ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ИРКУТСКОЙ ОБЛАСТИ «ЧЕРЕМХОВСКИЙ ГОРНОТЕХНИЧЕСКИЙ КОЛЛЕДЖ ИМ. М.И. ЩАДОВА»

Рассмотрено на
заседании ЦК
«04» 06 2020 г.
Протокол № 10
Председатель
_____ Т.В. Окладникова

УТВЕРЖДАЮ
Зам. директора по УР
_____ Н.А. Шаманова
«23» 06 2020 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для выполнения
практических работ студентов
по учебной дисциплине

ОП. 07 ОСНОВЫ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ

программы подготовки специалистов среднего звена

09.02.04 информационные системы (по отраслям)

Разработал преподаватель: Коровина Н.С.

2020г.

СОДЕРЖАНИЕ

1.ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Методические указания по выполнению практических (лабораторных) работ по учебной дисциплине **«Основы проектирования баз данных»** предназначены для студентов специальности **09.02.04 информационные системы (по отраслям),** составлены в соответствии с рабочей программой дисциплины «**Основы проектирования баз данных**» и направлены на достижение следующих целей:

– формирование у обучающихся умений проектировать реляционную базу данных;

– формирование у обучающихся умений использовать язык запросов для программного извлечения сведений из баз данных;

– приобретение обучающимися знаний основ теории баз данных;

– приобретение обучающимися знаний о принципах проектирования баз данных, обеспечение непротиворечивости и целостности данных;

– приобретение обучающимися знаний о языке запросов SQL;

– приобретение обучающимися знаний о особенностях реляционной модели и проектирование баз данных, изобразительных средствах, используемых в ER-моделировании;

– владение средства проектирования структур баз данных.

Методические указания являются частью учебно-методического комплекса по дисциплине **основы проектирования баз данных** и содержат задания, указания для выполнения практических (лабораторных) работ, теоретический минимум и т.п. Перед выполнением практической работы каждый студент обязан показать свою готовность к выполнению работы:

– пройти инструктаж по техники безопасности;

– ответить на теоретические вопросы преподавателя.

По окончанию работы студент оформляет отчет в тетради и защищает свою работу.

В результате выполнения полного объема практических работ студент должен у**меть:**

– проектировать реляционную базу данных;

– использовать язык запросов для программного извлечения сведений из баз данных формировать запросы для базы данных, состоящей из нескольких таблиц;

– формировать отчеты для базы данных, состоящей из нескольких таблиц;

– формировать формы для базы данных, состоящей из нескольких таблиц.

При проведении практических работ применяются следующие технологии и методы обучения:

1. проблемно-поисковых технологий
2. тестовые технологии
3. метод проектов

Правила выполнения практических работ:

1. Внимательно прослушайте инструктаж по технике безопасности, правила поведения в кабинете информатики.
2. Запомните порядок проведения практических работ, правила их оформления.
3. Изучите теоретические аспекты практической работы
4. Выполните задания практической работы.
5. Оформите отчет в тетради.

Требования к рабочему месту:

1. Количество ученических ПЭВМ, необходимых для оснащения кабинета ИВТ должно быть из расчета одной машины на одного обучающегося с учетом деления класса на две группы.
2. В состав кабинета ИВТ должна быть включена одна машина для учителя с соответствующим периферийным оборудованием.
3. Кабинет ИВТ должен быть оснащен диапроектором и экраном.

Критерии оценки:

Оценки «5» (отлично) заслуживает студент, обнаруживший при выполнении заданий всестороннее, систематическое и глубокое знание учебно - программного материала, учения свободно выполнять профессиональные задачи с всесторонним творческим подходом, обнаруживший познания с использованием основной и дополнительной литературы, рекомендованной программой, усвоивший взаимосвязь изучаемых и изученных дисциплин в их значении для приобретаемой специальности, проявивший творческие способности в понимании, изложении и использовании учебно - программного материала, проявивший высокий профессионализм, индивидуальность в решении поставленной перед собой задачи, проявивший неординарность при выполнении практических заданий.

Оценки «4» (хорошо) заслуживает студент, обнаруживший при выполнении заданий полное знание учебно - программного материала, успешно выполняющий профессиональную задачу или проблемную ситуацию, усвоивший основную литературу, рекомендованную в программе, показавший систематический характер знаний, умений и навыков при выполнении теоретических и практических заданий по дисциплине «Информатика».

Оценки «3» (удовлетворительно) заслуживает студент, обнаруживший при выполнении практических и теоретических заданий знания основного учебно-программного материала в объеме, необходимом для дальнейшей учебной и профессиональной деятельности, справляющийся с выполнением заданий, предусмотренных программой, допустивший погрешности в ответе при защите и выполнении теоретических и практических заданий, но обладающий необходимыми знаниями для их устранения под руководством преподавателя, проявивший какую-то долю творчества и индивидуальность в решении поставленных задач.

Оценки «2» (неудовлетворительно) заслуживает студент, обнаруживший при выполнении практических и теоретических заданий проблемы в знаниях основного учебного материала, допустивший основные принципиальные ошибки в выполнении задания или ситуативной задачи, которую он желал бы решить или предложить варианты решения, который не проявил творческого подхода, индивидуальности.

В соответствии с учебным планом программы подготовки специалистов среднего звена по специальности **09.02.04 Информационные системы (по отраслям)** и рабочей программой на практические (лабораторные) работы по дисциплине **«Основы проектирования баз данных»** отводится 50 часа.

2. ПЕРЕЧЕНЬ ПРАКТИЧЕСКИХ РАБОТ

№ п/п	Название практической работы (указать раздел программы, если это необходимо)	Количество часов
1.	Проектирование структуры базы данных.	**2**
2.	Создание однотабличной базы данных.	**2**
3.	Создание базы данных, состоящей из нескольких таблиц.	**4**
4.	Формирование запросов для базы данных, состоящей из нескольких таблиц.	**6**
5.	Формирование отчетов для базы данных, состоящей из нескольких таблиц.	**4**
6.	Формирование отчетов с применением языка VBA.	**4**
7.	Формирование SQL-запросов.	**6**
8.	Проектирование форм представлений и управление данными.	**2**
9.	Создание форм и отчетов для генерации выходных документов. Создание макросов	**2**
10.	Установка и нормализация отношений в базе данных (различные нормальные формы).	**4**
11.	Построение концептуальной модели базы данных.	**2**
12.	Создание логической и физической модели данных с помощью утилиты автоматизированного проектирования базы данных.	**2**
13.	Разработка серверной части базы данных в инструментальной оболочке.	**2**
14.	Разработка клиентской части базы данных в инструментальной оболочке.	**2**
15.	Построение запросов к базе данных на языке SQL (различных типов).	**2**
16.	Создание хранимых процедур в базах данных (различных типов).	**2**
17.	Создание триггеров в базах данных (различных типов).	**2**
	итого	**50**

3. СОДЕРЖАНИЕ ПРАКТИЧЕСКИХ РАБОТ

Практическая работа № 1
Проектирование структуры базы данных.

Цель: научиться проектировать структуру базы данных.

Задание 1. Изучить теоретический материал и составить структуру к базе данных студенты с указанием типа данных и ключевыми полями в каждой таблице.

Проектирование любого объекта осуществляется с:
а) определения его функционального назначения (зачем нужен, что и как делает проектируемый объект),
б) выявления логических связей (как осуществляет своё функциональное назначение проектируемый объект, какая информация и в какой последовательности обрабатывается),
в) выбора материальных средств реализации проектируемого объекта – функционально-технологический и технический аспект (носители, средства обработки данных и др.),
г) пространственного (территориального) размещения материальных средств реализации на выделенных или возможных для использования площадях,
д) формирования организационно-управленческой структуры проектируемого объекта (состав подразделений, полномочия и функциональные обязанности работников).

Современный подход к созданию информационных систем заключается не в создании системы на базе какого-либо интегрированного продукта, а в тщательном её проектировании и лишь потом реализации с помощью адекватных программных средств. Под проектированием любого объекта понимается процесс построения его образа, используемого затем для определённой (заданной) цели.

Проектирование информационных систем, включающих в себя базы данных, осуществляется на физическом и логическом уровнях.

Решение проблем *проектирования на физическом уровне* во многом зависит от используемой СУБД, зачастую автоматизировано и скрыто от пользователя. В ряде случаев пользователю предоставляется возможность настройки отдельных параметров системы, которая не составляет большой проблемы.

Логическое проектирование заключается в определении числа и структуры таблиц, формировании запросов к БД, определении типов отчетных документов, разработке алгоритмов обработки информации, создании форм для ввода и редактирования данных в базе и решении ряда других задач.

Решение задач логического проектирования БД в основном определяется спецификой задач предметной области. Наиболее важной является проблема структуризации данных.

При проектировании структур данных для автоматизированных систем можно выделить три основных подхода:

1. Сбор информации об объектах решаемой задачи в рамках одной таблицы (одного отношения) и последующая декомпозиция ее на несколько

взаимосвязанных таблиц на основе процедуры нормализации отношений.
2. Формулирование знаний о системе (определение типов исходных данных и их взаимосвязей) и требований к обработке данных. При этом с помощью CASE-систем можно получить готовые схемы БД или даже готовую прикладную информационную систему.
3. Структурирование информации для использования в информационной системе в процессе проведения системного анализа на основе совокупности правил и рекомендаций.

Для успешного проведения проектных работ рекомендуется выявить один или несколько прототипов проектируемого объекта, на их основе разработать некоторое количество возможных вариантов (их количество, как правило, в несколько раз больше числа выявленных прототипов). Например, для определения организационно-управленческой структуры автоматизируемой организации в качестве прототипа можно использовать существующую её структуру.

Затем из полученных вариантов следует отобрать альтернативные разновидности. С учётом местных условий и локальных ограничений сократить оставшиеся варианты, из которых выбрать наилучшие решения.

При проектировании рассматривается как внешняя, так и внутренняя среда объекта. В качестве макро среды организации выступают: пользователи (клиенты, заказчики) и их запросы; поставщики информации (информация об информации), исходящие и входящие информационные потоки, направляемые в организацию; внешние органы управления организацией. Применение унифицированных проектных решений базируется на типовых функциях и характеристиках организации.

В одном подразделении организации объектом может выступать отдельная операция или процесс (цикл), состоящий из нескольких операций.

Если группа подразделений является областью применения технических средств, то возникает потребность организации автоматизированной последовательности выполнения нескольких операций (процессов) конвейерным способом, например, применяемым в промышленности при изготовлении сложных агрегатов. Интеграция нескольких процессов создаёт предпосылки формирования комплексной системы для наиболее полного решения основных задач.

Исходными данными, промежуточными и конечными результатами процесса проектирования являются описания объекта, сделанные на специальном и (или) естественном языке.

Этапы проектирования баз данных

Этап 1. Определение сущностей.

Этап 2. Определение взаимосвязей между сущностями.

Этап 3. Задание первичных и альтернативных ключей, определение атрибутов сущностей.

Этап 4. Приведение модели к требуемому уровню нормальной формы.

Этап 5. Физическое описание модели.

Проектирование реляционных баз данных с использованием нормализации.
Приведение модели к требуемому уровню нормальной формы является основой построения реляционной БД.

В процессе нормализации элементы данных группируются в таблицы, представляющие объекты и их взаимосвязи. Нормализация основана на том, что определенный набор таблиц обладает лучшими свойствами при включении, модификации и удалении данных, чем все остальные наборы таблиц, с помощью которых могут быть представлены те же данные. Введение нормализации отношений при разработке информационной модели обеспечивает минимальный физический объем БД и ее максимальное быстродействие, что отражается на качестве функционирования ИС.

Нормализация информационной модели выполняется в несколько этапов. Данные, представленные в виде двумерной таблицы, являются первой нормальной формой реляционной модели данных.

Первый этап нормализации заключается в образовании двумерной таблицы, содержащей все необходимые атрибуты информационной модели, и в выделении ключевых атрибутов. При этом обычно получается внушительная таблица, содержащая разнородную информацию; наблюдаются аномалии включения, обновления и удаления данных, неимеющих никакого отношения к текущим действиям. Например, при включении в каталог продаж новой модели автомобиля сразу же придётся указать купившего ее клиента.

Ключевое поле – это одно или несколько полей, комбинация значений которых однозначно определяет каждую запись в таблице. Это уникальный идентификатор записей, используемый для поиска записей и объединения таблиц. Ключевое поле в таблице не может содержать пустые и повторяющиеся значения.

Ключевое поле в таблице используется для поиска записей. При ссылке на ключевое поле из другой таблицы оно называется полем внешнего ключа.

В качестве ключевого поля обычно рекомендуется использовать идентификационный или табельный номер. Так как оно является предметным словом и выбирается из текста для его индексирования, то ключевое слово является фрагментом текста, отражающим содержание документа.

Запись данных – это полный набор данных об определенном объекте; это любая строка в таблице данных. Физически в базах данных понятию запись данных соответствует понятие структуры.

Отношение считается заданым во второй нормальной форме, если оно является отношением в первой нормальной форме и каждый атрибут, не являющийся первичным атрибутом в этом отношении, полностью зависит от любого возможного ключа этого отношения.

Приведение отношений ко второй нормальной форме заключается в обеспечении полной функциональной зависимости всех атрибутов от ключа за счёт разбиения таблицы на несколько, в которых все имеющиеся атрибуты будут иметь полную функциональную зависимость от ключа этой таблицы. В процессе приведения модели ко второй нормальной форме в основном исключаются аномалии дублирования данных.

Отношение задано в третьей нормальной форме, если оно задано во второй нормальной форме и каждый атрибут этого отношения, не являющийся первичным, не транзитивно зависит от каждого возможного ключа этого отношения.

Транзитивная зависимость выявляет дублирование данных в одном отношении. Если A, B и C – три атрибута одного отношения и C зависит от B, а B от A, то говорят, что C транзитивно зависит от A.

Обоснование выбора СУБД

Обоснование выбора СУБД рассмотрим на примере СУБД Access, входящей в состав пакета прикладных программ (ППП) Microsoft Office. При выборе любой программы первоначально рассматриваются её возможности, среда применения, пользователи, их квалификация и т.п.

Широкое распространение имеет система управления базами данных Access. Она предназначена для работы на автономном компьютере или в локальной вычислительной сети под управлением операционной системы Microsoft Windows, поэтому все преимущества Windows (например, вырезать, копировать и вставлять данные из любого приложения Windows) могут использоваться в Access и наоборот.

Access привлекает простотой освоения и возможностью использования непрофессиональным программистом. Он имеет мощные средства подготовки отчетов из БД различных форматов. Поэтому его основное назначение – создание отчетов произвольной формы на основании различных данных и разработка некоммерческих приложений. Она является набором инструментальных средств, предназначенных для создания и эксплуатации информационных систем.

Access - это популярная настольная система управления базами данных. Её успех можно связать с великолепной рекламной кампанией, организованной Microsoft, и включением её в богатое окружение продуктов семейства Microsoft Office, с прекрасной реализацией продукта, рассчитанного как на начинающего, так и квалифицированного пользователя.

СУБД Access является набором инструментальных средств, предназначенных для создания и эксплуатации информационных систем, для управления базами данных.

К удобным для пользователей и разработчиков средствам Access относятся мастера и конструкторы таблиц, форм, запросов и отчётов. Она позволяет автоматизировать часто выполняемые операции (например, расчёт заработной платы, учёт материальных ценностей и т.п.), разрабатывать удобные формы ввода и просмотра данных, составлять сложные отчёты и др.

Таблица в Access является основным структурный объектом внутреннего строения БД. В неё включают записи определённого вида. Каждая запись таблицы содержит всю необходимую информацию об отдельном объекте – элементе БД. По многим причинам вводить все данные в одну таблицу нерационально, поэтому в Access предусмотрен механизм создания связанных между собой разных таблицы с различными видами данных. Таблицу Access можно связать с данными, хранящимися на другом компьютере или на сервере. В Access можно

использовать таблицу, созданную в СУБД Paradox или Dbase. Данные Access очень просто комбинировать и с данными из Excel и т.п.
С помощью средств Access можно выполнять следующие операции:
1. Проектировать базовые объекты ИС – двумерные таблицы с разными типами данных, включая поля объектов OLE. Например, прежде чем заполнять данными любую таблицу, надо создать её макет.
2. Устанавливать связями между таблицами с поддержкой целостности данных, каскадным обновлением полей и каскадным удалением записей.
3. Осуществлять ввод, хранение, просмотр, сортировку, модификацию и выборку данных из таблиц с использованием различных средств контроля информации, индексирования таблиц и аппарата алгебры логики (для фильтрации данных).
4. Создавать, модифицировать и использовать производные объекты ИС (формы, запросы и отчёты).

Совокупность связей между элементами, отражающими их взаимодействие называют структурой системы. Состояние системы в любой момент времени характеризуется её структурой.
Для проектирования БД необходимо, в первую очередь, представить себе предметную область (например, телефонная связь), её потребности, и возможности ПО для реализации данной задачи.
При разработке структуры любой таблицы реляционной БД изначально определяют названия полей, из которых она должна состоять, типы полей и их раз-меры. Каждому полю таблицы присваивается уникальное имя. Лучше, когда содержание или функция поля узнаются по нему.
Затем определяют типа данных, содержащихся в каждом поле. Один из типов данных должен быть присвоен каждому полю.

В таблице 1 представлены типы данных, используемые в Access и их описание. Тип данных в таблице характеризует вид хранящихся в поле данных.

Таблица 1. Типы данных Access.

Тип данных	Описание
Текстовый (по умолчанию)	текст или числа, не требующие проведения расчетов, например номера телефонов (до 255 знаков)
Числовой	числовые данные различных форматов, используемые для проведения расчетов
Дата/время	для хранения информации о дате и времени с 100 по 9999 год включительно
Денежный	денежные значения и числовые данные, используемые в математических расчетах, проводящихся с точностью до 15 знаков в целой и до 4 знаков в дробной части
Поле MEMO	для хранения комментариев; до 65535 символов
Счетчик	специальное числовое поле, в котором Access автоматически присваивает уникальный порядковый номер каждой записи. Значения полей типа счетчика обновлять нельзя

Логический	может иметь только одно из двух возможных значений (True/False, Да/Нет)
Поле объекта OLE	объект (например, электронная таблица Microsoft Excel, документ Microsoft Word, рисунок, звукозапись или другие данные в двоичном формате), связанный или внедренный в таблицу Access
Гиперссылка	строка, состоящая из букв и цифр и представляющая адрес гиперссылки. Адрес гиперссылки может состоять максимум из трех частей: текст, выводимый в поле или в элементе управления; путь к файлу (в формате пути UNC) или к странице (адрес URL). Чтобы вставить адрес гиперссылки в поле или в элемент управления, выполните команду Вставка, Гиперссылка
Мастер подстановок	создает поле, в котором предлагается выбор значений из списка или из поля со списком, содержащего набор постоянных значений или значений из другой таблицы. Это в дейст-вительности не тип поля, а способ хранения поля

Разработка информационно-логической модели реляционной БД начинается с рассмотрения необходимых для её создания информационных объектов. Рассмотрим вариант БД «Студенты». Выделим три объекта, не обладающие избыточностью: Студенты, Дисциплины и Преподаватели.

Представим состав реквизитов этих объектов в виде перечней:

- Студенты (номер зачетной книжки, фамилия, имя, отчество, дата рождения, стипендия, средний бал, увлечения).
- Дисциплины (код дисциплины, название дисциплины, преподаватель, объем часов),
- оценки (код оценки, наименование);
- увлечение (код увлечения наименование);
- стипендия (код стипендии, наименование, сумма)
- успеваемость (номер зачетной книжки, код дисциплины, код оценки).

Рассмотрим связь между объектами "Успеваекмость" и "Дисциплины". Студент изучает несколько дисциплин. Каждая дисциплина изучается множеством студентов – это тоже многозначная связь. Следовательно, связь между объектами "Успеваемость" и "Дисциплины" – «Многие-ко-многим» (M : N).

Практически любая реляционная БД (в том числе и в Access) создаётся из нескольких таблиц, на основе которых формируются формы и запросы. В таблицах Access форма является объектом, предназначенным для ввода данных. В форме Access можно разместить элементы управления и изображения.

Таблицы между собой связываются посредством общих полей, т.е. полей, одинаковых по форматам и, как правило, по названию, имеющихся в обеих таблицах. Такая организация данных позволяет уменьшить избыточность хранимых данных, упрощает их ввод и организацию запросов и отчётов. Каждая таблица включает в свой состав поле кода, используемого обычно как счётчик

(идентификатор) главного её параметра и, как правило, являющегося ключевым полем.
Обычно используются три вида связей между таблицами: «Один-ко-многим», «Многие-ко-многим» и «Один-к-одному».

Итог работы: тетрадь, защита работы.

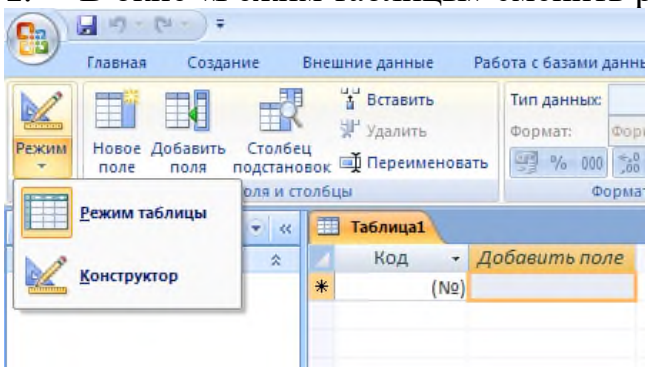
Практическая работа № 2
Создание однотабличной базы данных.

Цель: Изучение создание однотабличной базы данных в Microsoft Office Access 2007.

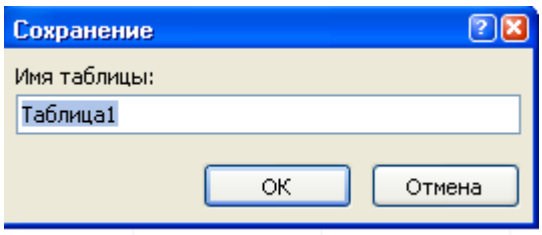
Задание 1. Запустить Microsoft Office Access 2007: пуск→все программы→ Microsoft Office 2007→ Microsoft Office Access 2007→ нова база данных →новая база данных→создать.

Задание 2.Создать новую таблицу «Студенты»

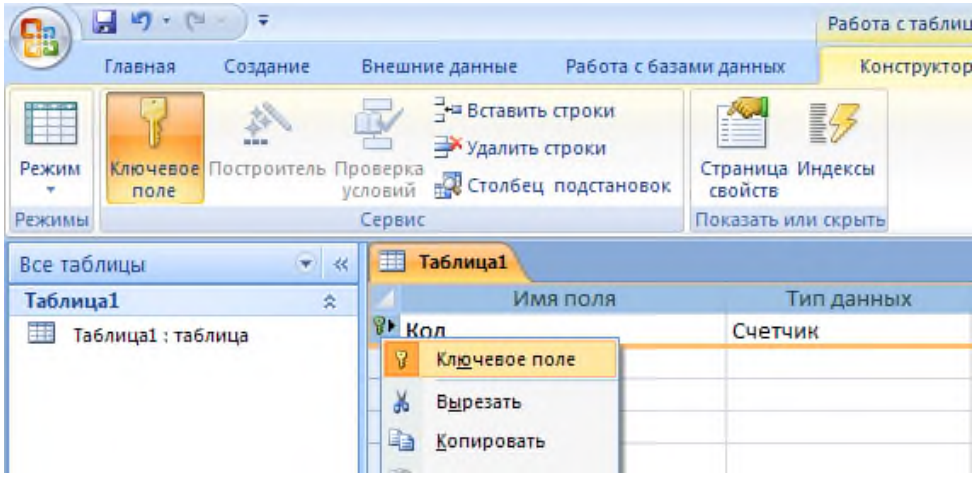
1. В окне «Режим таблицы» сменить режим на «конструктор».



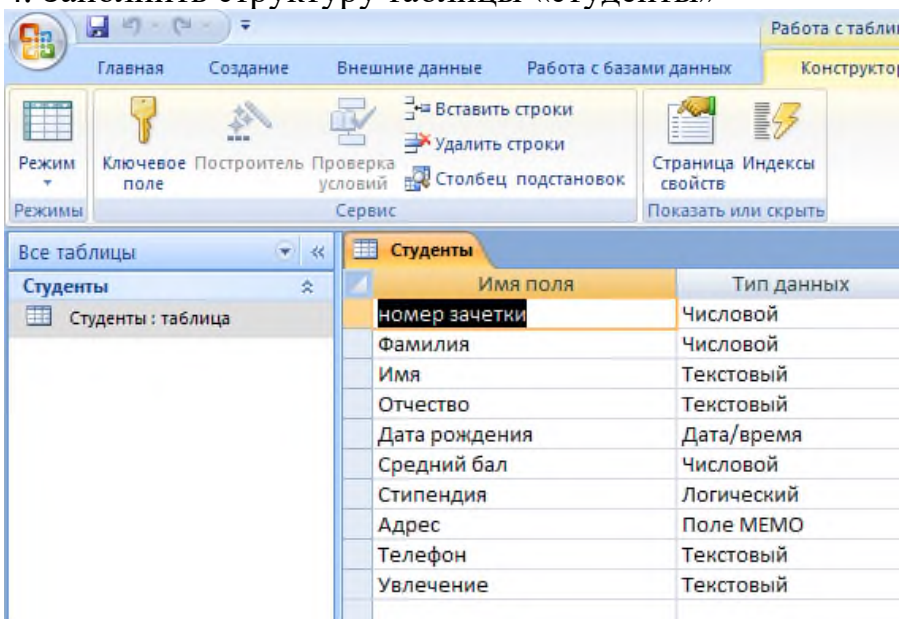
2. Сохранить таблицу по названием «Студенты».



3. Убрать ключевое поле нажав правой кнопкой на ключ и выбрав пункт «ключевое поле»



4. Заполнить структуру таблицы «студенты»



5.Перейти в режим таблицы и заполнить таблицу «студенты» на свою группу.

Студенты

номер зач	Фамилия	Имя	Отчество	День рождени	Средняя оц	стипендия	
0651002	Труфанова	Марина	Геннадьевна	22.01.1989	4	четверки и пя	г. И
0651003	Привольнев	Евгении	Сергеевич	16.02.1986	2,5	нет стипендии	г. И
0651004	Артамонова	Евгения	Александров	27.07.1988	4,25	четверки и пя	г. И
0651005	Толокнова	Анна	Андреевна	29.04.1989	3,25	нет стипендии	ул.
0651006	Юраго	Мария	Николаевна	01.03.1989	3	нет стипендии	г.И
0651007	Дорофеева	Маргарита	Сергеевна	30.07.1989	4,25	четверки и пя	г. И
0651008	Мартын	Андрей	Евгеньевич	01.09.1989	3,5	нет стипендии	с.Х
0651009	Чапайкин	Владислав	Николаевич	05.08.1989	4,25	только четвер	г. И
0651010	Рекинина	Наталья	Валерьевна	27.08.1988	3,25	нет стипендии	г. И
0651011	Зайнулина	Дарья	Сергеевна	19.02.1989	4,25	четверки и пя	г. И
0651012	Муранская	Юлия	Вкторовна	23.02.1990	2,25	нет стипендии	г. К
0651013	Безродных	Мария	Игоревна	12.05.1989	4,5	четверки и пя	г. И
0651014	Казаков	Александр	Михаилович	26.11.1988	3	нет стипендии	г. И
0651017	Трубачеев	Александр	Анатольевич	10.10.1990	4,25	только с одно	д. А
0651018	Морозов	Дмитрий	Викторович	12.02.1989	2,5	нет стипендии	г. Т
0651019	Мадаев	Дмитрии	Юрьевич	20.12.1987	2,5	нет стипендии	с. С
0651021	Березин	Сергей	Олегович	02.04.1989	3,25	нет стипендии	г. И
0651022	Поляков	Николай	Афанасьевич	13.04.1989	3	нет стипендии	г. И
0651024	Арнаутова	Татьяна	Михайловна	30.03.1989	4,5	четверки и пя	г. И

Итог работы: документ, защита работы.

Практическая работа № 3
Создание базы данных, состоящей из нескольких таблиц.

Цель: изучить возможности создание базы данных, состоящей из нескольких таблиц в Microsoft Office Access 2007.

Задание 1 . Создать Талицу «Дисциплины» и заполнить ее по итогам предыдущей экзаменационной сессии

Структура таблицы

Имя поля	Тип данных
Код Дисциплины	Числовой
Название	Текстовый
Ф.И.О. преподавателя	Текстовый
Количество часов	Числовой

Задание 2 . Создать Талицу «Оценки» и заполните её по пятибалльной шкале (исключая 1)

Структура таблицы

Имя поля	Тип данных
Код оценки	Числовой
Оценка	Текстовый

Задание 3.Создать Талицу «Стипендия» и заполните по видам академической стипендии.

Структура таблицы

Имя поля	Тип данных
Код стипендии	Числовой
Название	Текстовый

Задание 4.Создать Талицу «Увлечения» и заполнить.

Структура таблицы

Имя поля	Тип данных

Код увлечения	Числовой
увлечение	Текстовый

Итог работы: отчет, защита работы.

Практическая работа № 4
Формирование запросов для базы данных, состоящей из нескольких таблиц

Цель: изучить возможности формирование запросов для базы данных, состоящей из нескольких таблиц в Microsoft Office Access 2007.

Задание 1. Создание запроса в режиме конструктора
1. Войдите в режим конструктора запросов.→На вкладке Запросы выберите кнопку Создать или объект «Создание запроса в режиме конструктора».
В конструкторе запросы создаются вручную. Источником данных для запроса могут быть одна или несколько таблиц или запросов.
2. Добавьте таблицу Студенты как источник данных.
3. Изучите окно бланка запроса и инструментарий. Источники данных отображены в верхней части бланка запроса. Нижняя часть бланка предназначена:
• для задания полей, отображаемых в запросе (строка Поле и строка Имя таблицы);
• для задания способа упорядочения данных в запросе (строка Сортировка);
• для задания условий отбора (строки Условие отбора);
• для задания способа группировки данных в запросе;
• для задания вычисляемых выражений в полях запроса.
4. Включите в запрос все поля таблицы Студенты.
5. Включите сортировку по фамилии по убыванию и запустите запрос на выполнение (кнопка в панели инструментов).
6. При сохранении задайте имя «Сортировка на убывание». В запросе будут отображены все данные исходной таблицы.

Задание 2.Создать запрос «Студенты 2» с объединением столбцов Фамилия, Имя и Отчество в один столбец с именем ФИО. В Поле ввести выражение ФИО: [Фамилия]+" "+[Имя]+" "+[Отчество], в поле второго столбца выбрать Студенты.*.

Задание 3. Создать запрос «Адреса и телефоны», источник данных запрос «Студенты 2».

Задание 4. Создайте запросы на выборку. Задайте по очереди Условия отбора, указанные ниже, просмотрите результаты построенных запросов, и сохраните каждый из них под соответствующими именами. Итак, выберите:
• Студентов, фамилии которых начинаются с буквы, введенной с клавиатуры Like[введите букву]&((*));

- студентов, которые родились «с 04.04.1994 по 05.05.1995»(Between … and ….);
- Студентов, которые не проживают в городе Черемхово (Not …).

Запросы на выборку с логическими операциями.

Задание 5. Создать запрос «Стипендиаты», в который должны быть включены поля ФИО, Код стипендии (без вывода на экран), название, сумма. Источник данных запрос «Студенты 2» и таблица «Стипендия». В Условие отбора поля код стипендии ввести перечисление значении кода стипендий с помощью логический оператор OR .(# 1 Or 2 Or 3 Or 4)

Задание 6. Создайте запросы на выборку наложением условий на поля оценок за экзамены. Сохраните с указанными именами. Итак, выберите:

- учащихся только на отлично (имя «Отличники»).
- учащихся на хорошо и отлично (имя «Хорошисты»).
- учащихся, имеющих одну тройку (имя «Троечники»).

Задание 7. Запросы на выборку с параметром. Если вместо условия отбора в строке задать текстовое приглашение на ввод, заключенное в квадратные скобки, например [Введите фамилию], то при запуске запроса можно будет задать параметр. Параметр - это любое значение, по которому будут отыскиваться все записи с указанным значением поля (в примере это фамилия). В новых запросах выполните выборку по значению параметров:

- По имени студента. Условие отбора Like "*"+[Введите имя]+"*"
- По улице проживания (В запрос должны быть включены поля ФИО, адрес, телефон).

Обратите внимание, что при поиске с параметром используется точное соответствие значению параметра.

Вычисляемые поля в запросе:

Задание 8. Создать запрос «Средняя оценка по группе», Средняя оценка по группе: Avg(успеваемость![код оценки])

Итог работы: отчет, защита работы.

Практическая работа № 5
Формирование отчетов для базы данных, состоящей из нескольких таблиц

Цель: изучить возможности формирование отчетов для базы данных, состоящей из нескольких таблиц в Microsoft Office Access 2007

Задание 1. Создание отчетов с использованием мастера отчетов

Отчеты предназначены для отображения данных из таблиц и запросов базы данных в «бумажном» варианте. Отличаются от форм тем, что позволяют легко выполнить группировку данных и подведение итогов.

1.На вкладке **Отчеты** выберите кнопку **Создать**, далее **Мастер отчетов**.
2.Выберите источником данных таблицу **Студенты**.
2.Выберите для включения в запрос поля Номер Зачетки, Фамилия, Имя, Адрес, Телефон, в указанном порядке.
3.Не добавляя уровни группировки, отсортируйте поля по возрастанию имени, затем по возрастанию фамилии, затем имени.
4. Выберите макет Табличный, любой стиль, задайте имя Студенты.
Отчет открывается в режиме просмотра. Обратите внимание на содержание панели инструментов режима просмотра (она называется «Предварительный просмотр»).
5.Войдите в режим конструктора отчетов, ознакомьтесь с содержанием бланка отчета. Найдите области заголовков, примечаний, колонтитулов. Обратите внимание, что появилась панель элементов управления. Это означает, что инструменты конструктора отчетов такие же, как инструменты конструктора форм. Создание отчета аналогично созданию форм. В области заголовка располагается общий заголовок отчета, в верхнем колонтитуле – заголовки полей, они будут повторены на каждой странице отчета. В области нижнего колонтитула – текущая дата и номера страниц отчета, они тоже будут повторены на каждой странице. Посмотрите, какого типа эти элементы. Для заголовков используется элемент **Надпись**, для отображения данных элемент **Поле**. Элемент **Линия** отделяет область верхнего колонтитула.

Задание 2.Создайте группировку данных.

Для объединения записей в группы используется команда **Сортировка и группировка** из меню **Вид.** В отчете Студенты записи о студентах должны быть сгруппированы по имени. Используйте этот пункт меню, чтобы добавить в отчет область **Заголовок группы** для имени (значение Да), и задайте порядок сортировки по возрастанию. Перенесите в область **Заголовок отчета** надпись имя из верхнего колонтитула, и поле имя из области данных. Отделите тонкой линией сверху данные о различных группах.
Просмотрите результат в режиме просмотра.

Задание 3. Создайте нумерацию записей в отчетах.

Для нумерации записей в группе или по всему отчету, необходимо добавить в область данных отчета свободное поле, и задать значения свойства **Данные** равным формуле =1. Для свойства **Сумма с накоплением** нужно установить значение **Для группы**. Если нужна сквозная нумерация по всему отчету, свойство **Сумма с накоплением** должно иметь значение **Для всего**.
Просмотрите результат в режиме просмотра.

Задание 4. Создать отчеты «экзаменационная ведомость» по всем дисциплинам экзаменационной сессии.

Используйте **Мастер отчетов** для построения отчета. Выберите для включения в отчет поля ФИО, Номер зачетки и оценки за три экзамена (для каждого экзамена сделать отдельную ведомость). Задайте имя экзаменационная ведомость.

Задание 5. Подведение итогов в отчетах.

Для подведения итогов по группам используется команда **Сортировка и группировка** из меню Вид. Создать отчет о студентах уже сгруппированый по предметам. Добавьте в отчет область **Примечание группы** для предмета (значение Да). В область примечаний добавьте вычисляемые поля, чтобы вычислить средние значения оценок по предметам для каждой группы (функция Avg()).Добавьте надпись «Средние по предметам». Отделите тонкой линией снизу данные о различных группах.
Просмотрите результат в режиме просмотра.
Задание 6. Группировка данных и подведение итогов при создании отчетов с использованием мастера.
Мастер может автоматически подвести итоги при группировке данных. Создайте новый отчет на данных таблицы Успеваемость. На шаге Сортировка нажмите кнопку **Итоги**… и подведите итоги как средние оценки по экзаменам. Покажите **Только Итоги**. Сохраните отчет с именем **Итоги**.
Просмотрите результат в режиме просмотра.
Войдите в конструктор. Удалите поле, вычисляющее количество записей, текст надписи Avg замените на Среднее по предметам. Удалите поле Фамилия из области данных и надпись Фамилия из колонтитула. Запомните, как расположены элементы отчета.
Задание 7. Создание отчетов на все запросы базы данных «Студенты». Создание отчета в режиме конструктора.
Конструктор отчетов во многом схож с конструктором форм. Оба конструктора имеют одинаковую Панель элементов. Создание отчета аналогично созданию форм.
Создайте отчет на запросе **Средние по группам** в режиме конструктора. Названия предметов должны быть в верхнем колонтитуле. Поле и название группа должно быть в области заголовка группы. Средние оценки за экзамены должны быть в области данных.
Разместите в области верхнего колонтитула поле для вывода даты. Свойство **Данные** этого поля задайте с помощью встроенной функции Now**()** из списка функций. Свойство **Формат** этого поля должно быть форматом отображения даты.
В нижнем колонтитуле разместите номера страниц отчета, используя пункт меню **Вставка - Номера страниц.**

Итог работы: отчет, защита работы.

Практическая работа № 6
Формирование отчетов с применением языка VBA.

Цель: изучить формирование отчетов с применением языка VBA.

Задание 1. Изучить преобразование в модуль класса с помощью VBA любого макроса.

VBA представляет собой язык программирования, с помощью которого можно создавать в Access мощные приложения. VBA содержит сотни команд, которые позволяют выполнять намного более сложные операции, чем допускается макросами Access.

VBA можно использовать для интеграции Access с другими программами. Быстрый способ начать программировать на языке VBA — предварительно построить макрос Access, а затем преобразовать его в программу VBA.

При этом будет создан новый модуль, содержащий функцию VBA, выполняющую те же операции, что и макрос. Кроме того, сразу будет открыт редактор Visual Basic, где можно изменить данную процедуру. При работе в редакторе Visual Basic можно щелкнуть ключевые слова и нажать клавишу F1, чтобы открыть справку разработчика Access и узнать больше о каждом из этих ключевых слов. Преобразование макросов в программы VBA В Office Access 2007 можно автоматически преобразовать макросы в модули или модули класса VBA. Преобразование доступно в отношении макросов, прикрепленных к формам или отчетам независимо от того, представлены ли они как отдельные объекты или являются внедренными макросами. Можно также преобразовать глобальные макросы, не связанные с конкретной формой или отчетом. Преобразование макросов, связанных с формой или отчетом.

Следующая ниже процедура преобразовывает в VBA любые макросы, на которые есть ссылка в форме, отчете или их элементах управления (либо макросы, внедренные в эти объекты) и добавляет программу VBA в модуль класса формы или отчета. Модуль класса становится частью формы или отчета и переносится при перемещении или копировании.

1. В области переходов щелкните правой кнопкой мыши форму или отчет и выберите команду Конструктор.
2. На вкладке Инструменты для базы данных в группе Макрос щелкните Преобразовать макросы формы или Преобразовать макросы отчета.
3. В окне Преобразование макросов формы или Преобразование макросов отчета укажите, нужно ли добавлять программы обработки ошибок в создаваемые этими программами функции. Кроме того, если макросы имели примечания, укажите, должны ли они включаться в функции как комментарии. Нажмите кнопку Преобразовать, чтобы продолжить выполнение процедуры. Если для формы или отчета существует модуль класса, Access создаст такой модуль и добавит в него отдельную процедуру для каждого макроса, который был связан с формой или отчетом. Кроме того, Access изменит свойства события для формы или отчета таким образом, что впредь вместо макросов будут выполняться новые процедуры VBA.

4. Чтобы просмотреть и изменить программу VBA, выполните следующие действия:

1. Если форма или отчет открыты в режиме конструктора, а окно свойств не отображается, нажмите клавишу F4, чтобы открыть его.

2. На вкладке События окна свойств щелкните одно из полей свойств, которое содержит текст [Процедура события], а затем нажмите кнопку .
Чтобы просмотреть свойства событий для отдельного элемента управления, щелкните нужный элемент управления. Чтобы просмотреть свойства событий для всей формы или отчета, выберите в раскрывающемся списке, расположенном в верхней части окна свойств, параметр Форма или Отчет. Откроется редактор Visual Basic, где будет отображена процедура события в соответствующем модуле класса. Можно выполнить прокрутку вверх или вниз, чтобы просмотреть другие процедуры, включенные в тот же модуль класса.
Преобразование глобальных макросов
1. В области переходов щелкните имя макроса, который нужно преобразовать.
2. На вкладке Инструменты для базы данных в группе Макрос щелкните Преобразовать макросы.
3. В диалоговом окне Преобразование макроса укажите нужные параметры, а затем нажмите кнопку Преобразовать. Access преобразует макрос и откроет окно редактора Visual Basic.
4. Чтобы просмотреть и изменить программу VBA, выполните следующие действия:
1. Если в редакторе Visual Basic не отображается окно проекта, выберите в меню Представление команду Окно проекта.
2. Разверните дерево под узлом с именем базы данных, в которой ведется работа.
3. В узле Модули дважды щелкните модуль Преобразованный макрос-имя макроса. Нужный модуль будет открыт в редакторе Visual Basic. Прикрепление функции VBA к свойству события При преобразовании глобального макроса в VBA полученная программа VBA помещается в стандартный модуль.
В отличие от модуля класса, стандартный модуль не является частью формы или отчета. Скорее всего, потребуется связать эту функцию со свойством события формы, отчета или элемента управления, чтобы выполнить ее в нужное время. Для этого можно скопировать программу VBA в модуль класса, а затем связать ее со свойством события либо создать специальный вызов функции, находящейся в стандартном модуле, из свойства события с использованием следующей ниже процедуры.
1. В редакторе Visual Basic обратите внимание на имя функции. Например, если был преобразован макрос с именем МойМакрос, имя функции будет МойМакрос().
2. Закройте редактор Visual Basic.
3. В области переходов щелкните правой кнопкой мыши форму или отчет, которые нужно связать с функцией, и выберите команду Конструктор.
4. Щелкните элемент управления или раздел, которые нужно связать с функцией.
5. Если окно свойств еще не открыто, нажмите клавишу F4, чтобы открыть его.
6. На вкладке Событие в окне свойств щелкните поле свойства события, которое нужно связать с функцией.
7. В поле свойства введите знак равенства (=), а за ним имя функции: =МойМакрос(). Убедитесь, что не забыли поставить скобки.

8. Сохраните форму или отчет, нажав на панели быстрого доступа кнопку Сохранить .
9. В области переходов дважды щелкните форму или отчет, а затем проверьте, что программа выполняется, как ожидалось.

Итог работы: отчет, защита работы.

Практическая работа № 7
Формирование SQL-запросов.

Цель: изучить операторы, реализующие циклические алгоритмы.

Задание 1. Создайте базу данных «Товары» с таблицами «Заказы» (код заказа, сумма, дата приобретения, код заказа, код продавца), «Продавцы» (Код продавца, наименование, адрес, комиссионные), «Заказчики» (код заказчика, наименование, адрес, рейтинг, код продавца), «Продавцы2» (Код продавца, наименование, адрес, комиссионные) заполнить и нормализовать.

Задание 2. Создайте новый запрос для вывода сгруппированных данных в новую таблицу, перейдите в режим редактирования SQL и запустите запрос на выполнение:

```
SELECT КодПродавца, SUM(Сумма) AS [Общая сумма] INTO
[Продано продавцами]
FROM Заказы
GROUP BY КодПродавца;
```

Оператор	Назначение оператора
SELECT	Ключевое слово, которое сообщает базе данных, что эта команда - запрос. Все запросы начинаются этим словом, сопровождаемым пробелом.
КодПродавца, Сумма,	Это список столбцов из таблицы, которые выбираются запросом.
SUM(Сумма) AS [Общая сумма]	SUM – функция которая суммирует значения из столбца. В нашем случае из столбца «Сумма». AS [Общая сумма] – задает заголовок столбца в который будет выводиться сумма, в нашем случае «Общая сумма». []- квадратные скобки служат для определения текста, т.е. без квадратных скобок столбцу можем присвоить заголовок только «Общая» т.к. пробел будет считаться за переход к новой команде.
INTO [Продано продавцами]	INTO – обозначает что данные которые выведет запрос запишутся в новую таблицу «Продано продавцами», если
	данной таблицы нет он создаст ее автоматически.
FROM Заказы	Ключевое слово, подобно SELECT, которое должно быть представлено в каждом запросе. Оно сопровождается пробе- лом и затем именем таблицы используемой в качестве источника информации. В данном случае — это таблица «Заказы».
GROUP BY КодПродавца;	GROUP BY – сгруппирует выводимые данные, если в столбце, который вы укажете, будут повторяться записи.
;	Точка с запятой используется во всех интерактивных командах SQL, чтобы сообщать базе данных, что команда заполнена и готова выполниться

Задание 3.. Создание SQL – запроса для импорта данных из таблицы другой базы данных

Прежде чем выполнить запрос необходимо создать в своей папке копию текущей

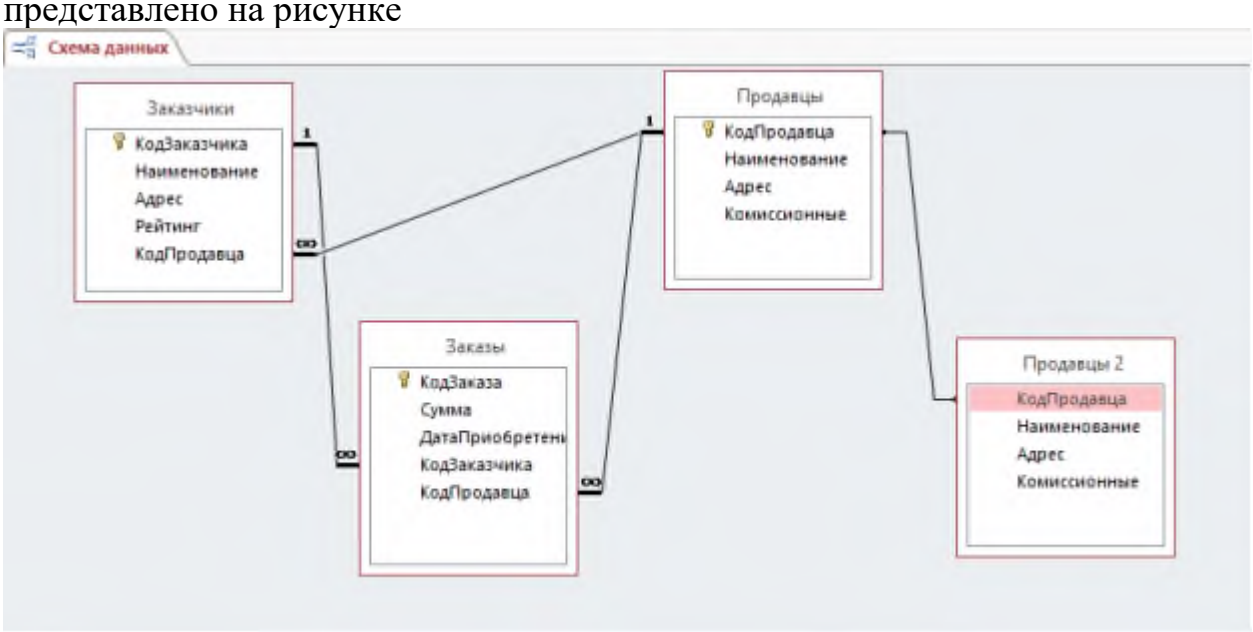
базы данных и присвойте ей новое имя «data». Создайте новый запрос в режиме редактирования SQL для импорта данных из таблицы другой базы данных и запустите его на выполнение:

SELECT * INTO [Продавцы 2]
FROM Продавцы IN 'C:\data.accdb';

Оператор	Назначение оператора
SELECT * INTO [Продавцы 2]	Выборка всех данных из таблицы в новую таблицу «Продавцы 2»
FROM Продавцы	Указываем, из какой таблицы выбрать данные
IN 'C:\data.accdb'	IN 'C:\data.accdb'– указываем из какого файла БД произвести импорт, в нашем случае «'C:\data.accdb'», обязательно нужно указать кавычки " так как в них записывается полный путь до файла в месте с расширением.
;	Точка с запятой используется во всех интерактивных командах SQL, чтобы сообщать базе данных, что команда заполнена и готова выполниться

Задание 4.. Создание SQL – запрос для удаления таблиц

Проверьте схему данных до выполнения запроса, она должна быть такой, как представлено на рисунке

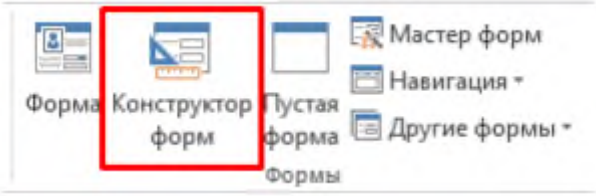


Создайте новый запрос в режиме редактирования SQL для удаления таблицы Продавцы 2 и запустите его на выполнение: DROP TABLE [Продавцы 2];

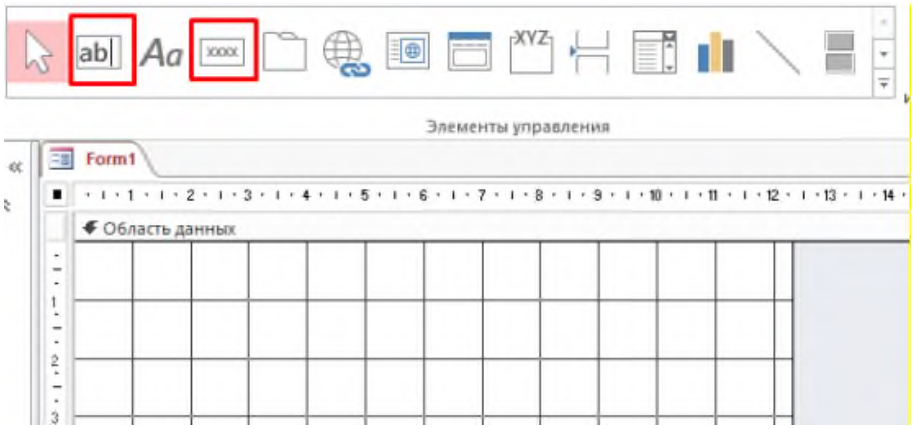
Оператор	Назначение оператора
DROP TABLE [Продавцы 2]	DROP TABLE – команда на удаление заданных таблиц. В нашем случае «Продавцы 2»

Задание 5. Создание формы импорта данных из другой таблицы, с использованием SQL-запроса и языка программирования VBA.

1.В используемой базе данных создайте новую, пустую форму при помощи вкладки меню Создать-Конструктор форм.



2. Добавьте на нее 3 элемента Поле и один элемент Кнопка



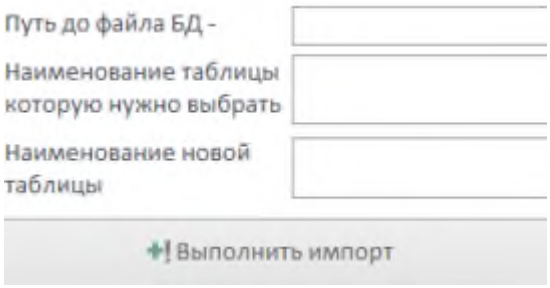
3.Расположите 3 элемента Поле и один элемент Кнопка

4.Для элемента Надпись1 зададим значение «Подпись» равной «Путь к файлу БД-»

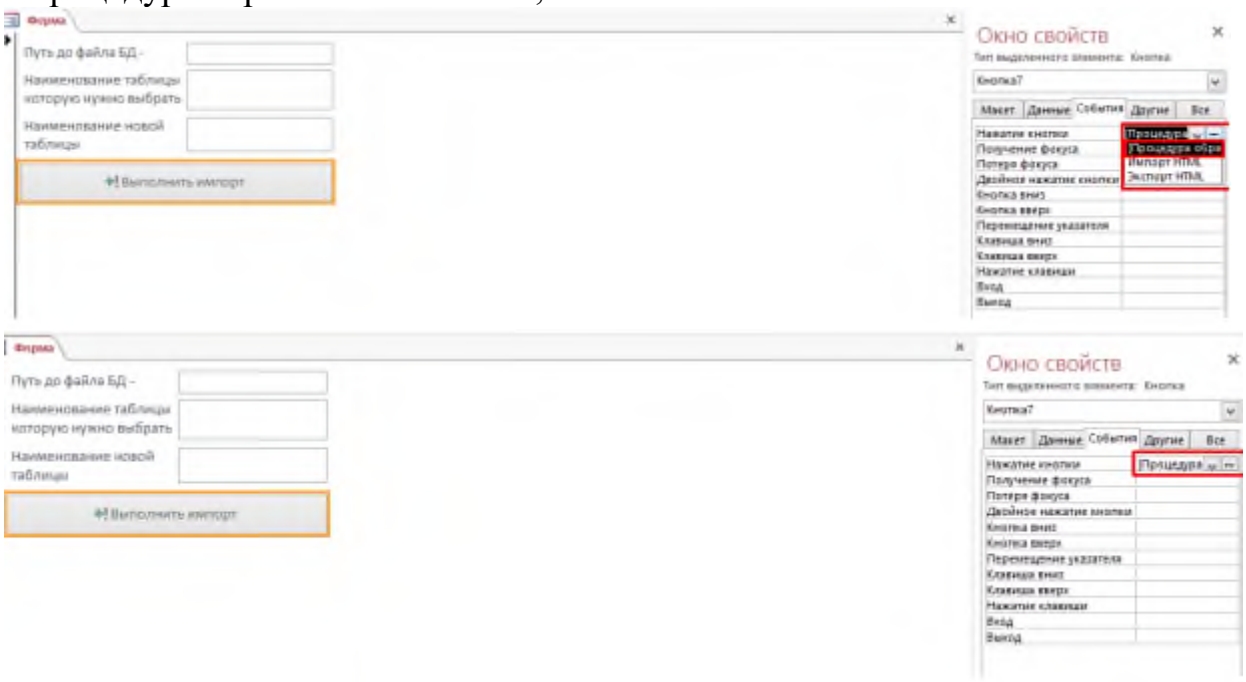
5.Для элемента Надпись2 зададим значение «Подпись» равной «Наименование таблицы которую нужно выбрать-»

6. Для элемента Надпись3 зададим значение «Подпись» равной «Наименование новой таблицы-»

7. Добавим в форму Кнопку, закроем мастер создания кнопки, присвоим значение свойства «Подпись» равной «Импортировать»

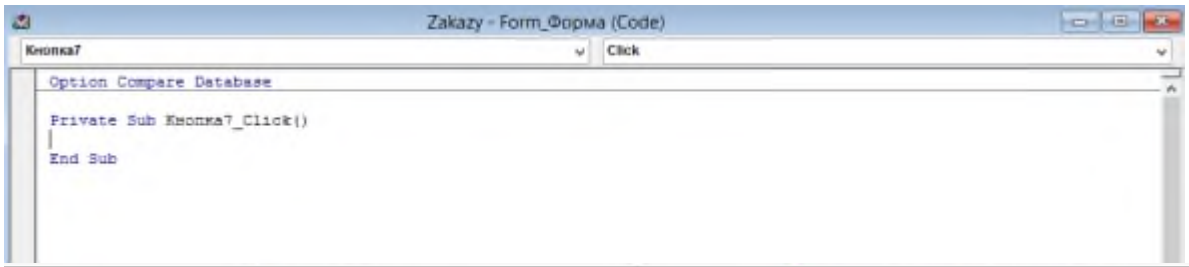


8. Для кнопки установим значение события «Нажатие кнопки» равной «Процедура обработки событий», нажимаем на ...



В результате база данных перешла в режим использования встроенного редактора кода VBA, при этом, откроется окно, представленное на рисунке выше, с кодом следующего содержания:

Private Sub Кнопка0_Click()
End Sub



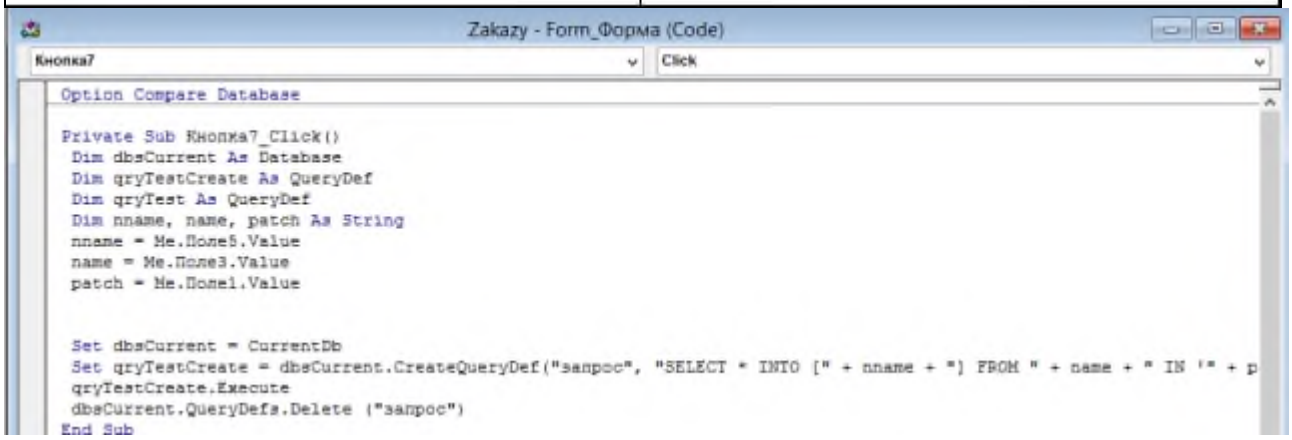
Оператор	Назначение оператора
Private Sub Кнопка0_Click()	Private Sub – объявление о начале выполнения процедуры по нажатию кнопки
End Sub	Коней процедуры

Впишем код для выполнения запроса SQL и в итоге получим:

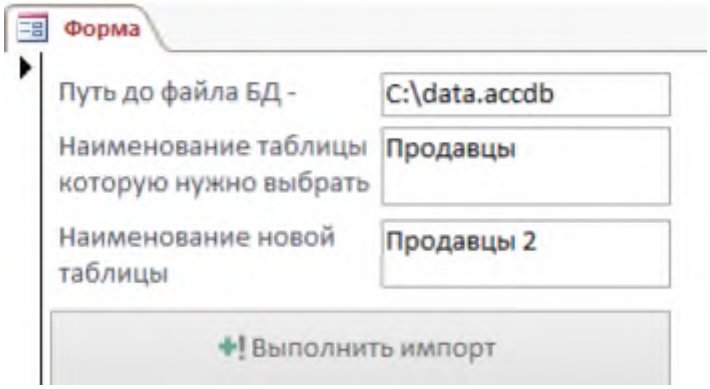
```
Private Sub Кнопка0_Click()
Dim dbsCurrent As Database
Dim qryTestCreate As QueryDef
Dim qryTest As QueryDef
 Dim nname, name, patch As String
nname = Me.Поле3.Value
name = Me.Поле2.Value
patch = Me.Поле1.Value
Set dbsCurrent = CurrentDb
Set qryTestCreate = dbsCurrent.CreateQueryDef("запрос", "SELECT * INTO [" + nname + "] FROM " + name + " IN '" + patch + "';")
qryTestCreate.Execute
dbsCurrent.QueryDefs.Delete ("запрос")
End Sub
```

Оператор	Назначение оператора
Dim dbsCurrent As Database	Объявление переменной dbsCurrent типа Database (База Данных)
Dim qryTestCreate As QueryDef	Объявление переменной qryTestCreate типа QueryDef (Запрос)
Dim qryTest As QueryDef	Объявление переменной qryTest типа QueryDef (Запрос)
Dim nname, name, patch As String	Объявление переменных nname, name, patch типа String (Строка)
nname = Me.Поле3.Value name = Me.Поле2.Value patch = Me.Поле1.Value	Присваиваем переменным значения полей которые введет пользователей
Set dbsCurrent = CurrentDb	Устанавливаем значение переменной dbsCurrent = CurrentDb т.е. что это переменная работает с нашей текущей БД

Set qryTestCreate = dbsCurrent.CreateQueryDef("запрос", "SELECT * INTO [" + nname + "] FROM " + name + " IN '" + patch + "';")	Устанавливаем значение переменной qryTestCreate равной созданному новому запросу (команда на создание нового запроса «dbsCurrent.CreateQueryDef») где "запрос" – Имя нашего запроса и далее идет текст запроса в который в нужные места вставляться значения наших переменных (вставка производиться благодаря операторам +)
qryTestCreate.Execute	Исполняем наш созданный запрос.
dbsCurrent.QueryDefs.Delete ("запрос")	Удаляем ранее созданный запрос.



Закрываем среду разработки, сохраняем созданную форму, запускаем ее с целью импортирования данных из другой таблицы. Внесем данные для импорта в созданную форму и выполним импорт нажав на кнопку импорта.



Итог работы: отчет, защита работы.

Практическая работа № 8
Проектирование форм представлений и управление данными.

Цель: изучить проектирование форм представлений и управление данными.

Задание. В режиме конструктора создайте таблицу в базе данных:

1. Создайте таблицу со структурой:

Имя поля	Тип поля	Свойства поля
Код	счетчик	без повторений
Фамилия	текстовый	
Имя	текстовый	
Отчество	текстовый	
Группа	текстовый	
Экзамен 1	числовой	подписями полей сделайте названия предметов, например, История, или другие, условие на значение Between 2 and 5, сообщение об ошибке «Ошибка ввода оценки».
Экзамен 2	числовой	подписями полей сделайте названия предметов, например, История, или другие, условие на значение Between 2 and 5, сообщение об ошибке «Ошибка ввода оценки».
Экзамен 3	числовой	подписями полей сделайте названия предметов, например, История, или другие, условие на значение Between 2 and 5, сообщение об ошибке «Ошибка ввода оценки».

Поле Код студента сделайте ключевым. Сохраните макет таблицы с именем «Успеваемость» и закройте ее.

2. Создайте форму с использованием мастера для ввода и редактирования данных в таблицу Сессия (формуляр с отображением всех полей из таблицы). Назовите ее Сессия. Для вызова мастера используйте кнопку «Создать» на панели инструментов. Обязательно укажите источник данных! Используя эту форму, заполните базу данными. Введите 8-10 строк для тех студентов, фамилии и номера групп которых есть и в первой таблице.

3. Создайте форму – диаграмму для графического отображения результатов сессии. Включите в диаграмму поля фамилии студентов и три оценки за экзамены. Выберите тип «Гистограмма». В макете формы данными оси абсцисс выберите фамилии, экзамены перенесите в область данных, где двойным щелчком мыши вызовите окно «вычисление итоговых значений» и поставьте «отсутствует». Сохраните форму с именем Диаграмма. Откройте форму, вызовите свойства рамки объекта, изменением свойства макета «Установка размеров» увеличьте размер рамки объекта (высоту и ширину).

4. Создайте форму с использованием конструктора для просмотра данных таблицы. В конструкторе формы создаются вручную. Основой для создания формы может быть таблица или запрос.

• В панели инструментов окна базы данных выберите кнопку Создать, далее выберите таблицу Сессия как источник данных для формы, далее выберите режим конструктора форм. Появляется бланк формы, содержащий пустую область данных. Ознакомьтесь с видом бланка формы и инструментами для создания

форм. Если отображена только область данных, в меню Вид добавьте Заголовок / примечания формы. Форма строится из элементов управления. Они расположены в панели инструментов. Элемент «Надпись» , предназначен для создания заголовков. Элемент поле предназначен для отображения данных из таблиц и выполнения вычислений.

- В область данных поместите все поля таблицы, используя буксировку поля из таблицы в область данных формы. При этом все поля данных формы связываются с данными соответствующих полей таблицы (свойство поля «Данные»). К каждому полю в области данных присоединяется надпись, значение которой совпадает с именем поля. Удалите ее.
- В области заголовков над полями поместите надписи с названиями столбцов таблицы (имена полей или надписи).

Поместите заголовок «Ведомость оценок за сессию» в область заголовка над названиями столбцов.

- Измените свойство всей формы целиком. Для этого используется щелчок правой кнопкой в верхнем левом углу формы: по умолчанию форма будет простая в один столбец. Свойство макета «Режим по умолчанию» нужно изменить на «Ленточная форма», тогда форма будет иметь вид таблицы записей.
- Сохраните форму с именем Ведомость. Снова войдите в режим конструктора и отредактируйте форму. Выровняйте все элементы управления по размеру и расположению. Используйте выделение группы элементов строки щелчком слева на вертикальной линейке, и выделение группы элементов столбца щелчком сверху на горизонтальной линейке. Ознакомьтесь со свойствами элементов управления «Надпись» и «Поле». Примените свойства макета: тип, цвет и ширина границы, оформление, цвета текста и фона.
- Убедитесь, что поля формы отображают данные из таблиц, для чего посмотрите свойства полей «Данные».

Вычисляемые поля форм. Такие поля не связаны с таблицей. Создаются с использованием элемента «Поле», взятого из панели элементов. Свойство «Данные» записывается через построитель выражений, вызываемый кнопкой .

5. В области данных добавьте вычисляемое поле Средний балл. Найдите среднюю оценку для каждого студента. В свойствах макета задайте формат поля «Фиксированный» и точность вывода на экран два знака после запятой.
6. В область примечаний формы добавьте итоговое вычисляемое поле Средний балл. Определите средние оценки по каждому из предметов с использованием функции Avg() (группа «Статистические») построителя выражений. Округлите до двух знаков после запятой. Добавьте поясняющую надпись «Средний балл по предметам».
7. Добавьте в форму вычисляемое поле Процент стипендии. Задайте в нем отображение данных через выбор по условию функцией Iif(): для среднего балла, равного 5, стипендия начисляется в размере 15%, для среднего балла, большего или равного 4 – в размере 10%, для среднего балла, большего или равного 3 – в размере базового размера, в остальных случаях она равна 0. Базовый размер стипендии примите равным 480 руб.

Итог работы: отчет, защита работы.

Практическая работа № 9
Создание форм и отчетов для генерации выходных документов. Создание макросов.

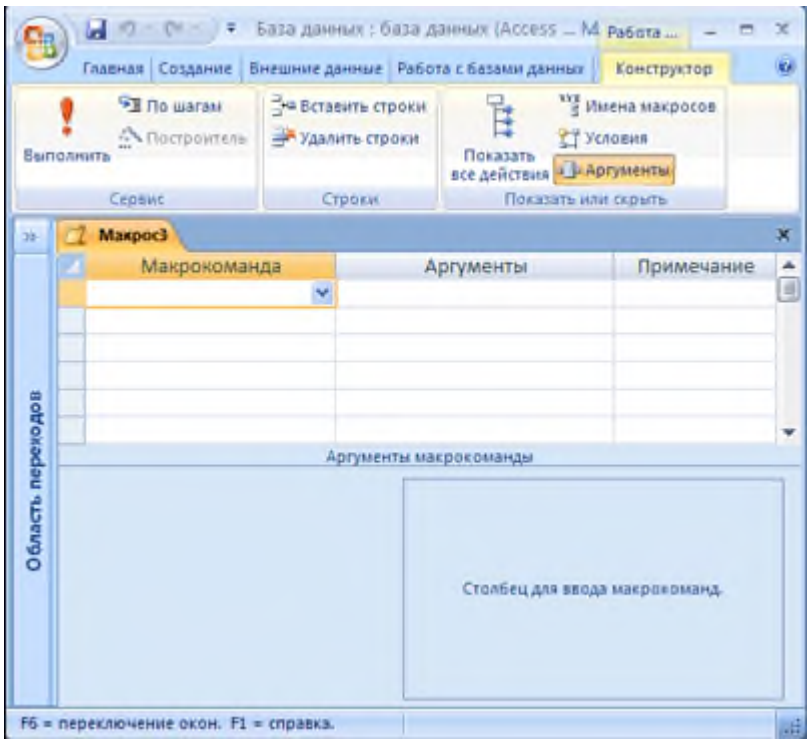
Цель: Изучить создание форм и отчетов для генерации выходных документов. Создание макросов.

Задание . Изучить теоретический материал и на его основе создать 3 макроса (изолированный макроса, группу макросов, внедренный макрос) к базе данных студенты.

Можно создать макрос, чтобы выполнять определенную последовательность макрокоманд, или группу макросов, чтобы выполнять связанный набор макрокоманд.
В Microsoft Office Access 2007 макросы могут содержаться в объектах макроса (иногда их называют изолированными макросами) или могут быть внедрены в свойства событий форм, отчетов, элементов управления. Внедренные макросы становятся частью объекта или элемента управления. Объекты макроса отображаются в области переходов в группе **Макросы**; внедренные макросы не отображаются.
Построитель макросов используется для создания и изменения макросов. Чтобы открыть построитель макросов:

- На вкладке **Создание** в группе **другое** нажмите кнопку **макрос**. Если эта команда недоступна, щелкните стрелку под **модуль** или **Модуль класса** кнопки и нажмите кнопку **макрос**. 

Откроется построитель макросов.



В окне построителя макросов создается список макрокоманд, которые требуется выполнить при запуске макроса. При первом открытии построителя макросов будут отображены столбцы **Макрокоманда**, **Аргументы** и **Примечание**.

В группе **Аргументы макрокоманды** с левой стороны при необходимости можно вводить и изменять аргументы для каждой макрокоманды. Поле описания с краткими сведениями о каждой макрокоманде или аргументе находится справа. Чтобы прочитать описание в поле, щелкните макрокоманду или аргумент.

Для создания, тестирования и запуска макросов можно использовать команды вкладки построителя макросов **Конструктор**.

В следующей таблице показаны команды, доступные на вкладке **Конструктор**.

Группа	**Команда**	**Описание**
Сервис	**Выполнить**	Выполнение макрокоманд, перечисленных в макросе.
	Пошаговое выполнение	Включение режима пошагового выполнения макроса. При запуске макроса в этом режиме происходит поочередное выполнение каждой макрокоманды. После завершения каждой макрокоманды отображается диалоговое окно **Пошаговое исполнение макроса**. Для перехода к следующей макрокоманде нажмите в этом диалоговом окне кнопку **Шаг**. Нажмите кнопку **Остановить все макросы**, чтобы остановить исполнение этого и всех остальных макросов. Нажмите кнопку **Далее**, чтобы выйти из пошагового режима и выполнить остальные макрокоманды без остановки.

Группа	Команда	Описание
	Построитель	Эта кнопка активируется при вводе аргумента макрокоманды, содержащего выражение. Чтобы открыть диалоговое окно **Построитель выражений** с целью создать выражение, нажмите кнопку **Построитель**.
Строки	**Вставить строки**	Вставка одной или нескольких пустых строк макрокоманды над выделенной строкой или строками.
	Удалить строки	Удаление выделенной строки или строк макрокоманды.
Показать или скрыть	**Показать все макрокоманды**	Отображение большего или меньшего количества макрокоманд в раскрывающемся списке **Макрокоманда**. • Чтобы отобразить полный список макрокоманд, выберите команду **Показать все макрокоманды**. Когда доступен полный список макрокоманд, кнопка **Показать все макрокоманды** выделена. При выборе макрокоманды из полного списка перед запуском макрокоманды может потребоваться предоставить базе данных явный статус доверия. • Чтобы перейти от полного списка макрокоманд к краткому, в котором находятся только макрокоманды, допустимые в базе данных без доверия, снимите выделение с кнопки **Показать все макрокоманды**. **СОВЕТ :** Если кнопка **Показать все макрокоманды** выделена, нажмите кнопку **Показать все макрокоманды**, чтобы снять выделение. Если кнопка **Показать все макрокоманды**не выделена, то доступен краткий список надежных макрокоманд.
	Имена макросов	Отображение или скрытие столбца **Имя макроса**. Имена макросов необходимы для того, чтобы отличать отдельные макросы друг от друга в группе макросов; в остальном имена для макросов необязательны. Дополнительные сведения см. в разделе Создание группы макросов.
	Условия	Отображение или скрытие столбца **Условие**. Этот столбец служит для ввода выражений,

Группа	Команда	Описание
		которые определяют условия для выполнения макрокоманды.
	Аргументы	Отображение или скрытие столбца **Аргументы**. В этом столбце отображаются аргументы для каждой макрокоманды, что упрощает просмотр макроса. Если столбец **Аргументы** не отображается, придется выбирать каждую макрокоманду и просматривать аргументы в разделе **Аргументы макрокоманды**. Ввод аргументов в столбце **Аргументы** невозможен.

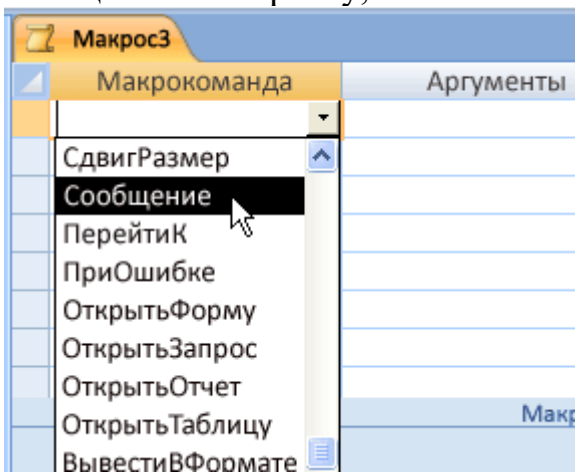
Создание изолированного макроса

1. На вкладке **Создание** в группе **другое** нажмите кнопку **макрос**. Если эта команда недоступна, щелкните стрелку под **модуль** или **Модуль класса** кнопки и нажмите кнопку **макрос**.

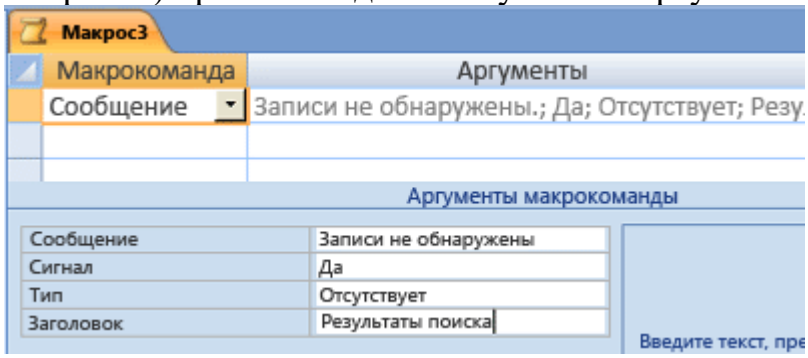
Откроется построитель макросов.

2. Добавление макрокоманды в макрос:

- В построителе макросов щелкните первую пустую ячейку в столбце **Макрокоманда**.
- Чтобы отобразить список макрокоманд, введите необходимую макрокоманду или щелкните стрелку, а затем выберите необходимую макрокоманду.



- В разделе **Аргументы макрокоманды** (в нижней части построителя макросов) при необходимости укажите аргументы для макрокоманды.



Примечание

- При вводе аргументов на панели **Аргументы макрокоманды** они отображаются в столбце списка макрокоманд **Аргументы**. Однако столбец **Аргументы** предназначен только для отображения; ввод аргументов в этот столбец невозможен.
- Чтобы увидеть краткое описание каждого аргумента, щелкните поле аргумента на панели **Аргументы макрокоманды** и прочитайте описание, находящееся в прилегающем поле.

Советы

- Для аргумента макрокоманды, значение которого является именем объекта базы данных, эти значения можно задавать путем перетаскивания объекта из области переходов в поле аргумента макрокоманды **Имя объекта**.
- Можно создать макрокоманду, перетащив объект базы данных из области переходов в любую пустую строку построителя макросов. При перетаскивании таблицы, запроса, формы, отчета или модуля в построитель макросов приложение Access добавляет макрокоманду, открывающую таблицу, запрос, форму или отчет. При перетаскивании макроса добавляется макрокоманда, запускающая макрос.
  - Можно дополнительно ввести текст описания для макрокоманды в столбец **Комментарий**.

3. Чтобы добавить дополнительные макрокоманды в макрос, перейдите на другую строку макрокоманды и повторите шаг 2.

При запуске макроса макрокоманды в приложении Access выполняются в порядке их очередности в списке.

Создание группы макросов

Группа макросов создается для объединения нескольких связанных макросов в один объект макроса.

1. На вкладке **Создание** в группе **другое** нажмите кнопку **макрос**. Если эта команда недоступна, щелкните стрелку под **модуль** или **Модуль класса** кнопки и нажмите кнопку **макрос**. 

Откроется построитель макросов.

2. На вкладке **Конструктор** в группе **Показать или скрыть** нажмите кнопку **Имена макросов**  , если она еще не была нажата.

В построителе макросов будет отображен столбец **Имя макроса**.

Примечание: Имена макросов необходимы в группах макросов для того, чтобы отличать отдельные макросы друг от друга. Имя макроса появляется в той же строке, где и первая макрокоманда. Столбец имени макроса остается пустым для любых следующих макрокоманд в макросе. Макрос заканчивается там, где появляется имя следующего макроса.

3. В столбце **Имя макроса** введите имя первого макроса из группы.
4. Введите макрокоманды, выполняемые в первом макросе:
   - В столбце **Макрокоманда** щелкните стрелку, чтобы раскрыть список макрокоманд.
   - Выберите имя макрокоманды.

- В разделе **Аргументы макрокоманды** при необходимости укажите аргументы для макрокоманды.

Чтобы увидеть краткое описание каждого аргумента, щелкните поле аргумента и прочитайте описание, находящееся справа от аргумента.

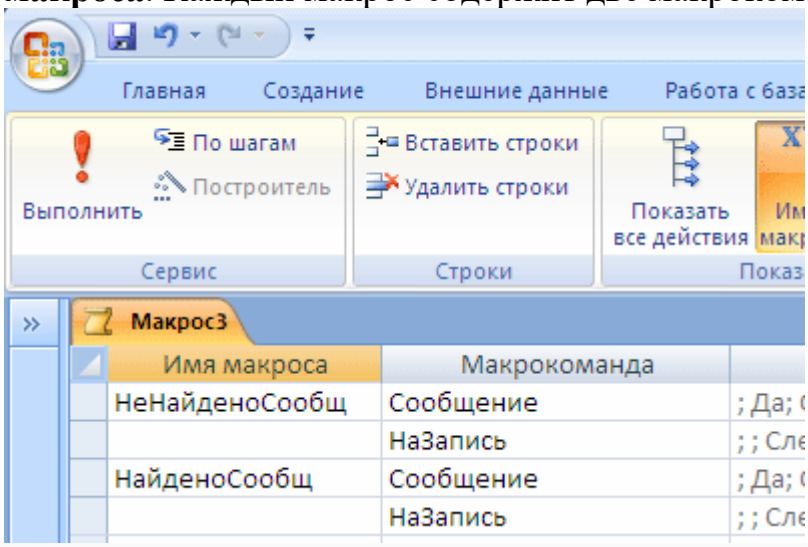
Советы

- Для аргумента макрокоманды, значение которого является названием объекта базы данных, эти значения можно задавать путем перетаскивания объекта из области переходов в поле аргумента макрокоманды **Имя объекта**.
- Можно создать макрокоманду, перетаскивая объект базы данных из области переходов в любую пустую строку построителя макросов. При перетаскивании в построитель макросов таблицы, запроса, формы, отчета или модуля Access добавляет макрокоманду, открывающую таблицу, запрос, форму или отчет. При перетаскивании макроса добавляется макрокоманда, запускающая макрос.

- Можно дополнительно ввести текст описания для макрокоманды.

5. Перейдите к следующей пустой строке и в столбце **Имя макроса** введите имя следующего макроса.
6. Введите макрокоманды, выполняемые в макросе.
7. Повторите шаги 5 и 6 для каждого макроса группы макросов.

На приведенной ниже иллюстрации показана небольшая группа макросов, содержащая два макроса. Имена этих макросов отображаются в столбце **Имя макроса**. Каждый макрос содержит две макрокоманды.



Примечание :

- Имя, указанное при сохранении группы макросов, будет служить именем группы макросов. В приведенном выше примере группа макроса называется «Macro3». Имя выводится в группе **Макрос** в области переходов. Для ссылок на макросы, входящие в группу макросов, используется следующий синтаксис:

ИмяГруппыМакросов.ИмяМакроса

Например, в приведенном выше примере выражение Macro3.FoundMsg является ссылкой на второй макрос группы.

- Если группа макросов запускается двойным щелчком в области переходов или нажатием кнопки **Выполнить** в группе **Сервис** вкладки **Конструктор**, в

приложении Access выполняется только первый макрос группы и процесс останавливается по достижении имени второго макроса.

Создание внедренного макроса

Внедренные макросы отличаются от изолированных макросов тем, что они хранятся в свойствах событий форм, отчетов, элементов управления. Они не отображаются в виде объектов в группе **Макрос** в области переходов. Это упрощает управление базой данных, поскольку не нужно следить за тем, в каком отдельном объекте макроса содержится макрос для формы или отчета. Внедренные макросы также включаются в состав формы или отчета при их копировании, импорте или экспорте.

Например, чтобы не отображать отчет, в котором отсутствуют данные, можно включить макрос в свойство события **Отсутствие данных**. Чтобы отобразить сообщение, можно использовать макрокоманду **MsgBox**, а затем чтобы отменить просмотр отчета вместо отображения пустой страницы, применить макрокоманду **ОтменитьСобытие**.

1. В области переходов щелкните правой кнопкой мыши форму или отчет, содержащий макрос, а затем выберите команду **Конструктор**  или **Режим макета**  .
2. Если окно свойств не открыто, нажмите клавишу F4.
3. Выберите элемент управления или раздел, содержащий свойства события, в который нужно встроить макрос. В верхней части окна свойств из выпадающего списка в разделе **Тип выбора**можно выбрать элемент управления или раздел, а также форму или отчет целиком.
4. В окне свойств перейдите на вкладку **События**.
5. Выберите свойство события, в которое нужно встроить макрос, затем нажмите кнопку  .
6. В диалоговом окне **Построитель** выделите пункт **Макросы** и затем нажмите кнопку **ОК**.
7. В построителе макросов щелкните первую строку столбца **Макрокоманда**.
8. В раскрывающемся списке **Макрокоманда** выберите нужный макрос.
9. В группе **Аргументы макрокоманды** введите необходимые аргументы.
10. Если требуется добавить другую макрокоманду, в столбце **Макрокоманда** выделите следующую строку и повторите шаги 8 и 9.
11. Закончив создание макроса, нажмите кнопку **Сохранить**, а затем нажмите кнопку **Закрыть**.

Макрос запускается каждый раз при возникновении свойства события.

Редактирование макроса

- **Вставка строки макрокоманды** Щелкните правой кнопкой мыши строку макрокоманды, над которой требуется вставить новую строку, а затем выберите команду **Вставить строки**  .
- **Удаление строки макрокоманды** Щелкните правой кнопкой мыши строку, которую требуется удалить, а затем выберите команду **Удалить строки**  .

- **Перемещение строки макрокоманды** Выделите строку макрокоманды, щелкнув заголовок строки слева от макрокоманды, а затем перетащите его в новое положение.

Чтобы вставить, удалить или переместить нескольких строк, следует сначала выделить группу строк, а затем выполнить требуемое действие. Чтобы выделить группу рядов, щелкните заголовок первой выделяемой строки, нажмите клавишу SHIFT, а затем щелкните заголовок последней выделяемой строки. (Заголовок строки — это затененное поле справа от строки макрокоманды.)

Выделить несколько строк также можно, расположив курсор над заголовком первой выделяемой строки, а затем нажав кнопку мыши и перетаскивая курсор вверх или вниз для выделения других строк.

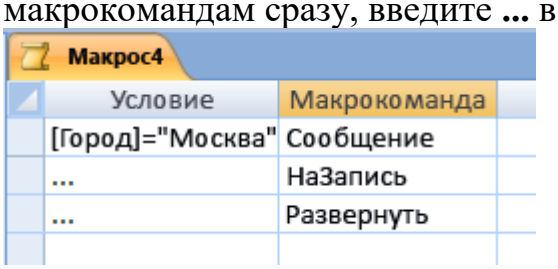
Использование условий для контроля за действиями макроса

В условие допускается использование любых выражений, возвращающих значения Истина/Ложь или Да/Нет. Макрокоманда будет выполняться, если выражение имеет значение Истина (Да).

Чтобы ввести условие для макрокоманды, сначала следует отобразить столбец **Условие** в построителе макросов:

- На вкладке **Конструктор** в группе **Показать или скрыть** нажмите кнопку **Условия**  .

Введите выражение в столбец **Условие**. Нельзя начинать выражение знаком равенства (=). Для того, чтобы условие применялось к нескольким макрокомандам сразу, введите **...** в каждой последующей строке. Например:



СОВЕТ : Для того чтобы временно пропустить макрокоманду, введите в качестве условия значение **False**. Такой прием часто используют при отладке макросов.

Примеры условных выражений в макросах

Выражение	**Условие выполнения макрокоманды**
[Город]="Париж"	Поле «Город» в форме, из которой запускается макрос, имеет значение «Париж».
DCount("[КодЗаказа]", "Заказы")>35	Количество записей в поле «КодЗаказа» таблицы «Заказы» превышает 35.
DCount("*", "Заказано", "[КодЗаказа]=Forms![КодЗаказа]![КодЗаказа]")>3	Имеется более трех записей в таблице «Заказано», у которых значение поля «КодЗаказа» совпадает со значением поля «КодЗаказа» в форме «ПримерФормы».
[ДатаИсполнения] Between #2-фев-	Значение поля «ДатаИсполнения» в

Выражение	**Условие выполнения макрокоманды**
2007# And #2-мар-2007#	форме, из которой запускается макрос, попадает в интервал со 2 февраля 2007 по 2 марта 2007 г.
Forms![Товары]![На складе]<5	Значение поля «На складе» в форме «Товары» меньше 5.
IsNull([Имя])	"Имя" в форме, из которого выполняется макрос значение **Null** (не имеет значения). Эквивалентна **["имя"**— это выражение**] — Null**.
[Страна]="Литва" And Forms![Сумма продаж]![Объем заказов]>100	Поле «Страна» в форме, из которой запускается макрос, содержит значение «Литва» и значение поля «Объем заказов» в форме «Сумма продаж» превышает 100.
[Страна] In ("Латвия", "Литва", "Эстония") And Len([Индекс])<>5	Поле «Страна» в форме, из которой запускается макрос, имеет значение «Франция», «Италия» или «Испания», и почтовый индекс содержит не 5 символов.
MsgBox("Подтвердить изменения?",1)=1	Нажата кнопка **ОК** в диалоговом окне, в котором функция **MsgBox**отображает текст **Подтвердить изменения?** Если в диалоговом окне нажата кнопка **Отмена**, эта макрокоманда будет пропущена.
[ВремПеременные]![Переменная1]=43	Значение временную MyVar переменной (созданные с помощью SetTempVarмакрокоманды) равно 43.
[ОшибкаМакроса]<>0	Значение свойства **Number** объект **MacroError** не равно 0, это означает, что в макросе произошла ошибка. Это условие можно использовать вместе с Устранитьошибкумакроса и OnError ма крокоманд для управления, что происходит при возникновении ошибки.

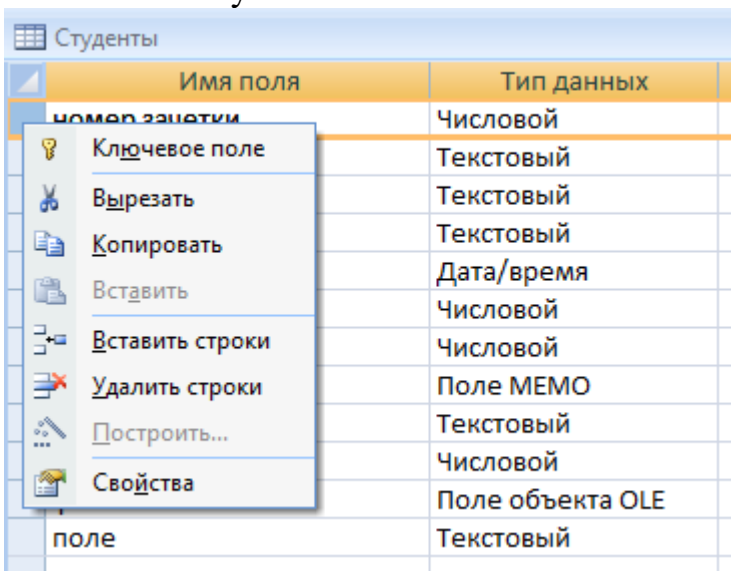
Итог работы: отчет, защита работы.

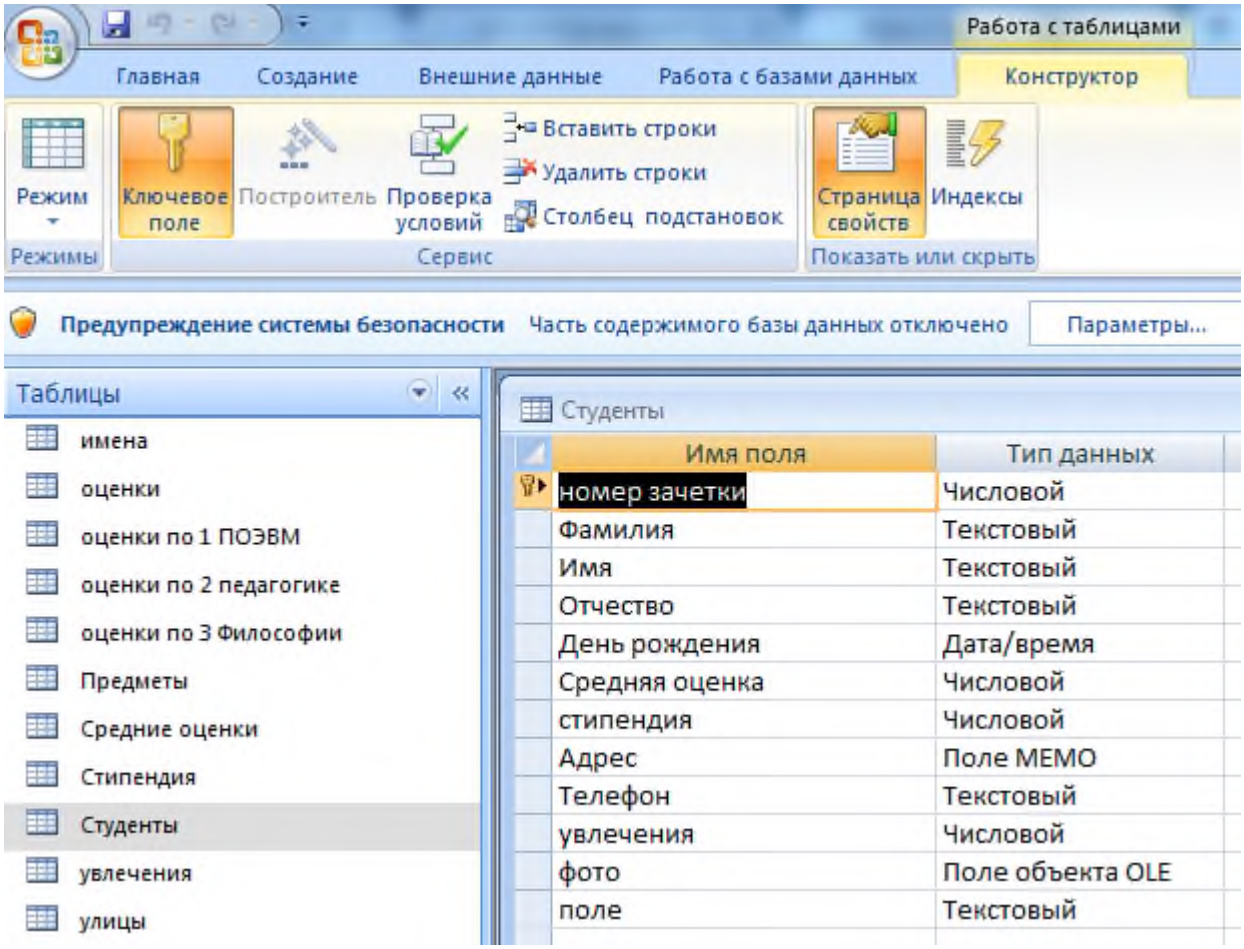
Практическая работа № 10

Установка и нормализация отношений в базе данных (различные нормальные формы).

Цель: Изучить установку и нормализацию отношений в базе данных (различные нормальные формы).

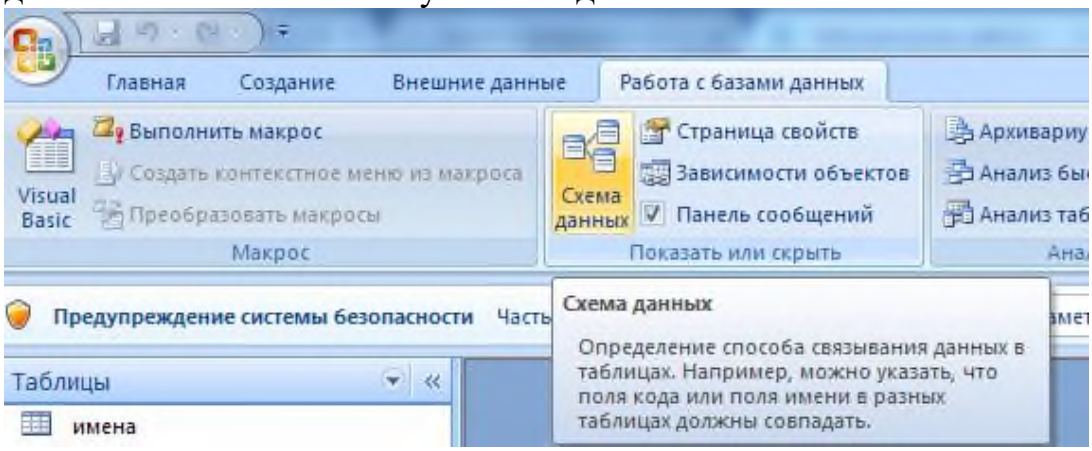
Задание 1 . Создать ключи для всех таблиц базы данных "Студенты"
Отдельные таблицы, содержащие информацию по определенной теме, необходимо связать в единую структуру базы данных. Для связывания таблиц следует задать ключевые поля. Ключ состоит из одного или нескольких полей, значения которых однозначно 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).
1. Откройте таблицу Студенты в режиме Конструктора.
2. 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.



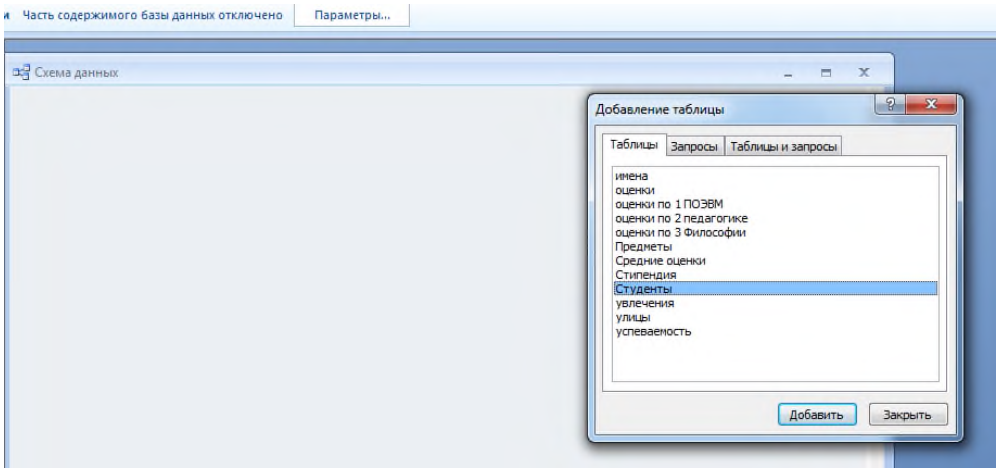


Задание 2 . Создать «Схему данных» определяющую все связь между таблицами базы данных «Студенты».

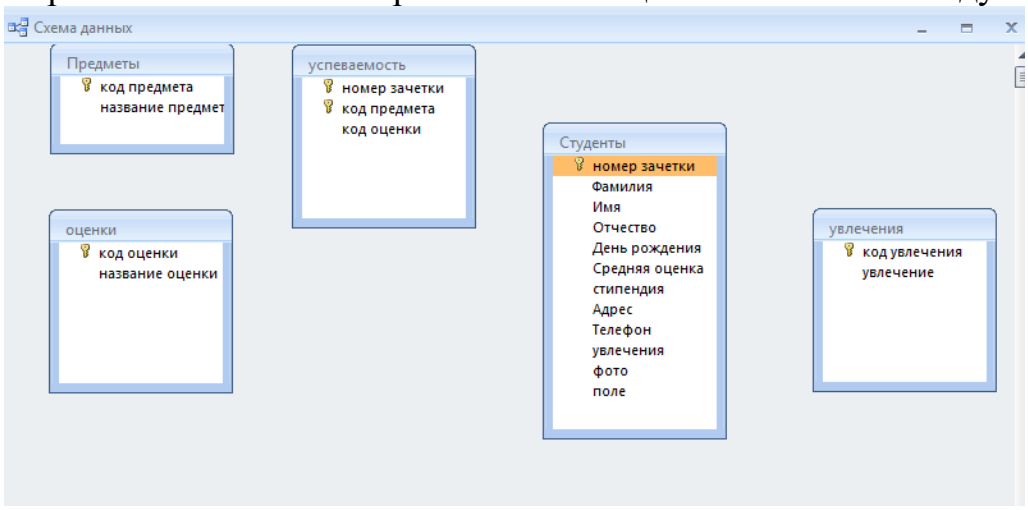
1.Создание связей Для того чтобы, создать или изменить связи между таблицами, нужно сначала закрыть все таблицы. Затем на панели инструментов «Работа с данными» нажмите кнопку «Схема данных».



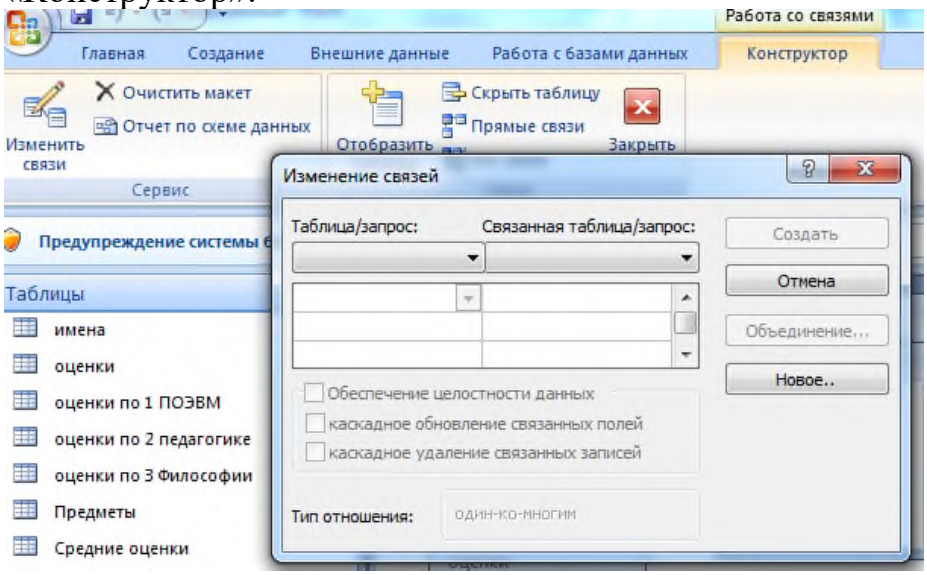
Добавьте на схему все существующие таблицы и закройте окно добавления объектов.



В рабочей области отобразятся пока еще не связанные между собой таблицы.

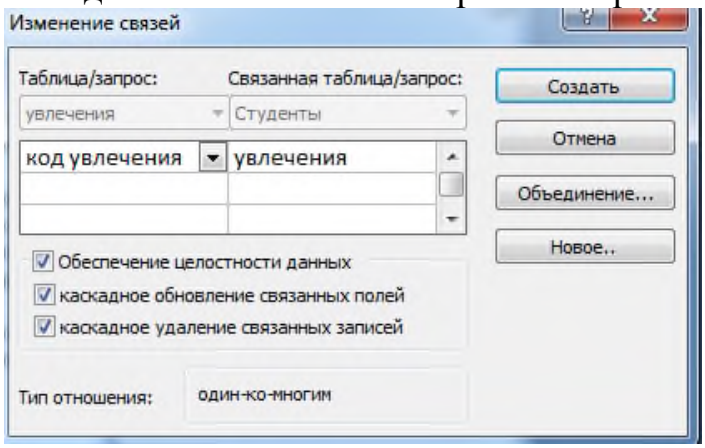


Для изменения и создания связей нажмите кнопку «Изменить связи» на панели «Конструктор».

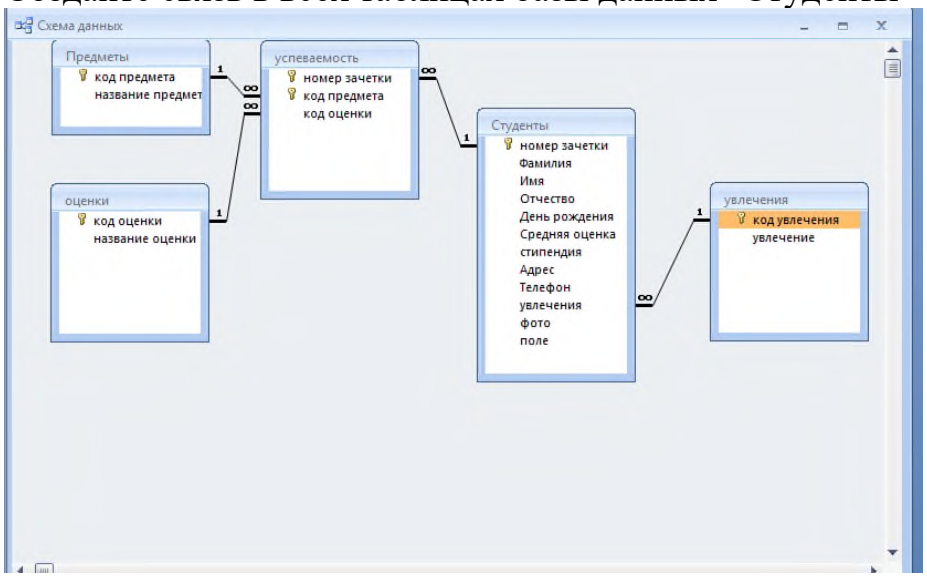


В появившемся окне нажмите кнопку «Новое». Откроется окно для создания связи. В качестве левой таблицы (сторона «один» связи «один ко многим») выберите таблицу «Студенты» и столбец «Увлечение». Для правой таблицы выберите таблицу «Увлечение» и столбец «Код увлечения». Нажмите кнопку

«ОК». В открывшемся окне оставьте галочку «Обеспечение целостности данных» и создайте связь. Связь отобразится в рабочей области.



Создайте связь в всех таблицах базы данных "Студенты"



Задание 3. Определить в структуре таблицы «Студенты» для поля «код увлечения» подстановку поля со списком со значениями таблицы «Увлечения», а для поля «Код Стипендии» подстановку поля со списками со значением из таблицы «Стипендия».

Задание 4. Определить в структуре таблицы «Успеваемость» для поля «Код дисциплины» подстановку поля со списком со значениями таблицы «Дисциплины», а для поля «Код оценки» подстановку поля со списками со значением из таблицы «Оценки».

Итог работы: отчет, защита работы.

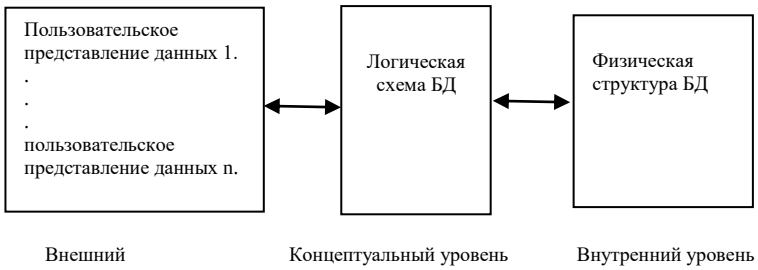
Практическая работа № 11
Построение концептуальной модели базы данных.

Цель: изучить построение концептуальной модели базы данных.

Задание 1. Проработать все описанные упражнения самостоятельно, руководствуясь методическими указаниями.

Трехуровневая архитектура БД.

Различие между логическим и физическим представлением данных было официально признано в 1978 году. Тогда была предложена обобщенная структура систем баз данных. Эта структура получила название *трехуровневой архитектуры БД,* которая состоит из следующих уровней: концептуальный, внешний, внутренний.



Внешний уровень (модель)- составляют пользовательские представления данных. К БД обращаются много пользователей, но не одному пользователю не нужна вся БД в целом, а только её часть. Представления могут пересекаться.

Каждое представление дает ориентированное на пользователя описание элементов данных. Объекты, атрибуты объектов, потоки и их направление, регламентные операции.

Отсюда можно вывести концептуальную логическую схему (модель) БД.

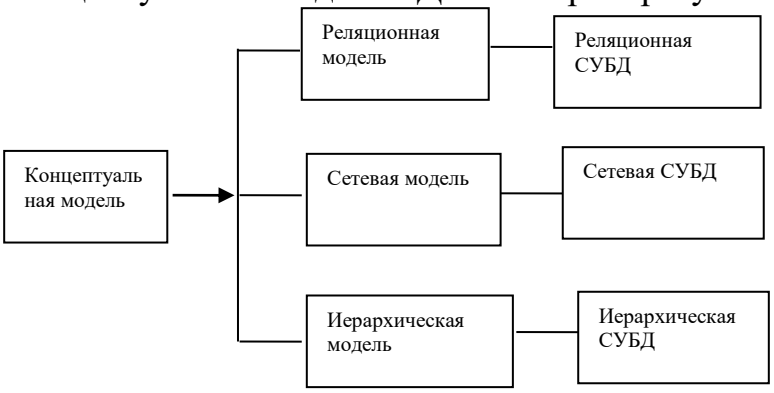
Концептуальный уровень –определяет логическую схему БД.

Внутренний уровень(модель) – обеспечивает физический взгляд на БД. Ни один пользователь не касается этого уровня. СУБД и ОС преобразовывают адреса и указатели в соответствующие имена и отношения. Такой перевод стоит большой системной поддержке (как и по сложности, так и по цене).

Концептуальная модель данных.

С начала 70-х годов было предложено несколько концептуальных моделей данных. Например, двоичная семантическая модель данных, категорно-относительная модель Чена, функциональная модель данных, семантическая модель данных Хаммера и Мак Леода.

Концептуальная модель БД легко преобразуется в реальные модели.



Проектировщик БД, выбирая для своей системы конкретную СУБД, прежде всего, сталкивается с задачей выбора наиболее подходящей модели для своей прикладной области. При этом оцениваются следующие свойства модели данных СУБД:

1. сложность и трудоемкость программ для манипулирования данных;
2. сложность модели для изучения пользователем;
3. наглядность структуры данных;
4. модель должна иметь минимальное число базисных структур.

Создание концептуальную модель БД рассмотрим на примере модели ”сущность-связь”. Ее отличает относительная простота, приме-нение естественного языка, легкость понимания.

Модель “сущность – связь”.

Это неформальная модель предметной области, которая используется на этапе концептуального проектирования базы данных. Основное назначение модели – описание предметной области и представление информации для обоснования выбора видов моделей и структур данных.

Существует несколько подходов к построению модели типа “сущность-связь”. Общим для всех подходов является использования нескольких элементов : сущность, атрибут, связь, время. Информация о проекте объединяется с помощью графических диаграмм.

Сущность – это объект, процесс или явление, о котором необходимо хранить информацию в системе. В качестве объектов или сущностей рассматриваются материальные (предприятия, изделия) и нематериаль-ные (счет, реферат) лексические или абстрактные (человек) объекты.

Атрибут – это поименованная характеристика объекта или сущ-ности, которая принимает значение из некоторого множества значений.

Например: книга – объект. Атрибуты- название, автор и т. д.

Основное назначение атрибута – описание свойства сущности, а также идентификация экземпляров сущностей.

Связь – выступает в модели в качестве средства с помощью которого представляются отношения между сущностями. Анализ связей между сущностями показал, что могут встречаться бинарные (между двумя сущностями), тернарные (между тремя) и в общем случае n-арные связи.

Информацию о проекте оформляют в виде диаграмм, для этого обозначают : типы сущностей- прямоугольниками, атрибуты- овалами, связи- ромбами соединяя их между собой.

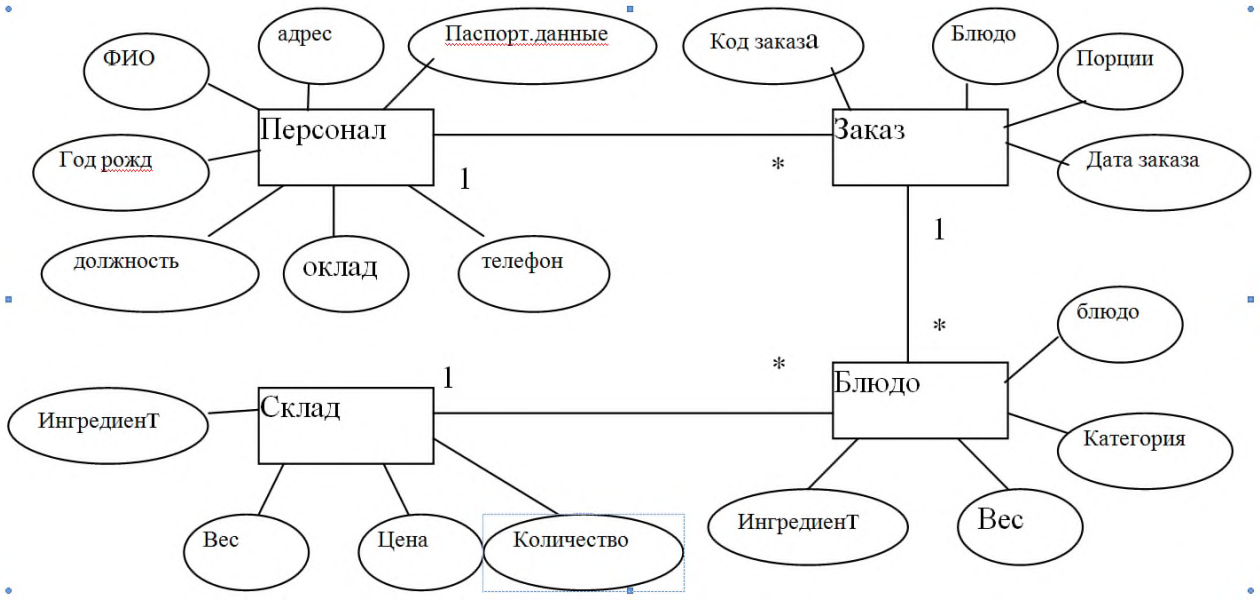
При моделировании предметной области проектировщик разбивает ее на ряд локальных областей, моделирует каждое локальное представление, а затем их объединяет.

Объединение локальных представлений. Перед объединением необходимо решить вопрос о порядке объединения локальных представлений. Обычно используется бинарное объединение (попарное). Перед объединением необходимо выполнить группировку локальных представлений (по смыслу или подобию). При объединении использу-ются следующие принципы: идентичность, агрегация, обобщение.

Два или более элементов модели идентичны, если имеют одинаковое смысловое значение.
Агрегация позволяет рассматривать связь между элементами модели как новый элемент (например, экзамен {фам_студента, наз_дисциплины, ФИО_препод, оценка}
Обобщением называется объединение данных, позволяющее трактовать класс различных типов объектов как один поименованный обобщенный тип объекта. Например, гарнитур{стол, стул, шкаф}

Проектирование схемы БД “Ресторан”

Проектирование базы данных «Ресторан». Будет производиться на основе описания реального объекта – ресторана.
Первый шаг проектирования базы данных для ресторана состоит в создании концептуальной модели данных, отвечающей практике ресторана. Задача проектирования состоит в определении требований к данным. Для этого необходимо поставить вопросы к данным. Например, необходимо решить каким образом будет производиться расчет прибыли? Из каких элементов она будет складываться? Каким образом будет осуществляться движение ингредиентов блюд? Как производится учет труда персонала его влияние на прибыль? Каким образом будет осуществляться компоновка меню? Каким образом будет подсчитываться дневная выручка и чистая прибыль? На основе этих данных можно будет проанализировать работу ресторана и тем самым увеличить прибыль.
Исходя, из этих вопросов выделяются объекты предметной области, о которых нужно собрать информацию в БД. Это следующие объекты: Персонал, Заказ, Блюдо, Склад. Объекты также можно выделить исходя из тех форм, которые циркулируют в той или иной предметной области. Мы не можем опереться на этот подход, так как у нас форм нет. Составляем концептуальную модель данных, объекты на ней представ-ляем в виде прямоугольников, свойства объектов в виде овалов. Вид модели представлен на рисунке



В представленной схеме модели данных показано, как объекты связаны друг с другом. Объекты представлены квадратами ,а отношения между ними отрезками. Символы «1» и «*» обозначают как именно они взаимодействуют.

- Отношения между объектами Персонал и Заказ(один ко многим): Каждый Служащий(официант) может выполнять один или нескольких Заказов (*).
- Каждый Заказ может состоять из одного или нескольких блюд (*).Отношение между объектом Заказ и Блюдо(один ко многим).
- Каждое блюдо может иметь несколько ингредиентов(*).

Мы видим тип отношений: один-ко-многим, существует еще один тип отношения не представленный на схеме. Это отношение один-к-одному.

Задание 2. В соответствии с индивидуальным вариантом создайте концептуальную схему базы данных.

№	Подсистема	Объекты и их характеристики	Запросы	Отчеты
1	«Кадры предприятия»	Работник: таб.номер,ФИО, дата рождения,адрес, должность,дата принятия, дата увольнения; Цех:номер,наименование, ФИО начальника, число бригад, количество рабочих в бригаде; Бригада:номер, наименование,цех, ФИО бригадира, план, факт.	Состав бригад по стажу работы; ФИО и адреса бригадиров; ФИО и адреса начальников цехов; ФИО работников ,выходящих в этом году на пенсию.	**1.** Список работников ,поступивших на предприятие в течение последнего месяца; **2.** ФИО работников ,уходящих в текущем месяце в отпуск.
2	«Материально-техническое снабжение»	Товар: наименование, шифр, ед. измерения; Движение: товар, поставщик, получатель, количество; Получатель:шифр, наименование, № счета.	1. Ведомость наличия товара на складе на i- е число; 2. Список поставщиков; 3. Список получателей.	Ведомость прихода; Ведомость расхода.

Итог работы: отчет, защита работы.

Практическая работа № 12
Создание логической и физической модели данных с помощью утилиты автоматизированного проектирования базы данных.

Цель: изучить создание логической и физической модели данных с помощью утилиты автоматизированного проектирования базы данных.

Задание 1 : Создать таблиц в логической модели базы данных средствами ErWin

1. На рабочем столе в папке "Для сохранения документов" создайте папку ErWin (для сохранения всех файлов работы)
2. Запустите программу проектирования структуры баз данных (ErWin 4.0)

3. Создайте новую модель с типом Логическая/Физическая, установите тип СУБД (DataBase) InterBase
4. Ознакомьтесь с интерфейсом программы (дерево объектов, рабочая область, название области), раскройте все «+»
5. Настройте рабочую область (на дереве объектов выберите 2ЛКМ хранимые отображения (Stored Display)), откроется окно с описанием отображений:

- Переименуйте - Уровень сущностей
- Введите как Автора свою Фамилию
- Проверьте, что отображение используется на логическом уровне
- Способ рисования связей ортогональный
- Уровень отображения (Display Level) – сущности (Entity)
- Показать глагольную фразу (verb phrase)
- Описание (Difinition) «Логическая схема БД «Учебный процесс» на уровне сущностей»

6. Аналогично создайте рабочую область:

- Имя: Уровень атрибутов
- Уровень отображения – атрибуты (Attribute)
- Показать глагольную фразу, установки ссылочной целостности (Referential Integrity)
- Описание (Difinition) «Логическая схема БД «Учебный процесс» на уровне атрибутов»
- Задайте свойства сущности: отображать первичный и вторичные ключи, типы данных, мигрирующий атрибут

7. Сохранить схему с именем Dekanat_tab (обратите внимание на расширение)
8. Сделайте активной Уровень сущностей.
9. Создайте сущности (кнопка Entity) Группа, Студент (при необходимости установите шрифт Arial Cyr). Проверьте есть ли они на дереве объектов и на странице Уровень атрибутов
10. Установите для созданных сущностей описание (2 ЛКМ на дереве Entity) «Данные об имеющихся в учебном заведении группах»
11. Создайте для дальнейшего использования домены (2 ЛКМ на дереве Domains):

- Кнопка New
- Имя для логической модели (Logical Name) – Номер
- Имя для физической модели (Physical Name) – Dn_nomer
- Базовый тип – Числовой. ОК
- Установите дополнительные характеристики домена для физического уровня (смените Edit Mode):
- На вкладе InterBase установите тип SmallInt, обязательное, ограничение от 1 до 100 (Valid … Min/max)

12. Аналогично создайте домен Dn_name для описания Названий (символьный, необязательный, 50 символов)
13. Сделайте активной Уровень атрибутов, откройте на дереве сущности
14. Добавьте в сущность Группа атрибуты (2 ЛКМ на дереве Группа - Attributes):

- Кнопка New

- Имя для логической модели (Attribut Name) – Код группы
- Имя для физической модели (Column Name) – Kod_gr
- Базовый тип – Номер. ОК
- Установите это поле ключевым
- Описание – Порядковый номер группы, целое, ключевое
- Самостоятельно добавьте поля Название (Name_gr) и ФИО Кл (Fio_kl), домен Названия, неключевое.

Сущность: Студент			
Логическое имя	Физичес-кое имя	Тип/домен	Кл. поле
Таб номер	Tab	Номер	Да
ФИО	Fio	Varchar (60)	
Дата рождения	Date_r	Date	

15.Самостоятельно для сущности **Студент** добавьте поля (см. таблицу)
16.Для рабочей области «Уровень атрибутов» отключите отображение доменов
17.Активизируйте Уровень сущностей
18.При помощи инструмента «Неидентифицирующая связь (Non- identifying relationship)» соедините сущность Группа и Студент (порядок соединения Главная - Подчиненная)
19.Установите характеристики связи (2 ЛКМ по связи):
- Текст глагольной фразы – «состоит из» со стороны главной, «обучаются в » со стороны подчиненной
- Степень связи – Ноль, один или более (проанализируйте, причину выбора этого типа)
- Тип связи – Неидентифицирующая, обязательная
- Описание – связь 1:М между Группой (главная) и Студентам (подчиненная) по Коду группы
- Установки ссылочной целостности (RI Actions)

— для главной (Удаление - Каскад, Добавление – ничего, Изменение - каскад)

— для подчиненной (Удаление - ничего, Добавление – запрет, Изменение - ничего)

20.Посмотрите как изменилась линия связи и глагольная фраза. Перейдите на Уровень атрибутов и найдите мигрирующий внешний ключ и описание ссылочной целостности
21.Сохранить схему с именем Dekanat_Rel
22.Создайте рабочую область «Физическая схема» аналогичными для Уровня атрибутов параметрами. Активизируйте ее
23.Замените все имена таблиц английскими и имена доменов на имена полей
24.Проверьте тип сервера (DataBase / Choose DataBase) нужен InterBase
25.Создайте физическую БД, т. Е. сгенерируйте SQL-скрипт (меню Tools / Forward Engineer/Schema Generation):

- На вкладке Summary просмотрите какие объекты будут созданный в SQL-скрипте
- Кнопка Preview для предварительного просмотра SQL-скрипта, после просмотра нажмите Close
- Кнопка Report для сохранения SQL-скрипта в файл (задайте имя Dekanat_sql)

26.Сохраните схему с именем Dekanat_Phis

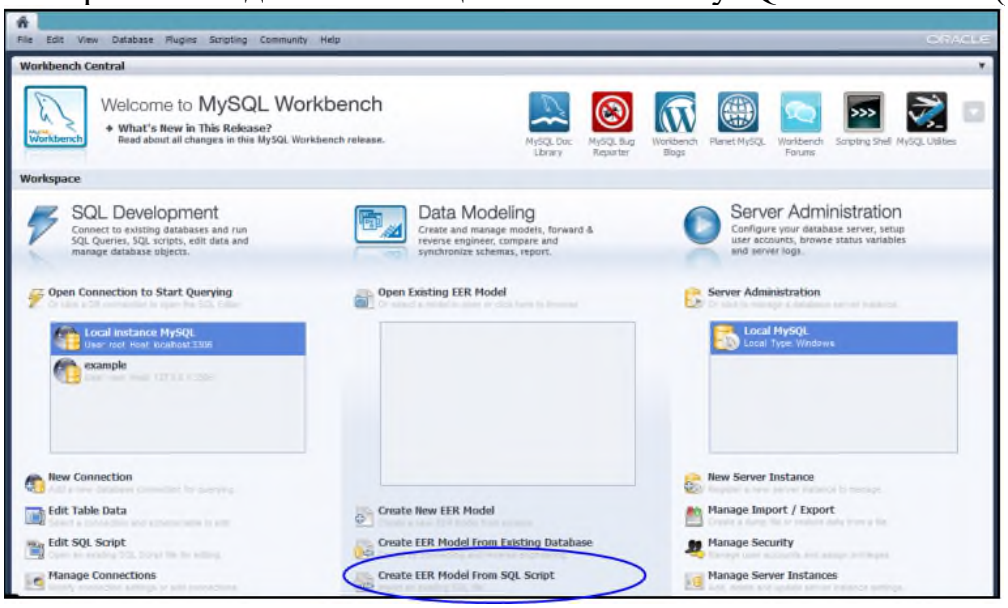
Задание 2:

Существует несколько концепций моделей баз данных (иерархическая, сетевая, объектная, реляционная). Наиболее распространенной моделью является реляционная модель, которая очень тесно переплетается с принципа- ми объектно-ориентированного анализа и еще одного популярного подхода в моделировании данных – ER-модели (модель «сущность-связь»).

ER-модель удобна для начального проектирования, поскольку она интуитивно понятна большинству пользователей. В ней выделяются понятия сущности (основные объекты базы), атрибуты (свойства сущности) и связи (взаимодействия между сущностями). В ряде оболочек именно в этих тер- минах и создан сервис создания модели данных.

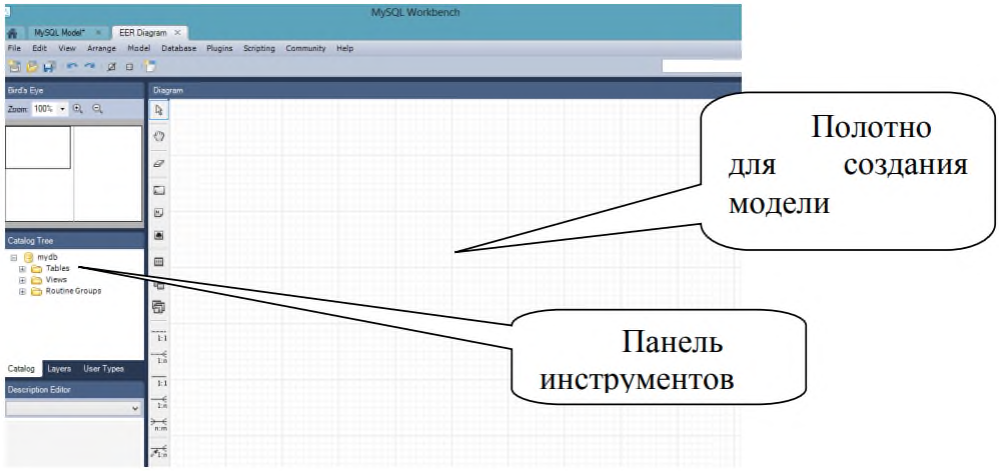
Реляционная модель представляет всю базу данных как набор связанных таблиц. Большинство таблиц отвечает за хранение информации о сущностях (столбцы таблиц характеризуют их атрибуты). Среди атрибутов сущности выделяют ключевые атрибуты – атрибуты, которые являются идентифицирующими, точно определяющими запись, объект сущности. С помощью внедрения ключевых атрибутов одних сущностей (родительские таблицы) в качестве столбцов в другие таблицы (дочерние) реализуются различные связи между сущностями.

Построение модели с помощью оболочки MySQL Workbench (версия 5.2.39 CE).

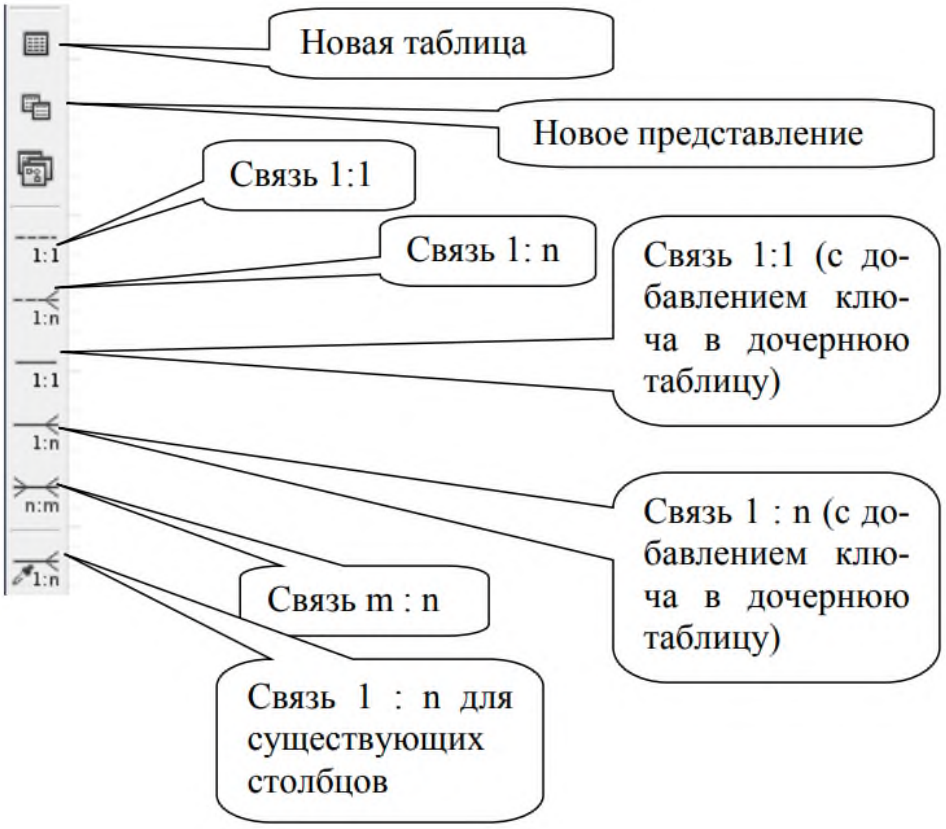


Создаем новую ER-модель и диаграмму в модели. В полученном окне модели представлено полотно, на которое можно наносить новые таблицы, с помощью визуальных средств редактирования, создать столбцы (атрибуты) таблиц и с помощью панели инструментов создать связи между таблицами. При установке

связи ключевые поля родительских сущностей добавляются в дочерние таблицы автоматически.



Состав панели инструментов окна редактирования модели данных.

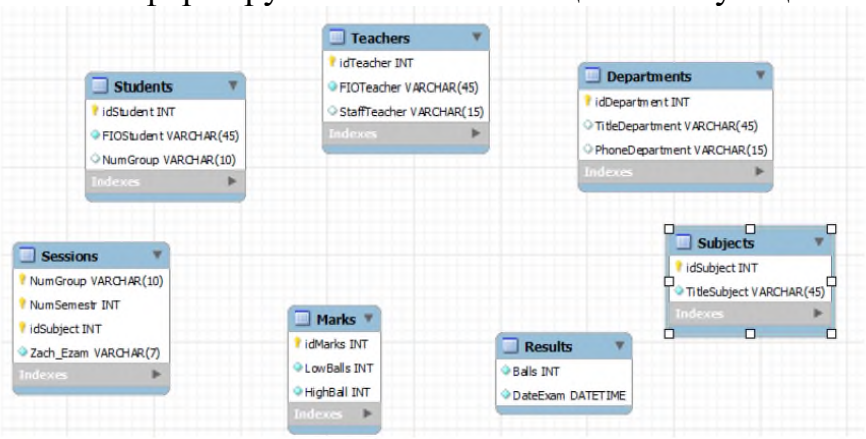


Проведем анализ состава таблиц для решаемой задачи. При описании столбцов таблицы поля, входящие в первичный ключ, будут подчеркнуты. Имеется таблица Студенты (Students): (номер зачетки, ФИО Студента).

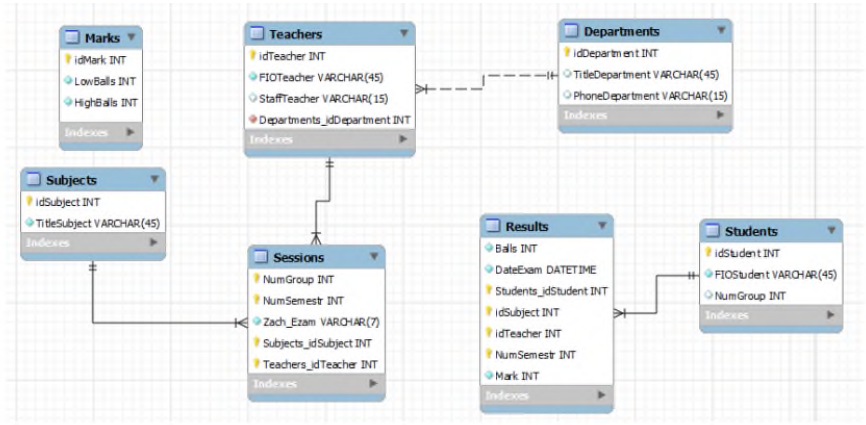
Для хранения групп не будем выделять отдельную таблицу. Имеется таблица Преподаватель (Teachers): (№Преподавателя, ФИО Преподавателя, Должность, №Кафедры).

Чтобы избежать дублирования информации с названием кафедры вве- дем справочную таблицу кафедр: таблица Кафедра (Departments): (№Кафедры,

Название, Телефон). Имеется таблица учебных дисциплин Дисциплина (Subjects): (№Дисциплины, Название).
Таблица Сессия содержит информацию о том, каков состав зачетов и экзаменов для каждой конкретной группы по семестрам, каким преподавателям следует сдавать зачеты и экзамены: Sessions (№Группы, №Семестра, №Дисциплины, Отчетность, №Преподавателя). Заметим, что отчетность может определяться номером дисциплины и номером семестра, но в предположении наличии нескольких специальностей один и тот же предмет может сдаваться в разных семестрах разными группами. Поэтому отчетность и преподаватель зависят и от группы тоже.
Наконец, результаты сдачи сессии хранятся в таблице результатов Results (№Студента, №Группы, №Семестра, №Дисциплины, Баллы, ДатаСдачи, Оценка). Окончательную оценку хранить не требуется, так как она определяется количеством набранных баллов и таблицей оценок.
Marks (Оценка, НижняяГраница, ВерхняяГраница) – эта таблица является справочной и не связана с основными таблицами базы. Ее роль за- ключается в определении правильной оценки по набранным баллам. В результате данного анализа задачи получится следующая модель:
- сначала формируется состав таблиц без связующих атрибутов:



- затем устанавливаем связи. Заметим, что можно было бы все связующие атрибуты сразу добавить в таблицы. Тогда все связи можно было бы добавить как связи «один-ко-многим» для существующих столбцов. Другой вариант решения этой проблемы, добавить все поля в таблицу результатов и не устанавливать связь на уровне модели. Далее после создания таблиц в базе данных добавить ограничения внешних ключей для полей номера дисциплины и номера преподавателя.



Отметим некоторую избыточность таблицы результатов относительно номера группы. Требуется обеспечить, чтобы номер группы и студенты были согласованы по таблицам студентов и результатов сессии.

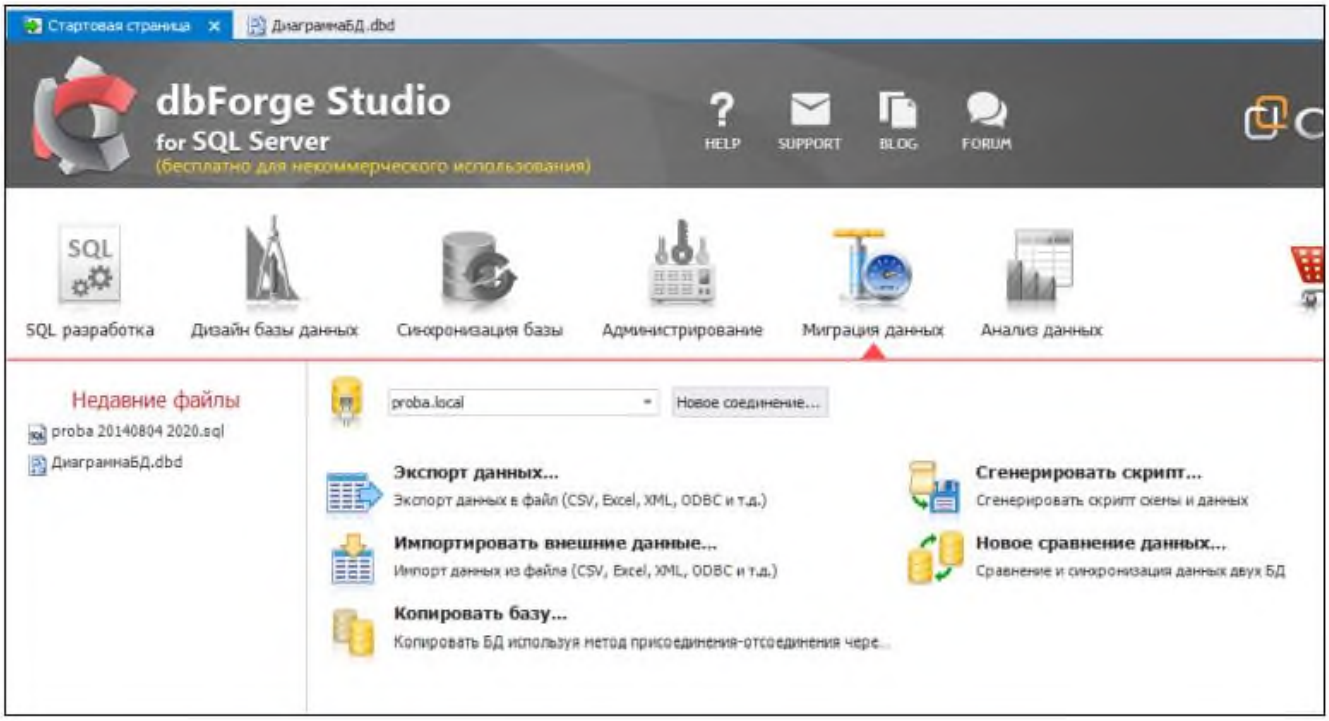
Итог работы: отчет, защита работы.

Практическая работа № 13
Разработка серверной части базы данных в инструментальной оболочке.

Цель: изучить разработку серверной части базы данных в инструментальной оболочке.

Задание: Создать резервную копию базы данных в оболочке dbForge Studio.

В оболочке dbForge Studio для SQL Server создание резервной копии (backup) можно осуществить двумя способами:
1. пункт меню «База данных» -> «Задачи» - > «Резервное копирование» (соответственно, для восстановления из резервной копии используется пункт меню «База данных» -> «Задачи» - > «Восстановление»). Этот способ связан с использованием специального формата MS SQL Server.
2. Генерация SQL-скрипта осуществляется с помощью пункта меню «База данных» -> «Задачи» - > «Сгенерировать скрипт…».
Многие важные опции, доступные через меню, доступны и на стартовой странице приложения, чтобы можно было получить к ним быстрый
доступ:
Вид стартовой страницы для вкладки «Миграция данных»



В результате будет сгенерирован файл, содержащий следующие SQL- команды. Выделим полужирным шрифтом те команды, которые касаются создания базы данных и всех ее таблиц, а также определение ограничений:

```
--
-- Скрипт сгенерирован Devart dbForge Studio for SQL Server, Версия 3.8.180.1
-- Домашняя страница продукта: http://www.devart.com/ru/dbforge/sql/studio
-- Дата скрипта: 04.08.2014 23:36:06
-- Версия сервера: 11.00.2100
-- Версия клиента:
-- USE master GO 21 IF DB_NAME() <> N'master' SET NOEXEC ON
--
-- Создать базу данных "proba"
-- PRINT (N'Создать базу данных "proba"') GO

CREATE DATABASE proba
ON PRIMARY(
NAME = N'proba',
FILENAME = N'c:\Program Files\Microsoft SQL
Server\MSSQL11.SQLEXPRESS\MSSQL\DATA\proba.mdf',
SIZE = 4160KB,
MAXSIZE = UNLIMITED,
FILEGROWTH = 1024KB )
LOG ON(
NAME = N'proba_log',
FILENAME = N'c:\Program Files\Microsoft SQL
Server\MSSQL11.SQLEXPRESS\MSSQL\DATA\proba_log.ldf',
SIZE = 1040KB,
MAXSIZE = UNLIMITED,
FILEGROWTH = 10%
)
GO
```

```
-- -- Изменить базу данных
-- PRINT (N'Изменить базу данных')
GO
ALTER DATABASE proba
SET
ANSI_NULL_DEFAULT OFF,
ANSI_NULLS OFF,
ANSI_PADDING OFF,
ANSI_WARNINGS OFF,
ARITHABORT OFF,
AUTO_CLOSE ON,
AUTO_CREATE_STATISTICS ON,
AUTO_SHRINK OFF,
AUTO_UPDATE_STATISTICS ON,
AUTO_UPDATE_STATISTICS_ASYNC OFF,
COMPATIBILITY_LEVEL = 110,
 CONCAT_NULL_YIELDS_NULL OFF,
 CONTAINMENT = NONE,
CURSOR_CLOSE_ON_COMMIT OFF,
 CURSOR_DEFAULT GLOBAL,
DATE_CORRELATION_OPTIMIZATION OFF,
DB_CHAINING OFF,
HONOR_BROKER_PRIORITY OFF,
MULTI_USER,
NUMERIC_ROUNDABORT OFF,
PAGE_VERIFY CHECKSUM,
PARAMETERIZATION SIMPLE,
QUOTED_IDENTIFIER OFF,
READ_COMMITTED_SNAPSHOT OFF,
RECOVERY SIMPLE,
RECURSIVE_TRIGGERS OFF,
TRUSTWORTHY OFF
WITH ROLLBACK IMMEDIATE
 GO
ALTER DATABASE proba
 SET ENABLE_BROKER
GO
ALTER DATABASE proba
 SET ALLOW_SNAPSHOT_ISOLATION OFF
GO
 ALTER DATABASE proba
SET FILESTREAM (NON_TRANSACTED_ACCESS = OFF)
GO
USE proba
GO
 IF DB_NAME() <> N'proba' SET NOEXEC ON
GO
--
-- Создать таблицу "dbo.Teachers"
-- PRINT (N'Создать таблицу "dbo.Teachers"') GO
CREATE TABLE dbo.Teachers (
 idTeacher int IDENTITY,
```

```
FIOTeacher varchar(50) NOT NULL,
idDepartment int NOT NULL,
CONSTRAINT PK_Teachers PRIMARY KEY (idTeacher) )
ON [PRIMARY]
GO
--
-- Создать таблицу "dbo.Subjects
" -- PRINT (N'Создать таблицу "dbo.Subjects"')
GO
CREATE TABLE dbo.Subjects (
idSubject int IDENTITY,
TitleSubject varchar(50) NOT NULL,
CONSTRAINT PK_Subjects PRIMARY KEY (idSubject) )
ON [PRIMARY]
 GO
 --
 -- Создать таблицу "dbo.Students"
-- PRINT (N'Создать таблицу "dbo.Students"')
GO
 CREATE TABLE dbo.Students (
idStudent int IDENTITY,
FIOStudent varchar(50) NOT NULL,
NumGroup int NOT NULL,
CONSTRAINT PK_Students PRIMARY KEY (idStudent) )
 ON [PRIMARY]
GO
--
-- Создать таблицу "dbo.Sessions"
-- PRINT (N'Создать таблицу "dbo.Sessions"')
GO
CREATE TABLE dbo.Sessions (
NumGroup int NOT NULL,
NumSemestr int NOT NULL,
idSubject int NOT NULL,
idTeacher int NOT NULL,
Zach_Exam varchar(7) NOT NULL,
CONSTRAINT PK_Sessions PRIMARY KEY (NumGroup, NumSemestr, idSubject, idTeacher) )
ON [PRIMARY]
GO
--
-- Создать таблицу "dbo.Departments"
 -- PRINT (N'Создать таблицу "dbo.Departments"')
GO
CREATE TABLE dbo.Departments (
 idDepartment int IDENTITY,
TitleDepartment varchar(50) NOT NULL,
PhoneDepartment varchar(50) NOT NULL,
CONSTRAINT PK_Departments PRIMARY KEY (idDepartment) )
 ON [PRIMARY]
GO
 --
-- Создать таблицу "dbo.Results"
```

```
-- PRINT (N'Создать таблицу "dbo.Results"')
GO
CREATE TABLE dbo.Results (
 idStudent int NOT NULL,
idSubject int NOT NULL,
idTeacher int NOT NULL,
DateExam datetime NOT NULL,
Balls int NOT NULL,
Mark int NOT NULL,
NumSemestr int NOT NULL,
CONSTRAINT PK_Results
PRIMARY KEY (idStudent, idSubject, idTeacher, NumSemestr) )
 ON [PRIMARY]
GO
--
-- Создать таблицу "dbo.Marks"
 -- PRINT (N'Создать таблицу "dbo.Marks"')
GO
 CREATE TABLE dbo.Marks (
idMark int IDENTITY,
LowBalls int NOT NULL,
HighBalls int NOT NULL,
CONSTRAINT PK_Marks PRIMARY KEY (idMark) )
ON [PRIMARY]
GO
--
-- секция для команд вставки данных из всех таблиц – ее пропустим
--
-- Создать внешний ключ "FK_Teachers" для объекта типа таблица "dbo.Teachers"
 -- PRINT (N'Создать внешний ключ "FK_Teachers" для объекта типа таблица "dbo.Teachers"')
GO
ALTER TABLE dbo.Teachers ADD CONSTRAINT FK_Teachers FOREIGN KEY (idDepartment) REFERENCES dbo.Departments (idDepartment)
GO
--
-- Создать внешний ключ "FK_Sessions_Subjects_idSubject" для объекта типа таблица "dbo.Sessions" -- PRINT (N'Создать внешний ключ "FK_Sessions_Subjects_idSubject" для объекта типа таблица "dbo.Sessions"')
GO
ALTER TABLE dbo.Sessions ADD CONSTRAINT FK_Sessions_Subjects_idSubject FOREIGN KEY (idSubject) REFERENCES dbo.Subjects (idSubject)
 GO
--
-- Создать внешний ключ "FK_Sessions_Teachers_idTeacher" для объекта типа таблица "dbo.Sessions" -- PRINT (N'Создать внешний ключ "FK_Sessions_Teachers_idTeacher" для объекта типа таблица "dbo.Sessions"')
GO ALTER TABLE dbo.Sessions ADD CONSTRAINT FK_Sessions_Teachers_idTeacher FOREIGN KEY (idTeacher) REFERENCES dbo.Teachers (idTeacher)
GO
--
 -- Создать внешний ключ "FK_Results" для объекта типа таблица "dbo.Results"
-- PRINT (N'Создать внешний ключ "FK_Results" для объекта типа таблица "dbo.Results"')
```

```
GO ALTER TABLE dbo.Results ADD CONSTRAINT FK_Results FOREIGN KEY (idStudent) REFERENCES dbo.Students (idStudent)
GO
--
-- Создать внешний ключ "FK_Results_Subjects_idSubject" для объекта типа таблица "dbo.Results"
 -- PRINT (N'Создать внешний ключ "FK_Results_Subjects_idSubject" для объекта типа таблица "dbo.Results"')
GO ALTER TABLE dbo.Results ADD CONSTRAINT FK_Results_Subjects_idSubject FOREIGN KEY (idSubject) REFERENCES dbo.Subjects (idSubject)
GO
--
-- Создать внешний ключ "FK_Results_Teachers_idTeacher" для объекта типа таблица "dbo.Results"
-- PRINT (N'Создать внешний ключ "FK_Results_Teachers_idTeacher" для объекта типа таблица "dbo.Results"')
GO ALTER TABLE dbo.Results ADD CONSTRAINT FK_Results_Teachers_idTeacher FOREIGN KEY (idTeacher) REFERENCES dbo.Teachers (idTeacher)
 GO
SET NOEXEC OFF
 GO
```

Итог работы: отчет, защита работы.

Практическая работа № 14

Разработка клиентской части базы данных в инструментальной оболочке

Цель: изучить разработку клиентской части базы данных в инструментальной оболочке.

Задание. Создать простую модель обращения к базе данных реализуется через три основных класса**.**

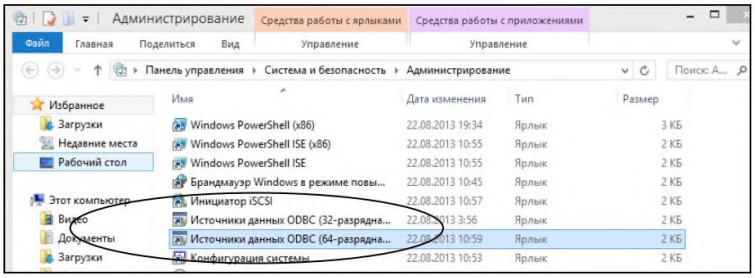
Простая модель обращения к базе данных реализуется через три основных класса, которые могут быть реализованы на основе как универсальных технологий (Ole Db или ODBC), так и конкретных СУБД (например, SQL Server или MySQL):

- Класс соединения;
- Класс команды;
- Класс курсора для получения данных из результата запроса

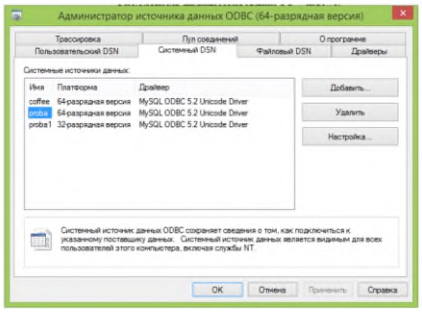
Класс соединения настраивает параметры базы данных, к которой следует подключиться. Открытие такого соединения позволяет формулировать команды к серверу баз данных. Команды оформляются с помощью языка SQL и оформляются в программе с помощью класса команды. Класс курсора предназначен для чтения данных результата запроса. Курсор обычно является последовательным, т.е. с помощью него мы можем получить последовательно все записи результата запроса, начиная с первой до последней, но не можем вернуться к уже просмотренным записям.

Настройка соединения осуществляется через понятие строки соединения –

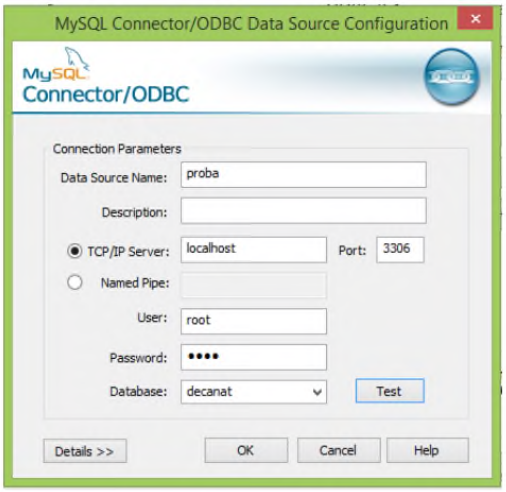
параметров, которые описывают драйверы СУБД, конкретные базы данных и параметры безопасности для подключения к базе.
Например, для настройки источника данных ODBC требуется воспользоваться средством администрирования операционной системы Windows.



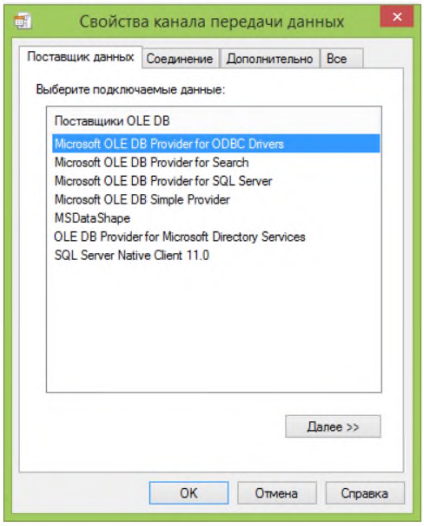
Для общения с серверными СУБД требуется, чтобы источник данных был системным:



При создании и редактировании источника данных требуется выбрать драйвер и задать параметры подключения к базе данных:

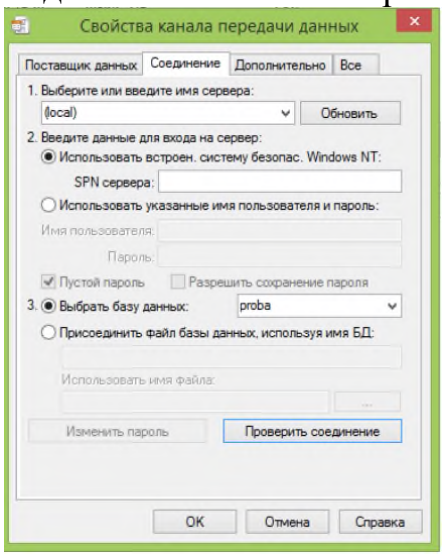


Драйверы могут устанавливаться и регистрироваться в операционной системе вместе с соответствующей СУБД, а могут быть и самостоятельным компонентами, которые следует установить отдельно. Так, например, для SQL Server драйвер SQL Native Client устанавливается в процессе установки СУБД, а для сервера MySQL драйвер (MySQL ODBC Connector) является дополнительной компонентой, которую следует установить отдельно.
Для настройки строки соединения через технологию Ole Db можно воспользоваться специальной утилитой операционной системы Windows. Для этого требуется создать файл с расширением “.udl”. При последующем открытии этого файла будут вызвано окно настройки источника данных:



После выбора поставщика данных на вкладке «Подключение» следует настроить параметры базы данных (имя, местоположение, параметры безопасности). Этот файл можно будет открыть с помощью текстового редактора – в нем будет записана строка соединения через выбранный провайдер Ole Db.

Для примера рассмотрим подключение к базе данных на SQL Server. Создадим файл connect.udl (например, с помощью «Блокнота») и откроем его с помощью службы OLE DB Core Services. На вкладке «Поставщик данных» следует выбрать пункт «SQL Server Native Client номер версии» (номер версии, очевидно, зависит от установленного на компьютере SQL Server'а). Далее на вкладке «Соединение» следует настроить параметры подключения к серверу – имя сервера, параметры входа (обычно устанавливаются из системы безопасности операционной системы), имя используемой базы данных. С помощью кнопки «Проверить подключение» можно протестировать созданную строку соединения.



Далее при открытии файла connect.udl с помощью «Блокнота» мы увидим сгенерированную строку подключения:

```
Provider=SQLNCLI11.1;Integrated Security=SSPI;
 Persist Security Info=False;
 User ID="";Initial Catalog=proba;Data Source=(local);
 Initial File Name="";Server SPN=""
```

Далее приведен программный код, в котором осуществляется подключение к источнику данных, заданному с помощью ODBC и получение данных из базы данных с помощью формирования SQL-запроса:

```
// создание подключения к базе данных на основе строки соединения
// с указанием источника данных ODBC
OdbcConnection con = new OdbcConnection("DSN=proba");
// подключение к источнику данных
con.Open();
// формирование команды SQL на выборку данных
OdbcCommand com = new OdbcCommand("select * from Session", con);
// выполнение команды на сервере и сохранение результата
// в курсоре типа OdbcDataReader
OdbcDataReader dr=com.ExecuteReader();
// переход к следующей строке посредством функции Read()
// пока строки в результате есть – печатаем информацию из строк
while (dr.Read())
 Console.WriteLine(“”+dr[“NumGroup”]+” “+dr[“idSubject”]+
 ”
”+dr[“Zach_exam”]);
// закрывается соединение
dr.Close();
con.Close();
```

Аналогичным будет программный код и в случае использования подключения по технологии Ole DB. Отличия будут только в именах используемых классов для доступа к базе данных (OleDbConnection, OleDbCommand, OleDbDataReader) и, соответственно, в пространстве имен, содержащее эти классы.

Отметим еще момент, связанный с обеспечением доступа к полям строки курсора. Текущая строка представляется как ассоциативный массив, доступ к элементам которого можно осуществлять по номерам или по именам столбцов, что очень удобно с точки зрения последующего чтения программного кода.

Итог работы: отчет, защита работы.

Практическая работа № 15
Построение запросов к базе данных на языке SQL (различных типов).

Цель: изучить построение запросов к базе данных на языке SQL (различных типов).

Задание. Создать запрос по номеру группы, для которой требуется распечатать план сессии, вводится с клавиатуры и становится условием выборки записей из таблицы плана сессии для студентов**.**

```
// создание подключения к базе данных на основе строки соединения
```

```
// с указанием источника данных ODBC
OdbcConnection con = new OdbcConnection("DSN=proba");
// подключение к источнику данных
con.Open();
// ввод номера группы
string group = Console.ReadLine();
// формирование команды SQL на выборку данных
OdbcCommand com = new OdbcCommand
 ("select * from Sessions where NumGroup=?",
con);
 // задание значение параметра запроса
com.Parameters.AddWithValue(“@par”,group);
// выполнение команды на сервере и сохранение результата
// в курсоре типа OdbcDataReader
OdbcDataReader dr=com.ExecuteReader();
// переход к следующей строке посредством функции Read()
// пока строки в результате есть – печатаем информацию из строк
while (dr.Read())
 Console.WriteLine(“”+dr[“NumGroup”]+” “+dr[“idSubject”]+
 ” ”+dr[“Zach_exam”]);
// закрывается соединение
dr.Close();
con.Close();
```

Задание 2. Создать запрос используя функцию GetMark1, которая позволяет перевести баллы в оценку.

```
// создание подключения к базе данных на основе строки соединения
// с указанием источника данных ODBC
OdbcConnection con = new OdbcConnection("DSN=proba");
// подключение к источнику данных
con.Open();
// ввод набранных баллов на экзамене
string b = Console.ReadLine();
// формирование команды SQL на вызов функции и получение ее резуль-
тата
OdbcCommand com = new OdbcCommand("select GetMark1(?)", con);
// задание значение параметра запроса
com.Parameters.AddWithValue(“@ball”,b);
// выполнение команды на сервере и сохранение результата
//в курсоре типа OdbcDataReader
OdbcDataReader dr=com.ExecuteReader();
// переход к первой строке – результат вызова функции
dr.Read();
Console.WriteLine(“Оценка - ”+dr[0]);
```

```
// закрывается соединение
dr.Close();
con.Close();
```

Задание 3. Создать новую учебную дисциплину.

```
// создание подключения к базе данных на основе строки соединения
// с указанием источника данных ODBC
OdbcConnection con = new OdbcConnection("DSN=proba");
// подключение к источнику данных
con.Open();
// ввод названия новой учебной дисциплины
string title = Console.ReadLine();
// формирование команды SQL на добавление данных – в таблице ключ
// задается с помощью поля-счетчика, так что указывать
// его в запросе на вставку не обязательно
OdbcCommand com = new OdbcCommand
 ("insert into Subjects values (‘?’)", con);
// задание значение параметра запроса
com.Parameters.AddWithValue(“@par”,title);
// выполнение команды на сервере
com.ExecuteScalar();
// закрывается соединение
con.Close();
```

Итог работы: отчет, защита работы.

Практическая работа № 16
Создание хранимых процедур в базах данных (различных типов).

Цель: изучить создания хранимых процедур в базах данных (различных типов).

Задание 1. Написать хранимую процедуру, которая получает в качестве входного параметра количество баллов и на основании шкалы оценок вычисляет полученную оценку. Результат возвращается через выходной параметр.

Хранимые процедуры, функции и триггеры вводятся в базу данных для обеспечения бизнес-логики приложения на уровне серверной его компоненты. Обычно хранимые процедуры и функции представляют собой утилиты, которые определенным образом обрабатывают данные или реализуют достаточно сложный алгоритм вычисления некоторых показателей.
В дереве элементов базы данных в любом СУБД имеются группы для определения этих программных элементов:

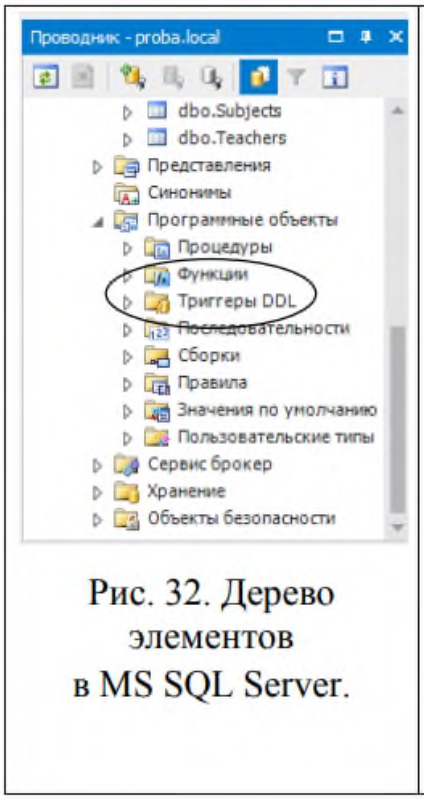


Рис. 32. Дерево элементов в MS SQL Server.

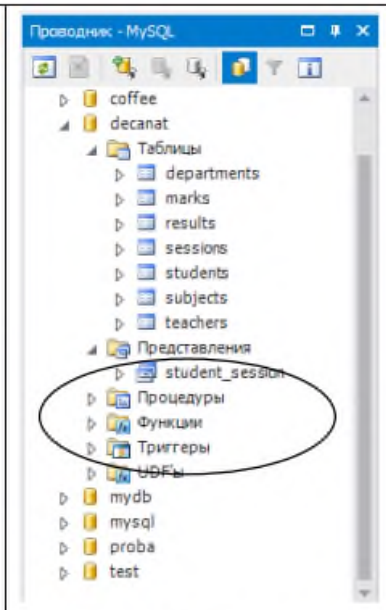


Рис. 33. Дерево элементов в MySQL.

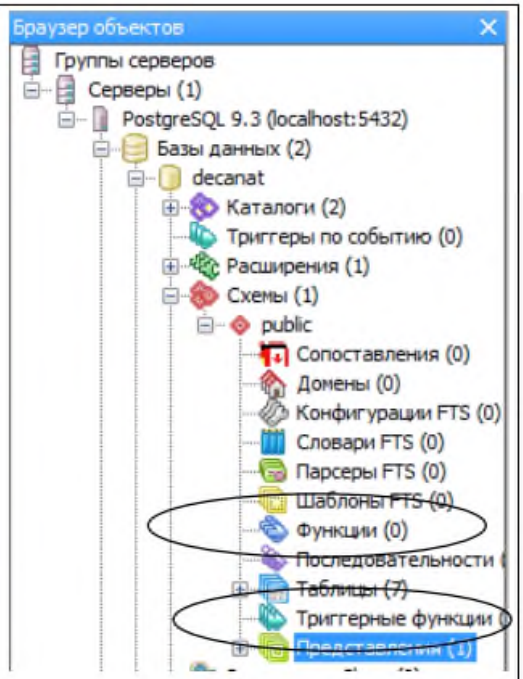


Рис. 34. Дерево элементов в PostgreSQL.

Для создания процедуры, функции или триггера требуется воспользоваться контекстным меню соответствующего элемента дерева. Для создания и редактирования существует специальное диалоговое окно, в котором, в частности, можно задать программный код процедуры, функции или триггера. Программный код формируется посредством перемешивания команд управления и SQL-команд.

Теперь приведем несколько примеров создания хранимых процедур и функций. Здесь мы уже заметим существенные отличия в синтаксисе используемых команд для различных СУБД. Поэтому для каждого из СУБД текст процедур, функций, триггеров и способы вызова укажем отдельно.

MS SQL Server. В SQL Server любая переменная именуется, начиная с символа ‘@’. Остальной код комментировать не требуется.

```
 CREATE PROCEDURE dbo.GetMark1 (@ball int, @mark INT OUT)
AS
BEGIN
IF @ball BETWEEN 55 AND 70 50
SET @mark=3;
ELSE IF @ball BETWEEN 71 AND 85
SET @mark=4;
 ELSE IF @ball BETWEEN 86 AND 100
SET @mark=5;
 ELSE SET @mark=2;
END
 GO
```

Для вызова процедуры требуется создать переменную для применения ее в качестве выходного параметра, после чего воспользоваться командой EXEC. Распечатать результат в выходном потоке можно с помощью оператора PRINT.

```
-- пример вызова процедуры GetMark1
DECLARE @mark INT;
EXEC GetMark1 78, @mark OUT;
PRINT '78 баллов соответствует оценке' + STR(@mark);
```

```
78 баллов соответствует оценке         4
Выполнение завершено успешно [0,112c]
```

Задание 2 . Создать триггер для вставки в таблицу результатов сессии, в котором проверяются ограничения целостности (студент с заданном кодом существует, предмет с заданным кодом существует, дисциплину нужно сдавать именно в этом семестре).

Если произойдет нарушение этих ограничений, то требуется откатить транзакцию, т.е. не выполнять вставку записи. Если же все данные будут корректными, проведем заполнение значений полей даты сдачи зачета/экзамена как текущей и вычислим оценку по указанным баллам. Для проверки корректности данных для вставки создадим вспомогательную хранимую функцию, чтобы код триггера был не очень сложным. (Для не- которых версий СУБД требуется, чтобы в триггере было упоминание только текущей записи, обращение к другим таблицам и записям недоступно).

MS SQL Server:

```
CREATE FUNCTION dbo.IsCorrect(@idStud INT, @idSubj INT, @Sem INT, @idTeach INT)
RETURNS INT
AS
BEGIN
IF EXISTS (SELECT * FROM Students INNER JOIN Sessions
ON Students.NumGroup=Sessions.NumGroup INNER JOIN Subjects ON
Sessions.idSubject=Subjects.idSubject
INNER JOIN Teachers
ON Sessions.idTeacher=Teachers.idTeacher
WHERE Students.idStudent=@idStud AND
Subjects.idSubject=@idSub
AND Teachers.idTeacher=@idTeach and NumSemestr=@Sem)
RETURN 1;
RETURN 0;
END
GO
```

Итог работы: отчет, защита работы

Практическая работа № 17
Создание триггеров в базах данных (различных типов).

Цель: изучить создание триггеров в базах данных (различных типов)..

Задание 1. Создать триггер на вставку новой записи в таблицу результатов.

Триггеры – это частный случай хранимой процедуры, который выполняется автоматически при выполнении команд обновления данных (INSERT, DELETE, UPDATE). Триггеры привязываются к конкретным таблицам базы данных. Для каждой команды должны быть свои триггеры

Триггер на вставку новой записи в таблицу результатов должен срабатывать после вставки и быть связан с подсчетом рейтинга студентов. Триггеры «после» часто используются для проведения специальной обработки данных на основании выполненной операции и могут быть связаны с другими таблицами.

Для этого введем в базу данных новую таблицу, например, с помощью следующей SQL-команды:

```
CREATE TABLE Reyting
 ( idStudent INT PRIMARY KEY,
 summ_balls INT,
 CONSTRAINT fk_reyting
 FOREIGN KEY (idStudent) REFERENCES Students (idStudent)
 )
```

Задание 2. Создать триггер на вставку записи в таблицу Results, который будет вызывать функцию проверки корректности, передавая в функцию поля из новой записи. Если запись будет корректной, будут скорректированы поля оценки и даты сдачи зачета/экзамена. В противном случае должен быть произведен откат транзакции.

MS SQL Server:

При вставке записи сначала запись попадает в виртуальную таблицу inserted (при удалении будет использоваться таблица deleted, при изменении записи используются обе таблицы – в inserted хранятся новые значения записи, в deleted – прежние значения полей записи). Поэтому сначала получаем данные новой записи из таблицы inserted, после чего проверяем их на корректность. В случае корректных данных оставшиеся поля (дата и оценка) изменяются посредством команды UPDATE. Откат транзакции в случае некорректных данных производится с помощью команды ROLLBACK.

```
CREATE TRIGGER trigger1
ON dbo.Results
FOR INSERT
AS
BEGIN
-- объявление необходимых переменных для хранения данных новой записи
DECLARE @idStudent INT, @idSubject INT,
@idTeacher INT, @NumSemestr INT, @Balls INT;
-- чтение данных новой записи
SET @idStudent =(SELECT idStudent FROM inserted);
SET @idSubject =(SELECT idSubject FROM inserted);
SET @idTeacher =(SELECT idTeacher FROM inserted);
SET @NumSemestr =(SELECT NumSemestr FROM inserted);
SET @Balls =(SELECT Balls FROM inserted);
-- проверка на корректность данных
```

```
IF dbo.IsCorrect(@idStudent, @idSubject, @NumSemestr, @idTeacher)=0
BEGIN
-- данные некорректны. Выводим сообщение об ошибке
-- и производим откат транзакции
PRINT 'Ошибка данных: данные некорректны';
ROLLBACK; END ELSE
-- изменение полей даты и вычисление оценки.
```

Итог работы: отчет, защита работы.

4. ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ПРАКТИЧЕСКИХ РАБОТ

4.1 Печатные изделия:

Основные:

О-1 Федорова Г.Н., Основы проектирования баз данных: учебник/ Г.Н.Федорова, - М.: ИД "Академия"- М, 2018.

О-2 Федорова Г.Н., Разработка, администрирование и защита баз данных: учебник/ М.: ИД "Академия"- М, 2018.

О-3 Перлова О.Н., Ляпина О.П., Соадминистрирование баз данных и серверов: учебник/ М.: ИД "Академия"- М, 2018.

Дополнительные:

Д-1. Карпова Т.С. Базы данных: модели, разработка, реализация: учебное пособие/ Т.С. Карпова – М.: Питер, 2001.

4.2. Электронные издания (электронные ресурсы)

1.Федорова Г.Н., Основы проектирования баз данных: учебник/ Г.Н.Федорова, - М.: ИД "Академия"- М, 2018, 15 подключений

2. Федорова Г.Н., Разработка, администрирование и защита баз данных: учебник/ М.: ИД "Академия"- М, 2018, 25 подключений

3 Перлова О.Н., Ляпина О.П., Соадминистрирование баз данных и серверов: учебник/ М.: ИД "Академия"- М, 2018, 15 подключений.

5.ЛИСТ ИЗМЕНЕНИЙ И ДОПОЛНЕНИЙ, ВНЕСЕННЫХ В МЕТОДИЧЕСКИЕ УКАЗАНИЯ

№ изменения, дата внесения, № страницы с изменением	
Было	**Стало**
Основание: **Подпись лица, внесшего изменения**	