Министерство науки и высшего образования Российской Федерации

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

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ И ЛАБОРАТОРНЫХ РАБОТ
по учебной дисциплине
ОПЦ.08. ОСНОВЫ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ

для студентов специальности

09.02.07 Информационные системы и программирование
Квалификация: программист

Магнитогорск, 2020

ОДОБРЕНО:
Предметно-цикловой комиссией
Информатики и вычислительной техники
Председатель И.Г. Зорина
Протокол № 7 от 17.02.2020

Методической комиссией МпК
Протокол №3 от «26» февраля 2020г

Составители:
преподаватель ФГБОУ ВО «МГТУ им. Г.И. Носова» МпК Н.В. Кучерова
преподаватель ФГБОУ ВО «МГТУ им. Г.И. Носова» МпК И.Г. Зорина

Методические указания по выполнению практических и лабораторных работ разработаны на основе рабочей программы учебной дисциплины «Основы проектирования баз данных».

Содержание практических и лабораторных работ ориентировано на подготовку обучающихся к освоению профессиональных модулей программы подготовки специалистов среднего звена по специальности 09.02.07 Информационные системы и программирование и овладению общими и профессиональными компетенциями.

СОДЕРЖАНИЕ

1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

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

Ведущей дидактической целью практических занятий является формирование учебных практических умений (использовать современные case – средства для проектирования баз данных; создавать объекты баз данных в современных СУБД; применять стандартные методы для защиты объектов базы данных), необходимых в последующей учебной деятельности по профессионального модуля ПМ.11 Разработка, администрирование и защита баз данных.

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

В соответствии с рабочей программой учебной дисциплины «Основы проектирования баз данных» предусмотрено проведение практических и лабораторных занятий. В рамках практического занятия обучающиеся могут выполнять одну или несколько практических работ.

В результате их выполнения, у обучающихся должны сформироваться предметные результаты:

уметь:

У1. Проектировать реляционную базу данных

У2. Использовать язык запросов для программного извлечения сведений из баз данных.

У 01.4 Выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы.

У 02.1 Определять задачи для поиска информации.

У 02.2 Определять необходимые источники информации.

У04.2 Взаимодействовать с коллегами, руководством, клиентами в ходе профессиональной деятельности.

У05.3 Излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке.

У09.1 Применять средства информационных технологий для решения профессиональных задач.

У09.2 Использовать современное программное обеспечение.

У 10.1 Понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые)

Содержание практических и лабораторных занятий ориентировано на подготовку студентов обучающихся к освоению профессионального модуля программы подготовки специалистов среднего звена по специальности и овладению профессиональными компетенциями:

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

ПК 11.2 Проектировать базу данных на основе анализа предметной области.

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

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

ПК 11.5 Администрировать базы данных.

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

А также формированию общих компетенций:

ОК 1 Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

ОК 2 Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.

ОК 4 Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.

ОК 5 Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.

ОК9 Использовать информационные технологии в профессиональной деятельности

ОК10 Пользоваться профессиональной документацией на государственном и иностранном языках.

Выполнение практических и лабораторных работ по учебной дисциплине «Основы проектирования баз данных» направлено на:

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

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

2 ПЕРЕЧЕНЬ ПРАКТИЧЕСКИХ РАБОТ

Разделы/темы	Темы практических/лабораторных занятий	Количество часов	Требования ФГОС СПО (уметь)
Тема 2. Взаимосвязи в моделях и реляционный подход к построению моделей	Лабораторное занятие № 1. Задание ключей. Создание основных объектов базы данных	6	У01.4, У02.1, У02.2, У09.1, У09.2
Тема 3. Этапы проектирования баз данных	Практическая работа №1. Проектирование реляционной базы данных.	2	У1, У09.1,У09.2
	Практическая работа № 2. Нормализация реляционной базы данных.	2	У1,У01.4 У02.1, У02.2, У04.2, У05.3, У09.1,У09.2
	Лабораторная работа №2. Преобразование реляционной базы данных в сущности и связи.	8	У1, У01.4, У02.1, У02.2, У04.2, У05.3, У09.1,У09.2
Тема 4. Проектирование структур баз данных	Лабораторная работа № 3 Редактирование, добавление и удаление записей в таблице. Применение логических условий к записям. Открытие, редактирование и пополнение табличного файла	2	У1, У04.2, У05.3,У09.1, У09.2, У10.1
	Лабораторная работа № 4 Создание ключевых полей. Задание индексов. Установление и удаление связей между таблицами.	2	У1, У09.1, У09.2,У10.1
	Лабораторная работа № 5 Проведение сортировки и фильтрации данных. Поиск данных по одному и нескольким полям. Поиск данных в таблице.	4	У1, У01.4, У02.1, У02.2, У04.2, У05.3, У09.1,У09.2, У10.1
	Лабораторная работа № 6 Создание меню различных видов. Модификация и управление меню	4	У1, У01.4, У02.1, У02.2, У04.2, У05.3, У09.1,У09.2, У10.1
	Лабораторная работа № 7 Создание файла проекта базы данных. Создание интерфейса входной формы. Использование исполняемого файла проекта базы данных, приемы создания и управления.	6	У1, У01.4, У02.1, У02.2, У04.2, У05.3, У09.1,У09.2, У10.1
	Лабораторная работа № 8 Создание формы. Управление внешним видом формы	4	У09.1,У09.2,
Тема 5. Организация запросов SQL	Лабораторная работа № 9 Создание проекта базы данных. Создание базы данных. Редактирование и модификация таблиц.	6	У1, У02.1, У02.2, У09.1, У09.2
	Лабораторная работа № 10 Заполнение	6	У02.1, У02.2,

	массива из табличного файла. Заполнение табличного файла из массива.		У09.1, У09.2
	Лабораторная работа № 11 Добавление записей в табличный файл из двумерного массива. Работа с командами ввода-вывода. Использование функций для работы с массивами.	4	У02.1, У02.2, У09.1, У09.2
	Лабораторная работа № 12 Задание значений и ограничений поля. Проверка введенного в поле значения. Отображение данных числового типа и типа дата.	4	У09.1, У09.2
	Лабораторная работа № 13 Создание и модификация таблиц БД. Выборка данных из базы данных. Модификация содержимого базы данных.	6	У2, У09.1, У09.2
	Лабораторная работа № 14 Обработка транзакций. Использование функций защиты для базы данных.	4	У09.1, У09.2
ИТОГО		**70**	

3 МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Тема 2. Взаимосвязи в моделях и реляционный подход к построению моделей

Лабораторная работа №1

Задание ключей. Создание основных объектов базы данных.

Цели:

1. Освоить технологию создания таблиц в СУБД Access и связей между ними.
2. Определять типы данных в полях таблиц

Выполнив работу, Вы будете:

уметь:

У 01.4 Выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы

У 02.1Определять задачи для поиска информации

У 02.2 Определять необходимые источники информации

У09.1 Применять средства информационных технологий для решения профессиональных задач

У09.2 Использовать современное программное обеспечение

Материальное обеспечение: персональный компьютер, MS Access, методические указания по выполнению практических занятий

Задание 1. Создать базу данных СТУДЕНТ и создать подчиненную форму для ее заполнения

Порядок выполнения задания 1:

1. Открыть Access.
2. Выполнить создание Новой базы данных, определить папку группы для размещения базы, определить имя базы данных СТУДЕНТЫ.
3. В режиме Конструктор определить следующие поля таблицы СТУДЕНТЫ:

Поле	Тип данных
№_студ_билета	Счетчик, определить как ключевое
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
Пол	Мастер подстановок Фиксированный набор значений: мужской,
Дата_рождения	Дата/время
Адрес	Текстовый
Отделение	Мастер подстановок Фиксированный набор значений:
Курс	Числовой
Группа	Текстовый

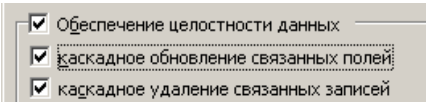
Сохранить структуру таблицы, указать ключевое поле **№студ_билета**

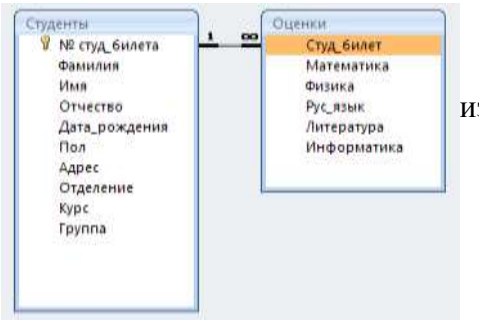
4. Создать новую таблицу ОЦЕНКИ со следующими полями

Поле	Тип данных
Студ_билет	Числовой
Математика	Числовой
Физика	Числовой
Рус_язык	Числовой
Литература	Числовой
Информатика	Числовой

5. Выполнить команду Схема данных на ленте РАБОТА С БАЗАМИ ДАННЫХ,

добавить таблицы СТУДЕНТЫ и ОЦЕНКИ. Для создания связи перетащить название поле **№студ_билета** из таблицы **СТУДЕНТЫ** на поле **Студ_билет** таблицы **ОЦЕНКИ**. В диалоговом окне связи установить флажки



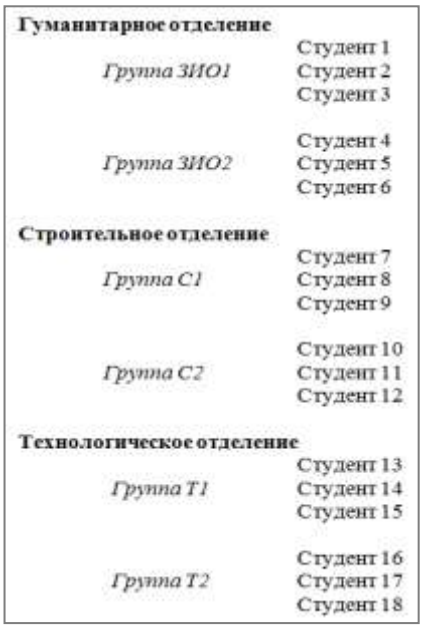


и щелкнуть кнопку Создать. Между таблицами появиться изображение связи. Закрыть окно Схемы данных, сохранив изменения.

6. Открыть таблицу СТУДЕНТЫ, ввести данные для одного студента. После перехода на новую запись таблицы для введенной записи появится значок, щелкнув который можно ввести данные об оценках этого студента. Остальные данные в режиме Таблица НЕ ВВОДИТЬ.
7. Закрыть все объекты базы данных СТУДЕНТЫ.
8. Перейти на ленту Создание, в списке Другие формы выбрать Мастер форм и пошагово выполнить создание формы:
   - Включить все поля из таблицы СТУДЕНТЫ, и все поля, кроме Студ_билет, из таблицы ОЦЕНКИ
   - Выбрать вид формы: подчиненные формы
   - Вид формы: табличный
   - Любой стиль
9. Открыть форму. Перейти в режим Макета (кнопка ) и увеличить размер таблицы, в которую будут вводиться оценки, подобрать ширину столбцов. Отформатировать элементы формы по своему усмотрению.
10. Вернуться в режим формы (кнопка ) и ввести записи о студентах разных групп, отделений.
11. Закрыть форму. Проверить введенные данные, открыв таблицу СТУДЕНТЫ.
12. Сформировать отчет по таблице СТУДЕНТЫ, назначив два уровня группировки:

1 уровень: по отделению

2 уровень по группе (см. рисунок). Сравнить с образцом



Задание 2. Сформировать запросы в базе данных СТУДЕНТЫ

Порядок выполнения задания 2:

1. Сформировать простые запросы:
   - запрос *Данные о студентах* на основе таблицы Студенты (с полями Фамилия, имя, Отчество, дата рождения, Отделение, курс, группа)
   - запрос под именем *Все оценки* (использовать поля из двух таблиц) с полями: Отделение, курс, группа, Фамилия, Имя, Математика, Физика, Русский язык, Литература, Информатика)
   - *Оценки по информатике* (поля: отделение, группа, фамилия, информатика)
2. Сформировать запросы на выборку:
   - запрос *Студенты Гуманитарного отделения* (отобразить Фамилия, Имя отчество, Дата рождения, группа)
   - *Студенты 1997 года рождения:* отобразить Фамилия, Имя отчество, Дата рождения (в условие отбора ввести шаблон ***.*. 1997**), отделение, группа
   - *Список неуспевающих студентов по Математике:* отобразить Фамилия, Имя отчество, отделение, группа, математика (условие отбора **2**)*Студенты строительного отделения, у которых по физике 5:* отобразить Фамилия, Имя отчество, отделение (условие отбора Строительное), группа, Физика (условие отбора **5**)
   - *Студенты-отличники*: отобразить Фамилия, Имя отчество, отделение, группа, математика, физика, Рус.Язык, Литература, Информатика (условие отбора для всех предметов **5**)
   - *Студенты технологического отделения, которые имеют двойку хотя бы по одному предмету:* отобразить Фамилия, Имя отчество, отделение, группа, математика, физика, Рус.Язык, Литература, Информатика (условие отбора **2** для оценок по разным дисциплинам вводить в разные строки «лесенкой»)
3. Сформировать запрос с параметром:
   - С параметром по фамилии: включить поля Фамилия (в строку условие отбора ввести LIKE[введите фамилию]), Имя, отделение, группа, оценки по всем предметам. Выполнить запрос, в окне ввести произвольную фамилию и проверить работу запроса
   - *С параметром по отделению* выводятся данные из таблицы Студенты: с полями Отделение (в строку условие отбора ввести LIKE[введите отделение]), группа, Фамилия, Имя, оценки по всем предметам. Выполнить запрос, в окне ввести произвольную фамилию и проверить работу запроса.
   - *С параметром по группе* вывести оценки по информатике и математике, указав фамилию и имя студента
4. Сформировать перекрестные запросы. Для этого перейти на ленту Создание, выбрать команду Мастер запросов, создать перекрестный запрос:
   a) На основе запроса ВСЕ ОЦЕНКИ, Далее
   b) в качестве заголовков строк использовать поле ГРУППА, Далее
   c) в качестве заголовков столбцов использовать поле ОТДЕЛЕНИЕ, Далее
   d) в качестве итоговых значений для каждой строки по полю ИНФОРМАТИКА использовать функцию среднее, Далее
   e) имя запроса *Средний балл по информатике,* Готово
5. Аналогично создать запросы:
   - о среднем балле по математике по группам всех отделенийо количестве студентов по группам на отделениях (в качестве итоговых значений использовать функцию Число для поля Фамилия)

Форма предоставления результата

Отчет по выполненной лабораторной работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

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

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

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 3 Этапы проектирования баз данных

Практическая работа №1

Проектирование реляционной базы данных.

Цель: получение практических навыков проектирования базы данных

Выполнив работу, Вы будете:

уметь:

У1. Проектировать реляционную базу данных

У09.1 Применять средства информационных технологий для решения профессиональных задач

У09.2 Использовать современное программное обеспечение.

Материальное обеспечение:

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

Задание:

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

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.

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

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.

Форма представления результата:

Оформленная схема базы данных в тетради.

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 Этапы проектирования баз данных

Практическая работа №2.

Нормализация реляционной базы данных.

Цель: получение практических навыков по нормализации базы данных

Выполнив работу, Вы будете:

уметь:

У1. Проектировать реляционную базу данных

У 01.4 Выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы

У 02.1 Определять задачи для поиска информации

У 02.2 Определять необходимые источники информации

У04.2 Взаимодействовать с коллегами, руководством, клиентами в ходе профессиональной деятельности

У05.3 Излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке

У09.1 Применять средства информационных технологий для решения профессиональных задач

У09.2 Использовать современное программное обеспечение.

Материальное обеспечение:

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

Задание:

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

Краткие теоретические сведения:

Нормализация баз данных

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

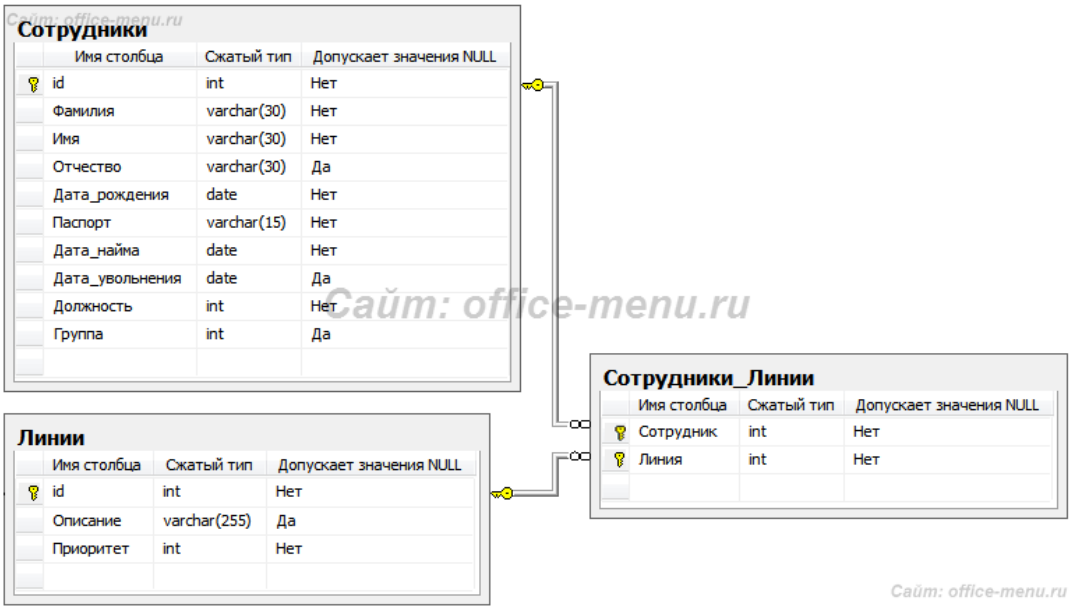
- В нормализованной структуре базы данных можно производить сложные выборки данных относительно простыми SQL-запросами.
- Целостность данных. Нормализованная база данных позволяет надежно хранить данные.
- Нормализация предотвращает появление избыточности хранимых данных. Данные всегда хранятся только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

Нормализация баз данных заключается в приведении структуры хранения данных к нормальным формам (NF). Всего таких форм существует 8, но часто достаточным является соблюдение первых трех. Рассмотрим их более подробно на примере учебной базы данных. Примеры будут строится по принципу «что было бы, если было иначе, чем сейчас».

Первая нормальная форма

Основным правилом первой формы является необходимость неделимости значения в каждом поле (столбце) строки – **атомарность** значений.

Рассмотрим таблицы сотрудников и телефонных линий.



Чтобы избавиться от связывающей таблицы «Сотрудники_Линии», мы могли бы записать идентификаторы сотрудников для каждой линии в виде перечня в дополнительном столбце:

	id	Описание	Приоритет	Сотрудники
1	1	Поддержка держателей пластиковых карт	2	20, 21, 8
2	2	Поддержка потребительского и наличного кредитова...	2	16, 5, 22, 15
3	3	Поддержка автокредитования	2	21, 10
4	4	Поддержка ипотечного кредитования	2	16, 14, 15, 4, 8

Но подобная структура не является надежной. Представьте, что Вам необходимо поменять некоторым сотрудникам подключенные линии. Потребуется осуществить разбор составного поля, чтобы определить наличие id сотрудника в каждой записи линий, затем скорректировать перечень. Получается слишком сложный и долгий процесс для такой простой операции.

Организации структуры таблиц с применением дополнительной связывающей избавляет от подобных проблем.

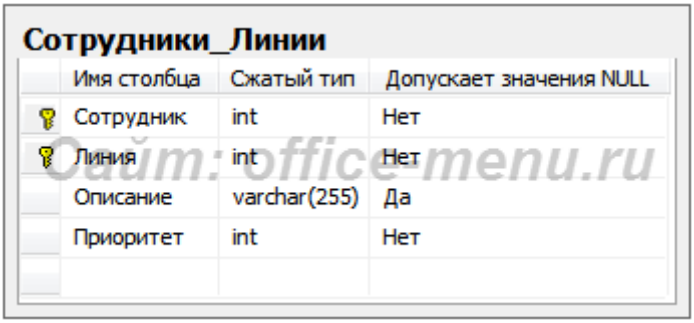
Помимо атомарности к первой нормальной форме относятся следующие правила:

- Строки таблиц не должны зависеть друг от друга, т.е. первая запись не должна влиять на вторую и наоборот, вторая на третью и т.д. Размещение записей в таблице не имеет никакого значения.
- Аналогичная ситуация со столбцами записей. Их порядок не должен влиять на понимание информации.
- Каждая строка должна быть уникальна, поэтому для нее определяется **первичный ключ**, состоящий из одного либо нескольких полей (**составной ключ**). Первичный ключ не может повторяться в пределах таблицы и служит идентификатором записи.

Вторая нормальная форма

Условием этой формы является отсутствие зависимости неключевых полей от части составного ключа.

Так как составной ключ в учебной базе наблюдается только в таблице «Сотрудники_Линии», то рассмотрим пример на ней.



Сотрудники_Линии

Имя столбца	Сжатый тип	Допускает значения NULL
Сотрудник	int	Нет
Линия	int	Нет
Описание	varchar(255)	Да
Приоритет	int	Нет

На представленной диаграмме столбцы описания и приоритета зависят от столбца «Линия», входящего в составной ключ. Это значит, что для каждой линии, подключенной разным сотрудникам, потребуется повторно указывать описание и приоритетность. Подобная структура приводит к **избыточности данных.**

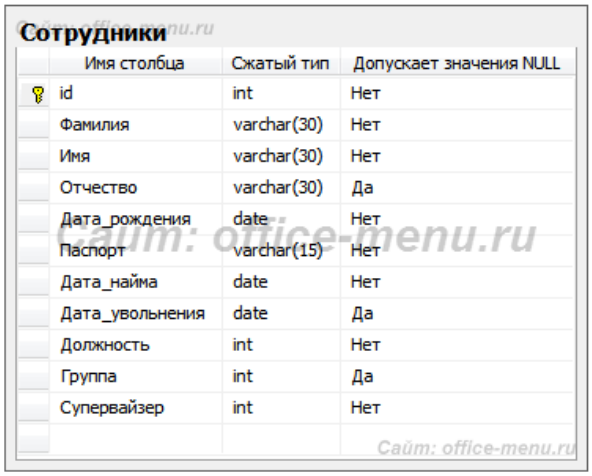
Также велика вероятность возникновения противоречивой информации. Изменяя приоритет или описание для линии, можно по ошибке оставить некоторые строки не обработанными. В таком случае, для одного и того же идентификатора линии значения зависимых полей будут различными.

Если соблюдены правила первой нормальной формы, то создание таблицы «Линии» и перенос в нее зависимых столбцов удовлетворяет второй нормальной форме.

Третья нормальная форма

3NF схожа по логике с 2NF, но с некоторым отличием. Если 2 форма ликвидирует зависимости неключевых полей от части ключа, то третья нормальная форма исключает зависимость неключевых полей от других неключевых полей.

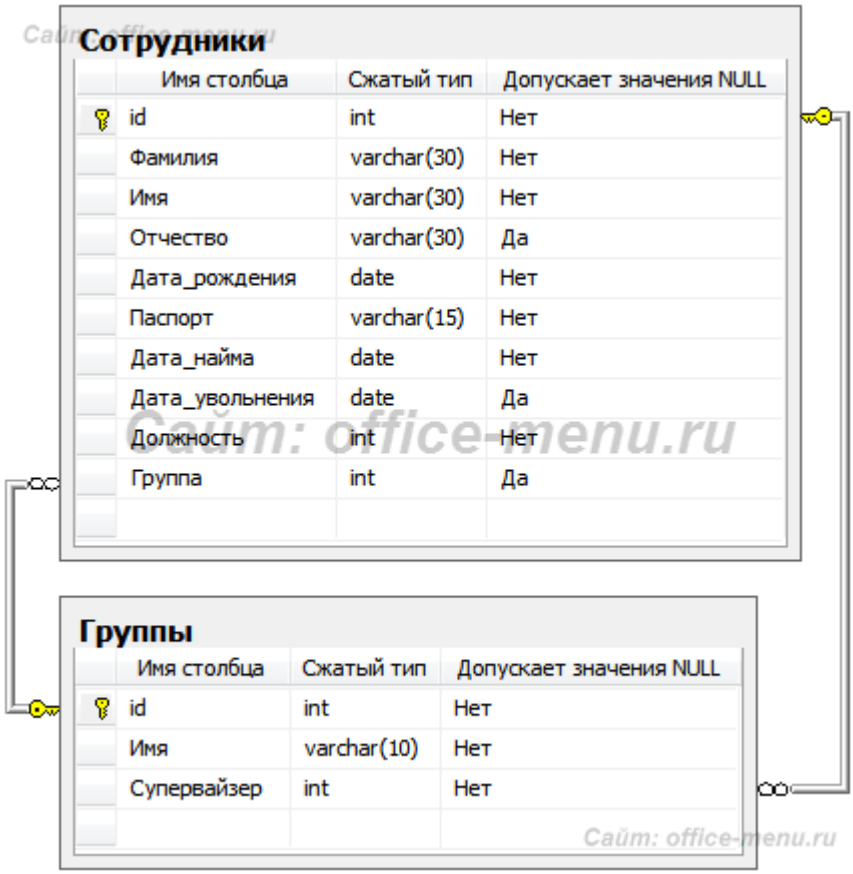
На приведенном примере таблицы сотрудников видно, что столбец «Супервайзер» имеет зависимость от столбца «Группа», а это значит, что при изменении значения поля группы, потребуется изменить значение поля супервайзера.



Сотрудники

Имя столбца	Сжатый тип	Допускает значения NULL
id	int	Нет
Фамилия	varchar(30)	Нет
Имя	varchar(30)	Нет
Отчество	varchar(30)	Да
Дата_рождения	date	Нет
Паспорт	varchar(15)	Нет
Дата_найма	date	Нет
Дата_увольнения	date	Да
Должность	int	Нет
Группа	int	Да
Супервайзер	int	Нет

Все риски, которые были рассмотрены для 2NF, так же относятся к 3NF и устраняются переносом зависимых полей в отдельную таблицу.



Денормализация базы данных

Теория нормальных форм не всегда применима на практике. Например, неатомарные значения не всегда являются «злом», а иногда наоборот. Связано это с необходимостью дополнительного объединения (следовательно, затрат производительности системы) при выполнении запросов, особенно когда производится обработка большого массива информации.

Порядок выполнения работы:

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

Форма представления результата:

представление отчета на образовательном портале (в соответствующем курсе).

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 Этапы проектирования баз данных

Лабораторная работа №2

Преобразование реляционной базы данных в сущности и связи.

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

Выполнив работу, Вы будете:

уметь:

У1. Проектировать реляционную базу данных

У 01.4 Выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы

У 02.1 Определять задачи для поиска информации

У 02.2 Определять необходимые источники информации

У04.2 Взаимодействовать с коллегами, руководством, клиентами в ходе профессиональной деятельности

У05.3 Излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке

У09.1 Применять средства информационных технологий для решения профессиональных задач

У09.2 Использовать современное программное обеспечение.

Материальное обеспечение:

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

Задание:

Создать схему базы данных по заданному варианту.

Краткие теоретические сведения:

Этапы проектирования базы данных:

– инфологическое проектирование;

– определение требований к операционной обстановке, в которой будет функционировать информационная система;

– выбор системы управления базой данных (СУБД) и других инструментальных программных средств;

– логическое проектирование БД;

– физическое проектирование БД.

Инфологическое проектирование.

Инфологическое проектирование – построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. Такая модель создается без ориентации на какую-либо конкретную СУБД и модель данных. Конкретный вид и содержание концептуальной модели базы данных определяется выбранными для этого формальным аппаратом. Обычно используют графические нотации, подобные ERдиаграммам. Чаще всего инфологическая (концептуальная) модель базы данных включает в себя: –описание информационных объектов или понятий предметной области и связей между ними; –описание ограничений целостности, то есть требований к допустимым значениям данных и к связям между ними. Пример инфологического проектирования показан на рисунке 1.

Рисунок 1 – Инфологическая модель

Entity-Relationship Diagrams

Имеется целый ряд методик создания информационно-логических моделей. Одна из наиболее популярных в настоящее время методик при разработке моделей использует ERD (Entity-Relationship Diagrams). В русскоязычной литературе эти диаграммы называют «объект – отношение» либо «сущность – связь». Модель ERD была предложена Питером

Пин Шен Ченом в 1976 г. К настоящему времени разработано несколько ее разновидностей, но все они базируются на графических диаграммах, предложенных Ченом. Диаграммы конструируются из небольшого числа компонентов. Благодаря наглядности представления они широко используются в CASE-средствах (Computer Aided Software Engineering). Рассмотрим используемую терминологию и обозначения.

Сущность (Entity) – реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области, информация о котором подлежит хранению.

Каждая сущность должна обладать уникальным идентификатором. Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа (сущности).

Каждая сущность должна обладать некоторыми свойствами:

– иметь уникальное имя; причем к этому имени должна всегда применяться одна и та же интерпретация (определение сущности). И наоборот: одна и та же интерпретация не может применяться к различным именам, если только они не являются псевдонимами;

– обладать одним или несколькими атрибутами, которые либо принадлежат сущности, либо наследуются ею через связь;

– обладать одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности.

Связь (Relationship) – поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Одна из участвующих в связи сущностей – независимая, называется родительской сущностью, другая – зависимая, называется дочерней или сущностью-потомком. Как правило, каждый экземпляр родительской сущности ассоциирован с произвольным (в том числе нулевым) количеством экземпляров дочерней сущности. Каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности-родителя. Таким образом, экземпляр сущностипотомка может существовать только при существовании сущности-родителя. Связи дается имя, выражаемое грамматическим оборотом глагола и помещаемое возле линии связи.

При построении ER-модели используются следующие принципы:

– сущности на диаграмме представляются прямоугольниками;

– каждый прямоугольник может иметь различные визуальные атрибуты;

– каждой сущности должно быть присвоено уникальное имя;

– имена сущностей необходимо задавать в единственном числе;

– связи на диаграмме представляются линиями, идущими от одной сущности (таблицы) к другой;

– каждой связи присваивается уникальное имя;

– связанные таблицы разделяют на родительские и дочерние;

– родительские таблицы отображаются прямоугольниками с прямыми углами, дочерние – со скругленными.

Логическое проектирование

Логическое проектирование – создание схемы базы данных на основе конкретной модели базы данных, например, реляционной модели данных. Для реляционной модели данных даталогическая модель – набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. Преобразование концептуальной модели в логическую модель, как правило, осуществляется по формальным правилам. Этот этап может быть в значительной степени автоматизирован. На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД. Пример логической модели представлен на рисунке 2.

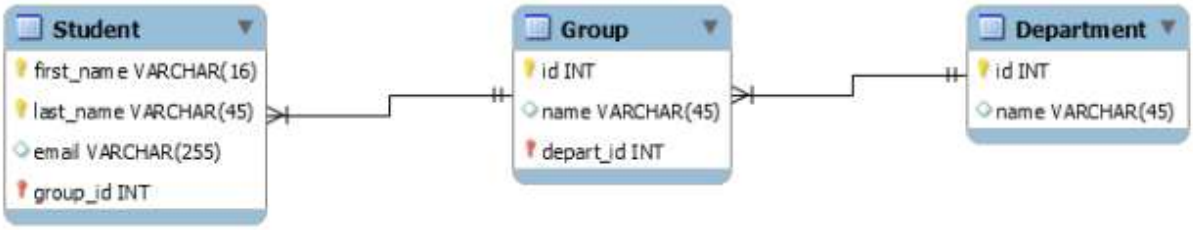


Рисунок 2 – Логическая модель

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 лабораторной работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

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

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

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 4 Проектирование структур баз данных

Лабораторная работа №3

Редактирование, добавление и удаление записей в таблице. Применение логических условий к записям. Открытие, редактирование и пополнение табличного файла.

Цель:

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

Выполнив работу, Вы будете:

уметь:

У1. Проектировать реляционную базу данных

У04.2 Взаимодействовать с коллегами, руководством, клиентами в ходе профессиональной деятельности

У05.3 Излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке

У09.1 Применять средства информационных технологий для решения профессиональных задач

У09.2 Использовать современное программное обеспечение

У 10.1 Понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые)

Материальное обеспечение:

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

Задание 1:

Создать базу данных Туризм и таблицы базы данных в СУБД MS Access.

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 ***MS Access***. 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*, *Код тура* 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.

Форма представления результата:

Отчет по выполненной лабораторной работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

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

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

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 4 Проектирование структур баз данных

Лабораторная работа №4

Создание ключевых полей. Задание индексов. Установление и удаление связей между таблицами.

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

Выполнив работу, Вы будете:

уметь:

У1. Проектировать реляционную базу данных

У09.1 Применять средства информационных технологий для решения профессиональных задач

У09.2 Использовать современное программное обеспечение

У 10.1 Понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые)

Материальное обеспечение:

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

Задание 1:

По своему варианту базы данных в СУБД Access для каждой таблицы создайте первичные и внешние ключи, установите связи и укажите индексированные поля.

Краткие теоретические сведения:

Индексы обеспечивают поиск и сортировку записей в Access. В индексе хранится местоположение записей на основе одного или нескольких полей, которые были выбраны для индексирования. После того как Access получает сведения о позиции в индексе, он может извлечь данные путем перемещения непосредственно к нужной позиции. Благодаря этому использование индекса гораздо эффективнее просмотра всех записей для поиска необходимых данных.

Выбор полей для индексирования

Вы можете создавать индексы, основанные на одном или нескольких полях. В основном требуется индексировать поля, в которых часто осуществляется поиск, сортируемые поля и поля, объединенные с полями в других таблицах, что часто используется в запросах по нескольким таблицам. Индексы ускоряют поиск и выполнение запросов, однако они могут привести к снижению производительности при добавлении или обновлении данных. Каждый раз, когда вы добавляете или изменяете запись в таблице, содержащей один или несколько индексов, Access приходится обновлять индексы. Добавление записей с помощью запроса на добавление или с помощью импортирования записей также, скорее всего, будет происходить медленнее, если таблица-получатель содержит индексы.

Примечание: Первичный ключ таблицы индексируется автоматически.

Индексировать поля с типом данных "Объект OLE", "Вычисляемый" или "Вложение" невозможно. Индексировать другие поля следует в тех случаях, когда выполняются все указанные ниже условия.

- Тип данных поля: короткий текст (текст в Access 2010), длинный текст (MEMO в Access 2010), число, Дата и время, счетчик, денежная единица, да/нет или гиперссылка.
- Предполагается поиск значений в поле.
- Предполагается сортировка значений в поле.
- Предполагается сохранение большого числа различных значений в поле. Если поле содержит много одинаковых значений, то применение индекса может не дать значительного ускорения выполнения запросов.

Составные индексы

Если предполагается, что необходимо будет часто выполнять поиск или сортировку по нескольким полям, вы можете создать индекс для этого сочетания полей. Например, если в одном запросе часто задаются условия для полей "Поставщик" и "Наименование_продукта", имеет смысл создать для этих полей составной индекс.

При сортировке таблицы по составному индексу Access сначала выполняет сортировку по первому полю, заданному для индекса. Последовательность полей определяется при создании составного индекса. Если в первом поле содержатся записи с повторяющимися значениями, затем выполняется сортировка по второму полю, заданному для индекса, и т. д.

В составной индекс можно включить до 10 полей.

Создание индекса

Перед созданием индекса необходимо решить, следует ли создать индекс для одного поля или составной индекс. Индекс для одного поля создается с помощью установки свойства **Индексированное поле**. В таблице ниже приведены возможные параметры свойства **Индексированное поле.**

Параметр свойства "Индексированное поле"	**Значение**
Нет	Не создавать индекс для этого поля (или удалить существующий индекс)
Да (допускаются совпадения)	Создать индекс для этого поля
Да (совпадения не допускаются)	Создать уникальный индекс для этого поля

При создании уникального индекса невозможно ввести новое значение в определенном поле, если такое значение уже существует в том же поле другой записи. Access автоматически создает уникальный индекс для первичных ключей, однако может потребоваться запретить создание повторяющихся значений и в других полях. Например, вы можете создать уникальный индекс для поля, в котором содержатся серийные номера, чтобы двум продуктам нельзя было присвоить один и тот же серийный номер.

Создание индекса для одного поля

1. В области навигации щелкните правой кнопкой мыши имя таблицы, в которой необходимо создать индекс, и выберите в контекстном меню пункт **Конструктор.**
2. Щелкните пункт **Имя поля** для поля, которое следует индексировать.
3. В разделе **Свойства поля** откройте вкладку **Общие.**
4. В свойстве **Индексированное** выберите значение **Да (допускаются совпадения)**, если следует разрешить повторяющиеся значения, или значение **Да (совпадения не допускаются)**, чтобы создать уникальный индекс.
5. Чтобы сохранить изменения, щелкните элемент **Сохранить** на **панели быстрого доступа** или нажмите клавиши CTRL+S.

Создание составного индекса

Чтобы создать составной индекс для таблицы, добавьте строку для каждого поля в индексе и укажите имя индекса только в первой строке. Все строки будут обрабатываться как часть одного индекса, пока не будет обнаружена строка с другим именем индекса. Чтобы вставить строку, щелкните правой кнопкой мыши место, куда вы хотите ее вставить, и выберите в контекстном меню команду **Вставить строки.**

1. В области навигации щелкните правой кнопкой мыши имя таблицы, в которой необходимо создать индекс, и выберите в контекстном меню пункт **Конструктор.**
2. На вкладке **Конструктор** в группе **Показать или скрыть** щелкните пункт **Индексы.**

Появится окно "Индексы". Измените размеры этого окна, чтобы отображались пустые строки и свойства индекса.

3. В первой пустой строке столбца **Индекс** введите имя индекса. Для индекса можно использовать либо имя одного из индексируемых полей, либо другое подходящее имя.
4. В столбце **Имя поля** щелкните стрелку, затем щелкните первое поле, которое следует использовать в индексе.
5. Следующую строку столбца **Индекс** оставьте пустой, затем в столбце **Имя поля** укажите второе индексируемое поле. Повторите этот шаг для всех полей, которые необходимо включить в индекс.
6. Чтобы изменить порядок сортировки значений полей, в столбце **Порядок сортировки** окна "Индексы" щелкните пункт **По возрастанию** или **По убыванию**. По умолчанию выполняется сортировка по возрастанию.
7. В разделе **Свойства индекса** окна **Индексы** укажите свойства индекса для строки в столбце **Имя индекса**, содержащем имя индекса. Задайте свойства в соответствии с таблицей ниже.

Подпись	Значение
Первичный	Если **Да**, то индекс является первичным ключом.
Уникальный	Если **Да**, то каждое индексируемое значение должно быть уникальным.
Пропуск пустых полей	Если **Да**, то записи с пустыми значениями в индексируемых полях будут исключены из индекса.

8. Чтобы сохранить изменения, нажмите кнопку **Сохранить** на **панели быстрого доступа** или нажмите клавиши CTRL+S.
9. Закройте окно "Индексы".

Удаление индекса

Если индекс становится ненужным или приводит к значительному снижению производительности, его можно удалить. При этом удаляется только сам индекс, а не поля, на которых он основан.

1. В области навигации щелкните правой кнопкой мыши имя таблицы, для которой необходимо удалить индекс, и выберите в контекстном меню пункт **Конструктор**.
2. На вкладке **Конструктор** в группе **Показать или скрыть** щелкните пункт **Индексы**.

Появится окно "Индексы". Измените размеры этого окна, чтобы отображались пустые строки и свойства индекса.

3. В окне "Индексы" выделите строки, содержащие индекс, который следует удалить, и нажмите клавишу DELETE.
4. Чтобы сохранить изменения, нажмите кнопку **Сохранить** на **панели быстрого доступа** или нажмите клавиши CTRL+S.
5. Закройте окно **Индексы**

Просмотр или редактирование индексов

Чтобы оценить влияние индексов на производительность или убедиться, что необходимые поля проиндексированы, просмотрите индексы в таблице.

1. В области навигации щелкните правой кнопкой мыши имя таблицы, индекс которой вы хотите изменить, и выберите в контекстном меню пункт **Конструктор**.
2. На вкладке **Конструктор** в группе **Показать или скрыть** щелкните пункт **Индексы**.

Появится окно "Индексы". Измените размеры этого окна, чтобы отображались пустые строки и свойства индекса.

3. Просмотрите или измените индексы и свойства индексов в соответствии со своими задачами.
4. Чтобы сохранить изменения, нажмите кнопку **Сохранить** на **панели быстрого доступа** или нажмите клавиши CTRL+S.
5. Закройте окно **Индексы**

Автоматическое создание индексов

В некоторых случаях индексы создаются автоматически. Например, индексы создаются для любых полей, которые определяются пользователем в качестве первичного ключа таблицы.

Для автоматического создания индекса также можно использовать параметр **Автоиндекс при импорте и создании** в диалоговом окне **Параметры Access**. Access автоматически проиндексирует все поля, имена которых начинаются с указанных в поле **Автоиндекс при импорте и создании** знаков или заканчиваются ими, например **ID, ключ, код** или **число**. Чтобы просмотреть или изменить текущие параметры, сделайте следующее:

1. Выберите **Файл** > **Параметры**.
2. Щелкните **Конструкторы объектов**, а затем в разделе **Конструктор таблиц**добавьте, измените или удалите значения в поле **Автоиндекс при импорте и создании**. Для разделения значений используйте точку с запятой (**;**).

Примечание: Если имя поля начинается со значения, указанного в списке, или заканчивается им, поле будет автоматически проиндексировано.

3. Нажмите кнопку **ОК**.

Так как каждый индекс требует дополнительной обработки, производительность при добавлении или обновлении данных снижается. Поэтому рекомендуется изменить значения, указанные в поле **Автоиндекс при импорте и создании** или уменьшить их число, чтобы сократить количество создаваемых индексов.

Порядок выполнения работы:

1. Используя рассмотренный ниже пример, установить для своей базы данных первичные ключи.
2. Установить необходимы индексированные поля.
3. Установить связи между таблицами на схеме данных.

Пример.

Установка ключевых полей.

Отдельные таблицы, содержащие информацию по определенной теме, необходимо связать в единую структуру базы данных. Для связывания таблиц следует задать **ключевые поля**.

Ключ состоит из одного или нескольких полей, значения которых однозначно определяют каждую запись в таблице. Наиболее подходящим в качестве ключевого поля является *Счетчик*, так как значения в данном поле являются уникальными (т. е. исключают повторы).

При создании таблиц в режиме конструктора ключевое поле устанавливается автоматически. Откройте созданные Вами таблицы в **режиме Конструктор** и проверьте установленные ключевые поля:

1) в таблице *Сотрудники* ключевое поле *Код сотрудника*

сотрудники

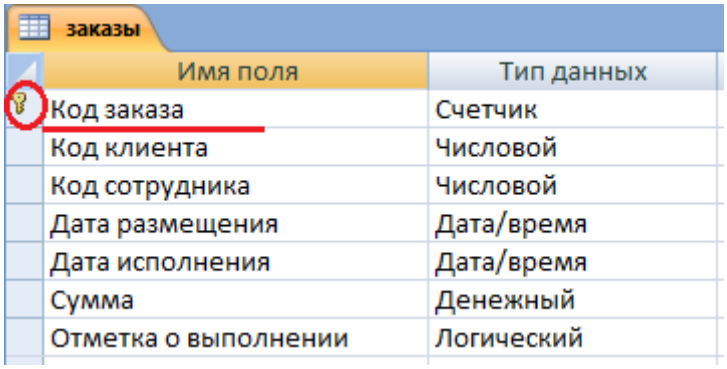
Имя поля	Тип данных
Код сотрудника	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
Должность	Текстовый
Телефон	Текстовый
Адрес	Текстовый
Дата рождения	Дата/время
Заработная плата	Денежный
Фото	Поле объекта OLE
Электронная почта	Гиперссылка

2) в таблице *Клиенты* ключевое поле *Код клиента*

клиенты

Имя поля	Тип данных
Код клиента	Счетчик
Название компании	Текстовый
Адрес	Текстовый
Номер телефона	Текстовый
Факс	Числовой
Электронная почта	Гиперссылка

3) в таблице *Заказы* ключевое поле *Код заказа*



Если значение *Ключевых полей* не задалось автоматически, то задайте их вручную. Для этого откройте **таблицу Сотрудники** в **режиме Конструктора.** Нажмите правой кнопкой мыши на поле **Код сотрудника** и в появившемся контекстном меню выберите команду **Ключевое поле**. Если в таблице необходимо установить несколько ключевых полей, то выделить их можно, удерживая клавишу Ctrl. Для **таблицы Клиенты** установите ключевое поле **Код клиента**, а для **таблицы Заказы - Код заказа.**

Создание связей между таблицами.

Существует несколько типов отношений между таблицами:

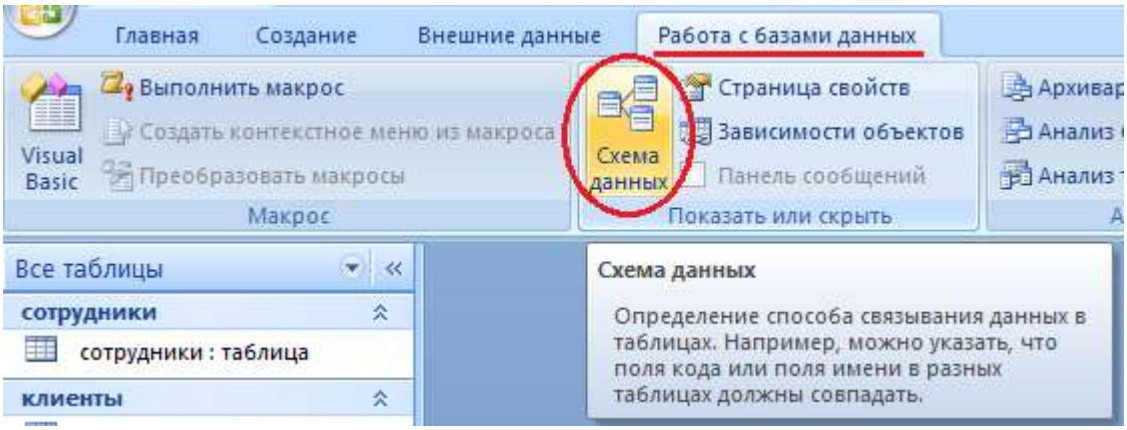
- *при отношении «один-к-одному»* каждой записи ключевого поля в первой таблице соответствует только одна запись в связанном поле другой таблицы, и наоборот. Отношения такого типа используются не очень часто. Иногда их можно использовать для разделения таблиц, содержащих много полей, для отделения части таблицы по соображениям безопасности;

- *при отношении «один-к-многим»* каждой записи в первой таблице соответствует несколько записей во второй, но запись во второй таблице не может иметь более одной связанной записи в первой таблице;

- *при отношении «многие-к-многим»* одной записи в первой таблице могут соответствовать несколько записей во второй таблице, а одной записи во второй таблице могут соответствовать несколько записей в первой.

Закройте все открытые таблицы, так как создавать или изменять связи между открытыми таблицами нельзя.

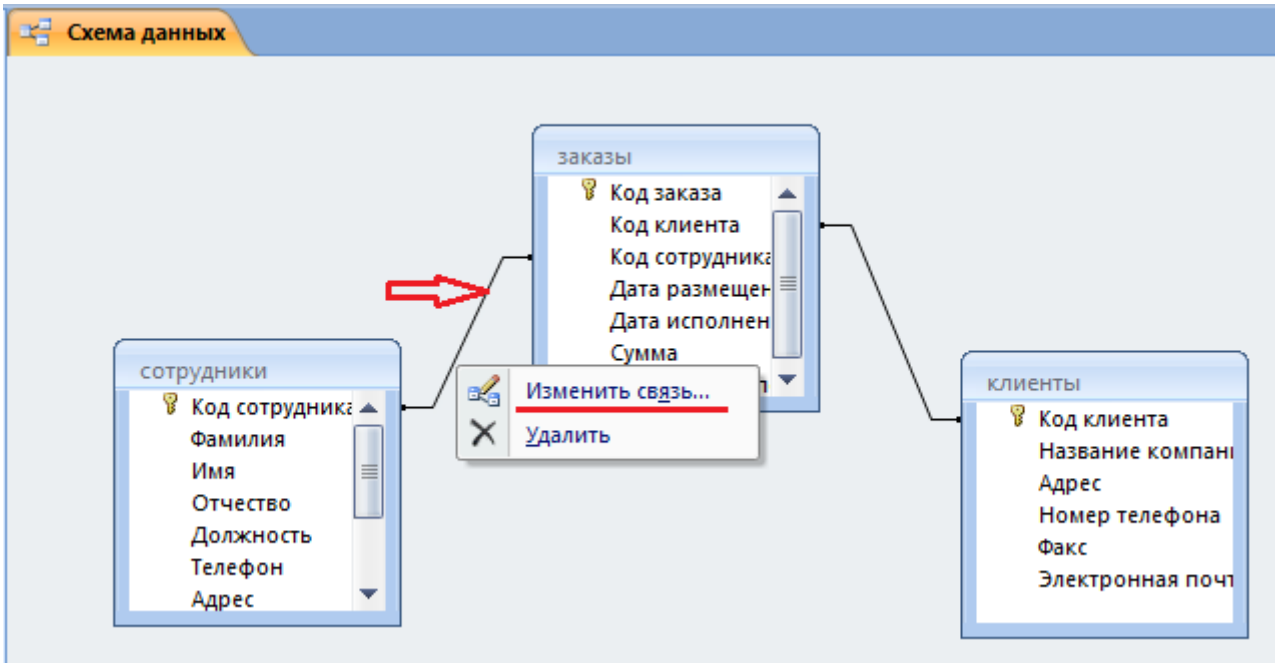
Выполните команду вкладки Лента **Работа с базами данных** кнопка **Схема данных**



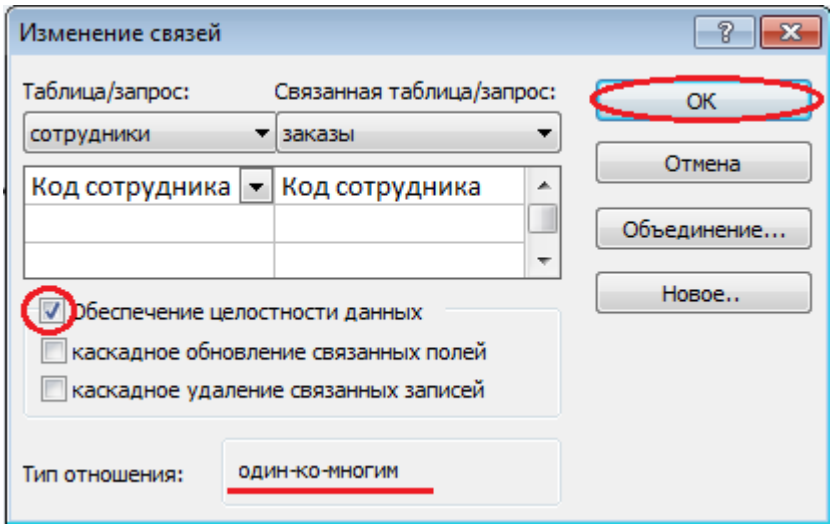
Если ранее никаких связей между таблицами базы не было, то при открытии окна **Схема данных** одновременно открывается окно **Добавление таблицы**, в котором выбираются нужные таблицы. Для добавления в схему данных новой таблицы необходимо щелкнуть правой кнопкой мыши на схеме данных и в контекстном меню выбрать пункт **Добавить таблицу**.

Если связи между таблицами уже были заданы, то откроется окно **Схема данных**, на котором будут отображены таблицы и связи между ними.

Отредактируйте связь между таблицами Сотрудники и Заказы, для этого щелкните правой кнопкой мыши (ПКМ) на линию связи и в открышемся контекстном меню выберите команду **Изменить связь.**



Откроется диалоговое окно **Изменение связей**, в котором включите флажок **Обеспечение целостности данных**. Это позволит предотвратить случаи удаления записей из одной таблицы, при которых связанные с ними данные других таблиц останутся без связи. Обратите внимание на **тип отношений: один-ко-многим.**

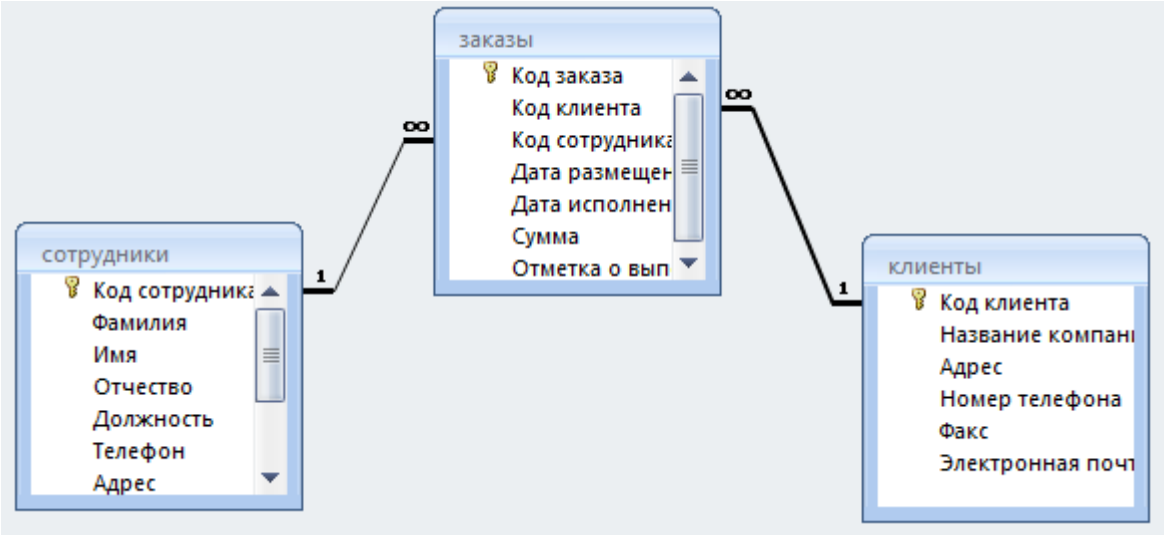


Флажки **Каскадное обновление связанных полей** и **Каскадное удаление связанных записей** обеспечивают одновременное обновление или удаление данных во всех

подчиненных таблицах при их изменении в главной таблице.Параметры связи можно изменить, нажав на кнопку Объединение. После установления всех необходимых параметров нажмите **кнопку ОК**.

Аналогично измените связь между таблицами Клиенты и Заказы.

В результате должна получиться схема данных, представленная на рисунке.



На схеме данных связи отображаются в виде соединительных линий со специальными значками около таблиц. Связь «один-к-многим» помечается «1» вблизи главной таблицы (имеющей первичный ключ) и «∞» вблизи подчиненной таблицы (имеющей внешний ключ). Связь «один-к-одному» помечается двумя «1» (оба поля таблиц имеют первичные ключи). Неопределенная связь не имеет никаких знаков. Если установлено объединение, то его направление отмечается стрелкой на конце соединительной линии (ни одно из объединенных полей не является ключевым и не имеет уникального индекса).

Форма представления результата:

Отчет по выполненной лабораторной работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

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

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

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 4 Проектирование структур баз данных

Лабораторная работа №5

Проведение сортировки и фильтрации данных. Поиск данных по одному и нескольким полям. Поиск данных в таблице.

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

Выполнив работу, Вы будете:

уметь:

У1. Проектировать реляционную базу данных
У 01.4 Выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы
У 02.1 Определять задачи для поиска информации
У 02.2 Определять необходимые источники информации
У04.2 Взаимодействовать с коллегами, руководством, клиентами в ходе профессиональной деятельности
У05.3 Излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке
У09.1 Применять средства информационных технологий для решения профессиональных задач
У09.2 Использовать современное программное обеспечение
У 10.1 Понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые)

Материальное обеспечение:

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

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 bill bull,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 MS Access 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. ***Расширенный фильтр***

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 конкретну ю страну, оформленный заданным сотрудником (например, «Какие договоры на посещение Испании заключил Сидоров?»);
    - номера телефонов, которые содержат цифру 9.

Форма представления результата:

Отчет по выполненной лабораторной работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

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

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

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 4 Проектирование структур баз данных

Лабораторная работа №6

Создание меню различных видов. Модификация и управление меню.

Цель: получение практических навыков по созданию, модификации и управлению меню.

Выполнив работу, Вы будете:

уметь:

У1. Проектировать реляционную базу данных

У 01.4 Выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы

У 02.1 Определять задачи для поиска информации

У 02.2 Определять необходимые источники информации

У04.2 Взаимодействовать с коллегами, руководством, клиентами в ходе профессиональной деятельности

У05.3 Излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке

У09.1 Применять средства информационных технологий для решения профессиональных задач

У09.2 Использовать современное программное обеспечение

У 10.1 Понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые)

Материальное обеспечение:

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

Задание1: Создать следующие макросы

Порядок выполнения задания1:

1. Откройте базу данных **Туризм.**
2. Убедитесь, что для всех основных таблиц существуют формы. При отсутствии какой-либо создайте ее любым способом.
3. Откройте форму, построенную на основе таблицы **Договоры.**
4. Для создания макроса ***Завершение***:
   - ***Макросы – Создать***;
   - ***Макрокоманда – Сообщение***;
   - ***Макрокоманда – Выход – Сохранить все.***

5. Для создания макроса ***Открыть и расположить***:
   - ***Макросы – Создать***;
   - Расположите окна базы данных и макроса без перекрытия командой ***Окно – Слева направо***;
   - В окне базы данных выберите вкладку **Формы**. Выделите и перетащите форму в окно макроса. Разместите в первой ячейке столбца **Макрокоманда**. В поле появится макрокоманда ***Открыть форму***. В столбец ***Примечание*** той же строки введите текст: «***Открытие формы***»;
   - Перейдите во вторую строку столбца **Макрокоманда**. С вкладки **Таблицы** перетащите таблицы **Клиенты** и **Сотрудники** в окно макроса во вторую и третью ячейки столбца **Макрокоманда.** Введите, если нужно, соответствующие примечания;
   - В следующей свободной ячейке столбца **Макрокоманды** выберите команду ***Выполнить Команду***. В области ***Аргументы макрокоманды*** в поле команда выберите элемент ***Рядом Вертикально***. В макрос будет включена операции разделения экрана в случае открытия нескольких окон.
6. Запустите созданный макрос. Закройте все окна, кроме окна базы данных.
7. Отредактируйте макрос ***Открыть и расположить***. Для этого на вкладке **Макросы** выберите его. Войдите в *Конструктор* и щелкните мышью в одном из полей первой строки (или выделите ее целиком), вызовите контекстное меню – ***Добавить строки***. В вставленной пустой строке в поле ***Макрокоманда*** поместите команду ***Свернуть***. Сохраните и запустите макрос.
8. Создайте макрос ***Открыть форму Договоры***. Проверьте его.
9. Закройте все окна, кроме окна базы данных.
10. Создайте группу макросов с общим именем ***Группа***. Включите в эту группу следующие макросы:
    - Подача сигнала – макрокоманда ***Сигнал***;
    - Открытие таблиц (**Открыть таблицу**) **Договоры**, **Клиенты**, **Страны**, **Сотрудники**;
    - Размещение открытых таблиц на экране в виде горизонтальной мозаики (**Выполнить команду – Рядом Вертикально**);

- Закрытие всех таблиц (**Закрыть**);
- Выход из Access (**Выход**).

11. Проверьте работу каждого из макросов созданной группы.

Задание2: Создать макрос *Autoexec.*

Порядок выполнения задания2:

1. Создайте макрос ***Открытие формы Договоры***;
2. Создайте новую форму, включив в нее заголовок «***Вас приветствует база данных Туризм***», какую-либо картинку и кнопку «***Открыть форму Договоры***»;
3. Назначьте для этой кнопки макрос «***Открыть форму Договоры***» (контекстное меню на кнопке – **Свойства** – вкладка **События** – **Нажатие кнопки** – имя макроса). Закройте созданную форму и дайте ей название ***Заставка***;
4. Создайте новый макрос с именем ***Autoexec***, состоящий из следующих макрокоманд:
5. ***Выполнить Команду – Окно Закрыть***;
6. Открыть форму;
7. Развернуть.
8. Выполните макрос ***Завершение***.
9. Загрузите базу данных **Туризм**. Проверьте работу макросов ***Autoexec*** и ***Открыть форму Договоры***.
10. Закройте базу данных. Запустите ее вновь с нажатой клавишей SHIFT. При этом макрос ***Autoexec*** не выполняется.

Задание3: Создать и выполнить макрос *Поиск договора.*

Порядок выполнения задания3:

1. Откройте новое окно макроса и перетащите форму ***Договоры*** в первую строку;
2. Во второй строке выберите макрокоманду ***К элементу Управления***. Для аргумента «Имя элемента» установите значение ***Код клиента***. Так задается поле, среди значений которого будет осуществляться поиск;
3. В следующей строке макроса выберите макрокоманду **Найти-Запись**. Для аргумента «*Образец поиска*» задайте любой код клиента. Остальные аргументы оставьте без изменения.
4. Сохраните макрос под именем «***Поиск договора***» и запустите его. В результате программа откроет форму, выполнит поиск и пометит найденное значение. Если заданное значение не будет найдено, то маркированным останется первый элемент (первая запись) формы.

Задание4: Создать и выполнить макрос для открытия формы

Порядок выполнения задания4:

1. Создайте три макроса для открытия форм ***Клиенты, Сотрудники, Страны***.
2. Откройте форму ***Договоры*** и создайте в ней несколько кнопок для открытия всех форм базы данных. Каждую кнопку снабдите понятным названием или изображением.

3. Свяжите каждую кнопку с соответствующим макросом.
4. Проверьте все кнопки.
5. Создайте кнопку на форме ***Договоры*** с изображением самолета. Свяжите ее с макросом ***Завершение***. Снабдите кнопку всплывающей подсказкой «***Завершение работы***».

Выполнение макроса с условиями

Задание5: Создайте макрос, который каждый раз после ввода в форму ***Договоры*** сведений о новом групповом договоре должен выводить на экран сообщение о том, что клиент является групповым.

Порядок выполнения задания5:

1. Создайте новый макрос и присвойте ему имя ***Групповой клиент***:
   - Если столбец **Условие** не отображается на экране, щелкните на кнопке **Условие** на панели инструментов или выполните команду **Вид – Условие**;
   - В первую ячейку столбца условий введите с помощью команды ***Построить*** логическое выражение **[Forms]![Договор]![Число туристов]>1**;
   - В той же строке, но в столбце **Макрокоманда** выберите макрокоманду **Сообщение** и задайте значение «*Групповой клиент*» для аргумента «*Сообщение*». Именно эта фраза отобразится на экране в окне сообщения при выполнении заданного условия. Для аргумента «*Тип*» установите значение «*Информационное*». При этом в окне сообщения, кроме текста, появится значок с изображением литеры «i» (стандартный вариант в Windows при выдаче сообщений), в поле ***Заголовок*** введите – «***Внимание!***»;
   - Сохраните созданный макрос с именем ***Групповой клиент***.
2. Свяжите макрос с формой ***Договоры***:
   - Откройте форму договоры в режиме *Конструктора*;
   - Откройте окно свойств формы, выполнив двойной щелчок в указанной на рисунке области;
   - Найдите на вкладке ***События*** поле ***После обновления*** и выберите в списке макрос ***Групповой клиент***;
   - Закройте окно свойств формы и перейдите в режим заполнения;
   - Перейдите к новой записи и заполните ее (в поле ***Число туристов*** введите любое число больше 1). СООБЩЕНИЕ ПОЯВИТСЯ ТОЛЬКО ПОСЛЕ ЗАВЕРШЕНИЯ РАБОТЫ С ЗАПИСЬЮ.
3. Закройте форму ***Договоры***.

Комбинации клавиш для запуска макросов

4. Назначьте для открытия таблицы **Договоры** – CTRL+ноль, для открытия формы **Сотрудники** – CTRL+F1.
5. Сохраните файл.

Задание6: Создать на основе макросов меню, контекстных меню и панель инструментов

Порядок выполнения задания 6:

1. На основе группы макросов **Группа** создайте меню, панель инструментов и контекстное меню.
2. Свяжите появление меню (контекстного меню или панели инструментов) с активизацией формы ***Договоры*** («Строка меню» («Панель инструментов», «Контекстное меню» в Свойствах) – **Группа**).
3. Откройте форму ***Договоры*** и проверьте все кнопки из группы макросов.
4. Сохраните файл. Сдайте работу преподавателю.

Форма представления результата:
Отчет по выполненной лабораторной работе.

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

Тема 4 Проектирование структур баз данных

Лабораторная работа №7

Создание файла проекта базы данных. Создание интерфейса входной формы. Использование исполняемого файла проекта БД, приемы создания и управления.

Цель:

получение практических навыков по освоению операций создания файла проекта базы данных

Выполнив работу, Вы будете:

уметь:

У1. Проектировать реляционную базу данных
У 01.4 Выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы
У 02.1 Определять задачи для поиска информации
У 02.2 Определять необходимые источники информации
У04.2 Взаимодействовать с коллегами, руководством, клиентами в ходе профессиональной деятельности
У05.3 Излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке
У09.1 Применять средства информационных технологий для решения профессиональных задач
У09.2 Использовать современное программное обеспечение
У 10.1 Понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые)

Материальное обеспечение:

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

Задание 1:

Создать файл проекта для базы данных Students.

Порядок выполнения работы:

Создание пользовательского интерфейса БД "Students" в "Microsoft Visual Studio". Для начала необходимо создать новый проект. Для этого запустите "Microsoft Visual 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 Microsoft SQL Server.

Для создания нового проекта на стартовой странице в области "Recent Projects" (Недавние проекты) необходимо щелкнуть ЛКМ 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 (Область Templates:) выберите "Windows Forms Application" (Приложение Windows). В качестве имени проекта (Поле ввода Name:) задайте "StudentsDB" и нажмите кнопку "Ok".

После создания нового проекта необходимо подключить к проекту созданную ранее в "Microsoft SQL Server" БД "Students". Для подключения БД к проекту в оконном меню среды разработки выберите пункт "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 к БД "Students" то выбираем вариант БД (Database) и нажимаем кнопку "Next" (Далее). Появится окно выбора подключения к БД (Choose Your Data Connection).

В окне выбора подключения к БД, 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 установке SQL сервера. В качестве логина и пароля для входа на сервер (Log on to the server) выберите "Use Windows Authentication" (Использовать логин и пароль учетной записи Windows). В качестве БД для подключения (Connect to a database) из выпадающего списка "Select or enter a database name:" (Выберите или введите имя БД) выберите БД "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 к БД (Choose Your Data Connection). Просмотрите созданную строку подключения "Connection string", щелкнув по знаку"+"в нижней части окна.

В окне выбора подключения к БД (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 в файл настроек приложения).

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

Появится окно выбора объектов подключаемой БД (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".

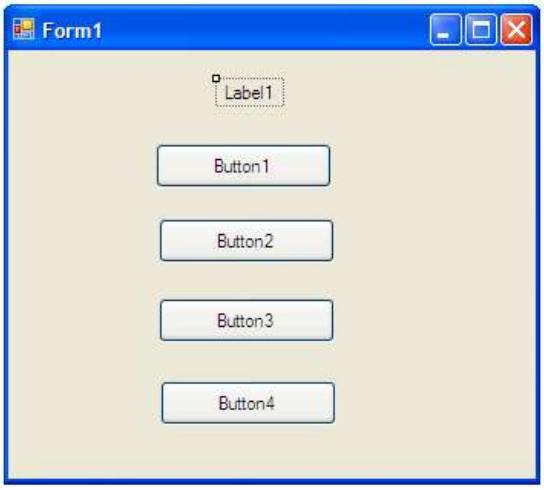
Задание 2:

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 на рисунке.



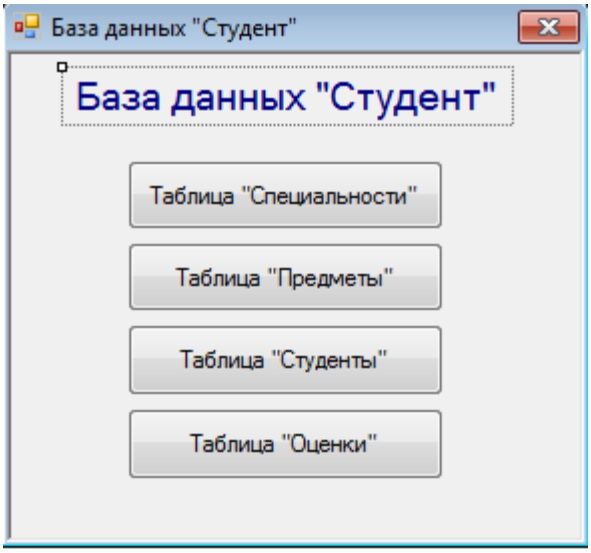
После создания объектов перейдем к настройке их свойств. Начнем с настройки свойств формы. Выделите форму, щелкнув ЛКМ в пустом месте формы. На панели свойств задайте свойства формы как представлено ниже:

- 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 ней ЛКМ и на панели свойств, задайте свойства надписи следующим образом:

- AutoSize (Авторазмер): False;
- Font (Шрифт): Microsoft Sans 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 на рисунке.



Задание3:
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', '');
```

Виды включений

К видам включений относятся `include`, `require`, `include_once` и `require_once`.

Команды `include` и `require`. почти идентичны. 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_once` и `require_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_once и require_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 лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

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

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

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 4 Проектирование структур баз данных

Лабораторная работа №8

Создание формы. Управление внешним видом формы

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

Выполнив работу, Вы будете:

уметь:

У09.1 Применять средства информационных технологий для решения профессиональных задач.

У09.2 Использовать современное программное обеспечение.

Материальное обеспечение:

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

Задание 1:

Разработать формы для базы данных Туризм в СУБД MS Access.

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*.

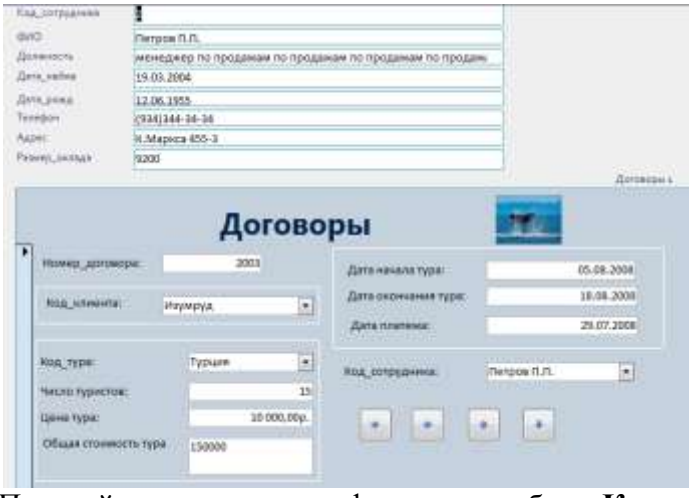
   Создание формы с помощью мастера
5. Создайте с помощью **Мастера форм** новую форму ***Сотрудники*** для одноименной таблицы. Включите в нее все поля исходной таблицы.
6. Выберите фон, на котором будут размещаться поля формы, перебрав в окне ***Мастера*** несколько вариантов оформления.
7. Завершите 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.

    Создание формы 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 ***Списка полей*** 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]
    Создание подчиненных 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 оконном меню выберите пункт **"Project/Add Windows Form"**. Появится окно **"Add New Item - StudentsDB"** (Добавить новый компонент).

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

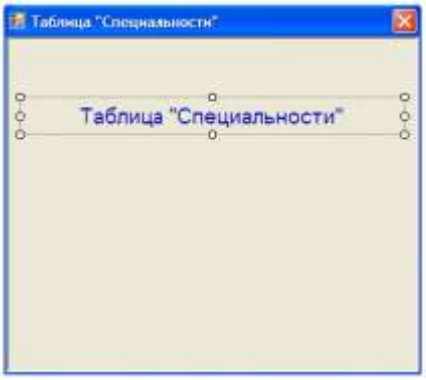
Перейдем к настройке свойств формы и надписи. Выделите форму, щелкнув **ЛКМ** в пустом месте формы. На панели свойств задайте свойства формы следующим образом:

- **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): Microsoft Sans 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"** (Data Sources), 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 виде подтаблицы располагается таблица **"Студенты"**. Подтаблица показывает, что таблица **"Студенты"** является вторичной по отношению к таблице специальности.

При выделении, какого-либо поля таблицы, оно будет отображаться в виде выпадающего списка, позволяющего выбирать объект, 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 помещаем поле **"Код специальности"** на нашу форму, так как данное поле является первичным полем связи и заполняется автоматически. Конечный пользователь не 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 к главной кнопочной форме, а затем запустить проект и открыть форму **"Специальности"** при помощи кнопки на главной кнопочной форме.

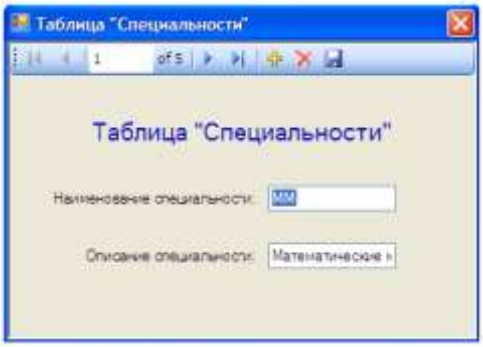
Отобразите главную кнопочную форму в рабочей области среды разработки, щелкнув по вкладке **"Form1.vb [Design]"** в верхней части рабочей области. Для подключения новой формы **"Специальности"** к главной кнопочной форме дважды щелкните **ЛКМ** по кнопке **"Таблица "Специальности""**, расположенной на главной 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 появится главная кнопочная форма. Для открытия формы, отображающей таблицу **"Специальности"** на главной кнопочной форме, нажмите кнопку **"Таблица "Специальности""**. Появится форма с соответствующей таблицей.



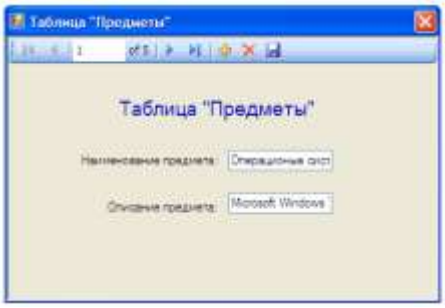
Проверьте работу панели навигации, расположенной в верхней части формы, понажимав на ней различные кнопки. Вернитесь в среду разработки, просто закрыв форму с таблицей **"Специальности"** и главную кнопочную форму.

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".**

На главной кнопочной форме дважды 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 на главной кнопочной форме нажмите кнопку **"Таблица "Предметы""**. Отобразится таблица предметы имеющая следующий вид:

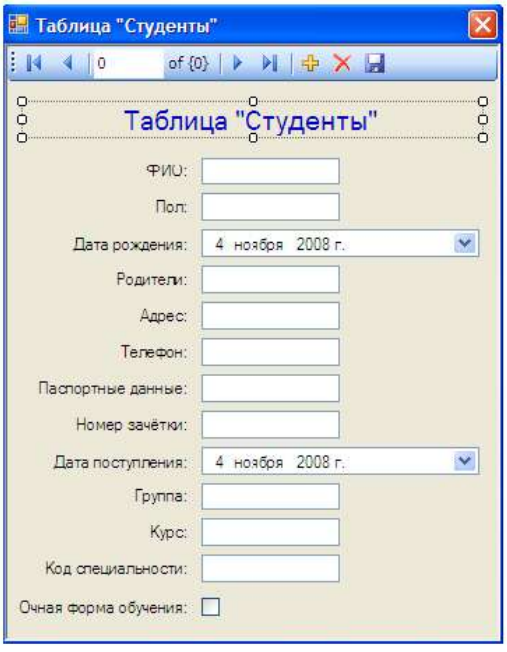


Проверьте работу панели навигации, нажатием на кнопки на данной панели в верхней части формы. Для возвращения в среду разработки закройте форму таблицы **"Предметы"** и главную кнопочную форму.

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 используется объект **"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 данное поле является первичным полем связи. Новая форма примет вид:

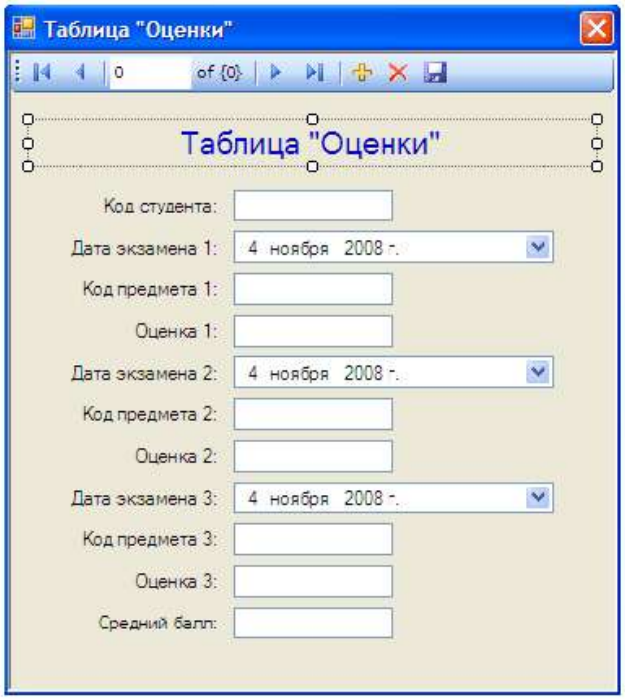


Обратите внимание на объекты, отображающие поля **"Дата рождения"**, **"Дата поступления"** и **"Очная форма обучения"**.

Подключим форму, отображающую таблицу **"Студенты"** к главной кнопочной форме. Отобразите главную кнопочную форму и 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.EventArgs) Handles Button3.Click
    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.

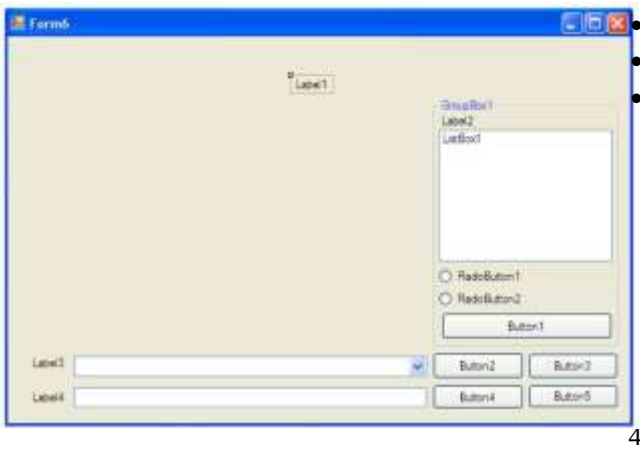
Задание 3:

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**).

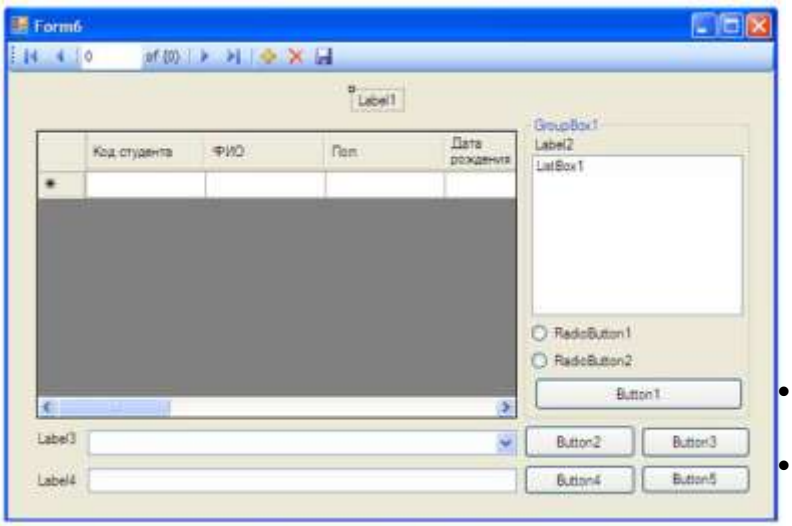


Расположите объекты как показано на рисунке.

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).

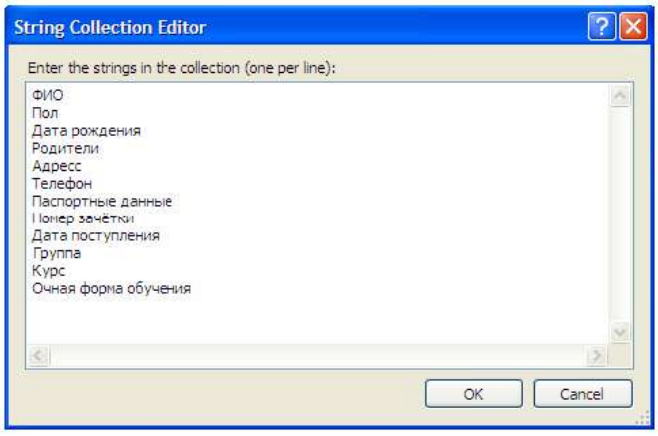
Для надписи **Label1** задайте:

- **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 на рисунке, 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…"**.

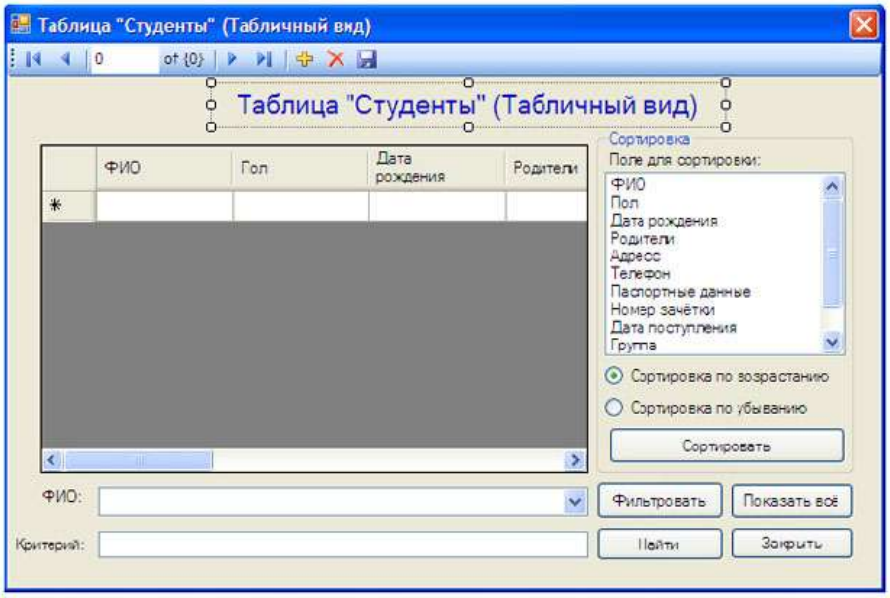
Появится окно настройки свойств полей таблицы **"Edit Columns"**

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 на рисунке 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 команду СтудентыBindingSource.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.

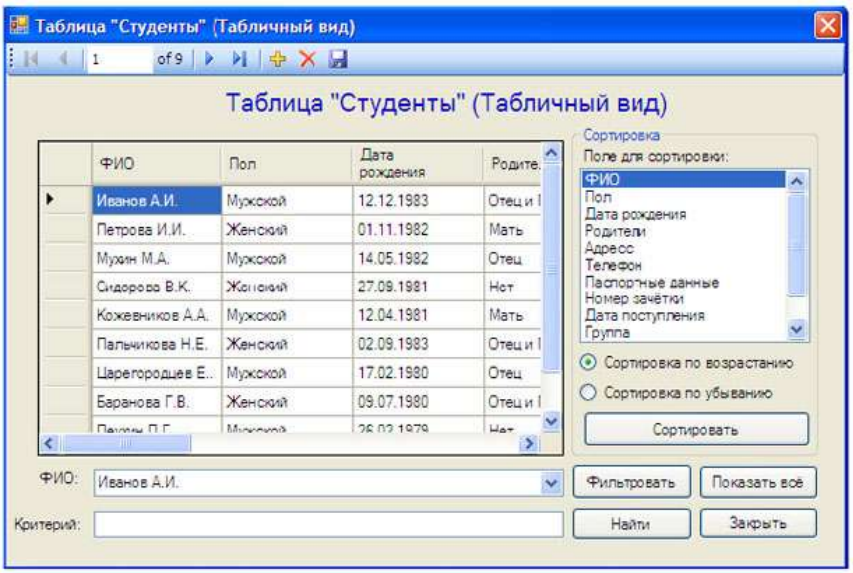
```
Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click
    Me.Close()

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 лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

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

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

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 5 Организация запросов SQL

Лабораторная работа №9

Создание проекта БД. Создание БД. Редактирование и модификация таблиц.

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

Выполнив работу, Вы будете:

уметь:

У1. Проектировать реляционную базу данных.

У 02.1 Определять задачи для поиска информации.

У 02.2 Определять необходимые источники информации.

У09.1 Применять средства информационных технологий для решения профессиональных задач

У09.2 Использовать современное программное обеспечение.

Материальное обеспечение:

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

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 будут заданы для всех столбцов таблицы, то приводить список столбцов необязательно. Если столбец таблицы не включен в список, то в этом столбце при добавлении строки будет автоматически установлено значение по умолчанию.

• Значения, которые нужно добавить в таблицу. Значения могут указываться в одном из следующих форматов:

– набор значений для каждой добавляемой строки заключается в скобки. Набор значений внутри каждой пары скобок должен соответствовать указанному списку столбцов, а если список столбцов не указан, то упорядоченному списку всех столбцов, составляющих

таблицу (список столбцов таблицы можно просмотреть с помощью команды DESCRIBE. Значения внутри набора, а также сами наборы отделяются друг от друга запятыми;

– символьные значения, а также значения даты и времени приводятся в одинарных кавычках. Для числовых значений кавычки необязательны. Десятичным разделителем для чисел 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 обычная символьная 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) автоматически вносятся порядковые номера строк, поскольку этот столбец имеет тип данных SERIAL.*

Порядок выполнения работы:

Используя команду INSERT вставить 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 мыши по таблице **"Специальности"** и в появившемся меню выберите пункт **"Изменить первые 200 строк"**. В рабочей области "Microsoft SQL Server Management Studio" 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 как поле **"Код специальности"** является первичным полем связи и ключевым числовым счетчиком, то оно заполняется автоматически (заполнять его не нужно).

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

После заполнения таблицы **"Специальности"** 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 таблицы

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

"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"**.

По окончании 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 лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

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

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

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 5 Организация запросов SQL

Лабораторная работа №10

Заполнение массива из табличного файла. Заполнение табличного файла из массива.

Цель: получение практических навыков по освоению операций экспорта и импорта данных.

Выполнив работу, Вы будете:

уметь:

У 02.1 Определять задачи для поиска информации.

У 02.2 Определять необходимые источники информации.

У09.1 Применять средства информационных технологий для решения профессиональных задач

У09.2 Использовать современное программное обеспечение.

Материальное обеспечение:

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

Задание1: Выполнить экспорт данных из базы данных

1. В базе данных **Туризм** выделите таблицу **Сотрудники**.
2. Экспортируйте эту таблицу в файл типа Excel. Для этого в команде ***Файл – Экспорт*** в поле ***Тип файла*** выберите из предлагаемого списка «**Microsoft Excel**».
3. Запустите Excel и откройте полученный файл. При необходимости поменяйте шрифт. Обратите внимание на наличие установленного системой примечания в первой ячейке. Измените подпись рабочего листа на «***Адреса***».

Задание2: Выполнить импорт данных в базу данных

1. Создайте с Excel на основе таблицы ***Адреса*** новую таблицу ***Картотека адресов***, исключив из таблицы ***Адреса*** поля ***Код сотрудника***, ***Должность***, ***Размер оклада***, ***Дата Найма***. Сохраните ее в файле с именем *Address.xls*.
2. Импортируйте ее (создайте на ее основе новую таблицу) в базу данных **Туризм**:
   - Откройте базу данных **Туризм**;
   - ***Файл – Внешние данные – Импорт***;
   - Если в файле Excel, который используется для импорта, находится несколько листов с таблицами, то необходимо указать, какую взять за основу для построения таблицы базы данных;
   - Далее ***Мастер импорта*** попросит уточнить, считать ли первую импортированную строку заголовками таблицы (на этом можно остановиться, нажав **Готово**);
   - Разрешите ***Мастеру импорта*** самому установить первичный ключ для импортируемой таблицы;
   - Присвойте новой таблице имя «**Адреса**».
3. Проверьте наличие новой таблицы на вкладке **Таблицы**. Откройте и просмотрите ее.
4. Закройте базу данных.

Задание3: Использовать базу данных, как источник при слиянии документов.

1. Для всех клиентов, которые являются групповыми, нужно подготовить и разослать письмо с сообщением о новом открывающемся туре в экзотическую страну.

2. Откройте *Word*. Создайте в окне следующее письмо:

Адрес: «Адрес» Телефон: «Телефон» Получатель «Клиент» Уважаемый «Контактное лицо!» Спешим информировать Вас, что наша фирма с 1 июня 2009 года открывает новый маршрут в экзотическую страну Острова Зеленого Мыса. Вас ждут приключения и неожиданности в роскошном природном оазисе Африки. Приглашаем Вас принять участие. Менеджер отдела продаж Петров А.А.

3. Выделенные и заключенные в кавычки поля должны соответствовать полям таблицы **Клиенты**.
4. Отправьте каждому групповому клиенту созданное письмо, осуществив слияние документов – текста письма и атрибутов адресата, взятых из базы данных. Для этого в окне Word выполните:
   - ***Сервис – Письма и рассылки – Мастер слияния***;
   - Далее нужно следовать указаниям *Мастера слияния* в нижней части окна;
   - ***Источник данных***: ***Получить данные – Выбор получателей*** – выберите базу данных **Туризм** – в нем таблицу ***Клиенты***;
   - Для выбора из таблицы только групповых клиентов установите автофильтр по ***Признаку группы***;
   - Посредством кнопки ***Другие Элементы*** внесите в письмо на место названий, заключенных в кавычки, соответствующие поля из таблицы **Клиенты**;
   - Осуществите слияние данных, используя кнопку ***Просмотр писем*** панели инструментов **Слияние**.
5. Просмотрите полученные результаты.

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

Тема 5 Организация запросов SQL

Лабораторная работа №11

Добавление записей в табличный файл из двумерного массива. Работа с командами ввода-вывода. Использование функций для работы с массивами.

Цель: получение практических навыков импортирования данных в базу данных

Выполнив работу, Вы будете:
уметь:
У 02.1 Определять задачи для поиска информации.
У 02.2 Определять необходимые источники информации.
У09.1 Применять средства информационных технологий для решения

профессиональных задач
У09.2 Использовать современное программное обеспечение.

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

Задание:
Выполнить импорт данных в базу данных.

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

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

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

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

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

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

4. Для загрузки данных из созданного файла выполните команду

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

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

Порядок выполнения работы:

1. Создайте базу данных на сервере MySQL.
2. Сценарий SQL предоставлен так, чтобы создать большинство таблиц и вставки данных в них. Все, что нужно сделать, это импортировать сценарий SQL в вашу базу данных. database.sql.
3. Таблица Сотрудники (персонал, должности) не включены в этот сценарий SQL. Обратитесь к диаграмме базы данных (ERD) и словарю данных.
4. Создайте таблицы сотрудников (персонал, положение и расписаний) согласно спецификации.
5. Все данные сотрудников представлены в файле employee-import.
6. Эти данные не отформатированы для импортирования непосредственно в базу данных, необходимо отформатировать данные и загрузить их в таблицы, которые вы только что создали.
7. В поле " Full Name" в формате "Имя Фамилия" используются разные символы разделителя.
8. Убедитесь, что адреса электронной почты в правильном формате

Форма представления результата:
Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

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

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

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 5 Организация запросов SQL

Лабораторная работа №12

Задание значений и ограничений поля. Проверка введенного в поле значения. Отображение данных числового типа и типа дата.

Цель: получение практических навыков по заданию и проверки ограничений**.**

Выполнив работу, Вы будете:

уметь:

У09.1 Применять средства информационных технологий для решения профессиональных задач

У09.2 Использовать современное программное обеспечение.

Материальное обеспечение:

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

Задание1:

Создать по своей базе данных необходимые ограничения на столбцы и при необходимости на таблицы.

Краткие теоретические сведения:

Существует два основных типа ограничений — ограничения на столбцы и ограничения на таблицу.

Ограничения на столбцы (COLUMN CONSTRAINTS) применимы только к отдельным столбцам, а ограничения на таблицу (TABLE CONSTRAINTS) применимы к группам, состоящим из одного или более столбцов.

Ограничения на столбец добавляются в конце определения столбца после указания типа данных и перед окончанием описания столбца (запятой). Ограничения на таблицу размещаются в конце определения таблицы, после определения последнего столбца.

Команда CREATE TABLE имеет следующий синтаксис, расширенный включением ограничений:

CREATE TABLE <ИМЯ таблицы >

(<имя столбца > <тип данных> Ограничения на столбец>,
<имя столбца> <тип данных> Ограничения на столбец>,
Ограничения на таблицу> (<имя столбца>[,<имя столбца>]));

Поля, заданные в круглых скобках после описания ограничений таблицы, — это поля, на которые эти ограничения распространяются. Ограничения на столбцы применяются к тем столбцам, в которых они описаны.

Ограничение NOT NULL

Чтобы запретить возможность использования в поле NULL-значений, можно при создании таблицы командой CREATE TABLE указать для соответствующего столбца ключевое слово NOT NULL.

Это ограничение применимо только к столбцам таблицы. NULL — это специальный маркер, обозначающий тот факт, что поле пусто. Но он полезен не всегда. Первичные ключи, например, в принципе не должны содержать NULL-значений (быть пустыми), поскольку это нарушило бы требование уникальности первичного ключа (более строго — функциональную зависимость атрибутов таблицы от первичного ключа). Во многих других случаях также необходимо, чтобы поля обязательно содержали определенные значения. Если ключевое

слово NOT NULL размещается непосредственно после типа данных

(включая размер) столбца, то любые попытки оставить значение поля пустым (ввести в поле NULL-значение) будут отвергнуты системой.

Например, для того, чтобы в определении таблицы STUDENT запретить использование NULL-значений для столбцов STUDENT_ID, SURNAME и NAME, можно записать следующее:

```
CREATE TABLE STUDENT
 (STUDENT_ID INTEGER NOT NULL,
SURNAME     CHAR ( 52)    NOT NULL,
NAME        CHAR (10)   NOT NULL,
STIPEND     INTEGER,
KURS        INTEGER,
CITY        CHAR ( 5 ,1)
BIRTHDAY    DATE,
UNIV_ID     INTEGER);
```

Важно помнить: если для столбца указано NOT NULL, то при использовании команды INSERT обязательно должно быть указано конкретное значение, вводимое в это поле. При отсутствии ограничения NOT NULL в столбце значение может отсутствовать, если только не указано значение столбца по умолчанию(DEFAULT). Если при создании таблицы ограничение NOT NULL не было указано, то его можно указать позже, используя команду ALTER TABLE. Однако для того, чтобы для вновь вводимогос помощью команды ALTER TABLE столбца можно было задать ограничение NOT NULL, таблица, в которую добавляется столбец, должна быть пустой.

Уникальность как ограничение на столбец

Иногда требуется, чтобы все значения, введенные в столбец, отличались друг от друга. Например, этого требуют первичные ключи. Если при создании таблицы для столбца указывается ограничение UNIQUE, то база данных отвергает любую попытку ввести в это поле какой-либо строки значение, уже содержащееся в том же поле другой строки. Это ограничение применимо только к тем полям, которые были объявлены NOT NULL. Можно

предложить следующее определение таблицы STUDENT, использующее ограничение UNIQUE:

```
CREATE TABLE STUDENT
(STUDENT_ID INTEGER NOT NULL UNIQUE,
   SURNAME     CHAR (25)   NOT NULL,
   NAME        CHAR (10)   NOT NULL,
   STIPEND     INTEGER,
   KURS        INTEGER,
   CITY        CHAR ( 5 ,1)
   BIRTHDAY    DATE,
   UNIV_ID     INTEGER);
```

Объявляя поле STUDENT_ID уникальным, можно быть уверенным, что в таблице не появится записей для двух студентовс одинаковыми идентификаторами. Столбцы, отличные

от первичного ключа, для которых требуется поддержать уникальность значений, называются возможными ключами или уникальными ключами (CANDIDATE KEYS ИЛИ UNIQUE KEYS).

Уникальность как ограничение таблицы

Можно сделать уникальными группу полей, указав UNIQUE в качестве ограничений таблицы. При объединении полей в группу важен порядок, в котором они указываются. Ограничение на таблицу UNIQUE является полезным, если требуется поддерживать уникальность группы полей. Например, если в нашей базе данных не допускается, чтобы студент сдавал в один день больше одного экзамена, то можно в таблице объявить уникальной комбинацию значений полей STUDENT_ID и EXAM_DATE. Для этого следует создать таблицу EXAM_MARKS следующим способом:

```
CREATE TABLE EXAM_MARKS
   (EXAM_ID     INTEGER NOT NULL,
   STUDENT_ID   INTEGER NOT NULL,
   SUBJ_ID      INTEGER NOT NULL,
   MARK         CHAR ( ) ,
 EXAM_DATE      DATE NOT NULL,
 UNIQUE    (STUDENT_ID, EXAM_DATE));
```

Обратите внимание, что оба поля в ограничении таблицы UNIQUE все еще используют ограничение столбца — NOT NULL.

Если бы использовалось ограничение столбца UNIQUE для поля STUDENT_ID, то такое ограничение таблицы было бы необязательным.

Если значение поля STUDENT_ID должно быть различным для каждой строки в таблице EXAM_MARKS, это можно сделать, объявив UNIQUE как ограничение самого поля STUDENT_ID.B этом случае не будет и двух строк с идентичной комбинацией значений полей STUDENT_ID, EXAM_DATE. Следовательно, указание UNIQUE как ограничение таблицы наиболее полезно использовать в случаях, когда не требуется уникальность индивидуальных полей, как это имеет место на самом деле в рассматриваемом примере.

Присвоение имен ограничениям

Ограничениям таблиц можно присваивать уникальные имена. Преимущество явного задания имени ограничения состоит в том, что в этом случае при выдаче системой сообщения о на-

рушении установленного ограничения будет указано его имя, что упрощает обнаружение ошибок.

Для присвоения имени ограничению используется несколько измененный синтаксис команд CREATE TABLE и ALTER TABLE.

Приведенный выше пример запроса изменяется следующим образом:

```
CREATE TABLE EXAM_MARKS
  (EXAM_ID      INTEGER NOT NULL,
 STUDENT_ID     INTEGER NOT NULL,
 SUBJ_ID        INTEGER NOT NULL,
 MARK           CHAR ( )
                1,
 EXAM_DATE      DATE NOT NULL,
 CONSTRAINT     STUD_SUBJ_CONSTR
 UNIQUE         (STUDENT ID, EXAM DATE);
```

В этом запросе STUD__SUBJ_CONSTR — это имя, присвоенное указанному ограничению таблицы.

Ограничение первичных ключей

Первичные ключи таблицы — это специальные случаи комбинирования ограничений UNIQUE и NOT NULL. Первичные ключи имеют следующие особенности:

• таблица может содержать только один первичный ключ;

• внешние ключи по умолчанию ссылаются на первичный ключ таблицы;

• первичный ключ является идентификатором строк таблицы (строки, однако, могут идентифицироваться и другими способами).

Улучшенный вариант создания таблицы STUDENTI с объявленным первичным ключом имеет теперь следующий вид:

```
CREATE TABLE STUDENT
(STUDENT_ID INTKGER PRIMARY KEY,
   SURNAME CHAR ( 5 NOT NULL,2)
   NAME CHAR (10) NOT NULL,
   STIPEND INTEGER,
   KURS   INTEGER,
   CITY CHAR ( 5 ,1)
   BIRTHDAY DATE,
   UNIV_ID INTEGER);
```

Проверка значений полей

Ограничение CHECK позволяет определять условие, которому должно удовлетворять вводимое в поле таблицы значение, прежде чем оно будет принято. Любая попытка обновить или заменить значение поля такими, для которых предикат, задаваемый ограничением CHECK, имеет значение ложь, будет отвергаться.

Рассмотрим таблицу STUDENT. Значение столбца STIPEND в этой таблице выражается десятичным числом. Наложим назначения этого столбца ограничение — величина размера стипендии должна быть меньше 200.

Соответствующий запрос имеет следующий вид:

```
CREATE TABLE STUDENT
 (STUDENT_ID INTEGER PRIMARY KEY,
SURNAME      CHAR ( 52)     NOT NULL,
NAME         CHAR (10)    NOT NULL,
STIPEND      INTEGER CHECK (STIPEND < 20 ,0)
KURS         INTEGER,
CITY         CHAR ( 5 ,1)
BIRTHDAY     DATE,
UNIV_ID      INTEGER);
```

Задание2.

По Учебной базе данных создайте таблицу EXAM MARKS так, чтобы не допускался ввод в таблицу двух записей об оценках одного студента по конкретным экзамену и предмету обучения и чтобы не допускалось проведение двух экзаменов по любым предметам в один день,

Задание3.

По Учебной базе данных создайте таблицу предметов обучения SUBJECT так, чтобы количество отводимых на предмет часов по умолчанию было равно 36, не допускались записи с отсутствующим количеством часов, поле SUBJ_ID являлось первичным ключом таблицы и значения семестров (поле SEMESTER) лежали в диапазоне от 1 до 12.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

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

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

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 5 Организация запросов SQL

Лабораторная работа №13

Создание и модификация таблиц БД. Выборка данных из БД. Модификация содержимого БД.

Цель: получение практических навыков по освоению операций создания запросов, представлений и хранимых процедур с помощью языка SQL

Выполнив работу, Вы будете:

уметь:

У2. Использовать язык запросов для программного извлечения сведений из баз данных

У09.1 Применять средства информационных технологий для решения профессиональных задач

У09.2 Использовать современное программное обеспечение.

Материальное обеспечение:

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

Задание 1:

Разработать запросы для базы данных Туризм с помощью конструктора в СУБД MS Access.

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. ***Параметрические запросы***

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. ***Итоговые запросы***

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. ***Перекрестные запросы***

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

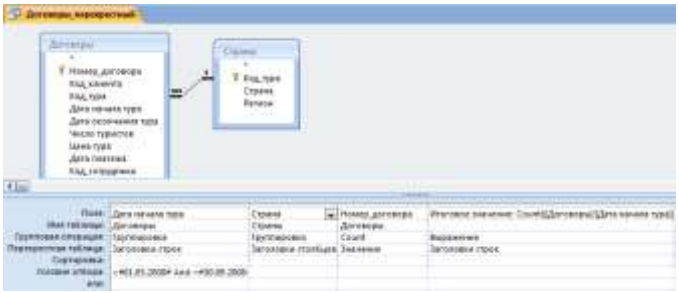
   Стоимость путевки = Sum(договоры![Цена тура] *договоры![Число туристов])

   Параметрические запросы
7. Сформируйте запрос для выборки всех туров по названию страны.
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 ***Код заказа*** >=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 MS access. 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 на создание

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 – Создание таблицы** – 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.

Запрос на добавление

3. Создайте путем копирования дубликат таблицы **Договоры** без данных, назвав ее **Договоры 2008 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.

Запрос на 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>]
[ORDER BY <список атрибутов>]
[GROUP BY <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 *
FROM STUDENT;

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 GROUP BY 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 GROUP BY 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 EXAM MARKS.
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

Разработать подзапросы 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

Разработать запросы, 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.

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 EXAM MARKS 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».

Задание №8

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 250 р./ч. 3. Работники отдела «Проектирование», разряд которых выше 10-го.
4	1. Поступления товара «Сухое молоко». 2. Товары для организации «Восход» в количестве от 1000 до 5000 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 5 шт. 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 от 1 до 5 т в Екатеринбург. 3. Перевозки груза, отправленные в Москву не позднее 10.09.2017.
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 в 2017 г. льготу «Инвалидность». 3. Абоненты, отключенные за неуплату во втором квартале 2017г.

Задание №9.

Разработать хранимые процедуры для базы данных на языке SQL.

Краткие теоретические сведения:

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

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

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

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

Для *выполнения хранимой процедуры* используется команда:

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

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

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

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

Из синтаксиса команды EXECUTE видно, что имена параметров могут быть опущены при вызове процедуры. Однако в этом случае пользователь должен указывать значения для параметров в том же порядке, в каком они перечислялись при создании процедуры.

Присвоить параметру значение по умолчанию, просто пропустив его при перечислении нельзя. Если же требуется опустить параметры, для которых определено значение по умолчанию, достаточно явного указания имен параметров при вызове хранимой процедуры. Более того, таким способом можно перечислять параметры и их значения в произвольном порядке.

Порядок выполнения работы:

Варианты заданий по созданию хранимых процедур с вычисляемым полем

Вариант	Имя таблицы	Вычисляемое поле
1	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

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

Варианты заданий по созданию хранимых процедур с параметрами

Вариант	Условие процедуры 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

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

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

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

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 5 Организация запросов SQL

Лабораторная работа №14

Обработка транзакций. Использование функций защиты для БД.

Цель: получение практических навыков работы с транзакциями и блокировками.

Выполнив работу, Вы будете:

уметь:

У09.1 Применять средства информационных технологий для решения профессиональных задач

У09.2 Использовать современное программное обеспечение.

Материальное обеспечение:

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

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 поллъзователем user2.
2. В базе данных sales создайте таблицу product:

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

3. Заполните таблицу значениями в соответствии с заданными.

```
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 (выполнить анализ)
5. DELETE FROM product WHERE id =4	6. SELECT * FROM product (выполнить анализ)
7. INSERT INTO product (name_pr, description, price, qty) VALUES ('Утюг паровой', 'BOSH', 5500.00, 23)	8. SELECT * FROM product (выполнить анализ)
	9. UPDATE product SET qty=qty+10 WHERE id=4 (выполнить анализ)
10. DELETE FROM product WHERE id =4 (выполнить анализ)	

	11. INSERT INTO product (name_pr, description, price, qty) VALUES ('Пароварка', 'BOSH', 3700.00, 14) (выполнить анализ)
12. ROLLBACK TRANSACTION trA	
	13. COMMIT TRANSACTION trB
14. LOCK TABLES product READ	
	15. SELECT * FROM product (выполнить анализ)
16. SELECT * FROM product (выполнить анализ)	
	17. UPDATE product SET qty=qty+20 WHERE id=4 (выполнить анализ)
18. UPDATE product SET qty=qty+20 WHERE id=4 (выполнить анализ)	
19. UNLOCK TABLES	
	20. (выполнить анализ)
21. LOCK TABLES product WRITE	
	22. SELECT * FROM product (выполнить анализ)
22. SELECT * FROM product (выполнить анализ)	
23. UNLOCK TABLES	
	24. (выполнить анализ)

5. Составить отчет

Анализ выполнения запросов при блокировках и транзакциях

№	Пользователь user1	Пользователь user2

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

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

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

Оценка «неудовлетворительно» ставится, если задание не выполнено.