МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал) СКФУ
Колледж Пятигорского института (филиал) СКФУ

ОСНОВЫ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

Специальности СПО
09.02.07 Информационные системы и программирование

Пятигорск 2025

Методические указания для практических занятий по дисциплине ОП.08 Основы проектирования баз данных составлены в соответствии с требованиями ФГОС СПО. Предназначены для студентов, обучающихся по специальности 09.02.07 Информационные системы и программирование.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

В результате изучения Основ проектирования баз данных студенты должны **уметь:**

- проектировать реляционную базу данных;
- использовать язык SQL для программного извлечения сведений из баз данных.

знать:

- основы теории баз данных;
- модели данных;
- особенности реляционной модели и их влияние проектирование баз данных,
- изобразительные средства, используемые в ER-моделировании;
- основы реляционной алгебры;
- принципы проектирования баз данных, обеспечение непротиворечивости и целостности данных;
- средства проектирования структур баз данных;
- язык запросов SQL

ОБЩИЕ МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

По Основам проектирования баз данных практические занятия содержат задачи и теоретические вопросы. Варианты для каждого обучающегося - индивидуальные.

Задачи и ответы на вопросы, выполненные не по своему варианту, не засчитываются.

Практическое занятие выполняется в отдельной тетради. Условия задачи и формулировки вопросов переписываются полностью. Формулы, расчеты, ответы на вопросы пишутся ручкой, а чертежи, схемы и рисунки выполняются карандашом, на графиках и диаграммах указывается масштаб. Вначале задача решается в общем виде, затем делаются расчёты по условию задания. Решение задач обязательно ведется в Международной системе единиц (СИ).

При выполнении практического занятия необходимо следовать методическим указаниям: повторить краткое содержание теории, запомнить основные формулы и законы, проанализировать пример выполнения аналогичного задания, затем преступить непосредственно к решению задачи. К зачету допускаются студенты, получившие положительные оценки по всем практическим работам.

Правила выполнения практических занятий.

1. Студент должен придти на практическое занятие подготовленным к выполнению практической работы.
2. Каждый студент после проведения работы должен представить отчет о проделанной работе с анализом полученных результатов и выводом по работе.
3. Таблицы и рисунки следует выполнять с помощью чертежных инструментов (линейки, циркуля, и.т.д.) карандашом с соблюдением ЕСКД.
4. Расчет следует проводить с точностью до двух значащих цифр.

5. Исправления проводить на обратной стороне листа. При мелких исправлениях неправильное слово (буква, число и т.п.) аккуратно зачеркивается и над ним пишут правильное пропущенное слово (букву, число и т.п.).
6. Вспомогательные расчеты можно выполнять на отдельных листах, а при необходимости на листах отчета.
7. Если студент не выполнит практическую работу или часть работы, то он выполнит ее во внеурочное время, согласованное с преподавателем.
8. Оценку по практической работе студент получает с учетом срока выполнения работы, если;

- расчеты выполнены правильно и в полном объеме;
- сделан анализ проделанной работы и вывод по результатам работы;
- студент может пояснить выполнение любого этапа работы;
- отчет выполнен в соответствии с требованиями к выполнению работы.

Практическое занятие №1
Концепция проектирования.

Цель: Получение навыков анализа предметной области и разработки структуры БД. Изучение средств реализации БД.

Теория

Архитектура базы данных содержит три уровня: **концептуальный, внешний и внутренний**.

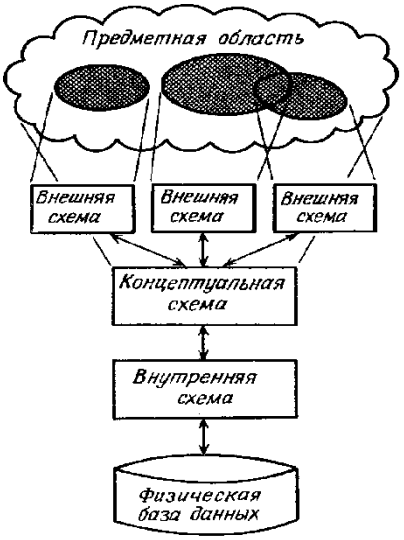


Рис.1. Три уровня архитектуры базы данных

На **концептуальном** уровне осуществляется концептуальное проектирование базы данных (БД), которое включает анализ информационных потребностей пользователей и определение необходимых элементов данных. В результате концептуального проектирования создается **концептуальная схема базы данных**, в которой на логическом уровне описываются все необходимые данные и связи между ними.

Внешний уровень- это структурный уровень базы данных, определяющий пользовательские представления о предметной области. Концептуальный уровень должен учитывать совокупность всех пользовательских представлений, то есть представление каждого пользователя можно вывести из концептуального уровня.

Внутренний уровень– это структурный уровень базы данных, определяющий физический вид базы данных. Проектирование на физическом уровне осуществляется с учетом конкретной СУБД (системы управления базами данных) и используемых технических средств (компьютеров, запоминающих устройств, доступа к сетям и т.д.)

На этапе концептуального проектирования разрабатывается концептуальная схема БД. Для этого описываются модели

пользовательских представлений данных, которые интегрируются в концептуальную модель, содержащую все элементы корпоративных данных, из которых будет состоять БД.

Разработчикам информационных систем необходимо владеть навыками разработки концептуальных моделей БД.

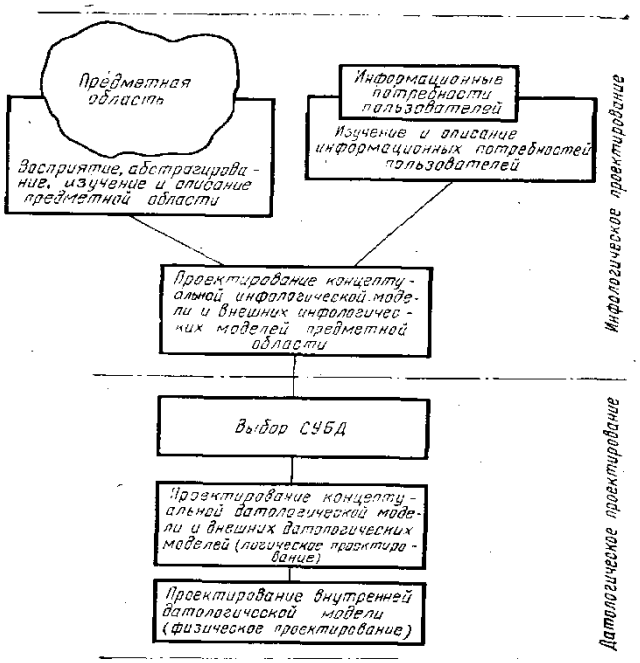


Рис. 2 Этапы проектирования базы данных

В результате выполнения лабораторной работы Вы сможете:

1. Пользоваться основными приемами концептуального моделирования данных для фиксации данных и отношений между ними, содержащихся в простых пользовательских запросах и существующих отчетах и формах**.**

2. Показать, как создаются составные объектные множества на основе существующих отношений, и объяснить, как они функционируют в качестве объектных множеств, обладающих атрибутами, и участвуют в отношениях.

3. Продемонстрировать, как концептуальное моделирование данных может быть применено к решению часто встречающихся в бизнесе информационных проблем.

Основные понятия концептуальных моделей

В процессе определения требований и концептуального проектирования выясняются информационных требования пользователей, которые представляются в виде хорошо сконструированной модели.

Модель - это представление реальности, отражающее лишь избранные детали.

Для создания модели необходимо **отобразить** элементы

реальности в элементы модели. Если процесс отображения выполнен должным образом, то моделью можно воспользоваться для решения задачи. Если нет, то модель не может послужить источником правильного решения.

Разработка **концептуальной модели данных** является методологической основой создания **схем баз данных** для конкретных практических ситуаций.

Объекты

Главными элементами концептуальной модели данных являются **объекты** и **отношения**. Объекты часто представляют в виде *существительных,* а отношения — в виде *глаголов.*

Объекты представляют собой вещи, которые пользователи считают важными в моделируемой нами части реальности. Примерами объектов могут быть люди, автомобили, деревья, дома, молотки и книги.

Объекты делятся на **конкретные** и **концептуальные**. Примеры конкретных объектов: люди, книги, дискеты и т.д. Концептуальными объектами будут компании, навыки, организации, проекты товаров, деловые операции, штатное расписание.

Для того, чтобы уточнить называется ли объектом конкретная вещь (отдельный человек, конкретный автомобиль, конкретный банк) или *множество* вещей (все люди, все автомобили, все банки) пользуются термином **объектное множество** для обозначения множества вещей одного типа **и объект-элемент** для обозначения одного члена (одного элемента) объектного множества.

Как показано на рис. 3 мы будем изображать объектные множества в виде прямоугольников, а объекты-элементы — в виде точек. Имя объектного множества пишется заглавными буквами в единственном числе. Так «ЧЕЛОВЕК» — имя объектного множества, представляющего людей. Строчными буквами («человек») обозначается элемент из объектного множества ЧЕЛОВЕК. Мы пишем «человек в ЧЕЛОВЕК», чтобы обозначить, что человек является элементом объектного множества ЧЕЛОВЕК.

Объектное множество. Множество вещей одного типа.

Объект-элемент. Конкретный элемент объектного множества.

Рис. 3 Объектное множество и объект-элемент

Объектные множества бывают **лексическими и абстрактными.** Элементы лексических объектных множеств можно напечатать, тогда как элементы абстрактных объектных множеств напечатать нельзя.

Так, например, ИМЯ будет лексическим объектным множеством, поскольку его элементами являются имена, то есть строки символов, которые можно напечатать. ДАТА. КОЛИЧЕСТВО и НОМЕР-ПАСПОРТА также являются примерами лексических объектных множеств, так как даты, количества и номера паспортов также можно распечатать.

Лексическое объектное множество. Объектное множество, состоящее из элементов, которые можно распечатать.

Абстрактное объектное множество. Объектное множество, состоящее из элементов, которые нельзя распечатать.

С другой стороны, ЧЕЛОВЕК является абстрактным объектным множеством, поскольку человека напечатать нельзя.

В компьютерной реализации концептуальной модели элементы лексических объектов будут представлены в виде **строк символов**.

Элементы абстрактных объектов будут представлены **внутренними номерами**, не имеющими смысла вне системы. Внутренний номер иногда называют «*Идентификатор объекта»* или **суррогатным ключом**, так как он представляет и однозначно определяет абстрактный объект-элемент реального мира. Примерами суррогатных ключей являются: для гражданина – данные паспорта, для сотрудника предприятия – табельный номер и т.д.

Конкретизиция и обобщение

Некоторые объектные множества содержатся внутри других объектных множеств. Например, МУЖЧИНА (множество мужчин) содержится внутри множества ЧЕЛОВЕК. Это означает, что каждый мужчина (элемент множества МУЖЧИНА) является также человеком (элементом множества ЧЕЛОВЕК). Аналогично, множество ЖЕНЩИНА содержится внутри множества ЧЕЛОВЕК (ЧЕЛОВЕК).

В данном случае МУЖЧИНА — **конкретизация** (или подмножество) множества ЧЕЛОВЕК. Мы можем представить это, написав МУЖЧИНА - ЧЕЛОВЕК.

ЧЕЛОВЕК, с другой стороны, является **обобщением** или надмножеством множества МУЖЧИНА (и множества ЖЕНЩИНА).

Конкретизация. Объектное множество, являющееся подмножеством другого объектного множества (содержащее его).

Обобщение. Объектное множество, являющееся надмножеством другого объектного множества.

Графическое изображение конкретизации/обобщения представлено на рис. 4. U-образный символ обозначает направление включения. Верхняя часть U«открывается» в сторону большего или объемлющего множества.

Рис. 4 Альтернативные представления конкретизации и обобщения

Представим себе мужчину по имени Джордж. Тогда Джордж является также человеком. Это представлено графически на рис. 5 Обратите внимание, что две точки обозначают одного и того же человека. Одна точка представляет его как элемент множества ЧЕЛОВЕК, а вторая — как элемент множества МУЖЧИНА. На самом деле это один объект. Он просто показан принадлежащим двум разным объектным множествам. Мы вскоре покажем важность такого представления.

Рис. 5 Две точки, представляющие один и тот же объект

Задание на лабораторную работу:

1. Для выбранной ПрО провести анализ информационных объектов и связей между ними.
2. Составить концептуальную модель ПрО и изобразить ее в виде ER-диаграммы.
3. Определить атрибуты информационных объектов. Составить информационно-логическую модель БД.
4. Определить первичные и вторичные ключи в таблицах и типы связей между таблицами.
5. Создать структуру БД

Методические указания

Описание предметной области должно содержать подробное описание информационных объектов и связей между ними.

Существует два подхода к описанию и анализу предметной области:

Проблемно-ориентированный (функциональны). В первом подходе сначала определяются основные задачи, для решения которых строится база, выявляются потребности задач в данных и, соответственно, определяется состав и структура информационных объектов.

Объектно-ориентированный (предметный). При втором подходе сразу устанавливаются типовые объекты ПрО.

Наиболее рационально сочетание обоих подходов.

Концептуальная модель схематически описывает информационные объекты предметной области: сущности и их атрибуты, связи между объектами. Представляется в виде диаграмм «сущность- связь» (Entity-Relation).

В реляционной модели данные формируются в виде связанных двумерных таблиц.

Между двумя или более таблицами БД могут существовать отношения подчиненности. Отношения подчиненности определяют, что для каждой записи главной таблицы (master, называемой еще родительской) может существовать одна или несколько записей в подчиненной таблице (detail, называемой еще дочерней).

Существует три разновидности связей между таблицами: “один-ко-многим” (1:M), “один-к-одному” (1:1), “многие-ко-многим” (M:M).

Отношение “один-ко-многим” имеет место, когда одной записи родительской таблицы может соответствовать несколько записей в дочерней таблице.

Отношение “один-к-одному” имеет место, когда одной записи в родительской таблице соответствует одна запись в дочерней таблице (рис.1.8).

Отношение “многие-ко-многим” имеет место, когда:

а) записи в родительской таблице может соответствовать больше одной записи в дочерней таблице;

б) записи в дочерней таблице может соответствовать больше одной записи в родительской таблице.

Целостность данных в реляционной модели обеспечивается двумя правилами :

- целостность сущности представляется первичным ключом ;
- целостность связей (представляется внешним ключом) .

В каждой таблице БД может существовать первичный ключ. Под первичным ключом понимают поле или набор полей, однозначно идентифицирующий запись.

Для обеспечения ссылочной целостности в дочерней таблице создается внешний ключ. Во внешний ключ входят поля связи дочерней таблицы. Для связей типа “один-ко-многим” внешний ключ по составу полей должен совпадать с первичным ключом родительской таблицы.

Содержание отчета

1. Титульный лист с темой работы и вариантом задания.
2. Цель работы, задание.
3. Описание предметной области.
4. Разработка концептуальной модели ПрО (ER-диаграмма).
5. Разработка информационно-логической модели БД.
6. Результаты реализации БД в MSSQL Server.
7. Выводы по работе.

Контрольные вопросы:

1. Понятия предметной области и базы данных.
2. Преимущества использования БД при работе с информацией.
3. Классификация моделей данных. Реляционная модель данных.
4. Основные этапы проектирования БД.
5. Состав задач концептуального проектирования.
6. Основные подходы к проектированию предметной области.
7. Метод «сущность-связь». Диаграммы «сущность-связь».
8. Типы связей между таблицами.
9. Целостность данных.
10. Понятия первичного и внешнего ключей.

Практическое занятие №2
Проектирование концептуальной модели БД.

Сущности и атрибуты

Каждая сущность является множеством подобных индивидуальных объектов, называемых экземплярами. Каждый экземпляр индивидуален и должен отличаться от всех остальных

экземпляров. Атрибут выражает определенное свойство объекта. С точки зрения БД (физическая модель) сущности соответствует таблица, экземпляру сущности – строка в таблице, а атрибуту – колонка таблицы.

Построение модели данных предполагает определение сущностей и атрибутов, т.е. необходимо определить, какая информация будет храниться в конкретной сущности или атрибуте. Сущность можно определить как объект, событие или концепцию, информация о которой должна сохраняться. Сущности должны иметь наименование с четким смысловым значением, именоваться существительным в единственном лице, не носить «технических» наименований и быть достаточно важными для того, чтобы их моделировать. Именование сущности в единственном числе облегчает в дальнейшем чтение модели. Фактически имя сущности дается по имени ее экземпляра.

Каждая сущность должна быть полностью определена с помощью текстового описания. Каждый атрибут хранит информацию об определенном свойстве сущности, а каждый экземпляр сущности должен быть уникальным. Атрибут или группа атрибутов, которые идентифицируют сущность, называется первичным ключом. При установлении связей между сущностями атрибуты первичного ключа родительской сущности мигрируют в качестве внешних ключей в дочернюю сущность.

Очень важно дать атрибуту правильное имя. Атрибуты должны именоваться в единственном числе и иметь четкое смысловое значение. Соблюдение этого правила позволяет частично решить проблему нормализации данных уже на этапе определения атрибутов.

Связи

Связь является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой. Имя связи выражает некоторое ограничение или бизнес-правило и облегчает чтение построенной модели данных.

Различают зависимые и независимые сущности. Тип сущности определяется ее связью с другими сущностями. Идентифицирующая связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями. При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности переносятся в состав первичного ключа дочерней сущности. Эта операция дополнения атрибутов дочерней сущности при создании

связи называется миграцией атрибутов. В дочерней сущности атрибуты помечаются как внешний ключ (FK).

При установлении не идентифицирующей связи дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав не ключевых компонентов родительской сущности. Не идентифицирующая связь служит для связывания независимых сущностей.

Имя связи – фраза, характеризующая отношение между родительской и дочерней сущностями. Для связи один-ко-многим идентифицирующей или не идентифицирующей достаточно указать имя, характеризующее отношение от родительской к дочерней сущности.

Тип связи (идентифицирующая/неидентифицирующая). Для не идентифицирующей связи можно указать обязательность. В случае обязательной связи атрибут внешнего ключа получит признак NOT NULL, несмотря на то, что внешний ключ не войдет в состав первичного ключа дочерней сущности. В случае необязательной связи внешний ключ может принимать значение NULL. Необязательная не идентифицирующая связь помечается прозрачным ромбиком со стороны родительской сущности.

Правила ссылочной целостности – логические конструкции, которые выражают бизнес-правила использования данных и представляют собой правила вставки, замены и удаления.

Информацию о предметной области суммируют составлением спецификаций по сущностям, атрибутам и отношениям с использованием графических диаграмм, в чем и заключается процесс моделирование данных.

Порядок выполнения работы

Для запуска пакета Power Designer в меню программы (Windows) найдите папку Sybase и запустите файл Power Designer. Для создания концептуальной модели данных необходимо выбрать File/ New или на панели инструментов выбрать значок  . Далее появится окно для выбора создаваемой модели (рис. 24), в котором надо выбратьConceptual Data Model

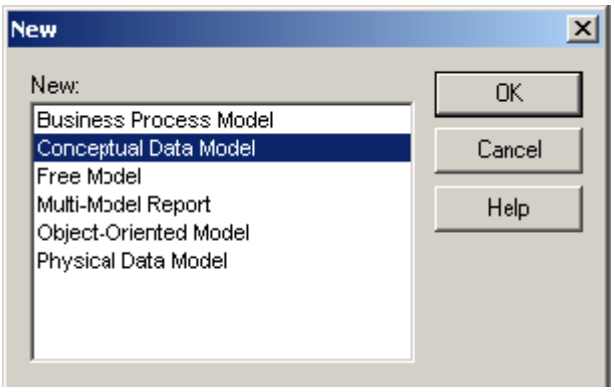


Рис. 24. Окно выбора модели.

После нажатия кнопки OK появиться окно, в котором создается ER-диаграмма.

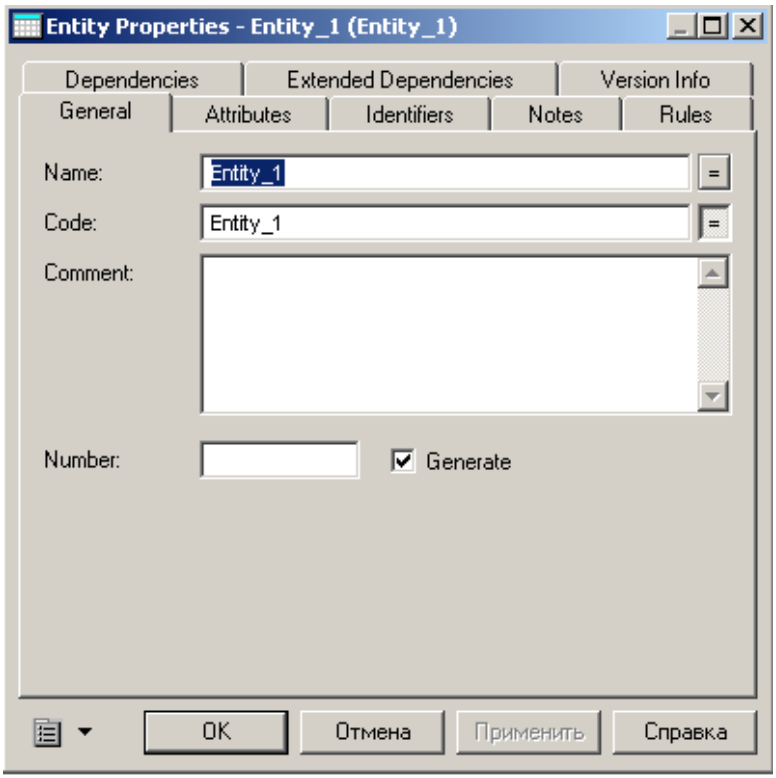
Создание сущностей

Для создания сущности, в панели TOOLS нажмите кнопку с белым прямоугольником (с подсказкой Entity).

Панель элементов с выбранным элементом сущность

Далее, поместите указатель мыши на рабочее поле в нужном месте и щелкните кнопкой мыши. Прямоугольник, изображающий сущность появится в указанном месте. При этом, курсор мыши на рабочем поле выглядит как выбранный элемент, т.о. можно создавать несколько выбранных элементов одного типа без повторного их выбора на панели элементов.

Для того, чтобы изменить свойства созданной сущности, дважды щелкните на нее левой кнопкой мыши или нажмите правую кнопку и в выпавшем меню, выберите пункт Properties, в результате чего откроется окно свойств сущности.



Окно свойств сущности

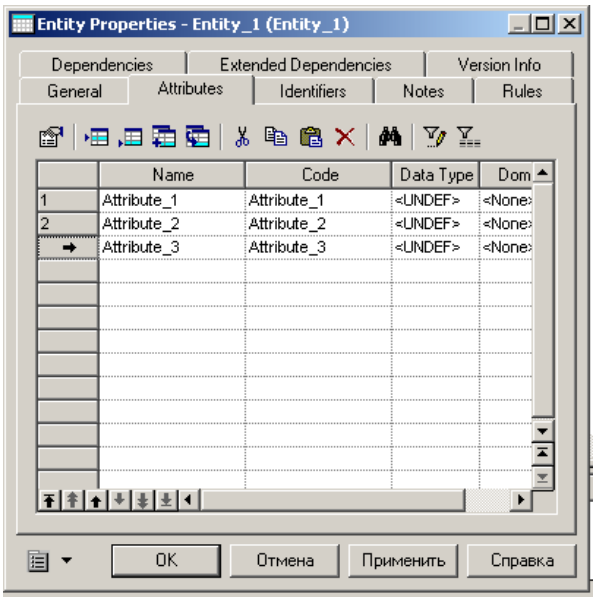
В открывшемся окне восемь закладок.

Description и Annotation предназначены для словесного описания

сущности (что улучшает понимание модели).

Закладка General позволяет ввести следующие параметры:

- Name — имя сущности, которое будет видеть пользователь;
- Code — имя кода сущности, которое будет использоваться при генерации физической модели;
- Number — ограничение количества записей в таблице после генерации физической модели;
- Comment — комментарий, предназначенный для улучшения понимания модели.



Окно ввода атрибутов сущностей

Закладка Attributes содержит таблицу и позволяет определять ***атрибуты*** сущности:

- Name — имя атрибута, которое будет видеть пользователь;
- Code — имя кода атрибута, которое будет использоваться при генерации физической модели;
- Data type — тип данных атрибута, который может быть выбран из выпадающего списка или вручную, щелкнув в поле Data type;
- Domain — принадлежность к домену, если он определен. Использование доменов позволяет, определив один раз пользовательский тип данных, использовать его в дальнейшем при определении типа данных атрибута. О создании домена будет сказано ниже;
- M (mandatory) — обязательный атрибут, указывает может ли данный атрибут принимать неопределенные значения (обязательно ли данное поле для заполнения в таблице БД);
- P (Primary Identifier) — первичный идентификатор сущности (в физической модели данных атрибут будет являться первичным ключом или его составной частью);

- D (Displayed) — отображаемый, т.е. будет ли атрибут показываться в модели.

Более полную информацию по свойствам атрибута можно получить, дважды щелкнув по полю, расположенному слева от поля с именем атрибута. Здесь можно вводить комментарий в поле Comment, задавать список значений для данного атрибута, определять верхние и нижние границы значений

Закладка Identifiers содержит автоматически заполняемую таблицу первичных идентификаторов сущности, но позволяет делать это вручную, когда необходимо создать суррогатный первичный идентификатор.

Закладка Rules позволяет вводить необходимые правила на ввод значений в таблицу.

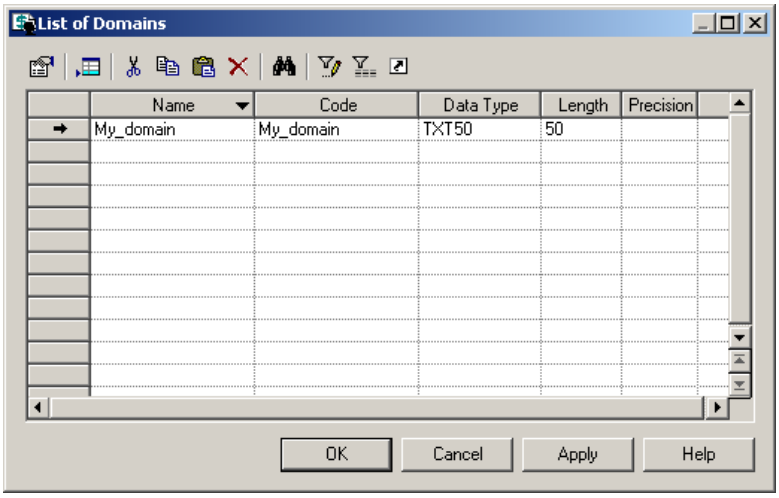
Остальные закладки Notes, Version info носят описательный характер для улучшения понимания модели.

Для фиксации всех изменений в необходимо нажать кнопку Apply.

Домен

Домен – это множество допустимых значений атрибута определенного типа данных. Домен определяется заданием стандартного типа данных, к которому относятся элементы домена и заданием произвольного логического выражения, применяемому к этому типу данных.

Для создания домена необходимо в главном меню выбрав пункт Model, выбрать пункт Domains. На рисунке показано форма определения домена. Данная форма содержит аналогичные поля как для определения свойств атрибутов (в закладке Attributes в свойствах формы), а также дополнительные поля Length и Precision для описания длины и точности значений атрибутов.

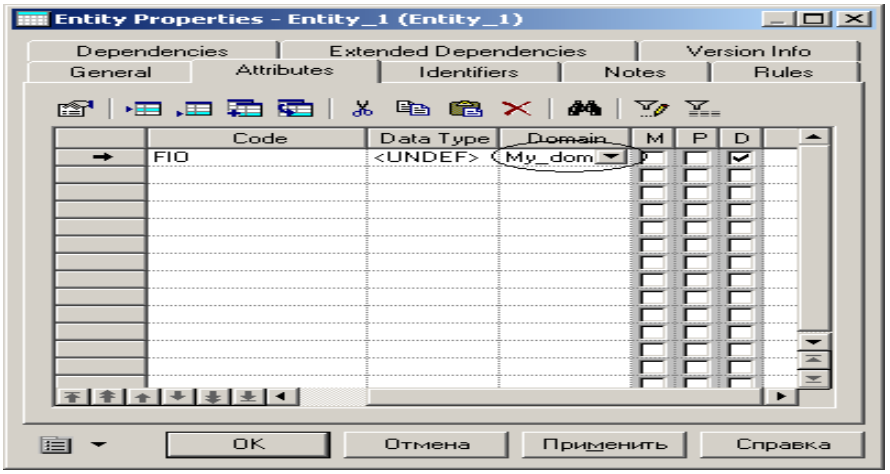


После создания нового домена его имя появится в списке доменов при задании свойств атрибутов сущностей.

При создании атрибутов сущностей

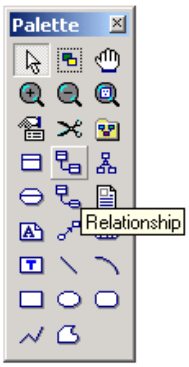
в концептуальной модели не создаются атрибуты, являющиеся внешними ключами сущностей.

После того, как созданы все необходимые сущности и атрибуты, необходимо определить связи между ними.



Установка связей

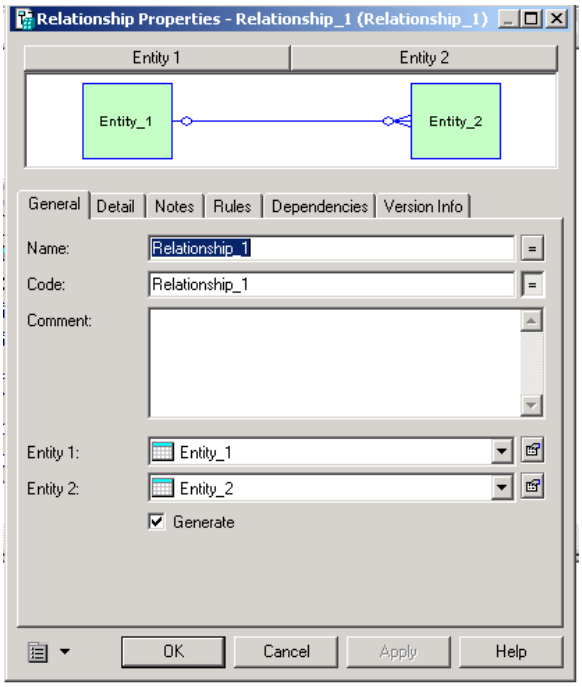
Для установки связи между двумя сущностями, необходимо нажать кнопку



Необходимо перевести курсор мыши на одну сущность и, нажав левую кнопку мыши и, не отпуская ее, перевести курсор на вторую сущность. Далее можно отпустить кнопку мыши – связь установлена.

Для изменения свойств связи, необходимо дважды щелкнуть левой кнопкой мыши на линию связи (или нажать правую кнопку мыши и выпавшем меню выбрать пункт Properties). Откроется окно, с закладками:

- закладки Notes и Version info используются для подробного описания связи. В закладке Rules можно задавать параметры ограничения связи;
- в закладке General указывается имя и код связи, а две кнопки с именами используемых сущностей позволяют вызвать окно со свойствами соответствующей сущности.
- Закладка Detail позволяет указать вид связи (один–к–одному, один–ко–многим, многие–ко–многим и т.д.) и устанавливает свойства связи от Сущности1 к Сущности2 и наоборот:
- Mandatory определяет обязательность связи, показывая, что экземпляр Сущности1 (запись) может существовать только при наличии соответствующего экземпляра в Сущности2;



- Dependent показывает, что каждый экземпляр Сущности1 отождествляется с экземпляром в Сущности2 (первичный ключ на стороне «один» при создании физической модели войдет в состав первичного ключа в таблице на стороне «многие»);
- Role – текст, описывающий связь от Сущности1 к Сущности2.

Связи многие-ко-многим преобразуются в физической модели в промежуточные таблицы.

После того, как созданы все сущности, указаны атрибуты и установлены все связи необходимо проверить концептуальную правильность построения концептуальной модели. Для этого необходимо выбрать в меню Tools/Check Model (или нажать F4). Появится окно (рис. 32), в котором предлагается выбрать объекты для проверки.

Package – система проверит правильность циклических связей.

Domain – система проверит правильность заполнения доменов.

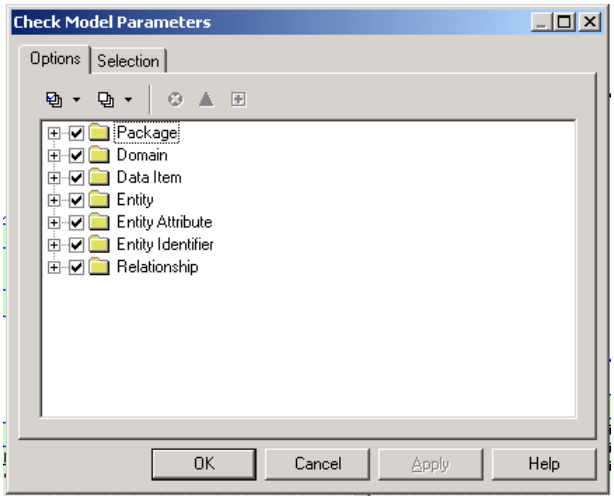
Data items – проверять ли атрибуты.

Entities – система проверит правильность создания сущностей.

Entity attributes – проверка правильности свойств сущности

Entity identifier - проверка правильности идентификаторов сущности.

Relationships – проверка связей.



После нажатия кнопки ОК система проверит всю концептуальную модель, выдаст ошибки (или предупреждения), если таковые имеются. Для просмотра сведений об ошибке необходимо дважды «щелкнуть» по ней кнопкой мыши.

Работа считается полностью выполненной, если при проверке модели не выдаются ошибки.

Практическое занятие №3
Формализация реляционной модели

С точки зрения теории реляционных баз данных, существуют следующие понятия.

Основные элементы реляционной модели

Отношение — плоская таблица, состоящая из столбцов и строк.

Атрибут — поименованный столбец отношения.

Домен — набор допустимых значений для одного или нескольких атрибутов.

Кортеж — строка отношения.

Степень — количество атрибутов в отношении.

Кардинальность — количество кортежей, которое содержит отношение.

Реляционная БД — набор нормализованных отношений.

Свойства отношений. Реляционные ключи

Ключ отношения — атрибут или набор атрибутов, значения которых могут однозначным образом идентифицировать кортеж данного отношения. Ключ отношения должен удовлетворять следующим условиям:

- **Условие уникальности** — в отношении не может быть двух кортежей с одинаковым набором значений ключевых атрибутов;
- **Условие минимальности** — из ключа нельзя исключить ни одного атрибута без потери условия уникальности.

В отношении может быть несколько ключей, из которых складывается множество потенциальных ключей отношения.

Первичный ключ — потенциальный ключ, который выбран для уникальной идентификации кортежей внутри отношения.

Внешний ключ — атрибут или множество атрибутов внутри отношения, которое соответствует потенциальному ключу некоторого (возможно, того же самого) отношения и служит для связи отношений между собой.

Реляционная целостность

Определитель NULL — значение атрибута в данный момент неизвестно или неприемлемо.

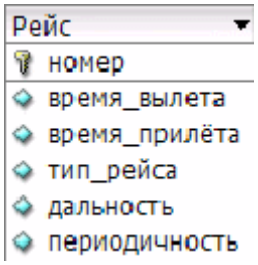
Целостность сущностей — в базовом отношении ни один атрибут первичного ключа не может содержать отсутствующих значений, обозначаемых оператором NULL.

Ссылочная целостность — значение внешнего ключа должно соответствовать значению потенциального ключа некоторого кортежа либо задаваться определителем NULL.

Согласно определению, реляционная база данных представляет собой набор нормализованных отношений (таблиц). Под «нормализованными отношениями» понимают соответствие отношений определённому набору правила, а процесс приведения модели БД в нормализованную форму называют нормализацией. Нормализация будет подробно рассмотрена в следующей части работы.

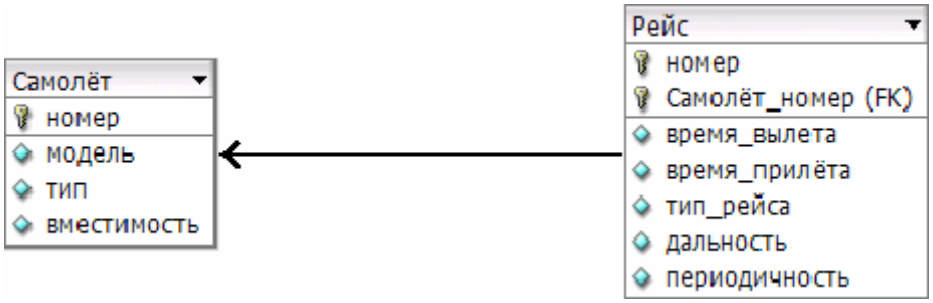
Перед тем как заняться нормализацией, сформируем из ER-диаграммы набор таблиц. На диаграмме логической модели базы данных присутствуют следующие элементы.

Таблицы, представленные в виде прямоугольника, в верхней части которого располагается название таблицы, отчёркнутое линией, а в нижней — список полей таблицы с указанием ключей:



Помимо первичных ключей, рассмотренных нами в ER-моделировании, логическая модель включает в себя внешние, либо вторичные ключи, которые необходимы для реализации связей между таблицами.

Связи между таблицами изображаются в виде прямых, ломаных или кривых линий со стрелками, ведущих от внешних к первичным ключам связываемых таблиц:



Правила преобразования ER-диаграммы в логическую модель

Для построения такой диаграммы существуют следующие, достаточно несложные правила:

1. Сущность становится таблицей с соответствующим именем.
2. Атрибут становится столбцом с таким же именем.
3. Связи типа 1:1 преобразуются одним из следующих вариантов:
1. Связанные сущности-таблицы сливаются в одну таблицу.
2. В одну из таблиц добавляется столбец-внешний ключ, содержащий ссылку-значение на первичный ключ другой таблицы.
3. В обе таблицы добавляются столбцы-внешние ключи, содержащие ссылки на первичные ключи других таблиц.
4. Связи типа 1:N реализуются в виде добавления столбца внешнего ключа в ту таблицу, которой соответствует N единиц сущностей отношения. Такая таблица называется подчинённой, или дочерней, а таблица, соответствующая одной сущности отношения — родительской или главной.
5. Связи типа M:N реализуются с помощью дополнительно вводимой вспомогательной таблицы, используемой лишь для хранения пар внешних ключей, ссылающихся на первичные ключи связываемых таблиц. Обычно такая таблица получает составное название из названий связываемых ею таблиц вида «Таблица1_Таблица2».

Рассмотрим примеры применения таких правил.

Преобразование для отношения 1:1

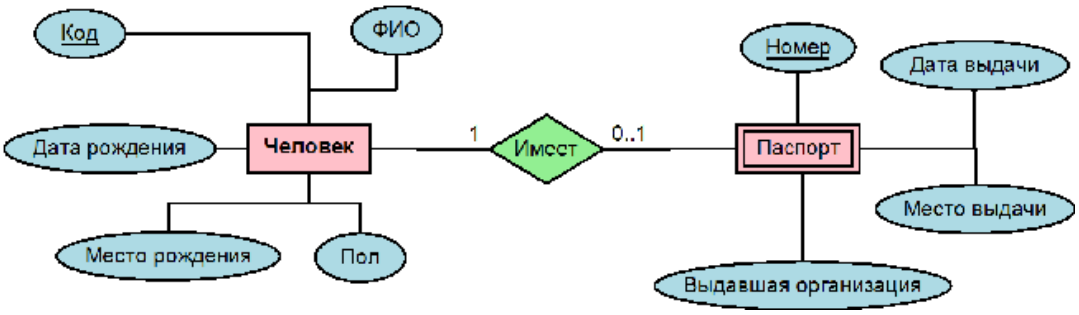
Пример «Человек и паспорт»

Текстовое описание задачи:

1. Человек характеризуется ФИО, Датой рождения, Местом рождения, Полом. 2. Человек может иметь Паспорт, если ему исполнилось 14 лет. 3. Паспорт характеризуется Номером, Датой выдачи, Местом выдачи, Выдавшей организацией.

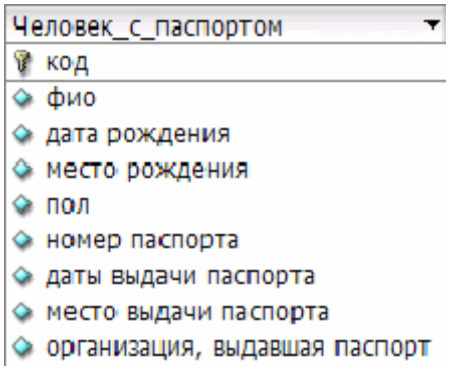
Для идентификации Человека необходимо ввести уникальный ключ — например, числовой Код.

ER-диаграмма:



При преобразовании такой модели мы можем воспользоваться одним из 3-х вышеупомянутых вариантов. Попробуем выбрать наиболее подходящий.

Самый простой вариант — объединить обе сущности в одну таблицу. Он особенно подходит для тех случаев, когда по условиям задачи всегда существуют обе сущности, входящие в связь.



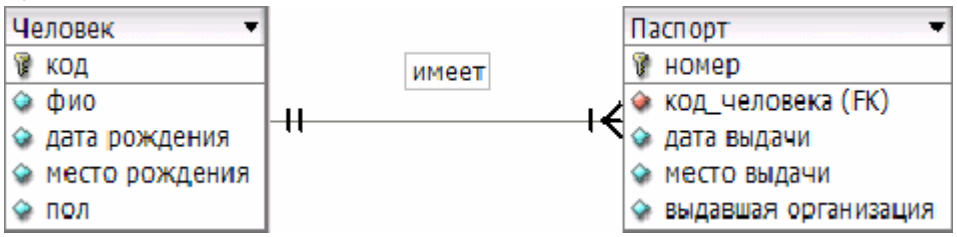
Однако для данного примера это не так — человек может существовать без паспорта, а паспорт без соответствующего ему человека смысла не имеет. В такой ситуации сущность

Паспорт называется «зависимой», т.к. её существование зависит от существования сущности Человек.

Хранение обеих сущностей для связи 1:1 в одной таблице хорошо тем, что исключаются

затраты разработчика на поддержку связей и работу с ними.

В нашем же случае больше подойдёт 2-й способ — добавить в зависимую таблицу Паспорт поле Код_человека — внешний ключ на таблицу Человек:



Таким образом, имея в таблице Паспорт код человека, мы по паспорту можем установить

человека. Говорят, что существует возможность «навигации», перемещения от сущности

Паспорт к сущности Человек, что так или иначе показывает стрелка на диаграмме.

В то же время, работая с конкретным экземпляром сущности Человек (строкой таблицы Человек), можно получить информацию о его паспортных данных только проходя полностью по всей таблице Паспорт и проверяя совпадение Кода_человека с соответствующем значением кода данного экземпляра Человека. Т.е. прямая «навигация» в направлении «Человек —> Паспорт» в такой модели отсутствует.

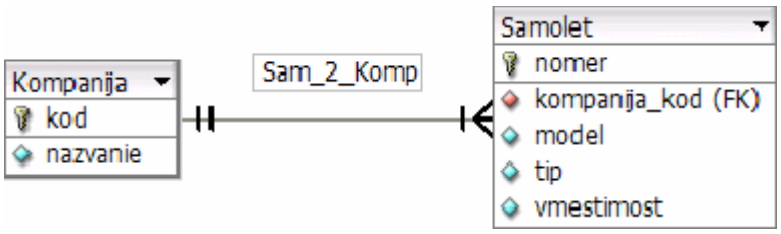
Третий способ, в котором обе таблицы обзаводятся взаимными ссылками хорош тем, что в схеме БД будет существовать возможность прямой двухсторонней навигации от Паспорта к Человеку и наоборот. Однако в случае смены человеком паспорта затраты на обновление обоих внешних ключей явно выше, чем в случае 1-й общей таблицы (1-й вариант) или односторонней внешней ссылки (2-й вариант).

Интересно, что при замене паспорта при использовании 1-го варианта организации хранения данных, информация о старом паспорте будет теряться, перетираться новыми данными, а при использовании вариантов 2 и 3 можно будет создать новую запись для паспорта, а запись, соответствующую старому паспорту, сделать неактивной, например введя у Паспорта технический атрибут «статус_блокировки» и установив его значение в «заблокирован».

Преобразование для отношения 1:N

Такой тип отношения встречается наиболее часто, более того, более сложные типы отношений, M:N, в процессе построения логической модели БД всегда сводят к 2-м отношениям типа 1:N.

Преобразование для такого типа отношения производится однозначным добавлением внешней ссылки в дочернюю таблицу. Например, для нашей задачи «Аэропорт» для отношения между сущностями Авиакомпания и Самолёт мы получим следующую диаграмму:



Здесь и далее для задачи «Аэропорт» мы будем использовать транслитерированные имена таблиц, столбцов и внешних ключей. Это

связано с возможными проблемами при использовании русскоязычных имён в различных системах управления базами данных.

Аналогичным образом преобразуются все прочие отношения 1:N ER-модели «Аэропорт».

Преобразование для отношения N:M

Поскольку технически невозможно, а точнее — крайне неэффективно хранить информацию о двух наборах сущностей, связанных между собой отношением типа M:N в одной или даже двух таблицах, возникает необходимость создавать промежуточную вспомогательную таблицу, содержащую пары ключей на обе таблицы, таким образом разрешая одно отношение типа M:N в два отношения типа 1:N.

В отношениях таких типов, очень часто само отношение, а не только сущности, может

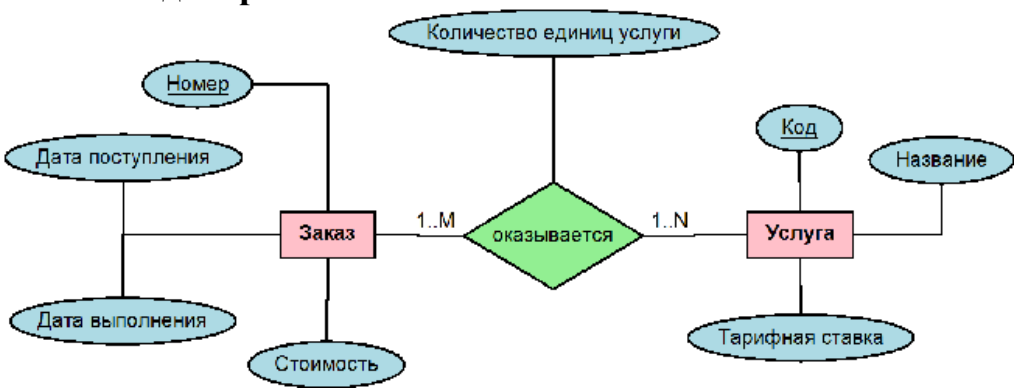
обладать атрибутами.

Пример «Заказы и услуги»

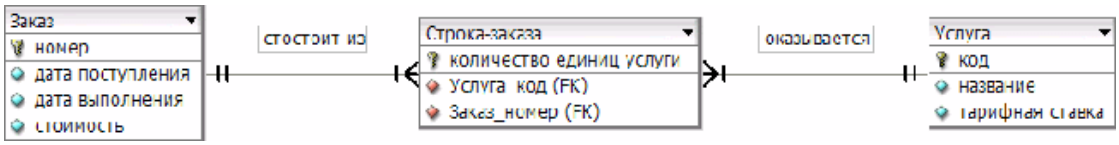
Текстовое описание задачи:

1. Заказ характеризуется номером, датой поступления, датой выполнения и суммой заказа.
2. В процессе выполнения заказа выполняются различные услуги. Одна и та же услуга может оказываться в процессе выполнения разных заказов. Таким образом, одному заказу соответствует набор услуг, и каждой услуге соответствует набор заказов — это и есть определение отношения «многие-ко-многим», M:N.
3. Услуга характеризуется кодом, названием и тарифной ставкой за единицу.
4. Процесс оказания услуги в рамках выполнения конкретного заказа также характеризуется определённым выполненным количеством единиц услуги, например, количеством листов.

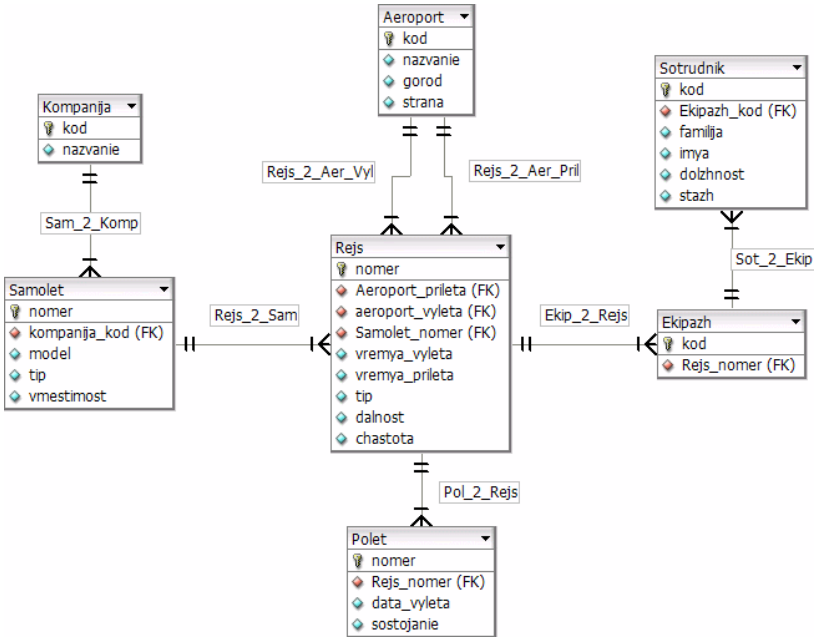
ER-диаграмма:



Согласно правилу преобразования для отношения M:N введём промежуточную таблицу Заказ-Услуга или Строка_заказа, которая будет содержать составной первичный ключ из двух внешних ключей-ссылок на таблицы Заказ и Услуга и атрибут Количество единиц услуги:



Полная логическая реляционная модель базы данных для задачи «Аэропорт» будет выглядеть следующим образом:



Задание: Создайте логическую реляционную модель БД из концептуальной модели анализа (ER-диаграммы), используя правила преобразования.

Практическое занятие №4

Проектирование Баз данных Проектирование структуры базы данных. Нормализация таблиц.

Организация данных

Слово "реляционная" происходит от английского relation - отношение. Отношение - тематическое понятие, но в терминологии моделей данных отношения удобно изображать в виде таблицы. При этом строки таблицы соответствуют кортежам отношения, а столбцы - атрибутам. Ключом называют любую функцию от атрибутов кортежа, которая может быть использована для идентификации кортежа. Такая функция может быть значением одного, из атрибутов (простой ключ),

задаваться алгебраическим выражением, включающим значения нескольких атрибутов (составной ключ). Это означает, что данные в строках каждого из столбцов составного ключа могут повторяться, но комбинация данных каждой строки этих столбцов является уникальной. Например, в таблице Студенты есть столбцы Фамилии и Год рождения. В каждом из столбцов есть некоторые повторяющиеся данные, т.е. одинаковые фамилии и одинаковые года рождения. Но если студенты, имеющие одинаковые фамилии, имеют разные года рождения, то эти столбцы можно использовать в качестве составного ключа. Как правило, ключ является уникальным, т.е. каждый кортеж определяется значением ключа однозначно, но иногда используют и неуникальные ключи (ключи с повторениями). В локализованной (русифицированной) версии Access вводится термин ключевое поле, которое можно трактовать как первичный ключ.

В Access можно выделить три типа ключевых полей: простой ключ, составной ключ и внешний ключ.

Одно из важнейших достоинств реляционных баз данных состоит в том, что вы можете хранить логически сгруппированные данные в разных таблицах и задавать связи между ними, объединяя их в единую базу. Для задания связи таблицы должны иметь поля с одинаковыми именами или хотя бы с одинаковыми форматами данных. Связь между таблицами устанавливает отношения между совпадающими значениями в этих полях. Такая организация данных позволяет уменьшить избыточность хранимых данных, упрощает их ввод и организацию запросов и отчетов. Поясним это на примере. Допустим, вам в базе надо хранить, данные о студентах (фамилия, изучаемая дисциплина) и преподавателях (фамилия, номер кафедры, ученая степень, преподаваемая дисциплина). Если хранить данные в одной таблице, то в строке с фамилией студента, изучающего конкретную дисциплину, будут храниться все атрибуты преподавателя, читающего эту дисциплину. Это же огромная избыточность данных. А если хранить данные о студенте в одной таблице, о преподавателе - в другой и установить связь между полями "Читаемая дисциплина" - "Изучаемая дисциплина" (фактически это одинаковые поля), то избыточность хранимых данных многократно уменьшится без ущерба для логической организации информации.

В Access можно задать три вида связей между таблицами; Один-ко-многим,, Многие-ко-многим и Один-к-одному.

Связь Один-ко-многим - наиболее часто используемый тип связи между таблицами. В такой связи каждой записи в таблице А может

соответствовать несколько записей в таблице В (поля с этими записями называют внешними ключами), а запись в таблице В не может иметь более одной соответствующей ей записи в таблице А.

При связи Многие-ко-многим одной записи в таблице А может соответствовать несколько записей в таблице В, а одной записи в таблице В - несколько записей в таблице А. Такая схема реализуется только с помощью третьей (связующей) таблицы, ключ которой состоит по крайней мере из двух полей, одно из которых является общим с таблицей А, а другое - общим с таблицей В.

При связи Один-к-одному запись в таблице А может иметь не более одной связанной записи в таблице В и наоборот. Этот тип связи используют не очень часто, поскольку такие данные могут быть помещены в одну таблицу. Связь с отношением Один-к-одному применяют для разделения очень широких таблиц, для отделения части таблицы в целях ее защиты, а также для сохранения сведений, относящихся к подмножеству записей в главной таблице.

Тип создаваемой связи зависит от полей, для которых определяется связь:

- связь Один-ко-многим создается в том случае, когда только одно из полей является ключевым или имеет уникальный индекс, т.е. значения в нем не повторяются;
- связь Один-к-одному создается в том случае, когда оба связываемых поля являются ключевыми или имеют уникальные индексы;
- связь Многие-ко-многим фактически представляет две связи типа один-ко-многим через третью таблицу, ключ которой состоит, по крайней мере, из двух полей, общих для двух других таблиц.

Целостность данных

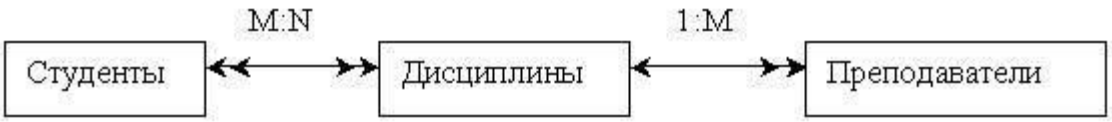
Целостность данных означает систему правил, используемых в СУБД Access для поддержания связей между записями в связанных таблицах (таблиц, объединенных с помощью связи), а также обеспечивает защиту от случайного удаления или изменения связанных данных. Контролировать целостность данных можно, если выполнены следующие условия:

- связанное поле (поле, посредством которого осуществляется связь) одной таблицы является ключевым полем или имеет уникальный индекс;

- связанные поля имеют один тип данных. Здесь существует исключение. Поле счетчика может быть связано с числовым полем, если оно имеет тип Длинное целое,
- обе таблицы принадлежат одной базе данных Access. Если таблицы являются связанными, то они должны быть таблицами Access. Для установки целостности данных база данных, в которой находятся таблицы, должна быть открыта. Для связанных таблиц из баз данных других форматов установить целостность данных невозможно.

1. Перед разработкой информационно-логической модели реляционной базы данных рассмотрим, из каких информационных объектов должна состоять эта база данных. Можно выделить три объекта, которые не будут обладать избыточностью, - Студенты, Дисциплины и Преподаватели. Представим состав реквизитов этих объектов в виде "название объекта (перечень реквизитов)": Студенты (код студента, фамилия, имя, отчество, номер группы, дата рождения, стипендия, оценки). Дисциплины (код дисциплины, название дисциплины), Преподаватели (код преподавателя, фамилия, имя, отчество, дата рождения, телефон, заработная плата).

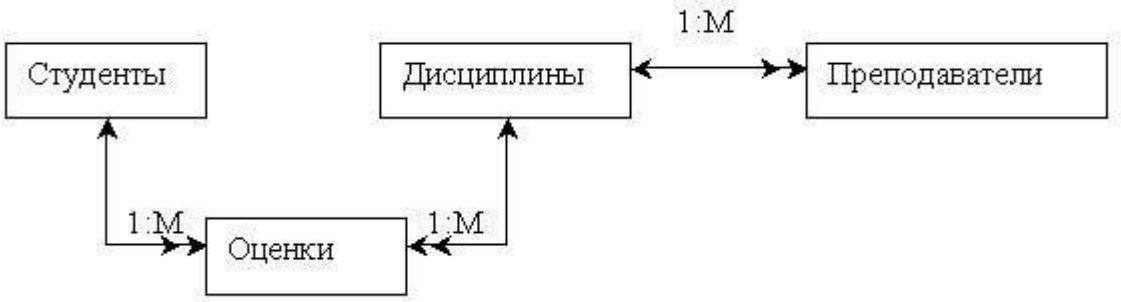
Рассмотрим связь между объектами Студенты и Дисциплины. Студент изучает несколько дисциплин, что соответствует многозначной связи и отражено на рис.1 двойной стрелкой. Понятно, что каждая дисциплина изучается множеством студентов. Это тоже многозначная связь, обозначаемая двойной стрелкой (связь "один" обозначена одинарной стрелкой). Таким образом, связь между объектами Студенты и Дисциплины - Многие-ко-многим (M : N).



Типы связей между объектами Студенты, Дисциплины и Преподаватели

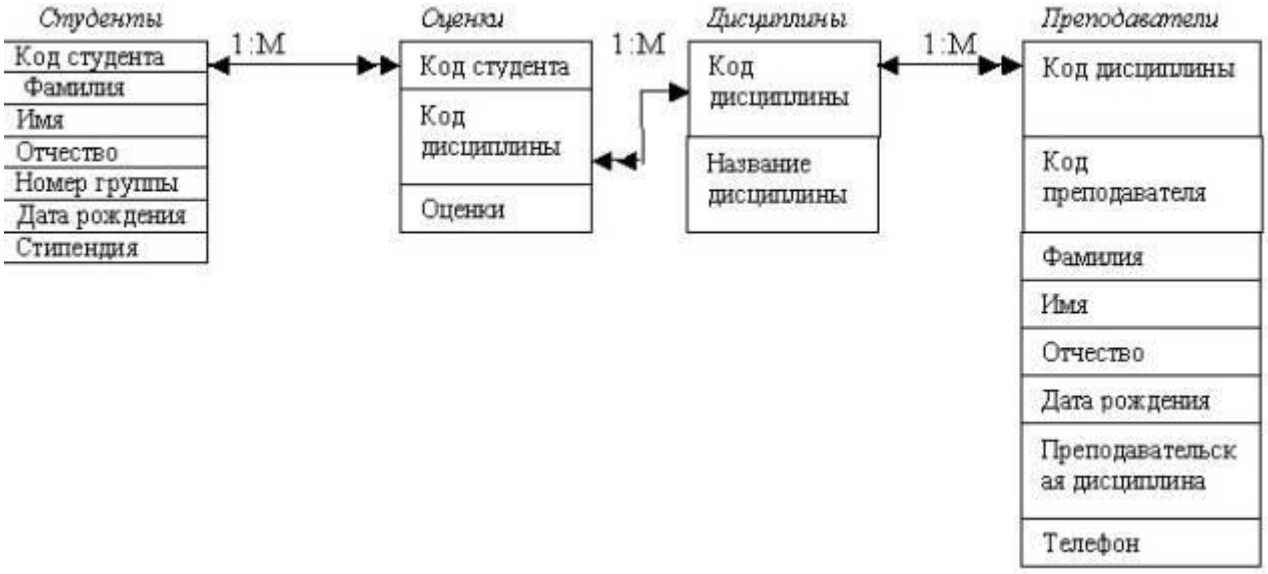
Множественные связи усложняют управление базой данных, например, в СУБД Access при множественных связях нельзя использовать механизм каскадного обновления. Поэтому использовать такие связи нежелательно и нужно строить реляционную модель, не содержащую связей типа Многие-ко-многим. В Access дня контроля целостности данных с возможностью каскадного обновления и

удаления данных необходимо создать вспомогательный объект связи, который состоит из ключевых реквизитов связываемых объектов и который может быть дополнен описательными реквизитами. В нашем случае таким новым объектом для связи служит объект Оценки, реквизитами которого являются код студента, код дисциплины и оценки. Каждый студент имеет оценки по нескольким дисциплинам, поэтому связь между объектами Студенты и Оценки будет Один-ко-многим (1:М). Каждую дисциплину сдает множество студентов, поэтому связь между объектами Дисциплины и Оценки также будет Один-ко-многим (1:М). В результате получаем информационно-логическую модель базы данных, приведенную на рис. 2



Информационно-логическая модель реляционной базы данных

2. В реляционной базе данных в качестве объектов рассматриваются отношения, которые можно представить в виде таблиц. Таблицы между собой связываются посредством общих полей, т.е. одинаковых по форматам и, как правило, по названию, имеющихся в обеих таблицах. Рассмотрим, какие общие поля надо ввести в таблицы для обеспечения связности данных. В таблицах Студенты и Оценки таким полем будет "Код студента", в таблицах Дисциплины и Оценки - "Код дисциплины", в таблицах Преподаватели и Дисциплины - "Код дисциплины". Выбор цифровых кодов вместо фамилий или названий дисциплин обусловлен меньшим объемом информации в таких полях: например, число "2". по количеству символов значительно меньше слова "математика". В соответствии с этим логическая модель базы данных представлена на рис. 3, где жирными буквами выделены ключевые поля.



Варианты заданий

Вариант 1

Описание предметной области (Ресторан)

Посетители ресторана обслуживаются за столиками. За одним столом может располагаться не более 4 посетителей, каждый из которых может сделать заказ тех или иных блюд. Столики обслуживают официанты. У одного официанта в обслуживании несколько столов.

Задачи для БД:

Есть ли свободные столы? Сколько посетителей обслужил официант за смену? Сколько каких блюд было реализовано?

Вариант 2

Описание предметной области (Колледж)

Студенты колледжа объединены в группы. Набор дисциплин, изучаемых студентом, зависит от номера группы в которой он учится. Преподаватели читают дисциплины и выставляют зачеты студентам. Один преподаватель может читать несколько дисциплин, но каждую дисциплину ведет один преподаватель. Задачи для БД: Какие дисциплины изучает студент? Какая оценка у студента по данной дисциплине? Кто выставил эту оценку?

Вариант 3

Описание предметной области (Театральная касса)

В театральной кассе продаются билеты на спектакли. Стоимость билета зависит от ряда, театра и спектакля. Каждый день в театре может идти не более одного спектакля. Спектакль характеризуется названием и автором. Каждый покупатель может купить сколько угодно билетов на любые спектакли.

Задачи для БД:

Какие спектакли идут в определенный день? Есть ли билеты на конкретный спектакль? Сколько стоит конкретный билет?

Вариант 4

Описание предметной области (Грузоперевозки)

АТП имеет грузовые автомобили с гос. номерами и организует перевозки для своих заказчиков. Стоимость перевозки зависит от расстояния и грузоподъемности автомобиля, который ее выполняет. Каждый заказчик может сделать заказ нескольких перевозок. Одну перевозку выполняет один грузовик.

Задачи для БД:

Какие грузовики свободны? Какой заказчик сделал самый дорогой заказ? Какой грузовик выполнил наибольшее количество заказов?

Вариант 5

Информационная система туристического клуба

Туристы, приходящие в туристический клуб, могут не только ходить в плановые походы, но и заниматься в различных секциях в течение всего года. Для этого они записываются в группы, относящиеся к определенным секциям.

Туристов можно условно разделить на любителей, спортсменов и тренеров. Каждая из перечисленных категорий может иметь свой набор характеристик-атрибутов. Секции клуба возглавляются руководителями, в функции которых входит контроль за работой секции. Руководитель секции назначает каждой группе тренера. Тренер может тренировать несколько групп, причем необязательно принадлежащих его секции.

В течение года клуб организует различные походы. Каждый поход имеет свой маршрут, на который отводится определенное количество дней. По маршруту и количеству дней определяется категория сложности данного похода. Поход возглавляет инструктор, которым может быть какой-либо тренер или спортсмен. Он набирает группу в количестве 5-15 человек для своего похода, исходя из типа похода (пеший, конный, водный, горный) и физических данных туристов (по их занятиям в секциях: водники, спелеологи, альпинисты и другие, с учетом специфики занятий - не умеющего плавать никогда не возьмут на сплав, а в пеший поход небольшой категории сложности могут взять любого туриста).

1. Для организации работы клуба необходимо: получить список и общее число туристов, занимающихся в клубе, в указанной секции, группе, по половому признаку, году рождения, возрасту; получить

список и общее число тренеров указанной секции; получить перечень и общее число маршрутов, по которым ходили туристы из указанной секции, в обозначенный период времени.

Вариант 6

Информационная система зоопарка

Служащих зоопарка можно подразделить на несколько категорий: ветеринары, уборщики, дрессировщики, строители-ремонтники, работники администрации. Каждая из перечисленных категорий работников имеет уникальные атрибуты-характеристики, определяемые профессиональной направленностью. За каждым животным ухаживает определенный круг служащих, причем только ветеринарам, уборщикам и дрессировщикам разрешен доступ в клетки к животным.

В зоопарке обитают животные различных климатических зон, поэтому часть животных на зиму необходимо переводить в отапливаемые помещения. Животных можно подразделить на хищников и травоядных. При расселении животных по клеткам необходимо учитывать не только потребности данного вида, но и их совместимость с животными в соседних клетках (нельзя рядом селить, например, волков и их добычу - различных копытных).

Для кормления животных необходимы различные типы кормов: растительный, живой, мясо и различные комбикорма. Растительный корм это фрукты и овощи, зерно и сено. Живой корм - мыши, птицы, корм для рыб. Для каждого вида животных рассчитывается свой рацион. Таким образом у каждого животного в зоопарке имеется меню на каждый день, в котором указывается количество и время кормлений в день, количество и вид пищи (обезьянам необходимы фрукты и овощи, мелким хищникам - хорькам, ласкам, совам, некоторым кошачьим, змеям - надо давать мышей).

Ветеринары должны проводить медосмотры, следить за весом, ростом, развитием животного, ставить своевременно прививки и заносить все эти данные в карточку.

Необходимо получить список и общее число служащих зоопарка; получить перечень и общее число всех животных в зоопарке либо животных указанного вида; получить перечень и общее число нуждающихся в теплом помещении на зиму; получить перечень животных, которым не сделаны прививки вовремя; составить рацион для каждого вида животных.

Контрольные вопросы:

5.1. Назовите виды моделей баз данных. Отличия, достоинства, недостатки

5.2. Назовите виды взаимосвязей между объектами баз данных. Отличия, достоинства, недостатки

5.3. Что такое нормализация таблиц, для какой цели проводят процедуру нормализации?

5.4. Перечислите этапы проектирования

Практическое занятие №5

Создание базы данных в программе MS Access, определение полей и типы данных. Нормализация таблиц.

Теоретическая часть

База данных (БД, database) поименованная совокупность структурированных данных, относящихся к определенной предметной области.

Система Access – реляционного типа, т. е. ее база данных состоит из совокупности связанных между собой таблиц. Каждая таблица имеет строгую структуру.

Таблица базы данных (table) – регулярная структура, состоящая из однотипных строк, которые называются записями (records), разбитых на поля (fields). Каждое поле записи обязательно имеет имя, тип и формат (или ширину).

Для связей между таблицами используются ключи (физическая реализация ключей – индексы).

Первичный ключ (primary key) – главный ключевой элемент, однозначно идентифицирующий запись в таблице.

В системе Access под термином Ключевое поле подразумевается первичный ключ, для других ключей (уникальных или внешних) используется атрибут Индексированное поле (Совпадения не допускаются) или Индексированное поле (Совпадения допускаются).

Главный принцип проектирования – совокупность связанных таблиц создается таким образом, чтобы суммарный объем хранимой информации был минимален, и любую информацию можно было быстро найти.

Обычно в состав базы данных входят таблицы для хранения главной информации, которые могут постоянно пополняться данными, и справочные таблицы, редко изменяющиеся.

Связи между таблицами в системе Access задаются с использованием режима Схема данных. Для связей следует задать условия соблюдения ссылочной целостности.

Ссылочная целостность данных (referential integrity) – набор правил, обеспечивающих соответствие ключевых значений в связанных таблицах.

В состав информационной системы кроме информации базы данных входят также компоненты пользовательского интерфейса, важнейшие из которых – формы и печатные отчеты. Особенность системы Access – эти компоненты могут храниться в одном файле с таблицами или в разных файлах

Все имена в БД (таблиц, полей, форм, отчетов, запросов и пр.) конечно же лучше писать с использованием латинских букв и английских слов (если система разрабатывается для международных корпораций), но в учебном примере для простоты будем пользоваться русскими названиями.

Практическая часть

Задание. Разработать структуру БД «Студенты».

1. **Проектирование и создание базы данных**

Процесс создания базы данных рассмотрим на примере разработки информационной системы «Студенты», которая должна хранить информацию о студентах и их экзаменационных оценках.

В результате проектирования был сделан вывод о необходимости создания в ней 5-ти таблиц:

1) Студенты – для хранения основных данных о студенте;

2) Оценки – для хранения информации об оценках студентов;

3) Институты – справочник институтов;

4) Специальности – справочник специальностей;

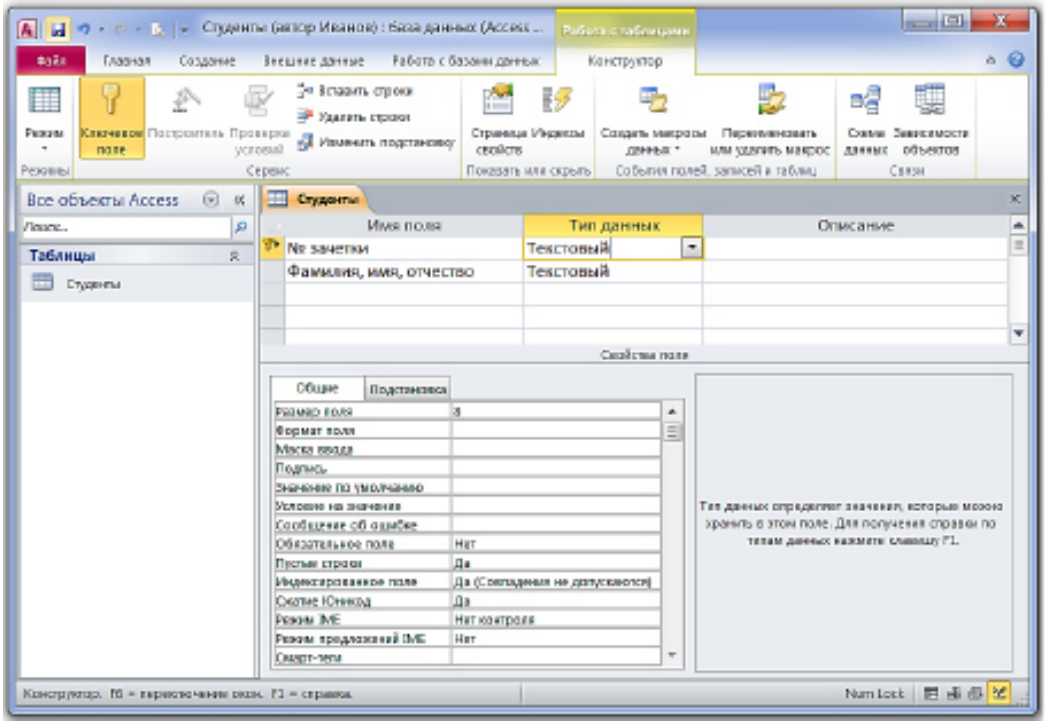
5) Предметы – справочник предметов.

Для создания файла базы данных в папке хранения Ваших файлов вызовем контекстное меню и в нем выберем команду Создать → Microsoft Access База данных. Зададим имя базы данных Студенты (автор <Ваша фамилия>). Откроем базу данных двойным щелчком на созданном файле.

2. **Описание структуры таблиц и связей**

Выберем на ленте вкладку **Создание** и в группе **Таблицы** нажмем на кнопку **Конструктор** таблиц. По умолчанию для окна базы данных установлен параметр Вкладки, поэтому внутри главного окна мы увидим вкладку (вложенное окно с ярлычком сверху) Конструктора таблиц, показанное на рисунке

(данные двух полей уже заполнены и была нажата кнопка Сохранить на верхней рамке окна).



Описание структуры таблицы Студенты в Конструкторе

Далее в Конструкторе добавим остальные поля в соответствии с данными таблицы т.е. зададим имя, тип данных, размер или формат каждого поля таблицы, а также ключевое поле (если необходимо), индексированные поля и подписи. После чего закроем вкладку Конструктора таблицы Студенты (крестиком справа на темно-серой полоске или из контекстного меню ярлычка) с сохранением изменений структуры.

Затем снова выберем команду Создание → Конструктор таблиц и опишем структуру следующей таблицы – Оценки в соответствии с данными таблицы Сохраним таблицу и закроем Конструктор данной таблицы.

Аналогично поступим при создании еще трех таблиц – Институты, Специальности и Предметы.

В результате получим в базе данных 5 пустых таблиц с заданной структурой. При необходимости в любой момент можно обратиться к модификации структуры каждой из таблиц, открыв ее в Конструкторе.

Структура таблицы **Студенты**

Таблица 9.1 – Структура таблицы **Студенты**

Имя поля	Тип данных	Размер поля	Индексированное поле
№ зачетки	Текстовый	8	**Ключевое поле**
Фамилия, имя, отчество	Текстовый	45	Нет
Дата поступления	Дата/время	Краткий формат даты	Нет
№ института	Числовой	Байт	Да (Допускаются совпадения)
Код специальности	Текстовый	9	Да (Допускаются совпадения)
Курс	Числовой	Байт	Нет
Группа	Текстовый	4	Нет

Структура таблицы **Оценки**

Таблица 9.2 – Структура таблицы **Оценки**

Имя поля	Тип данных	Размер поля	Индексированное поле	Обязательное поле
№ зачетки	Текстовый	8	Да (**Допускаются совпадения**)	Да
Семестр	Числовой	Байт	Нет	Да
№ предмета	Числовой	Целое	Да (Допускаются совпадения)	Да
Оценка	Текстовый	1	Нет	Да
Дата получения	Дата/время	Краткий формат даты	Нет	Да
Преподаватель	Текстовый	45	Нет	Да

Структура таблицы **Институты**

Таблица 9.3 – Структура таблицы **Институты**

Имя поля	Тип данных	Размер поля	Индексированное поле
№ института	Числовой	Байт	**Ключевое поле**
Название института	Текстовый	120	Нет

Структура таблицы **Специальности**

Таблица 9.1 – Структура таблицы **Специальности**

Имя поля	Тип данных	Размер поля	Индексированное поле
Код специальности	Текстовый	9	**Ключевое поле**
Название специальности	Текстовый	120	Нет

Структура таблицы **Предметы**

Таблица 9.5 – Структура таблицы **Предметы**

Имя поля	Тип данных	Размер поля	Индексированное поле
№ предмета	Числовой	Целое	**Ключевое поле**
Название предмета	Текстовый	120	Нет

Далее задаем связи (Один ко многим) между таблицами в базе. Для этого на вкладке ленты **Работа с базами данных** выбираем в группе **Отношения** команду **Схема данных**, добавляем в окно схемы все таблицы и, перетаскивая название поля первичного ключа к аналогичному полю другой таблицы создать связи. При этом задаем в окне Изменение связей для всех связей между таблицами 3 условия: обеспечения целостности данных, каскадное обновление связанных

полей и каскадное удаление связанных записей. Схема базы данных показана на рисунок

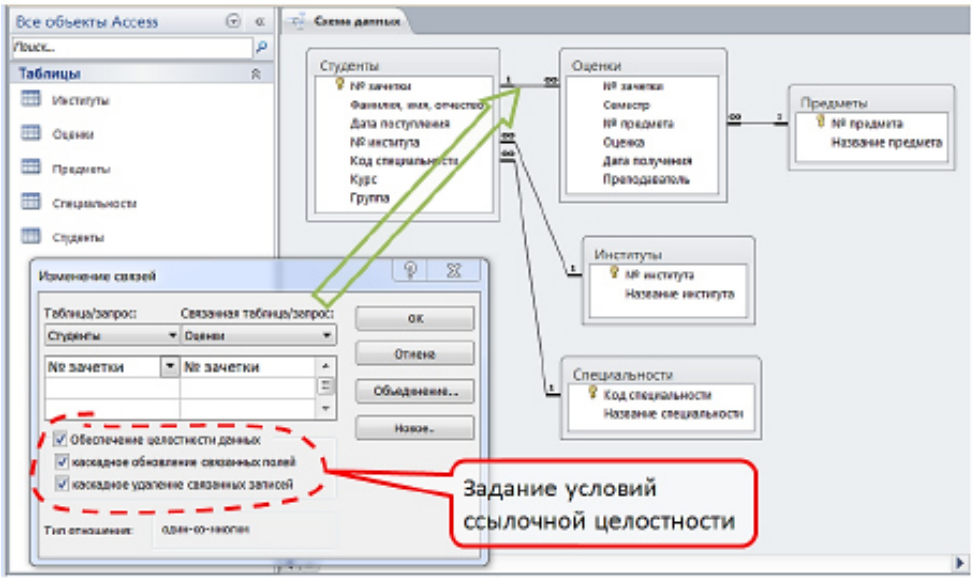


Схема базы данных и задание условий ссылочной целостности для связи между таблицами Студенты – Оценки

Практическое занятие №6

Открытие, редактирование и пополнение табличного файла.

Краткие сведения

СУБД Access использует **реляционную** модель базы данных, в которой данные представлены в виде взаимосвязанных таблиц (отношений по англ. - *relations*).

Важнейшим этапом проектирования базы данных является разработка информационно-логической ***(инфологической***) модели предметной области, не ориентированной на СУБД, но отражающей предметную область в виде совокупности информационных объектов и их информационных связей.

СУБД Access позволяет работать с объектами базы данных, к которым относятся ***таблицы, запросы, формы, отчеты, страницы доступа, макросы и модули.***

Таблицы служат для хранения данных в определенной структуре.

Запросы создаются для выборки данных из одной или нескольких связанных таблиц.

Формы предназначены для ввода, редактирования и просмотра табличных данных на экране в удобном виде.

Страницы доступа к данным представляют специальный тип

веб-страниц, предназначенный для просмотра и работы через Интернет или интрасеть с данными, хранящимися в базах данных Microsoft Access или в базах данных Microsoft SQL Server.

Отчеты являются выходными документами, предназначенными для вывода на принтер.

Макросы используются для автоматизации различных процедур обработки данных, являются программами, состоящими из макрокоманд высокого уровня. Макропрограммирование в Access не требует знания языка Visual Basic. Имеющийся в Access набор из около 50 макрокоманд обеспечивает практически любые действия, необходимые для решения задач.

Модули являются программами на языке Visual Basic, которые служат для реализации нестандартных процедур обработки данных.

Все данные БД Microsoft Access и средства их отображения хранятся в одном файле с расширением **MDB**.

Задание 1. Ознакомьтесь с **учебной базой данных** компании «**Борей»,** входящей в комплект поставки Microsoft Access

Технология

1. Загрузите **Microsoft Access.** Установите низкий уровень безопасности. Для этого выполните команду **СЕРВИС/Макрос/*Безопасность***. На вкладке *Уровень безопасности* включите переключатель *Низкая.*

2. Если *Область задач* не открыта, то включите ее, выполнив команду **ВИД/Панели инструментов/*Область задач***. В *Области задач*, которая появится в правой части открытого окна Access, в разделе *Открыть* щелкните по имени базы данных **Борей**.

3. Если *Область задач* не открыта, то включите ее, выполнив команду **ВИД/Панели инструментов/*Область задач***. В *Области задач*, которая появится в правой части открытого окна Access, в разделе *Открыть* щелкните по имени базы данных **Борей.** Можно также получить доступ к базе данных **Борей**, если выполнить команду **СПРАВКА/Примеры баз данных/*Учебная база данных «Борей»***.

4. Закройте заставку и перейдите в окно базы данных и щелкните по типу объектов: *таблицы*. Просмотрите данные каждой таблицы, открыв их.

5. Просмотрите структуру каждой таблицы в режиме **конструктора**. Обратите внимание на типы и свойства полей. Для

переключения из режима таблицы в режим конструктора используйте кнопку ***Вид*** на панели инструментов.

6. Откройте таблицу «Клиенты». Выполните следующие операции:

- Определите количество записей в таблице;
- Просмотрите 45-ую запись, введя ее номер в окно номеров записей, расположенное в внизу таблицы.
- Рассортируйте таблицу по должностям. Для этого установите указатель мыши на заголовок столбца и щелкните правой кнопкой мыши. Столбец будет выделен и появится контекстное меню. Выберите в контекстном меню пункт *Сортировка по возрастанию.*
- **Скройте** столбец *Обращаться к*, выделив его и выполнив команду **ФОРМАТ/Скрыть столбцы** Отобразите скрытый столбец. Для этого выполните команду **ФОРМАТ/Отобразить столбцы**.
- Используя клавишу ***Shift*** выделите первые два столбца и **закрепите** их, выполнив **команду ФОРМАТ/Закрепить столбцы**. Прокрутите таблицу по горизонтали. Отмените закрепление, выполнив команду **ФОРМАТ/Освободить столбцы.**.
- Примените **фильтр** для выделения строк с клиентами в г. Лондон. Для этого выделите в любой строке поле со значением «Лондон» и вызовите контекстное меню. Выберите пункт *Фильтр по выделенному*. Отмените фильтр, щелкнув в контекстном меню по пункту *Удалить фильтр*.
- Измените **вид сетки таблицы**, используя соответствующую кнопку инструментальной панели *Таблица*, если такой кнопки нет, то ее необходимо добавить на панель.

7. Просмотрите **запросы и их структуру**

- откройте запрос на выборку товаров с ценой выше средней;
- переключите запрос в режим конструктора и просмотрите структуру запроса.

8. Просмотрите **формы**:

- «Сотрудники», вкладки «Служебные данные», «Личные данные»;
- «Товары»;
- «Типы»;

9. Просмотрите **форму** «Сотрудники» в **режиме конструктора**.

10. Просмотрите **отчеты**:

- «Каталог»;
- «Продажи по типам»;

- «Суммы продаж по годам»;
- «Счет»;
- «Продажи по сотрудникам и странам», введя дату начала: 01.01.1998 и дату окончания: 31.12.1998

11. Просмотрите многостраничный **отчет** «Каталог» **в режиме конструктора**.

12. Закройте базу данных Борей, щелкнув по кнопке *Закрыть* в окне базы данных.

Таблицы

Таблицы составляют основу базы данных - именно в них хранятся все данные. Таблицы должны быть тщательно спланированы. Прежде всего, должна быть спланирована структура каждой таблицы. Структура таблиц определяется содержанием тех выходных форм и отчетов, которые должны быть затем получены. При планировании таблиц необходимо избежать дублирования информации в разных таблицах.

Таблица - это объект БД, который хранит данные определенной структуры. Таблица состоит из ***записей*** (строк), каждая из которых описывает одну сущность. Каждый столбец таблицы - это ***поле.*** Столбец содержит однотипную информацию.

Длина имени таблицы - не более 64 символов.

Длина имени поля - не более 64 символов.

Количество полей в одной таблице - не более 255.

Количество записей - неограниченно.

Суммарный объем информации во всей БД - не более 2 гигабайта.

Для каждого поля необходимо указать тип данных. Тип данных определяет вид и диапазон допустимых значений, которые могут быть введены в поле, а также объем памяти, выделяющийся для этого поля.

Таблица может содержать следующие типы полей (всего 8):

Текстовый Короткий текст. Текст и числа, например, имена и адреса, номера телефонов и почтовые индексы. Текстовое поле может содержать до 255 символов.

Поле Memo Длинный текст и числа, например, комментарии и пояснения. Memo-поле может содержать до 65 536 символов.

Числовой Общий тип для числовых данных, допускающих проведение математических расчетов, за исключением расчетов для денежных значений. Свойство Размер поля позволяет указать различные типы числовых данных. Длина - 8 байт. Точность – 15

знаков.

Дата/время Значения даты и времени. Пользователь имеет возможность выбрать один из многочисленных стандартных форматов или создать специальный формат. Длина - 8 байт.

Денежный Денежные значения. Числа представляются с двумя знаками после запятой. Не рекомендуется использовать для проведения денежных расчетов значения, принадлежащие к числовому типу данных, так как последние могут округляться при расчетах. Значения типа "Денежный" всегда выводятся с указанным числом десятичных знаков после запятой. Длина - 8 байт.

Счетчик Автоматически вставляющиеся последовательные номера. Счетчик увеличивается на единицу для каждой следующей записи. Нумерация начинается с 1. Поле счетчика удобно для создания ключа. В таблице может быть только одно такое поле. Длина - 4 байта.

Логический Значения "Да"/"Нет", "Истина"/"Ложь", "Вкл"/"Выкл", т.е. одно из двух возможных значений. Длина - 1 байт.

Поле объекта OLE Объекты, созданные в других программах, поддерживающих протокол OLE, например графики, рисунки и т.п. Объекты связываются или внедряются в базу данных Microsoft Access через элемент управления в форме или отчете. Максимальный объем информации объекта OLE -1 Гбайт.

Гиперссылка. Поле, в котором сохраняются адреса гиперссылок, позволяющих переходить к файлам, фрагментам файлов или веб-страницам. Гиперссылка может иметь вид пути UNC либо адреса URL. Сохраняет до 64 000 знаков

Индексирование полей таблицы. Индексирование позволяет ускорить сортировку и поиск данных в таблице. Можно индексировать числовые, денежные, текстовые, логические поля, а также поля типа Счетчик и Дата. Не следует создавать слишком много индексов для одной таблицы, т.к. это замедлит ввод и редактирование ее данных.

Первичный ключ - это специальный тип индекса, который однозначно идентифицирует каждую запись. В первичный ключ могут входить несколько полей, но значение первичного ключа должно быть уникальным для каждой записи. Первичные ключи используются для установления связей между таблицами.

Связи между таблицами. Таблицы могут быть связаны отношениями **один-к-одному, один-ко-многим и многие-к-многим**. Access позволяет использовать только отношения первых двух типов.

При установлении связей нужно определить, какая таблица является ***главной***, а какая - ***подчиненной***.

Отношение ***один-к-одному*** означает, что одной записи подчиненной таблицы соответствует только одна запись в главной таблице. Такие отношения встречаются очень редко, т.к. требую неоправданно много места в БД. Вместо них можно просто добавить поля подчиненной таблицы к полям главной.

Наиболее часто используются отношения ***один-ко-многим***. В этом случае одной записи в главной таблице соответствует несколько записей в подчиненной таблице.

Для создания отношений необходимо указать поля в двух таблицах, которые содержат одни и те же данные. Обычно такое поле в одной из таблиц (главной) является ключевым. Имена связывающих полей могут отличаться, но типы и свойства должны совпадать. Возможна связь между полем типа *Счетчик* и полем типа *Число* с форматом *Длинное целое*.

Рекомендации для ввода данных в таблицы

Для ввода в поле текущей записи значения из того же поля предыдущей записи нажать клавиши <**Ctrl**> и <**“**>. (Двойной апостроф на русском регистре - на клавише “2”).

Для редактирования ранее введенного значения нажимайте клавишу **F2**.

Задание 2. Создайте базу данных **«Академия»** на основе **инфологической модели**, приведенной на рисунке. База данных должна содержать 4 взаимосвязанных таблицы: *Студент, Группа, Специальность и Факультет.*

Таблица Студент:

- N зачетной книжки – ключевое поле, длинное целое
- N группы – числовое поле, целое
- ФИО – текстовое поле 15 символов
- Дата рождения – поле типа «дата»
- Коммерческий – логическое поле (вкл/выкл)

Таблица Группа:

- N группы – ключевое поле числового типа, целое
- N специальности – числовое поле, длинное целое
- N факультета – числовое поле, байтовое
- Курс – числовое поле, байтовое

Таблица Факультет:

- N факультета – ключевое поле числового типа, байтовое
- Наименование факультета – текстовое поле, 30 символов

- Декан - тестовое поле, 15 символов

Таблица Специальность:

- N специальности – ключевое поле числового типа, длинное целое
- Наименование специальности – текстовое поле, 40 символов
- Стоимость обучения – денежного типа.

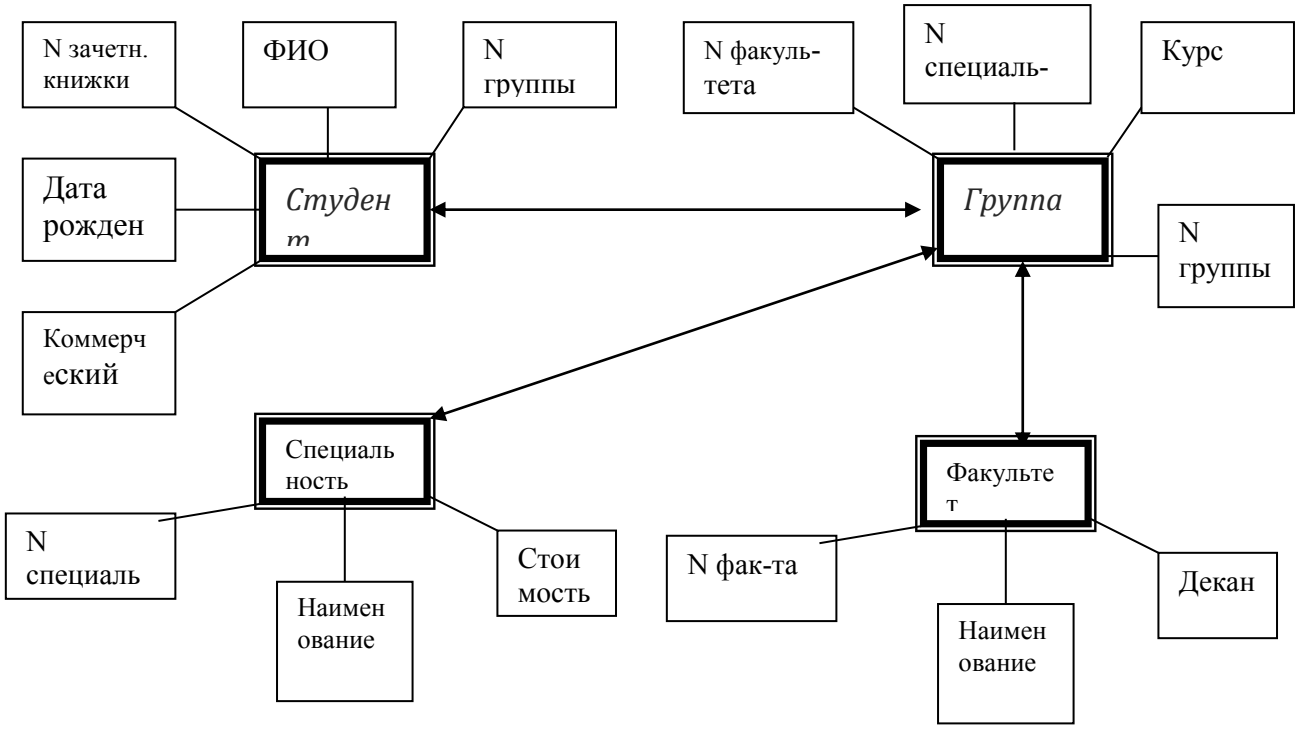


Рис.1.1 Инфологическая модель базы данных

Технология **создания таблицы Студент**

1. Создайте новую базу данных, щелкнув по соответствующей кнопке инструментальной панели. Появится Область задач с заголовком Создание файла. Выберите гиперссылку Новая база данных. В окне Файл новой базы данных выберите для файла папку Мои документы и присвойте файлу имя Академия2003. Щелкните по кнопке *Создать*. Появится окно базы данных с перечнем объектов.

2. В окне базы данных выберите объект **Таблицы** и выберите вариант создания с помощью **мастера таблиц**.

3. На 1-м шаге работы мастера выберите в качестве образца таблицу *Студенты* и, дважды щелкая по образцам полей, включите в создаваемую таблицу поля:

- *код студента*
- *фамилия*

- *специализация*
- Из таблицы *Сотрудники*:
- *дату рождения*
- *код отдела*

4. Переименуйте поля в соответствии с заданием:
- Фамилия - *ФИО*
- код студента - *N зачетной книжки*
- специализация – *коммерческий*
- код отдела – *N группы*.

5. Щелкните по кнопке *Далее*.

6. На шаге 2 в окне *Создание таблиц* дайте имя таблице *Студент* и сохраните включенным флажок *Microsoft Access автоматически определяет ключ*. Щелкните по кнопке *Далее*.

7. На шаге 3 нажмите на кнопку *Готово* и приступите к непосредственному вводу данных в таблицу.

8. Переключите таблицу в режим конструктора, щелкните по кнопке *Вид* инструментальной панели. Проведите корректировку типов данных в соответствии с заданием.

9. Вернитесь в режим таблицы с помощью аналогичной кнопки и приступите к вводу данных.

10. Введите данные для 3-х групп по 10 студентов в каждой с различными значениями полей.

11. После ввода данных сохраните базу данных.

12. Измените вид представления данных в столбце *Коммерческий* - замените флажки на значения логического типа - *Да/Нет*. Для этого в режиме *конструктора выделите поле Коммерческий, и раскройте список на вкладке Подстановка. Выберите тип элемента управления Поле*. Перейдите в режим таблицы и просмотрите ее. Верните назад тип элемента управления - *Флажок*.

13. Для того, чтобы сделать столбец уже, измените название столбца таблицы, соответствующего полю *N зачетной книжки*. Замените название столбца на сокращенное: *НЗК*. Для этого в режиме конструктора выделите указанное поле и введите на вкладке *Общие* в строку *Подпись* новое название. Просмотрите таблицу.

14. Предусмотрите контроль данных. Запретите ввод даты рождения меньше заданной, например, даты более ранней, чем 01.01.1970 г. При попытке ввода такой даты предусмотрите вывод сообщения: *Слишком старый студент*. Для этого в режиме конструктора установите окно свойств поля *Дата рождения*. Для

свойства *Условие на значение* введите: >#01.01.1970#. Для свойства *Сообщение об ошибке* введите: *Слишком старый студент.* Проверьте правильность установленного контроля значений поля *Дата рождения*.

Освоение приемов работы с фильтрами в таблицах

MS Access позволяет применять 3 вида фильтров для работы с таблицами: **Фильтр по выделенному**, **Фильтр для** и **Расширенный фильтр**.

Задание 4. Найдите студентов, фамилия которых начинается на заданную букву, например на букву «В». Список найденных студентов должен быть упорядочен по алфавиту. Для поиска использовать **расширенный фильтр**.

Технология поиска с помощью фильтра по выделенному

1. Найдите в поле *ФИО* любую фамилию, начинающуюся на букву «В» и выделите мышкой эту букву.

2. Щелкните правой клавишей мыши и в контекстном меню выберите пункт *Фильтр по выделенному.* В результате на экране останутся только строки таблицы с фамилиями, начинающимися на заданную букву.

3. Для отмены фильтра щелкните по кнопке *Удалить фильтр* на инструментальной панели.

Технология поиска с помощью расширенного фильтра

1. Для установки расширенного фильтра введите команду **ЗАПИСИ/Фильтр/Расширенный фильтр.** Появится окно с бланком фильтра**.**

2. Укажите поле, по которому должна происходить фильтрация. В окне бланка дважды щелкните по полю *ФИО*, расположенном в таблице *Студент*. Поле *ФИО* появится в 1-ой строке *Поле* нижней половины бланка (столбец 1).

3. Укажите в строке бланка *Сортировка* **порядок сортировки**. Для этого щелкните левой клавишей по этой строке в 1-м столбце. Появится список вариантов сортировки. Выберите вариант: *по возрастанию*.

4. Введите условие отбора. Для этого введите в 3-ью строку 2 символа: В*

5. Примените фильтр**.** Для этого можно воспользоваться 3-мя способами:

- выполнить команду **ФИЛЬТР/Применить фильтр**.
- щелкнуть по кнопке инструментальной панели *Применение фильтра.*
- щелкнуть правой клавишей по свободной зоне бланка и в контекстном меню выбрать пункт **Применить фильтр**.

6. Отмените фильтр**.** Для просмотра таблицы в полном виде нужно выполнить команду **Удалить фильтр** либо в меню ***ЗАПИСИ***, либо в контекстном меню, либо с помощью соответствующей кнопки инструментальной панели.

Задание 5. Найдите студентов, родившихся в заданном году, например в 1978 г.

Технология поиска с помощью Фильтра для

1. Щелкните правой клавишей мышки в поле *Дата рождения* любой записи. В контекстном меню выберите пункт *Фильтр для:*
2. Введите в строку условия фильтрации значение: ***.*.1978** и нажмите клавишу **Enter**. Удалите фильтр.

Технология поиска с помощью расширенного фильтра

1. Вызвав контекстное меню, очистите бланк фильтра.
2. Введите в 1-ый столбец бланка условие для поля *дата рождения >= заданная дата*, а во 2-ой столбец для того же поля условие <= *заданная дата*, где заданная дата – какая-либо дата по выбору студента или указанию преподавателя.
3. Для просмотра результата фильтрации щелкните по кнопке инструментальной панели *Применить фильтр*. Удалите фильтр.

Задание 6. Найдите студентов, родившихся в заданном году и обучающихся на коммерческой основе в заданной группе.

Для решения задачи используйте 4 столбца бланка с названиями полей: *дата рождения, дата рождения, N группы, коммерческий.*

Задание 7. Предварительно создав, введите данные в таблицы: **Группа, Факультет, Специальность**.

Таблица **Группа** должна иметь не менее 5 строк и содержать поле *N группы* того же типа и с таким же названием как в таблице *Студент*. Кроме того, должны иметься 3 строки со значением этого поля таким же, как в таблице *Студент.*

Таблица **Факультет** должна иметь не менее 5 строк и содержать поле *N факультета* того же типа и с таким же названием как в таблице **Группа**. Кроме того, должны иметься 3 строки со значением этого поля таким же, как в таблице **Группа.**

Таблица **Специальность** должна иметь не менее 5 строк и содержать поле *N специальности* того же типа и с таким же названием как в таблице **Группа**. Кроме того, должны иметься 3 строки со значением этого поля таким же, как в таблице **Группа.**

Задание 8. Создайте сводную таблицу по данным таблицы **Группа**, показывающую распределения студенческих групп по специальностям и факультетам.

Практическое занятие №7
Модификация структуры табличного файла.

Модификация самой структуры БД включает в себя возможности расщепления, объединения таблиц; изменения первичных ключей и структуры связей между таблицами.

1. Изменение первичных ключей базы данных

При работе с базой может возникнуть необходимость изменения первичных ключей некоторых таблиц. Так, если бы мы в таблице "Преподаватели" определили первичный ключ по полю "Фамилия", он бы не всегда однозначно идентифицировал записи в таблице, т.к. в таблице может быть несколько преподавателей с фамилиями "Петров", "Сидоров" и т.д.
Выходом из этой ситуации является или выбор "более уникального" поля, или же вводится поле типа "Счетчик", значение которого никогда не повторяется. В нашей таблице таким полем является "Номер_П", поскольку у каждого преподавателя свой уникальный номер.

Переопределение первичного ключа производится или сбросом определения ключа в описании индексов (команда "Индекс" меню "Вид" выводит список индексов БД, в ней активизируем строку "Первичный ключ" для поля, являющегося первичным ключом, и нажимаем кнопку "Del" клавиатуры); а затем установив новый

первичный ключ (командой "определить ключ" меню "Правка"). Либо же активизацией поля будущего первичного ключа, и щелчком на кнопке "Первичный ключ" панели инструментов.
Для определения ключа можно выделить несколько полей одновременно. Для этого удерживаем нажатой клавишу "Ctrl" клавиатуры и щелкаем по области маркировки требуемых строк. При невозможности переопределения выводится информационное окно Access.

2. Модификация структуры связей базы данных

Возможности определения и модификации структуры связей базы данных были рассмотрены в предыдущей лабораторной работе, и особенных затруднений не представляют.

Сложная структура связей базы , особенно с множественными связями, при установленных требованиях каскадного обновления и удаления связанных полей, приводит к замедлению работы Access. Поэтому после создания базы и какого-то периода ее эксплуатации рекомендуется пересмотреть структуру таблиц и их связей в целях оптимизации.

Из видов модификации базы данных иногда может потребоваться расщепление одной таблицы на несколько, или же объединение группы таблиц в одну. рассмотрим их применение по отношению к базе STUD.

3. Расщепление таблиц

Расщепление таблиц может потребоваться в том случае, если какая-то из таблиц проекта содержит редко используемую группу полей. Например, в запросах используются основные признаки объекта, а более детальная информация бывает востребована редко: в итогах, сводках, и т.д.

Имеет смысл расщепить такую таблицу на две или более частей, связав их по какому-либо признаку объекта.
Также повышает производительность системы расщепление таблицы, если к части данных необходимо ограничить доступ пользователей. К конфиденциальным данным можно отнести информацию о адресах заказчиков, зарплате сотрудников, деталях контрактов и т.д. В том случае закрытая информация обосабливается в отдельные таблицы, и доступ к ним пользователей ограничивается.

Можно использовать присоединение данных через сеть, ограничив доступ пользователей к данным средствами сетевой ОС.
Рассмотрим расщепление на примере таблицы "Преподаватели". Отделим личную информацию о преподавателях (домашний адрес,

домашний телефон, зарплата) от служебной (стаж, должность и т.д.). Создадим копию проекта базы данных, т.к. расщепление таблицы будет нести демонстрационный характер, и в следующих лабораторных работах мы будем пользоваться сохраненным вариантом. Для создания копии базы средствами Windows или Dos скопируем закрытый проект базы данных (файл STUD.mdb) в другой каталог.

Простейший метод расщепления таблицы - создание ее копии, удаление из обеих таблиц лишних полей; настройка связей - и занесение модифицированных таблиц обратно в проект.

Рассмотрим общий алгоритм расщепления таблиц:

1. Создаем 2 копии таблицы.
2. Обеим копиям таблиц даются уникальные имена.
3. Из обеих таблиц удаляются лишние поля.
4.В окне "Свойства таблиц" удаляются все условия на значение, содержащие в себе ссылки на удаленные поля.
5. В окне "Индексы" удаляются все индексы и ключи, построенные на удаленных полях.
6. При необходимости генерируются первичные ключи для созданных таблиц (возможно введение нового поля типа "Счетчик").
7. В окне "Схема данных" (команда "Схема данных" меню "Сервис"), выбирается режим "Все связи" нажатием соответствующей кнопки на панели инструментов.
8. Удаляются все связи между таблицей - оригиналом и другими таблицами (щелкая по линии связи таблиц для активизации связи и нажимается клавиша "Del" на клавиатуре).
9. В схему данных вводятся таблицы - копии (щелкнув на кнопке "Добавить таблицу" панели инструментов).
10. Устанавливаются связи таблиц - копий с другими таблицами.
11. Возможно, удаляется таблица - оригинал.
12. Сохраняется структура проекта БД.

Применительно к таблице "преподаватели":

- создадим таблицы "Преподаватели_личное" с полями "Зарплата", "Адрес_дом", "Телефон_дом",”Номер_П” ; и "Преподаватели_служебное", со всеми оставшимися полями плюс "Номер_П" для связи;
- удаляем индексы на таблице "Преподаватели" и описание первичного ключа по полю "Номер_П";
- В окне схемы данных устанавливаем связи: "Преподаватели_служебное.Номер_П" с "Занятия.Номер_П" типа

“один-ко-многим” и "Преподаватели_служебное.Номер_П" с "Преподаватели_личное.Номер_П" типа “один-к-одному”;
- удаляем таблицу "Преподаватели";
- сохраняем проект под именем STUD_2.mdb.

Практическое занятие №8
Индексирование и сортировка таблиц.

Индексирование полей таблицы.

Индексирование позволяет ускорить сортировку и поиск данных в таблице. Можно индексировать числовые, денежные, текстовые, логические поля, а также поля типа Счетчик и Дата. Не следует создавать слишком много индексов для одной таблицы, т.к. это замедлит ввод и редактирование ее данных.

Первичный ключ- это специальный тип индекса, который однозначно идентифицирует каждую запись. В первичный ключ могут входить несколько полей, но значение первичного ключа должно быть уникальным для каждой записи. Первичные ключи используются для установления связей между таблицами.

Связи между таблицами.

Таблицы могут быть связаны отношениями **один – к -одному, один -к многим и многие – к - многим.**Access позволяет использовать только отношения первых двух типов.

При установлении связей нужно определить какая таблица является *главной,* а какая - *подчиненной.*

Отношение *один-к-одному* означает, что одной записи подчиненной таблицы соответствует только одна запись в главной таблице. Такие отношения встречаются очень редко, т.к. требую неоправданно много места в БД. Вместо них можно просто добавить поля подчиненной таблицы к полям главной.

Наиболее часто используются отношения *один-ко-многим.* В этом случае одной записи в главной таблице соответствует несколько записей в подчиненной таблице.

Для создания отношений необходимо указать поля в двух таблицах, которые содержат одни и те же данные. Обычно такое поле в одной из таблиц (главной) является ключевым. Имена связывающих полей могут отличаться, но типы и свойства должны совпадать. Возможна связь между полем типа Счетчик и полем типа Число с форматом Длинное целое.

Требования к содержанию, оформлению и порядку

выполнения

Перед выполнением лабораторной работы создайте папку «Ваша фамилия Lab 1» (например: «Ivanov Lab 1»). В эту папку в ходе выполнения работы необходимо сохранять требуемые материалы. Далее необходимо изучить теоретический материал и выполнить последовательно все предложенные задания. После выполнения лабораторной работы ответьте на контрольные вопросы.

Теоретическая часть

СУБД Access использует **реляционную** модель базы данных, в которой данные представлены в виде взаимосвязанных таблиц (отношений по англ. - *relations).*

Важнейшим этапом проектирования базы данных является разработка информационно-логической модели предметной области, не ориентированной на СУБД, но отражающей предметную область в виде совокупности информационных объектов и их информационных связей.

СУБД Access позволяет работать с объектами базы данных, к которым относятся *таблицы, запросы, формы, отчеты, страницы, макросы и модули*

Таблицы служат для хранения данных в определенной структуре.

Запросы создаются для выборки данных из одной или нескольких связанных таблиц.

Формы предназначены для ввода, редактирования и просмотра табличных данных на экране в удобном виде.

Отчеты являются выходными документами, предназначенными для вывода на принтер.

Страницы доступа к данным - это WEB- страницы, обеспечивающие функциональность стандартных форм и отчетов Access: ввод, редактирование и представление данных. Страницы доступа к данным можно открывать в программах просмотра WEB-страниц (например, Internet Explorer) и использовать для ввода, просмотра и отбора информации в базе данных.

Макросы используются для автоматизации различных процедур обработки данных, являются программами, состоящими из макрокоманд высокого уровня. Макропрограммирование в Access не требует знания языка VisualBasic. Имеющийся в Access набор из около 60 макрокоманд обеспечивает практически любые действия, необходимые для решения задач.

Модули являются программами на языке, которые служат для реализации нестандартных процедур обработки данных.

Все данные БД Microsoft Access и средства их отображения хранятся в одном файле с расширением **MDB.**

Задание 1.

Ознакомиться с **учебной базой данных**компании **«Борей»**

Технология выполнения:

1. Запустить приложение Microsoft Access 2010**.**

2. В диалоговом окне, которое появится в процессе загрузки в группе **Создание** базы данных выбрать вариант **Образцы шаблонов**.

3. В диалоговом окне **Доступные шаблоны** выберите пример базы данных компании **Борей**.

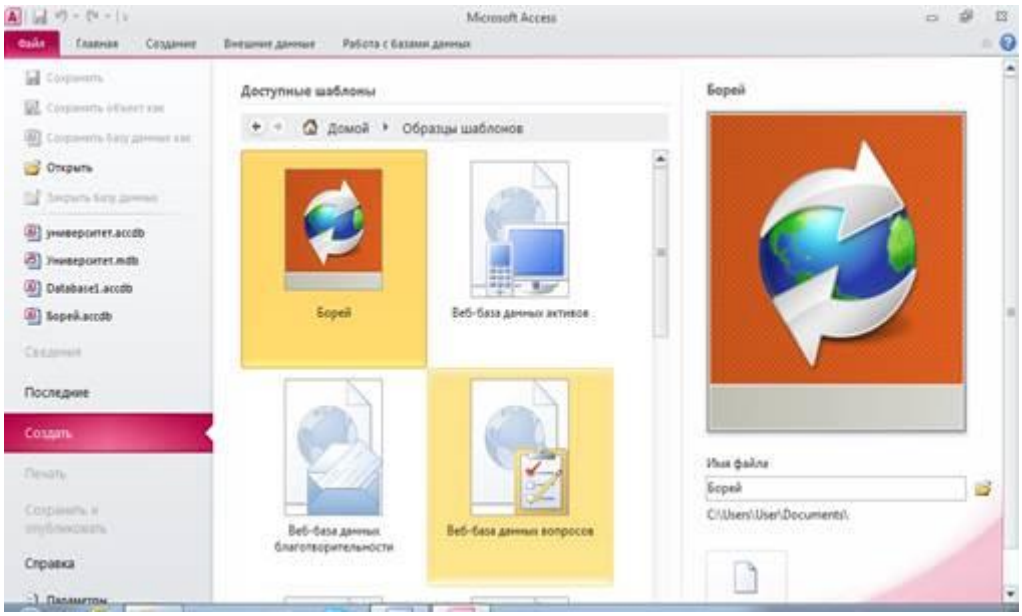


Рисунок 1.

4. Для того чтобы открыть БД «Борей», необходимо активировать ее двойным щелчком правой клавишей мыши.

5. В окне появится заставка учебной базы данных «Борей».

Рисунок 2.

6. Чтобы просмотреть БД, на панели сообщений выберите команду «**Включить содержимое**».

7. Появится **окно входа**, где необходимо из списка выбрать Ф.И. сотрудника компании и нажать на командную кнопку **Вход**.

8. Просмотреть **данные**каждой **таблицы,**открыв их, перемещаясь по **быстрым ссылкам**.

9. Просмотреть **структуру**каждой **таблицы**в режиме **конструктора: Главная =>Режим =>Конструктор.**

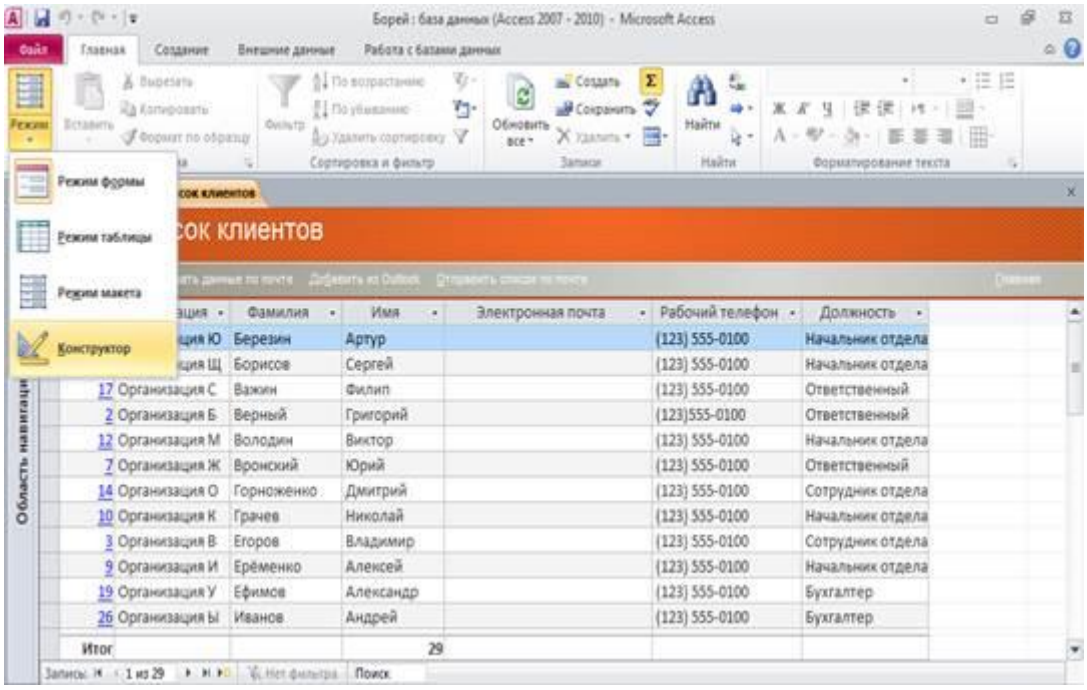


Рисунок 3.

10. Открыть **таблицу «Клиенты»** и выполнить следующие операции:

· определить количество записей в таблице;

· просмотреть 25-ую запись, введя ее номер в окно номеров записей;

· выполнить сортировку таблицы по должностям. Для этого необходимо установить указатель мыши на треугольник в заголовке столбца и щелкнуть правой кнопкой мыши. Появится контекстное меню, выберите пункт **«Сортировка по возрастанию».**

· скрыть столбец «**Организация**». Для этого необходимо выделить его и вызвать контекстное меню правой клавишей мыши выбрать вкладку **Скрыть поля**. Отмените скрытие и добавьте с помощью переключателей столбцы: факс, адрес, город, область, индекс.

· выделить первые два столбца и **закрепить** их с помощью контекстного меню, выполнив команду **Закрепить поля.** Прокрутить таблицу по горизонтали (убедиться, что столбцы не перемещаются). Отменить закрепление;

· применить фильтр для выделения строк с клиентами, у которых имя начинается на букву **А**. Для этого выделить любую строку в столбце с полем **Имя** и вызвать контекстное меню.

Выполнить команду **Текстовые фильтры => Начинается с => введите букву**.

· Отменить фильтр, щелкнув в контекстном меню по пункту **«Снять фильтр с Имя»**

·  изменить вид сетки таблицы, используя соответствующую кнопку инструментальной панели.

11. Просмотреть **таблицу**«Сотрудники» в **режиме конструктора.**

12. Просмотреть отчеты (для этого в области навигации щелкните по кнопке «Открыть/Закрыть границу области перехода»).

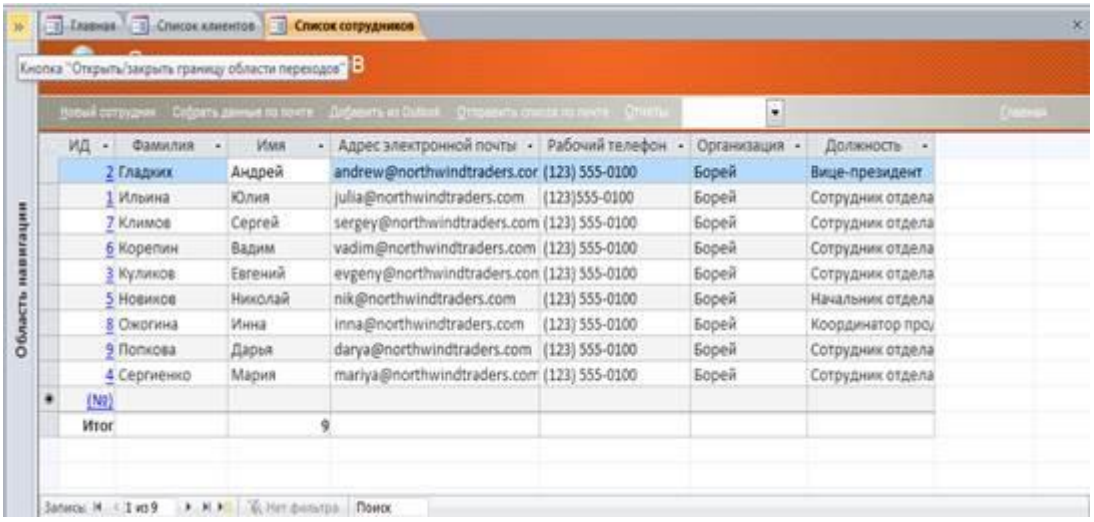


Рисунок 4.

Из появившегося списка найдите вкладку Отчеты и просмотрите некоторые из них, например:

· «Адресная книга клиентов»;

· «Годовой отчет о продажах»;

· «Счет»;

Таблицы

Таблицы составляют основу базы данных - именно в них хранятся все данные. Таблицы должны быть тщательно спланированы. Прежде всего, должна быть спланирована структура каждой таблицы. Структура таблиц определяется содержанием тех выходных форм и отчетов, которые должны быть, затем получены. При планировании таблиц необходимо избежать дублирования информации в разных таблицах.

Таблица - это объект БД, который хранит данные определенной структуры. Таблица состоит из *записей* (строк), каждая из которых описывает одну сущность. Каждый столбец таблицы - это *поле.* Столбец содержит однотипную информацию.

Длина имени таблицы - не более 64 символов.

Длина имени поля - не более 64 символов.

Количество полей в одной таблице - не более 255.

Количество записей - неограниченно.

Суммарный объем информации во всей БД - не более 1 гигабайта.

Для каждого поля необходимо указать тип данных. Тип данных определяет вид и диапазон допустимых значений, которые могут быть введены в поле, а также объем памяти, выделяющийся для этого поля.

Таблица может содержать следующие типы полей:

Текстовый Короткий текст. Текст и числа, например, имена и адреса, номера телефонов и почтовые индексы. Текстовое поле может содержать до 255 символов.

Поле Мемо Длинный текст и числа, например, комментарии и пояснения. Memo- поле может содержать до 65 535 символов.

Числовой Общий тип для числовых данных, допускающих проведение математических расчетов, за исключением расчетов для денежных значений. Свойство Размер поля позволяет указать различные типы числовых данных. Длина - до 8 байт. Точность -до 15 знаков.

Дата/время Значения даты и времени. Пользователь имеет возможность выбрать один из многочисленных стандартных форматов или создать специальный формат. Длина - 8 байт.

Денежный Денежные значения. Числа представляются с двумя знаками после запятой. Не рекомендуется использовать для проведения денежных расчетов значения, принадлежащие к числовому типу данных, так как последние могут округляться при расчетах. Значения типа "Денежный" всегда выводятся с указанным числом десятичных знаков после запятой. Длина - 8 байт.

Счетчик Автоматически вставляющиеся последовательные номера. Счетчик увеличивается на единицу для каждой следующей записи. Нумерация начинается с 1. Поле счетчика удобно для создания ключа. В таблице может быть только одно такое поле. Длина - 4 байта.

Логический Значения "Да"/"Нет", "Истина'/Ложь", "Вкл"/"Выкл", т.е. одно из двух возможных значений. Длина - 1 байт.

Поле объекта OLE Объекты, созданные в других программах, поддерживающих протокол OLE, например графики, рисунки и т.п. Объекты связываются или внедряются в базу данных Microsoft Access через элемент управления в форме или отчете.

Гиперссылка- позволяет вставлять в поле гиперссылку, с помощью которой можно ссылаться на произвольный фрагмент

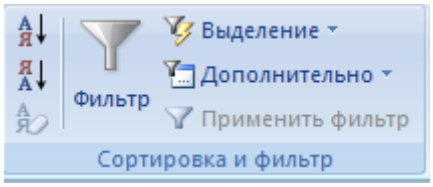
данных внутри поля или страницы на том же компьютере, в локальной сети или в Internet.

Практическое занятие №9
Поиск, сортировка и фильтрация данных в MS Access

Сортировка данных

Для удобства просмотра можно сортировать записи в таблице в определенной последовательности, например, в таблице *Преподаватели* записи можно отсортировать в порядке убывания стажа преподавателей.

Для сортировки и фильтрации записей в таблице предназначена группа команд, обеспечивают задание стандартных типов сортировки (**по возрастанию** ,**по убыванию**), а также отмену ранее заданной сортировки (**Очистить все сортировки**), находящихся в разделе **Сортировка и фильтр**, вкладки <Главная>.



Раздел **Сортировка и фильтр**

Прежде чем щелкнуть по кнопке сортировки, следует выбрать поля, используемые для сортировки: для выбора поля достаточно поместить курсор в любую его запись. После этого щелкнуть по кнопке сортировки - и данные отобразятся в отсортированном порядке. В режиме таблицы можно выделить сразу два или несколько соседних столбцов, а затем выполнить сортировку. По умолчанию в Access 2007 сортировка записей начинается с крайнего выделенного столбца. При этом записи таблицы будут отсортированы сначала по крайнему левому столбцу, затем (для одинаковых значений в первом сортируемом столбце) - по второму и т.д.

Примечание. Обратите внимание на то, что сортировка может применяться к разным столбцам таблицы. Признак использования сортировки – стрелочка, отображаемая слева от названия поля.

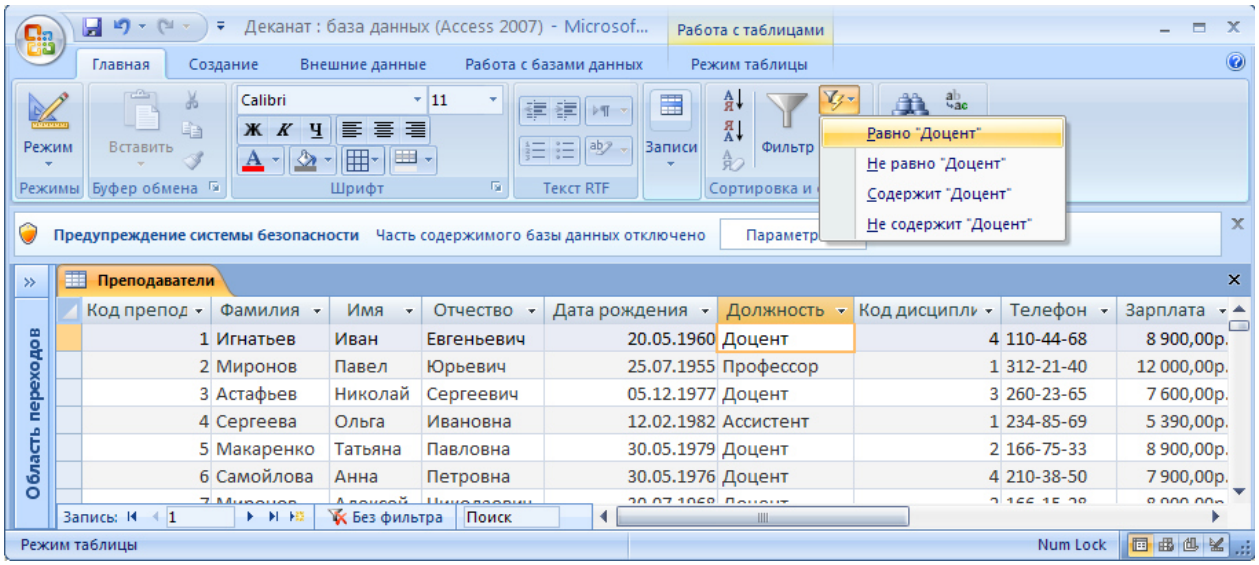
Отбор данных с помощью фильтра

Фильтр - это набор условий, применяемых для отбора подмножества записей. В Access 2007 существуют фильтры четырех типов: фильтр по выделенному фрагменту, обычный, расширенный фильтр и фильтр по вводу.

Фильтр по выделенному фрагменту - это способ быстрого

отбора записей по выделенному образцу. Например, нужно просмотреть в таблице записи только о доцентах. Для этого необходимо выделить слово *доцент* в любой из записей, щелкнуть мышью на стрелочке рядом с кнопкой Применить фильтр в разделе **Сортировка и фильтр** вкладки <Главная>, и Access выберет только те записи, для которых значение в столбце <Должность> равно *Доцент.*

Аналогичный результат получится, если использовать кнопку <Выделение> - Выделение, расположенной в разделе **Сортировка и фильтр** вкладки **Главная.** При выборе команды **Равно "Доцент"** будет выделена таблица, содержащая только записи, в котором поле <<Должность>> имеет значение *Доцент.* При нажатии на кнопку <Удалить фильтр> вновь будет выведена вся таблица.



Создание фильтра по выделению.

Обратите внимание, что в строке состояния окна таблицы присутствуют слова *С Фильтром.* В дополнение к этому кнопка <Применить фильтр> затенена, а это означает, что используется фильтр. При отключении этой кнопки все фильтры будут сняты.

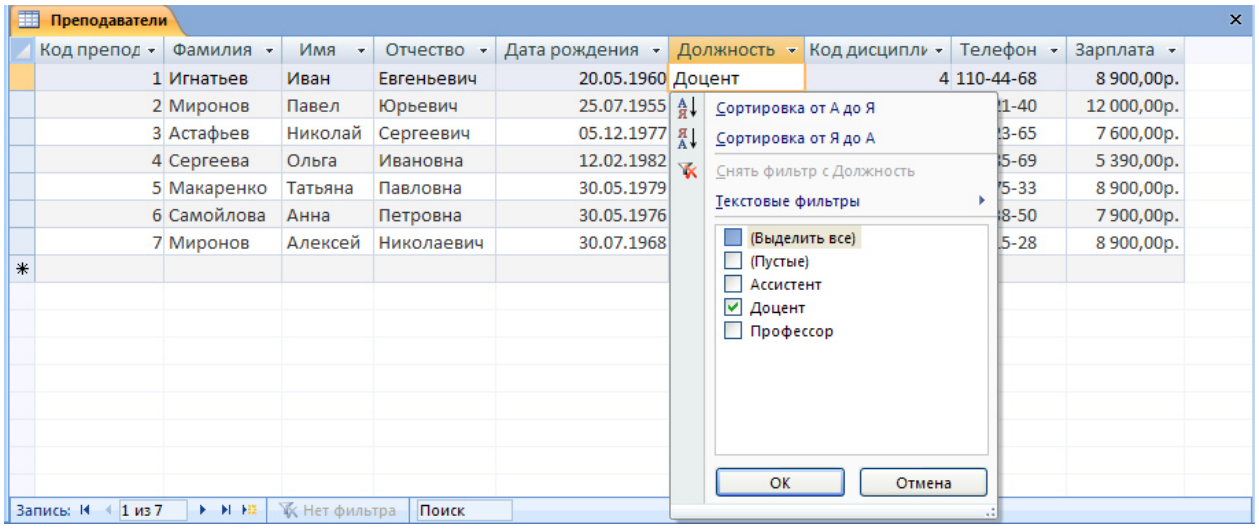
Фильтр по выделенному может собирать вместе критерии выбора при каждом использовании кнопки <Выделение>. Например, можно выделить должность *доцент, а* затем дисциплину *Информатика* (конечно, если такая дисциплина присутствует в вашей таблице). В этом случае появятся только записи о доцентах, которые преподают информатику.

Обычный фильтр. Фильтрование данных в Access производится также с помощью кнопки -<Фильтр> - Фильтр, Каждое поле становится

полем со списком (когда в нем находится курсор), в котором можно выбрать из списка значения для данного поля.

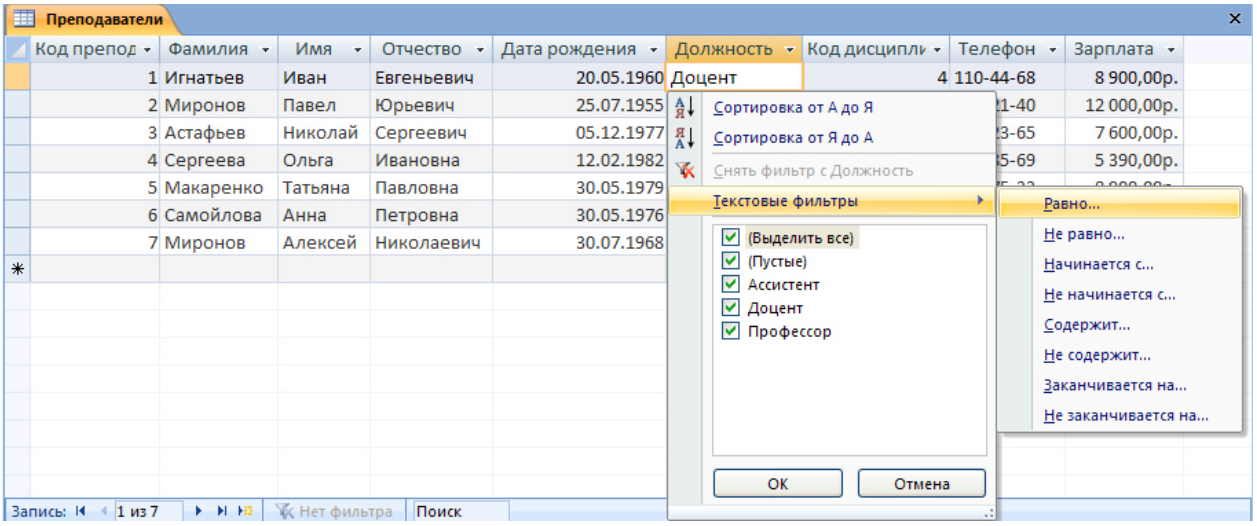
После щелчка по кнопке -<Фильтр> - 

отображается диалоговое окно установки параметров фильтра. Можно просто отменить установку флажка **Выделить все** с последующей установкой флажка напротив той записи, которая будет отображена на экране. В результате на экране появится именно та запись, напротив которой был установлен флажок.



Выбор записей, которые будут отображены на экране

В окне установки параметров фильтра имеется также список **Текстовые фильтры.** После раскрытия этого списка отображается набор предопределенных текстовых фильтров Выбрать требуемый фильтр можно, щелкнув на соответствующем варианте мышью.



Выбор записей с помощью текстовых фильтров

Еще более сложные условия фильтрации можно задать командой **Главная, Сортировка и фильтр,**

Дополнительно, **Расширенный фильтр**...

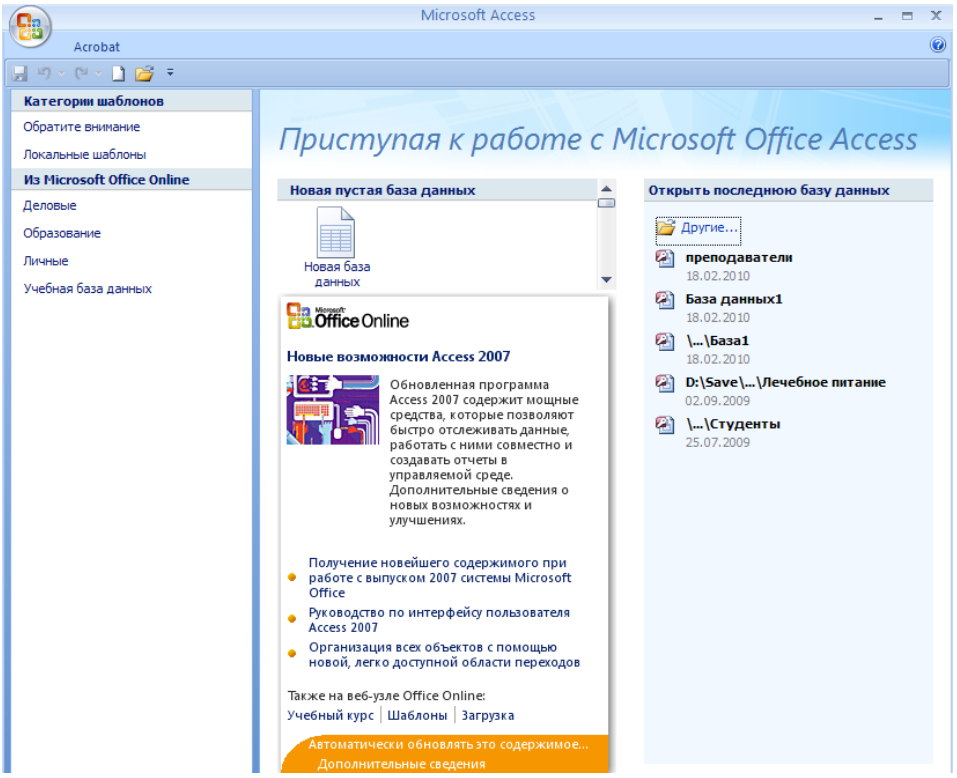
Практическое занятие №10

Поиск данных в таблице. Установка даты и вывод записей на экран

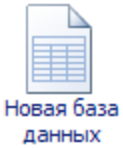
Создайте новую базу данных.

Для создания новой базы данных:

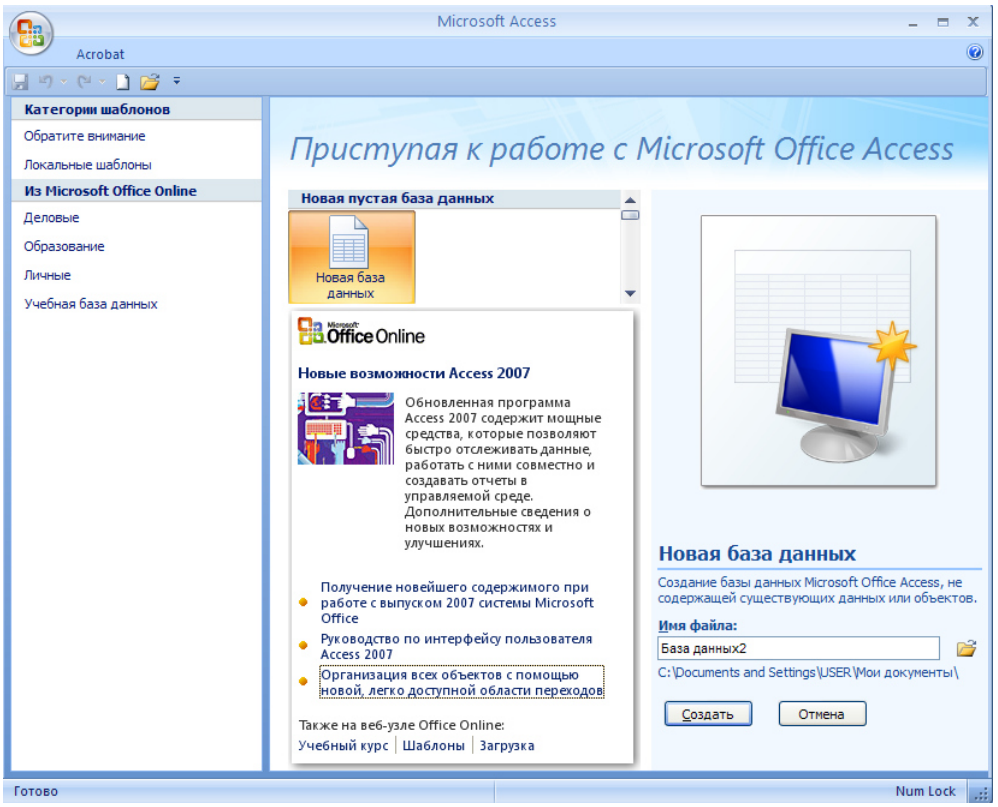
- загрузите Access 2007, например, с помощью меню **Пуск** на панели задач. На экране появится начальное окно Access 2007 (рис.1).



Начальное окно Access 2007

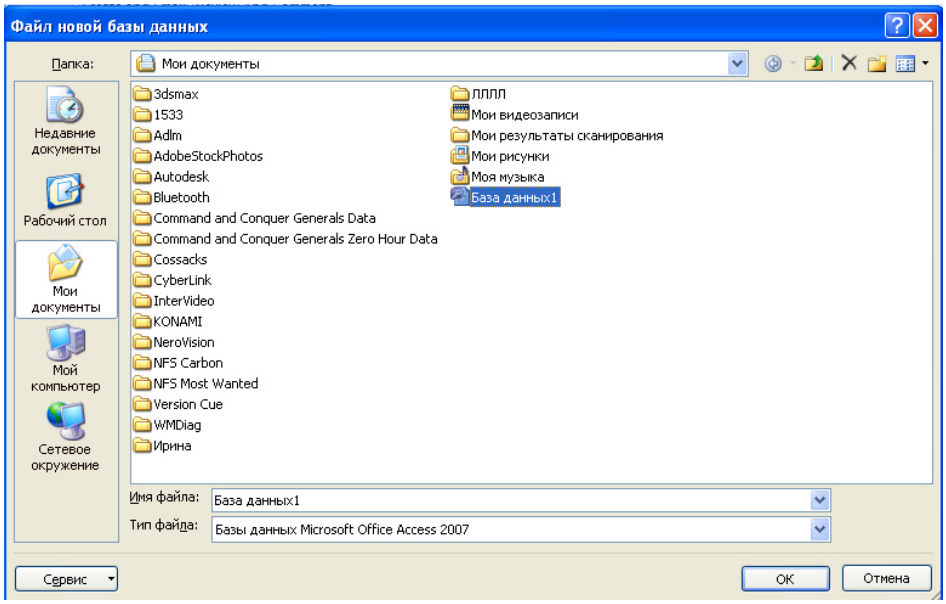
- в появившемся окне на центральной панели, на вкладке *Новая пустая база данных* выберите пиктограмму 

. В правой части окна появится панель, запрашивающая имя новой базы данных



Создание новой базы данных.

- На этой панели, на вкладке *Новая база данных* задайте имя вашей базы данных (пункт **Имя файла**). Имя файла задайте *свою фамилию*. Под текстовым полем с именем базы данных приведен путь текущего каталога, в котором она будет сохранена при щелчке на кнопке <Создать>.
- Если необходимо сохранить новую базу данных в другом каталоге, то нужно щелкнуть на пиктограмме <Поиск расположения для размещения базы данных>. На экране появится диалоговое окно <<Файл новой базы данных>>

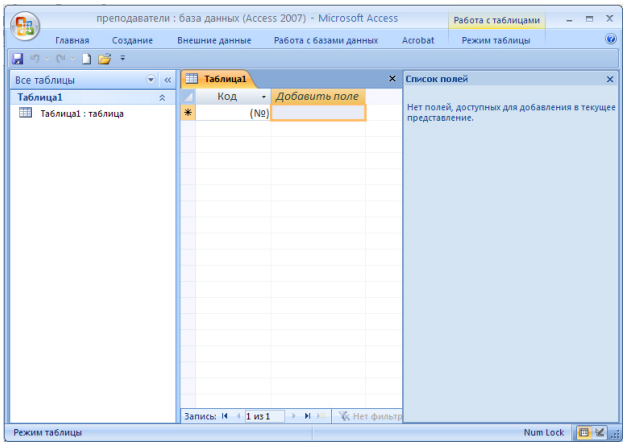


Диалоговое окно <<Файл новой базы данных>>

- По умолчанию Access предлагает вам имя базы - *База данных 1*, а тип файла - *База данных Microsoft Office Access 2007*. Имя файла задайте *Свою фамилию,* а тип файла оставьте прежним;
- щелкните по кнопке <ОК>
- щелкните по кнопке <Создать>.

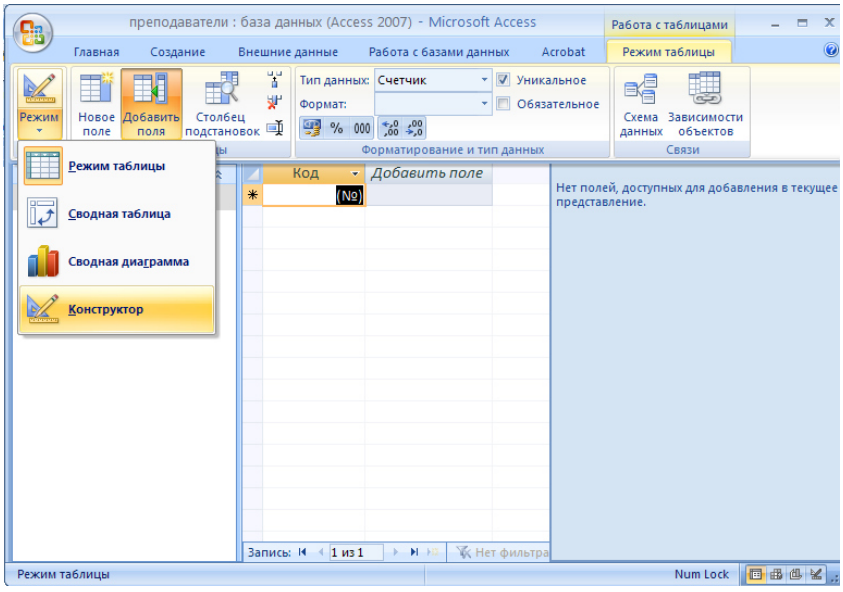
Создать новую таблицу базы данных

После выполнения предыдущего пункта задания будет создана новая база данных и открыта первая таблица базы данных с именем *Таблица1*, которая создается автоматически (рис.4). На левой панели приведен список существующих таблиц, состоящий пока из единственного пункта.



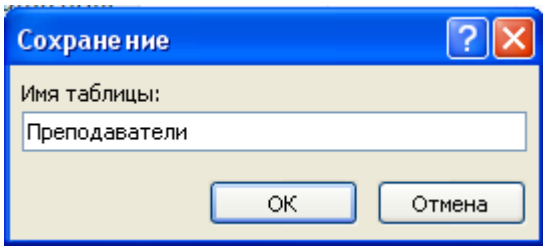
Создание первой таблицы базы данных

- На вкладке ленты *Режим таблицы* выберите режим **Конструктор**



Выбор режима таблицы **Конструктор**

- В появившемся диалоговом окне «Сохранение» введите имя таблицы *Преподаватели* (рис.5);



Диалоговое окно «Сохранение»

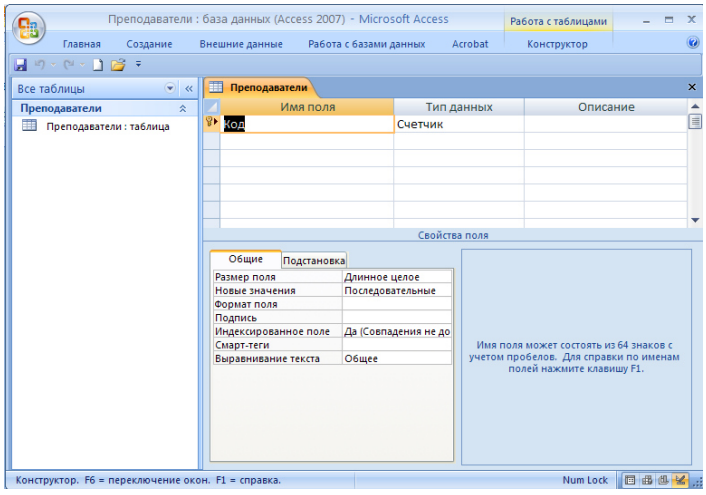
- щелкните по кнопке <OK>.

В результате проделанных операций открывается окно таблицы в режиме конструктора (рис. 6), в котором следует определить поля таблицы.

Определите поля таблицы в соответствии

Для определения полей таблицы:

- введите в строку столбца «Имя поля» имя первого поля *Код преподавателя;*
- в строке столбца «Тип данных» щелкните по кнопке списка и выберите тип данных *Счетчик*. Поля вкладки *Общие* оставьте такими, как предлагает Access.



Окно таблицы в режиме конструктора

Примечание. Заполнение строки столбца «Описание» необязательно и обычно используется для внесения дополнительных сведений о поле.

Для определения всех остальных полей таблицы базы данных *Преподаватели* в соответствии с табл. выполните действия, аналогичные указанным выше.

Таблица данных *Преподаватели*

Имя поля	**Тип данных**	**Размер поля**	**Формат поля**
Код преподавателя	Счетчик		

Фамилия	Текстовый	15	
Имя	Текстовый	15	
Отчество	Текстовый	15	
Дата рождения	Дата/время		Краткий формат даты
Должность	Текстовый	9	
Дисциплина	Текстовый	11	
Телефон	Текстовый	9	
Зарплата	Денежный		Денежный

Введите данные в таблицу в соответствии с табл. 2 и проверьте реакцию системы на ввод неправильных данных в поле «Должность».

Попробуйте в поле <Должность> ввести слово *Лаборант*. На экране должно появиться сообщение: "Такой должности нет, правильно введите данные". Введите правильное слово.

Код преподавателя	**Фамилия**	**Имя**	**Отчество**	**Дата рожд.**	**Дисциплина**	**Зарплата**
1	Игнатьев	Иван	Евгеньевич	20.05.60	Информатика	8900р.
2	Миронов	Павел	Юрьевич	25.07.55	Экономика	12000 р.
3	Астафьев	Николай	Сергеевич	05.12.77	Математика	7600 р.
4	Сергеева	Ольга	Ивановна	12.02.82	Математика	4500 р.
5	Макаренко	Татьяна	Павловна	30.05.79	Экономика	8900 р.
6	Самойлова	Анна	Петровна	30.05.76	Информатика	7900 р.
7	Миронов	Алексей	Николаевич	30.07.68	Физика	8900 р.

Произведите поиск в таблице преподавателя Миронова

Для поиска в таблице преподавателя Миронова:

- переведите курсор в первую строку поля «Фамилия»;
- выполните команду **Найти,** расположенную на ленте на вкладке <Главная> в разделе **Найти**;

- в появившейся строке параметра *Образец* введите *Миронов*;
- в строке параметра *Просмотр* должно быть слово *ВСЕ* (имеется в виду искать по всем записям);
- в строке параметра *Совпадение* выберите из списка *С любой частью поля;*
- щелкните по кнопке <Найти>. Курсор перейдет на вторую запись и выделит слово *Миронов*;
- щелкните по кнопке <Найти далее>. Курсор перейдет на седьмую запись и также выделит слово *Миронов*;
- щелкните по кнопке  Закрыть> для выхода из режима поиска.

Произведите замену данных: измените заработную плату ассистенту Сергеевой с 4500р. На4700р.

- переведите курсор в первую строку поля «Зарплата»;
- выполните команду **Найти,** расположенную на ленте на вкладке <Главная>;
- в появившемся окне в строке *Образец* введите 4 500;
- в строке *Заменить на* введите 4700. Обратите внимание на остальные опции – вам надо вести поиск по всем записям данного поля;
- щелкните по кнопке <Найти далее>. Курсор перейдет на четвертую запись;
- щелкните по кнопке <Заменить>. Данные будут изменены;

Примечание.Чтобы заменить сразу все данные, надо использовать кнопку <Заменить все>

- щелкните по кнопке  Закрыть> для выхода из режима замены.

9. **Произведите сортировку данных в поле «Дата рождения» по убыванию.**

Для сортировки данных в поле <<Дата рождения>> по убыванию:

- щелкните по любой записи поля <<Дата рождения>>;
- щелкните по кнопке  , расположенной в разделе**Сортировка и фильтр** вкладки **Главная**. Все данные в таблице будут отсортированы в соответствии с убыванием значений в поле <<Год рождения>>;

9. **Произведите фильтрацию данных по полям «Должность» и «Дисциплина».**

Для фильтрации данных по полям «Должность» и «Дисциплина:

- щелкните на маленьком треугольнике, расположенном справа от имени поля «Должность». Откроется диалоговое окно сортировки и фильтрации (рис.7);
- снимите флажки напротив значений полей, которые нужно исключить в процессе фильтрации;
- нажмите на кнопку <OK>. В таблице останутся только записи о преподавателях – доцентах;

Практическое занятие №11

Работа со связями, ключевыми полями и индексами. Ссылочная целостность.

Создание связей между таблицами

Чтобы определить связь между таблицами «Студенты», «Предметы» и «Оценки»:

1 Закройте все окна таблиц базы данных. Access не позволяет создать или изменить связь, если открыта хотя бы одна таблица.

2 Выберите команду «Сервис, Схема данных» или нажмите кнопку «Схема данных» на панели инструментов. Появится окно «Схема данных», которое используется для просмотра и изменения существующих связей и для определения новых связей между таблицами и/или запросами.

3 Выберите команду «Связи, Добавить таблицу» или нажмите кнопку «Добавить таблицу» на панели инструментов. Появится диалоговое окно «Добавление таблицы».

4 Раскройте вкладку «Таблицы», в списке таблиц выделите «Оценки» и нажмите кнопку «Добавить». Аналогично добавьте таблицы «Предметы» и «Студенты». Нажмите кнопку «Закрыть». Таблицы появятся в окне «Схема данных».

5 Связь между таблицами «Оценки» и «Студенты» строится по значению полей «КодСтудента». Поместите указатель мыши над полем «КодСтудента» таблицы «Оценки», нажмите левую кнопку мыши и, не отпуская ее, перетащите появившийся значок поля на поле «КодСтудента» таблицы «Студенты». Отпустите левую кнопку мыши. Появится диалоговое окно «Связи».

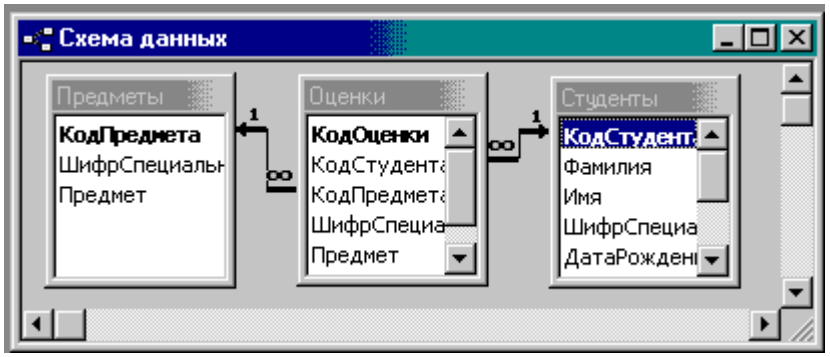
При создании новой связи важна последовательность действий. Перетащите с помощью мыши поле со стороны «один», связи «один ко многим» и отпустите его на стороне «многие». Такая последовательность действий гарантирует, что основная таблица

стороны «один» появится в списке «Таблица/запрос», а таблица стороны «многие» появится в списке «Связанная таблица/запрос». Если вы попытаетесь обеспечить ссылочную целостность, перетаскивая поле в обратном направлении (со стороны «многие» на сторону «один»), то на последнем шаге создания такой связи вы получите сообщение об ошибке.

6 Нажмите кнопку «Объединение». Появится диалоговое окно «Параметры объединения». В нашем примере создается связь с отношением «один ко многим», поэтому выберите переключатель «3». Нажмите кнопку ОК.

7 В диалоговом окне «Связи», можно установить для связи режим обеспечения ссылочной целостности. В этом режиме Access автоматически будет следить за тем, чтобы в таблице «Оценки» не появились коды сотрудников, которых нет в таблице «Студенты». Для обеспечения ссылочной целостности установите флажок «Обеспечение целостности данных».

Access обеспечивает ссылочную целостность, используя каскадное обновление и удаление данных. Чтобы включить режимы каскадного обновления и удаления данных, установите флажки «Каскадное обновление связанных полей» и «Каскадное удаление связанных полей» соответственно. Эти флажки доступны, только если установлен флажок «Обеспечение целостности данных».



8Нажмите кнопку «Создать», чтобы подтвердить создание связи и перейти в окно «Схема данных». Аналогично создайте связь между таблицами «Оценки» и «Предметы по полю «КодПредмета»

9 Закройте окно «Схема данных», нажав кнопку «Закрыть» в правом верхнем углу окна. Появится диалоговое окно, запрашивающее подтверждение изменения схемы данных. Подтвердите изменение, нажав кнопку «Да»

Практическое занятие №12
Создание меню различных типов.

Краткие сведения

При работе с базой данных часто приходится многократно выполнять одинаковые порой рутинные операции. Вполне естественно было бы автоматизировать их выполнение. Для этого Access располагает достаточными средствами, позволяющими во многом автоматизировать и упорядочить работу с базой данных. К числу таких средств относятся:

- пользовательские меню и инструментальные панели;
- кнопочные формы управления базой данных;
- средства настройки параметров запуска базы данных;
- макросы и модули.

Создание пользовательского меню

Задание 1. Создайте пользовательское меню для управления базой данных, содержащее категории **Формы** и **Отчеты** с пунктами (командами) для открытия ранее составленных форм и отчетов.

Технология

1. Для создания новой строки меню откройте окно *Настройка*. Для этого выполните команду **ВИД/Панели инструментов/*Настройка*** или, щелкнув правой клавишей по любой панели инструментов, выберите в контекстном меню пункт *Настройка*.

2. В окне *Настройка* на вкладке *Панели инструментов* щелкните по кнопке *Создать*.

3. В окне *Создание панели инструментов* введите имя панели инструментов: *Управление базой данных*. Нажмите кнопку *Ok*. В окне Access появится небольшая миниатюра панели. Перетащите созданную панель инструментов в верхнюю часть окна Access, установив над строкой меню.

4. В окне *Настройка* нажмите кнопку *Свойства* и определите тип созданной панели - *Строка меню*. Закройте окно установки свойств.

5. Добавьте в строку созданного меню категорию **Формы**. Для этого в окне *Настройка* откройте вкладку *Команды* и в списке категорий щелкните по категории *Новое меню*. Перетащите команду *Новое меню* из списка команд в правом подокне на строку меню *Управление базой данных*. Не закрывая окно *Настройка*, щелкните правой клавишей в строке меню по категории *Новое* меню и в контекстном меню замените имя категории на *Формы*.

6. Добавьте в меню категорию **Отчеты** аналогично пункту 5.

7. Аналогично добавьте в меню *Формы* новое подменю, назвав его *Простые*.

8. В окне *Настройка* на вкладке *Команды* выделите категорию *Все формы*. Перетащите строку с названием одной из созданных ранее форм - *Студент простая* в область команд (пунктов) категории *Формы* строки меню *Управление базой данных*. Включив контекстное меню новой команды, установите стиль отображения - *Только текст.*

9. Аналогично добавьте в область команд категории **Формы/Простые** пункты с названием форм – *Группа* и *Простая форма по запросу*.

10. Добавьте в категорию *Отчеты* меню *Управление базой данных* пункты с названиями отчетов. Закройте окно *Настройка*. Проверьте работу меню.

11. Выполните команду **СЕРВИС/Параметры запуска** и установите в окне *Параметры запуска* следующие параметры запуска при открытии базы данных:

- введите в качестве заголовка приложения название Академия;
- выберите в качестве строки меню строку Управление базой данных.
- отмените вывод на экран окна базы данных, строки состояния, полного набора меню Access, встроенных панелей инструментов.

12. Закройте окно *Параметры запуска*. Закройте базу данных, затем повторно откройте. Откроется окно Access, содержащее только одну строку пользовательского меню *Управление базой данных* с категориями *Формы* и *Отчеты*. Убедитесь в правильной работе команд меню.

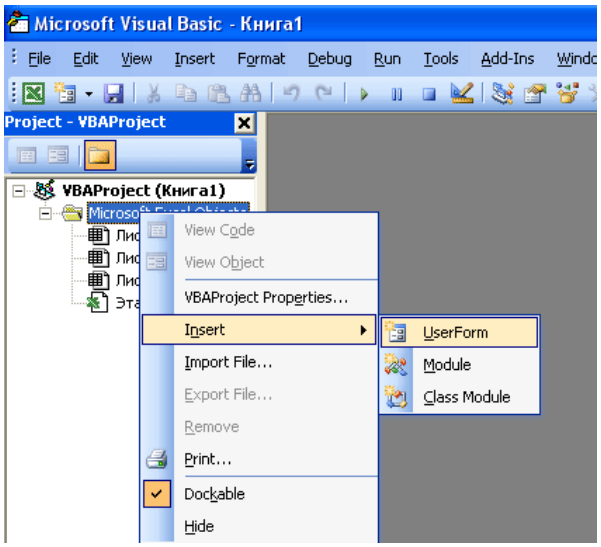
13. Восстановите для базы данных *Академия* отображение окна базы данных, полного набора меню Access, встроенных панелей инструментов. Для этого перезагрузите базу данных и при повторном открытии держите нажатой клавишу **SHIFT**. Выполните команду **СЕРВИС/Параметры запуска** и восстановите исходное состояние флажков.

Практическое занятие №13
Создание элементов управления рабочим окном.

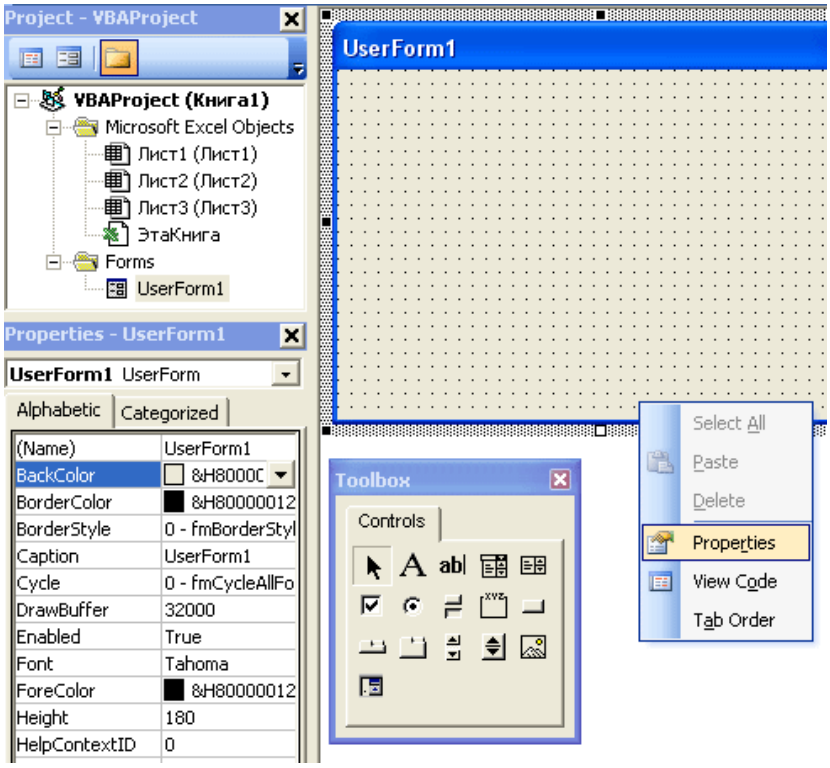
Цель: получение практических навыков построения пользовательских форм и программирования действий, связанных с выводом информации на формы

Вы можете создавать свои окна с элементами управления. Они обычно нужны, когда требуется запросить информацию у пользователя. Разнообразные диалоги настроек, параметров и т.д. Такие окна называются пользовательскими формами.

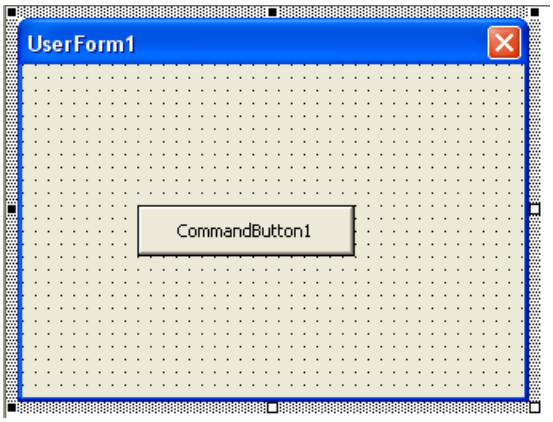
Чтобы создать форму, откройте редактор VBA (Alt + F11), выберите в списке слева Microsoft Excel Objects и в его меню выбираем Insert->UserForm.



На экране появилась пустая форма. В ее контекстном меню выбираем Properties - открывается окно свойств формы.



В этом окне можно изменить заголовок формы, цвет фона, шрифт и многие другие параметры. Кроме окна свойств есть еще инструментальная панель Toolbox. В ней содержатся элементы управления, которые можно разместить на форме. Для начала добавим например кнопку.

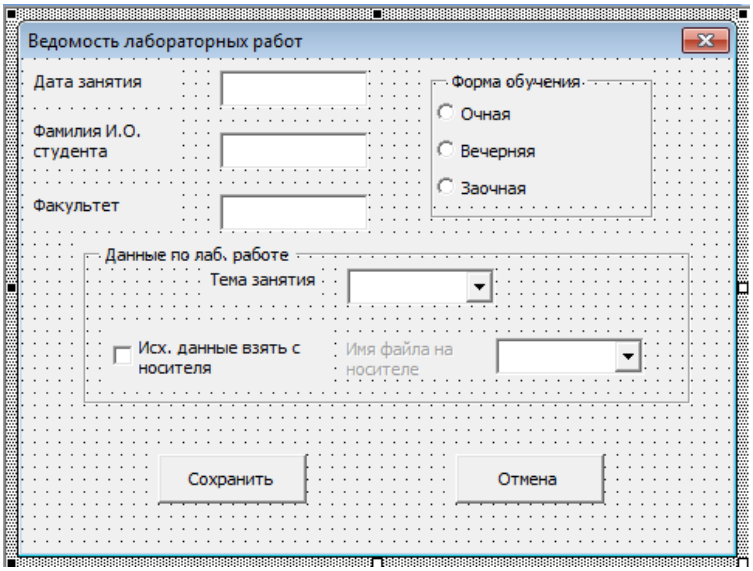


Теперь в окне свойств доступны свойства кнопки. Можно изменить надпись на ней, или добавить картинку. Осталось только привязать к этой кнопке свой код. Это очень просто - по двойному щелчку на кнопке вы попадете в окно редактора VBA, где уже создана процедура обработки нажатия на кнопку.

```
Private Sub CommandButton1_Click()
End Sub
```

Задание

Создать экранную форму «Ведомость лабораторных работ» для заполнения журнала учёта лабораторных работ



Решение

Разобьем задачу на несколько этапов.

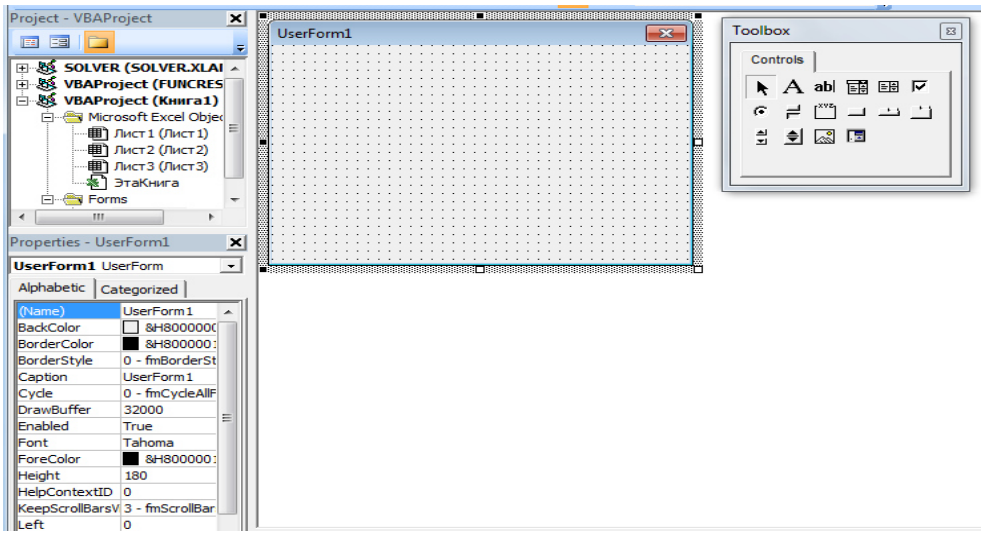
Создание экранной формы

Для этого:

1.1 Открыть новую рабочую книгу в Excel.

1.2 Перейти в редактор Visual Basic (нажмите комбинацию клавиш Alt+Fll).

1.3 Для вставки формы выполните команды Вставка – Экранная форма (для нерусифицированной версии Insert —UserForm). В рабочую книгу будет вставлена экранная форма

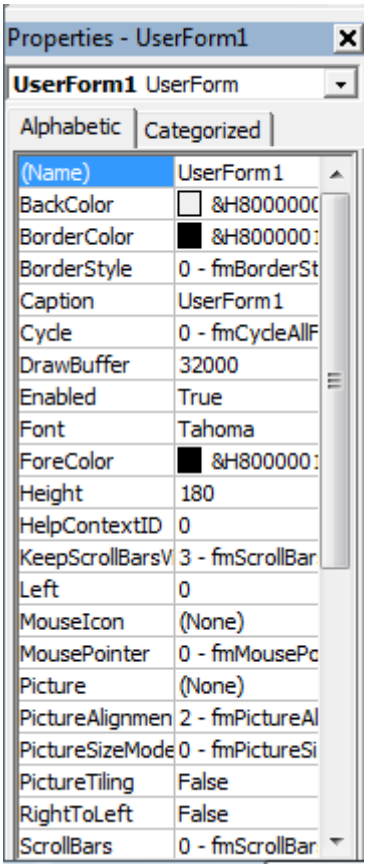


Новая форма представляет собой пустое серое окно со строкой заголовка. Это та основа, на которой пользователь может создать собственные окна любых типов.

1.4 Установим значения свойств формы: т.е. дадим ей имя и введём название в строку заголовка.

Для этого:

- Если окно свойств не отображено на экране, выполнить команды Вид – Окно свойств (или нажать клавишу F4).



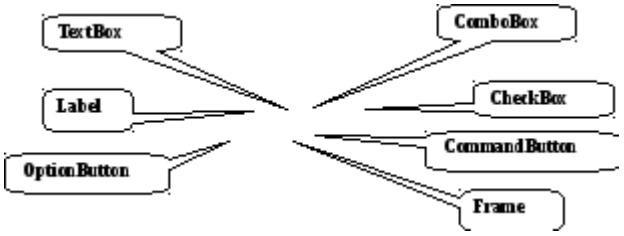
- Выберите свойство Name (Имя), которое находится в верхней части списка, упорядоченного по алфавиту. Вместо UseiForm1 введите Ведомость. Это имя, по которому к форме будет обращаться программа

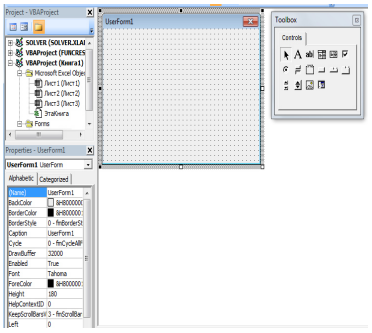
- Для ввода заголовка формы выберите свойство Caption. Вместо UseiForml (Visual Basic присваивает свойствам Name и Caption одинаковые названия) введите Ведомость лабораторных работ. Введённый текст сразу появится в строке заголовка формы

1.5 Вставим элементы управления в экранную форму. Мы хотим получить форму, показанную на рис. 4. Как видно, в форме использованы различные элементы управления .

- Чтобы вставить их в форму, выполните следующие действия:

Если панель элементов не отображена на экране, выполните команды Вид - Панель элементов (View- Toolbox).





- На панели элементов выберите элемент Надпись (Label): A. Поставив указатель мыши на этот элемент, зажмите левую клавишу и переместите его в левый верхний угол формы.
- Если окно свойств элемента Надпись скрыто, нажмите клавишу F4. 'Задайте свойству Caption значение Дата занятия. При этом на форме вместо слова Labell появится надпись «Дата занятия». Свойство Name оставляем без изменения, т.е. за этой надписью закреплено предложенное Visual Basic имя Labell.
- На панели элементов выберите элемент Поле (TextBox): abl, поместите его правее надписи Дата занятия (см. рис. 4). Сохраним за полем предлагаемое Visual Basic имя TextBoxl.
- Используя рис. 4 как образец, продолжайте аналогично предыдущим подпунктам создавать элементы управления и описывать значения их свойств согласно табл. 1.

Таблица 1

Тип элемента управления	Значение свойства Name	Значение свойства Caption	Другие свойства
Надпись	Label1	Дата занятий	-
Поле	TextBoxl	-	-

Надпись	Label2	Фамилия И.О. студента	–
Поле	TextBox2	-	-
Надпись	Label3	Факультет	-
Поле	TextBox3	-	-
Рамка	Frame1	Форма обучения	-
Переключатель	OptionButtonl	Очная	Value=True
Переключатель	OptionButton2	Вечерняя	-
Переключатель	OptionButton3	Заочная	–
Рамка	Frame2	Данные по лаб. работе	-
Надпись	Label4	Тема занятия	-
Поле со списком	ComboBoxl	-	-
Флажок	CheckBoxl	Исх. данные взять с носителя	Value=False
Надпись	Label5	Имя файла на носителе	Enabled=False
Поле со списком	ComboBox2	-	-
Командная кнопка	ComandButtonl	Сохранить	Default=True
Командная кнопка	ComandButton 2	Отмена	Cancel=True

Замечание 1. В форму дважды нужно вставить элемент Рамка (Frame). Сначала следует *разместить на* форме этот элемент, а затем поместить в нём другие элементы управления, иначе ража закроет их собой.

Замечание 2. При вставке элементов управления лучше придерживаться того порядка, который указан в табл. 1 (в противном случае Visual Basic присвоит им имена с другой нумерацией и при автоматизации формы они будут работать неправильно).

Для каждого элемента мы сохраняем предложенное Visual Basic свойство Name, для элементов Надпись. Переключатель (Опция), Рамка и Флажок задаём свойство Caption. Для ряда элементов управления необходимо также задать дополнительные свойства.

Прокомментируем эти дополнительные свойства.

Свойства командных кнопок Кнопка Сохранить. Чтобы командная кнопка работала как кнопка Сохранить, необходимо задать ей значение свойства *Default* (По умолчанию), равное *True* (Истина). В экранной форме только одна командная кнопка может иметь значение *True* свойства *Default.*

Кнопка Отмена. Щелчок по этой кнопке должен вызывать отмену введённых команд. Чтобы командная кнопка работала как кнопка Отмена, следует для неё задать свойству *Cancel* значение *True.* Так же. как и в случае свойства *Default,* в экранной форме только одна командная кнопка может иметь значение *True* свойства *Cancel*

Свойства переключателей Для переключателей в группе Форма обучения прежде всего нужно указать, какой из них будет выбран по умолчанию (т.е. будет находиться во включённом состоянии до выбора, сделанного пользователем).

Для этого нужно задать значение *True* свойства *Value* (Значение). Установим его для первого переключателя Очная. Отметим, что в группе только один переключатель может иметь значение *True* свойства *Value.*

Блокировка элементов управления В нашей форме элемент управления ***Имя файла*** необходим только тогда, когда установлен флажок ***Исходные данные взять с носителя***. Чтобы поле со списком ***Имя файла*** было недоступно пользователю, а также отображалось серым цветом (такой цвет сигнализирует о недоступности элемента), его свойство *Enabled* (Разблокировка) следует положить равным *False* (Ложь). Когда флажок ***Исходные данные взять с носителя*** будет установлен, свойство *Enabled*следует поменять на *True.* Это будет осуществляться программным путём.

Свойства флажка. Чтобы в исходном состоянии флажок не был установлен, следует задать равным False значение его свойства Value.

Сохранить созданную форму с именем Ведомость.

Автоматизация экранных форм

Итак, нами создана экранная форма для ввода сведений об учёте лабораторных работ. Теперь нужно описать процедуры для вывода этой формы на экран и для сохранения введённых в неё данных.

Инициализация экранных форм

Инициализация означает создание процедуры, осуществляющей ввод в форму начальных данных (заполнение полей со списками и ввод даты занятий).

В процедуре инициализации экранной формы должны быть предусмотрены блоки задания исходных данных для полей со списками.

Для создания списков выполните следующие действия:

1. Перейдите в текущую рабочую книгу. Переименуйте рабочий Лист2 на «Списки».
2. На рабочем листе «Списки» введите исходные данные согласно табл. 2.

Таблица 2

	A	B	C
	Тема занятия		**Имя файла**
	Операционная система Windows		Список абонентов
	Текстовый процессор Word		Инвестиционные проекты
	Табличный процессор Excel		Оптимизационные задачи
	СУБД Access		
	Visual Basic		

3. Для присвоения диапазону ячеек имени ***Занятия*** выполнить следующее:
   - выделить диапазон ячеек A2:A6;
   - выполнить Вкладка ленты управления Формулы/(группа определенные имена)/ Присвоить имя:
   - в открывшемся диалоговом окне ввести имя *Занятия;*
4. Аналогично диапазону ячеек C2:C4 присвойте имя *Файл* .
5. Сохраните рабочую книгу с прежним именем.
6. Перейдите в Visual Basic.
7. Дважды шёлкннте по экранной форме (не на элементах управления, а именно по форме!). Откроется окно с заголовком процедуры UserForm_CKck (Щелчок по форме) – рис.8.

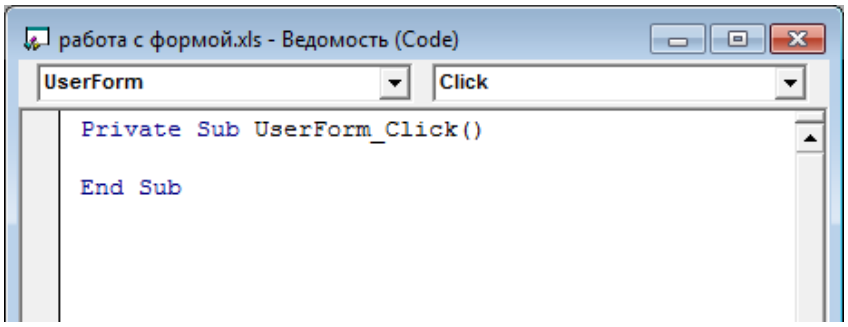


Рис.8

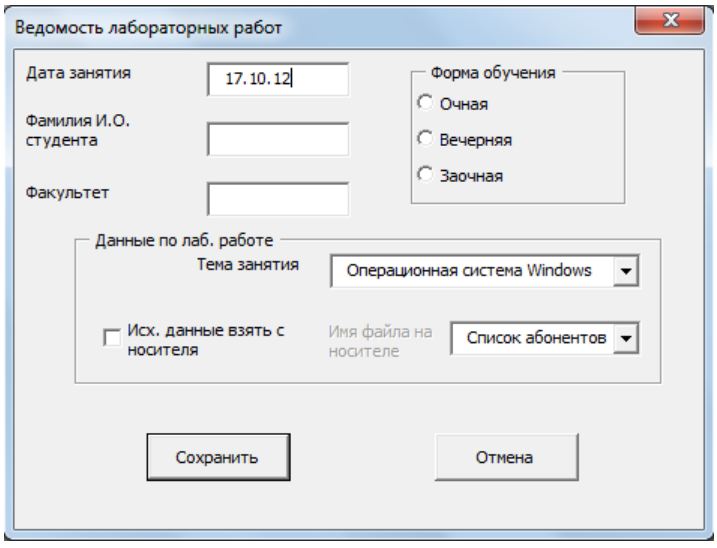
8. Нам пока не нужна процедура UserForm Click. Которая вызывается после щелчка по форме. Нам нужна процедура, которая будет выполняться при открытии формы. На языке программистов она называется процедурой, обрабатывающей событие Activate (Активизация). Список процедур, возможных для формы, находится в правой верхней части окна (рис. 8).

Щёлкнем по кнопке со стрелкой рядом со словом Click и в раскрывшемся списке выберем событие Activate. В окне программы появится заголовок процедуры UserForm_ Activate.

9. Ввести код процедуры для записи исходных данных с листа *Список* в поля со списками и ввода данных

```
Private Sub UserForm_Activate()
With ComboBox1
.RowSource = "Занятия"
.ListIndex = 0
End With
With ComboBox2
.RowSource = "Файл"
.ListIndex = 0
End With
TextBox1 = Format(Now, "dd/mm/yy")
End Sub
```

10. Нажать клавишу F5 для выполнения процедуры. Отобразится экранная форма с заполненными списками *Тема занятия* и *Имя файла,* а также заполненным полем *Дата занятия*



Создание процедуры для вызова формы

Для создания процедуры вызова экранной формы нужно воспользоваться командой Visual Basic «вывести на экран» - Show. Для этого следует выполните следующие действия:

11. Перейти в редактор Visual Basic (если вы не находитесь в нём) и вставить новый модуль в рабочую книгу (Вставка – Модуль)/(Insert-Module).

12. Создать новую процедуру (Вставка – Процедура)/ (Insert-Prosedure) и назвать её *Загрузка_формы.*

13. Ввести код процедуры.

```
Public Sub Загрузка_формы()
Ведомость.Show
End Sub
```

14. Перейти в рабочую книгу на Лист1 и вставить командную кнопку:

- Вкладка Разработчик (группа элементы управления)/Вставить/Кнопка (рис.9)

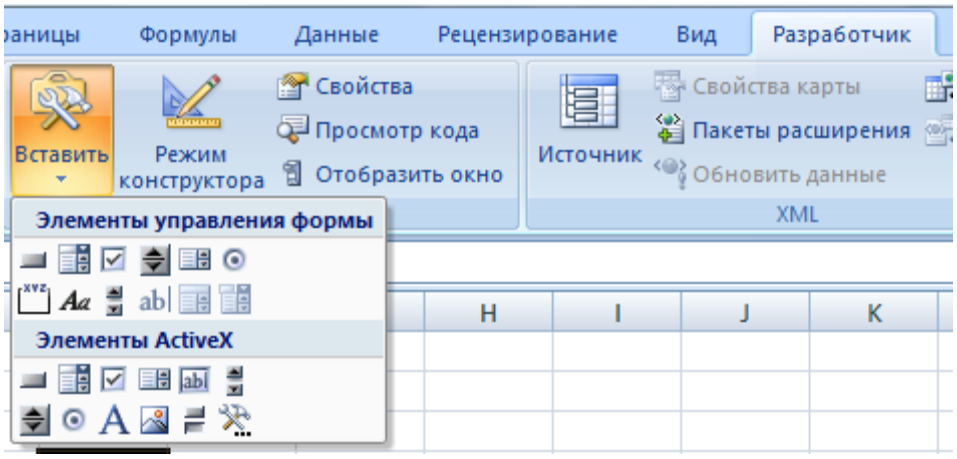
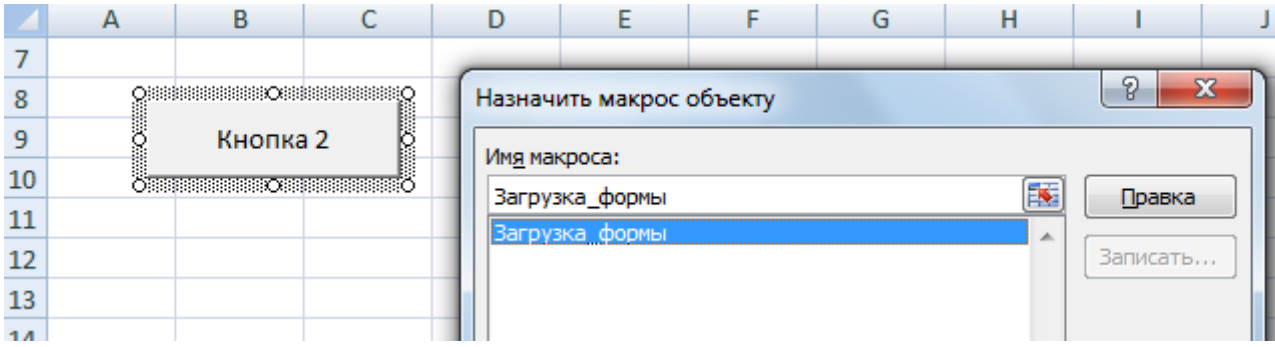


Рис.9

- Разместить кнопку на рабочем листе1

15. После вставки кнопки откроется диалоговое окно Назначить макрос объекту. Выбрать в открывшемся списке Имя макроса пункт



16. Щёлкнуть по командной кнопке, стереть имя Кнопка 2 и дать кнопке имя *Ведомость работ.* После этого следует щёлкнуть вне кнопки, чтобы снять с неё выделение.

17. Щёлкнуть по этой командной кнопке. Откроется наша форма. Обратите внимание: списки *Темы занятий* и *Имя Файла* заполнены, как и поле *Дата занятий.*

18. Закрыть форму.

2.3 **Управление флажком Исходные данные взять с носителя**

При установке флажка ***Исходные данные взять с носителя*** надо снять блокировку с поля ввода Имя файла. За подобные действия должна отвечать процедура, вызываемая событием Change (Изменить). Это событие генерируется Visual Basic всякий раз. Когда изменяется значение свойства *Value* (Значение) какого-либо элемента. Выполните следующую последовательность действий:

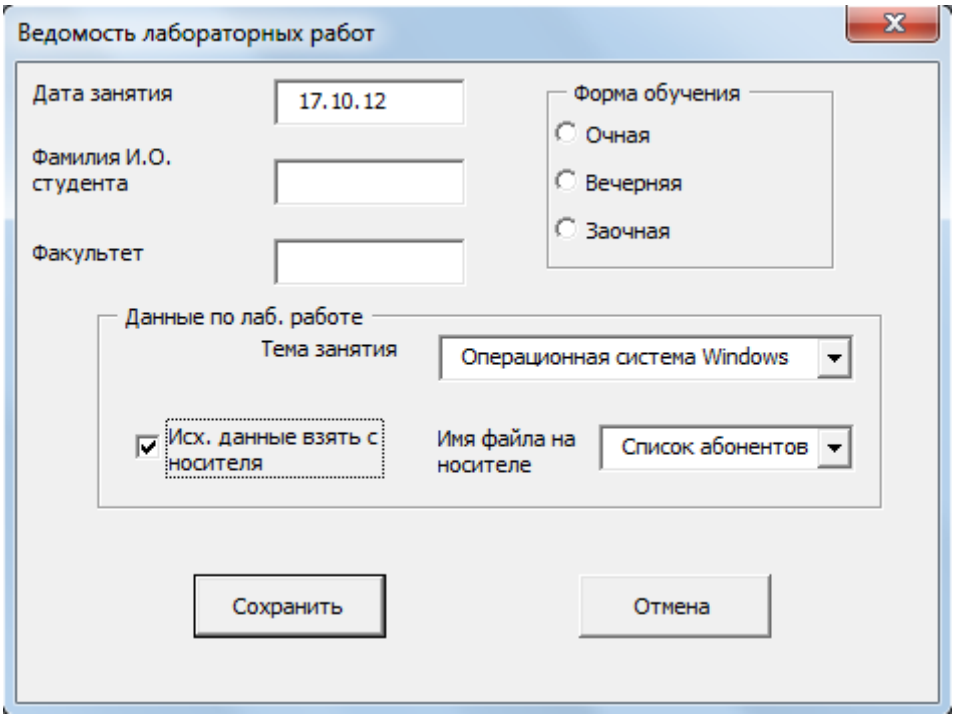
1. Дважды щёлкните по флажку ***Исходные данные взять с носителя***, чтобы открыть окно кода программы.

2. В списке событий в верхней части окна кода выберите событие *Change* и создайте процедуру

```
Private Sub CheckBoxl_Change()
If CheckBoxl. Value = True Then
Label5.Enabled = True
Else
Label5.Enabled = False
End If
End Sub
```

3. Перейдите в рабочую книгу на Лист1 и шёлкните по командной кнопке Ведомость работ. Отобразится форма

4. Установите флажок ***Исходные данные взять с носителя.*** Поле ввода ***Имя файла на носителе*** станет доступным



5. Сохраните рабочую книгу.

Практическое занятие №14
Создание запросов.

Для отбора или поиска данных из одной или нескольких таблиц используются запросы. С помощью запросов можно просматривать, анализировать и изменять данные из нескольких таблиц. Они также используются в качестве источника данных для форм и отчётов. Запросы позволяют вычислять итоговые значения и выводить их в компактном формате, а также выполнять вычисления над группами записей.

Мы будем разрабатывать запросы в режиме ***Конструктора***.

В ***Access*** можно создавать следующие типы запросов:

- ***Запрос на выборку***. Является наиболее часто используемым типом запроса. Запросы этого типа возвращают данные из одной или нескольких таблиц и отображают их в виде таблицы. Запросы на выборку можно также использовать для группировки записей и вычисления сумм, средних значений, подсчета записей и нахождения других типов итоговых значений. Для изменения условий отбора надо изменять запрос.
- ***Запрос с параметрами***. Это запрос, при выполнении отображающий в собственном диалоговом окне приглашение ввести данные или значение, которое требуется вставить в поле. Эти данные или значение могут меняться при каждом обращении к запросу.
- ***Перекрестный запрос***. Используется для расчетов и представления данных в структуре, облегчающей их анализ.

Перекрестный запрос подсчитывает сумму, среднее, число значений или выполняет другие статистические расчеты, после чего результаты группируются в виде таблицы по двум наборам данных, один из которых определяет заголовки столбцов, а другой заголовки строк.

- ***Запрос на изменение***. Это запрос, который за одну операцию изменяет или перемещает несколько записей. Существует четыре типа запросов на изменение:

1. На удаление записи. Этот запрос удаляет группу записей из одной или нескольких таблиц.

2. На обновление записи. Вносит общие изменения в группу записей одной или нескольких таблиц. Позволяет изменять данные в таблицах.

3. На добавление записей. Добавляет группу записей из одной или нескольких таблиц в конец одной или нескольких таблиц.

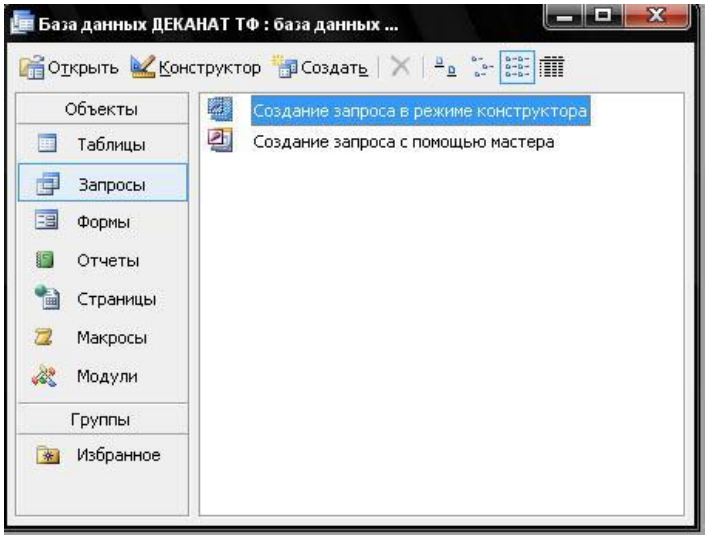
4. На создание таблицы. Создает новую таблицу на основе всех или части данных из одной или нескольких таблиц.

- ***Запросы SQL***. Создаются при помощи инструкций языка ***SQL***, используемого в ***БД***.

Создание запроса на выборку

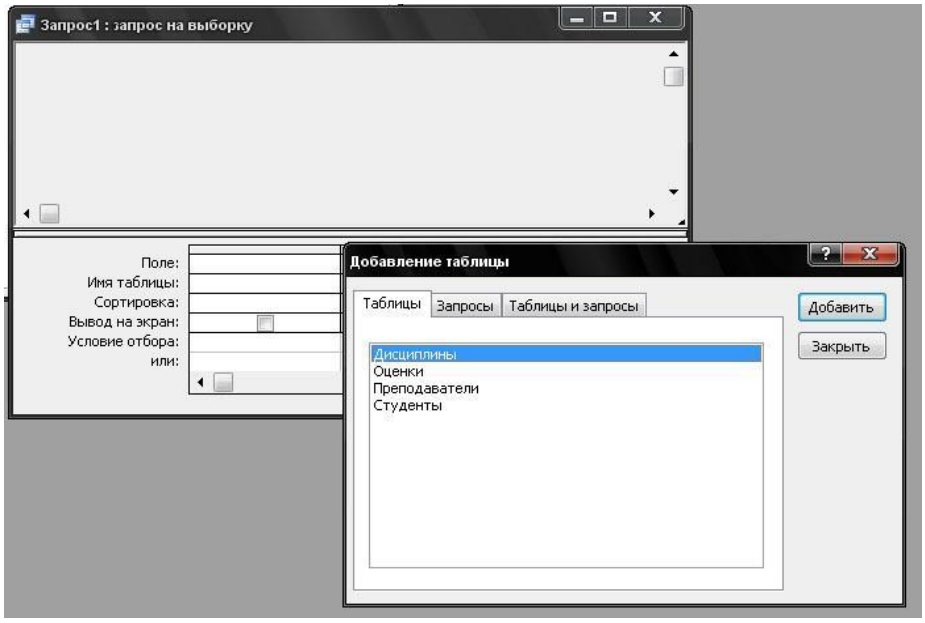
Создадим запрос ***Зарплата***, в котором должны отображаться фамилии преподавателей, их заработная плата, должности и названия преподаваемых ими дисциплин. Эти сведения хранятся в двух таблицах: ***Преподаватели*** и ***Дисциплины***.

Для создания запроса в окне базы данных выбрать объект ***Запросы*** (рис. 23). Дважды щелкнуть по строке «***Создание запроса в режиме конструктора***». Либо один раз щелкнуть по строке «***Создание запроса в режиме конструктора***», а затем по кнопке «*Открыть*» на панели инструментов (рис. 23).



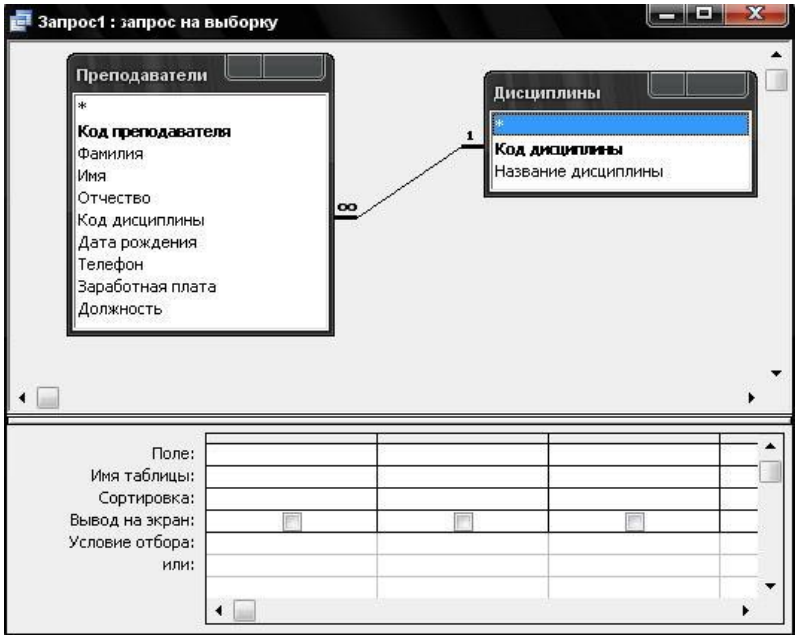
Внешний вид *Конструктора Базы Данных* с открытым разделом «*Запросы*»

Открывается бланк запроса на выборку и окно «***Добавление таблицы***» (рис. 24)**.**



24. Добавление таблиц в открытый запрос

Выбрать таблицы ***Преподаватели*** **и** ***Дисциплины****,* используя кнопку «***Добавить***»**.** Закрыть окно «***Добавление таблицы***» (рис. 25)**.**

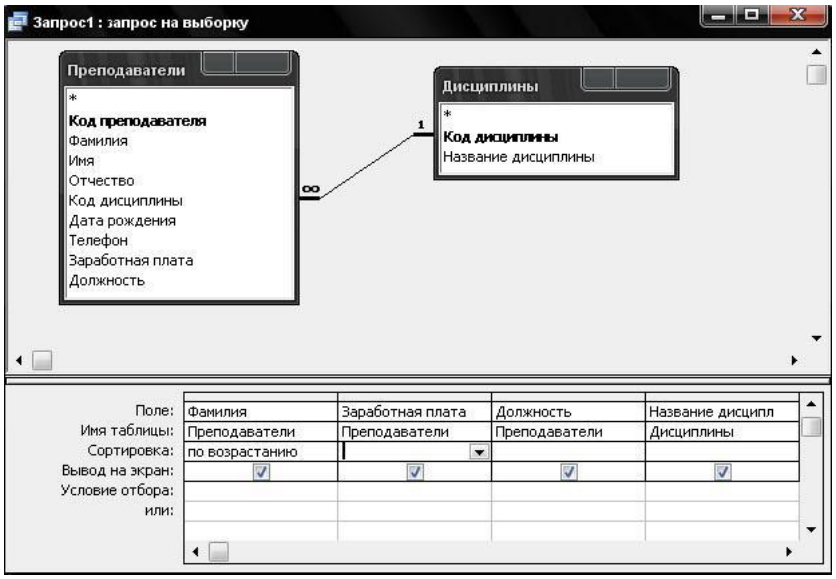


Форма бланка запроса с добавленными таблицами «*Преподаватели*» и «*Дисциплины*»

После этого надо перенести нужные поля в нижнюю часть бланка запроса. Для этого, последовательно устанавливая указатель мыши в нижней части бланка запроса, в окне с названием «***Поле***» два раза щелкнуть по соответствующим полям в таблицах.

При этом в строке «***Поле***» появляется название перенесенного поля, а в строке «***Имя таблицы***» - название таблицы, из которой взято это поле (рис. 26).

В строке «***Сортировка***» можно задать способ сортировки (по возрастанию или убыванию) или отсутствие сортировки. Добавить в запрос поля «Фамилия», «Заработная плата», «Должность», «Название дисциплины» (рис. 26)



Сформированный бланк запроса

При закрытии бланка запроса дать ему имя ***Зарплата***

Для изменения запроса нужно его выделить и щелкнуть по кнопке «Конструктор» или воспользоваться соответствующим контекстно-зависимым меню (рис. 28) в окне базы данных. Откроется бланк запроса, в который можно внести изменения.

Запускается запрос двойным щелчком по его имени. На экране появляется результат выполнения запроса в виде таблицы

Практическое занятие №15
Создание форм

Цель изучение приемов создания простых форм, базирующихся на таблицах и запросах, а также главной кнопочной формы.

Задание

1. Изучить технологию создания форм в среде Access.

2. На базе проекта, созданного в индивидуальной работе разработать формы ленточного типа для редактирования, добавления и удаления записей в таблицах. Каждая таблица должна иметь свою форму для ввода данных. В соответствие с вариантом (документ Var18.doc) создать запрос и ленточную форму, базирующуюся на этом запросе и выводящую требуемую информацию. Разработать главную кнопочную форму. На форме расположить кнопки, открывающие все имеющиеся формы, а также кнопку "Выход", закрывающую главную кнопочную форму.

3. Продемонстрировать работу СУБД на компьютере.

Общий порядок выполнения работы

Сначала, в соответствие с вариантом задания, разработайте запрос. Порядок разработки запроса описан в предыдущей лабораторной работе. Затем, приступайте к разработке комплекта форм.

Порядок разработки простых форм

1. В окне базы данных выбрать вкладку "Форма" и нажать кнопку "Создать".

2. Выбрать в поле "Источник данных ..." имя таблицы или запроса, на котором будет базироваться форма, а также способ создания формы - мастер форм.

3. В окне "Создание форм" переведите поля, размещаемые на форме из области "доступные поля" в область "выбранные поля" и нажмите "Далее".

4. Выберите тип формы - ленточный и нажмите "Далее".

5. Выберите стиль (фон) формы и нажмите "Далее". Для формы, выводимой на печать, желательно не задавать темный фон.

6. Задайте имя формы (в соответствие с базовой таблицей или запросом) и нажмите "Готово".

7. Откройте созданную форму, просмотрите ее и, при необходимости, перейдите в режим конструктора и вручную измените подписи в области заголовка (если они слишком широки по сравнению с содержимым соответствующих полей) или измените размер и расположение полей в рабочей области.

Создание главной кнопочной формы

1. Выполните ***Сервис/Служебные программы/Диспетчер кнопочных форм***.

2. В окне "Диспетчер кнопочных форм" нажмите "Изменить...".

3. В окне "Изменение страницы кнопочной формы" нажмите "Создать" - откроется окно "Изменение элемента кнопочной формы" с тремя полями: в поле "Текст" введите надпись, соответствующую действию кнопки; в поле "Команда" выберите команду "Открытие формы в режиме редактирования" - для кнопок, открывающих форму или команду "Выход из приложения" - для кнопки завершения работы; информация, выбираемая в третьем поле, зависит от второго - для открытия формы надо указать имя открываемой формы.

4. Повторите предыдущий пункт для создания остальных кнопок кнопочной формы.

5. Закройте окно создания главной кнопочной формы, нажав "Закрыть"

Варианты заданий к занятию

№ вар.	Название формы	Поля, выводимые на форму
1, 16	Сдача экзаменов	Фамилия, название специальности, название дисциплины, оценка
2, 17	Табельный учет	Фамилия, должность, разряд, ставка, количество часов за месяц
3, 18	Отпуск товаров заказчикам	Название товара, единица измерения, отпущенное количество, название организации
4, 19	Потребление продуктов	Год и месяц, наименование продукта, единица измерения, количество, цена
5, 20	Выдача книг	Автор, название книги, читательский билет, фамилия читателя, дата возвращения
6, 21	Нарушители ПДД	Фамилия водителя, название нарушения, дата нарушения, гос. номер автомобиля
7, 22	Протокол соревнований	Номер участника, фамилия, группа, время старта, время финиша
8, 23	Перевозки	Дата, пункт назначения, название груза, единица измерения, перевезенное количество, марка автомобиля
9, 24	Итоги сессии	Фамилия студента, группа, название дисциплины, экзаменационная оценка, дата

		сдачи
10, 25	Фактическая нагрузка	Дата, номер пары, название нагрузки, номер группы, название темы, специальность
11, 26	CD-картотека	Название CD-ROM, фамилия владельца, название файла, объем в Кбайтах
12, 27	Успеваемость в классе	Фамилия ученика, дата, название дисциплины, оценка.

Задание

1. Изучить информацию о составных формах и приемах их разработки.

2. В соответствии с вариантом задания (документ Var19.doc), добавить в СУБД, созданную в лабораторных работах №17-18, составную форму для отбора записей в подчиненной форме по значению, выбираемому из списка на главной форме. Технология создания описана в разделе "Порядок выполнения".

3. Продемонстрировать работающий проект.

Порядок выполнения

1. Запустите создание формы в режиме мастера.

2. Выберите в качестве источника записей таблицу, являющуюся главной среди связанных таблиц – источников.

3. В первом окне Мастера выберите поля таблицы, указанной в п.2, для размещения на форме. Не выходя из первого окна, выберите из списка "Таблицы и запросы" таблицу – источник данных для подчиненной формы (подчиненность определите по схеме данных) и добавьте в список "Выбранные" требуемые поля. Нажмите "Далее".

4. На втором шаге Мастера выберите вид представления данных так, чтобы на схеме подчиненная форма была выделена рамкой (переключатель "Подчиненные формы" должен быть включен).

5. На третьем шаге выберите ленточный вид подчиненной формы.

6. На следующих шагах Мастера задайте стиль и имя формы.

7. При необходимости перейдите в режим конструктора и доработайте полученную форму вручную.

8. Откройте разработанную форму и измените значение счетчика записей на главной форме – записи в подчиненной форме должны меняться в соответствии с содержимым поля на главной форме.

9. Добавьте кнопку на Главную кнопочную форму проекта,

запускающую созданную составную форму.

Варианты заданий

№ вар.	Источник для главной формы	**Поле на главной форме**
1, 16	Данные о специальностях	Название специальности
2, 17	Тарифная сетка	Должность
3, 18	Товары	Название товара
4, 19	Продукты	Наименование продкута
5, 20	Читатели	Фамилия читателя
6, 21	Нарушения	Название нарушения
7, 22	Участники	Шифр группы
8, 23	Доставка	Название груза
9, 24	Дисциплины	Название дисциплины
10, 25	Виды нагрузки	Название нагрузки
11, 26	Владельцы	Фамилия владельца
12, 27	Ученики	Фамилия ученика
13, 28	Информация о местах	Номер комнаты
14, 29	Поставщики	Название поставщика
15, 30	Промышленные предприятия	Название предприятия

Разработка макроса, автоматизирующего работу с запросами

Цель: изучение технологии разработки запросов на создание и обновление таблиц и автоматизации выполнения запросов при помощи макроса.

Задание

1. Изучить информацию о запросах на создание и обновление таблиц, выполнение групповых операций в запросах, а также технологию разработки и применения макросов.

2. В соответствии с вариантом задания (документ Var20.doc)

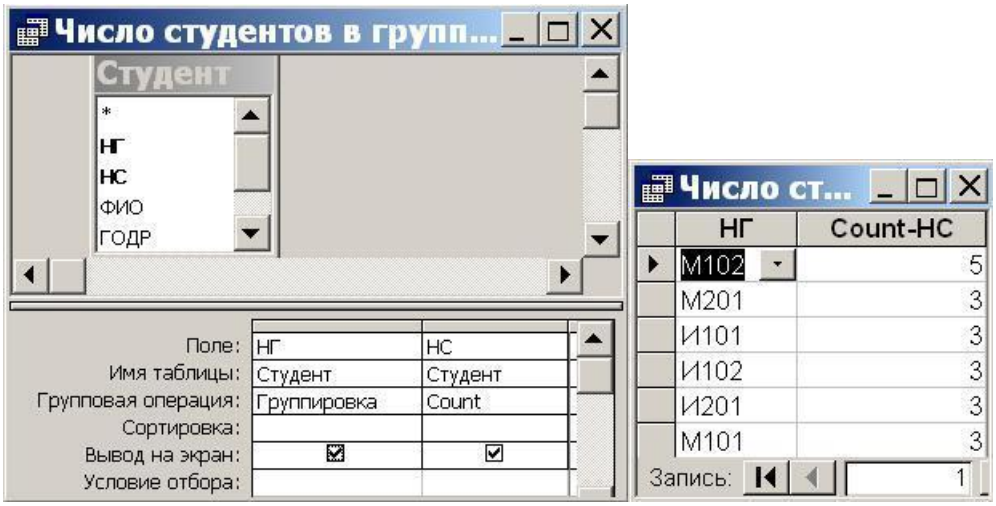
добавить в одну из таблиц проекта поле, которое будет обновляться запросом (в 7 и 22 вариантах – создать отдельную таблицу). Создать макрос, автоматизирующий выполнение запросов, в соответствии с приведенным ниже примером. На главную кнопочную форму добавить кнопку, запускающую макрос.

3. Продемонстрировать работу макроса на компьютере.

Пример

В качестве исходной БД будем использовать базу, спроектированную в разделе **Проектирование базы данных**электронного учебника.

Пусть необходимо рассчитать количество студентов в группах и внести эти данные в поле **КОЛ** таблицы **ГРУППА.** Информация о количестве студентов хранится в таблице **Студент**. Для решения задачи следует сгруппировать студентов, принадлежащих одной группе, подсчитать количество студенов в группах и внести эти данные в таблицу **ГРУППА**. Подсчет количества студентов реализован запросом **Число студентов в группе**



На панели инструментов нажмем кнопку **Групповые операции** – в бланк запроса будет добавлена соответствующая строка. Заменим слово **Группировка** в столбце НС на функцию **Count**. Сохраним запрос под именем **Число студентов в группах**.

Замечание. *В соответствии с вариантами потребуется подбирать поля запроса по смыслу задания. Обычно первым является поле, содержащее признак группировки, а второе – обрабатываемое поле. Функции обработки также могут быть различны, например, сумма значений полей группы* ***sum***.

Преобразуем запрос на выборку в запрос на создание таблицы. Для этого в режиме конструктора выберем меню **Запрос/Создание таблицы**. В окне **Создание таблицы**введем имя таблицы – **Число студентов**.

При открытии запроса будут выданы предупреждения о создании таблицы и о количестве добавляемых записей. При последующих открытиях – выдается предупреждение об удалении старой таблицы **Число студентов**.

Информация в таблице **Число студентов** должна быть использована для обновления поля**КОЛ**таблицы**ГРУППА**. Для этой цели создадим запрос на обновление**Обновление ГРУППА_КОЛ**. Оформим окно конструктора запроса

Окно конструктора запроса на обновление

В схему данных запроса добавлены таблица **Группа**и таблица**Число студентов**, полученная после выполнения запроса на создание**Число студентов в группе**. Преобразуем запрос на выборку в запрос на обновление. Для этого выполним команду меню**Запрос/Обновление**. В строке**Обновление**введем имя поля [**Count-НС**] таблицы**Число студентов**, из которой выбираются значения для обновления.

Сохраним запрос на обновление под именем **Обновление ГРУППА_КОЛ.**

Создадим макрос, состоящий из макрокоманд

Сохраним макрос под именем **Расчет количества студентов в группах**. Для выполнения макроса необходимо в окне БД нажать

кнопку**Запуск.**

Практическое занятие №16
Создание кнопочной формы.

Кнопочное меню представляет собой форму, на которой расположены элементы управления – кнопки с поясняющими надписями. Щелчок на кнопке открывает соответствующую таблицу, запрос, форму или отчет. Меню - удобный инструмент работы с базами данных, и он практически всегда присутствует в базах созданных для предприятий или фирм.

Кнопочное меню создают с помощью *Диспетчера кнопочных форм.*

Практическая часть.

Ход работы.

1) Откройте свою базу данных.

2) Создайте форму с помощью <Мастера форм> на базе таблицы <Ведомость успеваемости>.

· Откройте таблицу <Ведомость успеваемости>.

· Выберите закладку <Формы >, щелкните мышкой по кнопке <Другие формы>. 

· В появившемся диалоговом окне выберите <Мастер форм>.

· В поле <Таблицы/Запросы> выберите таблицу <Ведомость успеваемости>, в поле <Доступные поля> выберите поля <Фамилия>, <Имя> и перенесите их стрелкой в поле <Выбранные поля>. Также перенесите поля с названием предметов, щелкните по кнопке <Далее>.

· Выберите внешний вид формы – ***Табличный***, щелкните по кнопке <Далее>.

· Выберите требуемый стиль (н-р, ***Обычная***), щелкните по кнопке <Далее>.

Задайте имя формы **<Успеваемость>** и щелкните по кнопке <Готово>. В результате получите форму, в которой можно менять данные и вводить новые значения.

· Закройте форму.

3) Создайте форму на основе таблицы **<Преподаватели>.**

· Откройте таблицу <Преподаватели>.

· Выберите закладку <Формы >, щелкните мышкой по кнопке <Другие формы>. 

· В появившемся диалоговом окне выберите <Мастер форм> .

· Выберите внешний вид формы - < ***ленточный***>.

· Выберите любой стиль.

· Получите готовую форму. Сохраните ее под именем **<Преподаватели>.**

· Закройте форму.

·

4) Создайте форму **<Личные данные>** с помощью инструмента <Пустая форма>

· На вкладке **Создание** в группе **Формы**щелкните **Пустая форма**. 

Access открывает пустую форму в режиме макета и отображает область **Список полей**.

· В области **Список полей** щелкните знак плюс (+) рядом с таблицей или таблицами, содержащими поля, которые нужно включить в форму.

· Чтобы добавить поле к форме, дважды щелкните его или перетащите его на форму. Чтобы добавить сразу несколько полей, щелкните их последовательно, удерживая нажатой клавишу CTRL. Затем перетащите выбранные поля на форму.

· Закройте окно списка полей.

· Перейдите в режим Конструктора

Примечание 1 *Размер окошка для названия поля и для его значений меняются мышкой.*

Для этого выделите черный квадратик рамки (рамка станет цветной), установите курсор на границу рамки и с помощью двунаправленной стрелки измените размеры рамки.

Примечание 2 *С помощью кнопок панели инструментов Шрифт меняйте соответственно цвет фона, текста, линии/границы и т.д.*

· Расположите элементы удобно по полю.

· Задайте размер текста поля <Фамилия> равным **24 пт**, шрифт - **синего цвета**.

· Увеличьте в высоту рамку поля <Фотография>.

· Сохраните форму с именем **<Данные студентов>.**

· Посмотрите все способы представления форм: в режиме *Конструктора*, режиме *Макета* и режиме *Форм.*

· Закройте форму.

5) Добавьте в таблицу <Личные данные> логическое поле <**Институт**> (т.е., собирается ли в дальнейшем учащийся поступать в институт). Значение этого поля <*ДА*> или <*НЕТ*>.

· Откройте таблицу <Личные данные> в режиме *Конструктор*. Добавьте поле с именем <Институт> и типом ***Логический***. Закройте таблицу.

· Перейдите на закладку *Формы* и откройте форму <Данные студентов> в режиме *Конструктор*

·  Щелкните по кнопке <Список полей> на панели инструментов, выделите название <Институт> и перетащите его мышкой в область данных, появиться значок и надпись <Институт>.

· Расположите новые элементы по правилам оформления формы (с помощью мыши).

· Закройте <Список полей>

Примечание 3 *Если флажок установлен, поле в таблице имеет значение <**ДА**>, если*

*снят, то <**НЕТ**>.*

· Перейдите в режим <*Раздельная форма***>** и посмотрите записи. Установите флажки у восьми разных учащихся.

· Закройте форму, ответив утвердительно на вопрос о сохранении.

6) Создайте кнопочную форму **<Заставка>** с помощью *Конструктора*.

· Щелкните по кнопке <Создать>.

· Выберите <Конструктор>. Появиться пустая форма. Задайте мышкой ширину формы, равную 10см, а высоту – 7см.

· Сохраните работу с именем **<Заставка>.**

· Откройте созданную форму <Заставка> в режиме Конструктора.

· Выберите на панели инструментов <Элементы управления> кнопку **Аа** – <*Надпись*>. Курсор мышки примет вид крестика с «приклеенной» буквой **А**. Щелкните мышкой по месту начала надписи и введите:

База данных

«Гостиница»

(после слов **База данных** нажмите одновременно комбинацию клавиш **Shift+Enter**.)

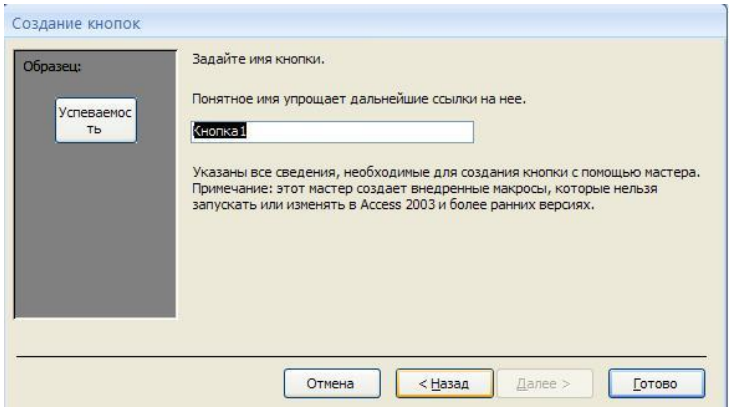
·  Нажмите клавишу <Enter>. Выберите размер букв **18**, а выравнивание **- по центру**. Цвет фона – **голубой**. Растяните мышкой надпись на ширину окна.

· Выберите на панели элементов значок **- Кнопка**. Щелкните мышкой по тому месту области данных, где должна быть кнопка. Появиться диалоговое окно <Создание кнопок>.

· Выберите категорию <Работа с формой>, а действие <Открыть форму>, и щелкните по кнопке <Далее>.

· Выберите форму <Успеваемость> , открываемую этой кнопкой щелкните по кнопке <Далее>. В следующем окне также щелкните по кнопке <Далее>.

· В следующем окне поставьте переключатель в положение <**Текст**>, наберите в поле слово <Успеваемость> и щелкните по кнопке <Далее>.



· Задайте имя кнопки <**Успеваемость>** и щелкните по кнопке <Готово>.

Примечание 3 *Размер и расположение кнопок можно менять мышкой в режиме* ***Конструктор***.

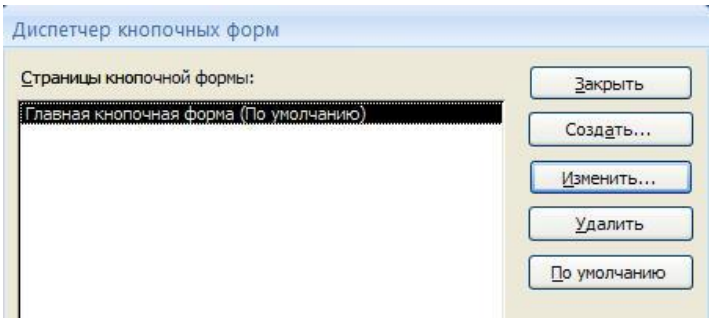
Самостоятельно создайте кнопки для форм <**Личные данные>** и <**Преподаватели>**.

· Перейдите в режим формы Теперь при щелчке мышью по соответствующим кнопкам будут открываться соответствующие формы для работы.

· Закройте форму.

7) Создайте кнопочную форму при помощи **Диспетчера кнопочных форм**.

ü Откройте вкладку ***Работа с базами данных,*** команда ***- Диспетчер кнопочных форм***. Вы получите диалоговое окно, представленное на Рисунке 6.



Щелкните в этом окне по кнопке <Изменить>.

В следующем окне щелкните по кнопке <Создать> и в появившемся окне измените содержимое полей в соответствии с Рисунком 7 (**Команду** и **Форму** выбирайте из списка, а не набирайте вручную). Щелкните по кнопке <ОК>.

Практическое занятие №17

Создание файла проекта базы данных: создание форм и их форматирование.

Цель Освоение приемов проектирования базы данных, описания структуры таблиц и связей между ними.

Ход работы:

1. Изучить теоретическую часть.
2. Выполнить задания практической части.
3. Представить файлы для проверки преподавателю.

Задание. Разработать структуру БД «Студенты».

1. **Проектирование и создание базы данных**

Процесс создания базы данных рассмотрим на примере разработки информационной системы «Студенты», которая должна хранить информацию о студентах и их экзаменационных оценках.

В результате проектирования был сделан вывод о необходимости создания в ней 5-ти таблиц:

1) Студенты – для хранения основных данных о студенте;

2) Оценки – для хранения информации об оценках студентов;

3) Институты – справочник институтов;

4) Специальности – справочник специальностей;

5) Предметы – справочник предметов.

Для создания файла базы данных в папке хранения Ваших файлов вызовем контекстное меню и в нем выберем команду Создать → Microsoft Access База данных. Зададим имя базы данных Студенты (автор <Ваша фамилия>). Откроем базу данных двойным щелчком на созданном файле.

2. Описание структуры таблиц и связей

Выберем на ленте вкладку **Создание** и в группе **Таблицы** нажмем на кнопку **Конструктор** таблиц. По умолчанию для окна базы данных установлен параметр Вкладки, поэтому внутри главного окна мы увидим вкладку (вложенное окно с ярлычком сверху) Конструктора таблиц, показанное на рисунке 1.1 (данные двух полей уже заполнены и была нажата кнопка Сохранить на верхней рамке окна).

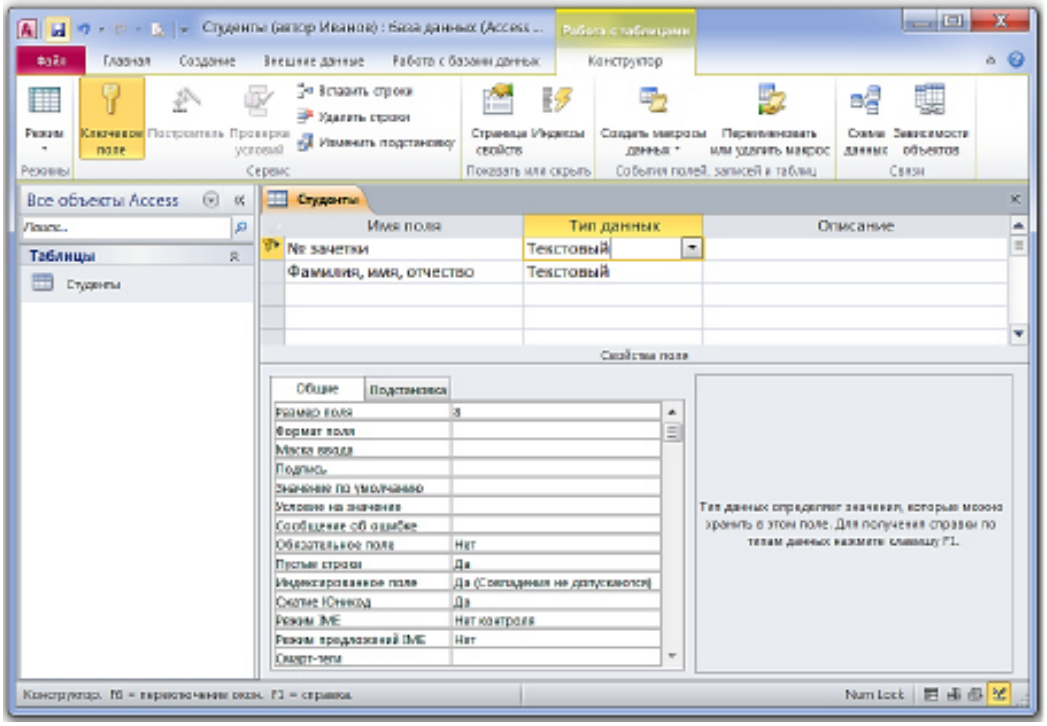


Рисунок 1.1 – Описание структуры таблицы Студенты в Конструкторе

Далее в Конструкторе добавим остальные поля в соответствии с данными таблицы 1.1, т.е. зададим имя, тип данных, размер или формат каждого поля таблицы, а также ключевое поле (если необходимо), индексированные поля и подписи. После чего закроем вкладку Конструктора таблицы Студенты (крестиком справа на темно-серой полоске или из контекстного меню ярлычка) с сохранением изменений структуры.

Затем снова выберем команду Создание → Конструктор таблиц и опишем структуру следующей таблицы – Оценки в соответствии с данными таблицы 1.2. Сохраним таблицу и закроем Конструктор данной таблицы.

Аналогично поступим при создании еще трех таблиц – Институты (структура приведена в таблице 1.3), Специальности (структура приведена в таблице 1.4) и Предметы (структура приведена в таблице 1.5).

В результате получим в базе данных 5 пустых таблиц с заданной структурой. При необходимости в любой момент можно обратиться к модификации структуры каждой из таблиц, открыв ее в Конструкторе.

Таблица 1.1 – Структура таблицы **Студенты**

Таблица 9.1 – Структура таблицы **Студенты**

Имя поля	Тип данных	Размер поля	Индексированное поле
№ зачетки	Текстовый	8	**Ключевое поле**
Фамилия, имя, отчество	Текстовый	45	Нет
Дата поступления	Дата/время	Краткий формат даты	Нет
№ института	Числовой	Байт	Да (Допускаются совпадения)
Код специальности	Текстовый	9	Да (Допускаются совпадения)
Курс	Числовой	Байт	Нет
Группа	Текстовый	4	Нет

Таблица 1.2 – Структура таблицы **Оценки**

Таблица 9.2 – Структура таблицы **Оценки**

Имя поля	Тип данных	Размер поля	Индексированное поле	Обязательное поле
№ зачетки	Текстовый	8	Да (**Допускаются совпадения**)	Да
Семестр	Числовой	Байт	Нет	Да
№ предмета	Числовой	Целое	Да (Допускаются совпадения)	Да
Оценка	Текстовый	1	Нет	Да
Дата получения	Дата/время	Краткий формат даты	Нет	Да
Преподаватель	Текстовый	45	Нет	Да

Таблица 1.3 – Структура таблицы **Институты**

Таблица 9.3 – Структура таблицы **Институты**

Имя поля	Тип данных	Размер поля	Индексированное поле
№ института	Числовой	Байт	**Ключевое поле**
Название института	Текстовый	120	Нет

Таблица 1.3 – Структура таблицы **Специальности**

Таблица 9.1 – Структура таблицы **Специальности**

Имя поля	Тип данных	Размер поля	Индексированное поле
Код специальности	Текстовый	9	**Ключевое поле**
Название специальности	Текстовый	120	Нет

Таблица 1.5 – Структура таблицы **Предметы**

Таблица 9.5 – Структура таблицы **Предметы**

Имя поля	Тип данных	Размер поля	Индексированное поле
№ предмета	Числовой	Целое	**Ключевое поле**
Название предмета	Текстовый	120	Нет

Далее задаем связи (Один ко многим) между таблицами в базе. Для этого на вкладке ленты **Работа с базами данных** выбираем в группе **Отношения** команду **Схема данных**, добавляем в окно схемы все таблицы и, перетаскивая название поля первичного ключа к

аналогичному полю другой таблицы создать связи. При этом задаем в окне Изменение связей для всех связей между таблицами 3 условия: обеспечения целостности данных, каскадное обновление связанных полей и каскадное удаление связанных записей. Схема базы данных показана на рисунок 1.2.

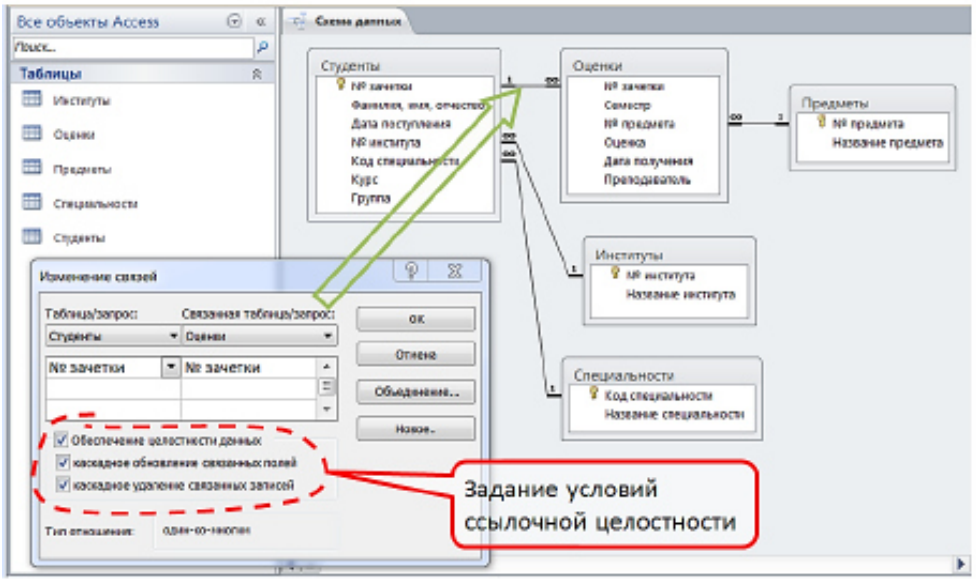


Рисунок 1.2 – Схема базы данных и задание условий ссылочной целостности для связи между таблицами Студенты – Оценки

Практическое занятие №18
Создание сложных многотабличных форм.

Целевая установка:

Учебные цели:

1. Исследовать особенности проектирования реляционной базы данных.

2. Получить практические навыки по использованию табличных бах данных экономического

Сложные формы помимо кнопок могут содержать поля различного назначения , раскрывающиеся списки, флажки, формулы т.д.

Создание сложных форм рассмотрим на примере формы для работы с подчиненной таблицей Заказы. Для таблицы «**Заказы**» можно создать форму с помощью**Мастера создания форм** и «доработать» ее с помощью**Конструктора**.

Задание 7*. Создайте с помощью* ***Мастера*** *форму для таблицы «****Заказы****», включив в нее только поля «****НомерЗаказа****», «****ДатаПриема****» и «****ДатаИсполнения****». В последнем диалоговом окне установите переключатель «****Изменение макета формы****» (это приведет к*

*переключению в режим **Конструктора**) и щелкните кнопку **Готово**.*

*В окне конструктора расширьте форму так, чтобы в ней можно было разместить новые элементы (переместите поле **Примечание формы** вниз с помощью мыши).*

*Измените надписи перед полями «**НомерЗаказа**», «**ДатаПриема**» и «**ДатаИсполнения**», установив курсор в надпись двойным щелчком и введя с клавиатуры строки «**Номер заказа**», «**Дата приема**» и «**Дата исполнения**» соответственно.*

Дата приема заказа обычно совпадает с текущей датой. Поэтому можно установить для этого поля значение по умолчанию – текущую дату и запретить ее «ручной» ввод с клавиатуры. Для этого:

1) *щелкните по полю, находящемуся за надписью «**Дата приема**» правой кнопкой мыши и в открывшемся контекстном меню выберите строку **Свойства**;*

*2) в диалоговом окне команды на вкладке «**Данные**» выберите строку «**Значение по умолчанию**» и щелчком по кнопке **…** вызовите **Построитель выражений** (см. рис.);*

*3) в окне программы-построителя раскройте щелчком по значку + папку «**Функции**» и двойным щелчком вложенную в нее папку «**Встроенные функции**»;*

*4) в списке категорий встроенных функций выберите категорию «**Дата/время**» и в перечне функций этой категории – функцию «**Date**», задающую текущую системную дату, установленную в компьютере;*

*5) вставьте эту функцию щелчком по кнопке **Вставить** в выражение, вычисляющее значение по умолчанию;*

6) закройте построитель выражений (см. рис.);

7) в диалоговом окне свойств поля установите, что к полю нет доступа, и есть блокировка;

8) закройте окно.

***Задание 8.** Для включенияв форму данные об услуге можно воспользоваться раскрывающимся списком. Вставьте раскрывающийся список в форму, выполнив следующие операции:*

1. *Выберите на панели элементов управления элемент «**Поле со списком**». Разместите поле на форме с помощью мыши.*

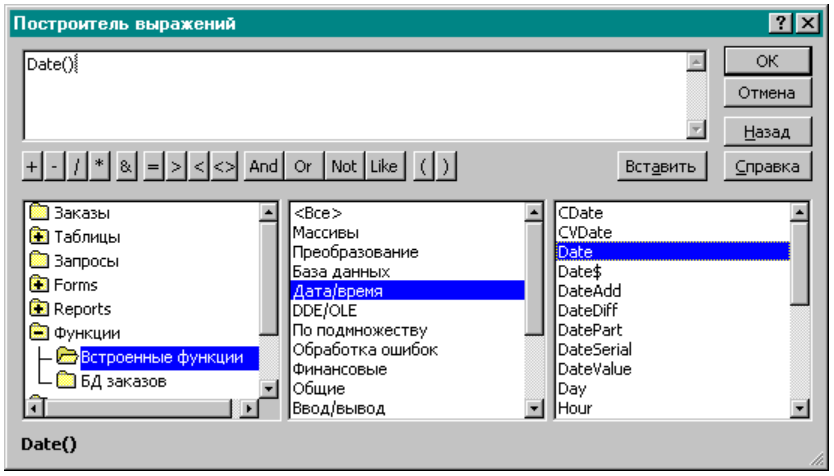
2. *В открывшемся диалоговом окне установите переключатель «**Поле со списком использует значения из таблицы или запроса**» и щелкните кнопку **Далее**.*

3. *В следующем диалоговом окне установите в группе «**Показать**» переключатель «**Таблицы**» и выберите в открытом списке таблиц таблицу «**Услуги**». Щелкните кнопку **Далее**.*

4. *Выберите в очередном окне поля «**КодУслуги**» и «**Наименование**» таблицы «**Услуги**» для включения в форму и щелкните кнопку **Далее**.*

5. *В следующем окне установите флажок «**Скрыть ключевой столбец**» и щелкните кнопку **Далее** для перехода к очередному окну **Мастера**.*

6. *Установите переключатель «**Сохранить в поле**» и выберите из списка поле «**КодУслуги**» таблицы «**Заказы**». Щелчком по кнопке **Далее** перейдите в следующее окно.*



7. *Задайте имя «**Услуга**» и щелкните кнопку **Готово**.*

Пользователю будет удобнее работать, если наряду с раскрывающимся списком услуг в форму вставить поле **Стоимость**.

***Задание.** Для включения поля **Стоимость** в форму заказа выполните следующие операции:*

1. *На **Панели элементов** выберите элемент **Поле** и введите поле в форму.*

3. *Щелкните по полю правой кнопкой мыши и в открывшемся контекстном меню выберите строку **Свойства**;*

2. *В диалоговом окне команды на вкладке «**Данные**» выберите строку «**Данные**» и щелчком по кнопке **…** вызовите **Построитель выражений***

3. *Запишите формулу для определения стоимости (выбора из таблицы «**Услуги**» стоимости по коду услуги). Для этого нужно ввести выражение, содержащее, например, следующую функцию:*

=DLookUp("[Стоимость]";"Услуги";"[Услуги]![КодУслуги] = " & [Forms]![Данные о клиентах]![Заказы]![КодУслуги])

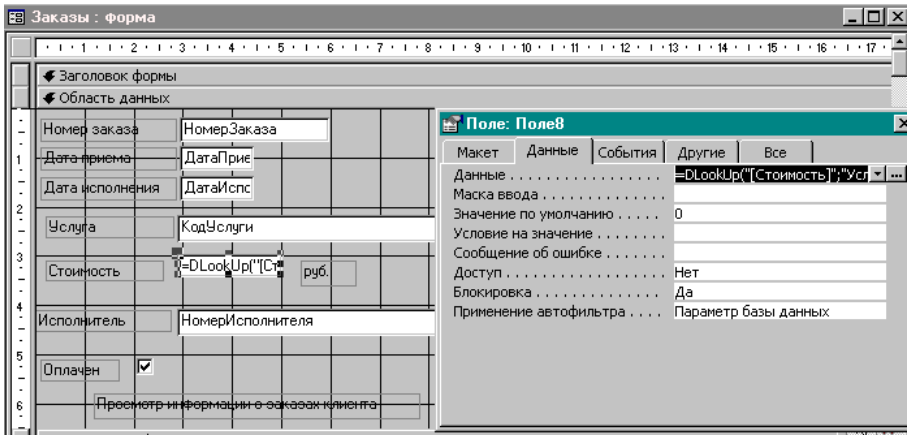
*(стоимость выбирается из поля «**Стоимость**» таблицы «**Услуги**», причем выбирается стоимость той услуги, код которой*

будет зафиксирован при оформлении заказа клиентом – см. рис. ниже).

4. Установите блокировку и отсутствие доступа в свойствах поля (стоимость нельзя менять при оформлении заказа).

Задание 10. *Разместите самостоятельно на форме поле со списком для выбора* ***исполнителя*** *заказа.*

Для отображения информации о том, оплачен ли заказ, на форме следует разместить Флажок.



Задание. *Для размещения флажка выполните следующие операции:*

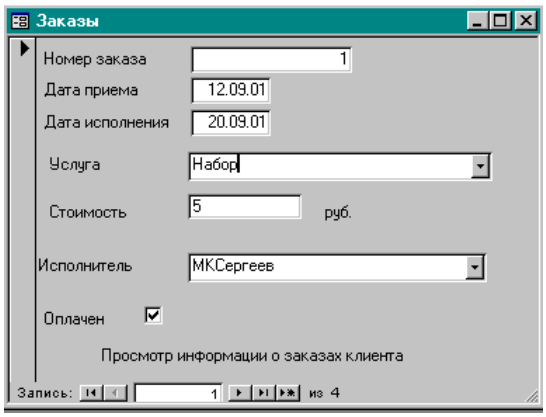
1. *Выбрать элемент управления «**Флажок**» на **Панели элементов** и с помощью мыши разместить его на форме.*

2. *Щелчком мыши установить текстовый курсор в поле подписи флажка и ввести подпись «**Оплачено**».*

3. *Щелчком правой кнопки мыши по флажку вызвать контекстное меню и выполнить в нем команду **Свойства**. На вкладке «**Данные**» диалогового окна свойств флажка выбрать строку «**Данные**» и в списке выбрать поле «**Оплачен**» (таким образом значение флажка связывается в полем таблицы «**Заказы**»). После ввода этой информации диалоговое окно нужно закрыть.*

4. *Закройте **Конструктор** и подтвердите сохранение внесенных в форму изменений.*

5. *Откройте форму Заказы для просмотра (рис.).*



Для отображения полной информации о заказчике и всех его заказах нужно создать реляционную форму, связывающую информацию из нескольких таблиц.

Задание*: Создать реляционную форму путем внедрения формы «**Заказы**» в*

*форму «**Данные о клиентах**» Для этого:*

1. *Выделите форму «**Данные о клиентах**» и щелкните кнопку **Конструктор**; перетяните значок формы «**Заказы**» в нижнюю часть формы «**Данные о клиентах**» (копия формы «**Клиенты**») на свободное место;*
2. *Отформатируйте элементы формы так, чтобы все они размещались на ней, и закройте конструктор.*

Полученная форма должна иметь вид, показанный на рисунке.

***Задание 13.** Введите с помощью формы **Данные о клиентах** данные о нескольких заказчиках и «оформите» заказы для них, используя ранее введенную информацию из таблиц «**Услуги**» и «**Исполнители**».*

Формы позволяют последовательно просматривать записи БД, искать нужные записи, удалять записи при необходимости. Для отбора информации по более сложным критериям, анализа и подведения итогов в БД создаются запросы и отчеты. Рассмотрим порядок их создания.

Вопросы для самоконтроля:

1. Как в форму вставить текущую дату или формулу?
2. Как вставить в форму раскрывающийся список и флажок?
3. Как создать реляционную форму из нескольких форм?

Практическое занятие №19
Создание отчетов

Цель Освоение приемов работы с Microsoft Access, создание отчетов.

Отчёты служат для форматированного вывода данных на печатающее устройство.

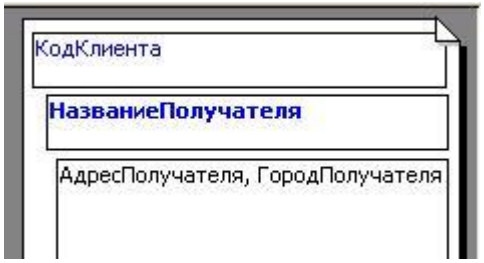
Структура готового отчёта отличается от структуры формы только увеличенным количеством разделов. Кроме разделов заголовка, примечания и данных, отчёт может содержать разделы верхнего и нижнего колонтитулов. Если отчёт занимает более одной страницы, эти разделы необходимы для печати служебной информации, например, номеров страниц.

Для создания сложного отчета, содержащего данные из нескольких таблиц, существует два способа:

- первый – непосредственно без создания запроса;
- второй – создать предварительно запрос и по нему создать отчет.

Создать отчет можно с помощью конструктора или с помощью мастера. Мастер отчётов работает в пять этапов:

1. выбор таблицы или запросов, на которых отчёт базируется; выбор полей, отражаемых в отчёте;
2. выбор полей группировки (уровней и интервалов группировки);



3. выбор полей и методов сортировки;
4. выбор структуры отчёта печатного макета (блочный, ступенчатый, выровненный по левому краю и т.п.)
5. на последнем этапе выполняется сохранение отчёта под заданным именем.

Создание отчета

Сформируйте отчет для приемной комиссии о результатах вступительных экзаменов, используя мастер отчетов.

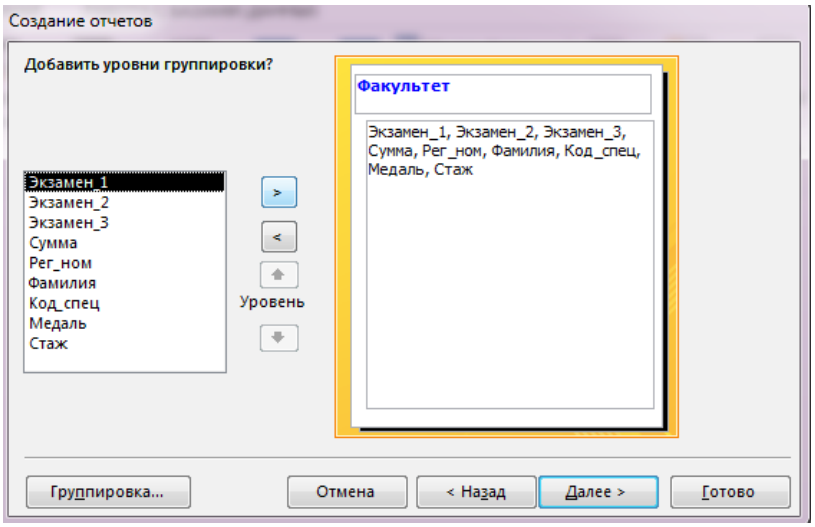
Создадим отчет **первым способом,** непосредственно без создания запроса.

- На вкладке **Создание** в группе **Отчеты**выполните команду 

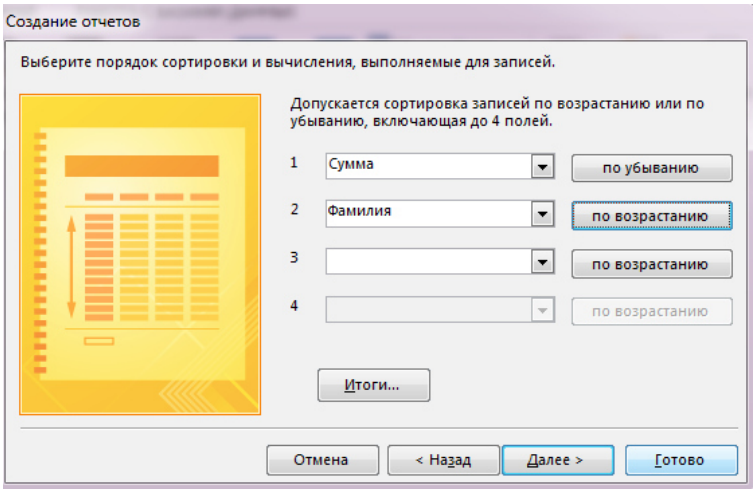
.
- Выберите из таблицы
  - *Факультеты* поля Факультет, Экзамен_1, Экзамен_2, Экзамен_3,
  - из *Запроса IX*поле Сумма,
  - из таблицы *Абитуриенты*– Рег_ном, Фамилия, Код_спец, Медаль, Стаж,

щелкните на кнопке **Далее**.

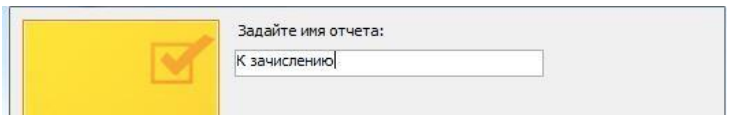
- Задать уровень группировки по полю Факультет (перенести поле Факультет в правую часть), щелкнуть на кнопке **Далее**.



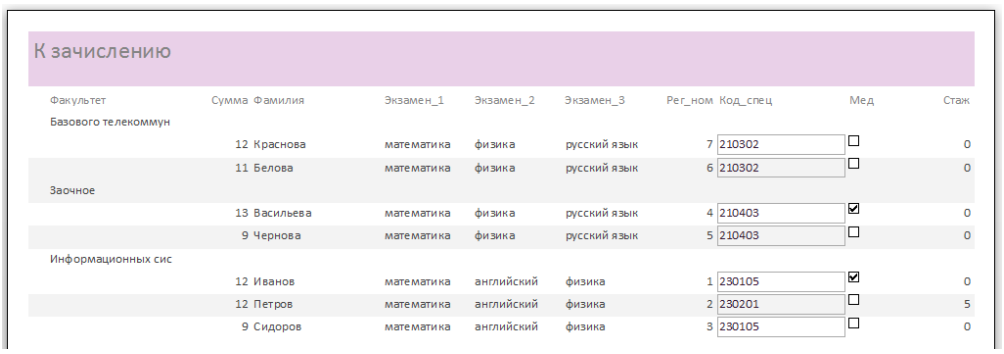
- Указать порядок сортировки 1). Сумма по убыванию; 2). Фамилия по возрастанию, щелкнуть кнопку **Далее**.



- Выбрать вид макета **Ступенчатый**, ориентация**Альбомная**, включить**Настроить ширину полей для размещения на одной странице**.**Далее**.
- Указать имя отчета**К зачислению**, щелкнуть на кнопке**Готово**.



- Результат отчета будет иметь следующий вид



- Используя режим Конструктора и режим Макета, приведите отчет к следующему виду:

К зачислению

Факультет	Экзамен_1	Экзамен_2	Экзамен_3	Сумма	Фамилия	Рег_ном	Код_спец	Медаль	Стаж
Базового телекоммуникационного образования	математика	физика	русский язык						
				12	Краснова	7	210302	☐	0
				11	Белова	6	210302	☐	0
Заочное	математика	физика	русский язык						
				13	Васильева	4	210403	☑	0
				9	Чернова	5	210403	☐	0
Информационных систем и технологий	математика	английский	физика						
				12	Иванов	1	230105	☑	0
				12	Петров	2	230201	☐	5
				9	Сидоров	3	230105	☐	0

Контрольные вопросы

1. Назначение отчетов.
2. Виды отчетов в БД MS Access.
3. Способы создания отчетов.
4. Исходные данные для отчета.
5. Ввод формул в отчет.

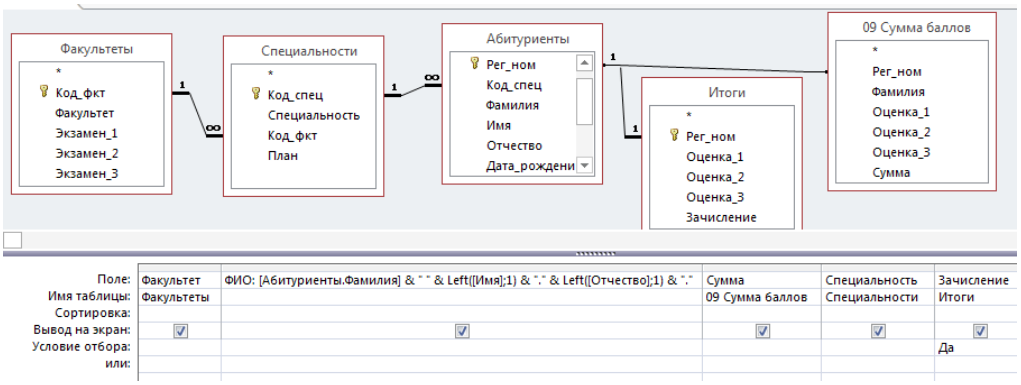
Практическое занятие №20
Создание отчетов в режиме Конструктор

Создайте отчет, содержащий список зачисленных в студенты. Списки должны быть сгруппированы по факультетам. В списке должны отражаться фамилия и инициалы, количество баллов, название специальности. Вычислите общее количество поступивших абитуриентов.

1. Создадим вспомогательный запрос. В этом запросе

- соберем все необходимую информацию;
- выберем только зачисленных по полю Зачисление из таблицы Итоги;
- из имени и отчества создадим инициалы, используя формулу (просто скопируйте формулу, см. рисунок)

ФИО: [Абитуриенты.Фамилия] & " " & Left([Имя];1) & "." & Left([Отчество];1) & "."



2. Но основе созданного запроса создайте отчет.

3. Вычислите общее количество поступивших абитуриентов.

Список зачисленных

Факультет	Специальность	ФИО	Сумма баллов
Базового телекоммуникационного образования			
	Радиотехника		
		Белова Л.Щ.	11
		Краснова К.З.	12
Заочное			
	Защищенные системы связи		
		Васильева О.Н.	13
		Смирнов А.Д.	14
Информационных систем и технологий			
	Информационные системы и технологии		
		Петров П.П.	12
	Программное обеспечение		
		Иванов И.И.	12

Всего зачислено: 6

Дополнительное задание. Посчитайте, сколько абитуриентов поступило на каждый факультет.

Практическое занятие №21
Работа с макросами

Макросы великолепно подходят для автоматизации повторяющихся действий. С помощью макросов Вы можете решить большинство своих проблем. Если макросов станет недостаточно, нужно обратиться к интегрированному в MS Access языку программирования Visual Basic. Данный язык является наиболее доступным языком программирования в мире, с другой стороны, Visual Basic обеспечивает очень высокую производительность.

Макросы встречаются гораздо реже, чем программы. Как правило, они очень тесно связаны с объектами среды MS Access. Практически каждый управляющий элемент располагает множеством так называемых «реакций на события». События определяют, что должно произойти при выполнении нажатия кнопки, перехода в поле и выходе из него.

Если в окне свойств активизировать одно из событий, MS Access выведет на экран список всех сохраненных в базе данных макросов. Пользователю остается только выбрать один из элементов списка, чтобы связать свойство элемента управления с макросом.

Построитель макросов

Прежде всего, следует открыть в режиме**«Конструктора»** форму, в которой находится соответствующий объект (например, кнопка). Поместите указатель мыши на объект (в данном случае на кнопку) и нажмите правую кнопку мыши. Объект будет выбран, и откроется контекстное меню,

из которого следует выбрать элемент **«Свойства»**. В окне свойств щелкните по вкладке **«События»**. Выберите то событие, в результате которого будет вызываться макрокоманда. В той же строке выполните щелчок мышью на кнопке вызова построителя, которая расположена рядом со строкой ввода значения свойства.

Появляется диалоговое окно **«Построитель»** (рисунок 5.1), в котором для запуска построителя макросов необходимо выбрать **«Макросы»** и нажать **«ОК»**. MS Access создает новый макрос и предлагает пользователю ввести его имя. После этого указанное имя макроса будет автоматически внесено в строку ввода свойства события. Одновременно макрос открывается в режиме **«Конструктора»**, после чего можно непосредственно приступать к его созданию. Каждый макрос может содержать одну или несколько команд. Макрокоманды могут запускаться с использованием условий или строго последовательно. После завершения работы над макросом его следует сохранить и закрыть окно макроса. Теперь можно снова нажать кнопку вызова построителя. Однако MS Access не создает нового макроса, а открывает заданный, непосредственно готовый к редактированию в режиме **«Конструктора»**.

Автоматический запуск макроса

Как правило, при открытии базы данных постоянно выполняются одни и те же действия. Типичным примером этого является открытие формы, содержащая панель управления прикладной программой. Было бы удобно, если бы MS Access выполнял это действие автоматически. Все, что для этого нужно – это автоматически выполняемый макрос, который называется **«AutoExec»**.

В окне базы данных выберите вкладку**«Макросы»**. Затем нажмите кнопку **«Создать»**, в результате чего MS Access откроет пустое окно макроса в режиме **«Конструктора»**. Нажмите мышью самую верхнюю строку столбца **«Макрокоманда»**. MS Access отображает на экране список макрокоманд. Осуществите прокрутку списка и выберите элемент **«Свернуть»**, чтобы при запуске макроса окно базы данных автоматически было свернуто и помещено в левый нижний угол экрана.

Далее выберите команду **«Открыть Форму»**. Теперь необходимо определить аргументы макрокоманды. Нажмите мышью строку **«Имя форм»**в нижней части окна. Здесь MS Access отображает список всех форм базы данных. Выберите ту форму, которая должна

открываться автоматически (например, **«Кнопочная форма»**). Для всех других аргументов можно оставить стандартные установки без изменений (рисунок 5.2).

Сохраните новый макрос под именем **«AutoExec»**. При следующем открытии базы данных автоматически будет открываться форма **«Кнопочная форма»**.

Если в исключительном случае потребуется подавить выполнение **«AutoExec»**-макроса, при открытии базы данных следует удержать нажатой клавишу SHIFT.

В более серьезных случаях в макрос **«AutoExec»** помещают команды подготовки приложения к работе, например, сюда можно поместить команды подсоединения таблиц из другой БД.

Макросы в качестве заменителей команд меню

Взаимодействие с формой можно упростить, разместив наиболее часто используемые команды меню в виде кнопок. Нажатие кнопки выполняется, как правило, гораздо быстрее, чем выбор команды меню. Это оправдывает себя в том случае, если часто приходится возвращаться в окно базы данных.

MS Access позволяет создавать макрос, в котором можно объединить несколько макрокоманд. Создание такой группы рекомендуется в тех случаях, когда для формы нужно несколько макрокоманд.

При использовании группы макрокоманд можно избежать того, что в окне базы данных появляется каждая отдельная макрокоманда. Это значительно улучшает наглядность.

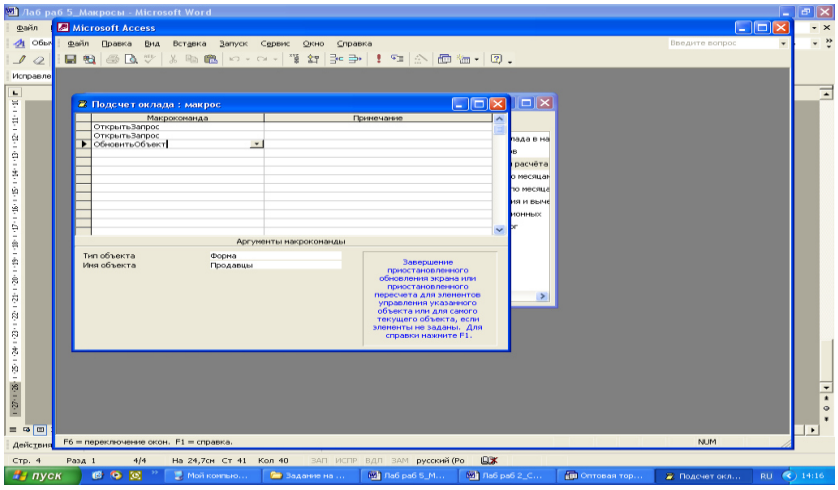
Определение группы макрокоманд выполняется точно так же, как определение отдельной макрокоманды.

Напишите макрос начисления заработной платы продавцам, в котором выполните следующую последовательность действий:

Создайте новый макрос. Для этого выберите «Макросы» в списке «Объекты» и нажмите на кнопку «Создать». Откроется окно Конструктора. В столбец «Макрокоманда» необходимо ввести все действия, которые должны выполняться при запуске макроса. В данном случае нам необходимо автоматизировать начисление заработной платы. За это действие отвечает запрос **«09\1_Подсчёт оклада»**, но сначала должен выполниться запрос **«08_Стаж продавцов»**, так как без определения стажа начисление оклада не осуществится. Поэтому в столбце «Макрокоманда» в поле со списком необходимо выбрать пункт «ОткрытьЗапрос». Имя запроса - 08_Стаж продавцов; Режим – Таблица; Режим данных – Изменение.

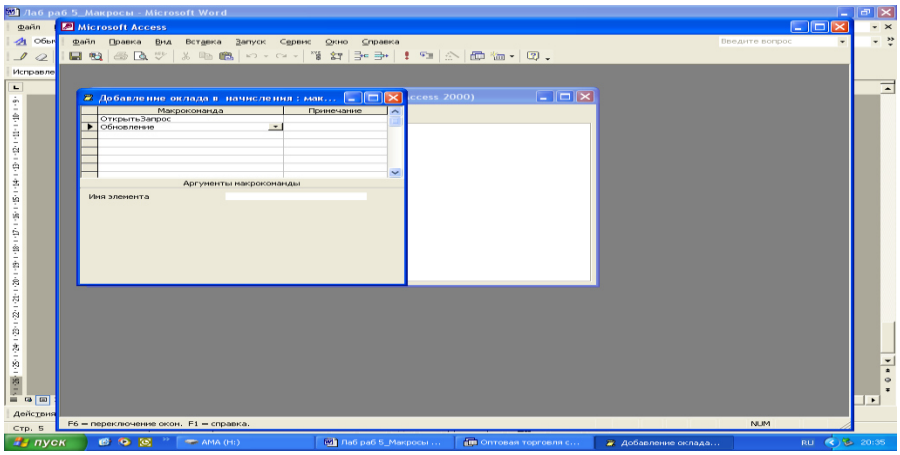
Теперь необходимо включить в макрос запрос 09\1_Подсчёт оклада. Вся последовательность действий аналогична предыдущим.

На этом создание макроса можно было бы считать завершенным, если бы не один нюанс. Так как этот макрос будет закреплен за кнопкой на форме, то следует обновить форму, чтобы увидеть изменения, которые произошли в результате выполнения запросов. Для этого в макрос следует ввести еще одно действие «Обновить объект».



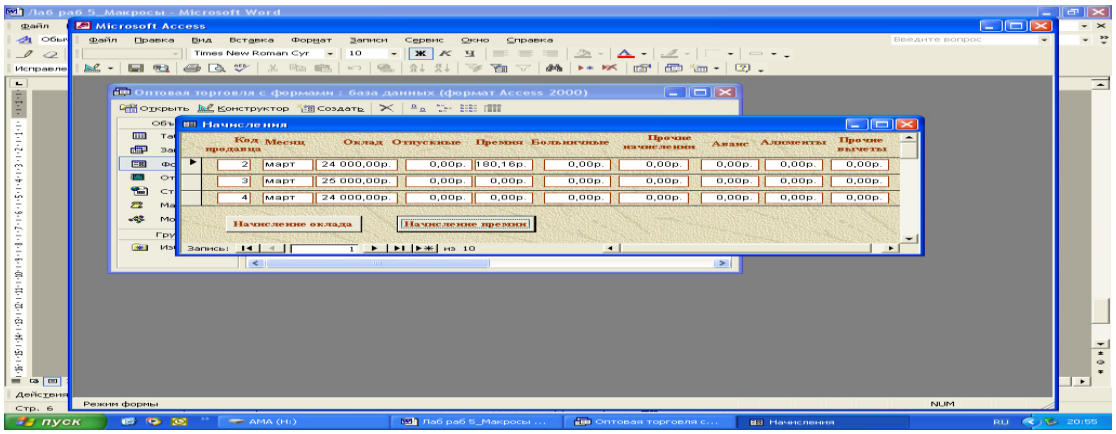
Теперь следует открыть форму «Продавцы» в режиме Конструктора, создать новую кнопку, за которой закрепить действие – «Выполнение макроса» (оно находится в категории «Разное»). Теперь можно проверить работу данной кнопки.

2.1 Напишите макрос, который позволял бы добавлять оклад продавца в начисления. Для этого необходимо создать новый макрос на основе запроса **09\2_Добавление оклада в начислениях**. И не забыть обновить объект. Закрепите данный макрос за кнопкой на форме «Начисления»



2.2 Напишите макрос «Начисление премии». Постарайтесь самостоятельно создать данный макрос.(Здесь нужно использовать два запроса **12\1_Запись суммы по месяцам 12\2_** и **Запись премии по**

месяцам). Закрепите данный макрос за кнопкой на форме «Начисления».



3.1 Создайте макрос для очистки формы «Месяц расчет» (Используйте запрос **11_Очистка таблицы расчёта**).

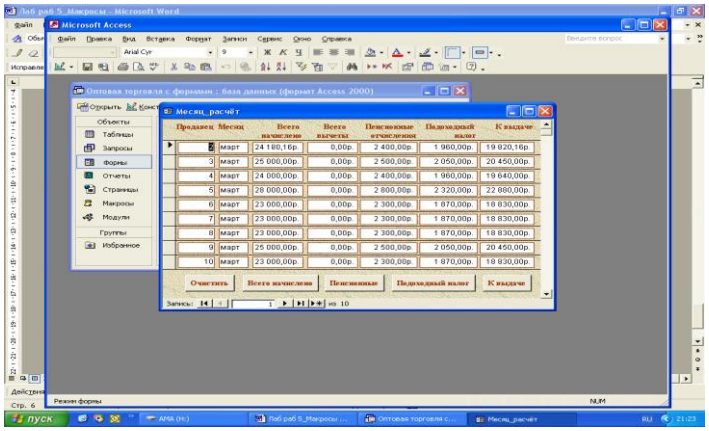
3.2 Создайте макрос «Всего начислено» на основе запроса **12\3_всего начисления и вычеты.**

3.3 Создайте макрос «Пенсионные», используя запрос **13_Обновление пенсионных.**

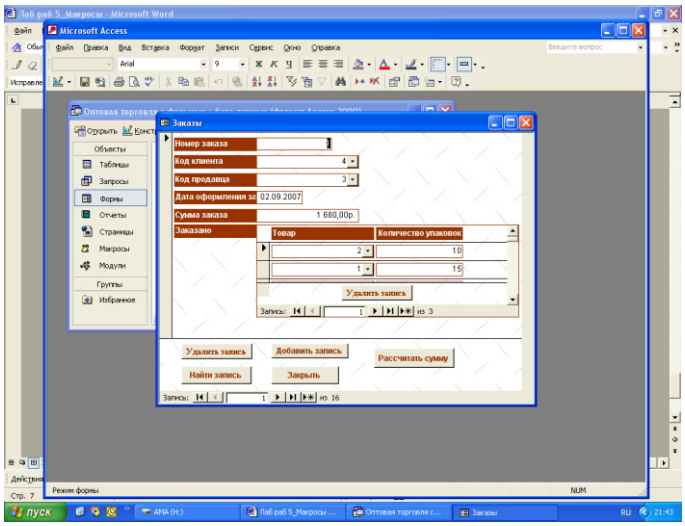
3.4 Создайте макрос «Подоходный налог». (Используйте запрос **14_Подоходный налог**).

3.5 Создайте макрос «К выдаче» на основе запроса **15_К выдаче.**

После написания макросов, создайте на форме «Месяц расчет» кнопки и закрепите за каждой определенный макрос. Образец готовой формы представлен на рисунке 5.6



4.1 Создайте макрос, который бы позволял рассчитывать сумму нового заказа. Для этого макроса необходимо использовать два запроса: **04\1_Сумма заказанных товаров** и **04\2_Обновление сумм заказов.** Так же в столбце «Макрокоманда» необходимо установить действие «ОбновитьОбъект» (Тип объекта – Форма, Имя объекта - Заказы). На форме «Заказы» необходимо создать кнопку «Рассчитать сумму» и закрепить за ней данный макрос



Практическое занятие №22
Создание запросов SQL.

Цель: Ознакомиться с основными операторами подъязыка DML SQL СУБД Access, научиться создавать SQL-запросы и преобразовывать QBE-запросы в SQL.

Аббревиатура SQL означает Structured Query Language (структурированный

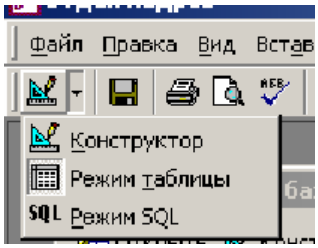
язык запросов). Синтаксис SQL разрабатывался для удобства формирования запросов «близко к естественному английскому». Предполагалось, что его смогут использовать рядовые пользователи баз данных.

В Access при обращении к БД также применяется язык SQL (Access изнутри

организован по системе «клиент-сервер»). Любой запрос, построенный с помощью мастера или конструктора, имеет соответствующее представление на языке SQL. Конструктор – лишь визуальное средство для создания запросов. В Access имеется возможность редактировать запросы непосредственно в режиме SQL. Причем, не всякий составленный на SQL запрос, может быть отображен в режиме конструктора – SQL имеет более широкие возможности, чем визуальный конструктор.

Для переключения режимов отображения запросов используется кнопка

«**Вид**» панели инструментов.



Чаще всего возникает задача построения запросов на извлечение данных.

Для этих целей используется SQL-оператор SELECT. С него мы и начнем изучение языка SQL.

Задание

1. Выдать таблицу студентов (преподавателей), задавая следующий порядок следования полей **ФИО, Индекс, Город, Адрес.**
2. Выдать **Наименование** всех дисциплин, проводимых определенным преподавателем.
3. Выдать список студентов из **Горно-Алтайска**.
4. Составить список преподавателей, упорядоченных по **факультету**, а в пределах факультета по **Наименованию** дисциплин.
5. Составить **объединенный** список преподавателей и студентов.
6. Выдать список студентов, **№ зачетной книжки** которых начинается с 1(или любого определенного символа в зависимости от ваших данных).
7. Определить общее количество студентов (преподавателей), нагрузку преподавателя (часы), средний балл студента.
8. Увеличьте стипендию студентов (оклад преподавателей) на 10 %.
9. Удалите сведения об определенном студенте (преподавателе).
10. Вставьте запись о новом студенте (преподавателе).

Основные сведения

С помощью языка структурированных запросов SQL, реализованного в

Access 2000, можно составить любое число сложных запросов. Этот язык позволяет также управлять обработкой запросов. SQL-запрос представляет собой последовательность инструкций, в которую могут входить выражения и статистические функции SQL.

В основе большинства sql-запросов лежит инструкция select. По этой инструкции ядро базы данных Microsoft Jet возвращает данные из базы данных в виде набора записей.

SELECT [*предикат*] { * |*таблица*.* | [*таблица*.]*поле_1*[AS*псевдоним_1*] [, [*таблица*.]*поле_2*[AS*псевдоним_2*] [, ...]]} FROM*выражение*[, ...] [IN*внешняяБазаДанных*] [WHERE...][GROUP BY...] [HAVING...] [ORDER BY...] [WITH OWNERACCESS OPTION]

Ниже перечислены аргументы инструкции SELECT:

Элемент	**Описание**
предикат	Один из следующих предикатов отбора: ALL, DISTINCT, DISTINCTROW или TOP. Предикаты используются для ограничения числа возвращаемых записей. Если они отсутствуют, по умолчанию используется предикат ALL.
*	Указывает, что выбраны все поля заданной таблицы или таблиц.
таблица	Имя таблицы, из которой должны быть отобраны записи.
поле_1,*поле_2*	Имена полей, из которых должны быть отобраны данные. Если включить несколько полей, они будут извлекаться в указанном порядке.
псевдоним_1,*псевдоним_2*	Имена, которые станут заголовками столбцов вместо исходных названий столбцов в *таблице*.
выражение	Имена одной или нескольких таблиц, которые содержат отбираемые данные.
внешняяБазаДанных	Имя базы данных, которая содержит таблицы, указанные с помощью аргумента *выражение*, если они не находятся в текущей базе данных.

При выполнении этой операции ядро базы данных Microsoft® Jet находит указанную таблицу или таблицы, извлекает заданные столбцы, выделяет строки, соответствующие условию отбора, и сортирует или группирует результирующие строки в указанном порядке.

Инструкции SELECT не изменяют данные в базе данных.

Обычно слово SELECT является первым словом инструкции SQL. Большая часть инструкций SQL является инструкциями SELECT или SELECT...INTO.

Ниже приведен минимальный синтаксис инструкции SELECT:

SELECT *поля* FROM *таблица*

Для отбора всех полей таблицы можно использовать символ звездочки (*). Следующая инструкция отбирает все поля из таблицы «Клиенты»:

SELECT * FROM Клиенты;

Если несколько таблиц, включенных в предложение FROM, содержат одноименные поля, перед именем такого поля следует ввести имя таблицы и оператор **.**(точка). Предположим, что поле «КодКлиента» содержится в таблицах «Клиенты» и «Заказы». Следующая инструкция SQL отберет поле «КодКлиента» из таблицы «Заказы» и поле «Название» из таблицы «Клиенты»:

SELECT Заказы.КодКлиента, Клиенты.Название
FROM Заказы INNER JOIN Клиенты
WHERE Заказы.КодКлиента=Клиенты.КодКлиента;

При создании объекта **Recordset**ядро базы данных Microsoft Jet использует имя поля таблицы в качестве имени объекта**Field**в объекте**Recordset**. Если требуется другое имя поля, или выражение, создающее поле, не определяет имя, используйте зарезервированное слово AS. В следующем примере заголовок «Товар» становится именем объекта**Field**в результирующем объекте**Recordset**:

SELECT Наименование
AS Товар FROM Сотрудники;

При работе со статистическими функциями или запросами, которые возвращают повторяющиеся имена объекта **Field**, используйте предложение AS для задания другого имени объекта**Field**. В следующем примере заголовок «Численность» задается для возвращаемого объекта**Field**результирующего объекта**Recordset**:

SELECT COUNT(КодКлиента)
AS Численность FROM Клиенты;

Для дальнейшего отбора и организации искомых данных в инструкцию SELECT можно добавлять многие другие предложения.

Спецификатор **WHERE** добавляется для выборки определенных строк или указания условия соединения. Спецификатор **ORDER BY** добавляется для выполнения сортировки данных выходного набора в заданной последовательности. Сортировка может осуществляться по нескольким полям, которые перечисляются

через запятую после ключевого слова **ORDER BY**. Спецификатор **GROUP BY** (спецификатор группировки) объединяет все записи, содержащие в заданном поле идентичные значения, в один элемент выходного набора. Спецификатор **HAVING** (спецификатор условия выборки группы) позволяет выполнять более сложные выборки данных. Спецификатор **IN**используется при работе с базами данных другого формата, с которыми может работать Access 2000 (например, dBase или Paradox), а также для отбора данных из неактивной базы Access 2000.

Предикат ALL задает включение в выходной набор всех дубликатов, отобранных по критерию WHERE. В команде SELECT предикат ALL следует сразу за ключевым словом SELECT. Предикат DISTINCT следует применять в тех случаях, когда необходимо исключить записи, которые содержат повторяющиеся данные в выбранных полях. Предикат DISTINCTROW используется для исключения дубликатов из всех полей, а также для исключения повторяющихся записей

Операция INNER JOIN

Объединяет записи из двух таблиц, если связующие поля этих таблиц содержат одинаковые значения.

SELECT *поля*FROM*таблица_1*INNER JOIN*таблица_2*ON*таблица_1.поле_1оператортаблица_2.поле_1*AND ON*таблица_1.поле_2оператортаблица_2.поле_2*) OR ON*таблица_1.поле_3оператортаблица_2.поле_3*)];

Операции JOIN могут быть вложенными; в таком случае используйте следующий синтаксис

SELECT *поля* FROM *таблица_1* INNER JOIN (*таблица_2* INNER JOIN [( ]*таблица_3* [INNER JOIN [( ]*таблица_X* [INNER JOIN ...)] ON*таблица_3.поле_3оператортаблица_X.поле_X*)] ON*таблица_2.поле_2оператортаблица_3.поле_3*) ON*таблица_1.поле_1оператортаблица_2.поле_2*;

Ниже перечислены аргументы операции INNER JOIN:

Элемент	**Описание**
таблица_1,*таблица_2*	Имена таблиц, записи которых подлежат объединению.
поле_1,*поле_2*	Имена объединяемых полей. Если эти поля не являются числовыми, то должны иметь одинаковый тип данных и содержать данные

	одного рода, однако поля могут иметь разные имена.
оператор	Любой оператор сравнения: "=", "<", ">", "<=", ">=" или "<>".

Операцию INNER JOIN можно использовать в любом предложении FROM. Это самые обычные типы связывания. Они объединяют записи двух таблиц, если связующие поля обеих таблиц содержат одинаковые значения.

Операцию INNER JOIN можно использовать с таблицами «Отделы» и «Сотрудники» для отбора всех сотрудников каждого отдела. Для отбора же всех отделов (в том числе тех, в которых нет ни одного сотрудника) или всех сотрудников (в том числе тех, кто не приписан ни к одному отделу) следует использовать операцию LEFT JOIN или RIGHT JOIN, которая создает внешнее объединение.

Попытка объединить поля Memoилиобъекта OLEприведет к возникновению ошибки.

Допускается объединение любых двух числовых полей подобных типов. Например, поле счетчикаможно объединить с полемтипа «Длинное целое».Однако нельзя объединить типы полейSingleиDouble.

Следующая инструкция SQL объединяет таблицы «Типы» и «Товары» по полю «КодТипа»:

```
SELECT Тип,Наименование
FROM Типы INNER JOIN Товары
ON Типы.КодТипа = Товары.КодТипа;
```

В предыдущем примере поле «КодТипа» используется для объединения таблиц, однако оно не включается в результат выполнения запроса, поскольку не включено в инструкцию SELECT. Чтобы включить связующее поле (в данном случае поле Типы.КодТипа) в результат выполнения запроса, следует включить имя этого поля в инструкцию SELECT.

:Операции LEFT JOIN или RIGHT JOIN могут быть вложены в операцию INNER JOIN, но операция INNER JOIN не может быть вложена в LEFT JOIN или RIGHT JOIN.

Подчиненные запросы sql

Подчиненным запросом называют инструкцию SELECT, вложенную в инструкцию SELECT, SELECT...INTO, INSERT...INTO, DELETE или UPDATE или в другой подчиненный запрос.

Синтаксис

Подчиненный запрос создается одним из трех способов:

сравнение[ANY | ALL | SOME] (*инструкцияSQL*)
выражение[NOT] IN (*инструкцияSQL*)
[NOT] EXISTS (*инструкцияSQL*)
Ниже перечислены аргументы подчиненного запроса:

Элемент	Описание
сравнение	Выражение и оператор сравнения, который сравнивает выражение с результатами подчиненного запроса.
выражение	Выражение, для которого проводится поиск в результирующем наборе записей подчиненного запроса.
инструкцияSQL	Инструкция SELECT, которая соответствует формату и всем правилам, принятым для инструкций SELECT. Она должна быть заключена в круглые скобки.

Дополнительные сведения

Подчиненный запрос можно использовать вместо выражения в списке полей инструкции SELECT или в предложениях WHERE и HAVING. Инструкция SELECT используется в подчиненном запросе для задания набора конкретных значений, вычисляемых в выражениях предложений WHERE или HAVING.

Предикаты ANY или SOME, являющиеся синонимами, используются для отбора записей в главном запросе, которые удовлетворяют сравнению со всеми записями, отобранными в подчиненном запросе. В следующем примере отбираются все товары, цена которых больше, чем цена любого товара, проданного со скидкой в 25 процентов или более:

```
SELECT * FROM Товары
WHERE Цена > ANY
(SELECT Цена FROM Заказано
WHERE Скидка >= .25);
```

Предикат ALL используется для отбора в главном запросе только тех записей, которые удовлетворяют сравнению со всеми записями, отобранными в подчиненном запросе. Если в предыдущем примере предикат ANY заменить предикатом ALL, результат запроса будет включать только те товары, чья цена больше, чем цена всех товаров, проданных со скидкой 25 или более. Это условие является значительно более жестким.

Предикат IN используется для отбора в главном запросе только тех записей, которые содержат значения, совпадающие с одним из

отобранных подчиненным запросом. Следующий пример возвращает все товары, проданные со скидкой, большей или равной 25 процентам:

```
SELECT * FROM Товары
WHERE КодТовара IN
(SELECT КодТовара FROM Заказано
WHERE Скидка >= .25);
```

И наоборот, предикат NOT IN используется для отбора в главном запросе только тех записей, которые содержат значения, не совпадающие ни с одним из отобранных подчиненным запросом.

Предикат EXISTS (с необязательным зарезервированным словом NOT) используется в логическом выражении для определения того, должен ли подчиненный запрос возвращать какие-либо записи.

В подчиненном запросе можно использовать псевдонимытаблиц для ссылки на таблицы, перечисленные в предложении FROM, расположенном вне подчиненного запроса. В следующем примере отбираются фамилии и имена сотрудников, чья зарплата равна или больше средней зарплаты сотрудников, имеющих ту же должность. В данном примере таблица «Сотрудники» получает псевдоним «T1»:

```
SELECT Фамилия,
Имя, Должность, Оклад
FROM Сотрудники AS T1
WHERE Оклад >=
(SELECT Avg(Оклад)
FROM Сотрудники
WHERE T1.Должность = Сотрудники.Должность) Order by Должность;
```

В последнем примере зарезервированное словоне является обязательным.

Некоторые подчиненные запросы можно использовать в перекрестных запросахкак предикаты (в предложении WHERE). Подчиненные запросы, используемые для вывода результатов (в списке SELECT), нельзя использовать в перекрестных запросах.

Практическое занятие №23
Запросы SQL на объединение.

Чаще всего возникает задача построения запросов на извлечение данных.

Для этих целей используется SQL-оператор SELECT. С него мы и начнем изучение языка SQL.

Простейшая форма оператора SELECT

SELECT <список полей>

FROM <список источников>**;**

Список полей – имена полей (столбцов), которые следует извлечь из источников (таблиц) и поместить в результирующий набор.

Пример 1.1.

SELECT Фамилия, Имя, Отчество

FROM Клиент**;**

Смысл этого запроса таков: извлечь поля «Фамилия», «Имя», «Отчество» из таблицы «Клиент».

Если следует ограничить результирующее множество комбинациями только тех записей двух таблиц, которые связаны между собой (содержат одинаковые значения в полях «**КодКлиента**»).

В предложении «**FROM**» вместо оператора объединения «**,**» (запятая) будем использовать оператор «**INNER JOIN**» - внутреннее объединение. Оператор «**INNER JOIN**» позволяет наложить ограничение на объединяемые записи.

Предложение **ON <условие>** определяет связь между полями объединяемых

таблиц.

SELECT Клиент.Фамилия, Подписка.КодЖурнала

FROM Клиент **INNER JOIN** Подписка

ON Клиент.КодКлиента = Подписка.КодКлиента;

Результат содержит только комбинации записей, для которых выполняется

заданное условие.

Рассмотрим подробнее разные виды объединения.

• **INNER JOIN** (внутреннее объединение). В результат включаются только

те записи из обеих таблиц, которые связаны между собой.

• **LEFT JOIN** (левое внешнее объединение). В результат включаются все записи из первой таблицы. Если для них нет связанных записей во второй таблице, соответствующие поля результата будут пустыми.

• **RIGHT JOIN** (правое внешнее объединение). Операция, зеркально симметричная левому объединению. Включаются все записи из второй таблицы и связанные с ними записи из первой.

Группировка, сортировка, имена столбцов.

Пример. Изменим запрос таким образом, чтобы фамилия каждого клиента выводилась только один раз, и для него отображалось

общее количество выписанных журналов, а не коды каждого из журналов.

Зададим группировку записей по фамилии (фамилии не должны повторяться) и для групп определим групповую операцию «подсчет количества».

SELECT Клиент.Фамилия, **Count**(Подписка.КодЖурнала)
FROM Клиент **LEFT JOIN** Подписка
ON Клиент.КодКлиента = Подписка.КодКлиента
GROUP BY Клиент.Фамилия;

В предложении **GROUP BY** могут быть перечислены несколько полей через запятую. Если группировка производится по нескольким полям, объединяться в группу будут строки, для которых попарно равны значения всех группируемых полей. Для всех полей, перечисленных в предложении **SELECT**, но не вошедших в предложение **GROUP BY**, должны быть определены групповые операции.

Практическое занятие №24

Создание запросов SQL на изменение.

Пример. Название второго столбца (в примере выше) сформировано автоматически. Надо задать осмысленное название. Кроме того, является лишь совпадением то, что фамилии расположены по алфавиту. Следует явно указать способ сортировки результирующего набора. Внесем соответствующие изменения в запрос.

SELECT Клиент.Фамилия, **Count**(Подписка.КодЖурнала) **AS** К оличество
FROM Клиент **LEFT JOIN** Подписка
ON Клиент.КодКлиента = Подписка.КодКлиента
GROUP BY Клиент.Фамилия
ORDER BY Клиент.Фамилия **ASC**;

Ключевое слово **AS** указывает имя (псевдоним) для столбца. Если имя не задано явно, используется соответствующее ему имя поля исходной таблицы.

Если же столбец формируется с помощью некоторого выражения, Access присваивает ему имя самостоятельно.

Псевдонимы можно задавать не только для столбцов, но и для источников

данных (таблиц), указанных в предложении **FROM**. Тогда к ним можно обращаться внутри запроса по новому имени.

Параметры сортировки задаются в предложении **ORDER BY**. В предложении можно перечислять несколько полей через запятую. Сортировка будет выполняться сначала по первому полю, затем по второму (если совпадают значения первого поля) и т.п.

В предложении **ORDER BY** для каждого из столбцов может указываться

одно из ключевых слов, задающих направление сортировки – **ASC** (по возрастанию) или **DESC** (по убыванию). Если направление не указано, подразумевается значение **ASC**.

Ограничение результирующих наборов.

Для выполнения дальнейшего сужения в операторе **SELECT** могут присутствовать еще два вида предложений: **WHERE** и **HAVING**.

Предложение **WHERE <условие>** располагается после предложения

FROM и позволяет наложить дополнительные ограничения на результат объединения. Ограничения, заданные словом **ON** иногда могут быть перенесены также в предложение **WHERE**.

Предложение **HAVING <условие>** может располагаться после предложения **GROUP BY** и применяться к данным каждой сформированной группы.

При использовании предложения **HAVING** без предложения **GROUP BY**, оно применяется ко всей результирующей таблице и действует аналогично предложению **WHERE**.

Например.

WHERE Left(Клиент.Фамилия,1) = "И" **AND** Подписка.КодЖурнала > 1

GROUP BY Клиент.Фамилия

HAVING Count(Подписка.КодЖурнала) > 0

В предварительный результат (после предложения **WHERE**) входят записи

только для тех клиентов, фамилия которых начинается с «И» и при этом используются только журналы, коды которых больше «1». К сформированному набору применяется операция группировки. Уже после группировки исключаются те клиенты, у которых нет подписки (число выписанных журналов равно «0»), при подсчете количества не учитывается информация о подписке на журналы, исключенные ранее в предложении **WHERE**.

Синтаксисе SQL

Общая форма оператора SELECT

SELECT [**DISTINCT**] список_выражений|*
[**INTO** новая_таблица]
[**FROM** объединение_источников]
[**WHERE** условие]
[**GROUP BY** список_столбцов]
[**HAVING** условие]
[**UNION** [**ALL**] **SELECT** …]
[**ORDER BY** список_столбцов]

Необязательное ключевое слово **DISTINCT** отвечает за то, чтобы в результирующем наборе не было полностью совпадающих строк (записей). Повторяющиеся строки будут исключены.

Вложенные запросы в предложении WHERE. Вложенные запросы могут

быть использованы не только в качестве источников данных, но и в предложении **WHERE**, при определении ограничений результирующего набора.

Пример. Пусть существует таблица

Зарплата(Номер, ФИО, Оклад)

Запрос, выводящий ФИО и оклад работников, оклад которых выше средне-

го, может иметь следующий вид:

SELECT ФИО, Оклад
FROM Зарплата
WHERE Оклад >
(**SELECT AVG**(Оклад) **FROM** Зарплата)
ORDER BY ФИО;

Для того чтобы выполнить проверку условия «выше среднего», требуется

вычислить это самое среднее. Эту работу выполняет вложенный запрос, который обращается к той же самой таблице «Зарплата».

Вместе с операциями сравнения могут быть использованы операторы **ALL**

(все) и **ANY** (какой-нибудь). **ALL** требует, чтобы заданное условие выполнялось для всех записей из вложенного запроса, **ANY** - хотя бы для одной.

Для работы с вложенными запросами также существуют операторы **IN** и

NOT IN. Оператор **IN** требует, чтобы значение его левого аргумента содержалось в результирующем наборе правого аргумента

(вложенного запроса). Оператор **NOT IN** имеет противоположное назначение.

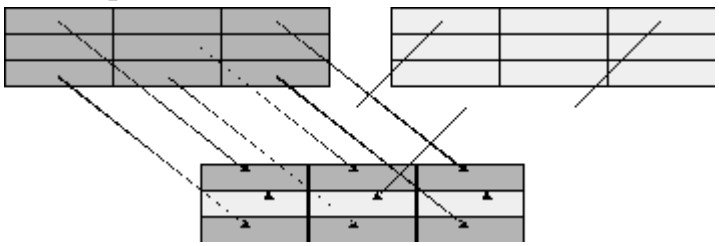
Пример. Список работников с окладом выше среднего:

```
SELECT ФИО, Оклад
FROM Зарплата
WHERE Оклад NOT IN (
SELECT Оклад FROM Зарплата
WHERE Оклад <=
(SELECT AVG(Оклад) FROM Зарплата)
)
ORDER BY ФИО;
```

Для обработки вложенных запросов могут использоваться операторы

EXISTS (существует) и **NOT EXISTS** (не существует). Оператор **EXISTS** возвращает значение «истина», если вложенный запрос содержит хотя бы одну запись (не важно, с какими данными). Оператор **NOT EXISTS** имеет противоположное назначение.

Предложение UNION … SELECT. Иногда может возникнуть необходимость объединить данные из нескольких источников, как показано на рис.



Для этой цели используется предложение **UNION** оператора **SELECT** (см.

выше общую форму оператора **SELECT**). В основном операторе **SELECT** выполняется выборка, формирующая первый результирующий набор, затем, после ключевого слова **UNION** помещается еще один оператор **SELECT**, формирующий второй набор. При выполнении такого запроса оба набора объединяются. Объединяться могут данные более чем двух источников. При этом предложение **UNION … SELECT** будет использовано несколько раз.

По умолчанию при слиянии двух наборов, Access объединяет строки, которые полностью совпадают. Если этого делать не нужно, следует использовать ключевое слово **ALL** (**… UNION ALL SELECT …**).

Предложение INTO … Использование в операторе **SELECT** конструкции

вида

SELECT <поля> **INTO** <новая таблица>

…

позволяет поместить результат запроса в таблицу, которая будет для этого создана.

Оператор INSERT INTO. Оператор добавляет новые записи в таблицу.

INSERT INTO имя_таблицы

[(имя_столбца[,имя_столбца …])]

VALUES (значение[, значение …])

Добавляет в таблицу строку, присваивая указанным полям перечисленные

значения.

INSERT INTO имя_таблицы

[(имя_столбца[,имя_столбца …])]

SELECT …

Добавляет в таблицу строки, сформированные оператором **SELECT**.

Оператор DELETE FROM

DELETE FROM имя_таблицы

[**WHERE** условие]

Оператор удаляет из указанной таблицы записи в соответствии с заданным

условием. При отсутствии предложения **WHERE** удаляются все записи.

Оператор UPDATE

UPDATE имя_таблицы

SET

имя_столбца = значение|(**SELECT**…)

[,имя_столбца = значение|(**SELECT**…)…]

[**WHERE** условие]

Устанавливает значения для указанных столбцов в строках таблицы, соответствующих условию отбора. Значение может быть получено как результат вложенного запроса.

Оператор CREATE TABLE. Оператор создает таблицу с заданным именем

и набором столбцов.

Упрощенная форма:

CREATE TABLE имя_таблицы (
имя_столбца **тип_данных**[**(размер)**]
[,имя_столбца **тип_данных** [**(размер)**] …]
)

Кроме имен и типов данных, при создании таблицы могут быть заданы различные ограничения. Различают **простые** и **составные** ограничения. Простые ограничения задаются в форме атрибутов конкретных столбцов, составные указываются отдельно и могут применяться к нескольким столбцам. Все ограничения могут быть заданы также визуальными средствами – в конструкторе таблиц или схеме данных.

Общая форма оператора:

CREATE TABLE имя_таблицы (
имя_столбца **тип**[**(размер)**] [атрибуты]
[,имя_столбца **тип**[**(размер)**] [атрибуты] …]
[,**PRIMARY KEY** (имя_столбца [, имя_столбца …])]
[,**FOREIGN KEY** (имя_столбца [, имя_столбца …])
REFERENCES имя_таблицы
[(имя_столбца [, имя_столбца])]
[**ON UPDATE CASCADE** | **SET NULL**]
[**ON DELETE CASCADE** | **SET NULL**]
]
)

Некоторые атрибуты, задающие простые ограничения для столбцов:

• **NULL** или **NOT NULL**

Соответственно разрешает или запрещает помещать пустые значения в

этот столбец.

• **UNIQUE**

Накладывает требование уникальности значений. Для любых двух строк

таблицы значения в этом столбце не могут быть одинаковыми.

• **PRIMARY KEY**

Объявляет столбец первичным ключом таблицы. Имеет тот же смысл, что

и задание ключевого поля в конструкторе таблиц.

• **REFERENCES** имя_таблицы [(имя_столбца)]
[**ON UPDATE CASCADE** | **SET NULL**]
[**ON DELETE CASCADE** | **SET NULL**]

Объявляет данный столбец внешним ключом, который связан с заданным

полем заданной таблицы. Поле, с которым устанавливается связь, должно

быть ключевым, или, по крайней мере, уникальным.

Необязательные атрибуты **ON UPDATE** и **ON DELETE** могут принимать

значения **CASCADE** или **SET NULL**.

Использование атрибутов **ON UPDATE** и **ON DELETE** равносильно заданию соответствующих свойств этой связи («Обеспечение целостности данных», «Каскадное обновление связанных полей», «Каскадное удаление связанных записей») в схеме данных.

Последовательность действий:

1. Запустите Microsoft Access и откройте базу данных.

2. Создайте запрос с именем «Самый простой запрос», который вычисляет

значение выражения «2+2» и выдает текущее время, не используя при этом

таблицы или другие запросы базы данных.

• Щелкните по ярлыку «**Создание запроса в режиме конструктора**».

• Закройте диалоговое окно «**Добавление таблиц**».

• Перейдите в режим SQL ().

• В окне редактирования введите текст запроса «**SELECT 2+2, NOW();**».

• Перейдите в режим просмотра результатов работы запроса  . Обратите

внимание, что названия столбцов сформированы автоматически.

• Перейдите в режим конструктора (). Обратите внимание, какие имена

присвоил конструктор столбцам запроса.

• Перейдите в режим SQL и назначьте столбцам новые имена: «**ДваПлюс-**

Два» и «**ДатаИВремя**» соответственно.

• Просмотрите результат работы запроса.

• Закройте окно запроса, сохранив его под именем «**Самый простой за-**

прос».

Практическое занятие №25
Виды запросов SQL.

Чтобы создать в режиме конструктора и модифицировать в режиме SQL запрос «Журналы с кодами клиентов», выводящий список журналов и коды клиентов, которые на них подписаны необходимо выполнить следующие действия.

• Запустите конструктор для создания нового запроса.

• Добавьте в запрос таблицы «**Журнал**» и «**Подписка**».

• Перенесите в бланк запроса имена полей «**Название**» и «**Клиент**».

• Просмотрите результат работы запроса.

Запрос выдает названия только тех журналов, на которые подписан хотя бы один клиент. Следует изменить способ объединения таблиц с внутреннего на левое внешнее.

*Обратите внимание, из таблицы «**Подписка**» извлекаются именно коды клиентов, а их фамилии отображаются в результирующей таблице благодаря тому, что для таблицы «**Подписка**» были настроены параметры подстановки.*

• Перейдите в режим SQL.

• Замените в предложении «**FROM**» ключевые слова «**INNER JOIN**» на

«**LEFT JOIN**».

• Просмотрите результат работы запроса.

*Теперь в столбце «**Название**» присутствуют названия и тех журналов, у*

которых нет ни одного подписчика. Для таких журналов в столбец

*«**Клиент**» помещено пустое значение.*

• Перейдите в режим конструктора. Обратите внимание на то, что в конструкторе способ объединения также изменился.

Линия связи, обозначающая объединение таблиц, теперь снабжена

стрелкой на одном конце.

• Для того, чтобы сохранить запрос, не закрывая окна конструктора, на-

жмите кнопку «**Сохранить**»  на панели инструментов Access. В диалоговом окне введите имя запроса «**Журналы с кодами клиентов**» и

нажмите кнопку «**OK**».

• Выясните, как изменится SQL – представление запроса, если связь между

таблицами будет отсутствовать. Нажмите правой кнопкой мыши на линии связи и в контекстном меню выберите пункт «**Удалить**».

В случае, когда связь между полями таблиц не определена, запрос формирует декартово произведение таблиц.

• Перейдите в режим SQL.

*Теперь в предложении «**FROM**» вместо оператора объединения «**LEFT***

***JOIN**» с условием равенства значений столбцов двух таблиц, используется оператор «**запятая**», объединяющий записи таблиц безусловно, в виде декартова произведения.*

• Закройте окно запроса, при этом **откажитесь от сохранения последних**

изменений.

4. Создайте в режиме SQL запрос «Журналы с количеством клиентов», выводящий список журналов и количество клиентов, которые подписаны на каждый из них.

• Запустите конструктор для создания нового запроса.

• Закройте окно добавления таблиц. Перейдите в режим SQL.

• Введите в окне редактирования запрос следующего вида:

SELECT Журнал.Название,
Count(Подписка.Журнал) **AS** Количество
FROM Журнал **LEFT JOIN**
Подписка **ON** Журнал.КодЖурнала = Подписка.Журнал
GROUP BY Журнал.Название
ORDER BY Журнал.Название;

• Просмотрите результат работы запроса.

• Просмотрите запрос в режиме конструктора. Найдите соответствие между всеми элементами, размещенными в бланке и области таблиц конструктора запроса, и элементами запроса на языке SQL.

• Закройте окно запроса, сохранив запрос под именем «**Журналы с количеством клиентов**».

5. В форме «Подписчики - подчиненная» в качестве источника записей используется специальный запрос «Клиенты с кодами журналов». Измените свойства формы таким образом, чтобы вместо внешнего запроса использовался запрос на языке SQL, заданный непосредственно внутри формы.

• Откройте форму «**Подписчики - подчиненная**» в режиме конструктора.

• Просмотрите свойство «**Источник записей**» данной формы.

*В качестве источника указан запрос «**Клиенты с кодами журналов**».*

• Откройте запрос «**Клиенты с кодами журналов**» в режиме SQL.

• Выделите и скопируйте в буфер обмена текст SQL – запроса.

• Вставьте скопированный текст запроса из буфера обмена в строку свойства «**Источник записей**».

• Просмотрите результат работы формы. Убедитесь, что ее поведение осталось прежним.

• Закройте форму «**Подписчики - подчиненная**», сохранив внесенные изменения. Закройте окно запроса.

• Откройте форму «**Список журналов**», которая использует измененную

выше форму в качестве подчиненной. Убедитесь, что форма работает верно.

*Теперь запрос «**Клиенты с кодами журналов**» больше не нужен. Источник записей формы образуется встроенным запросом на языке SQL.*

6. Создайте запрос с именем «**ДобавитьСтрану**». В режиме SQL введите

текст запроса на добавление данных в таблицу:

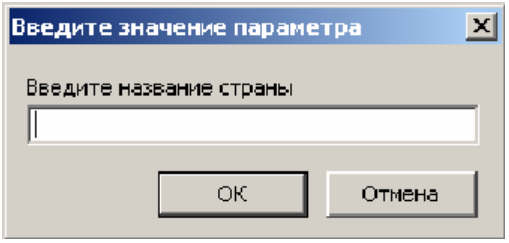
INSERT INTO Страна

(название) **VALUES** ([Введите название страны])

;

Этот запрос добавляет в таблицу «**Страна**» новую запись. Запрос содержит

параметр, поэтому при его запуске будет выдано окно для ввода дополнительной информации.



Основная литература:

1. Схиртладзе, А.Г. Проектирование единого информационного пространства виртуальных предприятий : учебник / А.Г. Схиртладзе, А.В. Скворцов, Д.А. Чмырь. - Изд. 2-е, стер. - Москва ; Берлин : Директ-Медиа, 2017. - 617 с. : ил., схем., табл. - Библиогр.: с. 606. - ISBN 978-5-4475-8634-8 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=469047

2. Сирант, О.В. Работа с базами данных / О.В. Сирант, Т.А. Коваленко. - 2-е изд., испр. - М. : Национальный Открытый Университет «ИНТУИТ», 2016. - 150 с. : схем., ил. - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=428978

3. Федорова Г.Н. Основы проектирования баз данных. –М.: ОИЦ «Академия» 2015.

Дополнительная литература:

1. Баженова, И.Ю. Основы проектирования приложений баз данных / И.Ю. Баженова. - 2-е изд., испр. - М. : Национальный Открытый Университет «ИНТУИТ», 2016. - 238 с. : ил. - Библиогр. в кн. - ISBN 5-94774-539-9 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=428933