Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет
им. Г.И. Носова»
Многопрофильный колледж

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

по ПМ.02 Разработка и администрирование баз данных

МДК.02.02Технология разработки и защиты баз данных

для студентов специальности
09.02.03 Программирование в компьютерных системах

(базовой подготовки)

Магнитогорск, 2017

ОДОБРЕНО:
Предметно-цикловой комиссией Информатика и вычислительная техника

Председатель: И.Г.Зорина

Протокол № 7 от 14 марта 2017

Методической комиссией МпК

Протокол №4 от «23» марта 2017г

Составитель:
преподаватель ФГБОУ ВО «МГТУ им. Г.И. Носова» МпК Ирина Геннадьевна Зорина

Методические указания по выполнению практических работ разработаны на основе рабочей программы ПМ.02 Разработка и администрирование баз данных, МДК.02.02 Технология разработки и защиты баз данных.

Содержание практических работ ориентировано на формирование общих и профессиональных компетенций по программе подготовки специалистов среднего звена по специальности 09.02.03 Программирование в компьютерных системах.

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

Важную часть теоретической и профессиональной практической подготовки обучающихся составляют практические занятия.

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

Ведущей дидактической целью практических занятий является формирование практических умений - профессиональных (умений выполнять определенные действия, операции, необходимые в последующем в профессиональной деятельности), необходимых в последующей учебной деятельности по профессиональным модулям.

В соответствии с рабочей ПМ.02 Разработка и администрирование баз данных, МДК.02.02 Технология разработки и защиты баз данных предусмотрено проведение практических занятий.

В результате их выполнения, обучающийся должен:

уметь:

- создавать объекты баз данных в современных системах управления базами данных и управлять доступом к этим объектам;
- работать с современными case-средствами проектирования баз данных;
- формировать и настраивать схему базы данных;
- разрабатывать прикладные программы с использованием языка SQL;
- создавать хранимые процедуры и триггеры на базах данных;
- применять стандартные методы для защиты объектов базы данных.

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

ПК.2.1. Разрабатывать объекты базы данных.

ПК.2.2. Реализовывать базу данных в конкретной системе управлении базами данных (СУБД).

ПК.2.3. Решать вопросы администрирования базы данных.

ПК.2.4. Реализовывать методы и технологии защиты информации в базах данных.

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

ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.

ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

ОК 3. Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность.

ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.

ОК 5. Использовать информационно-коммуникационные технологии в профессиональной деятельности.

ОК 6. Работать в коллективе и в команде, эффективно общаться с коллегами, руководством, потребителями.

ОК 7. Брать на себя ответственность за работу членов команды (подчиненных), за результат выполнения заданий.

ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.

ОК 9. Ориентироваться в условиях частой смены технологий в профессиональной деятельности.

Выполнение обучающимися практических работ по ПМ.02 Разработка и администрирование баз данных, МДК.02.02 Технология разработки и защиты баз данныхнаправлено на:

- обобщение, систематизацию, углубление, закрепление, развитие и детализацию полученных теоретических знаний по конкретным темам междисциплинарного курса;

- 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;

- 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 занятия проводятся после соответствующей темы, которая обеспечивает наличие знаний, необходимых для ее выполнения.

2МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Тема 2.1. Разработка и проектирование баз данных

Практическая работа № 1
Проектирование базы данных с использованием современных case-средств

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: MySQLWorkbench.

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 «Морские перевозки».* Таблица «Суда»: номер судна; название; водоизмещение; скорость хода. Таблица «Грузы»: код груза; груз; единица измерения. Таблица «Рейсы»: код рейса; номер судна; код груза; количество груза; порт отправления; дата отправления; порт назначения; дата прибытия; доход за рейс, р.; затраты за рейс, р
2	*БД «Абитуриенты».* Таблица «Специальности»: шифр специальности; специальность. Таблица «Анкета»: номер анкеты; шифр специальности; Ф.И.О.; дата рождения; оконченное среднее учебное заведение (наименование, номер); дата окончания; знак отличия (золотая (серебряная) медаль или красный диплом); город; адрес; телефон. Таблица «Дисциплины»: шифр дисциплины; наименование дисциплины. Таблица «Результаты экзаменов»: номер анкеты; шифр дисциплины; оценка
3	*БД «Зарплата».* Таблица «Должности»: код должности; должность. Таблица «Тарифы»: код должности; разряд; ставка, р./ч. Таблица «Работники»: отдел; код должности; разряд; табельный номер; Ф.И.О. Таблица «Табель»: год; месяц; табельный номер; количество отработанных часов; дата начисления зарплаты
4	*БД «Оптовая база».* Таблица «Товары»: код товара; товар; единица измерения; цена. Таблица «Склад»: дата поступления, код товара; количество. Таблица «Заявки»: номер заявки; организация; код товара; требуемое количество товара. Таблица «Отпуск товаров»: номер заявки; код товара; дата отпуска товара; количество отпущенного товара
5	*БД «Библиотека».* Таблица «Книги»: жанр; шифр книги; автор; название; год издания; количество экземпляров. Таблица «Читатели»: номер читательского билета; Ф.И.О.; адрес. Таблица «Выдачи»: дата выдачи; номер читательского билета; шифр книги; количество экземпляров; срок возврата; фактическая дата возврата
6	*БД «ГИБДД».* Таблица «Автомобили»: модель автомобиля; номер двигателя; номер кузова; серия и номер технического паспорта; государственный номер автомобиля. Таблица «Владельцы»: государственный номер автомобиля; Ф.И.О.; адрес, серия и номер водительского удостоверения. Таблица «Виды нарушений»: код нарушения; вид нарушения. Таблица «Нарушители»: дата нарушения; код нарушения; государственный номер автомобиля; размер штрафа, р.

7	*БД «Соревнования».* Таблица «Команда»: спортивная организация; шифр команды, название команды. Таблица «Участники»: шифр команды; номер участника; Ф.И.О. Таблица «Старт»: стартовый номер; номер участника; время старта; отметка о невыходе на старт. Таблица «Финиш»: порядковый номер финиширования; стартовый номер; время финиша; отметка о сходе с дистанции
8	*БД «Перевозки».* Таблица «Транспорт»: модель автомобиля; государственный номер автомобиля; удельный расход топлива, л/100 км. Таблица «Заявки»: номер заявки; дата; пункт отправления; пункт назначения; груз; единица измерения; количество груза. Таблица «Доставка»: номер заявки; государственный номер автомобиля; дата отправления; дата возвращения; пройденное расстояние; расход топлива
9	*БД «Сессия».* Таблица «Кафедры и дисциплины»: номер кафедры; кафедра; шифр дисциплины; наименование дисциплины; вид аттестации (зачет или экзамен). Таблица «Преподаватели»: номер кафедры; табельный номер преподавателя; Ф.И.О. Таблица «Студенты»: шифр студента; Ф.И.О. Таблица «Оценки»: дата; шифр дисциплины; табельный номер преподавателя; шифр студента; оценка
10	*БД «Телефонная компания».* Таблица «Тарифы»: код тарифа; вид тарифа; цена, р./мин. Таблица «Виды льгот»: код льготы; вид льготы; размер, %. Таблица «Абоненты»: лицевой счет; телефон; Ф.И.О.; адрес; код льготы. Таблица «Платежи»: лицевой счет; код тарифа; дата оплаты; сумма платежа; дата отключения за неуплату
11	*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
12	*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
13	*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
14	*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
15	*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
16	*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
17	*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
18	*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
19	*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 MySQLWorkbench.

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 2.1. Разработка и проектирование баз данных

Практическая работа № 2
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 практических работ, вариант задания, компьютер, программное обеспечение: MySQLWorkbench.

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

Краткие теоретические сведения:

Нормализация баз данных
Нормальные формы – эторекомендациипо 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.
- Нормализациипредотвращает 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 представлениесущностисистемы, 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 социальногострахования 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 суррогатныйпервичный 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. Полеbrandмогло бы быть выделено в отдельную таблицу. Также, как и полеtype(модель), которое также могло бы быть выделено в отдельную таблицу, которая бы имела связь многие-к-одному с таблицейbrandпотому, что у бренда могут быть разные модели.

Колонкаstoreсодержит 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 автомобилейсвязью 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	Buikslote	76	1031 AB	020387
4	The Hague	Neherstre	82	2491JJ	070387

В примере выше таблицасагимеет внешний ключ – ссылку на таблицытypeиstore. Столбец brand исчез потому, что на бренд есть неявная ссылка через таблицуtype. 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_originв таблицеbrandпокадубликатов нет потому, что есть только четыре бренда из разных стран. Внимательный разработчик базы данных должен выделить названия стран в отдельную таблицуcountry.

И даже сейчас нельзя удовлетвориться результатом потому, что могжно бы выделить полecolorв 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 возможновыбиратьцвет машины из заранее заданного списка. В этом случае нужно хранить все возможные цвета в базе данных. Дажеесли 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 странзитивными 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.

Такуютранзитивную 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 отчетана 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 2.1. Разработка и проектирование баз данных

Практическая работа № 3
Построение и настройка схемы базы данных

Цель: получение практических навыков по освоению операций настройки схемы базы данных.

Выполнив работу, Вы будете:

уметь:

- создавать базу данных;
- устанавливать связи между таблицами;
- обеспечивать целостность данных;
- формировать и настраивать схему базы данных.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MySQLWorkbench.

Задание:

Создать схему базы данных по заданному варианту.

Порядок выполнения работы:

1. Разработать логическую модель схемы по своему варианту.
2. Посмотреть и проанализировать физическую модель.
3. Сгенерировать скрипт создания таблиц.
4. Используя сгенерированный скрипт создать базу данных в выбранной СУБД.
5. Настроить схему базы данных.

Форма представления результата:

Отчет по выполненной практической 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 2.2. Реализация баз данных в конкретной СУБД

Практическая работа № 4, 5, 6
Создание базы данных. Редактирование и модификация таблиц

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 практических работ, вариант задания, компьютер, программное обеспечение: MSAccess, MySQL, MSSQLServer, MySQLWorkbench, PhpMyAdmin.

Задание 1:

Создать базу данных Туризм и таблицы базы данных в СУБД MSAccess.

Краткие теоретические сведения:

Связи ***автоматические*** устанавливаются **Мастером подстановок.** Посмотреть, установить, отредактировать связи можно командой ***Работа с базой данных – Схема данных.***

Если связи устанавливаются первично, то откроется окно ***Таблицы,*** а если повторно, то окно ***Связи.*** Двойной щелчок на нужной таблице позволит перенести их в окно ***Связи.***

Для установки связи между таблицами ***вручную*** нужно перетянуть связываемое поле из главной таблицы и наложить его на соответствующее поле подчиненной таблицы.

Удаление и изменение связей производится с помощью контекстного меню на линии связи, а также клавишей DEL.

В окне ***Схема данных*** двойной щелчок по линии связи позволит открыть окно **Связи.** В нем можно установить флажок у опции ***Целостность данных***, линия связи станет гораздо темнее и появятся значки «1» и «∞», означающие отношение «один» или «многие».

Если система определила тип связи (в нижней части диалогового окна) «один-к-одному» или «один-ко-многим», то можно поставить флажок «Поддерживать целостность данных».

Целостность данных – это набор правил, защищающих данные от случайных изменений или удалений с помощью механизма поддержки корректности связей между связанными таблицами.

Если связь определена и система взяла на себя поддержку целостности данных, то при просмотре главной таблицы (отношение «один») слева, рядом с полосой выделения появится колонка со знаками «+». Щелчок на «+» позволит открыть подчиненную таблицу (отношение «много» или «один»).

Порядок выполнения работы:

1. Создайте в своей рабочей папке папку с именем *База данных*.
2. Запустите ***MSAccess***. Создайте в созданной папке новую базу данных с именем ***Туризм***.

Создание таблиц с помощью ***Конструктора***

3. Создайте таблицу **Сотрудники**в режиме Конструктора. Наименования и типы полей представлены в приведенной таблице.

Название поля	Тип данных
Код сотрудника	Числовой

ФИО	Текст
Должность	Текст
Дата найма	Дата/Время
Дата рождения	Дата/Время
Домашний телефон	Текст
Адрес	Текст
Размер оклада	Числовой

4. Для поля ***Домашний телефон***задайте маску, набрав, например, следующий шаблон (999) 999-99-99.
5. Для поля ***Оклад***задайте условие, что он больше 5000 р., но не больше 10000. Для этого в свойстве «Условие на значение» установите (>5000) AND (<10000). Предусмотрите выдачу сообщения при ошибке ввода данных.
6. Установите для ***Даты рождения*** и ***Даты найма*** маску ввода. Используйте краткий формат даты.
7. Создайте первичный ключ.
8. Перейдите в режим просмотра таблицы.
9. Спрячьте некоторые столбцы. Сделайте их опять видимыми командами **Контекстное меню — Скрыть/Показать столбцы.**
10. Зафиксируйте столбцы, содержащие фамилию и имя.Освободите столбцы.
11. Поменяйте тип шрифта и его начертание (**Формат — Шрифт**).
12. Закройте окно таблицы ***Сотрудники***, сохранив изменения.
13. Создайте новые таблицы **Клиенты** и**Страны**. В качестве первичного ключа задайте ***Код Клиента*** и ***КодТура***.

Название поля	Тип данных
Код клиента	Числовой
Название клиента	Текст
Контактное лицо	Текст
Признак группы	Логический
Телефон	Текст
Адрес	Текст

*Использование **Мастера подстановок***

14. Создайте в режиме **Конструктора** таблицу ***Договоры***, которая должна иметь следующие поля:

Название поля	Тип данных
Номер договора	Число
Код клиента	Числовой
Код тура	Число
Дата начала тура	Дата/Время
Дата окончания тура	Дата/Время
Число туристов	Числовой
Цена тура	Денежный
Дата платежа	Дата/Время
Код Сотрудника	Числовой

Поля *Код сотрудника*, *Код клиента*, *Кодтура* являются полями подстановки. Для их задания используется Мастер подстановок:

- в Типе данных поля *Код сотрудника* раскрыть список типов и выбрать **Мастер подстановок**;
- указать, что столбец подстановки получает свои значения из таблицы ***Сотрудники***;
- выбрать поля *Код сотрудника* и *Фамилия*;
- установить мышью подходящую ширину столбца;

- согласиться с предлагаемой подписью столбца подстановок *Фамилия*;
- сохранить таблицу с именем ***Договоры***.

Аналогично для подстановки *Кода клиента*и *Кода тура*вызывается **Мастер подстановок**. При этом для *Кода клиента*выбираем поля *Код клиента*и *Название клиента*из таблицы ***Клиенты***, а для *Кода тура*— поля *Код тура*и *Страна*из таблицы ***Страны***.

19. Просмотрите и проанализируйте уже установленные при работе с **Мастером подстановки**связи в окне ***Схема данных***.
20. В окне***Схема данных*** двойным щелчком по линии связи откройте окно Связи и установите *Целостность данных*, *Каскадное обновление данных*, *Каскадное удаление данных*.
21. Введите в таблицы 10 разнообразных записей. В таблице ***Сотрудники****осуществите* ввод заведомо некорректных данных для проверки работоспособности условия на значение.
22. Просмотрите главную таблицу каждой связи (с помощью «+») и вызовите *подчиненную* таблицу для каждой записи.

Задание 2:

Для своего варианта создать базу данных в СУБДMySQL. Скрипт на создание сохранить в файле.

Порядок выполнения работы:

Пример создания базы данных

1. Создать базу данных:

```
create database sales;
```

2. Выбрать только что созданную базу:

```
use sales;
```

3. Создать таблицу `product`в выбранной базе данных:

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

4. Создать таблицу `customers`в выбранной базе данных:

```
create table customers(
id serial,
fiovarchar(100) not null,
address varchar(200),
phone varchar(20),
  rating int)
  Engine InnoDB character set utf8;
```

5. Создать таблицу `orders`в выбранной базе данных:

```
create table orders(
id serial,
date_ord date,
qty int,
id_prod bigint unsigned not null,
id_cust bigint unsigned not null)
  Engine InnoDB character set utf8;
```

Задание 3:

Создать базу данныхпо учету успеваемости студентов в СУБДMSSQLServer.

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 "*SQLServerManagemen* Studio". Дляэтоговыбираем*пункт***"Всепрограммы\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 появившемсяменювыберитепункт**"Создать базу данных"**(*New Database*). Появится окно настроек параметров файла данных новойбазы данных**"Создать базу данных"**. В левой части окна настроек имеется список**"Выбор страницы"** (*Select a page*). Этотсписокпозволяет переключаться между группами настроек.

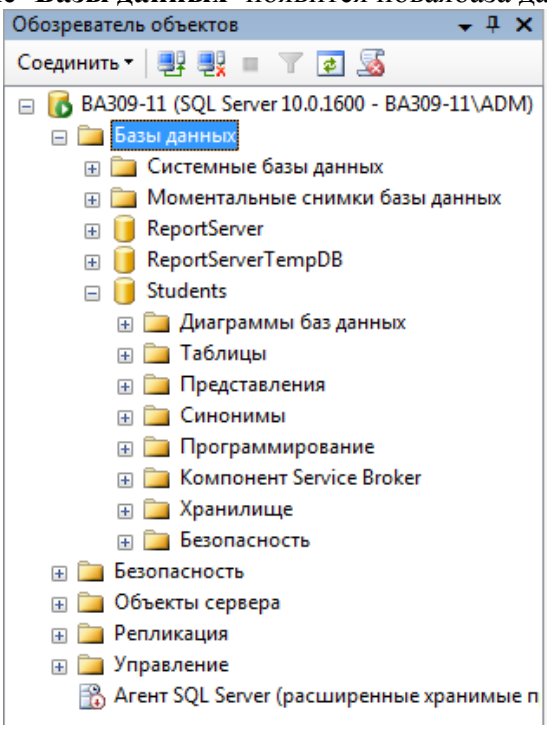
Настроим основные настройки**"Общие"**. Для выбора основных настроек нужно просто щелкнуть мышью по пункту **" Общие "**в списке**"Выбор страницы"**. В правой части окна**" Создать базу данных "**появятся основные настройки.

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

В нашем случае мы оставим все основные настройки без изменений.

Для принятия всех настроек исоздание файладанных и журнала транзакций нашейбазы данныхв 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 новаябаза данных**"Students".**



Для переименованияБДнеобходимо в обозревателе объектов щелкнуть по нейПКМи в появившемсяменювыбратьпункт**"Rename"**. Для удаления в это жеменювыбираемпункт**"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.

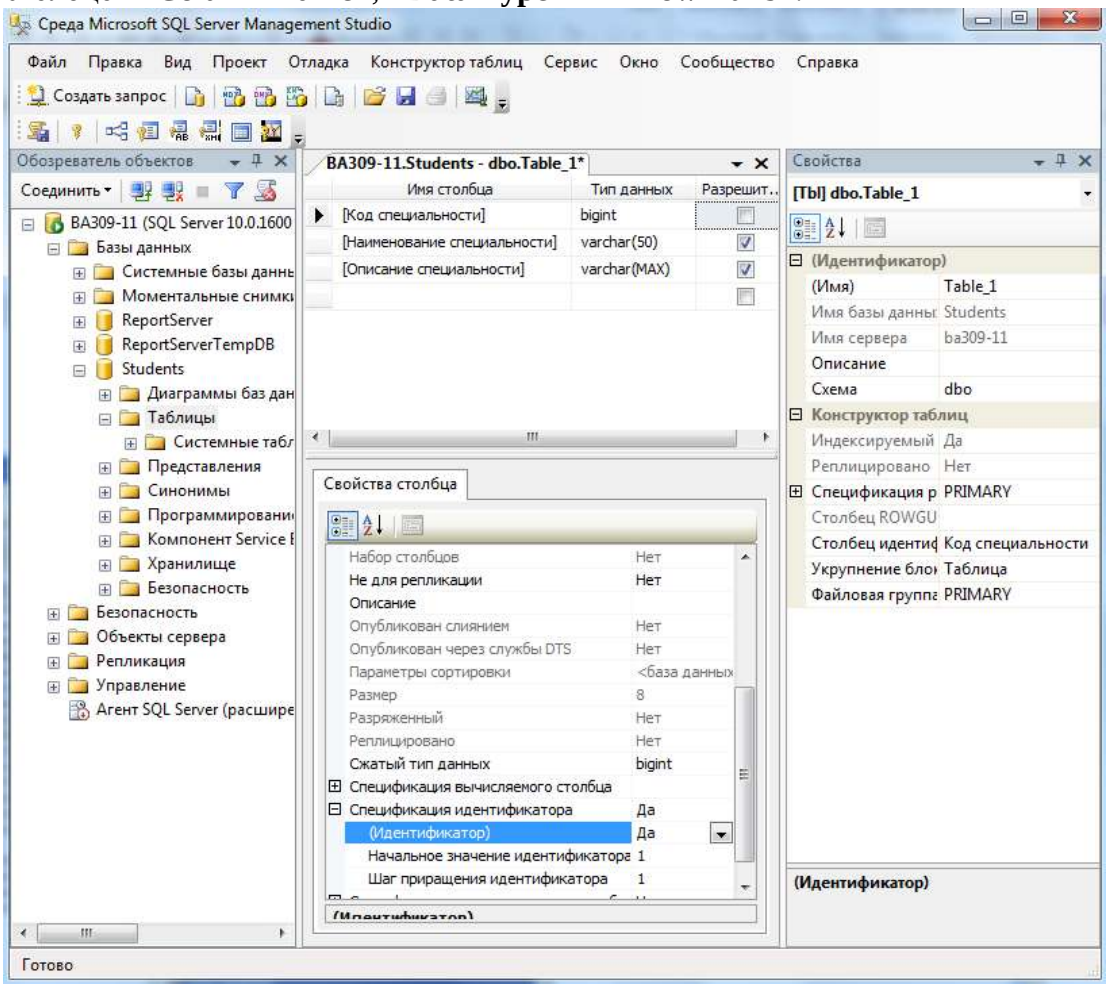
Создадим таблицу **"Специальности"**. Для этого щелкните **ПКМ** по папке **" Таблицы "** и в появившемся меню выберите пунк**"Создать таблицу…"**. Появится окно создания новой таблицы.

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

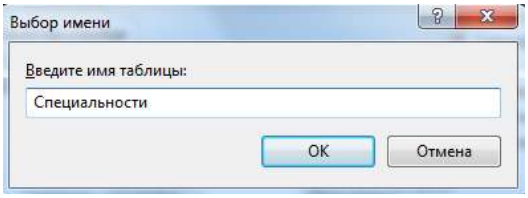
идентификатора). Свойство **"(Is Identity)"** (Идентификатор) установите в значение**"Yes"** (Да). Задайте свойства **"Identity Increment"** (Шаг приращения идентификатора) и **"Identity Seed"** (Начальное значениеидентификатора) равными 1. Эти настройки показывают, что значение поля **"Код специальности"** 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 полеключевое.

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"**.

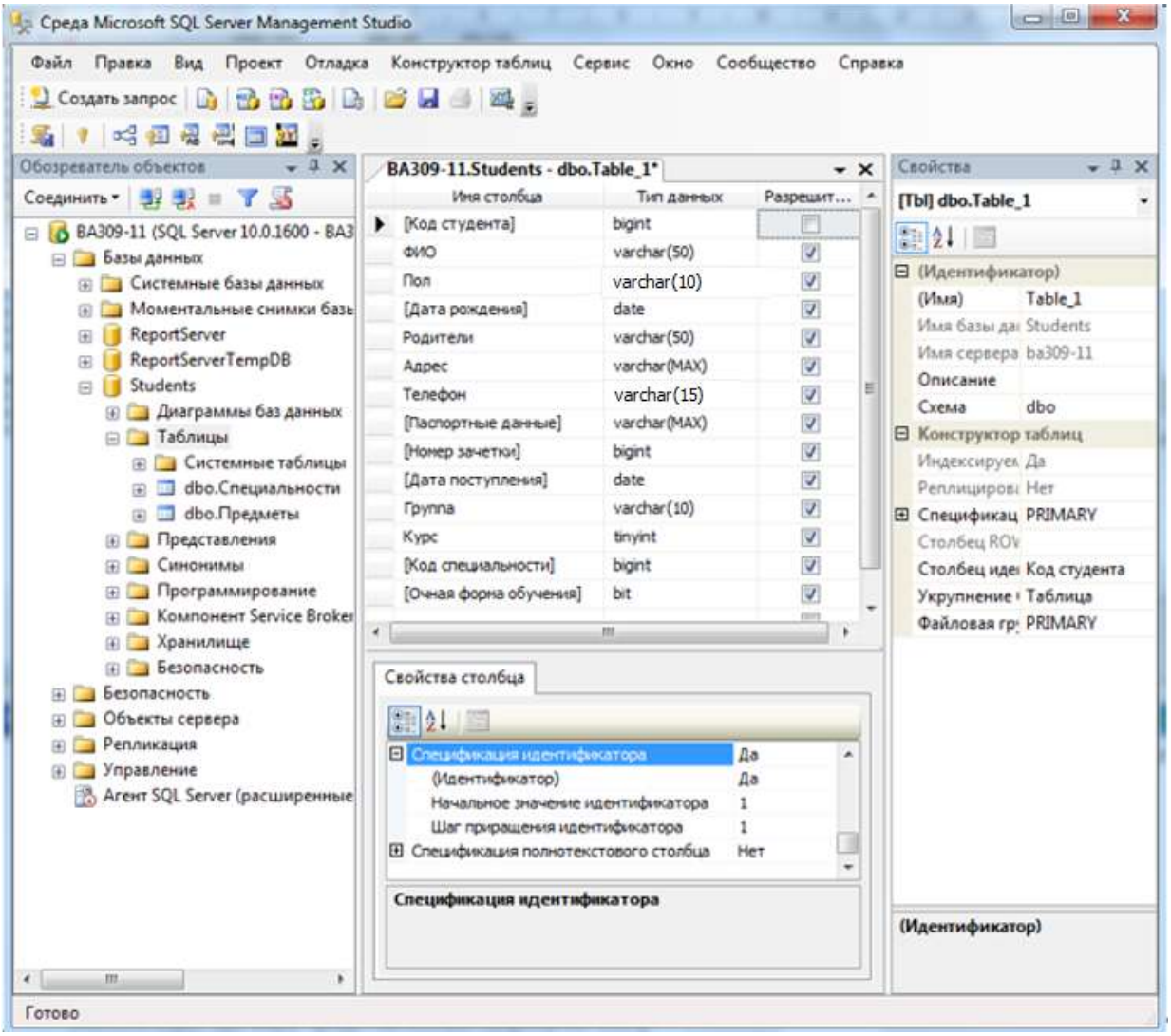
В обозревателе объектов таблица**"Специальности"** отображается как **"dbo.Специальности"**. Префикс "dbo" обозначает, что таблица является объектом БД (DataBaseObject). В дальнейшем при работе с объектами БДпрефикс "dbo" можно опускать.

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.

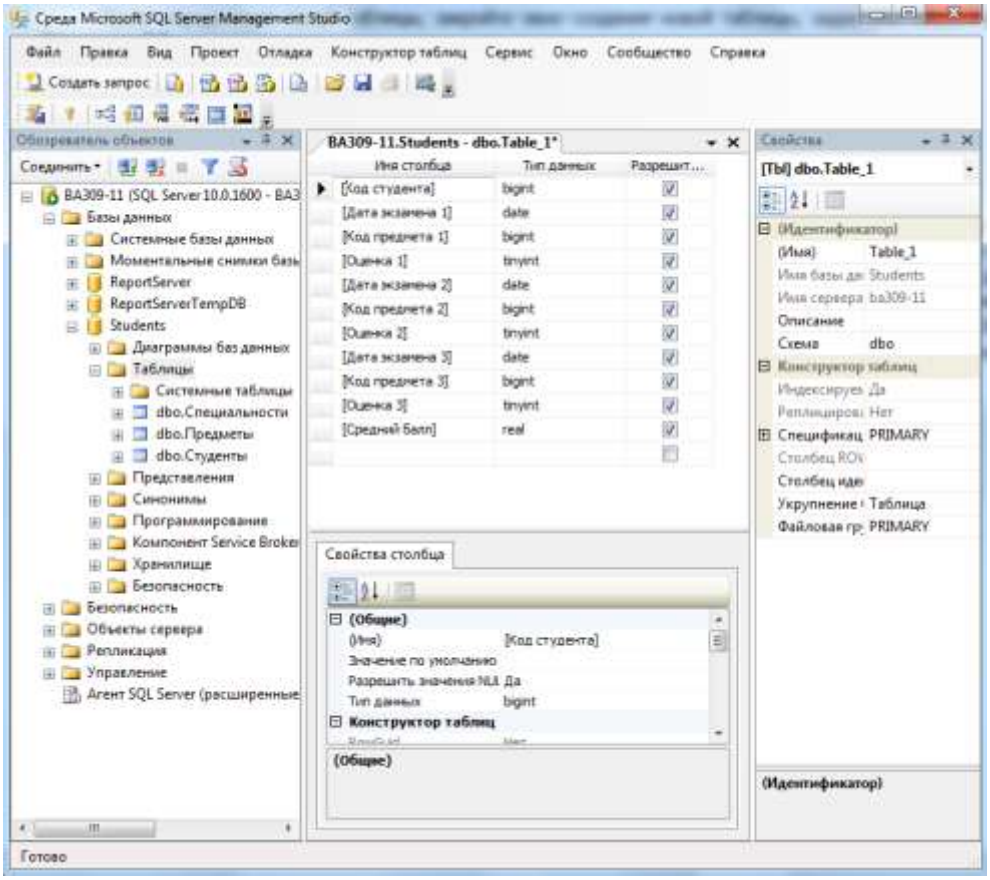


Рассматривая поля новой таблицы можно прийти к следующим выводам:

- 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 $-2^{63}\ldots+2^{63}$ (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, этатаблица 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 являютсяцелочисленными *(*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. Задайтетип данных для этого поля "tinyint". Наконец,поле**"Средний балл"** хранит дробные числа и имеет тип "real".

Закройте окно создания новой таблицы, задавимя таблицы как "Оценки".

Форма представления результата:

Отчет по выполненной практической 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 2.2. Реализация баз данных в конкретной СУБД

Практическая работа № 6
Создание ключевых полей. Установление и удаление связей между таблицами

Цель: получение практических навыков по созданию ключевых полей, установлению связей между таблицами

Выполнив работу, Вы будете:

уметь:

- создавать первичные и внешние ключи.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MySQL,MSSQLServer,PhpMyAdmin.

Задание 1:

По своему варианту базы данных в СУБД MySQLдля каждой таблицы создайте первичные и внешние ключи.

Краткие теоретические сведения:

После того как определены все столбцы таблицы, можно перечислить через запятуюключевые столбцы и индексы. Вы можете использовать следующиеконструкции:

[CONSTRAINT<Имяключа>] PRIMARYKEY (<Списокстолбцов>)

Определяет первичный ключ таблицы. В таблице может быть только один первичный ключ, состоящий из одного или нескольких столбцов. Столбцам, входящим в первичный ключ, автоматически присваивается свойство NOT NULL. Ключевое слово CONSTRAINT и имя ключа можно опустить, так как дляпервичного ключа заданное имя игнорируется и используется имя PRIMARY.

Если в состав первичного ключа входят столбцы с типом TINYBLOB, TINYTEXT,BLOB, TEXT, MEDIUMBLOB, MEDIUMTEXT, LONGBLOB и LONGTEXT, необходимоуказать количество символов в начале значения столбца; при этом первичный ключ содержит не полные значения столбца, а только начальные подстроки значений.

Пример определения первичного ключа:

PRIMARY KEY (id)

INDEX [<Имя индекса>] (<Список столбцов>)

Создает индекс для указанных столбцов. Индекс – это вспомогательный объект, позволяющий значительно повысить производительность запросов с условием на значение столбцов, включенных в индекс. Например,чтобы создать индекс для быстрого поиска по именам клиентов, в команду создания таблицы Customers можно включить определение

INDEX (name)

Аналогично первичному ключу, при создании индекса для столбцов с типомTINYBLOB, TINYTEXT, BLOB, TEXT, MEDIUMBLOB, MEDIUMTEXT, LONGBLOB иLONGTEXT необходимо указать количество символов в начале значения столбца, по которым будет проведено индексирование.

[CONSTRAINT<Имя внешнего ключа>].
FOREIGNKEY [<Имя индекса>] (<Список столбцов>)
REFERENCES<Имя родительской таблицы>
(<Список столбцов первичного ключа родительской таблицы>)
[<Правила поддержания целостности связи>]

Определяет внешний ключ таблицы.

Внешний ключ, создает связь между данной (дочерней) таблицей и родительской таблицей. Внешние ключи поддерживаются только для таблиц с типомInnoDB (причем и дочерняя, и родительская таблица должны иметь тип InnoDB), для остальных типов таблиц выражение FOREIGN KEY игнорируется.

Столбцы, составляющие внешний ключ, должны иметь типы, аналогичные типамстолбцов первичного ключа в родительской таблице. Для числовых столбцов долженсовпадать размер и знак, для символьных – кодировка и правило сравнения значений.Столбцы с типом TINYBLOB, TINYTEXT, BLOB, TEXT, MEDIUMBLOB, MEDIUMTEXT,LONGBLOB и LONGTEXT не могут входить во внешний ключ.

Правила поддержания целостности связей

• ON DELETE|UPDATE CASCADE – каскадное удаление|обновление строк дочерней таблицы;

• ON DELETE|UPDATE SET NULL – обнуление значений внешнего ключа в соответствующих строкахдочернейтаблицы;

• ONDELETE|UPDATERECTRICT или ONDELETE|UPDATENOACTION– запретудаления|обновлениестрокродительскойтаблицыприналичииссылающихся на них строк дочерней таблицы.

Для столбцов внешнего ключа автоматически создается индекс, поэтому проверки значений внешних ключей в ходе контроля целостности связи выполняются быстро.

Порядок выполнения работы:

1. Используя команду ALTERTABLEизмените структуру всех таблиц, добавив первичные и внешние ключи.
2. В PhpMyAdmin в дизайнере посмотрите созданные связи между таблицами с помощью Дизайнера.

Задание 2:

В СУБД Microsoft SQLServer создайте диаграмму для базы данных Students.

Порядок выполнения работы:

Создание диаграммы

В БД "Microsoft SQLServer 2008" все диаграммы находятся в папке **"Диаграммы баз данных"** обозревателя объектов.

Создадим диаграмму, обеспечивающую целостность данных нашей БД**"Students"**. Для создания новой диаграммы в БД**"Students"** щелкните **ПКМ** по папке **" Диаграммы баз данных "** и в появившемся меню выберем пункт**"Создать диаграмму базы данных "**. Сначала появится окно с вопросом о добавлении нового объекта **"Диаграмма"**. В этом окне нужно нажать кнопку **"Да"**. Затем появится окно **"Добавление таблиц"** предназначенное для добавления таблиц в новую диаграмму.

В окне добавления таблиц выделите все таблицы нашей БД и нажмите кнопку **"Добавить"**, потом **"Закрыть".**

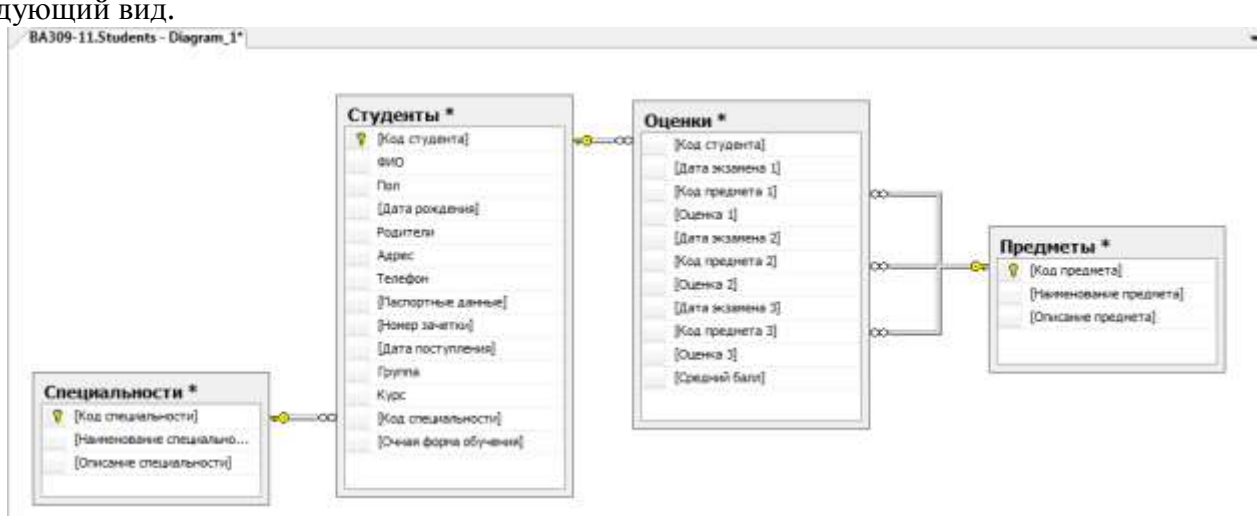
Появится окно диаграммы, где будут отображены отобранные таблицы. Теперь необходимо определить связи между таблицами. Перетащите поле**"Код специальности"** из таблицы **"Специальности"** на такое же поле в таблице **"Студенты"**. Появится окно создания связи между таблицами **"Таблицы и столбцы"**.

В окне создания связи нажмите кнопку **"Ok"**. Появится окно настройки свойств связи **"Связь по внешнему ключу"**.

Оставьте свойства связи без изменений и в окне свойств связи нажмите кнопку **"Ok"**.В диаграмме между таблицами **"Студенты"** и **"Специальности"** появится связь в виде ломанной линии.

Аналогичным образом создайте связь таблицы **"Студенты"** с таблицей **"Оценки"**, перетащив поле**"Код студента"** из таблицы **"Студенты"** на одноименное поле в таблице **"Оценки"**. Затем, свяжите таблицы **"Предметы"** и **"Оценки"**, перетащив поле**"Код предмета"** из таблицы **"Предметы"** на поля **"Код предмета 1", "Код предмета 2"** и **"Код предмета**

3"таблицы **"Оценки"**. После выполнения вышеперечисленных действий диаграмма примет следующий вид.



Закройте окно с диаграммой. Появится окно с вопросом о сохранении новой диаграммы, где необходимо нажать кнопку **"Да"**.

Появится окно определения имени новой диаграммы. В данном окне, задайте имя диаграммы как "ДиаграммаБД Студенты" и нажмите кнопку **"Ok"**.

Появится окно **"Сохранить"** с запросом сохранения таблиц, входящих в диаграмму. В данном окне необходимо нажать кнопку **"Да"**.

Форма представления результата:

Отчет по выполненной практической 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 2.2. Реализация баз данных в конкретной СУБД

Практическая работа № 7
Вставка данных в базу данных

Цель: получение практических навыков по освоению операций добавления записей в базу данных

Выполнив работу, Вы будете:
уметь:
- выполнять вставку данных.

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

Задание 1:
В СУБД MySQL заполнить базу данных (свой вариант) значениями.

Краткие теоретические сведения:
Для добавления одной или нескольких строк в таблицу используется команда:

INSERT [INTO] <Имя таблицы>
[(<Список столбцов>)]
VALUES
(<Список значений 1>),
(<Список значений 2>),
...
(<Список значений N>);

В команде INSERT используются следующие основные параметры.
• Имя таблицы, в которую добавляются строки.
• Список имен столбцов, для которых будут заданы значения. Если значения будутзаданы для всех столбцов таблицы, то приводить список столбцов необязательно.Если столбец таблицы не включен в список, то в этом столбцепри добавлении строки будет автоматически установлено значение поумолчанию.
• Значения, которые нужно добавить в таблицу. Значения могут указываться в одномиз следующих форматов:
– набор значений для каждой добавляемой строки заключается в скобки. Набор значений внутри каждой пары скобок должен соответствовать указанному списку столбцов, аесли список столбцов не указан, то упорядоченному списку всех столбцов, составляющихтаблицу (список столбцов таблицы можно просмотреть с помощью команды DESCRIBE. Значения внутри набора, а такжесами наборы отделяются друг от друга запятыми;
– символьные значения, а также значения даты и времени приводятся в одинарныхкавычках. Для числовых значений кавычки необязательны. Десятичным разделителем длячисел с дробной частью служит точка. Время и даты вводятся, соответственно, в формате«YYYY-MM-DD» и «HH:MM:SS»;
– чтобы ввести в столбец неопределенное значение, то необходимо указать вместо значения ключевое слово NULL *без кавычек* (слово в кавычках рассматривается как обычнаясимвольная строка);
– вместо значения можно указать ключевое слово DEFAULT *без кавычек,* тогда в столбец будет введено значение по умолчанию (если оно задано для этого столбца).Например, добавьте в таблицу Product сведения о продукции компании.
INSERT INTO Product (name_pr, description,price, qty)

```
VALUES
(' Инфракрасный обогреватель','3 режима нагрева: 400 Вт, 800 Вт, 1200 Вт',1445.00, 4),
('Гриль','Мощность 1440 Вт. Быстрый нагрев',4115.00, 6),
('Кофеварка','Цвет: черный. Мощность: 450 Вт',1710.00, 10),
('Чайник','Цвет: белый. Мощность: 2200 Вт. Объем: 2 л',1725.00, 14),
('Утюг','Цвет: фиолетовый. Мощность: 1400 вт',5300.00, 16);
```

Эта команда добавляет значения в столбцы name_pr(наименование), description (описание), price (цена) и qty (количество) таблицы Product. При этом в столбец id (идентификатор) автоматическивносятся порядковые номера строк, поскольку этот столбец имеет тип данных SERIAL.

Порядок выполнения работы:

Используя команду INSERTвставить данные в таблицы базы данных.

Задание 2:

Таблицы базы данных Students заполнить начальными значениями.

Порядок выполнения работы:

Заполним таблицу**"Специальности"**. Для заполнения этой таблицы в обозревателе объектов щелкните правой кнопкой мышипотаблице**"Специальности"**и в появившемсяменювыберитепункт**"Изменить первые 200 строк"**. В рабочей области "Microsoft SQL ServerManagementStudio" проявится окно заполнения таблиц. Заполните таблицу**"Специальности"**.

BA309-11.Studen...bo.Специальности

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

Так какполе**"Код специальности"**является первичным полем связи и ключевым числовым счетчиком, то оно заполняется автоматически (заполнять его не нужно).

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

После заполнения таблицы**"Специальности"**заполним таблицу**"Предметы"**. Откройте ее для заполнения как описано выше, и заполните.

BA309-11.Students - dbo.Предметы

Код предмета	Наименование предмета	Описание предмета
1	Операционные системы	Microsoft Windows7
2	Офисные пакеты	Microsoft Office 2010
3	Базы данных	Microsoft Access 2010
4	Языки программирования	Microsoft Visual Studio 2010
5	Проектирование информационных систем	Microsoft SQL Server 2008
NULL	NULL	NULL

Закройте окно заполнения таблицы**"Предметы"**и перейдите к заполнению

BA309-11.Students - dbo.Студенты

Код сту...	ФИО	Пол	Дата рождения	Родители	Адрес	Телефон	Паспортные да...	Номер зачетки	Дата поступл...	Группа	Курс	Код спец...	Очная фо...
1	Иванов А.И.	Мужской	1990-12-12	Отец, Мать	Москва	+74957895674	8567-567543	13245	2013-09-01	ММ11	1	1	True
2	Петрова И.И.	Женский	1989-11-01	Мать	Москва	+74957889876	4567-765432	34563	2012-08-01	ПИ21	2	2	False
3	Мухин М.А.	Мужской	1989-05-14	Отец	Самара	+78462875690	5438-098787	56732	2011-07-05	СТ22	2	3	False
4	Сидорова В.К.	Женский	1988-09-27	Нет	Саратов	+79027868909	1287-987509	27543	2010-06-23	МО31	3	4	True
5	Кожевников А.А.	Мужской	1988-04-12	Мать	Казань	+79160543467	2312-671400	34217	2010-07-21	БУ33	3	5	True
6	Пальчикова Н.Е.	Женский	1990-09-02	Отец, Мать	Челябинск	+74569098723	8743-856780	43278	2013-08-01	ММ12	1	1	False
7	Царев Е.В.	Мужской	1988-02-17	Отец	Самара	+78462234769	6543-834521	43765	2009-07-04	ПИ41	4	2	True
8	Баранова Г.В.	Женский	1988-07-09	Отец, Мать	Чебоксары	+79027834638	2133-896567	10387	2009-08-09	СТ42	4	3	False
9	Леухин П.Г.	Мужской	1990-02-26	Нет	Казань	+79067453678	2769-634904	67348	2008-07-23	МО51	5	4	True
10	Николаева А.П.	Женский	1988-03-03	Мать	Саратов	+78546456432	3256-090932	45287	2008-06-21	БУ53	5	5	False
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

таблицы**"Студенты"**. Откройте таблицу**"Студенты"** для заполнения и заполните ее.

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

Поле**"Код специальности"**является вторичным полем связи (для связи с таблицей**"Специальности"**). Следовательно, значения этого поля необходимо заполнять значениями поля**"Код специальности"**таблицы**"Специальности"**. В нашем случае это значения от 1 до 5. Если у Вас коды специальностей в таблице**"Специальности"**имеют другие значения, то внесите их в таблицу**"Студенты"**.

Поокончании заполнения, закройте окно заполнения таблицы**"Студенты"**.

Наконец заполним таблицу**"Оценки"**.

BA309-11.Students - dbo.Оценки

Код студе...	Дата экзамена 1	Код предмета 1	Оценка 1	Дата экзамена 2	Код предмета 2	Оценка 2	Дата экзамена 3	Код предмета 3	Оценка 3	Средний балл
1	2015-02-01	1	5	2015-02-09	4	3	2015-02-14	2	4	0
2	2015-01-30	5	4	2015-02-23	3	5	2015-02-27	1	5	0
3	2015-01-26	3	5	2015-02-05	1	3	2015-02-15	5	3	0
4	2014-12-26	2	3	2015-01-11	4	4	2015-01-21	3	4	0
5	2015-01-12	4	4	2015-01-18	5	4	2015-01-25	1	4	0
6	2014-12-17	2	4	2014-12-26	4	5	2015-01-11	1	3	0
7	2015-02-21	5	2	2015-02-25	1	2	2015-02-27	2	4	0
8	2015-02-03	3	3	2015-02-12	5	3	2015-02-20	4	5	0
9	2015-01-25	1	5	2015-02-02	3	5	2015-02-14	5	5	0
10	2014-12-28	4	4	2015-01-11	1	4	2015-01-23	2	3	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Поля с датами заполняются, как и в таблице**"Студенты"**.Поля**"Код предмета 1","Код предмета 2"**и**"Код предмета 3"**являются вторичными полями связи с таблицей **"Предметы"**. Поэтому они должны быть заполнены значениями поля**"Код предмета из этой таблицы"**, то есть значениями от 1 до 5.

Закройте окно заполнения таблицы**"Оценки"**.

Форма представления результата:

Отчет по выполненной практической 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 2.2. Реализация баз данных в конкретной СУБД

Практическая работа № 8
Сортировка, поиск и фильтрация данных

Цель: получение практических навыков по освоению операций сортировки, поиска и фильтрации данных.

Выполнив работу, Вы будете:

уметь:

-осуществлять сортировку данных;

- выполнять поиск и замену данных;

- осуществлять фильтрацию данных.

Материальное обеспечение:

Методические указания для выполнения практических работ, компьютер, программное обеспечение: MSAccess.

Задание:

Освоить операции сортировки, поиска и фильтрации данных на примере базы данных ***Туризм***.

Краткие теоретические сведения:

Поиск и представление данных из базы данных – одна из основных задач СУБД. В зависимости от информационной потребности можно использовать простые приемы поиска данных или более сложные, позволяющие формировать непростые критерии отбора.

К простейшим видам поиска относится использование команд ***Найти и Заменить.*** В условиях поиска могут быть использованы операции сравнения (>, <, <= ,>= ,= , <>), а также подстановочные символы:

***** - любая цифра или символ. Может быть первым или последние символом текстовой строки.

Например, Wh* - поиск слова What, white и why.

? – любой тестовый символ. Например – B?ll – поиск слова ball, bell и bill.

[ ] – любой один символ из заключенных в скобки. Например – B[ae]ll – поиск слов ball, bell, но не bill.

! – любой один символ, кроме заключенных в скобки. Например - b[!ae]ll – поиск слова billbull,но не bell и ball.

- -любой символ из диапазона. Нужно указывать по возрастанию (от A до Z,но не от Z До A).

Например, b[a-c]d – поиск слов bad, bbd и bcd.

- любая цифра. Например, 1#3 – поиск значений 103, 113, 123.

ИСПОЛЬЗОВАНИЕ ФИЛЬТРОВ

Фильтр – это способ показать в окне только те записи базы данных, которые удовлетворяют требованиям пользователя. Фильтры – это одноразовые запросы, без имени. Они просты в использовании. Можно применять фильтры к таблице, запросу или форме, но фильтруются всегда данные только одной таблицы. В фильтре отображаются все поля.

В СУБД MSAccess несколько видом фильтров.

1. ***Фильтр по выделенному***

Необходимо выделить фрагмент содержимого нужного поля и установить фильтр одним из способов: ***Фильтр – Фильтр по выделенному,*** контекстное меню - ***Фильтр по выделенному.*** В результате останутся записи, совпадающие по этому полю или его части.

2. Фильтр по форме или изменение фильтра

При использовании *фильтра по форме* получается свернутая в строку пустая таблица с пиктограммой списка в каждом поле, где можно задать критерий отбора. В критерии можно использовать и логические операторы AND, OR, NOT.

Инструментом сортировки можно найденные записи упорядочить.

Например, если нужно в базе данных **Туризм** просмотреть только те записи, в которых **Дата начала тура** после 15.02.02, то нужно открыть таблицу **Договоры**, ***Фильтр – Изменить фильтр***, в этом поле набрать условие >#15.02.02#, имея в виду, что константы типа Дата/Время заключаются в #. После этого нужно нажать кнопку ***Применить фильтр.***

В результате на экране останутся только соответствующие критерию записи.

3. Фильтр по вводу

Устанавливается при помощи вызова контекстного меню на нужном поле таблицы. Может применяться в таблицах и формах. Позволяет найти записи, удовлетворяющие нескольким условиям одновременно.

4. ***Расширенныйфильтр***

Вызывается командой ***Фильтр – Расширенный фильтр***. В приведенном окне бланка фильтра пользователь имеет возможность создать фильтр, введя условия отбора, с помощью которых из всех записей в открытой форме или таблице выделяется подмножество, удовлетворяющее данным условиям. Кроме того, в бланке фильтра задается порядок сортировки для одного или нескольких полей.

Порядок выполнения работы:

1. Откройте базу данных ***Туризм***.
2. Откройте таблицу ***Сотрудники***.

Поиск данных

3. Осуществите следующие операции поиска:
   - найдите все записи о служащих в должности «Менеджер»;
   - определите домашний телефон, который начинается на цифру 5;
   - выберите телефоны, содержащие цифру 5;
   - определите фамилии, имеющие вторую букву «а» или «о».

Замена данных

4. Замените все должности «Менеджер» на «Менеджер по продажам».

Сортировка данных в таблицах

5. Отсортируйте фамилии сотрудников по алфавиту.
6. Отсортируйте записи по должностям, а для одинаковых должностей — по фамилиям. Для этого расположите поле ***Должность***слева от поля ***Фамилия,*** выделите оба поля и выполните сортировку.

Использование фильтров

7. Откройте таблицу ***Сотрудники***.
8. Установите фильтры по выделенному (снимая фильтр каждый раз после получения результата):
   - конкретная фамилия (например, Иванов);
   - записи, в которых фамилии заканчиваются на «вич», «ов»;
   - выборка менеджеров по продажам;
   - выборка проживающих в одном районе (по первым трем цифрам телефона).
9. Установите фильтрование данных с помощью исключения данных (вместо включения). Для этого выберите предложенные в предыдущем пункте критерии как исключающие («все, кроме этих» — контекстное меню — **Исключить выделенное**).
10. Установите фильтр по форме:
    - фамилии, начинающиеся на «О» или «К»;

- сотрудники не старше 25 лет;
- с окладом меньше 1500.

11. Установите фильтр по вводу (контекстное меню на нужном поле таблицы):
    - фамилии, начинающиеся на букву А;
    - договоры, заключенные в 2008 году;
    - клиенты, зарегистрированные как групповые.
12. Выберите команду **Фильтр — Расширенный фильтр**:
    - конкретная фамилия сотрудника;
    - договор в конкретную страну, оформленный заданным сотрудником (например, «Какие договоры на посещение Испании заключил Сидоров?»);
    - номера телефонов, которые содержат цифру 9.

Форма представления результата:

Отчет по выполненной практической 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 2.2. Реализация баз данных в конкретной СУБД

Практическая работа № 9
Построение запросов (различного уровня сложности)

Цель: получение практических навыков по освоению операций создания запросов с помощью языка SQL

Выполнив работу, Вы будете:

уметь:

- создавать запросы с помощью конструктора и языка SQL;
- создавать запросы на выборку, с вычисляемыми полями, параметрические, итоговые и перекрестные.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MSAccess, MySQL, PhpMyAdmin, MSSQLServer.

Задание 1:

Разработать запросы для базы данных Туризмс помощью конструктора в СУБД MSAccess.

Краткие теоретические сведения:

Запрос является объектом базы данных. Он представляет собой сформулированную информационную потребность.

При работе с запросом можно выделить два этапа: формирование (проектирование) и выполнение. При выполнении запроса выбирается информация из всех таблиц базы данных в соответствии с критерием запроса.

В верхней части окна **Конструктора** размещаются нужные таблицы посредством команды ***Запрос – Добавить таблицу*** или та же команда в контекстном меню. В нижней части окна расположен бланк запроса, информация в него заносится путем перетаскивания нужных полей из таблиц в верхней части окна в строку «поле» или двойным щелчком мыши. При этом имя таблицы в бланке подставляется автоматически.

Наличие «галочки» в строке «Вывод на экран» означает присутствие данного поля в таблице результатов поиска. Критерии запроса устанавливаются в строке «Условия отбора» и последующих строках, связанных логическим оператором OR. Все критерии отбора, указанные в одной строке, объединяются оператором AND.

В качестве «Условия отбора» могут быть выражения (вычисляемое поле) даты, текст, которые либо вносятся вручную, либо инструментом, либо с помощью команды контекстное меню ***Построить.*** Константы типа Дата/Время заключаются в #.

Запросы бывают разных типов: *на выборку, на создание, на обновление, на добавление, на удаление, перекрестный, итоговый, параметрический* и др. По умолчанию формируется запрос на выборку. Тип запроса может быть преобразован в любой другой командой***Запрос.***

При создании критерия можно использовать инструмент ***Построить***или такую же команду контекстного меню для категории «условие отбора».

1. ***Вычисляемые поля в запросах***

С помощью запросов можно задать вычисления над данными и сделать вычисляемое поле новым полем в наборе данных. Для создания нового поля в пустой ячейке строки ***Поле*** в бланке запроса вводится формула:

Имя поля: выражение

Для построения выражений имеется специальное средство – **Построитель** выражений, вызываемый правой кнопкой мыши на поле или кнопкой ***Построить.***

В верхней части размещается область ввода. Нижняя содержит три списка для выбора имен полей и функций. В папке ***Функции*** размещаются встроенные функции, сгруппированные по категориям.

2. ***Параметрическиезапросы***

Условия запроса могут быть включены непосредственно в бланк запроса, но для того чтобы сделать его более универсальным, можно вместо конкретного значения отбора включить в запрос параметр, т.е. создать параметрический запрос. Для этого в строку «условия отбора» вводится фраза в квадратных скобках, которая будет выводиться в качестве «подсказки» в процессе диалога, например, [введите фамилию]. Таких параметров может быть несколько, каждый для своего поля.

3. ***Итоговыезапросы***

При выборе данных может понадобиться найти какую-либо функцию, например, сумму значений или максимальное значение в поле. Запросы, выполняющие вычисления над группой записей, называются итоговыми. В бланке запроса появится новая строка с наименованием «Групповая операция», в ней содержится слово «Группировка». В этой строке следует указать, какое вычисление необходимо выполнить.

Возможные операции в строке «Групповые операции»:

SUM – сложение;
AVG – среднее значение;
MIN – минимальное значение;
MAX – максимальное значение;
COUNT – количество записей со значениями (без пустых значений);
STDEV – стандартное отклонение;
VAR – дисперсия;
FIRST – значение в первой записи;
LAST - значение в последней записи.

4. ***Перекрестныезапросы***

Особый тип итоговых запросов, представляющих результаты поиска в виде матрицы, называется перекрестным.

Для каждого поля такого запроса может быть выбрана одна из установок: «Заголовки строк», «Заголовки столбцов», «Значение», которое выводится в ячейках таблицы, и «Не отображаются».

Для перекрестного запроса надо обязательно определить хотя бы по одному полю в качестве заголовка строк, заголовка столбцов и значения. Можно использовать дополнительные условия отбора и сортировка.

Порядок выполнения работы:

Запрос на выборку

1. Откройте базу данных ***Туризм*** и перейдите на вкладку **Создать - Запрос**.
2. В режиме **Конструктора**создайте и сохраните следующие запросы на выборку, определив нужные таблицы:
   - список всех путешествий в определенную страну (например, Испанию);
   - список всех регионов в конкретной стране (например, Англии). Сохраните запрос под именем «*Страна-Регион*»;
   - все туры, проданные в 2008 году. Сохраните запрос с именем «*Туры 2008*»;
   - список сотрудников, работающих с 1999 года и раньше. Сохраните запрос с именем «Ветераны». Добавьте в запрос строку «Сортировка» и установите сортировку по фамилиям.
   - сотрудникам, которые родились в 1973 г., используя в качестве критерия выражение: ***Between... and***(**Построить — Операторы — Сравнения — Between**), а затем повторите запрос, построив выражение с помощью знаков «<» и «>»;
   - сотрудникам, фамилии которых с «Г» по «Я»;
   - сотрудникам, фамилии которых начинаются с «Н» по «Я» и с «А» по «В»;
   - индивидуальным клиентам, фамилии которых имеют вторую букву «о»;

- пяти фамилиям сотрудников, которые начинаются с букв «А»или «В».
- постоянным клиентам, количество договоров с которыми больше 3.

Запросы с вычисляемыми полями

5. Создайте запрос для расчета ведомости заработной платы для сотрудников агентства, включив в нее следующие поля: ***Фамилия сотрудника, Размер оклада, Стаж, Надбавка, Налог, На руки.***

   Для поля ***Стаж***нужно использовать формулу, построенную с помощью кнопки **Построить**, в которой учитывается сегодняшняя дата и **Дата найма**на работу:

   Стаж = (Date()-Сотрудники!ДатаНайма)\365

   Для поля ***Надбавка***нужно исходить из того, что она составляет 20% от ***Размера оклада***, если ***Стаж***меньше 5 лет, и 30% — если стаж больше 5 лет:

   IIf ([стаж]<5;0,2*[Сотрудники]![Размер оклада]; 0,3* [Сотрудники]! [Размер оклада])

   Поле ***Налог***рассчитывается как 13% от Размера оклада:

   [Сотрудники]![Размер оклада] *0,13

   Поле На руки рассчитывается как:

   [Размер оклада]+[надбавка]—[налог].

   В результате выполнения запроса будет получена новая ведомость.
6. Создайте запрос для определения стоимости путевок корпоративных клиентов, включив в него поля Клиент, Стоимость путевки

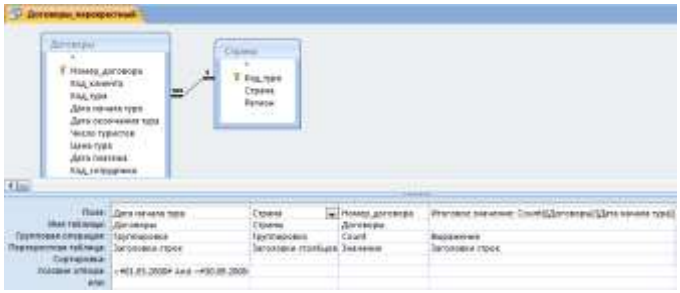
   Стоимость путевки = Sum(договоры![Цена тура] *договоры![Число туристов])

   Параметрические запросы
7. Сформируйте запрос для выборки всех туров по названиюстраны.
8. Создайте запрос для получения данных на сотрудников, работающих по турам в конкретную страну.
9. Создайте запрос по всем клиентам, оформившим договоры в определенную страну и регион.

   Итоговые запросы
10. Создайте запрос, используя подходящие функции, найдите наибольший и средний размеры цены тура.
11. Создайте запрос для подсчета объема продаж путевок в конкретную страну. Для этого:
    - добавьте в Конструкторе запросов таблицу **Договоры**и **Страны**
    - добавьте в бланк запроса поля ***Название страны***(из таблицы Страны) и расчетное поле Цена тура * Число туристов, которому присвоим название Стоимость путевок;
    - выберите команду **Вид — Групповые операции**и в выпадающем списке в строке «Группировка» для поля ***Стоимость путевок***установите функцию SUM;
    - запустите запрос и просмотрите результаты.
12. Создайте запрос для определения средней цены и общей суммы туров за 2005 год.
13. Для объединения записей в группы и получения итоговых значений по каждой группе используется опция «Группировка». Создайте новый запрос для базы данных **Туризм**, в котором определите общие суммы продаж путевок по годам:
    - добавьте таблицу Договоры в окно запроса;
    - в первый столбец поместите поле Год начала тура, рассчитав его с помощью функции **Year**, во второй — сумма общих продаж путевок — Sum(договоры![Цена тура]*договоры![Число туристов]);
    - установите для первого столбца в строке «*Групповая операция*» — «*Группировка*», для второго — *Выражение*;
    - выполните запрос и прокомментируйте результаты.
14. Дополните предыдущий запрос критерием, который включает в выборку только те заказы, которые оформлены в 2008 г. и позже. Для этого следует добавить в бланк запроса поле ***Дата заказа***из таблицы «**Заказы**». В строке «*Групповая операция*» выберите пункт «*Условие*». В строке «*Условие отбора*» укажите условие на дату. Обязательно снимите флажок «*Вывод на экран*» для этого поля. Выполните запрос и проанализируйте результаты.

15. Выберите записи, стоимость перевозок в которых превышает заданное значение.
16. Найдите записи, в которых для каждого вида доставки было оформлено более 5 заказов («***Доставка***» — «*Группировка*», ***Код заказа*** – COUNT, «*Условие отбора*» в поле ***Кодзаказа***>=5).

    Перекрестные запросы
17. Составьте запрос для выяснения: сколько туров организовано в каждую страну в конкретный регион.
18. Составьте перекрестный запрос по теме: сколько туров начались с мая по сентябрь 2008 г. в



разные страны.

Составьте перекрестный запрос для определения предпочтений клиентов разным регионам (сколько клиентов, в каком регионе побывали).

Задание 2:

Разработать модифицирующие запросы к базе данных Туризм.

Краткие теоретические сведения:

Модификация базы данных с помощью запросов на изменение

- **Запрос на обновление**

Запрос этого типа используется при необходимости внесения изменений во множество записей базы данных, поэтому целесообразно сделать резервную копию таблицы.

Выполняется этот вид запроса в два этапа: сначала проверяется правильность отбора обновляемых записей с помощью запроса на выборку, затем он преобразуется в запрос на обновление и выполняется повторно.

При обновлении полей следует иметь в виду, что если при проектировании таблицы в свойствах поля было указано «условие на значение», то при обновлении этого поля условие может быть нарушено, чего не допустит MSaccess. Поэтому нужно: или изменить условие на значение, или удалить это условие в Конструкторе.

- **Запрос на добавление**

Периодически убирая в архивные таблицы «старые» записи, можно увеличить быстродействие основных частей и улучшить обзорность базы данных.

Кроме того, при необходимости добавить данные в таблицу базы данных из другой базы можно также использовать запросы на добавление.

- **Запрос на удаление**

«Старые» или неиспользуемые записи таблиц можно удалить, но обязательно сначала произвести выборку и проверить ее. Целесообразно сделать копию.

Порядок выполнения работы:

Запрос насоздание

1. Создайте обобщенную таблицу **Договоры по странам**, включив в нее следующие поля:
   Из таблицы **Договоры**: ***Номер договора***; ***Название клиента***
   Из таблицы **Страны**: ***Название страны***; ***Регион***.

Для этого:

- Создайте запрос на выборку этих данных, выполните его и проверьте результаты;
- Если результаты корректны, то поменяйте статус у запроса: **Запрос** – **Созданиетаблицы** – укажите новое имя таблицы **Договоры по странам;**

- Выполните запрос с новым статусом еще раз;
- Перейдите на вкладку **Таблицы** и убедитесь, что появилась новая таблица. Просмотрите ее.

Запрос на обновление

2. Увеличьте ***Размер оклада*** у менеджеров по продажам на 15%.

Для этого:

- Составьте новый запрос на выборку, включив в него поля ***Фамилия***, **Должность** и *Размер оклада*;
- Проверьте составленный запрос;
- Видоизмените запрос, установив ему статус «*Обновление*» (**Запрос – Обновление**). В появившейся в бланке запроса строке «*Обновление*» для поля ***Размер оклада*** внесите с помощью **Построить** выражение [Размер оклада]*1,15;
- Выполните запрос, подтвердите обновление; сохраните запрос, дав ему имя и обратив внимание на появившийся значок у его имени; просмотрите результаты.

Запросна добавление

3. Создайте путем копирования дубликат таблицы **Договоры** без данных, назвав ее **Договоры2008 года**. Для этого в контекстном меню для таблицы Договоры выберите *Копировать*, затем выполните команду *Вставить*, в параметрах вставки укажите «*Только структуру*». Просмотрите таблицу **Договоры 2008 года** – она должна быть пустой и иметь такую же структуру, как и таблица **Договоры.**
4. Отберите в таблицу **Договоры 2008 года** записи обо всех договорах этого года.

Для этого:

- Создайте запрос на выборку, включив в него все поля таблицы **Договоры** в любой последовательности, и критерий по дате, выполните его для проверки правильности;
- Измените статус запроса на «*Добавление*», в появившемся окне задайте имя таблицы для добавления **Договоры 2008 года**, обратите внимание на появление строки «*Добавление*» в бланке запроса;
- Выполните запрос и подтвердите добавление; просмотрите результаты архивации и сохраните запрос, обратив внимание на значок у его имени.

Запросна удаление

5. Удалите из таблицы **Договоры** записи о договорах 2008 года, используя копию сохраненного запроса на добавление в таблицу **Договоры 2008 года**, изменив его статус на «*Удаление*».

Задание 3

Разработать простые запросы на языке SQL для учебной базы данных.

Краткие теоретические сведения:

Оператор SELECT (выбрать) языка SQL является самым важным и самым часто используемым оператором. Он предназначен для выборки информации из таблиц базы данных. Упрощенный синтаксис оператора SELECT выглядит следующим образом:

SELECT [DISTINCT] <список атрибутов>
FROM<список таблиц>
[WHERE<условие выборки>]
[ORDERBY<список атрибутов>]
[GROUPBY<список атрибутов>]
[HAVING<условие>]
[UNION<выражение с оператором SELECT>];

В квадратных скобках указаны элементы, которые могут отсутствовать в запросе.

Ключевое слово SELECT сообщает базе данных, что данное предложение является запросом на извлечение информации. После слова SELECT через запятую перечисляются наименования полей, содержимое которых запрашивается.

Обязательным ключевым словом в предложении-запросе SELECTявляется слово FROM (из). За ключевым словом FROM указывается список разделенных запятыми имен таблиц, из которых извлекается информация.

Например,

```
SELECT NAME, SURNAME
FROM STUDENT;
```

Любой SQL-запрос должен заканчиваться символом «;».

Если необходимо вывести значения всех столбцов таблицы, то можно вместо перечисления их имен использовать символ «*»/

```
SELECT *
FROMSTUDENT;
```

Для исключения из результата SELECT-запроса повторяющихся записей используется ключевое слово DISTINCT (отличный). Если запрос SELECT извлекает множество полей, то DISTINCT исключает дубликаты строк, в которых значения всех выбранных полей идентичны.

Использование в операторе SELECT предложения, определяемого ключевым словом WHERE (где), позволяет задавать выражение условия, принимающее значение истина или ложь для значений полей строк таблиц, к которым обращается оператор SELECT. Предложение WHERE определяет, какие строки указанных таблиц должны быть выбраны.

Порядок выполнения работы:

1. Из таблицы STUDENT «Учебной базы данных»:

1. Вывести все данные студента с номером 265;
2. Вывести информацию о студентах с именем Вадим;
3. Выбрать фамилии студентов со 2-го и 3-го курсов, получающих стипендию;
4. Вывести имена, фамилии и даты рождения студентов 5 курса;
5. Вывести информацию о студентах из Воронежа;
6. Выбрать фамилию, имя, курс, город студентов, получающих стипендию от 100 до 200 рублей;
7. Вывести список идентификаторов университетов, исключая повторения.

2. Из таблицы EMP:

- Найти всех служащих с зарплатой в диапазоне от $1000 до $2000;
- Выбрать все категории должностей;
- Вывести всех служащих, зачисленных на работу в 1981 году;
- Вывести данные о сотрудниках 10 и 20 отделов в алфавитном порядке по их именам;
- Вывести значения полей ENAME и JOB для всех клерков в 20 отделе;
- Найти всех служащих, имена которых содержат комбинации символов TH или LL;
- Вывести информацию о служащих имеющих премию;
- Вывести имя, зарплату и премию для всех продавцов (должность SALESMAN), у которых месячная зарплата (поле SAL) превосходит премию (поле COMM). Отсортируйте строки по полю SAL в порядке убывания.

3. Из таблицы STUDENT «Учебной базы данных»:

- Составить запрос для таблицы STUDENT «Учебной базы данных» таким образом, чтобы выходная таблица содержала один столбец, содержащий последовательность разделённых символом «;»значений всех столбцов этой таблицы, и при этом текстовые значения должны отображаться прописными символами (например,10;КУЗНЕЦОВ;БОРИС;0;БРЯНСК;8/12/1981;10).
- Составить запрос для таблицы STUDENT «Учебной базы данных» таким образом, чтобы выходная таблица содержала один столбец в следующем виде: Б.КУЗНЕЦОВ; место жительства – БРЯНСК; родился – 8.12.81.
- Составить запрос для таблицы STUDENT «Учебной базы данных» таким образом, чтобы выходная таблица содержала один столбец в следующем виде: б.кузнецов; место жительства – брянск; родился 8-ДЕК-1981.

– Составить запрос для таблицы STUDENT «Учебной базы данных» таким образом, чтобы выходная таблица содержала один столбец в следующем виде: Борис Кузнецов родился в 1981 году.

Задание №4

Разработать итоговые запросы на языке SQL для учебной базы данных.

Краткие теоретические сведения:

Агрегирующие функции позволяют получать из таблицы сводную (агрегированную) информацию, выполняя операции над группой строк таблицы. Для задания в SELECT-запросе агрегирующих операций используются следующие ключевые слова:

- COUNT определяет количество строк или значений поля, выбранных посредством запроса и не являющихся NULL-значениями;
- SUM вычисляет арифметическую сумму всех выбранных значений данного поля;
- AVG вычисляет среднее значение для всех выбранных значений данного поля;
- MAX вычисляет наибольшее из всех выбранных значений данного поля;
- MIN вычисляет наименьшее из всех выбранных значений данного поля.

Предложение GROUPBY позволяет группировать записи в подмножества, определяемые значениями какого-либо поля, и применять агрегирующие функции уже не ко всем записям таблицы, а раздельно к каждой сформированной группе.

При необходимости часть сформированных с помощью GROUPBY групп может быть исключена с помощью предложения HAVING.

Предложение HAVING определяет критерий, по которому группы следует включать в выходные данные, по аналогии с предложением WHERE, которое осуществляет это для отдельных строк.

В условии, задаваемом предложением HAVING, указывают только поля или выражения, которые на выходе имеют единственное значение для каждой выводимой группы.

Порядок выполнения работы:

По таблице EMP.

8. Составить запрос для получения минимальной, максимальной и средней зарплаты в компании.
9. Составить запрос для получения максимальной зарплаты по каждой должности.
10. Составить запрос для подсчёта количества менеджеров, работающих в компании.
11. Составить запрос, вычисляющий разницу между наибольшим и наименьшим окладами в компании.
12. Составить запрос, позволяющий найти отделы, в которых работает более трёх служащих.
13. Составьте запрос, позволяющий показать, что коды служащих (столбец EMPNO) уникальны.

По «Учебной базе данных».

14. Составить запрос для подсчёта количества студентов, сдававших экзамен по предмету обучения с идентификатором, равным 22 по таблице EXAMMARKS.
15. Составить запрос, который выполняет выборку для каждого студента значения его идентификатора и минимальной из полученных им оценок по таблице EXAM MARKS, используя предложение GROUP BY.
16. Составить запрос, выполняющий вывод фамилии первого в алфавитном порядке (по фамилии) студента, фамилия которого начинается на букву «П» по таблице STUDENT.
17. Составить запрос, который выполняет вывод (для каждого предмета обучения) наименования предмета и максимального значения номера семестра, в котором этот предмет преподаётся по таблице SUBJECT.
18. Составить запрос для определения количества студентов, сдававших каждый экзамен.
19. Составить запрос для определения количества студентов, проживающих в Воронеже.

20. Составить запрос, выполняющий вывод номера студента, фамилию студента и стипендию, увеличенную на 20%. Выходные данные упорядочить по значению последнего столбца(величине стипендии).
21. Составить запрос, выполняющий вывод списка предметов обучения в порядке убывания семестров. Поле семестра в выходных данных должно быть первым, за ним должны следовать имя предмета обучения и идентификатор предмета.
22. Составить запрос, который выполняет вывод суммы баллов всех студентов для каждой даты сдачи экзаменов и представляет результаты в порядке убывания этих сумм.

Задание №5

Разработатьподзапросы на языке SQL для учебной базы данных.

Краткие теоретические сведения:

SQL позволяет использовать одни запросы внутри других запросов, то есть вкладывать запросы друг в друга.

Как работает запрос SQL со связанным подзапросом:

- Выбирается строка из таблицы, имя которой указано во внешнем запросе.
- Выполняется подзапрос и полученное значение применяется для анализа этой строки в условии предложения WHERE внешнего запроса.
- По результату оценки этого условия принимается решение о включении или не включении строки в состав выходных данных.
- Процедура повторяется для следующей строки таблицы внешнего запроса.

Порядок выполнения работы:

1. Написать запрос с подзапросом для получения данных обо всех оценках студента с фамилией «Зайцева». Предположим, что его персональный номер неизвестен.
2. Написать запрос, выбирающий данные об именах всех студентов, имеющих по предмету с идентификатором 10 балл выше общего среднего балла.
3. Написать запрос, который выполняет выборку имён всех студентов, имеющих по предмету с идентификатором 10 балл ниже общего среднего балла.
4. Написать запрос, выполняющий вывод количества предметов, по которым экзаменовался каждый студент, сдававший более одного предмета.
5. Написать команду SELECT, использующую связанные подзапросы и выполняющую вывод имён и идентификаторов студентов, у которых стипендия совпадает с максимальным значением стипендии для города, в котором живёт студент.
6. Написать запрос, который позволяет вывести имена и идентификаторы всех студентов, для которых точно известно, что они проживают в городе, где нет ни одного университета.
7. Написать два запроса, которые позволяют вывести имена и идентификаторы всех студентов, для которых точно известно, что они проживают не в том городе, где расположен их университет:
   - с использованием соединения;
   - с использованием связанного подзапроса.
8. Написать запрос EXISTS, позволяющий вывести данные обо всех студентах, обучающихся в вузах, которые имеют рейтинг выше 300.
9. Написать предыдущий запрос, используя соединения.
10. Написать запрос с EXISTS, выбирающий сведения обо всех студентах, для которых в том же городе, где живёт студент, существуют университеты, в которых он не учится.
11. Написать запрос, выбирающий из таблицы SUBJECT данные о названиях предметов обучения, экзамены по которым сданы более чем одним студентом.

Задание №6

Разработатьзапросы, используя объединение таблиц на языке SQL для учебной базы данных.

Порядок выполнения работы:

1. Написать запрос, который выполняет вывод данных о фамилиях сдававших экзамены студентов (вместе с идентификаторами каждого сданного ими предмета обучения).
2. Написать запрос, который выполняет выборку значений фамилий всех студентов с указанием для студентов, сдававших экзамены, идентификаторов сданных ими предметов обучения.
3. Написать запрос, который выполняет вывод данных о фамилиях студентов, сдававших экзамены, вместе с наименованиями каждого сданного ими предмета обучения.
4. Написать запрос на выдачу для каждого студента названий всех предметов обучения, по которым этот студент получил оценку 4 или 5.
5. Написать запрос на выдачу данных о названиях всех предметов, по которым студенты получили только хорошие (4 и 5) оценки. В выходных данных должны быть приведены фамилии студентов, названия предметов и оценка.
6. Написать запрос, который выполняет вывод списка университетов с рейтингом, превышающим 300, вместе со значением максимального значения стипендии, получаемой студентами в этих университетах.
7. Написать запрос на выдачу списка фамилий студентов (в алфавитном порядке) вместе со значением рейтинга университета, где каждый из них учится, включив в список и тех студентов, для которых в базе данных не указано место их учёбы.
8. Написать запрос, выполняющий вывод списка всех пар фамилий студентов, проживающих в одном городе. При этом не включать в список комбинации фамилии студентов самих с собой (то есть комбинации типа «Иванов - Иванов») и комбинацию фамилий студентов, отличающиеся порядком следования (то есть включать одну из двух комбинаций типа «Иванов - Петров» и «Петров - Иванов»).
9. Написать запрос, выполняющий вывод списка всех пар названий университетов, расположенных в одном городе, не включая в список комбинаций названий университетов самих с собой и пары названий университетов, отличающиеся порядком следования.
10. Написать запрос, который позволяет получить данные о назначениях университетов и городов, в которых они расположены, с рейтингом, равным или превышающим рейтинг ВГУ.

Задание №7

Разработать запросы на соединение таблиц на языке SQL для учебной базы данных.

Порядок выполнения работы:

1. Написать запрос, выбирающий данные о названиях университетов, рейтинг которых равен или превосходит рейтинг Воронежского государственного университета.
2. Написать запрос, использующий ANY или ALL, выполняющий выборку данных о студентах, у которых в городе их постоянного местожительства нет университета.
3. Написать запрос, выбирающий из таблицы EXAMMARKS данные о названиях предметов обучения, для которых значение полученных на экзамене оценок (поле MARK) превышает любое значение оценки для предмета, имеющего идентификатор, равный 105.
4. Написать этот же запрос с использованием MAX.
5. Создать объединение двух запросов, которые выдают значения полей UNIV_NAME, CITY, RATING для всех университетов. Те из них, у которых рейтинг равен или выше 300, должны иметь комментарий «Высокий», все остальные – «Низкий».
6. Написать команду, которая выдаёт список фамилий студентов, с комментарием «успевает» у студентов, имеющих все положительные оценки, комментарием «не успевает» для сдававших экзамены, но имеющих хотя бы одну неудовлетворительную оценку и комментарием «не сдавал» - для всех остальных. В выводимом результате фамилии студентов упорядочить по алфавиту.
7. Вывести объединённый список студентов и преподавателей, живущих в Москве, с соответствующими комментариями: «студент» или «преподаватель».

8. Вывести объединённый список студентов и преподавателей Воронежского государственного университета с соответствующими комментариями: «студент» или «преподаватель».

Задание №8

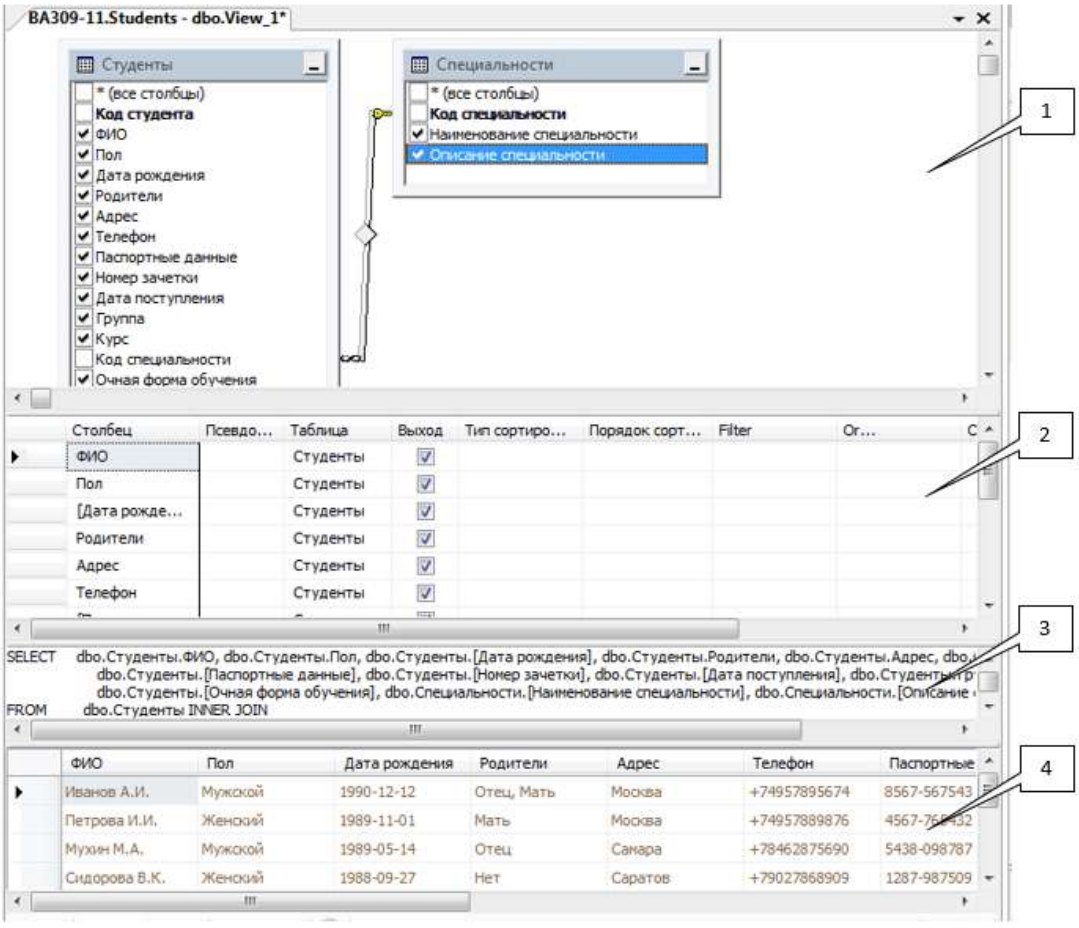
Разработать запросы для базы данных Studentsв СУБД MSSQLServer.

Порядок выполнения работы:

В обозревателе объектов "Microsoft SQL Server" все запросыбазы данныхнаходятся в папке **"Представления"**.

Создадимзапрос**"Запрос Студенты+Специальности"**, связывающий таблицы**"Студенты"**и**"Специальности"**пополю связи **"Код специальности"**. Для создания нового запроса необходимо в обозревателе объектов вбазе данных**"Students"**щелкнуть**ПКМ**попапке**"Представления"**, затем в появившемсяменювыбратьпункт**"Создать представление"**. Появится окно**"Добавление таблицы"**, предназначенное для выбора таблиц и запросов, участвующих в новом запросе.

Добавим в новыйзапростаблицы**"Студенты"**и**"Специальности"**. Для этого выделите таблицу**"Студенты"**и нажмите кнопку**"Добавить"**. Аналогично добавьте таблицу**"Специальности"**. После добавления таблиц, участвующих в запросе, закройте окно. Появится окно конструктора запросов.



Если необходимо удалить таблицу илизапросиз схемы данных, то для этого нужно щелкнуть**ПКМ**и в появившемся менювыбратьпункт**"Remove"**(Удалить).

Теперь определим поля, отображаемые при выполнении запроса. Отображаемые поля обозначаются галочкой (слева от имени поля) на схеме данных, а также отображаются в таблице отображаемых полей. Чтобы сделатьполеотображаемым при выполнении запроса необходимо щелкнуть мышьюпопустому квадрату (слева от имени поля) на схеме данных, в квадрате появится галочка.

Если необходимо сделатьполеневидимым при выполнении запроса, то нужно убрать галочку, расположенную слева от имени поля на схеме данных. Для этого просто щелкните мышьюпогалочке.

Если необходимо отобразить все поля таблицы, то необходимо установить галочку слева от пункта**"* (все столбцы)"** Все поля, принадлежащие соответствующей таблице на схеме данных.

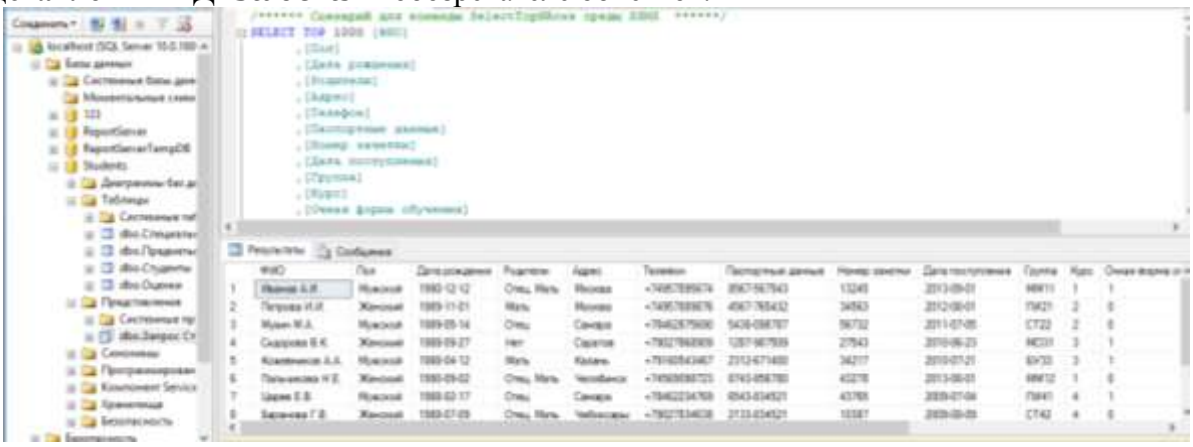
Определите отображаемые поля нашего запроса, как это показано нарисунке 3.1 (Отображаются все поля кроме полей с кодами, то есть полей связи).

На этом настройку нового запроса можно считать законченной. Перед сохранением запроса проверим его работоспособность, выполнив его.

Либо щелкните**ПКМ**в любом месте окна конструктора запросов и в появившемсяменювыберитепункт**"Execute SQL"**(Выполнить SQL). Результат выполнения запроса появиться в виде таблицы в области результата.

Если после выполнения запроса результат не появился, а появилосьсообщение об ошибке, то в этом случае проверьте, правильно ли созданасвязь. Ломанаялиния связидолжна соединять поля**"Код специальности"**в обеих таблицах. Если линия связисоединяет другие поля, то ее необходимо удалить и создать заново, как это описано выше.

Еслизапросвыполняется правильно, то необходимо сохранить. Для сохранения запроса закройте окно конструктора запросов, щелкнув мышьюпокнопке закрытия и задайте имя нового запроса**"Запрос Студенты+Специальности"**.Запроспоявится в папке **"Представления"**БД**"Students"**в обозревателе объектов.

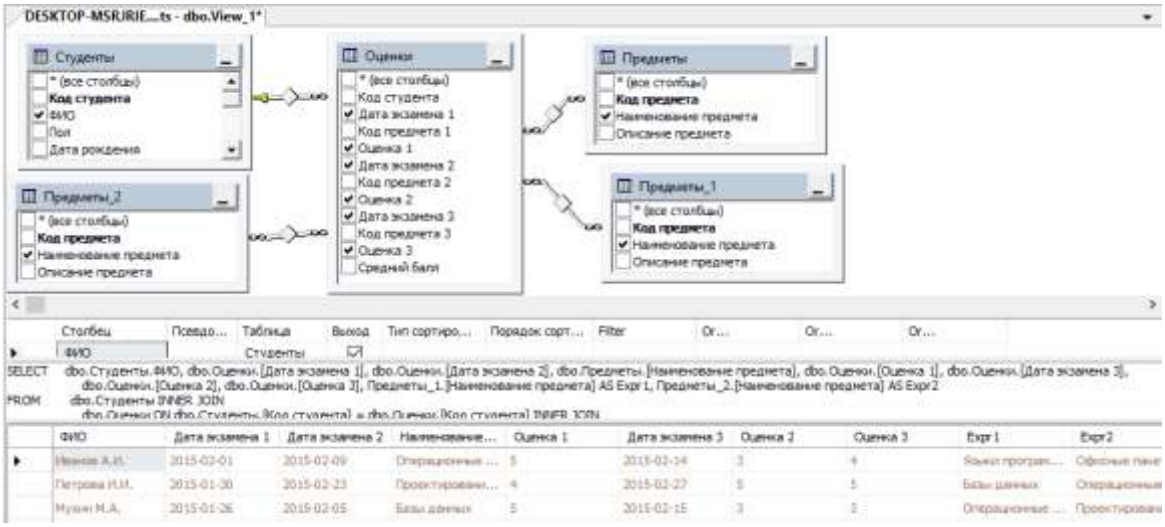


Проверим работоспособность созданного запроса вне конструктора запросов. Запустим вновь созданныйзапрос**"Запрос Студенты+Специальности"**без использования конструктора запросов. Для выполнения уже сохраненного запроса необходимо щелкнуть **ПКМ**позапросу и в появившемсяменювыбратьпункт**"Выбрать первые 1000 строк"**. Выполните эту операцию для запроса**"Запрос Студенты+Специальности".**

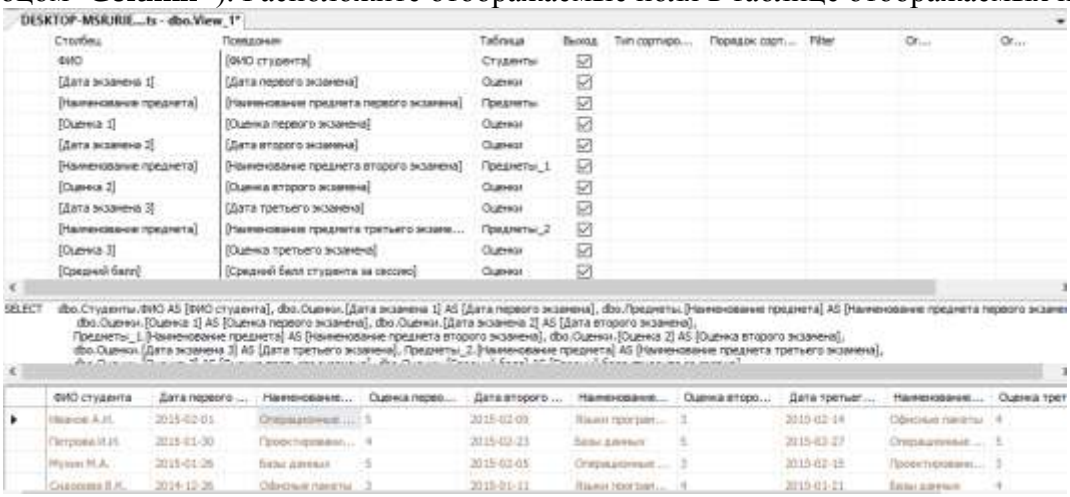
Перейдем к созданию запроса**"Запрос Студенты+Оценки"**. В обозревателе объектов вБД**"Students"**щелкните**ПКМ**попапке **"Представления"**, затем в появившемсяменювыберитепункт**"Создать представление"**. Появиться окно**"Добавление таблиц".**

В запросе**"Запрос Студенты+Оценки"**мы связываем таблицы**"Студенты"**и**"Оценки"***по*полям связи**"Код студента"**. Следовательно, в новыйзапросдобавляем таблицы**"Студенты"**и **"Оценки"**. В данном запросе таблица**"Оценки"**связывается с таблицей**"Предметы"**непоодному полю, апотрем полям. То есть поля**"Код предмета 1", "Код предмета 2"**и**"Код предмета 3"**таблицы**"Оценки"**связаны с полем**"Код предмета"**таблицы**"Предметы".**Поэтому добавим взапростри экземпляра таблицы**"Предметы"**(поодному экземпляру для каждого поля связи таблицы оценки). В итоге в запросе должны участвовать таблицы**"Студенты","Оценки"**и три экземпляра таблицы**"Предметы"**(в запросе они будут называться**"Предметы","Предметы_1"**и**"Предметы_2"**). После добавления таблиц закройте окно**"Добавление таблицы"**, появится окно конструктора запросов.

В окне конструктора запросов установите связи между таблицами и определите отображаемые поля.



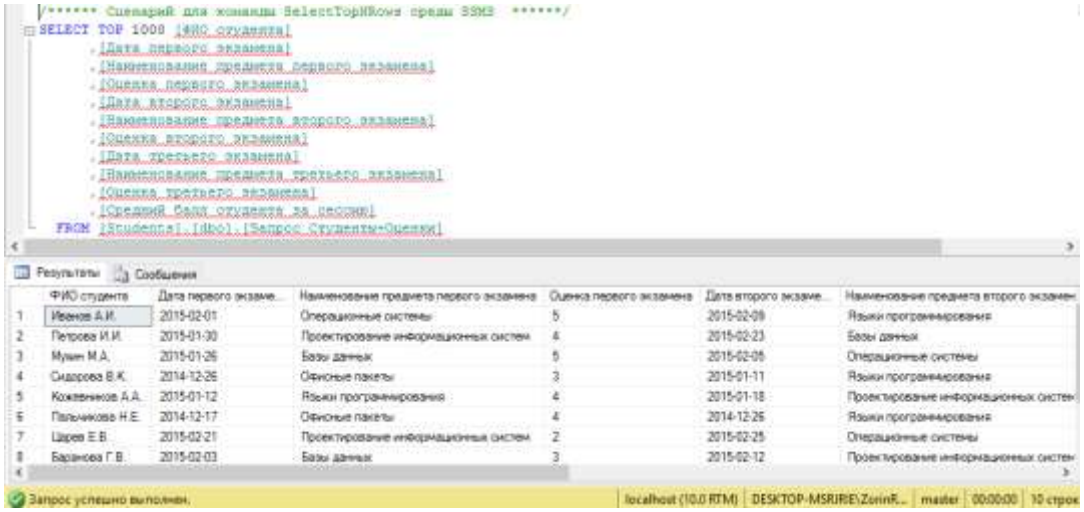
Теперь поменяем порядок отображаемых полей в запросе, для этого в таблице отображаемых полей необходимо перетащить поля мышью вверх или вниз за заголовок строки таблицы (столбец перед столбцом**"Column"**). Расположите отображаемые поля в таблице отображаемых полей.



Задайте псевдонимы для каждого из полей, просто записав псевдонимы в столбце**"Псевдонимы"**таблицы отображаемых полей.

Проверьте работоспособность нового запроса, выполнив его. Обратите внимание на то, что реальные названия полей были заменены их псевдонимами. Закройте окно конструктора запросов. Задайте имя нового запроса **"Запрос Студенты+Оценки"**.

Проверьте работоспособность нового запроса вне конструктора. Для этого запуститезапрос. Результат выполнения запроса **"Запрос Студенты+Оценки"**должен выглядеть как нарисунке.



Форма представления результата:

Отчет по выполненной практической 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 2.2. Реализация баз данных в конкретной СУБД

Практическая работа № 9
Создание файла проекта базы данных. Создание интерфейса входной формы

Цель: получение практических навыков по освоению операций создания файла проекта базы данных

Выполнив работу, Вы будете:

уметь:

- создавать проекты базы данных;
- подключать файлы базы данных к проекту;
- создавать интерфейс входной формы.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MSSQLServer, MicrosoftVisualStudio.

Задание 1:

Создать файл проекта для базы данных Students.

Порядок выполнения работы:

Создание пользовательского интерфейса БД"Students"в "MicrosoftVisual Studio". Для начала необходимо создать новый проект. Для этого запустите "MicrosoftVisual Studio".

Появится окно со стартовой страницей"Microsoft Visual Studio".

Перед созданием нового проекта необходимо подключиться к базе данных "Student", для этого выбрать Сервис/Подключиться к базе данных… В окне Выбора источника данных выбрать MicrosoftSQLServer.

Для создания нового проекта на стартовой странице в области"Recent Projects"(Недавние проекты) необходимо щелкнутьЛКМ по ссылке"Create: Project…"(Создать: проект…). Появится окно выбора типа создаваемого проекта, и используемого языка программирования"New Project"(Новый проект).

В нашем случае на дереве типов проекта"Project types:"(Типы проектов) выберите"Visual Basic\Windows", а в качестве шаблона проекта (ОбластьTemplates:) выберите"Windows Forms Application"(ПриложениеWindows). В качестве имени проекта (Полеввода Name:) задайте"StudentsDB"и нажмите кнопку"Ok".

После создания нового проекта необходимо подключить к проекту созданную ранее в "MicrosoftSQLServer"БД"Students". Для подключенияБДк проекту в оконномменюсреды разработки выберитепункт"Data\Add New Data Source…". Появится окно мастера подключения к новому источнику данных"Data Source Configuration Wizard".

В данном окне можно выбрать один из трех видов источников данных (Choose a Data Source Type):

- БД (Database);
- Служба (Service);
- Объект (Object).

Так как мы подключаем наш проект кБД"Students"то выбираем вариантБД(Database) и нажимаем кнопку"Next"(Далее). Появится окно выбора подключения кБД(Choose Your Data Connection).

В окне выбора подключения кБД, для создания нового подключения нажмите кнопку"New Connection…". Появится окно добавления нового соединения"Add Connection".

В окне"Add Connection"в выпадающем списке"Server Name"(Имя сервера) выберите имя сервера заданное при установкеSQL сервера. В качестве логина и пароля для входа насервер(Log on to the server) выберите"Use Windows Authentication" (Использовать логин ипарольучетной

записиWindows). В качествеБДдля подключения (Connect to a database) из выпадающего списка"Select or enter a database name:"(Выберите или введите имяБД) выберитеБД"Students".

Для проверки работоспособности создаваемого соединения нажмите кнопку"Test Connection". Появится сообщение"Test connection succeeded", говорящее о том, что соединение работоспособно.

Закройте окно, а затем в окне добавления нового соединения"Add Connection"нажмите кнопку"Ok". Произойдет возвращение к окну выбора подключения кБД(Choose Your Data Connection). Просмотрите созданную строку подключения"Connection string", щелкнув по знаку"+"в нижней части окна.

В окне выбора подключения кБД(Choose Your Data Connection) нажмите кнопку"Next"(Далее). Появится окно с запросом о сохранении строки подключения"Save the Connection String to the Application Configuration File"(Сохранить строку подключения вфайл настроекприложения).

Для сохранения строки подключения включите опцию"Yes, save the connection as:"(Да, сохранить подключение как:) и нажмите кнопку"Next".

Появится окно выбора объектов подключаемойБД(Choose Your Database Objects).

Выберите все объекты и нажмите кнопку"Finish"(Готово). Подключение завершено.

Для просмотра источника данных щелкните по вкладке"Data Sources".

Закройте окно среды разработки. Появится окно сохранения нового проекта"Save Project".

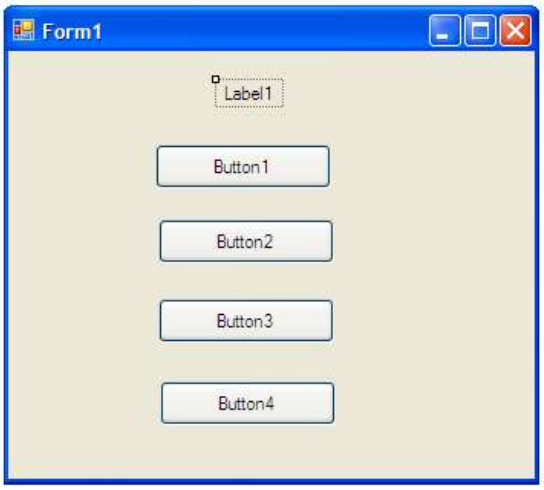
Задание 2:

Создать интерфейс входной формы для базы данных Students.

Порядок выполнения работы:

Перейдем теперь к созданию пользовательского интерфейса. Его создание начнем с создания главной кнопочной формы. Запустите "Microsoft Visual Studio" и откройте созданный ранее проект "StudentsDB".

После появления стандартного окна среды разработки в рабочей области на форму поместите надпись (Label) и четыре кнопки (Button) как показано нарисунке.



После создания объектов перейдем к настройке их свойств. Начнем с настройки свойств формы. Выделите форму, щелкнувЛКМв пустом месте формы. На панели свойств задайте свойства формы как представлено ниже:

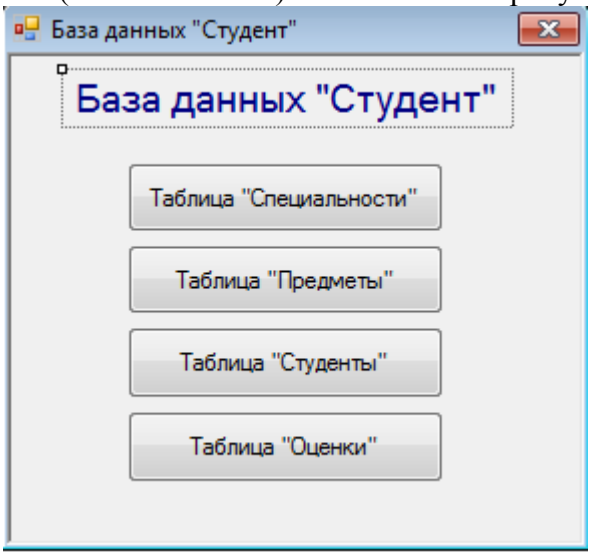
- FormBorderStyle(Стиль границы формы): Fixed3D;
- MaximizeBox(Кнопка развертывания формы во весь экран): False;
- MinimizeBox(Кнопка свертывания формы на панель задач): False;
- Text(Текст надписи в заголовке формы): База данных "Студент".

На форме выделите надпись, щелкнув по нейЛКМи на панели свойств, задайте свойства надписи следующим образом:

- AutoSize(Авторазмер): False;
- Font(Шрифт): MicrosoftSans Serif, размер 14;

- ForeColor(Цвет текста): Темно синий;
- Text(Текст надписи): База данных "Студент";
- TextAlign(Выравнивание текста): MiddleCenter.

У кнопок задайте надписи (свойство"Text") как показано нарисунке.



Форма представления результата:

Отчет по выполненной практической 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 2.2. Реализация баз данных в конкретной СУБД

Практическая работа №10
Создание формы. Управление внешним видом формы

Цель: получение практических навыков по освоению операций создания форм разными способами.

Выполнив работу, Вы будете:

уметь:

- создавать формы;
- добавлять различные элементы управления на форму;
- управлять внешним видом формы.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MSAccess, MSSQLServer, MySQL, PhpMyAdmin, MicrosoftVisualStudio.

Задание 1:

Разработать формы для базы данных Туризм в СУБД MSAccess.

Краткие теоретические сведения:

Для организации удобного интерфейса с базой данных используются формы. Форма позволяет вывести на экран одну запись в виде электронного бланка.

При создании формы в нее можно добавить объекты, улучшающие ее внешний вид и упрощающие работу с базой данных. К ним можно отнести поле ввода, надпись, кнопку, линии и прямоугольники. Большинство из них размещаются на ***Панели элементов.*** После выделения нужного элемента в панели его нужно растянуть на поле формы.

Макет формы состоит из разделов: область данных (содержит данные из источника), заголовок формы (верхняя часть первой страницы), примечание формы(нижняя часть последней страницы), верхний и нижний колонтитулы (при печати формы).

При создании форм в режиме **Конструктора** можно использовать также вычисляемые поля и подчиненные формы. Подчиненная форма – это форма, находящаяся внутри другой формы. Первичная форма называется главной, а форма внутри формы – подчиненной формой.

Подчиненная форма удобна для вывода данных из таблиц или запросов, связанных с отношением «один-ко-многим», «один-к-одному».

Главная форма и подчиненная форма в этом типе форм связаны таким образом, что в подчиненной форме выводятся только те записи, которые связаны с текущей записью главной формы.

При использовании формы с подчиненной формой для ввода новых записей текущая запись в главной форме сохраняется при входе в подчиненную форму. Это гарантирует, что записи из таблицы на стороне «многие» будут иметь связанную запись в таблице на стороне «один». Это также автоматически сохраняет каждую запись, добавляемую в подчиненную форму.

Подчиненная форма может быть выведена в **Режиме таблицы** как простая или ленточная форма. Главная форма может быть выведена только как простая.

Главная форма может содержать любое число подчиненных форм, если каждая подчиненная форма помещается в главную форму. Имеется также возможность создавать подчиненные формы двух уровней вложенности. Перед созданием подчиненных форм следует проверить связи «один-ко-многим» между таблицами.

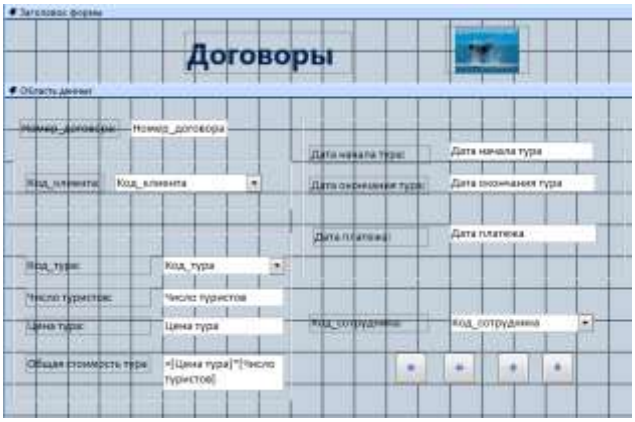
Порядок выполнения работы:

1. Откройте базу данных **Туризм**. Выберите на вкладке **Таблицы** таблицу **Клиенты**. Создайте для нее ***Автоформу***. Оцените результаты.
2. Зарегистрируйте новые договоры, используя кнопку со звездочкой, введите две новые записи.
3. Просмотрите в таблице новые данные и обратно закройте ее с сохранением.
4. Последовательно сделайте три ***Автоформы*** с различным размещением полей: *ленточная* / *в столбец* / *табличная*.

   Созданиеформы с помощью мастера
5. Создайте с помощью **Мастера форм** новую форму ***Сотрудники*** для одноименной таблицы. Включите в нее все поля исходной таблицы.
6. Выберите фон, на котором будут размещаться поля формы, перебрав в окне ***Мастера*** несколько вариантов оформления.
7. Завершите проектирование формы с помощью ***Мастера***.
8. Перейдите в режим *Конструктора*. Вставьте **Заголовок формы**.
9. Измените мышью расположение и ширину полей заголовка и размещение данных. Вернитесь в режим просмотра форм и оцените результаты. Добейтесь наилучших результатов размещения полей и заголовков формы.
10. Произведите сортировку данных по ***Дате рождения***.
11. Сохраните созданную форму.

    Созданиеформы с помощью Конструктора форм
12. Создайте форму для таблицы **Договоры** в режиме *Конструктора* форм.

    Для этого:
    - Кнопка **Создать** в окне базы данных – ***Конструктор*** – на основе таблицы **Договоры**;
    - Увеличить поле формы, растянув его за уголок;
    - Перетянуть каждое поле из окна ***Списки полей*** в область формы (если ***Спискаполей*** нет на экране, то можно его активизировать с помощью команды ***Вид – Список полей***);
    - Разместить поля по образцу;

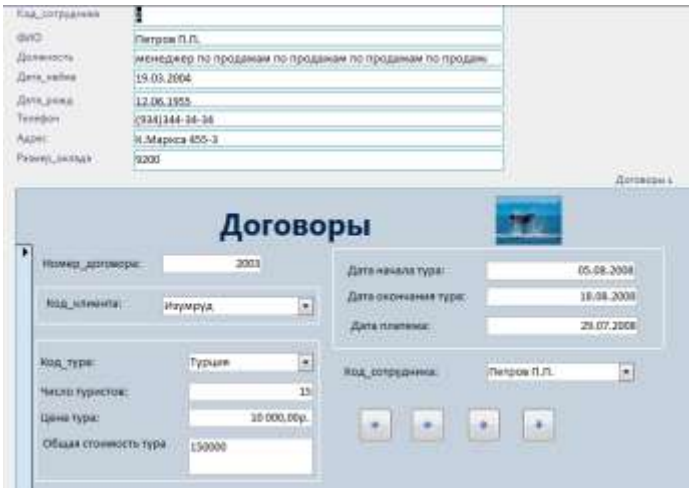


    - Добавить на форму некоторые дополнительные элементы, используя панель элементов: прямоугольники различных типов оформления, заголовок формы и др.
13. Измените размеры нескольких полей. Задайте группе полей одинаковые размеры, например ***По самому широкому***.
14. Задайте текст сообщения в строке состояния, которое будет появляться в момент ввода информации в поле (например, ***Дата окончания тура***). Для этого введите текст «*Окончание тура в день вылета до 12 часов*» в строке «*Текст строки состояния*» (контекстное меню поля ***Дата окончания тура*** – ***Свойства*** – вкладка ***Другие*** – «*Текст строки состояния*»). Проверьте в режиме формы, появляется ли в строке состояния заданный текст при активизации этого поля.
15. Задайте всплывающую подсказку «*Номер договора не должен повторяться*» для поля ***Номер договора*** (***Свойства*** – вкладка ***Другие*** – «*Всплывающая подсказка*»).
16. Добавьте кнопки для перехода к следующей и предыдущей записи, в конец и начало списка. Сохраните разработанную форму.

17. Включите в эту форму вычисляемое поле ***Общая стоимость тура***, которое рассчитывается как произведение значений поля ***Цена тура*** и поля ***Число туристов***. Для этого нужно создать поле с таким названием, и в его свойствах указать с помощью ***Построить*** расчетную формулу:
    =[Цена тура]*[Число туристов]

    Созданиеподчиненных форм

18. Постройте подчиненные формы для таблиц **Сотрудники** (отношение «один») и **Договоры** (отношение «много»).



19. Постройте подчиненную форму для таблиц **Клиенты** (отношение «один») и **Договоры** (отношение «много»).

Задание 2:

Разработать ленточные формы для базы данных Students.

Порядок выполнения работы:

Создадим ленточную форму, отображающую таблицу**"Специальности"**. Добавим в проект новую пустую форму. Для этого в оконномменювыберитепункт **"Project/Add Windows Form"**. Появится окно**"Add New Item - StudentsDB"**(Добавить новыйкомпонент).

В верхней части новой формы создайте надпись (**Label**).

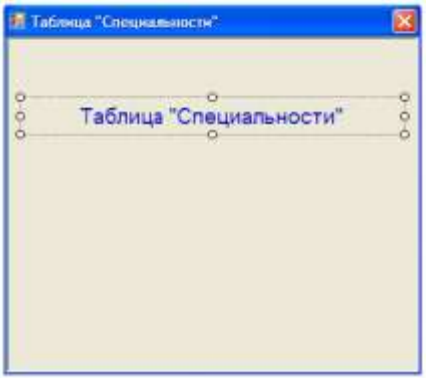
Перейдем к настройке свойств формы и надписи. Выделите форму, щелкнув**ЛКМ**в пустом месте формы. На панели свойств задайте свойства формы следующим образом:

- **FormBorderStyle**(Стиль границы формы): Fixed3D;
- **MaximizeBox**(Кнопка развертывания формы во весь экран): False;
- **MinimizeBox**(Кнопка свертывания формы на панель задач): False;
- **Text**(Текст надписи в заголовке формы): Таблица "Специальности".

На форме выделите надпись, щелкнув по ней**ЛКМ**и на панели свойств, задайте свойства надписи как показано ниже:

- **AutoSize**(Авторазмер): False;
- **Font**(Шрифт): MicrosoftSans Serif, размер 14;
- **ForeColor**(Цвет текста): Темно синий;
- **Text**(Текст надписи): Таблица "Специальности";
- **TextAlign**(Выравнивание текста): MiddleCenter.

После настройки всех вышеперечисленных свойств форма будет выглядеть следующим образом:



Теперь поместим на форму поля таблицы**"Специальности"**. Сначала откройте панель**"Источники данных"**(DataSources), щелкнув по ее вкладке в правой части окна среды разработки. На панели**"Источники данных"**отобразите поля таблицы**"Специальности"**, щелкнув по значку**"+"**, расположенному слева от имени таблицы.

Под полями таблицы специальности в видеподтаблицырасполагаетсятаблица**"Студенты"**.Подтаблица показывает, чтотаблица**"Студенты"**является вторичной по отношению к таблице специальности.

При выделении, какого-либо поля таблицы, оно будет отображаться в виде выпадающего списка, позволяющего выбиратьобъект, отображающий содержимое выделенного поля.

Для того чтобы поместить на новую форму поля таблицы их необходимо перетащить из панели**"Источники данных"**на форму. Из таблицы**"Специальности"**перетащите мышью на форму поля**"Наименование специальности"**и**"Описание специальности"**.

Мы не помещаемполе**"Код специальности"**на нашу форму, так как данноеполеявляется первичным полем связи и заполняется автоматически.Конечный пользовательне должен видеть такие поля.

Обратите внимание, что после перетаскивания полей с панели**"Источники данных"**на форму в верхней части формы появилась навигационная панель, а в нижней части рабочей области среды разработки появились пять невидимых объектов. Эти объекты предназначены для связи нашей формы с таблицей**"Специальности"**, расположенной на сервере.

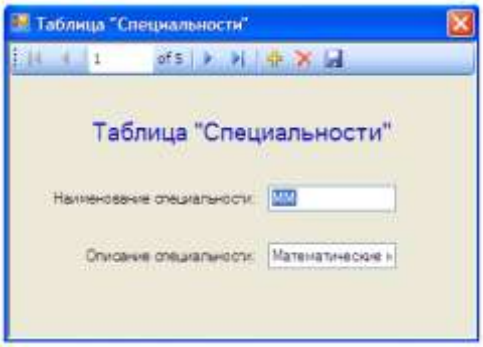
Теперь нам необходимо проверить работоспособность новой формы. Для отображения формы**"Специальности"**ее необходимо подключить кглавной кнопочной форме, а затем запустить проект и открыть форму**"Специальности"**при помощи кнопки на главной кнопочной форме.

Отобразитеглавную кнопочную формув рабочей области среды разработки, щелкнув по вкладке**"Form1.vb [Design]"**в верхней части рабочей области. Для подключения новой формы**"Специальности"**кглавной кнопочной формедважды щелкните**ЛКМ**по кнопке**"Таблица "Специальности""**, расположенной наглавной кнопочной форме. В появившемся окне кода формы в процедуре**"Button1_Click"**наберите команду**"Form2.Show()"**, предназначенную для открытия формы**"Таблица "Специальности""**(Form2).

```
Public Class Form1

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Form2.Show()
    End Sub
End Class
```

Теперь запустим проект.

На экране появитсяглавная кнопочная форма. Для открытия формы, отображающей таблицу**"Специальности"**наглавной кнопочной форме, нажмите кнопку**"Таблица "Специальности""**. Появится форма с соответствующей таблицей.



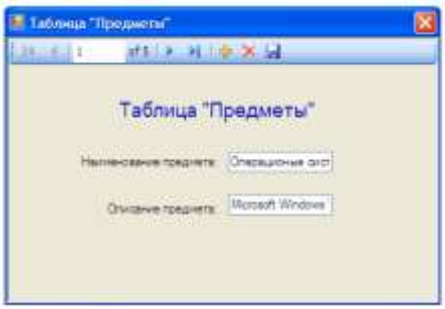
Проверьте работу панели навигации, расположенной в верхней части формы, понажимав на ней различные кнопки. Вернитесь в среду разработки, просто закрыв форму с таблицей**"Специальности"**иглавную кнопочную форму.

Теперь создадим форму для просмотра таблицы предметы. Добавьте в проект новую форму. На форму добавьте надпись. Настройте свойства формы и надписи, как это было сделано для формы таблицы**"Специальности"**. Затем из таблицы**"Предметы"**на новую форму поместите поля**"Наименование предмета"**и**"Описание предмета"**.

Наглавной кнопочной формедважды щелкните**ЛКМ**по кнопке**"Таблица "Предметы""**и в появившемся окне кода в процедуре**"Button2_Click"**наберите**"Form3.Show()"**.

```
    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
        Form3.Show()
    End Sub
End Class
```

Проверим работу новой формы, отображающей таблицу**"Предметы"**. Запустите проект и наглавной кнопочной форменажмите кнопку**"Таблица "Предметы""**. Отобразитсятаблицапредметы имеющая следующий вид:

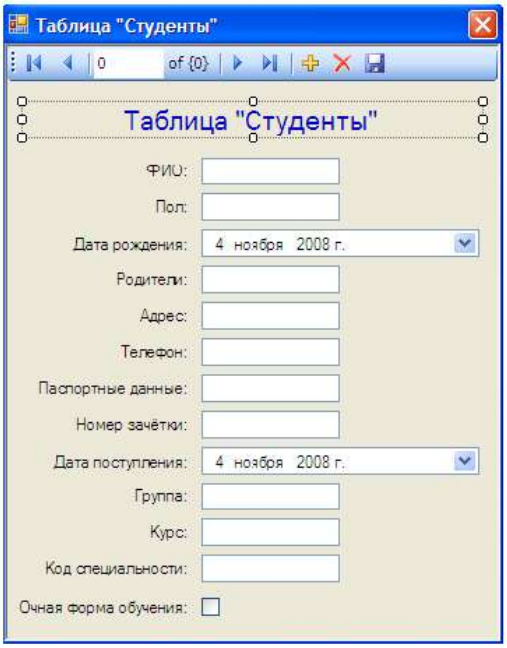


Проверьте работу панели навигации, нажатием на кнопки на данной панели в верхней части формы. Для возвращения в среду разработки закройте форму таблицы**"Предметы"**иглавную кнопочную форму.

Теперь создадим простую ленточную форму для отображения таблицы**"Студенты"**. Для начала отобразите поля таблицы**"Студенты"**на панели**"Источники данных"**, щелкнув**ЛКМ**по знаку**"+"**, расположенному слева от названия таблицы. Отобразятся все поля таблицы**"Студенты"**.

Обратите внимание на тот факт, что поля**"Дата рождения"**и**"Дата поступления"**отображаются объектом**"Выбор даты" (DataPicker)**, так как данные поля содержат значения дат.Поле**"Очная форма обучения"**является логическим, следовательно, для его отображения используетсяобъект**"Переключатель" (CheckBox)**. Остальные поля отображаются при помощи текстовых полей ввода (**TextBox**).

Создайте новую форму и поместите в ее верхнюю часть надпись. Задайте заголовок формы как**"Таблица "Студенты""**. В верхнюю часть формы поместите надпись. В качестве текста надписи задайте тот же самый текст, что был задан в качестве заголовка формы. Настройте свойства формы и надписи, аналогично формам, созданным ранее. На форму с панели**"Источники данных"**переместите все поля кроме поля **"Код студента"**. Так как данноеполеявляется первичным полем связи. Новая форма примет вид:

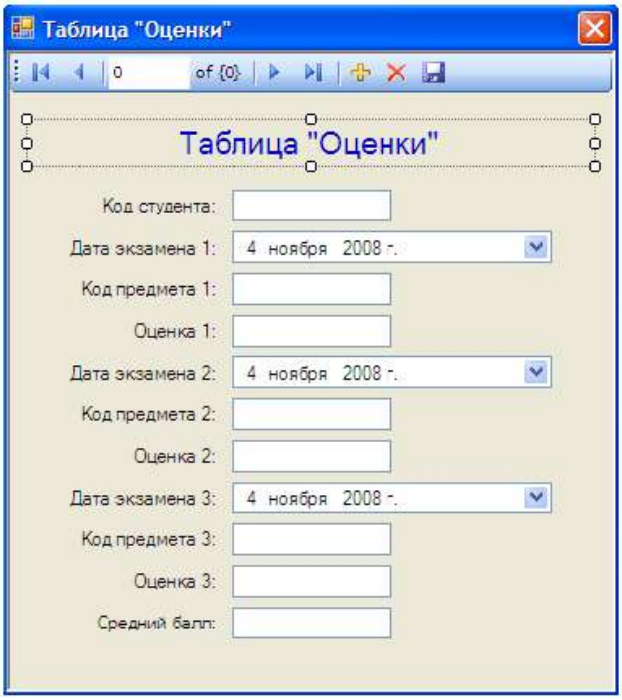


Обратите внимание на объекты, отображающие поля**"Дата рождения"**,**"Дата поступления"**и**"Очная форма обучения"**.

Подключим форму, отображающую таблицу**"Студенты"**кглавной кнопочной форме. Отобразитеглавную кнопочную формуи на ней дважды щелкните**ЛКМ**по кнопке**"Таблица "Студенты""**. В появившемся окне кода, в процедуре**"Button3_Click"**наберите следующую команду для открытия формы таблицы**"Студенты"**-**"Form4.Show"**.

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.
    Form4.Show()
End Sub
```

Теперь запустим проект. На экране появится*главная кнопочная форма*. Для открытия формы, отображающей таблицу**"Студенты"**на*главной кнопочной форме,*нажмите кнопку**"Таблица "Студенты""**. Появится форма с соответствующей таблицей.



Проверьте работу формы нажатием кнопок на панели навигации, расположенной в верхней части формы. Закройте форму, отображающую таблицу**"Студенты"**и*главную кнопочную форму*.

Аналогичным образом создайте форму для отображения таблицы**"Оценки"**. Добавьте на новую форму надпись, добавьте на форму все поля из таблицы**"Оценки"**и настройте их свойства, как описано выше. В итоге, форма для отображения таблицы**"Оценки"** будет выглядеть следующим образом:

Подключите вновь созданную форму таблицы**"Оценки"**к*главной кнопочной форме*. Для этого отобразите*главную кнопочную форму*и на ней дважды щелкните**ЛКМ**по кнопке**"Таблица "Оценки""**. В появившемся окне с кодом, в процедуре **"Button4_Click"**наберите команду**"Form5.Show"**(рис. 8.20).

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
    Form5.Show()
End Sub
```

Проверьте работу формы таблицы**"Оценки"**, запустив проект, и на*главной кнопочной форме*нажмите кнопку**"Таблица "Оценки""**. Появится вновь созданная форма.

В заключение, откройте обозреватель проекта (**Solution Explorer**) щелкнув по его вкладке в правой части окна среды разработки. На данной панели должны отобразиться все выше созданные формы.

Задание 3:

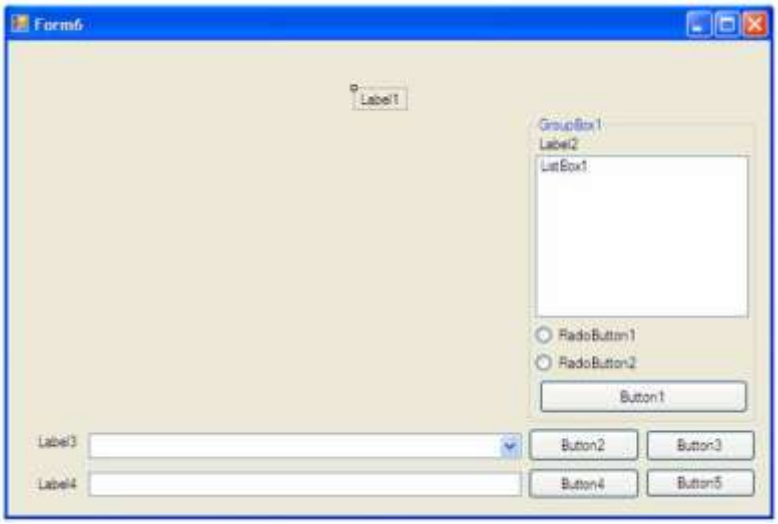
Разработать табличные формы для базы данных Students.

Порядок выполнения работы:

Рассмотрим создание табличной формы на примере формы, отображающей таблицу**"Студенты"**. Добавьте в проект новую форму и на нее поместите следующие объекты:

- четыре надписи (**Label**),
- пять кнопок (**Button**),
- выпадающий список (**ComboBox**),
- текстовое поле ввода (**TextBox**),
- группирующую рамку (**GroupBox**),
- список (**ListBox**),
- два переключателя (**RadioButton**).

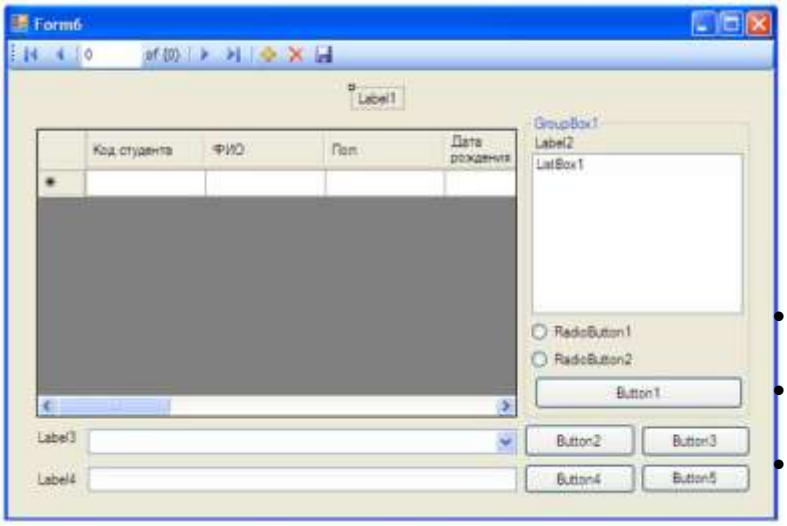
Расположите объекты как показано нарисунке.



Для создания объекта группирующая рамка используется кнопка ### на панели объектов (**Toolbox**), а для создания переключателя - кнопка ###.

Добавим на форму таблицу для отображения данных (**DataGridView**) из таблицы **"Студенты"**. Для этого на панели**"Источники данных" (Data Sources)**, нажмите кнопку  расположенную справа от таблицы**"Студенты"**. В появившемся списке объектов для отображения всей таблицы выберите **"DataGridView"**.

Перетащите таблицу**"Студенты"**из панели**"Источники данных"**на форму. Форма примет следующий вид:



Обратите внимание на то, что на форме появилась*таблица*для отображения данных, подключенная к таблице**"Студенты"**.

Теперь перейдем к настройке свойств объектов. Начнем с настройки свойств формы. Задайте свойства формы следующим образом:

- **FormBorderStyle (Стиль границы формы)**: Fixed3D;
- **MaximizeBox (Кнопка развертывания формы во весь экран)**: False;
- **MinimizeBox (Кнопка свертывания формы на панель задач)**: False;
- **Text (Текст надписи в заголовке формы)**: Таблица "Студенты" (Табличный вид).

Задайте свойства надписей (**Label1**,**Label2**,**Label3**и**Label4**) как:

- **AutoSize (Авторазмер)**: False;
- **Text (Текст надписи)**: "Таблица "Студенты" (Табличный вид)", "Поле для сортировки", "ФИО:" и "Критерий" (Соответственно для Label1, Label2, Label3 и Label4).

Для надписи**Label1**задайте:

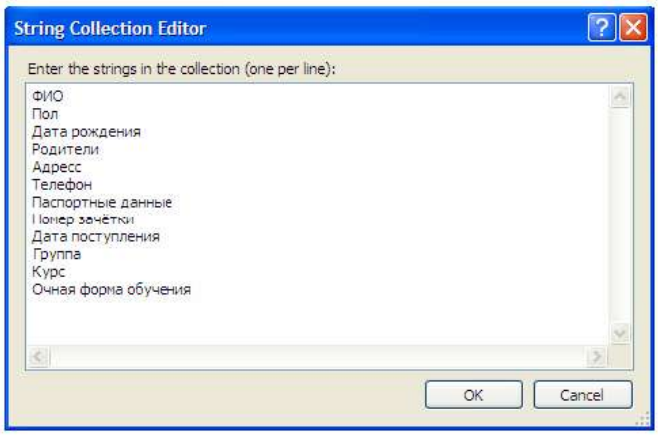
- **Font (Шрифт)**: Microsoft*Sans Serif*, размер 14;
- **ForeColor (Цвет текста)**: Темно синий;
- **TextAlign (Выравнивание текста)**: MiddleCenter.

Задайте надписи на кнопках как:**"Сортировать"**,**"Фильтровать"**,**"Показать все"**,**"Найти"**и**"Закрыть"**(Соответственно для

кнопок**Button1**,**Button2**,**Button3**,**Button4**и**Button5**). Для того чтобы нельзя было произвести сортировку, не выбрав поля изначально заблокируем кнопку**"Сортировать" (Button1)**.

У группирующей рамки задайте заголовок (Свойство**Text**) равным**"Сортировка"**. У переключателей (Объекты**RadioButton1**и **RadioButton2**) задайте надписи как**"Сортировка по возрастанию"**и**"Сортировка по убыванию"**, а у переключателя **"Сортировка по возрастанию" (RadioButton1)**задайте свойство**Checked (Включен)**равное**True (Истина)**.

Заполните*список*(**ListBox1**) значениями, представленными нарисунке, а затем нажмите кнопку**"Ok"**.



Настроим таблицу для отображения данных, удалив из нее поля с кодами. Выделите таблицу на форме и отобразите ее*меню* действий, щелкнув**ЛКМ**по кнопке  расположенной в верхнем правом углу таблицы. В*меню*действий выберите*пункт***"Edit columns…"**.

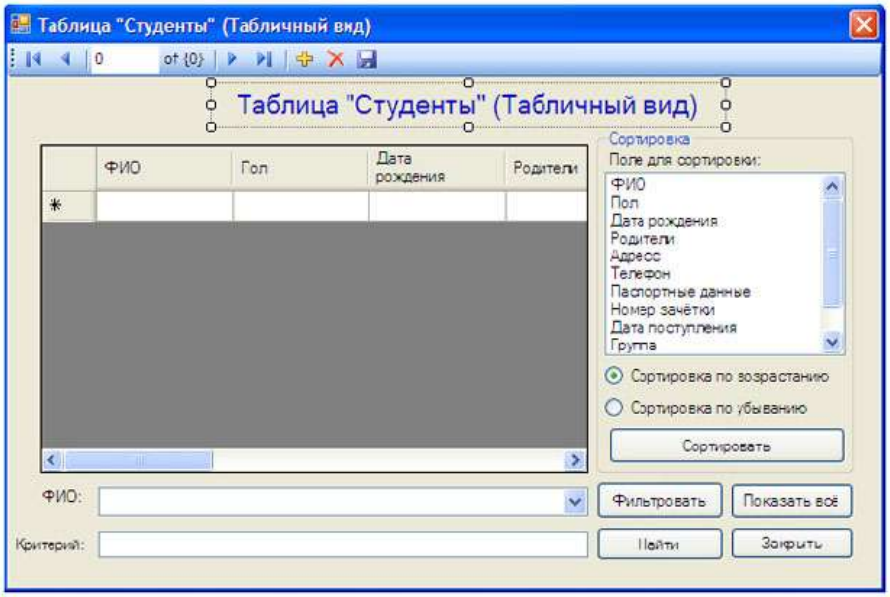
Появится окно настройки свойств полей таблицы**"Edit Columns"**

В окне**"Edit Columns"**из списка полей удалите поля**"Код студента"**и**"Код специальности"**, выделив их и нажав кнопку **"Remove"**(Удалить).*Список*полей примет вид показанный нарисунке 10.7. Для закрытия окна редактирования полей, и сохранения изменений нажмите кнопку**"Ok"**.

Настроим заполнение выпадающего списка именами студентов из таблицы студенты. Отобразите*меню*действий выпадающего списка. Включите опцию**"Use Data Bound Items"**. Установите*параметр***"Data Source"**равным**"Other Data Sources\Project Data Sources\StudentsDataSet\Студенты"**, а*параметр***"Display Member"**равным**"ФИО"**. Остальные параметры оставьте без изменений.

Закройте окно действий выпадающего списка. На панели невидимых объектов появится дополнительный*объект*связи **"СтудентыBindingSource1"**, предназначенный для заполнения выпадающего списка.

После настройки всех вышеперечисленных свойств объектов новая форма примет вид:



На этом мы заканчиваем настройку свойств объектов и переходим к написанию кода обработчиков событий объектов.

Работу с кодом начнем с написания кода для разблокирования кнопки**"Сортировать"**, при выборе пункта списка (**ListBox1**). Для создания *процедуры*события дважды щелкните**ЛКМ**по списку. Появится процедура обработки события, происходящего при выборе пункта списка

(**ListBox1_SelectedIndexChanged**). В процедуре наберите команду разблокировки кнопки**"Сортировать" (Button1)**:Button1.Enabled = True.

```
Private Sub ListBox1_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Button1.Enabled = True

End Sub
```

Теперь перейдем к созданию кода сортирующего нашу таблицу в зависимости от выбранного поля и порядка сортировки при нажатии кнопки**"Сортировать"**. Дважды щелкните**ЛКМ**по кнопке**"Сортировать"**. Появится процедура**"Button1_Click"**, выполняемая при щелчке**ЛКМ**по кнопке. В процедуре наберите код.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim Col As System.Windows.Forms.DataGridViewColumn
    Select Case ListBox1.SelectedIndex
        Case 0
            Col = DataGridViewTextBoxColumn2
        Case 1
            Col = DataGridViewTextBoxColumn3
        Case 2
            Col = DataGridViewTextBoxColumn4
        Case 3
            Col = DataGridViewTextBoxColumn5
        Case 4
            Col = DataGridViewTextBoxColumn6
        Case 5
            Col = DataGridViewTextBoxColumn7
        Case 6
            Col = DataGridViewTextBoxColumn8
        Case 7
            Col = DataGridViewTextBoxColumn9
        Case 8
            Col = DataGridViewTextBoxColumn10
        Case 9
            Col = DataGridViewTextBoxColumn11
        Case 10
            Col = DataGridViewTextBoxColumn12
    End Select
    If RadioButton1.Checked Then
        СтудентыDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Ascending)
    Else
        СтудентыDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Descending)
    End If

End Sub
```

Рассмотрим код обработчика события нажатия кнопки**"Фильтровать" (Button2)**. Дважды щелкните по кнопке**"Фильтровать"**и в процедуре обработки события**"Button2_Click"**наберите код:СтудентыBindingSource.Filter = "ФИО='" & ComboBox1.Text & "'".

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    СтудентыBindingSource.Filter = "ФИО='" & ComboBox1.Text & "'"

End Sub
```

У объекта**СтудентыBindingSource**имеется текстовое свойство**Filter**, которое определяет условие фильтрации. Условие фильтрации имеет*синтаксис*:"<Имя поля><Оператор>'<Значение>'".В нашем случае*значение*поля**"ФИО"** приравнивается к значению, выбранному в выпадающем списке (**ComboBox1.Text**).

Теперь перейдем к кнопке**"Показать все"**, отменяющей фильтрацию записей. Дважды щелкните по вышеперечисленной кнопке. Появится процедура**Button3_Click**. В появившейся процедуре наберите командуСтудентыBindingSource.Filter = "".

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
    СтудентыBindingSource.Filter = ""

End Sub
```

Заметим, что если присвоить свойству**"Filter"***значение*пустой строки (""), то его действие будет отменено.

Далее рассмотрим реализацию поиска информации в таблице. Дважды щелкните по кнопке**"Найти"**. В появившейся процедуре обработки нажатия кнопки**"Button4_Click"**наберите следующий код.

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
    For i = 0 To СтудентыDataGridView.ColumnCount - 1
        For j = 0 To СтудентыDataGridView.RowCount - 1
            СтудентыDataGridView.Item(i, j).Style.BackColor = Color.White
            СтудентыDataGridView.Item(i, j).Style.ForeColor = Color.Black
        Next j
    Next i
    For i = 0 To СтудентыDataGridView.ColumnCount - 1
        For j = 0 To СтудентыDataGridView.RowCount - 1
            If InStr(СтудентыDataGridView.Item(i, j).Value, TextBox1.Text) Then
                СтудентыDataGridView.Item(i, j).Style.BackColor = Color.AliceBlue
                СтудентыDataGridView.Item(i, j).Style.ForeColor = Color.Blue
            End If
        Next j
    Next i

End Sub
```

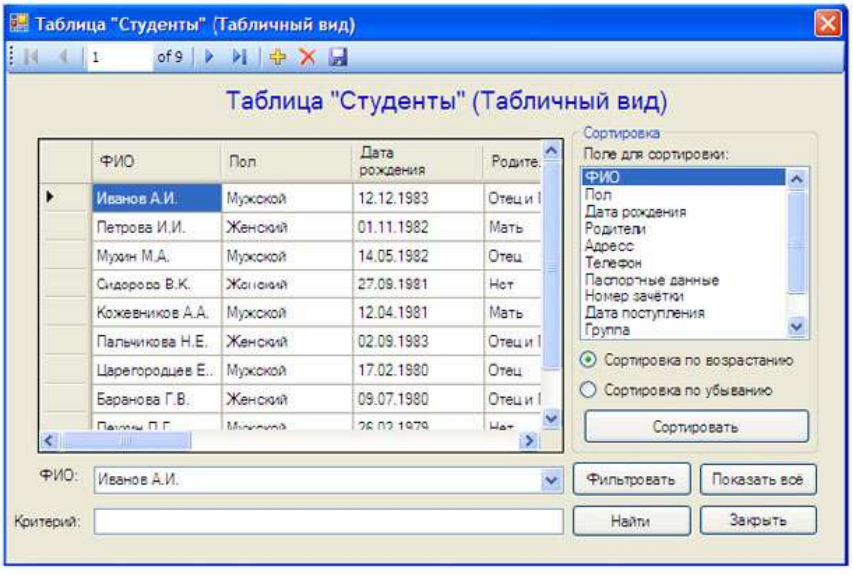
Наконец рассмотрим код для кнопки**"Закрыть"**. Дважды щелкните**ЛКМ**по этой кнопке и в появившейся процедуре **"Button5_Click"**наберите команду"Me.Close()", закрывающую выше рассматриваемую форму.

```
Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click
    Me.Close()

End Sub
```

В заключение создадим кнопку на ленточной форме, отображающей таблицу**"Студенты"**, для отображения соответствующей табличной формы. Откройте ленточную форму для таблицы**"Студенты" (Form4)**и поместите на нее новую кнопку.

Подключим к кнопке**"Таблица"**созданную ранее табличную форму (**Form6**). Для этого дважды щелкните**ЛКМ**по кнопке **"Таблица"**и в появившейся процедуре**"Button8_Click"**наберите команду"Form6.Show".

```
Private Sub Button8_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Form6.Show()
End Sub
```



Теперь проверим работоспособность созданной табличной формы. Запустите проект и на главной кнопочной форме нажмите кнопку**"Таблица "Студенты""**. На появившейся ленточной форме, отображающей таблицу**"Студенты"**нажмите кнопку **"Таблица"**. Появится новая табличная форма.

Проверьте, как работает*поиск*, фильтрация и*сортировка*записей в таблице, нажимая на соответствующие кнопки. После проверки работы формы для возвращения в среду разработки просто закройте все формы.

Форма представления результата:

Отчет по выполненной практической 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 2.2. Реализация баз данных в конкретной СУБД

Практическая работа № 11
Использование исполняемого файла проекта базы данных, приемы создания и управления

Цель: получение практических навыков по освоению операций создания и управления файла проекта базы данных

Выполнив работу, Вы будете:
уметь:
- создавать проекты базы данных;
- использовать исполняемый файл проекта базы данных.

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

Задание:
Для базы данных (по варианту) создать исполняемый файл проекта.

Краткие теоретические сведения:
Для соединения с базой данных необходимо создать объект класса PDO.

```
PDO: :_construct(
     string $dsn [,
     string $username [,
     string $password [,
     array $options]]])
```

Конструктор класса принимает в качестве первого параметра источник данных `$dsn`, содержащий название драйвера, адрес сервера и имя базы данных. Вторым параметром `$username` принимается имя пользователя, а третий параметр `$password` – его пароль. Последний параметр `$options` задает ассоциативный массив с дополнительными параметрами PDO и драйвера базы данных.

Пример:

```
$pdo = new PDO('mysql:host=localhost; dbname=sales', 'root', '');
```

Видывключений

Квидамвключенийотносятся`include`, `require`, `include_once`и`require_once`.

Команды`include`и`require`. почтиидентичны. Они лишь по-разному реагируют на ситуацию, `когда` указанный файл подключить невозможно: либо он не существует, либо у веб-сервера нет полномочий на его чтение. В таком случае `include` выводит предупреждение, при этом скрипт продолжает свою работу, а `require` отображает сообщение об ошибке, и скрипт останавливается.

Как правило, когда главный скрипт не способен продолжить работу без подключаемого файла, лучше `использовать` команду `require`. Однако по возможности старайтесь применять `include`. Если, например, файл `connect_db.php` не загрузится, скрипт сможет продолжить генерирование главной страницы.

Преимущество использования команды `include` для загрузки файлов заключается в том, что в одном скрипте она может применяться несколько раз, выводя разные шаблоны.

Команды include_onceиrequire_once работают аналогично своим прототипам include и require. Разница заключается в том, что, если указанный в первых двух выражениях файл уже хотя бы один раз был подключен в ходе текущего запроса к странице (с использованием любой из четырех команд), выражения игнорируются. Такой подход пригодится при подключении файлов, выполняющих одноразовые задачи, например, соединение с базой данных. Команды include_onceиrequire_once подходят также для загрузки библиотек с функциями.

Подключаемые файлы

В коде сайтов, написанных на PHP, часто требуется использовать один и тот же набор команд в разных местах. Вы уже научились работать с командой include при загрузке шаблонов внутрь контроллера. Оказывается, она также позволяет избежать многократного повторения одних и тех же фрагментов кода.

Подключаемые файлы (или **включения**) содержат фрагменты кода, которые доступны для загрузки другими PHP-скриптами, благодаря чему их не нужно набирать заново.

Подключение HTML-кода

Концепция подключения файлов появилась задолго до PHP. Технологиея Server Side Includes (SSI) — **включения на стороне сервера**. Поддерживаемая практически любым веб-сервером, SSI позволяет группировать часто повторяемые фрагменты HTML, JavaScript и CSS в отдельные файлы, которые затем используются на разных страницах.

В PHP подключаемые файлы чаще всего содержат чистый PHP-код или, если речь идет о шаблонах, смесь PHP и HTML. Обратите внимание, что хранить в таких файлах код на языке PHP не обязательно. При желании вы можете поместить туда сугубо статический фрагмент HTML. Такой способ хорошо подходит для выделения общих элементов дизайна, которые часто используются на сайте.

Порядок выполнения работы:

1. Создайте файл подключения к своей базе данных.

ПРИМЕР:

```php
<?php
// Соединение с базой данных
try
{
    $pdo = new PDO('mysql:host=localhost; port=3307; dbname=sales', 'root', '');
    $pdo->exec('SET NAMES "utf8"');
}
catch (PDOException $e)
{
    echo "Невозможно установить соединение с базой данных";
    include 'error.html.php';
    exit();
}
?>
```

2. Создайте форму для вывода данных из таблиц базы данных.

ПРИМЕР:

```php
<!DOCTYPE html>
<head>
<meta charset="utf-8">
<title>Вывод данных из базы данных</title>
</head>
<body>
    <h1>Товары</h1>
    <table>
        <tr>
            <td>Название</td>
            <td>Описание</td>
            <td>Цена</td>
            <td>Количество</td>
        </tr>
        <?php foreach($prod as $pro): ?>
        <tr>
            <td>
            <?php echo htmlspecialchars($pro['pname'], ENT_QUOTES, 'UTF-8'); ?></td>
            <td>
            <?php echo htmlspecialchars($pro['description'], ENT_QUOTES, 'UTF-8'); ?></td>
            <td>
            <?php echo htmlspecialchars($pro['price'], ENT_QUOTES, 'UTF-8'); ?></td>
            <td>
            <?php echo htmlspecialchars($pro['qty'], ENT_QUOTES, 'UTF-8'); ?></td>
        </tr>
        <?php endforeach; ?>
    </table>
</body>
</html>
```

3. Создайте контроллер для вывода информации.

ПРИМЕР:

```php
<?php ## Вывод содержимого таблицы product
  require_once("connect_db.php");
  try
  {
    $query = "SELECT pname, description, price, qty FROM product";
    $result = $pdo->query($query);
  }
  catch (PDOException $e)
  {
    echo "Ошибка выполнения запроса: " . $e->getMessage();
    include 'error.html.php';
    exit();
  }
  while ($row = $result->fetch())
  {
    $prod[]= array(
   'pname'=>$row['pname'],
   'description'=>$row['description'],
   'price'=>$row['price'],
   'qty'=>$row['qty']
    );
  }
   include 'product.html.php';
?>
```

4. Проверьте результат

ПРИМЕР:

Товары

Название	Описание	Цена	Количество
Утюг BOSH	Паровой удар, мощность 1000Вт	350.00	6
Холодильник INDESIT	Три камеры	15600.00	14
Стиральная машина BOSH	Загрузка вертикальная	6556.99	55
Стиральная машина Indesit	Загрузка горизонтальная	18000.00	4

Форма представления результата:

Отчет по выполненной практической 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 2.2. Реализация баз данных в конкретной СУБД

Практическая работа № 12
Импорт данных пользователя в базу данных

Цель: получение практических навыковимпортирования данных в базу данных

Выполнив работу, Вы будете:

уметь:

- создавать объекты базы данных;
- импортировать данные в базу данных.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MySQL, PhpMyAdmin, MSOffice.

Задание:

Выполнить импорт данных в базу данных.

Краткие теоретические сведения:

Если требуется добавить в таблицу большой массив данных, удобно использоватьдля этого команду загрузки данных из файла. Загрузка из файла выполняется программойMySQL 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 другот друга клавишу Tab, а для перехода на следующую строку – клавишу Enter.

Вместо отсутствующего значения необходимо при заполнении файлаввести символы «\N». Тогда в базу данных будет загружено неопределенноезначение (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 файлеemployee-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 практической 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 3.1. Администрирование баз данных

Практическая работа № 13
Создание представлений

Цель: получение практических навыков разработки представленийна языке SQL.

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

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

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.Перевозки из Челябинска во Владивосток. 2.Перевозки в Магнитогорск грузов в количестве более 500 кг. 3.Перевозки во Владивосток, совершенные не позднее 01.05.2017.
2	1.Абитуриенты, окончившие школу с золотой медалью. 2.Абитуриенты, поступающие на специальность «Электроснабжение» и проживающие в Челябинске и Магнитогорске. 3.Абитуриенты, окончившие школу с золотой медалью и сдавшие экзамен по математике на оценку «5».
3	1.Должность и тарифная ставка работника Р. Л. Иванова. 2.Работники отдела «Проектирование» с тарифной ставкой от 180 до 250р./ч. 3.Работники отдела «Проектирование», разряд которых выше 10-го.
4	1.Поступления товара «Сухое молоко». 2.Товары для организации «Восход» в количестве от 1000до5000 кг. 3.Товар «Сухое молоко», поступивший до 15.09.2017.
5	1.Книги, взятые на абонемент читателем Р.А. Петровым. 2.Книги, выданные в мае 2017 г. в количестве более 5шт. 3.Книги жанра «Наука», выданные читателю П.Л. Цветкову.
6	1.Модель автомобиля владельца А.Г. Зайцева. 2.Нарушения, допущенные водителем Г.Д. Беловым осенью 2017г. 3.Нарушения, допущенные владельцами автомобилей модели «Тойота» до 02.08.2017.
7	1.Данные о старте и финише участника соревнований Р.А. Краснова. 2.Участники команды«Юниор» спортивной организации «Чемпион». 3.Участники команды «Юниор», не вышедшие на старт.
8	1.Перевозки груза из Омска в Тюмень. 2.Перевозки груза в количестве от1 до5т в Екатеринбург. 3.Перевозки груза, отправленные в Москву не позднее10.09.2017.
9	1.Успеваемость студентов по математике. 2.Студенты, получившие по физике оценку «4» или «5». 3.Успеваемость студента А.П. Иванова по математике и физике.
10	1.Состояние лицевого счета абонента В.Д. Федорова. 2.Должники, имеющие в2017 г. льготу «Инвалидность».

	3.Абоненты, отключенные за неуплату во втором квартале2017г.
11	1.CD-диски с общим объемом файлов более 900 кбайт. 2.CD-диски с названием «Access», выпущенные с2017-го по 2018-й гг. 3.Владельцы CD-дисков с прикладным программным обеспечением.
12	1.Студенты, имеющие оценку «2» по химии. 2.Студенты, имеющие пропускизанятийпоматематике. 3.Успеваемость студента Р.Л. Ершова по математике и физике.
13	1.Постояльцы, проживающие в гостиничных номерах 5 и 40. 2.Постояльцы, которые забронировали гостиничный номер категории «люкс» и проживали в нем в июле2017г. 3.Постояльцы, проживающие в номерах со стоимостью места менее 1500 р.
14	1.Товары, полученные от поставщика – завода «Монолит». 2.Товары,проданныелетом2017г. вколичествеот1000 до3000шт. 3.Товары с наименованием «Миксер», поступившие до 01.02.2017.
15	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.
16	1.Поездки в Москву летом 2017 г. 2.Поездки пассажирского поезда № 5. 3.Поездки в Новосибирск поездов с количеством вагонов более 10.
17	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.
18	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 менее42кв. м.
19	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 практической 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 3.1. Администрирование баз данных

Практическая работа № 14
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: MySQL, MSSQLServer, PhpMyAdmin.

Задание1:
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:

```
{CREATE | ALTER }PROCEDURE имя_процедуры ([параметры])
BEGIN
  sql_оператор [...n]
END;
```

Для *выполнения хранимой процедуры* используется команда:

```
CALL имя_процедуры([параметры])
```

Входной параметр задается ключевым словом IN, выходной параметр – OUT.

Создание хранимой процедуры, которая будет добавлять новую должность, если такой еще нет.

```
    create procedure checkJob (in newName varchar(30), in newJob
varchar(30))
      begin
            declare str varchar(30);
       if not exists (select * from employee where empJob=newJob) then
        insert employee(empName,empJob) values(newName, newJob);
   else
       set str='У нас уже есть программист.';
      end if;
     end;
```

Порядок выполнения работы:

Варианты заданий по созданию хранимых процедур с вычисляемым полем

Вариант	Имя таблицы	Вычисляемое поле
1	Рейсы	Прибыль за рейс, выполненный судном
2	Анкета	Возраст абитуриента на текущую дату
3	Табель	Зарплата работника
4	Отпуск товаров	Доход от продажи товара
5	Выдачи	Просрочено дней читателем
6	Нарушители	Размер штрафа в долларах
7	Финиш	Размер бонуса, определяемый как разница порядковых номеров на старте и финише
8	Доставки	Количество дней доставки груза
9	Сессия	Индивидуальный код студента, представляющий собой сумму шифра студента и шифра дисциплины
10	Платежи	Сумма оплаты с учетом льготы абонента
11	CD-диски	Объем CD-диска в мегабайтах
12	Успеваемость	Индивидуальный код студента, представляющий собой сумму шифра студента и шифра дисциплины
13	Проживание	Сумма оплаты за проживание постояльца гостиницы
14	Продажа	Доход от продажи товара
15	Предприятия	Прибыль предприятия, рассчитанная в евро
16	Расписание	Прибыль за поездку, рассчитанная в долларах
17	Парк	Длительность маршрута
18	Недвижимость	Стоимость 1 кв. м общей площади
19	Перевозки	Количество свободных мест на рейсе

Варианты заданий по созданию хранимых процедур с групповыми операциями

Вариант	Имя таблицы	Итоговый показатель для расчета
1	Рейсы	Количество рейсов, выполненных каждым судном
2	Специальности	Количество анкет абитуриентов по каждой специальности
3	Работники	Количество отработанных часов каждым работником
4	Товары	Количество отпущенного товара по каждому наименованию
5	Книги	Количество книг, прочитанных каждым читателем
6	Нарушители	Количество нарушителей по каждому виду нарушений
7	Команда	Количество участников в каждой команде
8	Транспорт	Расстояние, пройденное каждым автомобилем
9	Дисциплина	Количество оценок «5» по каждой дисциплине

10	Абоненты	Количество абонентов по каждому виду льготы
11	Владельцы	Количество CD-дисков по каждому виду программного обеспечения
12	Студенты	Количество пропусков занятий по каждой дисциплине
13	Номерной фонд	Количество проживающих в гостиничных номерах каждой категории
14	Товары	Количество проданного товара по каждому наименованию
15	Предприятия	Сумма уплаченных налогов каждым предприятием
16	Поезда	Количество пассажиров, перевезенных каждым поездом
17	Парк	Количество автобусов по каждому маршруту
18	Сделки	Количество сделок, совершенных каждым риэлтором
19	Рейсы	Количество пассажиров, перевезенных каждым самолетом

Варианты заданий по созданию хранимых процедур с параметрами

Вариант	**Условие проецедуры с параметром**
1	Рейсы, совершенные судном N
2	Абитуриенты, поступающие на специальность N
3	Работники отдела N
4	Организации, которые приобрели товар N
5	Книги, выданные читателю N
6	Владельцы автомобилей, допустившие нарушение N
7	Команда, в состав которой входит участник N
8	Перевозки в пункт назначения N
9	Успеваемость по всем дисциплинам студента N
10	Льготы, которые имеет абонент N
11	Названия CD-дисков, принадлежащие владельцу N
12	Отметки о пропусках занятий студентом N
13	Постояльцы, проживающие в гостиничном номере категории N
14	Поставки товара поставщиком N
15	Продукция, выпускаемая предприятием N
16	Расписание движения пассажирского поезда N
17	Водители, работающие на автобусах по маршруту N
18	Сделки, совершенные риэлтором N
19	Рейсы, выполненные самолетами модели N

Задание2:

Разработать хранимые процедуры для базы данныхStudentsв СУБД MSSQLServer.

Порядок выполнения работы:

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

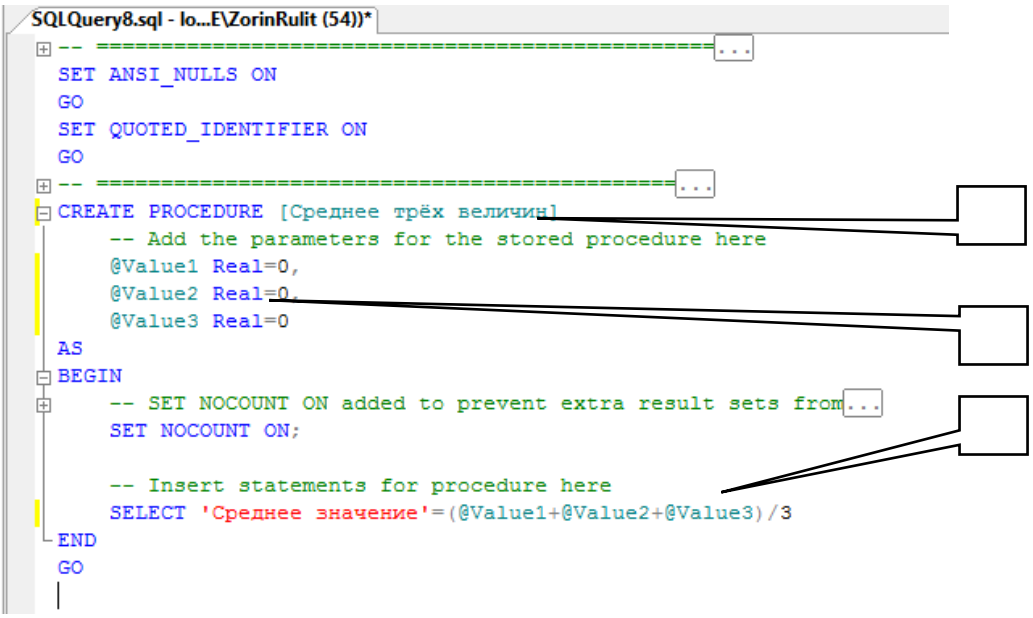
Создадим процедуру, вычисляющую среднее трех чисел. Для создания новой хранимой процедуры щелкните**ПКМ**по папке**" Хранимые процедуры "**и в появившемсяменювыберитепункт**"Создать хранимую процедуру"**. Появиться окно кода новой хранимой процедуры.

Хранимая процедураимеет 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. Конецтела 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 нарисунке 4.2.



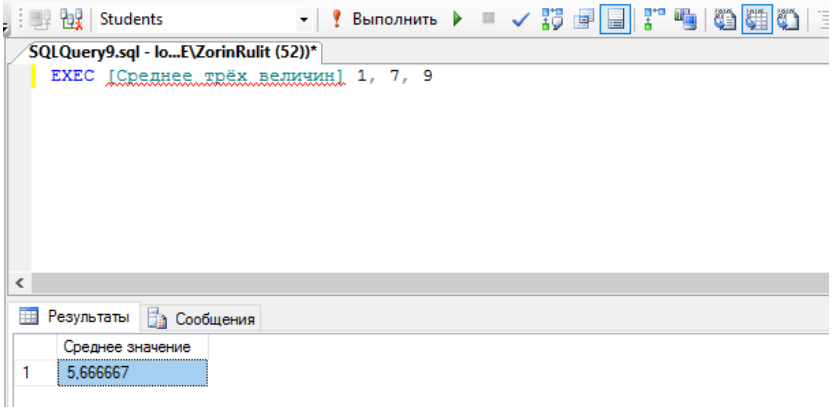
Рассмотрим код данной процедуры более подробно:

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 процедурыValue1,Value2и 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.

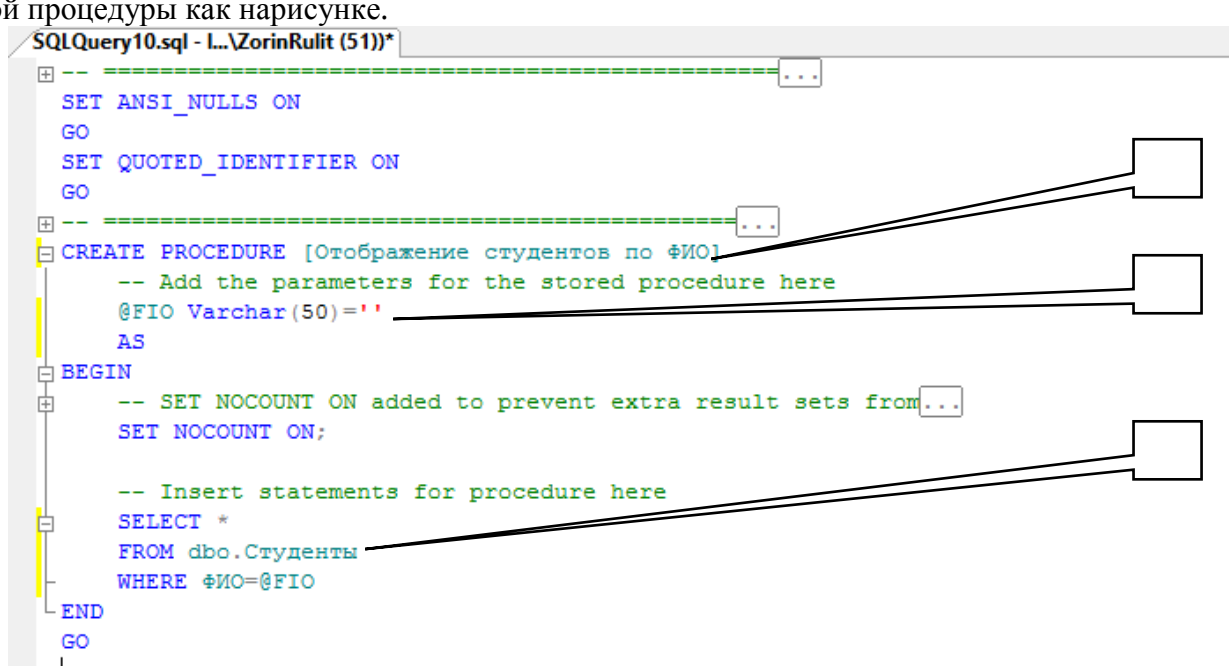
Для*создания процедуры*, выполним вышеописанный код, нажав кнопку Выполнить на панели инструментов. В нижней части окна с кодом появиться сообщение**"Выполнение команд успешно завершено."**. Закройте окно с кодом, щелкнув мышью по кнопке закрытия.

Проверим работоспособность созданной хранимой процедуры. Для запуска хранимой процедуры необходимо создать новый пустой запрос, нажав на кнопку Создать запрос на панели инструментов. В появившемся окне с пустым запросом наберите командуEXEC[Среднее трех величин] 1, 7, 9и 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 нарисунке.



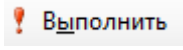
Рассмотрим код процедуры**"Отображение студентов по ФИО"**более подробно:

1. *CREATE PROCEDURE*[Отображение студентов по ФИО]- определяет имя создаваемой процедуры как "Отображение студентов по ФИО";

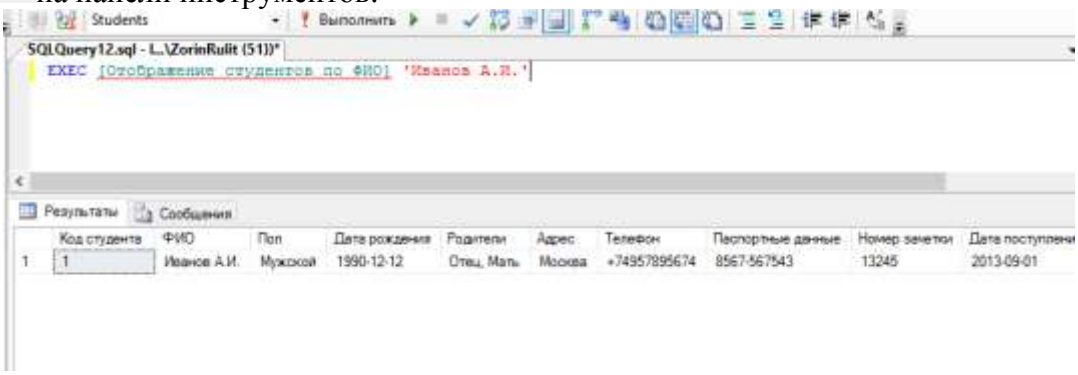
2. @FIO Varchar(50)="- определяют единственный параметр процедуры**FIO**. Параметру можно присвоить текстовые строки переменной длины, длинной до 50 символов (Тип данных Varchar(50)), значения по умолчанию равны пустой строке;
3. SELECT * FROM dbo.Студенты WHERE ФИО=@FIO- отобразить все поля (*) из таблицы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

Выполним вышеописанный код.

Проверим работоспособность созданной хранимой процедуры. Создайте новый пустойзапрос. В появившемся окне с пустым запросом наберите командуEXEC[Отображение студентов по ФИО] 'Иванов А.И.'и нажмите кнопку

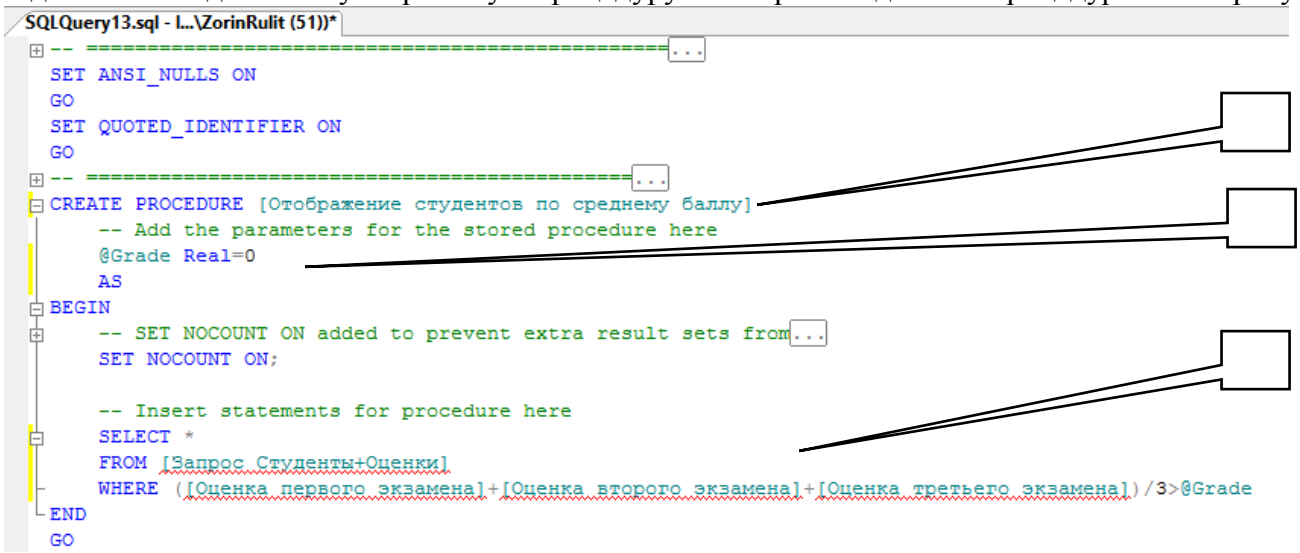


на панели инструментов.



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

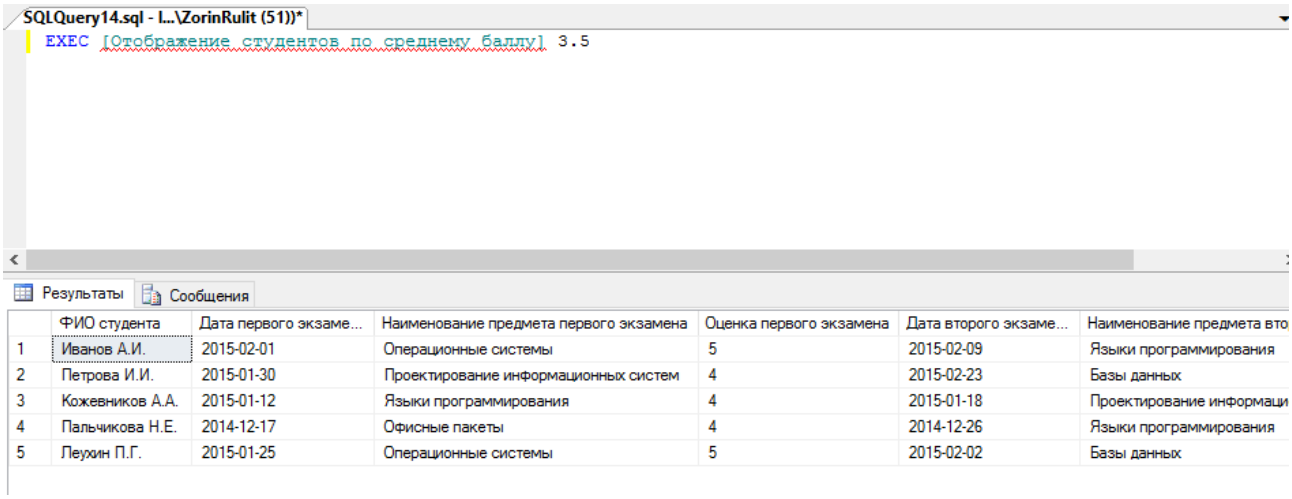
Теперь перейдем к следующей задаче - отобразить студентов, у которых средний балл выше заданного. Создайте новую хранимую процедуру и наберите код новой процедуры как нарисунке.



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 командуEXEC[Отображение студентов по среднему баллу] 3.5и выполните ее.

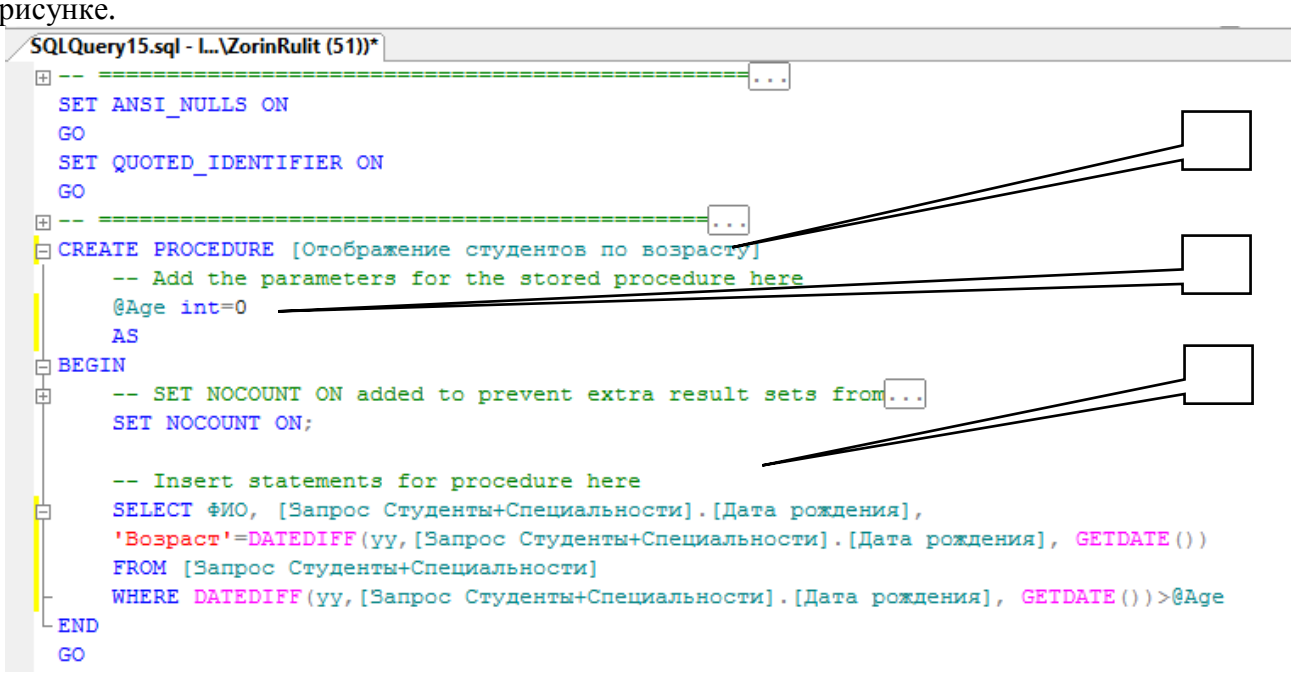


	ФИО студента	Дата первого экзаме...	Наименование предмета первого экзамена	Оценка первого экзамена	Дата второго экзаме...	Наименование предмета вто
1	Иванов А.И.	2015-02-01	Операционные системы	5	2015-02-09	Языки программирования
2	Петрова И.И.	2015-01-30	Проектирование информационных систем	4	2015-02-23	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 результатвыполнения хранимой процедуры**"Отображение студентов по среднему баллу"**.

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

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



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).

Встроеннаяфункция**DATEDIFF**вычисляющая количество периодов между двумя датами, имеет следующийсинтаксис: DATEDIFF(<период>,<начальная дата>, <конечная дата>)

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

Форма представления результата:

Отчет по выполненной практической 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 3.1. Администрирование баз данных

Практическая работа № 15
Создание триггеров

Цель: получение практических навыков разработки триггеров.

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

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

Задание 1:
Создать триггеры для базы данных в СУБД MySQL.

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:

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

Порядок выполнения работы:

Для базы данных Sales создать триггеры:

1. В добавляемой в таблицу Orders записи количество проданного товара должно быть не больше, чем его остаток из таблицы Product.
2. Создать триггер для обработки операции удаления записи из таблицы Orders. Для товара, код которого указан при удалении записи, необходимо откорректировать его остаток на складе.
3. Создать триггер для обработки операции изменения записи в таблице Orders. Для товара, код которого указан при изменении записи, необходимо откорректировать его количество на складе.

Задание2:
Создать триггеры для базы данных в СУБД MSSQLServer.

Порядок выполнения работы:

ВБД"MicrosoftSQLServer" 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 создадимтриггер, выводящий сообщение**"Запись добавлена"**при добавлении записи в таблицу**"Студенты"**. Создадим новыйтриггер, щелкнув**ПКМ**по папке**"Триггеры"**в таблице**"Студенты"**и в появившемсяменювыбравпункт**"Создать триггер"**. Появится следующее окно с новым триггером:

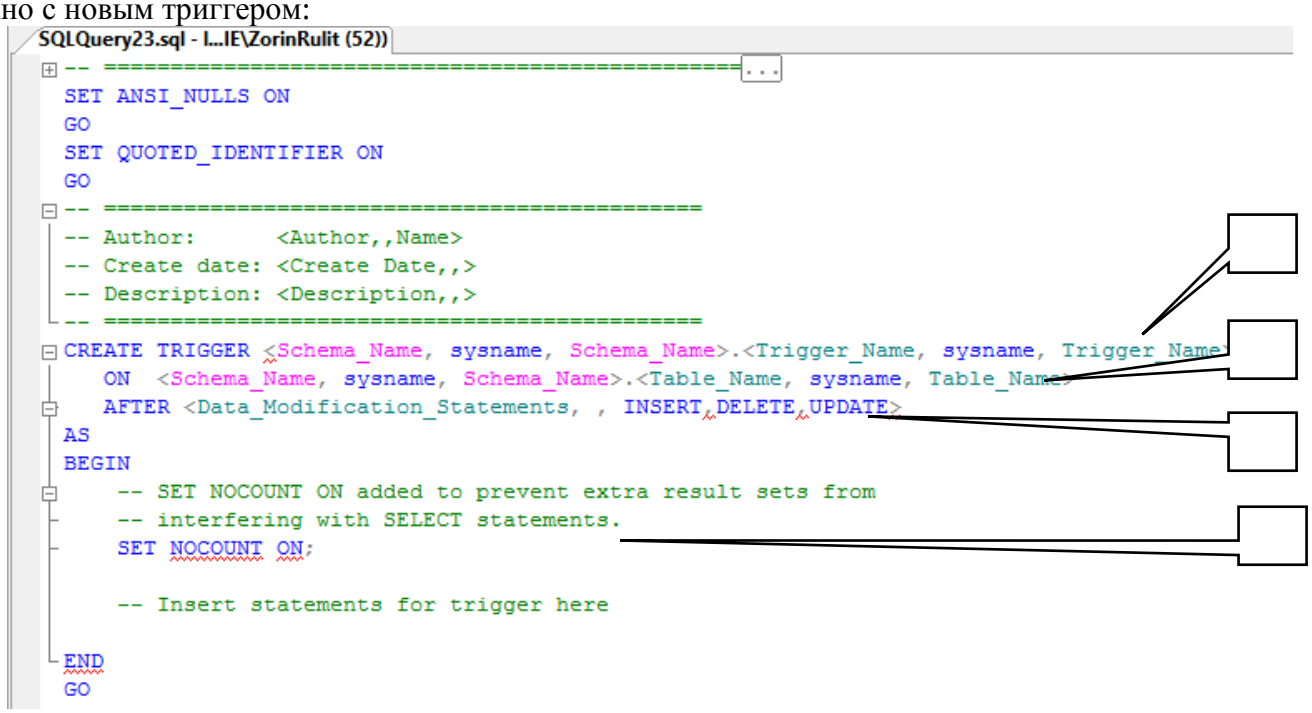


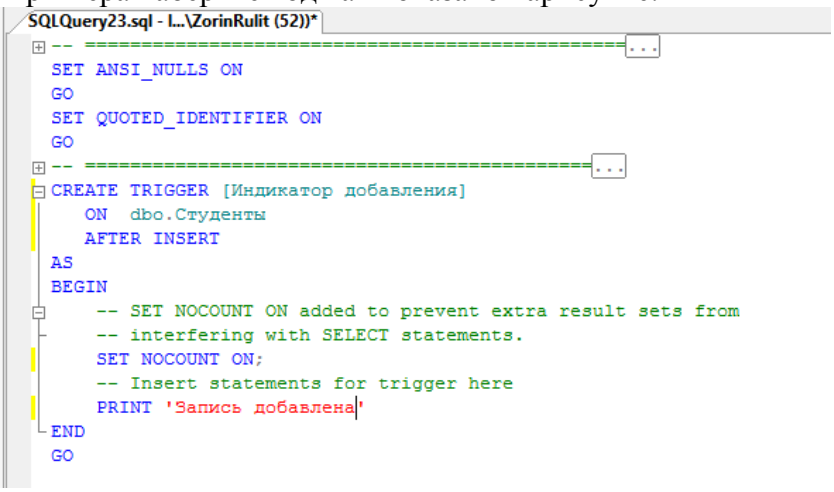
Рисунок 6.2

Рассмотрим структуру триггеров:

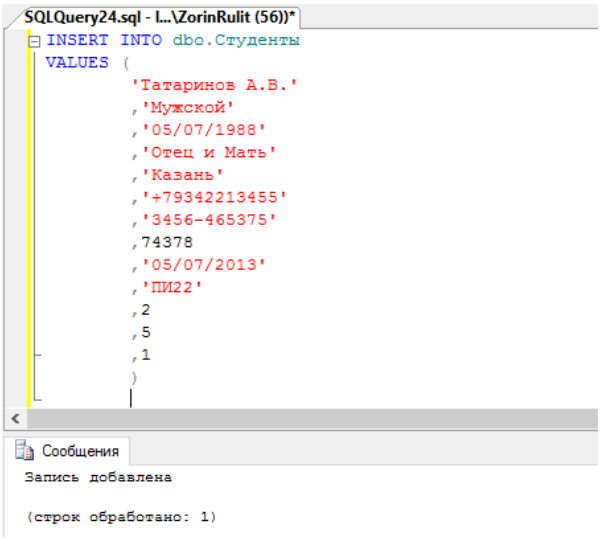
1. Область определения имени функции (**Trigger_Name**);
2. Область, показывающая для какой таблицы, создается триггер (**Table_Name**);

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 нарисунке.



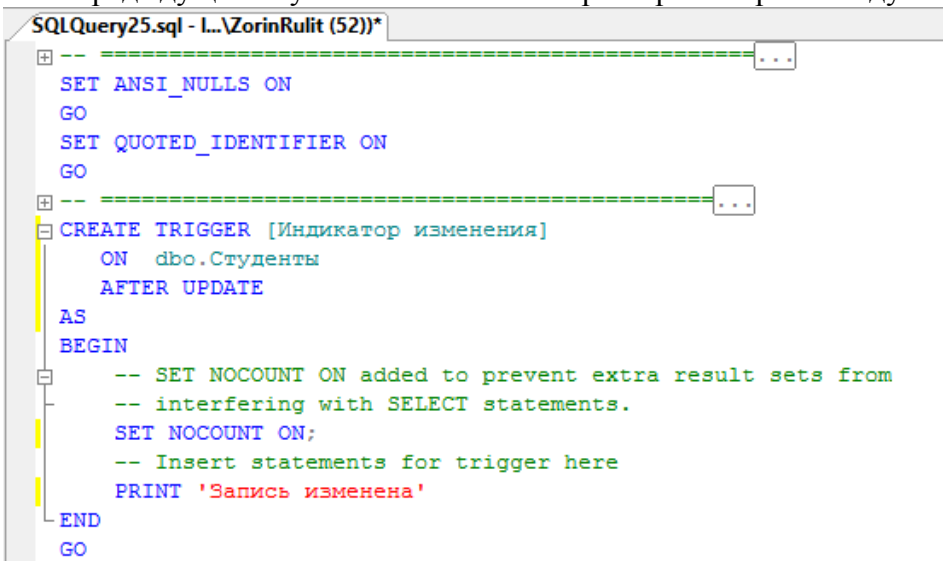
Изрисункавидно, что создаваемыйтриггер**"Индикатор добавления"**выполняется после добавления записи (**AFTER INSERT**) в таблицу "Студенты" (**ON dbo.Студенты**). После добавления записитриггервыведет на экран сообщение "Записьдобавлена" (**PRINT 'Запись добавлена'**). Выполните набранный код, нажав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



Проверим, как работает новыйтриггер. Создайте новый пустойзапроси в нем наберите следующую команду для добавления новой записи в таблицу**"Студенты"**:

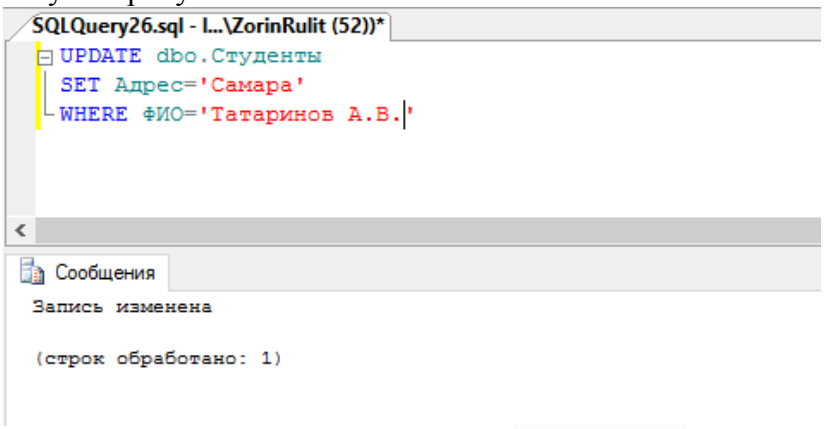
Выполните набранную команду, нажав кнопку Выполнить на панели инструментов. В таблицу будет добавлена новаязапись, итриггервыведет сообщение**"Запись добавлена"**.

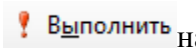
Теперь создадимтриггеротображающий сообщение**"Запись изменена"**. Создайте новыйтриггер, как в предыдущем случае. В окне нового триггера наберите следующий код:



Изрисунка видно, что новыйтриггер**"Индикатор изменения"**выполняется после изменения записи (**AFTER UPDATE**) в таблице "Студенты" (**ON dbo.Студенты**). После изменения записитриггервыведет на экран сообщение "Записьизменена" (**PRINT 'Запись изменена'**). Выполните набранный код. В нижней части окна с кодом появится сообщение**"Выполнение команд успешно завершено."**.

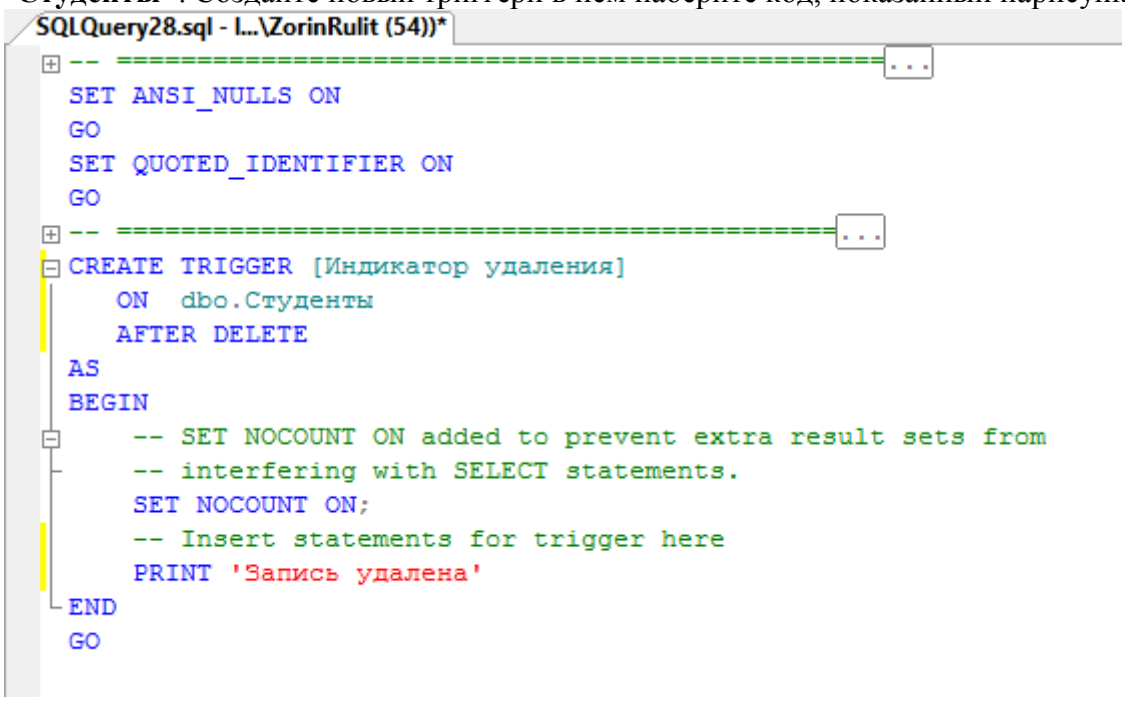
Проверим работоспособность созданного триггера. Создайте новый*запрос*и в нем наберите команду, представленную нарисунке 6.6.



Выполните набранную команду, нажав кнопку 

на панели инструментов. В таблице будет изменена однозапись, итриггервыведет сообщение**"Запись изменена"**.

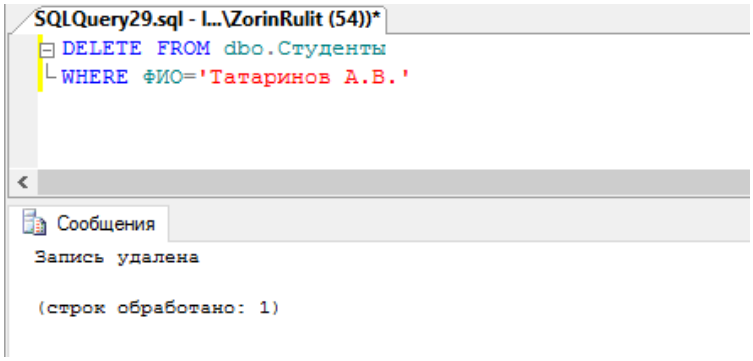
Для полноты картины создадимтриггер, выводящий сообщение при удалении записи из таблицы**"Студенты"**. Создайте новый триггери в нем наберите код, показанный нарисунке.



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

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

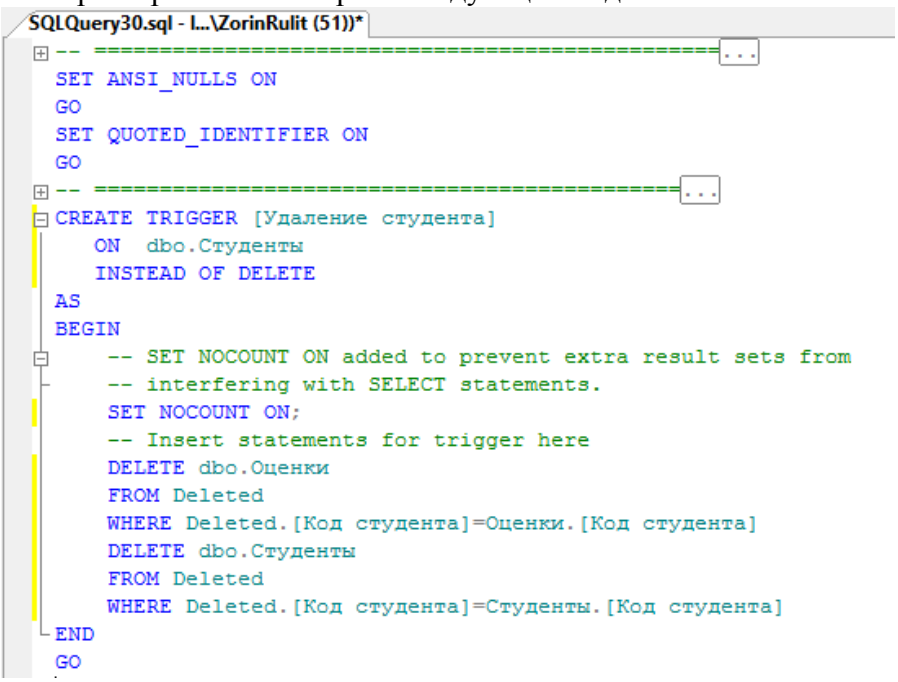
Проверим работу триггера**"Индикатор удаления"**удалив созданную ранеезаписьиз таблицы**"Студенты"**. Для этого создайте новыйзапроси в нем наберите следующую команду:



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

В заключение рассмотрим пример применения триггеров для обеспечения целостности данных. Создадимтриггер**"Удаление студента"**, который при удалении записи из таблицы студенты сначала удаляет все связанные с ней записи из таблицы**"Оценки"**, а затем удаляет самузаписьиз таблицы**"Студенты"**, тем самым обеспечиваетсяцелостность данных.

Создайте новыйтриггери в нем наберите следующий код:



Создаваемыйтриггер**"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**, содержащаяимя таблицыиз которой должно было быть произведено удаление.

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

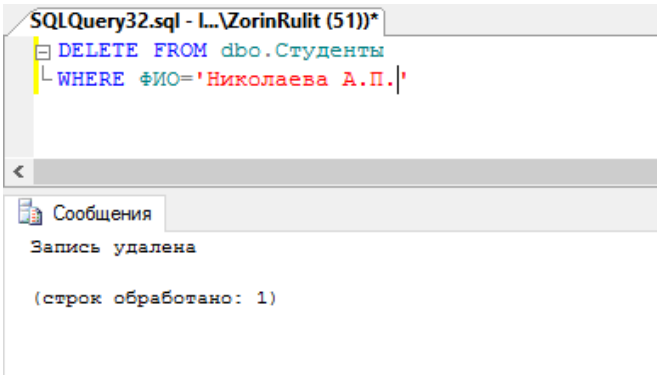
DELETE dbo.Оценки FROM Deleted
WHERE Deleted.[Код студента] = Оценки.[Код студента]

Затем удаляетсязаписьиз 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 работаеттриггер**"Удаление студента"**. Для этого создайте новыйзапроси в нем наберите следующий код:



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

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

Форма представления результата:

Отчет по выполненной практической 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 3.1. Администрирование баз данных

Практическая работа № 16
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 практических работ, вариант задания, компьютер, программное обеспечение: MySQL, PhpMyAdmin.

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 MySQL под пользователем user1 и во втором сеансе связи под полльзователем user2.
2. В базе данных sales создайте таблицу product:

```
create table product (
id serial,
name_prodvarchar(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.

```
1 SELECT * FROM product p;
```

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 (выполнитьанализ)
5. DELETE FROM product WHERE id =4	6. SELECT * FROM product (выполнитьанализ)
7. INSERT INTO product (name_pr, description, price, qty) VALUES ('Утюгпаровой', 'BOSH', 5500.00, 23)	8. SELECT * FROM product (выполнитьанализ)
	9. UPDATE product SET qty=qty+10 WHERE id=4 (выполнитьанализ)
10. DELETE FROM product WHERE id =4 (выполнитьанализ)	
	11. INSERT INTO product (name_pr, description, price, qty) VALUES ('Пароварка', 'BOSH', 3700.00,14) (выполнитьанализ)
12. ROLLBACK TRANSACTION trA	
	13.COMMIT TRANSACTION trB
14. LOCK TABLES product READ	
	15. SELECT * FROM product (выполнитьанализ)
16. SELECT * FROM product (выполнитьанализ)	
	17. UPDATE product SET qty=qty+20

	WHEREid=4 (выполнитьанализ)
18. UPDATE product SET qty=qty+20 WHERE id=4 (выполнитьанализ) 19. UNLOCK TABLES	
	20. (выполнитьанализ)
21. LOCK TABLES product WRITE	
	22. SELECT * FROM product (выполнитьанализ)
22. SELECT * FROM product (выполнитьанализ) 23. UNLOCK TABLES	
	24. (выполнитьанализ)

5. Составить отчет

Анализ выполнения запросов при блокировках и транзакциях

№	Пользователь user1	Пользователь user2

Форма представления результата:

Отчет по выполненной практической 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 3.1. Администрирование баз данных

Практическая работа № 17
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 практических работ, вариант задания, компьютер, программное обеспечение: MySQL, MySQLWorkbench, PhpMyAdmin.

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 заданиипути к файлу необходимо использовать прямую косую черту вместо принятой в Windowsобратной косой черты. Указанный файл создается на компьютере, на котором работаетсервер 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 командыSELECT

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

можно загрузить в таблицу Customers_copy (имеющую такую же структуру, что итаблица 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 Wordи Excel. Создать копии таблиц и выполнить загрузку данных из файла в базу данных.

Форма представления результата:
Отчет по выполненной практической 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 3.1. Администрирование баз данных

Практическая работа № 18
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: MySQL, PhpMyAdmin.

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:

```
GRANT {<привилегия>[,...n] |
ALLPRIVILEGES}
ON имя_объекта
TO {<идентификатор_пользователя>
[,...n]| PUBLIC}
[ WITH GRANT OPTION]
```

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:

```
REVOKE[GRANT OPTION FOR]
   {<привилегия>[,...n]
| ALLPRIVILEGES}
ON имя_объекта
FROM {<идентификатор_пользователя>
[,...n]| PUBLIC}
[RESTRICT | CASCADE]
```

Порядок выполнения работы:

Варианты заданий

Вариант	Наименование базы данных, пользователи и их права
1	*БД «Морские перевозки».* Администратор (доступны все таблицы для полного управления) Менеджер (доступны суда и рейсы для полного управления, грузы для просмотра) Контроллер (доступны суда и рейсы для просмотра, грузы для полного управления)
2	*БД «Абитуриенты».* Администратор (доступны все таблицы для полного управления) Отдел управления (доступны специальности, дисциплины для полного управления, анкета, результаты для просмотра) Сотрудник (доступны анкета, результаты для полного управления, специальности, дисциплины для просмотра и добавления)
3	*БД «Зарплата».* Администратор (доступны все таблицы для полного управления) Отдел кадров (доступны должности, тарифы для полного управления, работники, табель для просмотра и изменения и добавления) Сотрудник (доступны работники, табель для полного управления, должности, тарифы для просмотра, изменения и добавления)
4	*БД «Оптовая база».* Администратор (доступны все таблицы для полного управления) Менеджер (доступны товары и склад для полного управления, заявки для просмотра) Продавец (доступны заявки и отпуск товаров для просмотра, изменения и добавления, товары и склад для полного управления)
5	*БД «Библиотека».*

	Администратор (доступны все таблицы для полного управления) Отдел обработки книг (доступны читатели, выдачи для просмотра, книги, жанры для полного управления) Библиотекарь (доступны жанры, книги для просмотра, читатели и выданные книги для полного управления)
6	*БД «ГИБДД».* Администратор (доступны все таблицы для полного управления) Начальник отдела (доступны виды нарушений для полного управления, автомобили, владельцы, нарушители для просмотра и добаления) Постовой (доступны автомобили, владельцы, нарушители для полного управления, виды нарушенийдля просмотра)
7	*БД «Соревнования».* Администратор (доступны все таблицы для полного управления) Менеджер (доступны команда, участники для полного управления, старт, финиш для просмотра и добаления) Сортудник (доступны старт, финиш для полного управления, команда, участники для просмотра, добавления и изменения)
8	*БД «Перевозки».* Администратор (доступны все таблицы для полного управления) Менеджер (доступны транспорт для полного управления, заявки, доставка для просмотра и добаления) Оператор (доступны заявки, доставка для полного управления, транспорт для просмотра, добавления)
9	*БД «Сессия».* Администратор (доступны все таблицы для полного управления) Учебный отдел (доступны кафедры и дисциплины, преподаватели для полного управления, студенты, оценки для просмотра) Преподаватель (доступны студенты, оценки для полного управления, кафедры и дисциплины, преподаватели для просмотра и добавления)
10	*БД «Телефонная компания».* Администратор (доступны все таблицы для полного управления) Менеджер (доступны тарифы, виды льгот для полного управления, абоненты, платежи для просмотра и редактирования) Оператор (доступны абоненты, платежи для полного управления, тарифы, виды льгот для просмотра, добавления)
11	*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)
12	*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)
13	*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)

14	*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)
15	*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)
16	*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)
17	*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)
18	*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)
19	*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 практической 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 3.1. Администрирование баз данных

Практическая работа № 19
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 практических работ, вариант задания, компьютер, программное обеспечение: MSSQLServer.

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='alter index' + @indexname + 'on' + @shemaname + '.'+@objectname+'reorganize';
  if @frag>=30   --если общая фрагментация индекса больше 30, нечинается перестроение
 set @command='alter 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 практической 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 3.1. Администрирование баз данных

Практическая работа № 20
Мониторинг работы сервера

Цель: получение практических навыков по освоению операциймониторинга работы сервера

Выполнив работу, Вы будете:
уметь:
- выполнять мониторинг работы сервера.

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

Задание:
Выполнить мониторинг работы сервера MySQL.

Краткие теоретические сведения:
Расширенный мониторинг баз данныхMySQLс помощью бесплатной системы мониторингаIcinga2.

Рассмотрим необходимые плагины мониторинга.

Требования

Мониторинг службы базы данных начинается с проверки наличия самой службы. Благодаря сквозному мониторингу, мы можем фактически подключаться кMySQL, а не просто проверять, работает ли этот процесс. Другими словами, серверIcinga2должен иметь возможность подключаться к сервисуMySQL, работающему на сервере.

Существуют различные плагины мониторинга, которые можно использовать сIcinga2для проверки сервераMySQL. Все они являются частью стандартной установки инструментов мониторинга, поэтому можно использовать их без “изобретения велосипеда”:

- check_mysql: простой плагин, который можно использовать для проверки подключений к серверуMySQL;
- check_mysql_query: отправляетSQL-запросы на серверMySQLи проверяет их результаты на разные пороговые уровни;
- check_mysql_health: более сложный плагин для мониторинга времени безотказной работы, времени соединения, связанных потоков, попадания в поток или кеш запросов, задержки между ведущим и ведомым серверами и т. д.

Порядок выполнения работы:
Настройте свой файрволл соответствующе.

```
firewall-cmd --add-rich-rule 'rule family="ipv4" source address="$IP_or_IP_Range" service name="mysql" accept' --permanent
```

Загрузите плагин check_mysql_health

Прежде чем начать, обязательно создайте отдельного пользователя подMySQL для мониторинга. Он должен иметь ограниченные привилегии, потому что его пароль в открытом виде будет использоваться в конфигурационных файлахIcinga2. Для простой проверки соединения достаточно минимальных разрешений:

```
GRANT USAGE ON mysql.* TO 'monitoring'@'Icinga2.local' IDENTIFIED BY 'BEST_PASSWORD';
```

Мониторинг single-сервер MySQL

Спервасоздадимконфигурационныйфайлхоста,/etc/icinga2/conf.d/mpk.biz.conf соследующим содержимым:

```
object Host "mpk.biz" {
  import "generic-host"
  address = "IP_ADDRESS"

  vars.os = "Linux"
  vars.mysql = true
}
```

Основываясь на переменных, которые создали для хоста, теперь можем применить правила динамических сервисов.

Мониторинг доступности MySQL-сервера

Создаем конфигурационный файл/etc/icinga2/conf.d/apply_mysql.conf, со следующим содержимым:

```
apply Service "mysql-connect" {
  import "generic-service"

  display_name = "MySQL Connect"
  check_command = "mysql_health"

  vars.mysql_health_mode = "uptime"
  vars.mysql_health_username = "monitoring"
  vars.mysql_health_password = "BEST_PASSWORD"

  assign where host.vars.mysql == true
}
```

Это создаст службу с отображаемым именемMySQL Connectдля всех узлов, на которых определена пользовательская переменнаяvars.mysql. В случае с зонами - переменная будет только на указанных хостах.

Данный пример показывает время работы сервераMySQL(vars.mysql_health_mode= «время безотказной работы»).

Вывод вIcinga2Webбудет приблизительно следующий:

```
OK - database is up since 27 minutes
```

Мониторинг количества подключенных клиентов

По аналогии с предыдущим сервисом изменим режим работы плагина nathreads-connected:

```
apply Service "mysql-threads-connected" {
  import "generic-service"

  display_name = "MySQL threads connected"
  check_command = "mysql_health"

  vars.mysql_health_mode = "threads-connected"
  vars.mysql_health_warning = "25"
  vars.mysql_health_critical = "100"
  vars.mysql_health_username = "monitoring"
  vars.mysql_health_password = "BEST_PASSWORD"

  assign where host.vars.mysql == true
}
```

Вывод вIcinga2Webбудет приблизительно следующий:

```
OK - 1 client connection threads
```

Мониторинг задержки при репликации

Если мы имеем связку серверов с репликацией, то используя плагинcheck_mysql_healthможно контролировать задержку между серверами, особенно, если сервера разнесены по разным геолокациям.

Сперва создадим на MySQL кластере нового пользователя для мониторинга:

```
GRANT REPLICATION CLIENT ON *.* TO 'monitoring'@'%' IDENTIFIED BY
'BEST_PASSWORD';
```

Затемсоздадимновыйконфигурационныйфайлхоста,/etc/icinga2/conf.d/replication_bogachev.biz.confсоследующимсодержимым:

```
object Host "bogachev.biz" {
  import "generic-host"
  address = "IP_ADDRESS"

  vars.os = "Linux"
  vars.mysql = true
  vars.mysql_slave = true
}
```

Создаем конфигурационный файл/etc/icinga2/conf.d/apply_mysql.conf, (*либо используем существующий файл сервисов*) со следующим содержимым:

```
apply Service "mysql-slave" {
  import "generic-service"

  display_name = "MySQL Slave Lag"
  check_command = "mysql_health"

  vars.mysql_health_mode = "slave-lag"
  vars.mysql_health_warning = "3"
  vars.mysql_health_critical = "10"
  vars.mysql_health_username = "monitoring"
  vars.mysql_health_password = "BEST_PASSWORD"

  assign where host.vars.mysql && host.vars.mysql_slave
}
```

Не забудьте изменить пороговые значения на те, что подходят для вашей конфигурации. (*По умолчанию Warning - 10 и Critical - 20 секунд*)

Мониторинг размера базы

Плагин поддерживает отправкуSQL-запросов для получения необходимой информации из базы. Ради примера рассмотрим вариант с получениями размера базы данных.

Для получения размера базы будет использован следующийSQL-запрос:

```
SELECT SUM(data_length + index_length) / 1024 / 1024 AS 'db size'
FROM information_schema.tables
WHERE table_schema = 'TEST_DB';
```

Пример вывода:

```
+---------------+
| db size       |
```

```
+---------------+
| 2857.14062500 |
+---------------+
```

Режимsqlплагинacheck_mysql_healthпозволяет отправлять этотSQL-запрос в базу данных.Icinga2сможет не только получать информацию о размере, но и определять пороговые значения для предупреждений или критических состояний.

Определим новый конфигурационный файл хоста,/etc/icinga2/conf.d/sql_bogachev.biz.confco следующим содержимым:

```
object Host "bogachev.biz" {
  import "generic-host"

  address = "bogachev.biz"

  vars.os = "Linux"
  vars.mysql = true

  vars.database["TEST_DB"] = {
    mysql_health_username = "monitoring"
    mysql_health_password = "BEST_PASSWORD"
    mysql_health_warning = 2048
    mysql_health_critical = 8192
  }
}
```

Создаем конфигурационный файл/etc/icinga2/conf.d/apply_mysql.conf, (либо используем существующий файл сервисов) со следующим содержимым:

```
apply Service "db_size" for (db_name => config in host.vars.database) {
  import "generic-service"

  display_name = "DB Size " + db_name
  check_command = "mysql_health"

  vars.mysql_health_mode = "sql"
  vars.mysql_health_name = "SELECT SUM(data_length + index_length) / 1024 / 1024 AS 'db size'
FROM information_schema.tables WHERE table_schema = '" + db_name  +"';"
  vars.mysql_health_name2 = "db_size"
  vars.mysql_health_units = "MB"

  vars += config
}
```

Форма представления результата:

Отчет по выполненной практической 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 3.2. Защита баз данных

Практическая работа № 21
Выполнение резервного копированияи восстановления базы данных из резервной копии

Цель: получение практических навыков по освоению операций резервного копирования

Выполнив работу, Вы будете:

уметь:

- выпонять резервное копирование базы данных.

Материальное обеспечение:

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

Задание:

Создать резервную копию базы данных.

Краткие теоретические сведения:

Существует несколько вариантов резервирования баз данных.

1. С помощью сценария *mysqldump*, запустив его из командной строки.

mysqldump --opt -u имя_пользователя -p пароль
test>backup.sql

Для воссоздания базы необходимо создать базу с подходящим именем на нужной машине, загрузить файл копии с помощью команды.

mysql -u имя_пользователя -p<backup.sql

2. С помощью сценария mysqlhotcopy. Отличается от предыдущего тем, что копирует непосредственно файлы данных базы, а не данные, необходимые для восстановления.

mysqlhotcopy –u имя_пользователя –p имя_БД размещение_копии

Этот сценарий использует Perl. Поэтому если используется Windows, возможно потребуется установить Perl.

3. Резервирование и восстановление вручную. Это предполагает обновление данных (запись на диск содержимого файловых буферов) и блокировку таблиц, а затем копирование файлов. Сначала нужно открыть сеанс MySQL. Затем блокировать все таблицы, которые планируется резервировать.

lock tables имя_таблицы read, имя_таблицы read;

В операторе *locktables* в качестве параметров используется список имен таблиц и тип блокировки. Для резервирования блокировки для чтения обычно бывает достаточно. Это значит, что другие потоки могут продолжать читать данные из таблиц, но не смогут записывать в них данные, пока резервирование не будет завершено.

Затем следует применить команду FLUSHTABLES. Если необходимо резервировать все базы данных, можно совместить эти два шага.

flush tables with read lock;

Теперь можно копировать файлы данных.Очень важно, чтобы сеанс связи оставался открытым, пока выполняется копирование. Это обеспечит сохранение блокировок.

После завершения копирования файлов, следует разблокировать таблицы.

unlock tables;

4. Резервирование и восстановление с помощью BACKUPTABLE и RESTORETABLE. Эти команды работают только с таблицами MyISAM.

backup table имя_таблицы to 'путь';

Восстановление.

restore table имя_таблицы from 'путь';

Порядок выполнения работы:

1. Выполнить резервное копирование базы данных своего варианта спомощью сценария *mysqldump*.
2. Выполнить резервное копирование базы данных своего варианта с помощью BACKUPTABLE.
3. Выполнить сравнительный анализ скриптов базы данных.
4. Восстановить базу данных.
5. Выполнить тестирование восстановленной базы данных.

Форма представления результата:

Отчет по выполненной практической 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 3.2. Защита баз данных

Практическая работа № 22
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 практических работ, вариант задания, компьютер, программное обеспечениеMySQL, MSSQLServer.

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 практической 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 3.2. Защита баз данных

Практическая работа № 23
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 практических работ, вариант задания, компьютер, программное обеспечениеMSSQLServer, MySQL.

Задание 1:
Реализовать доступ пользователей к базе данныхв СУБД MSSQLServer.

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, указавдлянегопарольипринятое 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).

Созданиеновой учетной записи может быть произведено с помощью системной хранимой процедуры:

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

После завершенияаутентификации 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'

Задание 2:
Реализовать доступ пользователей к базе данных в СУБД MySQL.

Краткие теоретические сведения:

База данных mysql, которая присутствует на любомMySQL-сервере, используется для хранения информации о пользователях, ихпаролях и полномочиях. В главе 4 вы рассмотрели, как с помощью phpMy Admincоздать еще одну пользовательскую учетную запись, которая открывает доступтолько к базе данных сайта.

Система управления доступом к MySQL определяются содержимымпяти таблиц, расположенных в базе данных mysql: user, db, host, tables _ privi columns_priv. Если вы пожелаете отредактировать перечисленные таблицывручную с помощью команд **INSERT, UPDATE** и **DELETE,** ознакомьтесь с соответствующим разделом руководства MySQL. В остальном вам вполне хватит средствдля управления доступом к MySQL-серверу, которые предоставляет phpMy Admin.

Система управления доступом к MySQL имеет несколько особенностей, о которых вы обязаны знать, если планируете ответственно относиться к назначению полномочий пользователям на сервере баз данных.

Добавляя в phpMy Admin новых пользователей, которые проходят аутентификацию на MySQL-сервере, только на том компьютере, на котором этот сервер работает(чтобы, например, разрешить им запускать внутри системы утилиту командной строкиmysql или выполнять подключение с помощью таких серверных скриптов, как PHP),вы можете задаться вопросом: что именно вводить в поле Хост? Представьте, что серверработает на домене www.example.com. Что нужно указать: www.example.com или localhost?

На самом деле, когда речь идет о создании каких бы то ни было соединений,ни один из этих вариантов не надежен. Теоретически, если пользователь при подключении указал имя сервера (с помощью консольной утилиты mysql либо в PHPскрипте в классе PDO), это имя должно совпасть с записью в системе управлениядоступом. Однако вряд ли вам захочется, чтобы пользователи указывали имя сервера каким-то определенным образом (и на самом деле вряд ли у них есть желаниевникать в такие тонкости), поэтому лучше пойти другим путем.

Для тех пользователей, кому нужно подключаться к MySQL-серверу через компьютер, на котором он работает, в системе управления доступом лучше создать двезаписи: одну непосредственно с именем сервера, например www.example.com, другуюсо значением localhost. Конечно, вам придется назначать и снимать привилегии длякаждой записи отдельно, но это действительно единственный надежный способ.

Еще одна проблема, с которой часто сталкиваются администраторы MySQL,заключается в том, что записи пользователей, где в именах серверов содержатся заполнители (например, %.example.com), иногда не работают. Причины непредсказуемого поведения системы управления доступом чаще всего связаны с тем,как MySQL-сервер выставляет приоритеты для учетных записей. В частности,он сортирует пользователей таким образом, что более конкретные имена серверовидут в начале списка (к примеру, имя www.example.com абсолютно определенное,%.example.com менее конкретное, а % совершенно неопределенное)

По умолчанию после установки система управления доступом к MySQL содержит две анонимные учетные записи (они позволяют подключиться с локальногокомпьютера, используя любой логин, а также имя сервера или значение localhost, какописано ранее) и две администраторские. Вышеописанная проблема возникает,когда анонимные записи имеют более конкретное имя сервера и получают повышенный приоритет по сравнению с новым пользователем.

Рассмотрим фрагмент таблицы, куда включены пользователи доменаwww.example.com — вымышленного MySQL-сервера, для которого добавлена новаяучетная запись пользователя с именем jess . Строки показаны в том порядке,в каком их рассматривает сервер при проверке подключения.

Имя сервера	Пользователь	Пароль

localhost	root	зашифрованное значение
www.example.com	root	зашифрованное значение
localhost		
www.example.com		
%.example.com	jess	зашифрованное значение

Как видите, запись jess содержит наименее конкретное имя сервера, поэтомуона находится в конце списка. Когда пользователь с данным логином попытаетсяподключиться с адреса www.example.com, MySQL-сервер сопоставит его с одной изанонимных учетных записей (пустой логин соответствует любому пользователю).Пользователь jess, скорее всего, введет свой пароль (для анонимных записей этоделать необязательно), и в результате MySQL-сервер отклонит попытку подключения. Но даже если пользователь jess подключится без пароля, он получит оченьограниченные привилегии, свойственные анонимным учетным записям, которыевряд ли соответствуют правам, назначенным системой управления доступом дляего логина.

Эту проблему можно решить двумя способами: либо удалить анонимные учетные записи с самого начала (DELETE FROM mysql.user WHERE User= " "), либовыдать по две дополнительные записи всем пользователям, которым для работынеобходимо подключаться на локальном компьютере (по одной для localhost и настоящего имени сервера).

Имя сервера	Пользователь	Пароль
localhost	root	зашифрованное значение
www.example.com	root	зашифрованное значение
localhost	jess	зашифрованное значение
www.example.com	jess	зашифрованное значение
localhost		
www.example.com		
%.example.com	jess	зашифрованное значение

Поскольку для каждого логина поддерживать три учетные записи (с тремя наборами привилегий) довольно обременительно, лучше удалить анонимных пользователей, особенно если в них нет необходимости.

Имя сервера	Пользователь	Пароль
localhost	root	зашифрованное значение
www.example.com	root	зашифрованное значение
%.example.com	jess	зашифрованное значение

Забыть пароль от MySQL-сервера после того, как потрачен целый час на егоустановку и настройку, — это почти то же самое, что закрыть автомобиль вместес ключами внутри. К счастью, если у вас есть административный доступ к компьютеру, на котором работает MySQL, или возможность зайти в систему как пользователь, обладающий полномочиями для управления MySQL-сервером, то все не такуж плохо. Ознакомившись с этим разделом, вы узнаете, как восстановить контрольнад сервером.

Первым делом следует остановить MySQL-сервер. В обычных условиях выделали это с помощью консольной утилиты mysqladmin, которая требовала пароль. Теперь вам придется «убить» все процессы на сервере. Используйте Диспетчер задач: завершите процесс MySQL или остановитеMySQL-сервис. Еще один способ — заглянуть в PID -файл, который находитсяв директории с данными вашего MySQL-сервера, и завершить процесс с помощьюследующей команды.

kill pid

pid —идентификатор процесса MySQL-сервера.

Чтобы остановить сервер, этого достаточно. Не используйте команду принудительного завершения kill -9, если в этом нет крайней необходимости: онаспособна повредить файлы с

данными. На тот случай, если ситуация вынудила васпоступить именно так, в следующем разделе вы найдете инструкции по проверкеи восстановлению файлов.

Теперь, когда сервер остановлен, запустите его снова, используя параметрskip-grant-tables. Вы также можете добавить его в конфигурационный файлMySQL-сервера —my.ini или my.cnf.

[mysqld]

skip-grant-tables

Так сервер MySQL узнает о том, что необходимо открыть свободный доступдля любого пользователя. Поскольку такой режим работы несет потенциальнуюугрозу с точки зрения безопасности, сервер должен находиться в нем как можноменьше.

Подключившись к серверу (с помощью phpMyAdmin или консольной утилитыmysql), укажите новый пароль администратора.

UPDATE mysql.user SET Password=PASSWORD("newpassword")

WHERE User="root"

СноваостановитеMySQL-сервериудалитепараметрskip-granttables.

Форма представления результата:
Отчет по выполненной практической 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 3.2. Защита баз данных

Практическая работа № 24
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 практических работ, вариант задания, компьютер, программное обеспечение: MySQL, MSSQLServer.

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 практической 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 3.2. Защита баз данных

Практическая работа № 25
Перехват исключительных ситуаций

Цель: получение практических навыков по освоению операций перехвата исключительных ситуаций

Выполнив работу, Вы будете:

уметь:

- выполнять перехват исключительных ситуаций.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение:MySQL, MSVisualStudio, MSSQLServer.

Задание1:

Используя обработку исключительных ситуаций, выполните авторизацию пользователей.

Краткие теоретические сведения:

Принимая во внимание, что .NET Framework включает большое количество предопределенных классов исключений, возникает вопрос: как их использовать в коде для перехвата ошибочных условий? Для того чтобы справиться с возможными ошибочными ситуациями в коде C#, программа обычно делится на блоки трех разных типов:

- Блоки try инкапсулируют код, формирующий часть нормальных действий программы, которые потенциально могут столкнуться с серьезными ошибочными ситуациями.
- Блоки catch инкапсулируют код, который обрабатывает ошибочные ситуации, происходящие в коде блока try. Это также удобное место для протоколирования ошибок.
- Блоки finally инкапсулируют код, очищающий любые ресурсы или выполняющий другие действия, которые обычно нужно выполнить в конце блоков try или catch. Важно понимать, что этот блок выполняется независимо от того, сгенерировано исключение или нет.

Try и catch

Основу обработки исключительных ситуаций в C# составляет пара ключевых слов try и catch. Эти ключевые слова действуют совместно и не могут быть использованы порознь. Ниже приведена общая форма определения блоков try/catch для обработки исключительных ситуаций:

```
try {
// Блок кода, проверяемый на наличие ошибок.
}

catch (ExcepType1 exOb) {
// Обработчик исключения типа ExcepType1.
}
catch (ExcepType2 exOb) {
// Обработчик исключения типа ExcepType2.
}
...
```

где ExcepType — это тип возникающей исключительной ситуации. Когда исключение генерируется оператором try, оно перехватывается составляющим ему пару оператором catch, который затем обрабатывает это исключение. В зависимости от типа исключения выполняется и соответствующий оператор catch. Так, если типы генерируемого исключения и того, что указывается в операторе catch, совпадают, то выполняется именно этот оператор, а все остальные пропускаются. Когда исключение перехватывается, переменная исключения exOb получает свое значение. На самом деле указывать переменную exOb необязательно. Так, ее необязательно

указывать, если обработчику исключений не требуется доступ к объекту исключения, что бывает довольно часто. Для обработки исключения достаточно и его типа.

Следует, однако, иметь в виду, что если исключение не генерируется, то блок оператора try завершается как обычно, и все его операторы catch пропускаются. Выполнение программы возобновляется с первого оператора, следующего после завершающего оператора catch. Таким образом, оператор catch выполняется лишь в том случае, если генерируется исключение.

Порядок выполнения работы:

Рассмотрим пример, в котором будем обрабатывать исключение, возникающее при делении числа на 0:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static int MyDel(int x, int y)
        {
            return x / y;
        }
        static void Main()
        {
            try
            {
                Console.Write("Введите x: ");
                int x = int.Parse(Console.ReadLine());
                Console.Write("Введите y: ");
                int y = int.Parse(Console.ReadLine());

                int result = MyDel(x, y);
                Console.WriteLine("Результат: " + result);
            }
            // Обрабатываем исключение возникающее при делении на ноль
            catch (DivideByZeroException)
            {
                Console.WriteLine("Деление на 0 detected!!!\n");
                Main();
            }
            // Обрабатываем исключение при неккоректном вводе числа в консоль
            catch (FormatException)
            {
                Console.WriteLine("Это НЕ число!!!\n");
                Main();
            }

            Console.ReadLine();
        }
    }
}
```

2 _____.

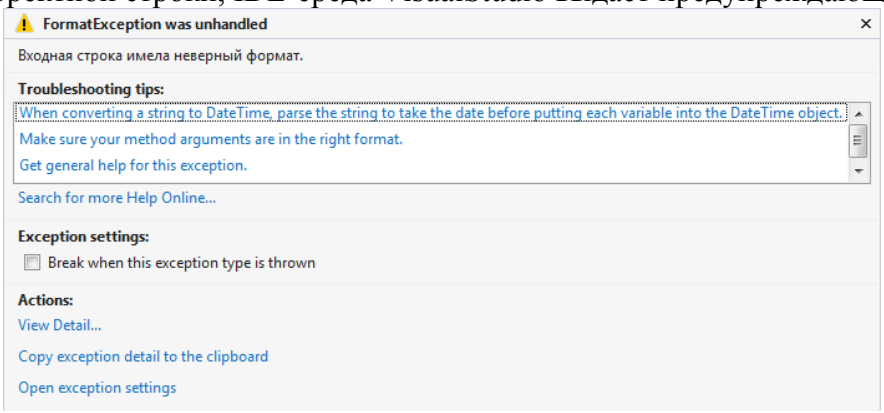
```
file:///C:/projects/test/ConsoleApplication1/ConsoleApplication1/bin/Debug/ConsoleApplication1....
Введите x: professorWeb :>
Это НЕ число!!!

Введите x: 10
Введите y: 0
Деление на 0 detected!!!

Введите x: 10
Введите y: 2
Результат: 5
```

Данный пример наглядно иллюстрирует обработку исключительной ситуации при делении на 0 (DivideByZeroException), а также пользовательскую ошибку при вводе не числа (FormatException).

Последствия неперехвата исключений

Перехват одного из стандартных исключений, как в приведенном выше примере, дает еще одно преимущество: он исключает аварийное завершение программы. Как только исключение будет сгенерировано, оно должно быть перехвачено каким-то фрагментом кода в определенном месте программы. Вообще говоря, если исключение не перехватывается в программе, то оно будет перехвачено исполняющей системой. Но дело в том, что исполняющая система выдаст сообщение об ошибке и прервет выполнение программы.

Например, если убрать из предыдущего примера исключение FormatException, то при вводе неккоректной строки, IDE-среда VisualStudio выдаст предупреждающее сообщение:



Задание2:

Используя обработку исключительных ситуаций, выполните подключение к MySQL.

Краткие теоретические сведения:

Поддержка подключений кMySQL обеспечиваетсявстроенным расширением PHPDateObject (PDO). Функция `newPDO` возвращает объект PDO, который определяет установленное соединение. Поскольку мы планируем использовать соединение в дальнейшем, этот объект следует поместить в переменную.

Может оказаться так, что сервер баз данных будет недоступен, например, из-за поломки сети, неверно предоставленной пары логина и пароля. В этом случае выражение newPDOне сработает и сгенерирует исключение.

Исключение возникает в том случае, когда вы заставляете PHP выполнить задачу, которую выполнить нельзя. Конечно, он попытается сделать то, что ему велено, но у него ничего не получится. Чтобы сообщить о своей неудаче, PHPсгенерирует для вас исключение. Если вы не перехватите исключение, то PHP прекратит выполнение скрипта и выведет внушительное сообщение об ошибке, в котором, кроме всего прочего, разместит и код этого самого скрипта. Как вы помните, код содержит логин и пароль для MySQL, поэтому очень важно, чтобы сообщение об ошибке не попалось на глаза пользователя.

Чтобы перехватить исключение, поместите код, который может привести к его выбросу, внутрь выражения `try – catch`.

В верхнем блоке `try` происходит подключение к базе данных с помощью команды `newPDO`. В случае успеха полученный объект PDO будет помещен в переменную `$pdo` – это позволит работать с новым подключением. Если попытка закончилась неудачей, PHP сгенерирует объект `PDOException` - разновидность исключений, которую выбрасывает newPDO. Блок `catch` перехватит объект `newPDO`и поместит его в переменную `$e`. Внутри блока переменной `$error` будет присвоено сообщение, информирующее пользователя о том, что произошло. В следующей строке кода подключается шаблон `error.html.php`– стандартная страница, на которой отображается значение ошибки.

После того как сообщение появится на экране, последнее выражение в блоке `catch` вызовет встроенную функцию `exit`. Это первый пример функции, которую можно запустить без параметров. Такой вызов `exit` приведет к тому, что PHP-скрипт тут же закончит свою работу, и в

случае неудачного подключения оставшаяся часть кода в контроллере (которая, скорее всего, зависит от успешного соединения с базой данных) выполнена не будет. Надеемся, что теперь приведенный выше код стал более понятен.

Порядок выполнения работы:

Чтобы перехватить исключение, поместите код, который может привести к его выбросу, внутрь выражения `try – catch`.

```php
<?php
try
{
    $pdo=new PDO('mysql:host=localhost;dbname=int_joke', 'jokesuser', '123');
}
catch(PDOException $e)
{
    $output='Невозможно подключиться к серверу баз данных:'.
    include 'output.html.php';
    exit();
}
?>
```

В верхнем блоке `try` происходит подключение к базе данных с помощью команды `newPDO`. В случае успеха полученный объект PDO будет помещен в переменную `$pdo` – это позволит работать с новым подключением. Если попытка закончилась неудачей, PHP сгенерирует объект `PDOException` - разновидность исключений, которую выбрасывает newPDO. Блок catch перехватит объект `newPDO`и поместит его в переменную `$e`. Внутри блока переменной `$output` будет присвоено сообщение, информирующее пользователя о том, что произошло. В следующей строке кода подключается шаблон `output.html.php`– стандартная страница, на которой отображается значение `$output`

```php
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Вывод скрипта</title>
</head>
<body>
<p>
<?php echo $output; ?>
</p>
</body>
</html>
```

После того как сообщение появится на экране, последнее выражение в блоке `catch` вызовет встроенную функцию `exit`. Это первый пример функции, которую можно запустить без параметров. Такой вызов `exit` приведет к тому, что PHP-скрипт тут же закончит свою работу, и в случае неудачного подключения оставшаяся часть кода в контроллере (которая, скорее всего, зависит от успешного соединения с базой данных) выполнена не будет

Форма представления результата:

Отчет по выполненной практической 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 3.2. Защита баз данных

Практическая работа № 26
Кэширование данных

Цель: получение практических навыков по освоению операций кэширования данных

Выполнив работу, Вы будете:
уметь:
- выполнять кэширование данных.

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

Задание:
Выполнить кэширование данных.

Краткие теоретические сведения:
Кэширование— это один из способов оптимизации Web приложений. В любом приложении встречаются медленные операции (SQL запросы или запросы к внешним API), результаты которых можно сохранить на некоторое время. Это позволит выполнять меньше таких операций, а большинству пользователей показывать заранее сохраненные данные.

Наиболее популярная технология кеширования для Web приложений — Memcache.

Старайтесь избегать кэширования, пока в этом не будет прямой необходимости. Это простая техника, но это снижает гибкость приложения. Не делайте лишнюю работу заранее, но закладывайте возможность использования кэширования в будущем:

- Используйте классы или функции, для работы с данными. Не используйте повторяющихся SQL выборок в основном приложении.
- Используйте обертки для работы с внешними API.

Кэшировать нужно данные, которые медленно генерируются и часто запрашиваются. На практике это обычно:

- Результаты запросов к внешним сервисам (RSS, SOAP, REST и т.п.).
- Результаты медленных выборок из базы данных.
- Сгенерированные html блоки либо целые страницы.

Кэширование выборок из баз данных

Запросы к базе данных — наиболее распространенный пример. На основе Memcache реализуется очень просто:

```php
<?
memcache_connect('localhost', 11211);

function get_online_users()
{
    if ( !$list = memcache_get('online_users') )
    {
        $sql = 'SELECT * FROM users WHERE last_visit > UNIX_TIMESTAMP() - 60*10';
        $q = mysql_query($sql);
        while ($row = mysql_fetch_assoc($q)) $list[] = $row;
        memcache_set('online_users', $list, 60*60);
    }
    return $list;
}
$list = get_online_users();
...
```

Обновление данных

Если кэшируются данные, которые могут обновляться, необходимо очищать кэш после каждого обновления:

```php
<?
memcache_connect('localhost', 11211);
function get_user($id)
{
    if ( !$data = memcache_get('user' . $id) )
    {
        $sql = 'SELECT * FROM users WHERE id= ' . intval($id);
        $q = mysql_query($sql);
        $data = mysql_fetch_assoc($q);
        memcache_set('user' . $id, $data, 60*60);
    }
    return $data;
}
function save_user($id, $data)
{
    mysql_query('UPDATE users SET ... WHERE id = ' . intval($id));
    memcache_delete('user' . $id);
}
```

Кэширование списков

Допустим, вы кэшируете данные каждого пользователя, а также их списки (например, список online пользователей). При обновлении данных пользователя, вы удаляете данные из кэша только для указанного пользователя. Но его данные могут также присутствовать в списке online пользователей, которые тоже лежат в кэше. Сбрасывать списки при каждом обновлении данных любого пользователя не эффективно. Поэтому обычно используют такой подход:

1. Кэшируют списки, которые состоят только из ID пользователей.
2. Для вывода списка отправляют отдельный запрос для получения данных каждого пользователя.

Реализация выглядит так:

```php
<?
memcache_connect('localhost', 11211);
function get_online_users()
{
    if ( !$list = memcache_get('online_users') )
    {
        $sql = 'SELECT id FROM users WHERE last_visit > UNIX_TIMESTAMP() - 60*10';
        $q = mysql_query($sql);
        while ($row = mysql_fetch_assoc($q)) $list[] = $row['id'];
        memcache_set('online_users', $list, 60*60);
    }

    return $list;
}
$list = get_online_users();
foreach ( $list as $id )
{
    $user = get_user($id);
    ...
}
```

Повторные запросы

Некоторые данные могут запрашиваться несколько раз в рамках одной страницы, например:

```
<html>
<body>
<h1><?=htmlspecialchars( get_user( $_SESSION['id'] )['name'] )?></h1>
...
Email: <?=get_user( $_SESSION['id'] )['email']?>
...
<a href="/<?=get_user( $_SESSION['id'] )['nick']?>">Моя страница</a>
...
```

Каждый вызов**get_user()**будет получать данные из кэша. Если Memcache стоит на отдельном сервере, это вызовет большой сетевой трафик и задержки.

Чтобы этого избежать, можно использовать дополнительный кэш внутри самого приложения:

```
<?
memcache_connect('localhost', 11211);
function get_user($id)
{
    global $app_cache;
    if ( $app_cache['user' . $id] ) return $app_cache['user' . $id];

    if ( !$data = memcache_get('user' . $id) )
    {
        $sql = 'SELECT * FROM users WHERE id= ' . intval($id);
        $q = mysql_query($sql);
        $data = mysql_fetch_assoc($q);
        memcache_set('user' . $id, $data, 60*60);
        $app_cache['user' . $id] = $data;
    }
    return $data;
}
function save_user($id, $data)
{
    global $app_cache;

    mysql_query('UPDATE users SET ... WHERE id = ' . intval($id));
    memcache_delete('user' . $id);

    unset($app_cache['user' . $id]);
}
```

В реальных приложениях, имеет смысл иметь обертку для Memcache с дополнительным кэшом:

```
<?
class mem_cache
{
    private $inner_cache = [];

    public static function get( $key )
    {
        if ( array_key_exists($key, $this->inner_cache) ) return $this->inner_cache[$key];

        $data = memcache_get( $this->resource, $key );
        $this->inner_cache[$key] = $data;

        return $data['value'];
    }

    public static function set( $key, $value, $ttl )
    {
        memcache_set($key, $value, $ttl);
        $this->inner_cache[$key] = $value;
    }

    public static function del( $key )
    {
        memcache_delete($key);
        unset($this->inner_cache[$key]);
    }
}
```

Использование этого подхода может приводить к утечкам памяти в случаях, когда идет работа с большим количеством данных в кэше. Например, в cron-задачах (допустим, мы перебираем всех пользователей для отправки рассылки). Тогда лучше добавить отключение внутреннего кэша:

```php
<?
class mem_cache
{
    private $inner_cache = [];
    public static $inner_cache_enabled = true;

    public static function get( $key )
    {
        if ( self::$inner_cache_enabled && array_key_exists($key, $this->inner_cache) ) return $this->inner_cache[$key];

        $data = memcache_get( $this->resource, $key );
        $this->inner_cache[$key] = $data;

        return $data['value'];
    }

    public static function set( $key, $value, $ttl )
    {
        memcache_set($key, $value, $ttl);
        if ( self::$inner_cache_enabled ) $this->inner_cache[$key] = $value;
    }

    public static function del( $key )
    {
        memcache_delete($key);
        unset($this->inner_cache[$key]);
    }
}
...
mem_cache::$inner_cache_enabled = false;
```

Порядок выполнения работы:

1. Выполнить кэшировавние выборок из баз данных.
2. Выполнить обновление данных.
3. Выполнитькэшировавние списков.
4. Выполнить кэширование повторныз запросов.

Форма представления результата:

Отчет по выполненной практической 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.