировать потерю производительности, необходимо вручную запускать команду по обновлению всей статистики в базе данных. Особенно это нужно выполнять сразу же после массовых изменений. В MS SQL Server существует специальная системная процедура, которая будет обновлять только ту статистику, которая требует обновления, тем самым предотвращая ненужные обновления статистики по неизменным срокам: sp_updatestats.

```
use students   --выбор базы
go
exec sp_updatestats  --системная процедура
```

Настройка чистки процедурного кэша

Процедурный кэш – это область оперативной памяти, зарезервированная для сервера MS SQL Server, в которой содержаться планы выполнения запросов. Если при выполнении запроса подходящий план для него не будет найден, то произойдет компиляция этого запроса и скомпилированный план будет помещен в кэш. Эта операция требует дополнительных ресурсов. Поэтому если на сервере установлен достаточный объем оперативной памяти и в качестве сервера баз данных используется MS SQL Server 2005 и более новые версии, то ручную чистку процедурного кэша следует выполнять при крайней необходимости. Выполнять такую операцию следует в следующих случаях:

- выполнено обновление статистики для всей базы данных;
- выполнено изменение индексов (перестроение или дефрагментация) для всей базы данных.

В этих случаях для всех запросов в этой базе данных автоматически будет использована перекомпиляция. Так что имеет смысл освободить оперативную память от уже ненужных планов (рисунок 5).

```
DBCC FREEPROCCACHE    -- системная процедура
```

Порядок выполнения работы:

Чтобы автоматизировать процесс выполнения описанных операций нужно воспользоваться системой SQL Server Agent:

1) Добавить новое Задание (Job) и задать имя «Обслуживание БД [ИмяБазы]»;

2) В задании перейти на вкладку Шаги (Steps) и добавить новый шаг с названием «Обслуживание Индексов». В свойствах шага указать типа= «Transact-SQL script» (по умолчанию). Скопировать скрипт из раздела Обслуживание индексов (заменив [ИмяБазы] на имя конкретной базы данных) и вставить его в поле Команда в шаге задания. Сохранить этот шаг;

3) Добавить второй шаг задания с названием «Обновление статистики». В свойствах шага указать типа= «Transact-SQL script» (по умолчанию). Скопировать скрипт из раздела Обновление статистики (заменив [ИмяБазы] на имя конкретной базы данных) и вставить его в поле Команда в шаге задания. Сохранить этот шаг;

4) Добавить третий шаг задания с названием «Чистка процедурного кэша». В свойствах шага указать типа= «Transact-SQL script» (по умолчанию). Скопировать скрипт из раздела Чистка процедурного кэша и вставить его в поле Команда в шаге задания. Сохранить этот шаг;

5) В задании перейти на вкладку Расписания (Shedules) и нажать кнопку Создать(New). Расписание для задания необходимо составить следующим образом: каждый рабочий день, рано утром в 05:00 (чтобы все операции были выполнены перед началом рабочего дня);

6) Сохранить расписание и сохранить задание;

7) На следующий день проверить по журналу заданий, что новое задание «Обслуживание БД [ИмяБазы]» успешно выполнилось.

Эти действия нужно выполнить для каждой пользовательской базы данных (на каждую базу, должно быть свое задание со своим расписанием, расписания не должны пересекаться по времени!)

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Организация защиты данных в хранилищах

Лабораторная работа № 13
Выполнение резервного копирования и восстановления базы данных из резервной копии

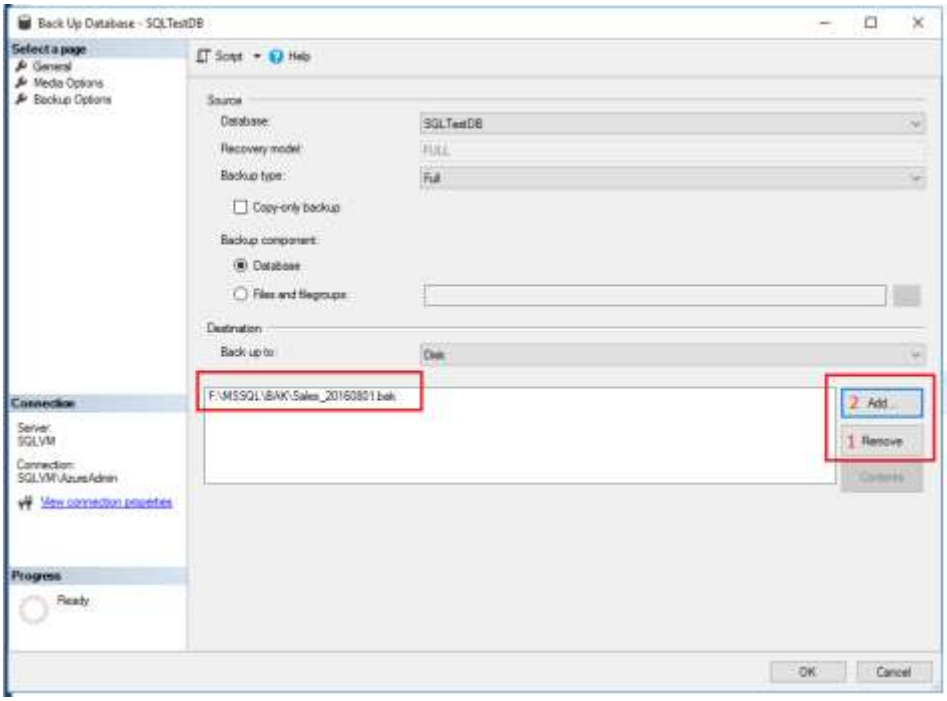
Цель: получение практических навыков по освоению операций резервного копирования и восстановления базы данных

Выполнив работу, Вы будете:
уметь:
- выполнять резервное копирование базы данных;
- выполнять восстановление базы данных из резервной копии.

Материальное обеспечение:
Методические указания для выполнения лабораторных работ, вариант задания, компьютер, программное обеспечение: MS SQLServer.

Задание 1:
Создать резервную копию базы данных.

Порядок выполнения работы:
1. После подключения к соответствующему экземпляру Microsoft Компонент SQL Server Database Engine в **обозревателе объектов** разверните дерево сервера.
2. Разверните узел **Базы данных** и выберите пользовательскую базу данных или разверните узел **Системные базы данных** и выберите системную базу данных.
3. Щелкните правой кнопкой мыши базу данных, резервную копию которой нужно создать, наведите указатель на пункт **Задачи** и выберите команду **Создать резервную копию...** .
4. В диалоговом окне **Резервное копирование базы данных** выбранная база данных приводится в раскрывающемся списке (ее можно изменить на любую другую базу данных на сервере).
5. В раскрывающемся списке **Тип резервной копии** выберите нужный тип. По умолчанию выбран тип **Полный**.
6. В разделе **Компонент резервного копирования** выберите **База данных**.
7. В разделе **Назначение** проверьте расположение по умолчанию для файла резервной копии (в папке ../mssql/data).
8. Чтобы выполнить резервное копирование на другое устройство, выберите его в раскрывающемся списке **Создать резервную копию на**. Чтобы создать чередующийся резервный набор данных на основе нескольких файлов для повышения скорости резервного копирования, нажмите кнопку **Добавить** и добавьте дополнительные объекты и (или) назначения резервного копирования.
9. Чтобы удалить носитель резервной копии, выберите его и нажмите кнопку **Удалить**. Чтобы просмотреть содержимое существующего назначения резервного копирования, выберите его и щелкните **Содержимое**.
10. (Необязательно) Просмотрите другие доступные параметры на страницах **Параметры носителя** и **Параметры резервного копирования**.
11. Нажмите кнопку **ОК**, чтобы запустить резервное копирование.
12. После успешного завершения резервного копирования нажмите кнопку **ОК**, чтобы закрыть диалоговое окно SQL Server Management Studio.



Задание 2:

Восстановление полной резервной копии базы данных.

Порядок выполнения работы:

1. В **обозревателе объектов** подключитесь к экземпляру компонента Компонент SQL Server Database Engine и разверните его.
2. Щелкните правой кнопкой мыши узел **Базы данных** и выберите команду **Восстановить базу данных...**
3. Чтобы указать источник и расположение восстанавливаемых резервных наборов данных, используйте страницу **Общие**, раздел **Источник**. Выберите один из следующих вариантов:
   - **База данных**
     Выберите из раскрывающегося списка базу данных для восстановления. Данный список содержит только базы данных, резервное копирование которых было выполнено в соответствии с журналом резервного копирования **msdb** .
   - **Устройство**
     Нажмите кнопку обзора (**...**), после чего откроется диалоговое окно **Выбор устройств резервного копирования**.

Диалоговое окно **Выбор устройств резервного копирования**.

Носитель данных резервной копии

Выберите тип носителя в раскрывающемся списке **Тип носителя данных резервной копии** . Примечание. Параметр **Лента** появляется только в случае, если на компьютере установлен ленточный накопитель, а параметр **Устройство резервного копирования** — только в случае, если имеется хотя бы одно устройство резервного копирования.

Добавление

В зависимости от типа носителя данных, выбранного в поле **Носитель резервной копии** , при нажатии кнопки **Добавить** открывается одно из следующих диалоговых окон. (Если список в поле со списком **Тип носителя резервной копии** заполнен, кнопка **Добавить** недоступна.)

Удалить

Удаляет один или несколько выбранных файлов, лент или устройств резервного копирования.

Содержание

Отображает содержимое носителя выбранного файла, ленты или устройства резервного копирования. Эта кнопка может не работать, если тип носителя — **URL-адрес.**

Тип носителя резервной копии

Перечисляет выбранные носители.

После добавления нужных устройств в списке **Носитель резервной копии** нажмите кнопку **ОК** для возвращения на страницу **Общие.**

В списке **Источник: Устройство: База данных** выберите имя базы данных, которую нужно восстановить.

4. В разделе **Назначение**, в поле **База данных** автоматически появится имя базы данных для восстановления. Для изменения имени базы данных введите новое имя в окно **База данных.**
5. В поле **Восстановить до** оставьте значение по умолчанию **До последней выбранной резервной копии** или щелкните **Временную шкалу**, чтобы перейти в диалоговое окно **Временная шкала резервной копии** и выбрать вручную конкретное пороговое время восстановления.
6. В сетке **Резервные наборы данных для восстановления** выберите нужные резервные наборы. В этой сетке отображаются резервные копии, доступные в указанном месте. По умолчанию предлагается план восстановления. Чтобы переопределить предложенный план восстановления, можно изменить выбранные элементы в сетке. Выбор всех резервных копий, которые зависят от восстановления более ранних копий, отменяется автоматически, как только отменяется выбор более ранних копий.
7. При необходимости нажмите кнопку **Файлы** на панели **Выбор страницы** и перейдите в диалоговое окно **Файлы**. Отсюда можно восстановить базу данных в новое расположение, определив новое место восстановления для каждого файла в сетке **Восстановить файлы базы данных как.**
8. Чтобы просмотреть или выбрать дополнительные параметры, на странице **Параметры** на панели **Параметры восстановления** вберите любой из следующих параметров, если он подходит к данной ситуации.
   a. Параметры**WITH** (необязательно):
      - Перезаписать существующую базу данных (WITH REPLACE)
      - Сохранить параметры репликации (WITH KEEP_REPLICATION)
      - Ограничить доступ к восстановленной базе данных (WITH RESTRICTED_USER)
   b. Выберите параметр в поле **Состояние восстановления.** В данном окне определяется состояние базы данных после операции восстановления.
      - По умолчанию установлена схема **RESTORE WITH RECOVERY**, при этом база данных находится в готовом состоянии для использования путем отката незафиксированных транзакций. Невозможно восстановить дополнительные журналы транзакций. Выберите данный параметр, если выполняется восстановление всех необходимых резервных копий.
      - Схема **RESTORE WITH NORECOVERY** оставляет базу данных в нерабочем состоянии и не выполняет откат незафиксированных транзакций. Можно восстановить дополнительные журналы транзакций. База данных не может быть использована, пока не будет восстановлена.
      - Схема **RESTORE WITH STANDBY** оставляет базу данных в режиме только для чтения. С помощью данного параметра можно отменить незафиксированные транзакции и сохранить отмененные действия в резервном файле, чтобы результаты восстановления можно было отменить.

c. **Создайте резервную копию заключительного фрагмента журнала до восстановления.** Не для всех сценариев восстановления требуется резервная копия заключительного фрагмента журнала.
d. Если имеются активные соединения с базой данных, то операция восстановления может завершиться ошибкой. Проверьте окно **Закрыть существующие соединения** и убедитесь, что все активные соединения между Среда Management Studio и базой данных закрыты. Этот параметр переводит базу данных в однопользовательский режим перед началом выполнения процедуры восстановления, а затем возвращает в многопользовательский режим после ее завершения.
e. Установите флажок **Выдавать запрос перед восстановлением каждой резервной копии ,** если хотите отследить каждую операцию восстановления. Обычно это не требуется, за исключением случаев, если необходимо наблюдать за состоянием операции восстановления базы данных большого объема.

9. Нажмите кнопку ОК.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Организация защиты данных в хранилищах

Лабораторная работа № 14
Управление привилегиями и доступом к данным

Цель: получение практических навыков управления привилегиями пользователей**.**

Выполнив работу, Вы будете:
уметь:
- создавать пользователей и предоставлять им привилегии;
- применять стандартные методы для защиты объектов базы данных.

Материальное обеспечение:
Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MS SQLServer.

Задание:
Создать пользователей базы данных и предоставить им привилегии.

Краткие теоретические сведения:
Стабильная система *управления пользователями* – обязательное условие *безопасности данных,* хранящихся в любой реляционной СУБД. В языке SQL не существует единственной стандартной команды, предназначенной для *создания пользователей* базы данных – каждая реализация делает это по-своему. В одних реализациях эти специальные команды имеют определенное сходство, в то время как в других их синтаксис имеет существенные отличия. Однако независимо от конкретной реализации все основные принципы одинаковы.

После проектирования логической структуры базы данных, связей между таблицами, ограничений целостности и других структур необходимо определить круг *пользователей,* которые будут иметь *доступ* к базе данных.

В системе SQL-сервер организована двухуровневая настройка ограничения *доступа* к данным. На первом уровне необходимо создать так называемую *учетную запись пользователя* (login), что позволяет ему подключиться к самому серверу, но не дает автоматического *доступа* к базам данных. На втором уровне для каждой базы данных SQL-сервера на основании *учетной записи* необходимо создать запись *пользователя*. На основе *прав,* выданных *пользователю* как *пользователю* базы данных (user), его регистрационное имя (login) получает *доступ* к соответствующей базе данных. В разных базах данных login одного и того же *пользователя* может иметь одинаковые или разные имена user с разными *правами доступа*. Иначе говоря, с помощью *учетной записи пользователя* осуществляется подключение к SQL-серверу, после чего определяются его уровни *доступа* для каждой базы данных в отдельности.

Каждая СУБД должна поддерживать механизм, гарантирующий, что *доступ* к базе данных смогут получить только те *пользователи*, которые имеют соответствующее разрешение. Язык SQL включает операторы `GRANT` и `REVOKE`, предназначенные для организации защиты таблиц в базе данных. Механизм защиты построен на использовании *идентификаторов пользователей*, предоставляемых им *прав* владения и *привилегий*.

Идентификатором пользователя называется обычный идентификатор языка SQL, применяемый для обозначения некоторого *пользователя* базы данных. Каждому *пользователю* должен быть назначен собственный *идентификатор*, присваиваемый администратором базы данных. Из очевидных соображений безопасности *идентификатор пользователя*, как *правило*, связывается с некоторым *паролем*. Каждый выполняемый СУБД SQL-оператор выполняется от имени какого-либо *пользователя*. *Идентификатор*

пользователя определяет, на какие объекты базы данных *пользователь* может ссылаться и какие операции с этими объектами он имеет *право* выполнять.

Каждый созданный в среде SQL объект имеет своего *владельца*, который изначально является единственной персоной, знающей о существовании данного объекта, и имеет *право* выполнять с ним любые операции.

Привилегиями, или ***правами***, называются действия, которые *пользователь* имеет *право* выполнять в отношении данной таблицы базы данных или представления.

Оператор GRANT применяется для ***предоставления привилегий*** в отношении поименованных объектов базы данных указанным *пользователям*. Обычно его использует *владелец* таблицы с целью *предоставления доступа* к ней другим *пользователям*. Оператор GRANT имеет следующий формат:

```
<предоставление_привилегий>::=
GRANT { ALL [ PRIVILEGES] | <привилегия>
    [,...n]}
{ [(    имя_столбца [,...n])]
    ON { имя_таблицы |
        имя_просмотра} |
            ON {имя_таблицы |
        имя_просмотра }
            ([имя_столбца
                [,...n])]
    | ON {имя_хранимой_процедуры |
        имя_внешней_процедуры}}
TO {    имя_пользователя | имя_группы |
        имя_роли} [,...n]
[WITH GRANT OPTION ]
[AS {имя_группы | имя_роли }]
```

Параметр <привилегия> представляет собой следующую конструкцию:

```
<привилегия>::=
{SELECT | DELETE | INSERT |
 UPDATE | EXECUTE | REFERENCES }
```

Параметр WITH GRANT OPTION поможет *пользователю*, которому вы предоставляете *права*, назначить *права* на *доступ* к объекту другим *пользователям*. Его использование требует особой осторожности, поскольку при этом *владелец* теряет контроль над *предоставлением прав* на *доступ* другим *пользователям*. Лучше всего ограничить круг *пользователей*, обладающих возможностью *управлять назначением прав*.

Необязательный параметр AS {имя_группы | имя_роли } позволяет указать участие *пользователя* в *роли*, обеспечивающей *предоставление прав* другим *пользователям*.

Единственное *право доступа*, которое может быть *предоставлено* для хранимой процедуры, – *право* на ее выполнение (EXECUTE). Естественно, кроме этого *владелец* хранимой процедуры может просматривать и изменять ее код.

Для функции можно выдать *право* на ее выполнение, а кроме того, выдать *право* REFERENCES, что обеспечит возможность связывания функции с объектами, на которые она ссылается. Такое связывание позволит запретить внесение изменений в структуру объектов, способных привести к нарушению работы функции

В языке SQL для ***отмены привилегий***, *предоставленных пользователям* посредством оператора GRANT, используется оператор REVOKE. С помощью этого оператора могут быть *отменены* все или некоторые из *привилегий*, полученных указанным *пользователем* раньше. Оператор REVOKE имеет следующий формат:

```
<неявное_отклонение_доступа>::=
REVOKE [GRANT OPTION FOR]
```

```
{ALL [ PRIVILEGES]| | <привилегия>
    [,...n]}
{ [(имя_столбца [,...n])] ON
    { имя_таблицы | имя_просмотра}
    | ON {имя_таблицы |
        имя_просмотра }
    [имя_столбца [,...n])]
    | ON {имя_хранимой_процедуры |
    имя_внешней_процедуры}}
TO | FROM {имя_пользователя |
    имя_группы |
    имя_роли}[,...n]
[CASCADE ]
[AS {имя_группы | имя_роли }]
```

Порядок выполнения работы:

Варианты заданий

Вариант	Наименование базы данных, пользователи и их права
1	*БД «Лицензионное программное обеспечение».* Администратор (доступны все таблицы для полного управления) Менеджер (доступны лицензии, CD-диски для полного управления, владельцы для просмотра и редактирования) Оператор (доступны владельцы для полного управления, лицензии, CD-диски для просмотра, добавления)
2	*БД «Студенты МпК».* Администратор (доступны все таблицы для полного управления) Зав.отделением (доступны группы, дисциплины для полного управления, студенты, успеваемость для просмотра и редактирования) Классный руководитель (доступны студенты, успеваемость для полного управления, тарифы, группы, дисциплины для просмотра)
3	*БД «Гостиница».* Администратор базы данных (доступны все таблицы для полного управления) Администратор гостиницы (доступны номерной фонд для полного бронирование, проживание для просмотра) Менеджер (доступны бронирование, проживание для полного управления, номерной фонд для просмотра и добавления)
4	*БД «Товарооборот».* Администратор (доступны все таблицы для полного управления) Менеджер (доступны поставщики, поступление товаров для полного управления, товары, продажа для просмотра) Оператор (доступны товары, продажа для полного управления, поставщики, поступление товаров для просмотра и добавления)
5	*БД «Промышленность региона».* Администратор (доступны все таблицы для полного управления) Менеджер (доступны предприятия, виды налогов для полного управления, налоговые платежи, прибыль для просмотра) Оператор (доступны налоговые платежи, прибыль для полного управления, предприятия, виды налогов для просмотра и добавления)
6	*БД «Пассажирские поезда».* Администратор (доступны все таблицы для полного управления)

	Менеджер (доступны поезда, составы вагонов, расписание для полного управления, перевозки для для просмотра) Кассир (доступны поезда, составы вагонов, расписание для просмотра, пассажиры, продажа билетов для полного управления)
7	*БД «Автопарк».* Администратор (доступны все таблицы для полного управления) Менеджер (доступны типы автобусов, парк, водители для полного управления, перевозки для просмотра) Оператор (доступны перевозки для полного управления, типы автобусов, парк, водители для просмотра и добавления)
8	*БД «Агентство недвижимости».* Администратор (доступны все таблицы для полного управления) Менеджер (доступны риэлторы, недвижимость для полного управления, сделки для просмотра) Оператор (доступны сделки для полного управления, риэлторы, недвижимость для просмотра и добавления)
9	*БД «Авиаперевозки».* Администратор (доступны все таблицы для полного управления) Менеджер (доступны авиапарк, рейсы и тарифы для полного управления, перевозки для для просмотра) Кассир (доступны авиапарк, рейсы и тарифы для просмотра, пассажиры, продажа билетов для полного управления)

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Организация защиты данных в хранилищах

Лабораторная работа № 15
Реализация доступа пользователей к базе данных

Цель: получение практических навыков по освоению реализации доступа пользователей к базе данных

Выполнив работу, Вы будете:
уметь:
- реализовывать доступ пользователей к базе данных.

Материальное обеспечение:
Методические указания для выполнения лабораторных работ, вариант задания, компьютер, программное обеспечение MS SQLServer.

Задание:
Реализовать доступ пользователей к базе данных в СУБД MS SQLServer.

Краткие теоретические сведения:

Стабильная система управления пользователями – обязательное условие безопасности данных, хранящихся в любой реляционной СУБД. В языке SQL не существует единственной стандартной команды, предназначенной для создания пользователей базы данных – каждая реализация делает это по-своему. В одних реализациях эти специальные команды имеют определенное сходство, в то время как в других их синтаксис имеет существенные отличия. Однако независимо от конкретной реализации все основные принципы одинаковы.

Порядок выполнения работы:
Для создания пользователя в среде MS SQL Server следует предпринять следующие шаги:

1. Создать в базе данных учетную запись пользователя, указав для него пароль и принятое по умолчанию имя базы данных (процедура sp_addlogin).
2. Добавить этого пользователя во все необходимые базы данных (процедура sp_adduser).
3. Предоставить ему в каждой базе данных соответствующие привилегии (команда GRANT).

Создание новой учетной записи может быть произведено с помощью системной хранимой процедуры:

```
sp_addlogin
[@login=] 'учетная_запись'
[, [@password=] 'пароль']
[, [@defdb=] 'база_данных_по_умолчанию']
```

После завершения аутентификации и получения идентификатора учетной записи (login ID) пользователь считается зарегистрированным, и ему предоставляется доступ к серверу. Для каждой базы данных, к объектам которой он намерен получить доступ, учетная запись пользователя (login) ассоциируется с пользователем (user) конкретной базы данных, что осуществляется посредством процедуры:

```
sp_adduser
```

[@loginame=] 'учетная_запись'
[, [@name_in_db=] 'имя_пользователя']
[, [@grpname=] 'имя_роли']

Отобразить учетную запись в имя пользователя позволяет хранимая процедура:

sp_grantdbaccess
[@login=] ‘учетная_запись’
[, [@name_in_db=]‘имя_пользователя’]

Пользователь, который создает объект в базе данных (таблицу, хранимую процедуру, просмотр), становится его владельцем. Владелец объекта (database object owner dbo) имеет все права доступа к созданному им объекту. Чтобы пользователь мог создать объект, владелец базы данных (dbo) должен предоставить ему соответствующие права. Полное имя создаваемого объекта включает в себя имя создавшего его пользователя.

Владелец объекта не имеет специального пароля или особых прав доступа. Он неявно имеет полный доступ, но должен явно предоставить доступ другим пользователям.

SQL Server позволяет передавать права владения от одного пользователя другому с помощью процедуры:

sp_changeobjectowner
[@objname=] ‘имя_объекта’
[@newowner=] ‘имя_владельца’

Роль позволяет объединить в одну группу пользователей, выполняющих одинаковые функции.

В SQL Server реализовано два вида стандартных ролей: на уровне сервера и на уровне баз данных. При установке SQL Server создаются фиксированные роли сервера (например, sysadmin с правом выполнения любых функций SQL-сервера) и фиксированные роли базы данных (например, db_owner с правом полного доступа к базе данных или db_accessadmin с правом добавления и удаления пользователей). Среди фиксированных ролей базы данных существует роль public, которая имеет специальное назначение, поскольку ее членами являются все пользователи, имеющие доступ к базе данных.

Можно включить любую учетную запись SQL Server (login) или учетную запись Windows в любую роль сервера.

Роли базы данных позволяют объединять пользователей в одну административную единицу и работать с ней как с обычным пользователем. Можно назначить права доступа к объектам базы данных для конкретной роли, при этом автоматически все члены этой роли наделяются одинаковыми правами.

В роль базы данных можно включить пользователей SQL Server, роли SQL Server, пользователей Windows.

Различные действия по отношению к роли осуществляются при помощи специальных процедур:

- создание новой роли:
- sp_addrole
- [@rolename=] 'имя_роли'
- [, [@ownername=] 'имя_владельца']
- добавление пользователя к роли:
- sp_addrolemember
- [@rolename=] 'имя_роли',
- [@membername=] 'имя_пользователя'
- удаление пользователя из роли:
- sp_droprolemember

- [@rolename=] 'имя_роли',
- [@membername=] 'имя_пользователя'
- удаление роли:
- sp_droprole
- [@rolename=] 'имя_роли'

Форма представления результата:
Отчет по выполненной лабораторной работе

Критерии оценки:
Оценка «отлично» ставится, если задание выполнено верно.
Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.
Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.
Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Организация защиты данных в хранилищах

Лабораторная работа № 16
Мониторинг безопасности работы с базами данных

Цель: получение практических навыков по освоению операций мониторинга безопасности работы с базами данных

Выполнив работу, Вы будете:

уметь:

- выполнять мониторинг безопасности работы с базами данных.

Материальное обеспечение:

Методические указания для выполнения лабораторных работ, вариант задания, компьютер, программное обеспечение MS SQLServer.

Задание:

Выполнить мониторинг безопасности работы с базами данных.

Краткие теоретические сведения:

Методы мониторинга баз данных

Существует три метода мониторинга работы баз данных: собственный аудит, установка централизованного агента и сетевой мониторинг. У большинства баз данных есть собственный механизм аудита, и системный администратор может отслеживать события, связанные с безопасностью базы данных. События, представляющие интерес: начало/окончание работы, доступ к объектам и выполняемые запросы.

Порядок выполнения работы:

Ведение журнала базы данных должно осуществляться по следующим видам деятельности:

- Добавление, изменение, приостановка действия и удаление учетных записей пользователей.
- Изменение прав у учетных записей пользователей (права доступа учетной записи).
- Повышение привилегий.
- Изменения владельца объектов.
- Начало/окончание сеанса, неудачные попытки авторизации под учетными записями администратора (учетными записями администраторов баз данных), учетными записями приложений и учетными записями, используемыми для прямого доступа к базе.
- Изменение паролей.
- Изменение политики безопасности базы данных / изменение настроек:
    - Режимы аутентификации.
    - Управление паролями.
    - Запрет/разрешение удаленного доступа.
    - Запрет/разрешение собственного аудита.
- Изменение настоек системы аудита и попытки изменения или удаления логов аудита или логов баз данных.
- Транзакции, связанные с конфиденциальными данными, на основе требований владельца данных.
- Санкционированный доступ к конфиденциальным ресурсам, на основе требований владельца ресурсов.

- Неудачные попытки доступа к конфиденциальным ресурсам, на основе требований владельца ресурсов.

- Неудачное выполнение SQL-запросов (из-за отсутствия объекта или недостаточности привилегий).

- Внесение изменений в схему базы данных (Команды DDL (Data Definition Language, язык описания данных)).

- Создание/восстановление резервных копий.

- Запуск/остановка базы данных.

- Попытки использования средств операционной системе через базу данных (выполнение команд, чтение/изменение файлов и настроек).

- В журнале должно быть достаточно информации для того, чтобы была возможность выяснить, какие произошли события, и кто был их инициатором:

  - Тип события.
  - Когда произошло событие.
  - Учетная запись, связанная с событием.
  - Программа или команда, ставшая инициатором события (точное SQL-выражение).
  - Имена таблиц, к которым осуществлялся доступ (если возможно).
  - Имя хоста или IP-адрес, с которого произошло соединение пользователя.
  - Статус попытки (успех/неудача).

Мониторинг должен сработать при наступлении следующих событий:

- Несогласованные добавления и изменения учетных записей пользователей.

- Случаи многократных неудачных попыток ввода паролей по множеству учетных записей за короткий промежуток времени (что может свидетельствовать о реализации хакерских намерений).

- Случаи попыток неудачного доступа к базе данных от имени учетной записи, у которой нет прав доступа.

- Попытки получить список пользователей и паролей.

- Все попытки прямого доступа к базе данных от имени учетных записей, доступ которым разрешен только из приложения.

- Использование нестандартных утилит (например, Excel, Access) для прямого доступа к СУБД.

- Использование «служебных программ» (например, Toad) для прямого доступа к СУБД.

- Использование Application ID (ApplID) из источника отличного от того, который закреплен за владельцем приложения (на основе имени хоста или IP-адреса).

- Неудачные входы в систему, попытки остановить ведение журнала и уничтожить логи.

- Попытки доступа к средствам ОС через базу данных.

- Обнаружение известных видов атак (например, переполнение буфера, отказ в обслуживании, SQL-инъекция).

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Организация защиты данных в хранилищах

Лабораторная работа № 17
Установка приоритетов

Цель: получение практических навыков по освоению операций установки приоритетов

Выполнив работу, Вы будете:

уметь:

- выполнять операции установки приоритетов.

Материальное обеспечение:

Методические указания для выполнения лабораторных работ, вариант задания, компьютер, программное обеспечение: MS SQLServer.

Задание:

Установить приоритет для запланированных задач.

Порядок выполнения работы:

1. Выполнить редактирование xml-файла.
2. Настроить приоритет с помощью PowerShell.
3. Использовать групповые политики.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Организация защиты данных в хранилищах

Лабораторная работа №18
Развертывание контроллеров домена

Цель: получение практических навыков по освоению операций развертывания контроллеров домена

Выполнив работу, Вы будете:

уметь:

- выполнять операции развертывания контроллеров домена.

Материальное обеспечение:

Методические указания для выполнения лабораторных работ, вариант задания, компьютер, программное обеспечение: MS SQLServer.

Задание:

Выполнить установку контроллера домена и сервера базы данных.

Краткие теоретические сведения:

Перед установкой корневого сервера сертификации или сервера лицензирования убедитесь, что установлены соответствующие домен и поддержка баз данных с помощью Active Directory и сервера базы данных

Порядок выполнения работы:

1. Установите на компьютер, соответствующий требованиям службы управления правами к аппаратуре, но еще не подключенный к сети, операционную систему Windows 2000 Server с пакетом обновления 3 или более поздним или Windows Server 2003. Для системного раздела используйте файловую систему NTFS.
2. Создайте подключение к сети, которое обеспечивает подключение к Интернету, но изолировано от рабочей среды.
3. Назначьте статический IP-адрес для этого компьютера.
4. Зарегистрируйтесь в системе как локальный администратор.

- Щелкните **Пуск**, затем выберите **Выполнить**, в поле Открыть введите **dcpromo**, а затем нажмите кнопку **ОК**.

- После запуска мастера установки Active Directory выполните все необходимые действия для создания нового домена в новом лесу и принятия параметров по умолчанию, кроме следующих: укажите имя домена, например contoso.com; разрешите мастеру настроить DNS-сервер на компьютере; выберите **Разрешения, совместимые только с серверами Windows**, если все контроллеры домена работают под управлением Windows. Укажите надежный пароль для локального администратора.

- Перезагрузите компьютер после появления соответствующего запроса.

- Проверьте режим работы, открыв оснастку **Active Directory - пользователи и компьютеры**. Для этого щелкните правой кнопкой мыши имя домена, выберите **Свойства**, а затем проверьте значение параметра в поле **Режим работы домена**.

5. Создайте учетную запись пользователя домена, которая будет использоваться в качестве учетной записи службы управления правами, например ContosoRMS@contoso.com. Укажите надежный пароль. Не забудьте указать адрес электронной почты пользователя. Если в Active Directory не указан адрес электронной почты, пользователь не сможет получать лицензии и сертификаты от службы управления правами. Примечание: Учетная запись службы управления

правами не должна совпадать с учетной записью домена, которая использовалась для установки службы управления правами. Использовать групповые политики.

6. Зарегистрируйтесь на сервере, на который предполагается установить базу данных. Если это тот же сервер, что и контроллер домена, необходимо зарегистрироваться на нем с правами администратора домена.

- Для установки программного обеспечения сервера базы данных следуйте инструкциям, поставляемым вместе с программным обеспечением.

- Опыт работы с базами данных подсказывает следующие моменты установки сервера:

  – Укажите имя для учетной записи системного администратора базы данных и название организации.
  – Используйте надежный пароль системного администратора.
  – Используйте встроенные методы проверки подлинности Windows.

- Убедитесь, что служба сервера базы данных остановлена.

- Установите все программные обновления для сервера базы данных. Когда появится запрос на ввод пароля, используйте тот же пароль, который был указан во время установки.

- Перезагрузите компьютер. Проверьте, что служба базы данных запущена.

- Проверьте, что учетные записи пользователей Active Directory имеют действительные атрибуты адреса электронной почты.

- Убедитесь, что пользователь домена, который будет администрировать службу управления правами (и подготавливать корневой сервер сертификации и сервер лицензирования) обладает достаточными разрешениями для сервера базы данных. При использовании SQL Server в качестве сервера базы данных можно добавить регистрационный идентификатор для пользователя, использующего оснастку **SQL Server Enterprise Manager**. Находясь в этой оснастке, разверните сервер и группу сервера, затем разверните пункт **Security**. Выберите **Logins item**, добавьте новое регистрационное имя для доменной учетной записи пользователя, откройте вкладку **Server Roles** и установите флажок **Server Administrators**

- Проверьте правильность настроек обозревателя и сервера (включая необходимые настройки прокси-сервера), TCP/IP и LMHOSTS/HOSTS для доступа в Интернет.

- Загрузите и установите последние обновления для программного обеспечения, установленного на этом компьютере.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Организация защиты данных в хранилищах

Лабораторная работа №19
Мониторинг сетевого трафика

Цель: получение практических навыков по освоению мониторинга сетевого трафика

Выполнив работу, Вы будете:
уметь:
- выполнять мониторинг сетевого трафика.

Материальное обеспечение:
Методические указания для выполнения лабораторных работ, вариант задания, компьютер, программное обеспечение: MS SQLServer.

Задание:
Есть дамп трафика одной из компаний, где проанализирована производительность CRM системы. В трафике убрать всю коммерческую информацию.

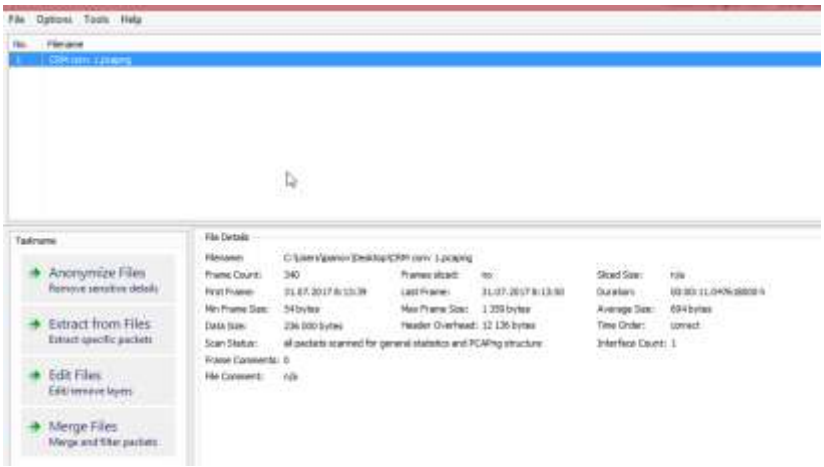
Краткие теоретические сведения:
Используя программу TraceWranler, можно спокойно передавать дамп с трафиком и не бояться за какую-либо утечку.

Интерфейс программы очень простой и позволит выполнять следующие изменения с файлами:

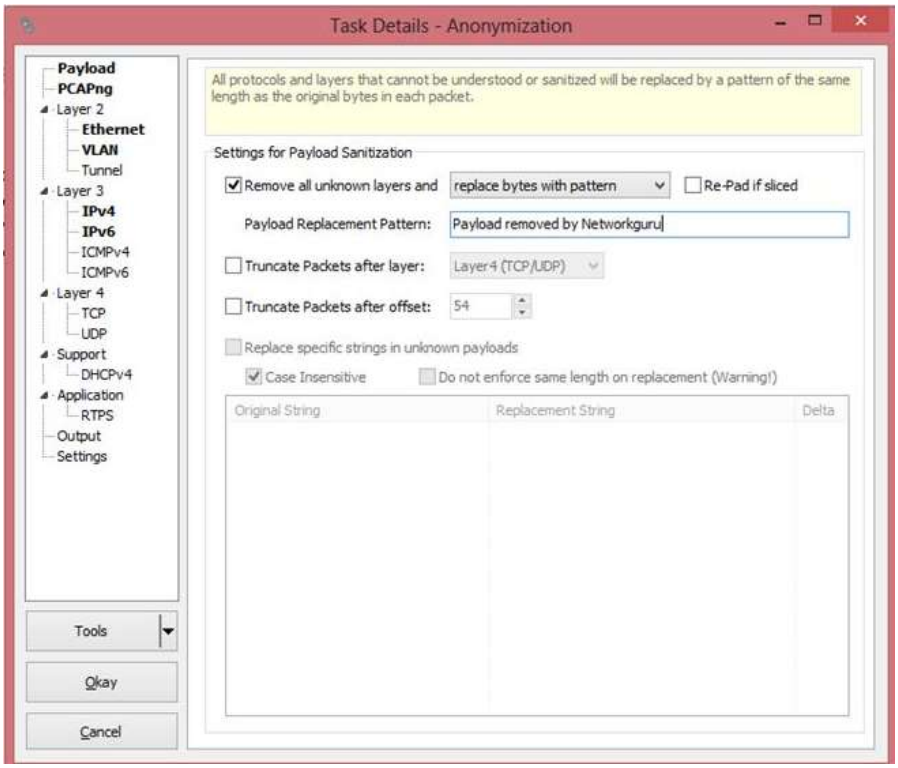
- провести его полное обезличивание дампа трафика — т. е. убрать или заменить всю коммерческую информацию;
- быстро проанализировать несколько файлов (иногда и больших размеров) и вывести их в виде списка с возможностью различной сортировки по адресам, сеансам обмена, временам отклика и настройки фильтров, например, для решения следующей задачи;
- вырезать отдельный сеанс связи и внести его в один файл выдачи;
- вырезать из пакетов лишнюю информацию, например: MPLS метки, заголовки GRE, GTP, VLAN и т.д.

Поддерживаются форматы файлов – PCAP и PCAPng, который сейчас используется в снифферe Wireshark.

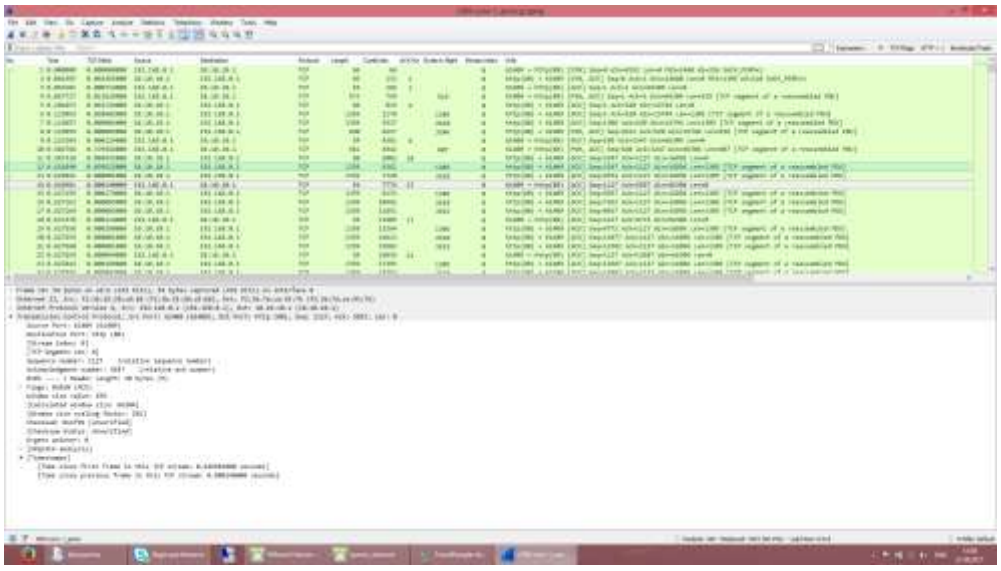
Порядок выполнения работы:
1. TraceWranler. Загружаем файл с трафиком CRM conversation в программу:



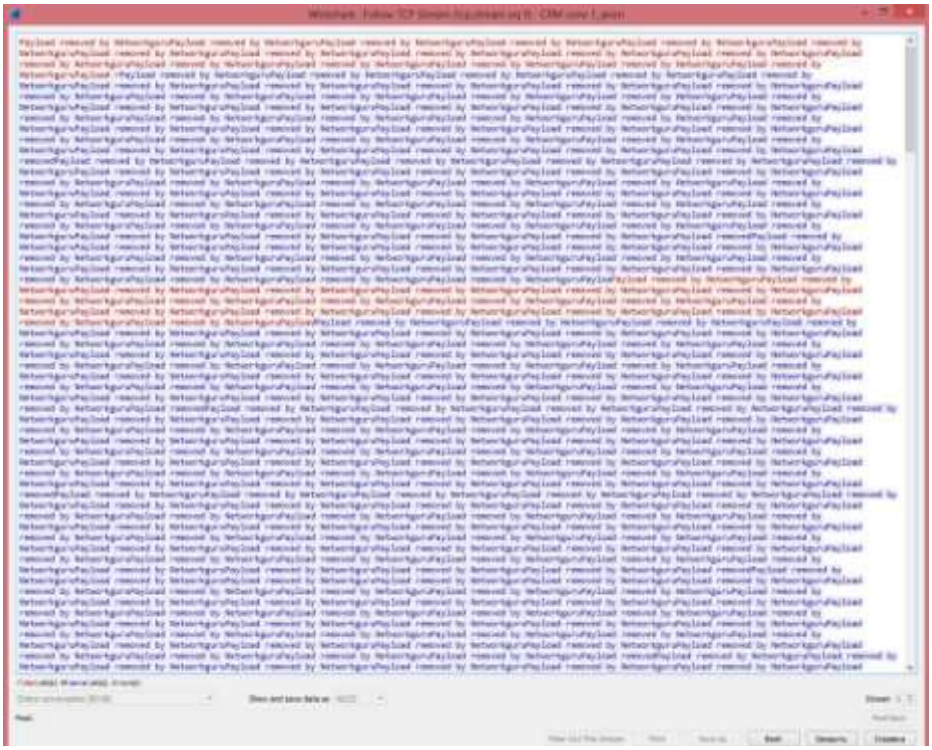
2. Нажимаем кнопку «Anonymize Files» и в настройках указываем поля, которые мы хотим удалить или заменить:



3. В данном случае мы отрежем полезную нагрузку и заменим ее словами, что она удалена, а также заменим IP адреса на фиктивные.
4. На выходе получаем файл с фейковыми IP адресами:



5. При этом все временные метки и порты останутся как в родном трафике. Зато при попытке посмотреть полезные данные, нажав «Follow TCP Stream», получаем:



Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.