Министерство науки и высшего образования
Российской Федерации

Федеральное государственное автономное образовательное
учреждение высшего образования
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий
и управления в технических системах

Кафедра информатики и управления в технических системах

ЛАБОРАТОРНЫЙ ПРАКТИКУМ ПО СИСТЕМЕ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ ACCESS

Методические указания
к выполнению цикла лабораторных работ
по дисциплине "Системы управления базами данных"
для студентов направления подготовки
27.03.04 – Управление в технических системах

Севастополь
СевГУ
2019

УДК 004.65
ББК 32.972.134
Л12

Рецензент:
В.А. Карапетьян – доцент кафедры "Информатика и управление в технических системах", канд. техн. наук, доцент

Составитель: Е.А. Шушляпин

Л12 Лабораторный практикум по системе управления базами данных ACCESS: метод. указания к выполнению цикла лабораторных работ по дисциплине «Системы управления базами данных» для студентов направления подготовки 27.03.04 Управление в технических системах / Сост. Е.А. Шушляпин. – Севастополь: СевГУ, 2019. – 92 с.

В методических указаниях приводится описание шести лабораторных работ, целью которых является изучение функциональных возможностей СУБД ACCESS и языка запросов SQL на примере учебной базы данных TUTORIAL.

Методические указания предназначены для студентов направления подготовки 27.03.04 Управление в технических системах.

УДК 004.65
ББК 32.972.134

Методические указания рассмотрены и утверждены на заседании кафедры "Информатика и управления в технических системах", протокол № 8 от 08 апреля 2019 г.

Методические указания рассмотрены и рекомендованы к изданию на заседании ученого совета Института информационных технологий и управления в технических системах, протокол № 5 от 25 апреля 2019 г.

© ФГАОУ ВО «Севастопольский государственный университет», 2019

СОДЕРЖАНИЕ

ЛАБОРАТОРНАЯ РАБОТА № 1. ОСНОВЫ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА И ФОРМИРОВАНИЕ ТАБЛИЦ 5
1. Цель работы 5
2. Объект исследования 5
3. Краткие теоретические сведения 5
4. Программа работ 15
5. Защита работы 18
ЛАБОРАТОРНАЯ РАБОТА №2. СОЗДАНИЕ ЗАПРОСОВ К БАЗЕ ДАННЫХ 19
1. Цель работы 19
2. Объект исследования 19
3. Краткие теоретические сведения 19
4. Программа работ 27
5. Защита работы 27
ЛАБОРАТОРНАЯ РАБОТА №3. РАЗРАБОТКА ЭКРАННЫХ ФОРМ 28
1. Цель работы 28
2. Объект исследования 28
3. Краткие теоретические сведения 28
4. Программа работ 37
5. Защита работы 38
ЛАБОРАТОРНАЯ РАБОТА №4. ФОРМИРОВАНИЕ ОТЧЕТОВ 39
1. Цель работы 39
2. Объект исследования 39
3. Краткие теоретические сведения 39
4. Программа работ 51
5. Защита работы 54
ЛАБОРАТОРНАЯ РАБОТА №5. ФОРМИРОВАНИЕ СХЕМ БАЗ ДАННЫХ 55
1. Цель работы 55

2. Объект исследования 55
3. Краткие теоретические сведения 55
4. Программа работ 63
5. Защита работы 67
Лабораторная работа №6. ЯЗЫК СТРУКТУРИРОВАННЫХ ЗАПРОСОВ SQL 69
Цель работы 69
Описание учебной базы данных 69
Создание SQL-запросов В СУБД MS Access 70
Краткие теоретические сведения 71
Реализация операций реляционной алгебры средствами языка SQL 81
Задание на работу 88
Упражнения 89
Порядок защиты работы и контрольные вопросы 90
Приложение А Основные различия языков Jet SQL и ANSI SQL 91
Библиографический список 92

ЛАБОРАТОРНАЯ РАБОТА № 1
ОСНОВЫ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА И ФОРМИРОВАНИЕ ТАБЛИЦ

1. ЦЕЛЬ РАБОТЫ

Цель работы – изучить и практически освоить пользовательский интерфейс системы управления базами данных (СУБД) настольного типа Access, научиться вводить и просматривать базы данных (БД) Access.

2. ОБЪЕКТ ИССЛЕДОВАНИЯ

Объектом исследования является СУБД Access фирмы Microsoft. Инструментом исследования является программный пакет Access.

3. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

3.1. Общие сведения о программном пакете Access

Пакет Access разработан фирмой Microsoft в виде составной части офисного пакета Microsoft Office, включающего тестовый процессор Word, электронную таблицу Excel, СУБД Access и ряд других программных средств. Версии офисного пакета соответствуют версиям операционных систем Windows фирмы Microsoft. Соответственно изменялись и версии СУБД Access. Традиционно, между версиями поддерживается совместимость “снизу – вверх”, когда базы данных, выполненных в более старой версии СУБД, могут эксплуатироваться в ее более новой версии. Последние версии Access, первоначально относящегося к СУБД «настольного» (desktop) типа, имеют некоторые характерные черты промышленных СУБД, в частности, поддерживают понятия «схемы», ключей и соответствующие средства проверки целостности баз данных (БД). Базы данных Access имеют расширение .mdb и имеют ту особенность, что содержат не только таблицы с данными, но и все сопровождающие программные средства: запросы, экранные формы, отчеты, макросы и др. СУБД Access поддерживает БД реляционного и древовидного (иерархического) типов. Реляционными называются такие БД, где данные представляются в виде обычных двумерных таблиц, обладающих некоторыми естественными ограничениями: отсутствием в клетках таблиц множественных значений и различающимися строками (хотя бы в одной ячейке). Такие таблицы называются «отношениями» (отношение по-английски – «relation»). Между таблицами-отношениями определяются связи, характеризуемые арностями, степенями и классами принадлежности. Арность показывает, сколько таблиц связывается между собой

данной связью. Наиболее распространены бинарные связи, которые связывают две таблицы. Для бинарных связей степени могут иметь один из типов: «один к одному», «один ко многим», «много ко многим». Степени показывают, в каком соответствии находятся строки одной и другой таблиц. Например, таблица «Студент» и таблица «Дисциплины» имеют степень связи «много ко многим», поскольку каждый студент изучает много дисциплин, а каждая дисциплина изучается многими студентами. Класс принадлежности связи определяется для каждого ее конца и показывает, обязательно ли должен быть связан каждый из элементов соответствующей таблицы. Например, если существуют дисциплины, которые никто из студентов не изучает, то класс принадлежности того конца связи, который примыкает к таблице «Дисциплины», необязателен. В настоящее время реляционная модель является наиболее часто используемой моделью данных. В древовидной модели предусмотрены иерархии таблиц, для чего в Access имеются подчиненные таблицы и формы.

3.2. Правила работы с пакетом

Главное окно пакета Access – 2003 имеет вид:

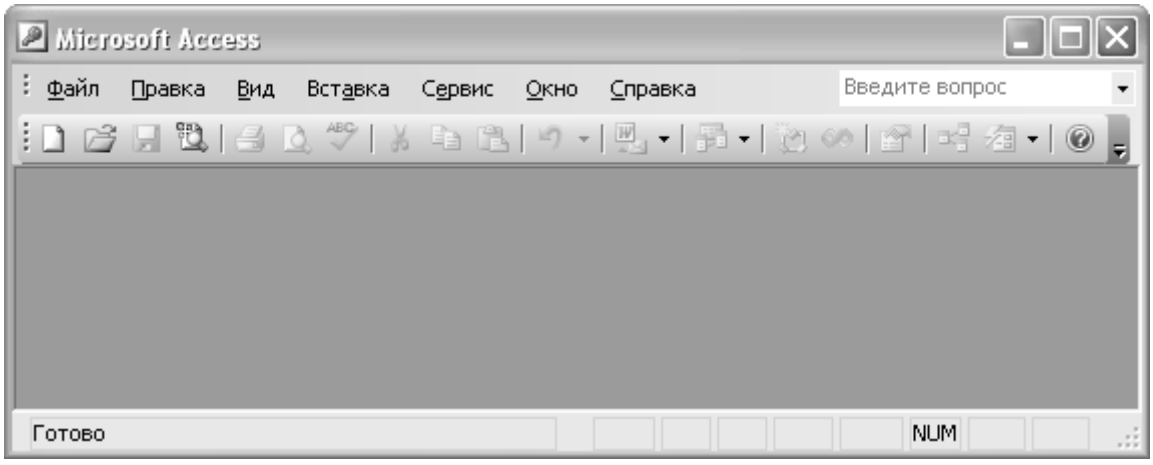


Рисунок 1 – Главное окно

Работа с программой начинается выбором одного из пунктов выпадающего меню опции «Файл» - «Создать базу данных» или «Открыть базу данных», в зависимости от того, создается новая БД или производится работа с уже существующей БД. При выборе опции создания новой БД появляется окно:

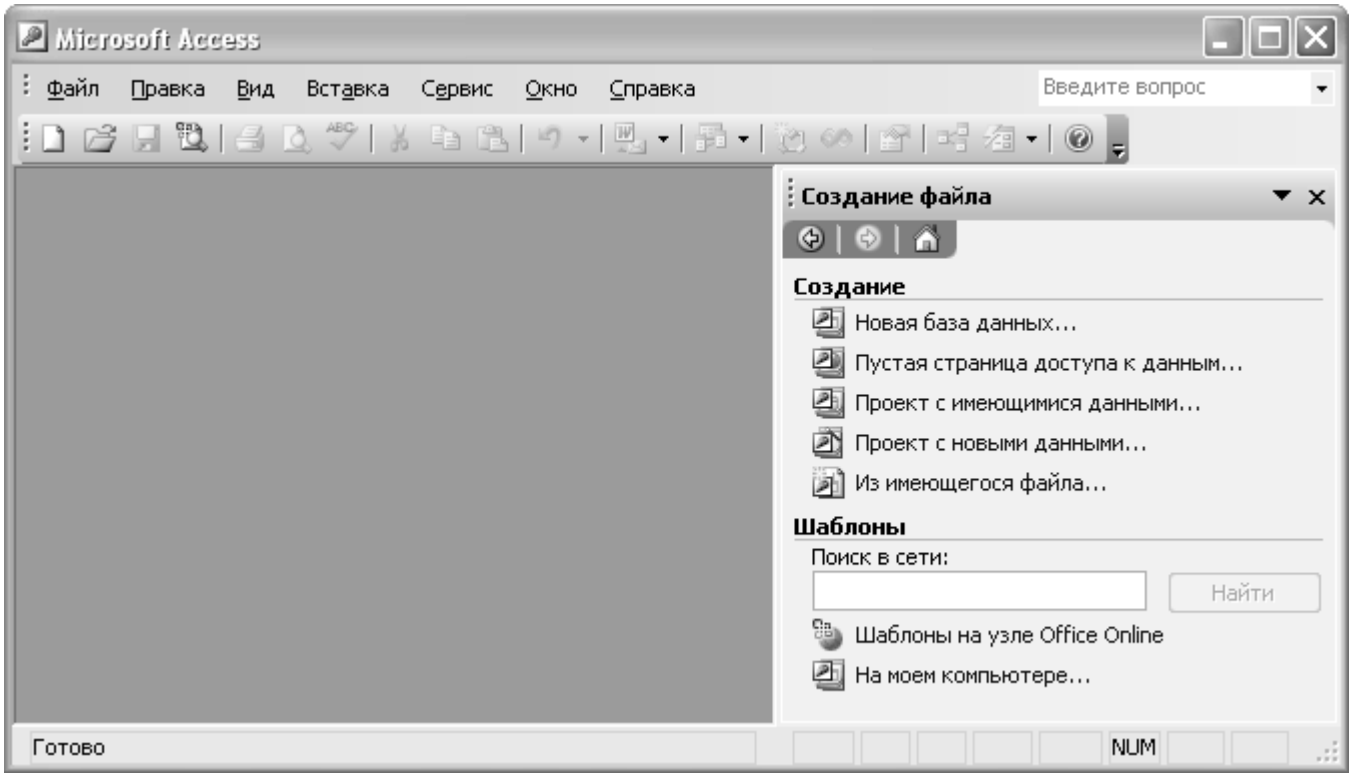


Рисунок 2 – Окно создания новой базы данных

В разделе «Создание» перечислены несколько опций, из которых рассмотрим только первую – «Новая база данных». Заметим, что следующие три опции предназначены для взаимодействия с удаленными базами данных по Internet или локальным сетям, а последняя – для формирования БД Access из файлов либо проектов Access, а также из веб-страниц.

После выбора опции по созданию новой БД открывается окно «Файл новой базы данных», в котором производится выбор пути и имени БД, после чего появляется окно базы данных вида:

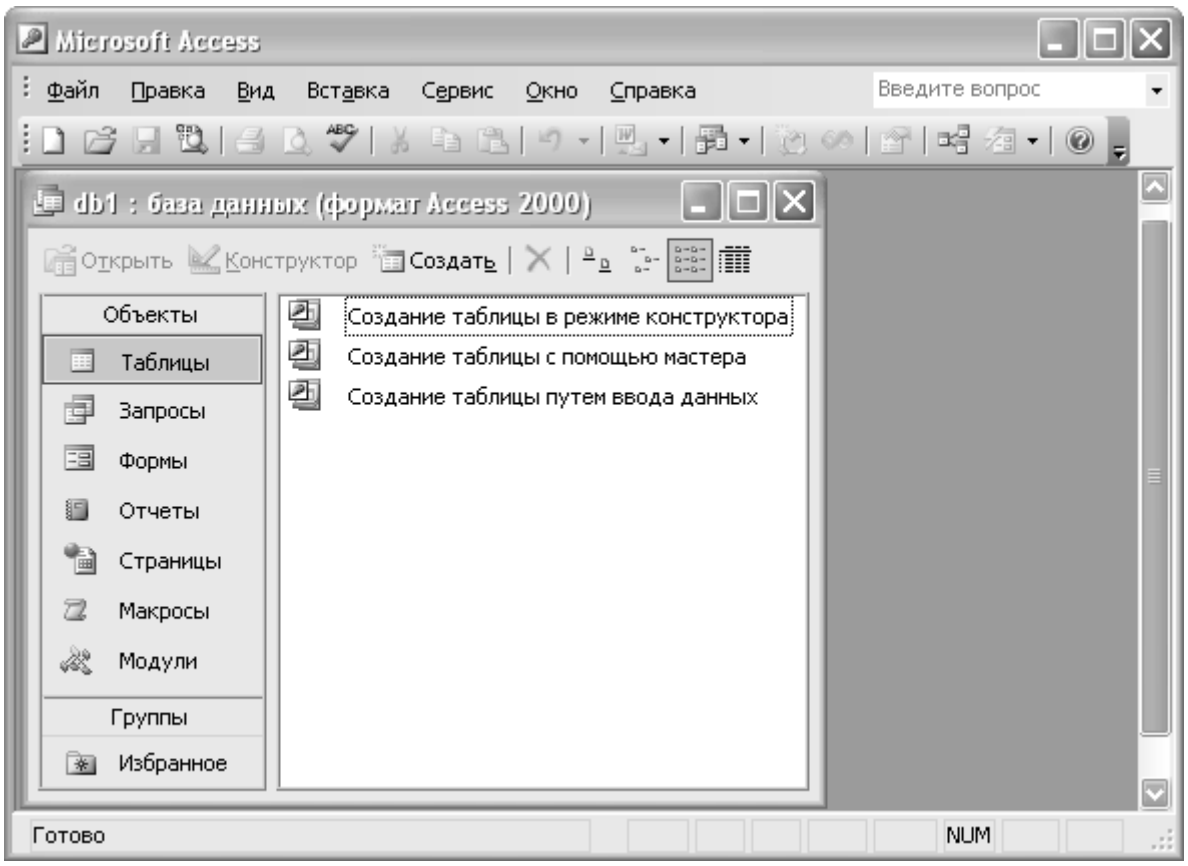


Рисунок 3 – Окно базы данных

Основными и наиболее часто используемыми объектами проектируемых БД являются таблицы, запросы, экранные формы и отчеты, для автоматизированной разработки которых в Access предусмотрены инструментальные средства «конструктор» и «мастер». Кроме того, для таблиц предусмотрено и непосредственное формирование таблиц.

Для проектирования таблиц существует несколько подходов. В настоящее время большую популярность приобрел ER-подход (от Entity-Relationship, т.е. сущность-связь). В итоге применения подхода образуется набор базовых и связных таблиц, которые затем вводятся в БД. Применению ER-подхода посвящены лабораторные работы №№ 7 – 9.

Одной из основных целей применения БД и СУБД является удовлетворение всевозможных запросов к БД для получения выборочных сведений. Для формирования запросов получил широкое распространение язык SQL (от Structure Query Language – язык структурных запросов). В Access запросы можно выполнять как непосредственно на SQL, так и с помощью конструктора и мастера запросов. Конструированию запросов с помощью языка SQL посвящена лабораторная работа № 6, запросы с помощью конструктора и мастера осваиваются в лабораторной работе № 2.

Для пользовательских приложений, имеющих дело с фиксированным набором запросов, разрабатываются интерактивные экранные формы и отчеты. Формы и отчеты предназначены для общения с БД пользователей, не являю-

щихся специалистами в области проектирования и эксплуатации БД (например, бухгалтеров или менеджеров). Различие между отчетами и формами в том, что формы являются типом объектов базы данных, который обычно используется для отображения данных в базе данных, а отчет является эффективным средством представления данных в печатном формате. Разработке форм и отчетов посвящены лабораторные работы №№ 3, 4.

При выборе опции «Открыть базу данных» появляется окно «Открытие файла базы данных» традиционного для системы Windows вида, с помощью которого выбирается требуемая БД с расширением .mdb либо другими, перечисленными в поле ввода в нижней части окна.

3.3. Просмотр базы данных

После открытия базы данных появляется такое же окно, как на рисунке 3, но с непустыми списками таблиц и других видов объектов в правой его части. Так, после открытия учебной БД из файла Tutorial.mdb автоматически устанавливается режим просмотра таблиц с окном, имеющим вид:

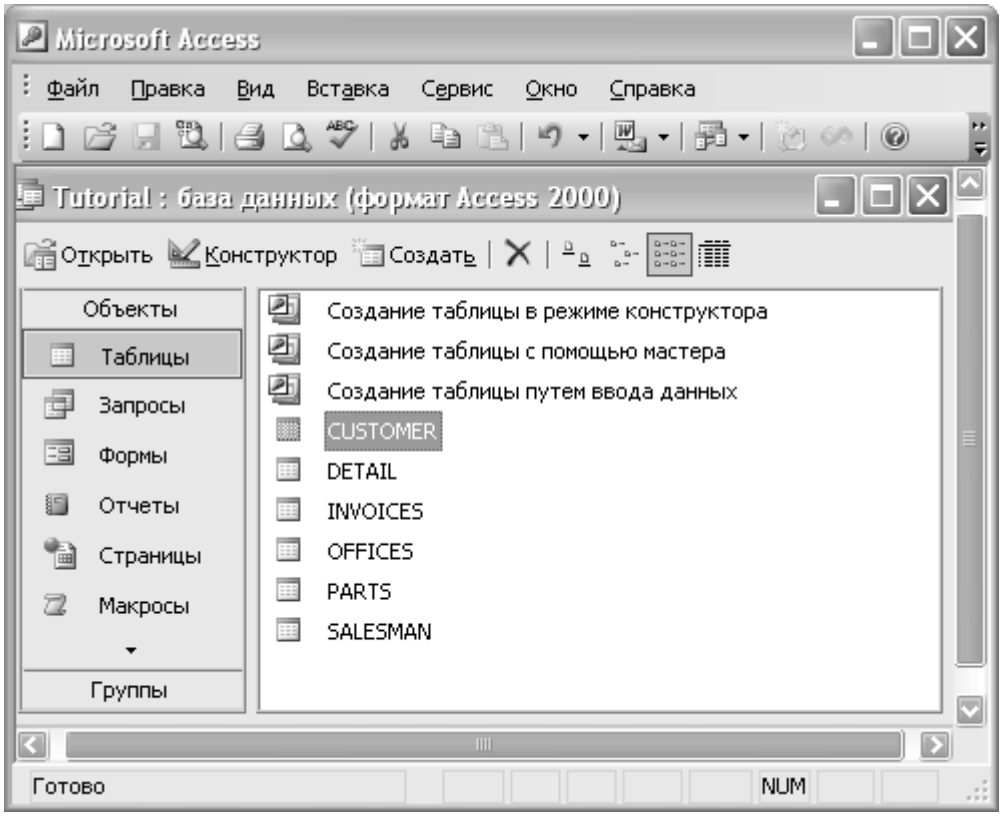


Рисунок 4 – Окно базы данных Tutorial.mdb в режиме просмотра таблиц

Как видно, БД состоит из пяти таблиц CUSTOMER (фирмы – покупатели), DETAIL (деталь – имеется в виду детальная информация о товаре), INVOICES (счета – имеются в виду заказы покупателей в виде оплаченных сче-

тов), OFFICES (офисы фирм – продавцов), PARTS (части – имеются в виду части заказа), SALESMAN (продавец). Передвигая световой маркер слева в позицию «Запросы», можно увидеть перечень запросов. Аналогично можно посмотреть и состав других объектов БД. В данном состоянии БД «Tutorial»других объектов не содержит. Отметим, что эта БД импортирована из учебной БД, прилагаемой к пакету Foxpro (широко распространенная СУБД настольного типа). Импорт производился путем выбора пункта «Внешние данные – Импорт» опции «Файл» из горизонтального меню главного окна Access (рис. 1). В поле «Тип файлов» открывшегося окна «Импорт» установлен тип «dBase IV (*.dbf)», после чего из учебной БД Foxpro импортированы пять файлов с перечисленными выше именами. Таким образом, это еще один способ создания баз данных Access на основе уже существующих БД.

Базы данных, представляемые только в виде набора таблиц, представляют собой простейший тип БД, характерный для непромышленных применений. Последние версии Access позволяют применить более сложную их организацию, определив для таблиц первичные ключи (primary key), внешние ключи (foreign key) и задать на основе ключей автоматическую проверку целостности БД. На основе таблиц и ключей может быть построена схема БД, чему посвящены лабораторные работы № 5 (в Access) и 8 (в Erwin).

Для просмотра таблицы следует дважды щелкнуть левой кнопкой мыши по ее имени, в результате чего откроется окно таблицы, где первая строка содержит наименования полей (шапку таблицы), а последующие строки – значения полей (кортежи). Ниже приведено соответствующее окно после открытия таблицы CUSTOMER:

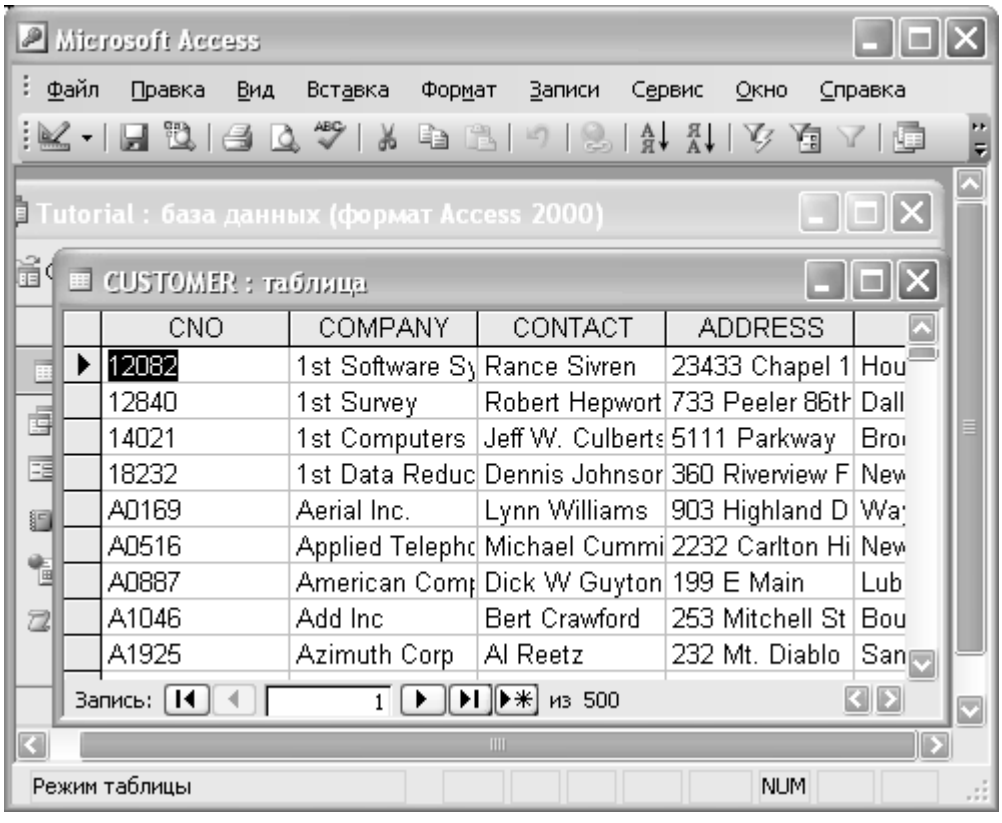


Рисунок 5 – Окно просмотра таблицы

Для удобства пользования прокруткой левую часть таблицы можно зафиксировать. После фиксации прокрутка по горизонтали не действует на закрепленные столбцы, и они не уходят с экрана. Для фиксации необходимо мышью выделить столбцы левой части таблицы (например, CNO и COMPANY) и в опции горизонтального меню «Формат» выбрать пункт «Закрепить столбцы». Если подобная операция будет выполнена над внутренними столбцами, то после их закрепления они переместятся в левую часть таблицы. Выделение столбцов производится нажатой левой кнопкой мыши, когда курсор перемещается по шапке таблицы. Следует отметить, что перемещенные влево внутренние столбцы можно возвратить на прежнее место, только закрыв таблицу без сохранения изменений макета. Изменение ширины столбцов удобно делать мышью, надвинув курсор на разделительную линию между заголовками столбцов и, нажав левую кнопку, переместить ее в нужное положение. Для сокрытия части столбцов при просмотре таблицы их также необходимо выделить и из опции «Формат» выбрать пункт «Скрыть столбцы» (обратная операция – «Отобразить столбцы»). Подробнее указанные операции описаны в справочной системе Access в пункте «Работа с данными – Просмотр данных – Работа с таблицами – Изменение внешнего вида таблиц».

3.4. Создание таблицы

3.4.1. Создание таблицы в режиме конструктора

Конструктор таблиц имеет вид окна:

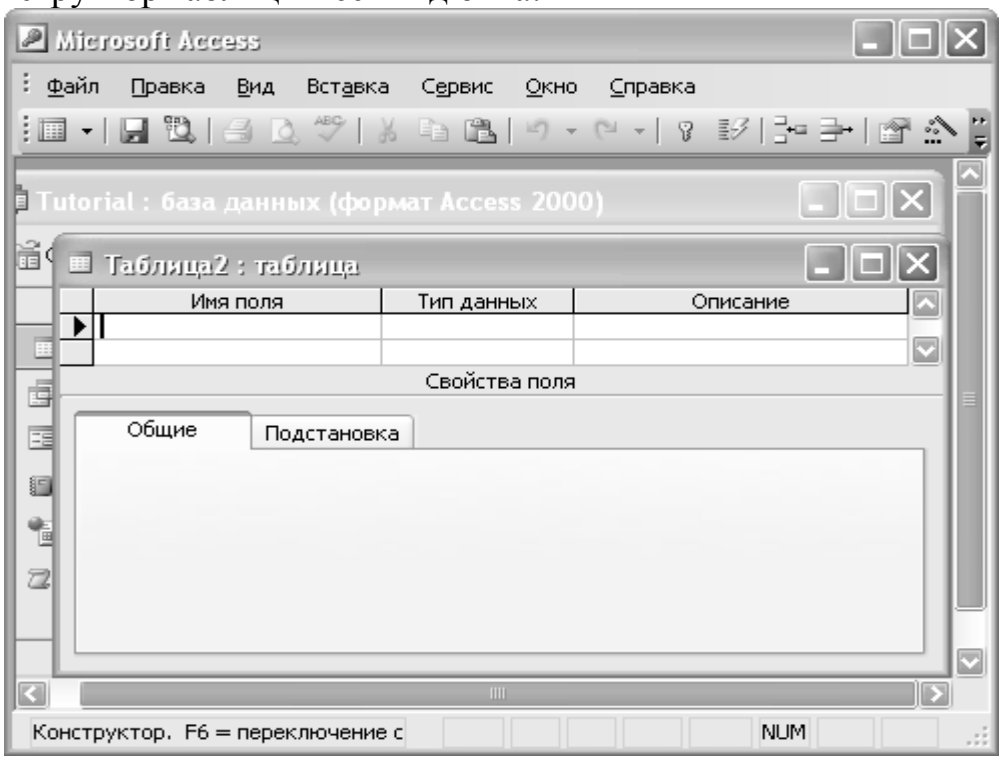


Рисунок 6 – Окно конструктора таблиц

В табличной части окна задаются имена полей (могут состоять из любых символов, включая кириллицу и пробелы, за исключением точки, восклицательного знака, квадратных скобок, апострофа). Длина имени поля – до 64 символов. Тип данных задается из зарезервированного списка (чаще всего числовые различных форматов, символьные, типа «дата» и др.). В частности, для полей, предназначенных для хранения длинных текстов, используется тип «Memo». Описание содержит произвольный текст, характеризующий содержательный смысл имени поля. В нижней части находятся две закладки – «Общие» и «Подстановка». Пример информации из закладки «Общие» для одного из текстовых полей таблицы CUSTOMER приведен на рисунке 7.

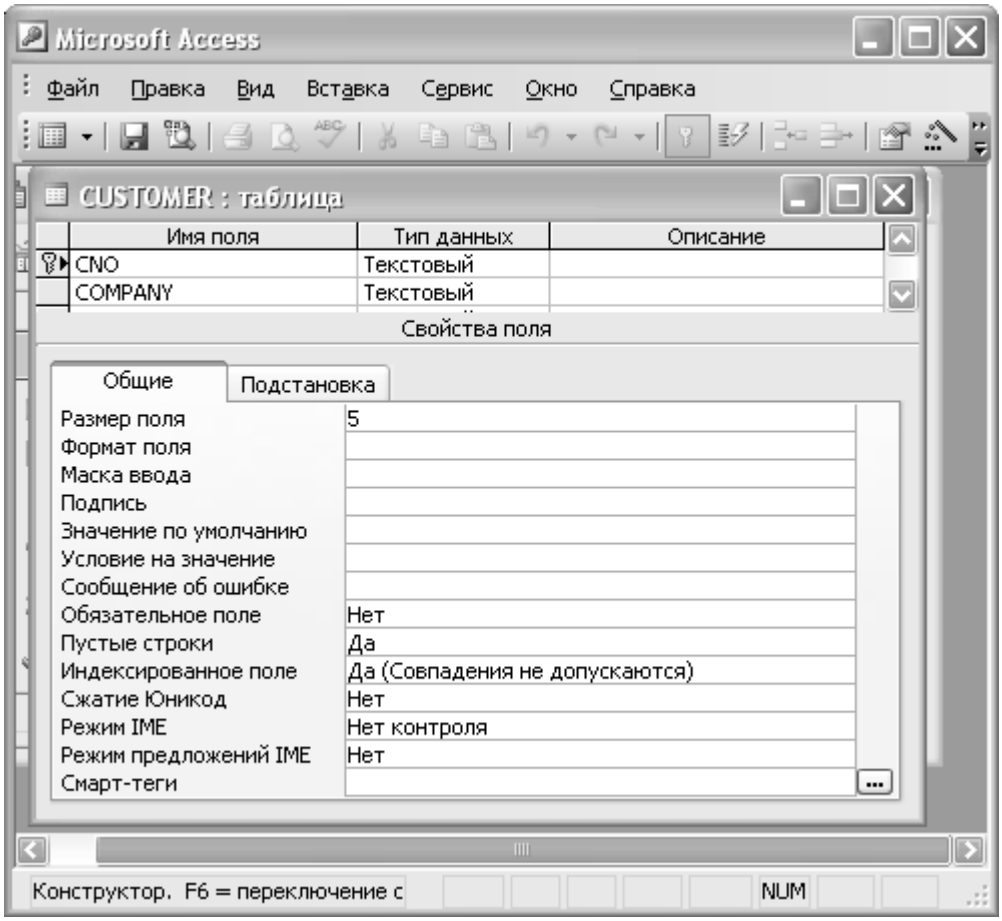


Рисунок 7 – Информация закладки «Общие» конструктор таблиц для текстового поля

Перечень пунктов зависит от типа поля. При этом наиболее часто задается размер поля (в символах) и формат (для чисел, дат). Остальные данные заполняются при необходимости и предназначены большей частью для контроля вводимых данных.

Закладка «Подстановка» содержит поле ввода, в котором выбирается тип элемента управления из множества («Поле», «Список», «Поле со списком»). При выборе типа «Поле» значения вводятся с клавиатуры. В остальных случаях

в правой части поля высвечивается инструментальная кнопка открытия списка, из которого выбираются значения. Например, если необходимо вводить в поле значения из некоторого заранее составленного списка (типа фамилий сотрудников), выбираем «Список» или «Поле со списком». При этом появляются дополнительные пункты, где против «Тип источника строк» выбираем «Таблица или запрос»; против «Источник строк» – запрос из таблицы (с данными о сотрудниках); против «Присоединенный столбец» – номер столбца результата запроса (с фамилией сотрудника). Пример приведен на рисунке 8.

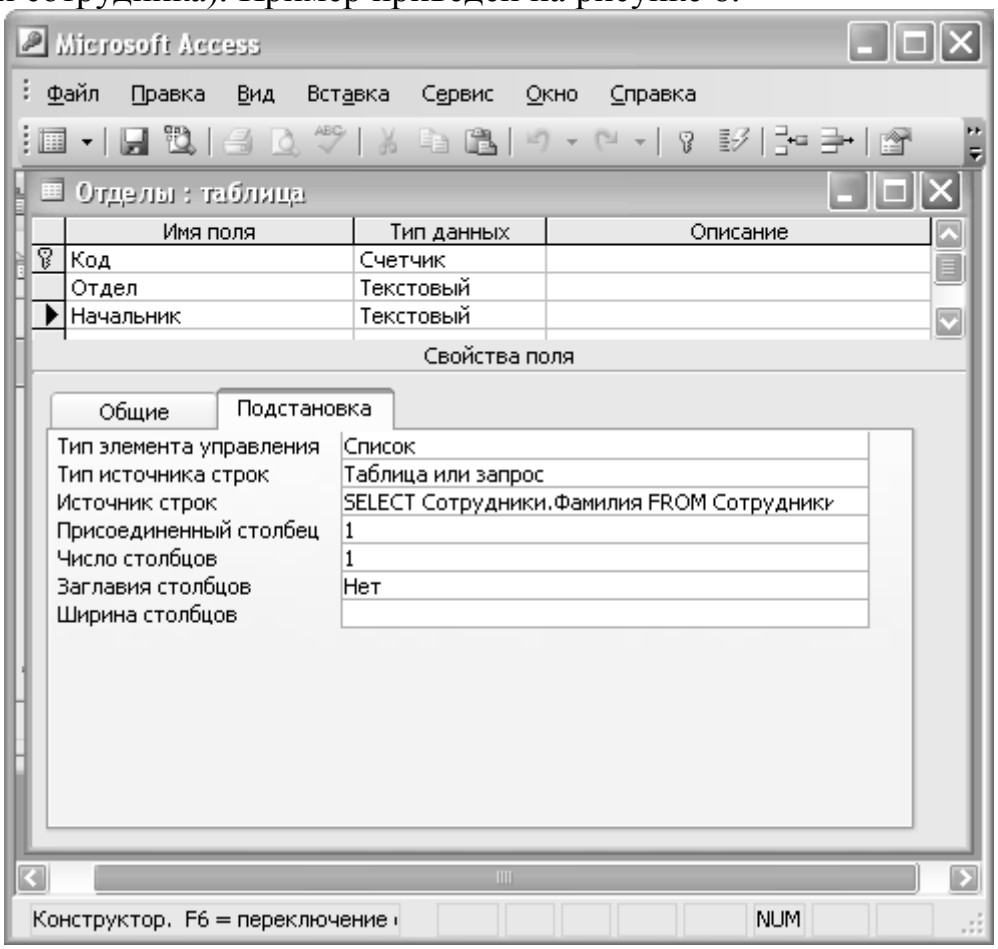


Рисунок 8 – Формирование поля с выборкой из списка

3.4.2. Создание таблицы в режиме мастера

Мастер предназначен для облегчения процесса создания таблиц, используя наборы шаблонов. Для подключения мастера необходимо в окне базы данных (рисунок 4) выбрать опцию «Создание таблицы с помощью мастера» и далее следовать инструкциям.

3.4.3. Создание таблицы путем ввода данных

Для создания таблицы этим способом необходимо в окне базы данных (рисунок 4) выбрать опцию «Создание таблицы путем ввода данных», после чего откроется окно с пустым макетом таблицы:

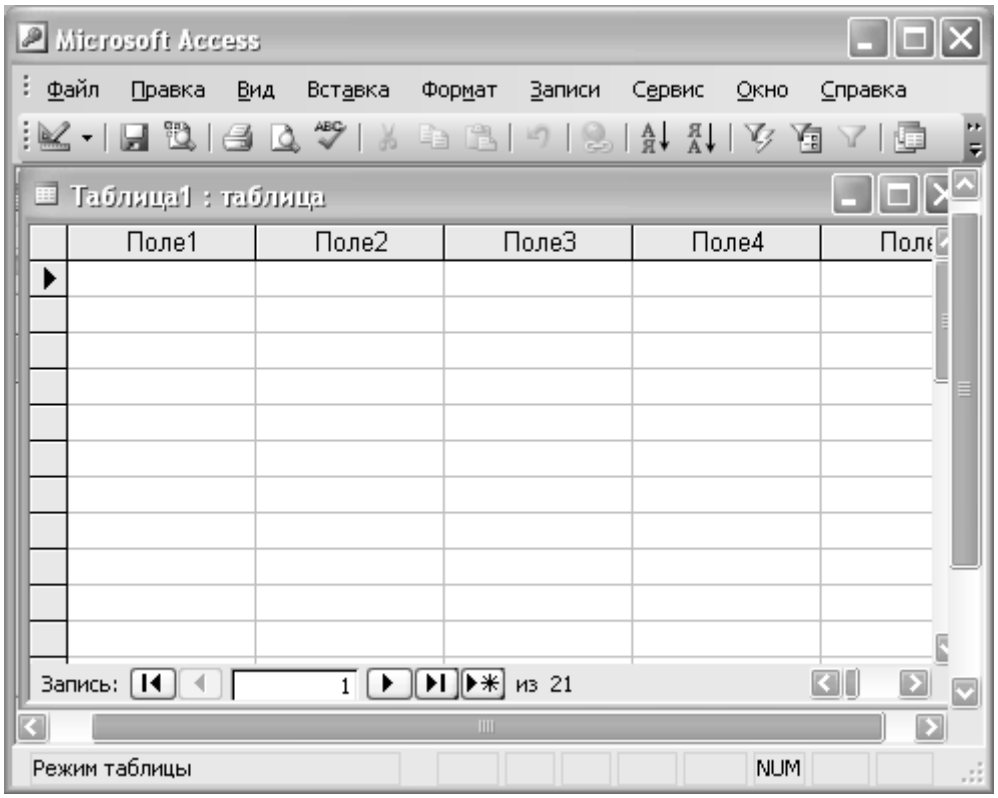


Рисунок 9 – Окно таблицы

Для изменения имен полей (Поле1, Поле2, …) на другие необходимо щелкнуть правой кнопкой мыши по имени поля и в открывшемся контекстном меню выбрать опцию «Переименовать столбец».

После переименования полей таблицу заполняют данными – символьными строками, числами в произвольном формате, датами и т.д. После этого либо закрывают таблицу, либо переходят в режим конструктора таблиц и проверяют, правильно ли установлены типы полей. При этом производится первоначальное сохранение таблицы из автоматически появляющегося окна:

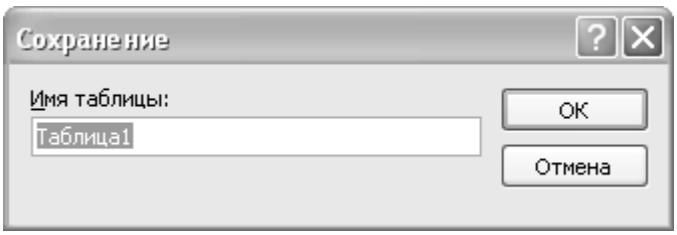


Рисунок 10 – Окно первоначального сохранения таблицы

Следует отметить, что таким способом нельзя задать поля типа «memo» и некоторые другие. Кроме того, при неправильном вводе дат (например, с номером месяца, большим 12) эти данные будут интерпретированы как символьные. Поэтому после формирования таблицы этим способом рекомендуется перейти в режим конструктора и проверить правильность типов полей. Таким образом, данный режим используется на предварительной стадии создания таблиц.

4. ПРОГРАММА РАБОТ

4.1. Импорт базы данных

Произведите импорт dbf-файлов учебной базы данных из …\POLSOV\COMMON\TUTORIAL. Для этого необходимо:

- Войти в Access.
- Выбрать опцию «Новая база данных».
- Выбрать пункт «Внешние данные – Импорт» опции «Файл» из горизонтального меню главного окна Access (рисунок 1). В поле «Тип файлов» открывшегося окна «Импорт» установить тип «dBase IV (*.dbf)», после чего из указанной выше директории поочередно импортировать файлы с именами CUSTOMER, DETAIL, INVOICES, OFFICES, PARTS, SALESMAN.

4.2. Просмотр базы данных

Откройте поочередно все таблицы БД и ознакомьтесь с их содержимым. Выпишите наименования полей всех таблиц и переведите их на русский язык. Для одной из таблиц выполните закрепление (фиксацию) нескольких левых столбцов и последующее их освобождение; сокрытие и отображение нескольких столбцов.

4.3. Создание таблицы путем ввода данных

Создайте новую таблицу «Отделы» (см. пункт 3.4.3), состоящую из полей «Наименование», «Фамилия начальника», «Количество сотрудников», «Дата образования», «Средняя зарплата» и введите три – четыре строки данных. В качестве десятичного разделителя используйте запятую. Ширину колонок сделайте такой, чтобы в них помещались полностью имена полей и данные. Переключитесь в режим конструктора (сохранив таблицу под именем «Отделы») и проверьте типы данных. Перед сохранением появится окно с вопросом о создании ключевого поля, на который ответьте утвердительно. Для поля «Средняя зарплата» исправьте тип «Числовой» на «Денежный». В процессе закрытия таблицы появится окно с запросом о сохранении измененного макета и

предупреждение об изменении размера модифицированного поля (FieldSize). На запрос и предупреждение ответьте утвердительно.

4.4. Создание таблицы в режиме мастера

Создайте таблицу «Сотрудники» (см. пункт 3.4.2) в режиме мастера. Для этого войдите в режим мастера из окна базы данных, в результате чего появится окно мастера таблиц:

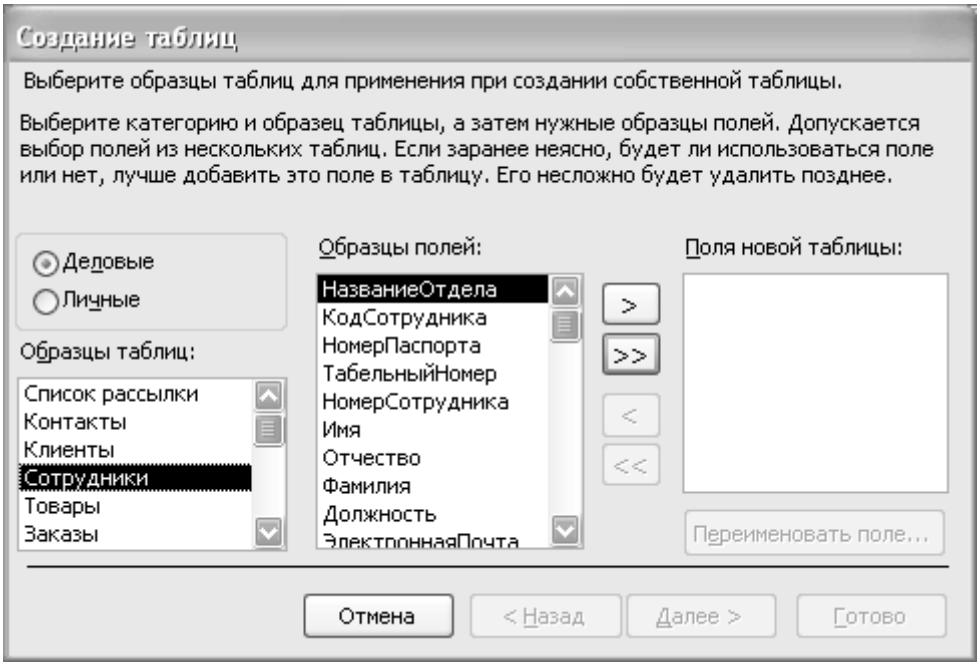


Рисунок 11 – Окно мастера таблиц

Далее выберите в качестве шаблона таблицу «Сотрудники», оставив поля «Код сотрудника», «Название отдела», «Фамилия», «Имя», «Отчество» (именно в таком порядке и с пробелами), выбрав в качестве ключа «Код сотрудника» в виде сочетания чисел и букв. В процессе прямого и обратного переноса требуемых полей пользуйтесь стрелками, а для переименования – кнопкой «Переименовать поле…». Таблицу «Сотрудники» свяжите с таблицей «Отделы» по типу «много к одному» (т.е. в одном отделе работает много сотрудников, но каждый сотрудник может работать только в одном отделе). Для этого в списке связей открывшегося окна выберите строку «Не связана с «Отделы»», нажмите кнопку «Связи…» и следуйте инструкциям. Перед вводом данных выберите опцию «Ввести данные в таблицу с помощью формы, создаваемой мастером», а далее заполните три – четыре строки таблицы. Обратите внимание на то, что в таблице «Сотрудники» появилось поле «Код». Это поле называется связным и «мигрировало» из таблицы «Отделы», где играло роль первичного ключа (primary key). В данной же таблице оно играет роль внешнего ключа (foreign key). При

его заполнении необходимо указывать номер (код) отдела, в котором работает данный сотрудник.

4.5. Модификация таблицы с помощью конструктора

Войдите в таблицу «Сотрудники» и переключитесь в режим конструктора. Поставим задачу модификации поля «Код» таким образом, чтобы значения в этом поле выбирались из выпадающего списка, автоматически формируемого из таблицы «Отделы». Для того, чтобы было понятно, о каком коде идет речь, изменим наименование поля на «Код отдела». Кроме того, поскольку этот список нужен для облегчения процесса заполнения колонки, выведем в список не одну, а две колонки – с кодом и названием отдела. При формировании поля с выборкой из списка заполните поля закладки «Подстановка» следующим образом:

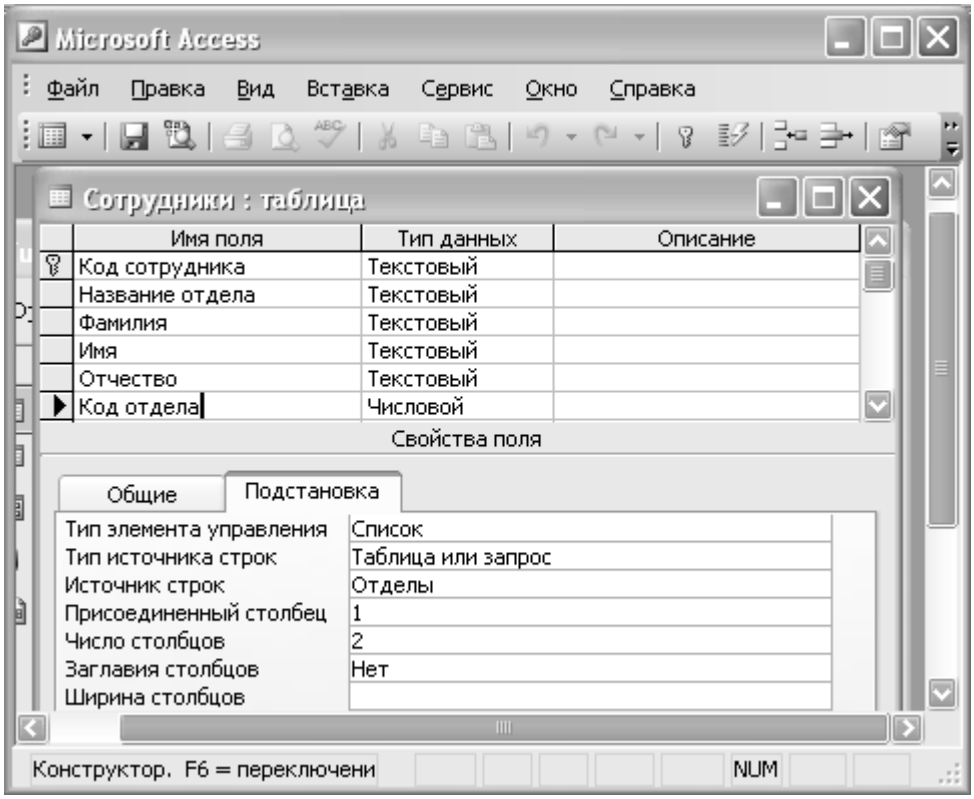


Рисунок 12 – Заполнение списка закладки «Подстановка»

После модификации таблицы «Сотрудники» перейдите вновь в режим таблицы (опция «Вид – Режим таблицы») и попробуйте ввести данные в последнюю колонку. Для того, чтобы была видна и колонка с названиями отделов, последнюю колонку расширьте.

5. ЗАЩИТА РАБОТЫ

Письменный отчет по работе не делается. Защита работы производится у компьютера в форме демонстрации результатов и отдельных операций, а также ответов на контрольные вопросы из следующего перечня (на вопрос модульного контроля в письменной форме):

1. Вопрос из блока №1 модульного контроля (в письменной форме)
2. Определение реляционной базы данных
3. Понятие «отношения»
4. Понятие связи и ее характеристики (арность, степень, класс принадлежности)
5. Объекты базы данных Access (таблицы, запросы, формы, отчеты)
6. Первичные и внешние ключи
7. Виды окон СУБД Access: главное, базы данных, конструктора таблиц с закладками, таблиц, мастера таблиц

ЛАБОРАТОРНАЯ РАБОТА № 2
СОЗДАНИЕ ЗАПРОСОВ К БАЗЕ ДАННЫХ

1. ЦЕЛЬ РАБОТЫ

Цель работы – изучить и практически освоить пользовательский интерфейс системы управления базами данных (СУБД) настольного типа Access, научиться создавать запросы к базам данных (БД) Access.

2. ОБЪЕКТ ИССЛЕДОВАНИЯ

Объектом исследования является СУБД Access фирмы Microsoft. Инструментом исследования является программный пакет Access.

3. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

3.1. Понятие запроса и средства их разработки в пакете Access

Запросы предназначены для поиска и вывода данных, удовлетворяющих заданным условиям, включая данные из нескольких таблиц. Запрос также может обновлять или удалять несколько записей одновременно и выполнять стандартные или пользовательские вычисления с данными.

Для создания запросов могут использоваться несколько средств. Наиболее современное универсальное средство – язык SQL (от Structure Query Language – язык структурных запросов), применяемый в качестве языка запросов практически во всех современных СУБД. В Access запросы можно выполнять как непосредственно на SQL, так и с помощью конструктора и мастера запросов. Созданию запросов с помощью языка SQL посвящена лабораторная работа № 6. В настоящей работе осваивается создание запросов с помощью конструктора и мастера.

3.2. Создание запросов с помощью конструктора

Для выхода в данный режим необходимо в окне базы данных установить световой маркер на опцию «Запросы» и выбрать опцию «Создание запроса в режиме конструктора».

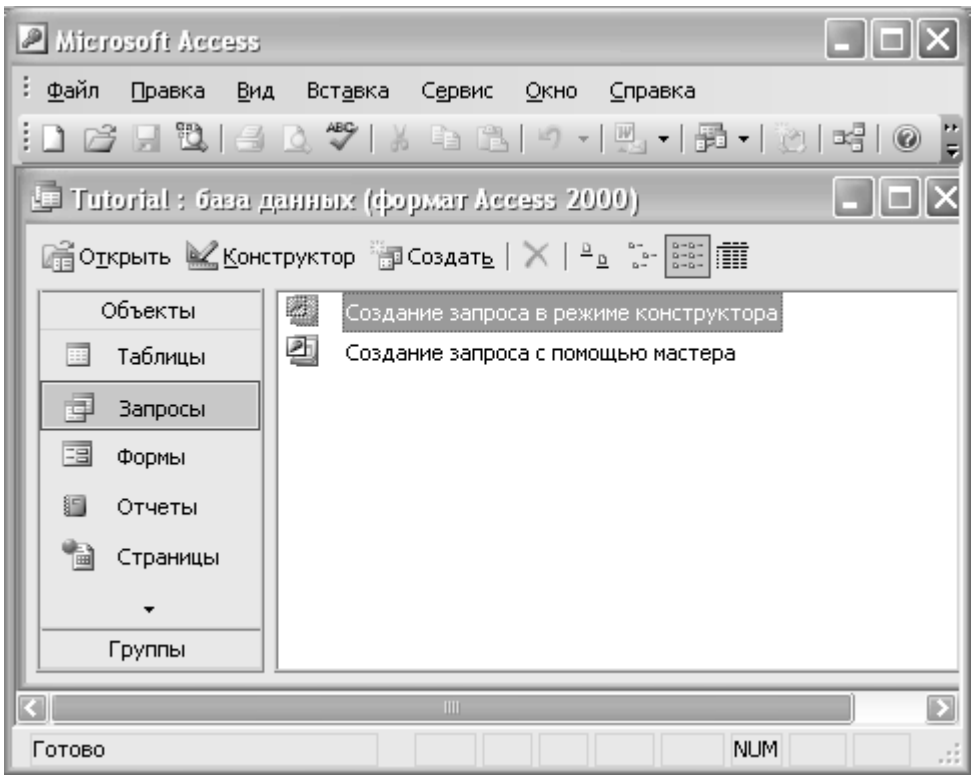


Рисунок 1 – Переход в режим создания запроса

После выхода в режим открывается окно вида:

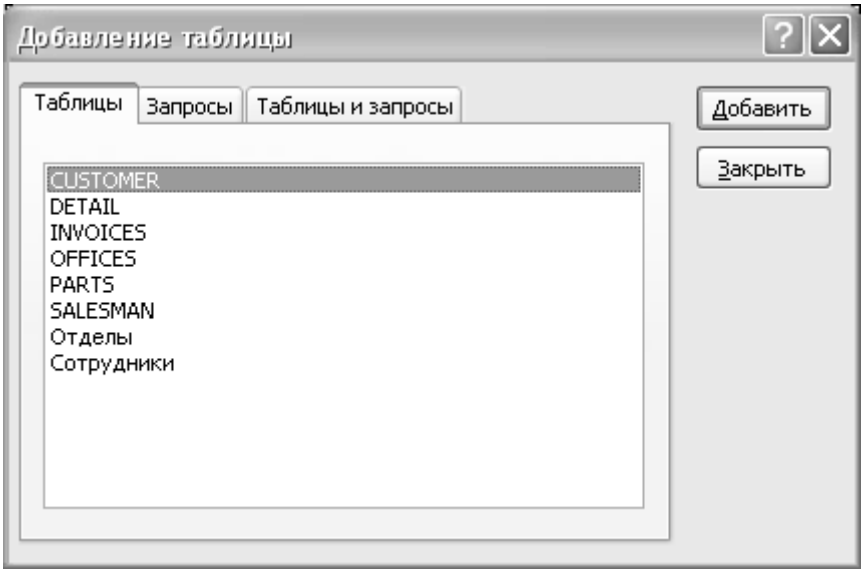


Рисунок 2 – Окно добавления источников запроса

Из этого окна необходимо выбрать таблицы, из которых будет формироваться результат запроса. В простых запросах данные выбираются из одной таблицы или запроса, в сложных запросах – из нескольких таблиц или запросов, которые должны быть связаны через внешние ключи [3]. Для выбора источников запроса в окне имеются три закладки – «Таблицы», «Запросы», «Таблицы и

запросы». Каждый из выбранных источников запроса с помощью кнопки «Добавить» переносим в специальное окно «Запрос на выборку» (открывается автоматически после нажатия кнопки), после чего окно «Добавление таблицы» закрываем.

После выбора источников запроса (например, одной таблицы CUSTOMER) переходим к нижней части окна «Запрос на выборку»:

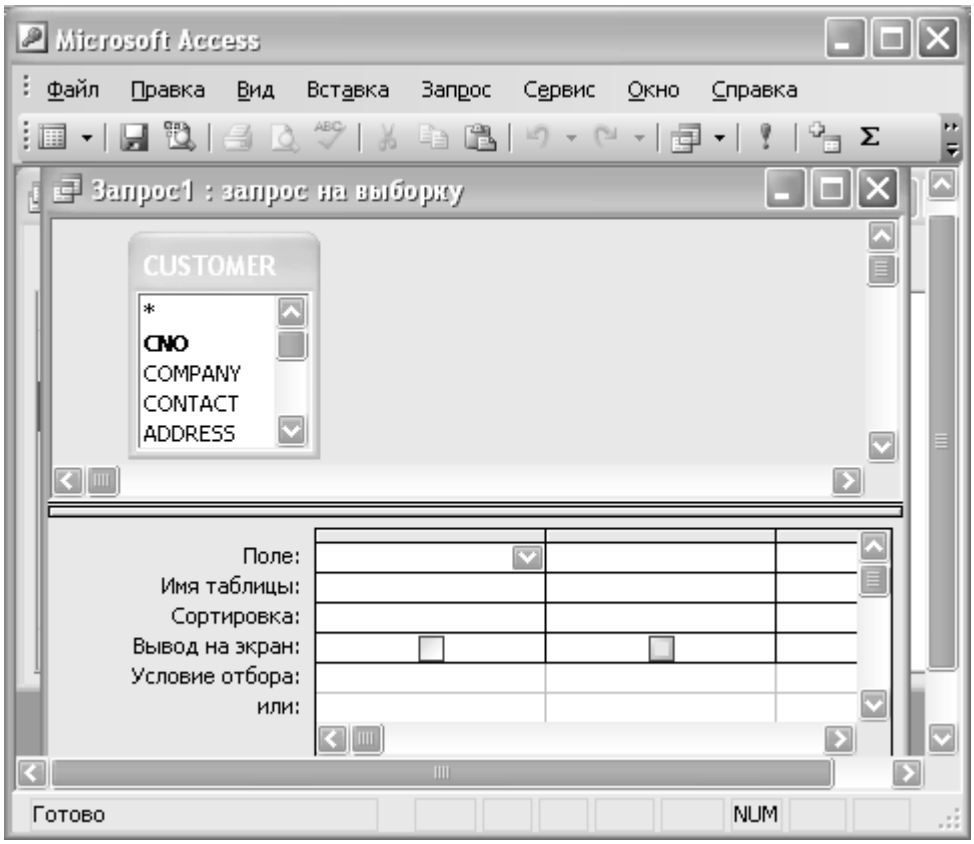


Рисунок 3 – Окно «Запрос на выборку»

В каждой из строк «Поле», «Имя таблицы», «Сортировка» имеются кнопки раскрытия списков (на рисунке 3 видна в первой клетке строки «Поле»). В строке «Поле» отбираются те поля, которые будут помещены в результат запроса. Если нужно выводить все поля, выбирается «*» (для всех полей таблицы CUSTOMER – соответственно CUSTOMER.*). В строке «Имя таблицы» задается имя таблицы, в строке «Сортировка» – вид сортировки, т.е. «по возрастанию» или «по убыванию». В строках «Условие отбора» и «или» задаются логические выражения отбора. Логические выражения строятся в предположении, что в клетках формируются операции простого сравнения со значением соответствующего поля, логические связки «и» реализуются как бы «по горизонтали», а связки «или» – по вертикали. Однако в клетках можно записывать и произвольные логические выражения, пользуясь синтаксисом языка Visual Basic. Для тех пользователей, которые не знают этого языка, предусмотрен конструктор выражений. Вызов конструктора осуществляется нажатием правой кнопки мыши путем выбора пункта «Построить» контекстного меню.

Например, составим запрос из таблицы CUSTOMER всех расположенных в алфавитном порядке городов (CITY) из штата Мичиган (‘MI’ в поле STATE), где имеются покупатели. Соответствующим образом заполненное окно запроса на выборку, где использован оператор сравнения строк Like (подобен), имеет вид:

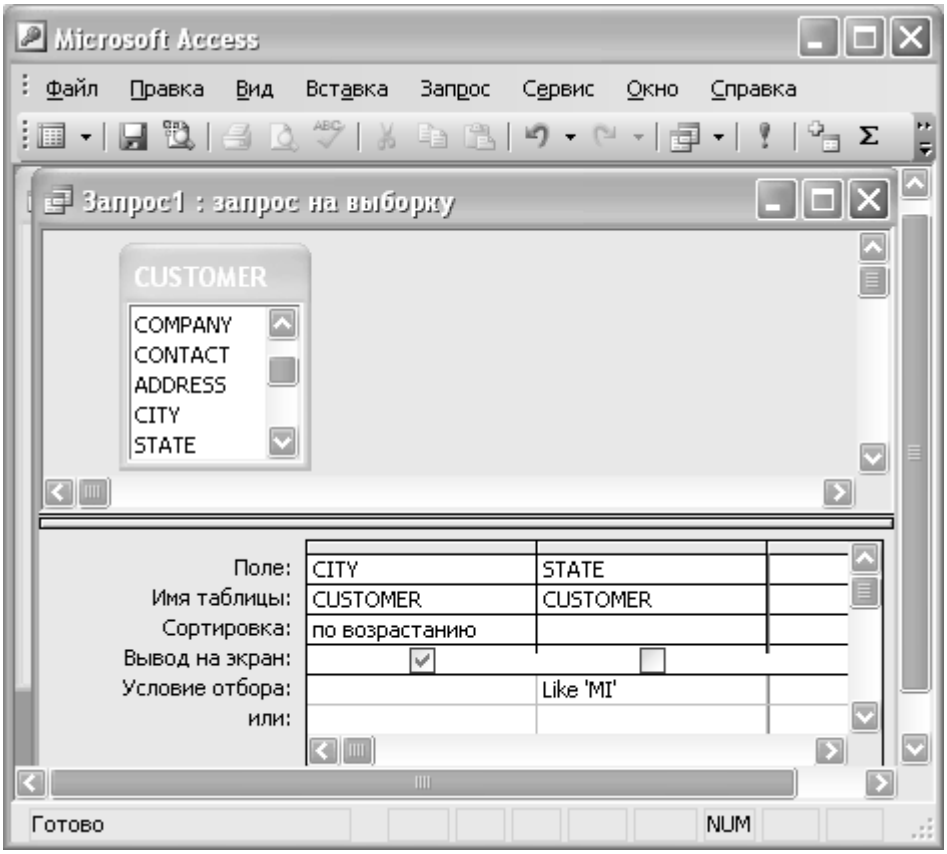


Рисунок 4 – Простой запрос к базе данных Tutorial.mdb

После закрытия окна запроса на выборку (с сохранением запроса под именем «Запрос1») в окне базы данных появится имя этого запроса. Открыв запрос двойным щелчком, получим результат запроса:

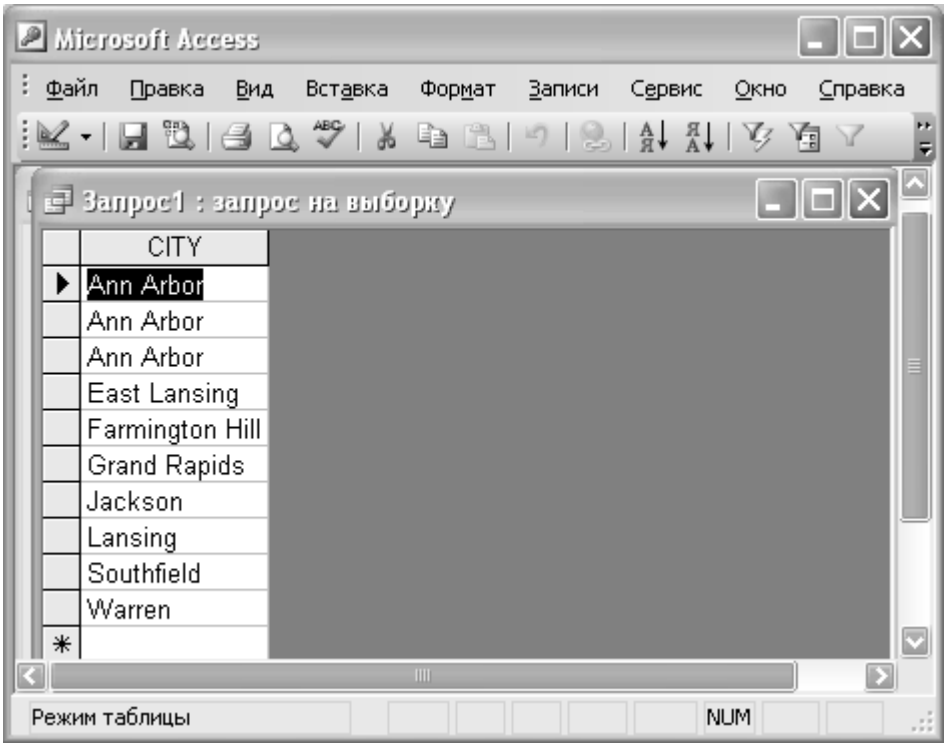


Рисунок 5 – Результат простого запроса с повторяющимися строками

Получился результат с первыми тремя одинаковыми строками (в городе Ann Arbor три покупателя). Это не совсем то, что мы хотели, однако в конструкторе убрать повторяющиеся строки нельзя. Для того, чтобы это сделать, раскроем опцию «Вид» и выберем пункт «Режим SQL». В результате появится окно:

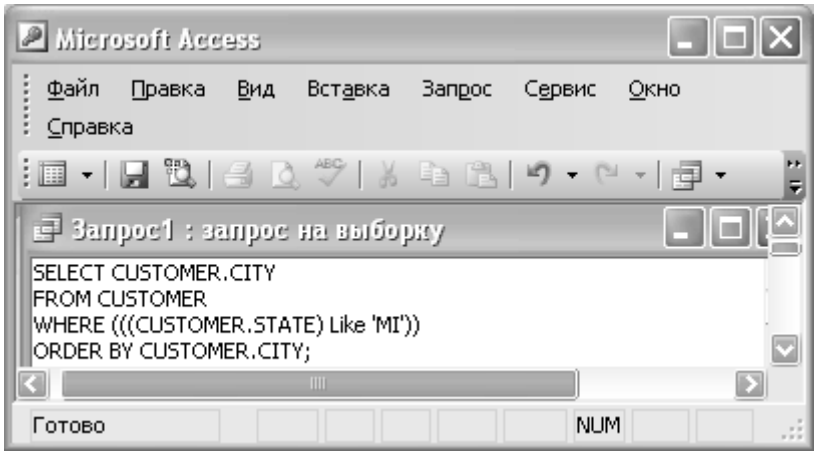


Рисунок 6 – Команда SQL

В окне находится команда SELECT, осуществляющая требуемый отбор. Добавим после слова SELECT слово DISTINCT (различный):

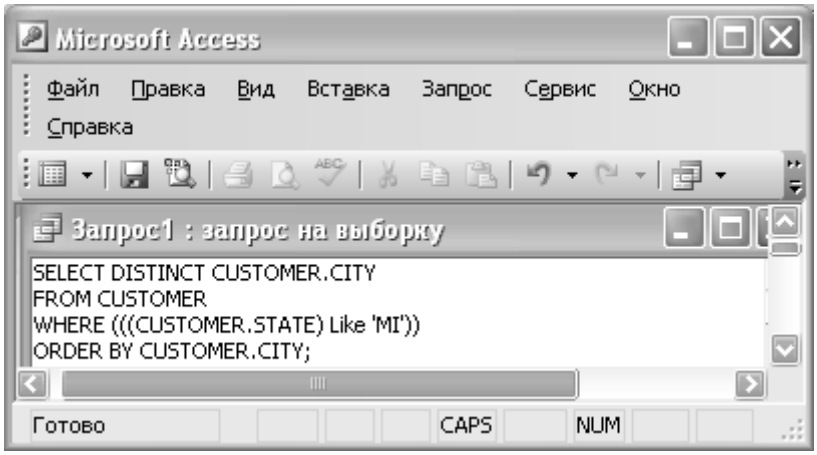


Рисунок 7 – Модифицированная команда SQL

В итоге получим результат запроса с различными строками. Переименуем запрос «Запрос1» на «Города из Мичигана» с помощью контекстного меню, вызываемого правой кнопкой мыши, когда ее курсор «надвинут» на «Запрос1» в окне базы данных.

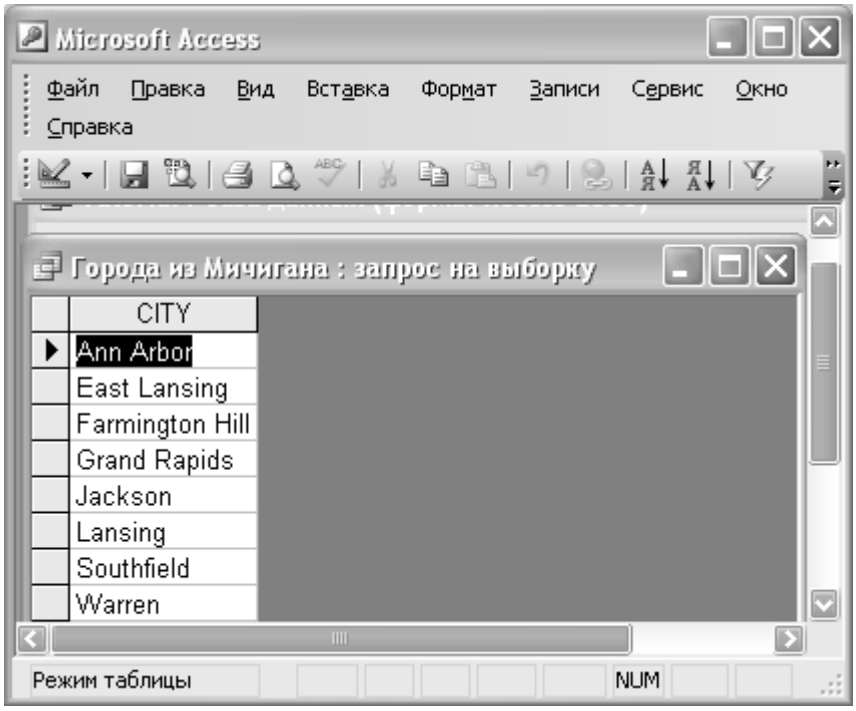


Рисунок 8 – Результат запроса без повторяющихся строк

Составим еще один запрос на основе полученного выше запроса с именем «Города из Мичигана» - запрос о городах с покупателями штат Мичиган, начинающиеся на букву «А». Для этого из окна базы данных вида рис.1 вызовем конструктор запросов и перейдем к закладке «Запросы» и так же точно, как создавали «Запрос1», создадим следующий запрос:

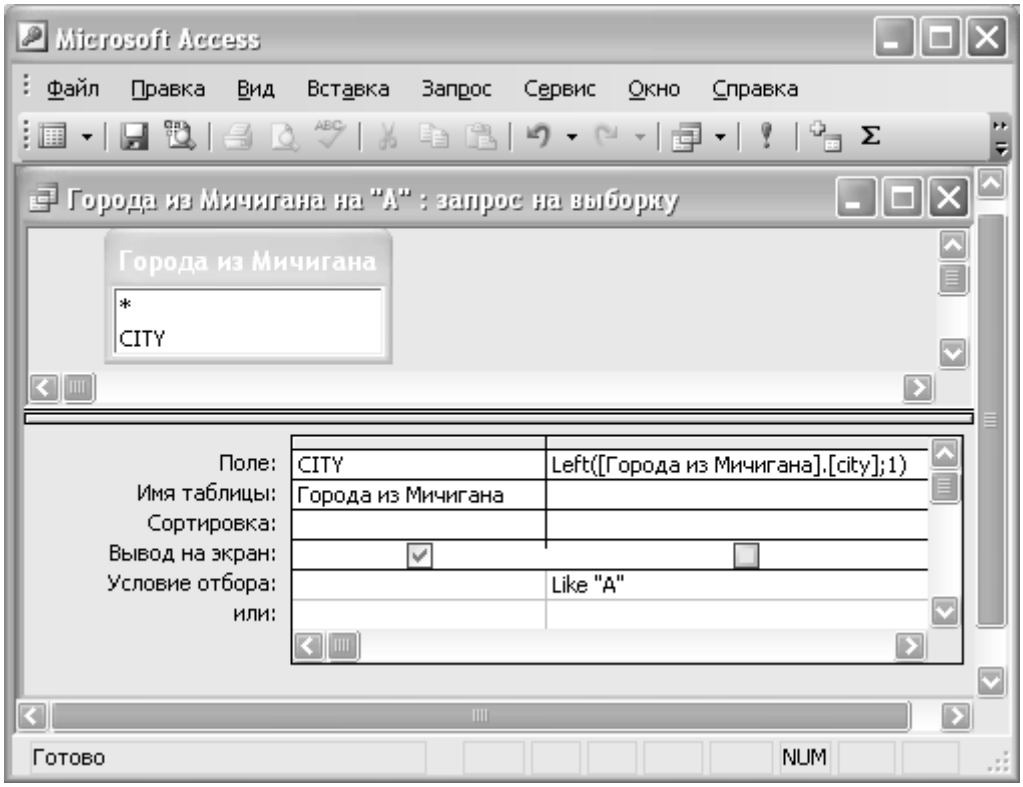


Рисунок 9 – Запрос на основе запроса

В данном случае использована функция выделения левой подстроки (длиной один) из поля City запроса «Города из Мичигана». Результатом запроса будет город Ann Arbor.

Примеры усложненных запросов, включающих связанные таблицы, а также групповые операции, смотри в лабораторных работах №3, №4, посвященных созданию форм и отчетов.

3.3. Создание запросов с помощью мастера

Мастер запросов предназначен для создания простых запросов, включающих отбор полей. Вызов мастера производится из окна базы данных выбором соответствующего пункта. В результате появляется окно:

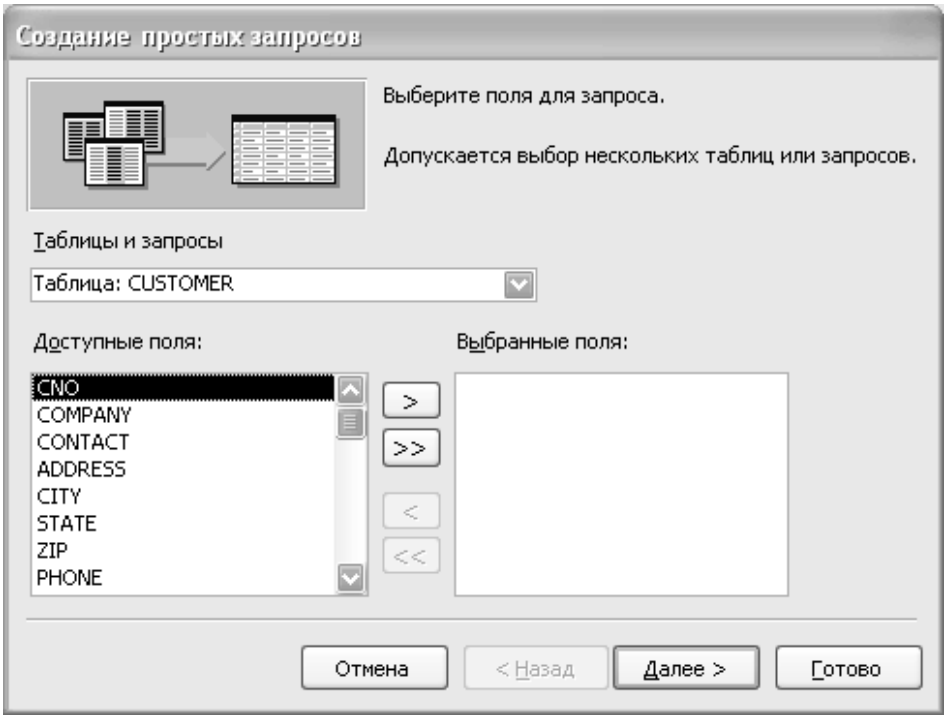
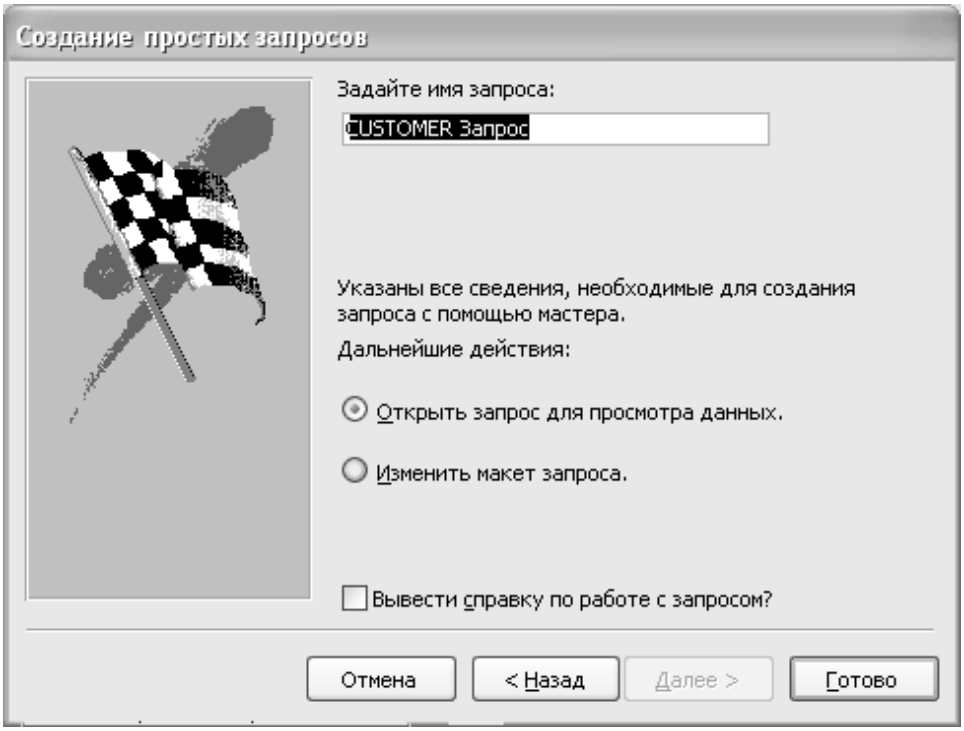


Рисунок 10 – Окно мастера запросов

После отбора таблиц и полей нажимаем кнопку «Далее» и в следующем окне:



переходим к просмотру данных в режиме таблицы (установлен переключатель «Открыть запрос для просмотра данных») либо к работе с конструктором запросов (установлен переключатель «Изменить макет запроса»). После завершения работы с мастером нажимаем кнопку «Готово». Как видим, в режиме

мастера могут быть созданы самые простые запросы, которые, как правило, необходимо модифицировать в режиме конструктора.

4. ПРОГРАММА РАБОТ

Создать запросы из таблицы CUSTOMER, каждый из которых сохранить под именами: «Запрос 1», «Запрос 2»,…, «Запрос 7» соответственно. Каждый из запросов просмотреть в режимах таблицы, конструктора и SQL. Проанализировать структуру SQL-оператора SELECT для каждого из запросов. Ниже приведены цели запросов.

1. Получить выборку полей «Company», «Contact», «Phone», «City», «State», «Ytdpurch» (от Yesterday purchase – вчерашние приобретения; имеются в виду приобретения за прошлый период).
2. То же, что запрос 1, но отсортированный по городам в алфавитном порядке.
3. То же, что запрос 1, но отсортированный вначале по штатам, а затем по городам в алфавитном порядке (замечание: порядок сортировки определяется порядком расположения полей в запросе).
4. То же, что запрос 3, но для городов, находящихся в штате Огайо (‘OH’).
5. То же, что запрос 3, но для городов, находящихся в штатах Огайо (‘OH’) или Мичиган (‘MI’) (замечания: здесь можно набрать в двух строках условий отбора «Like (“OH”)», «Like (“MI”)» соответственно либо одно условие «Like (“OH”) or Like(“MI”)»; попробуйте эти способы и сравните окна конструктора, сохранив оба варианта запроса; вместо оператора Like можно использовать In).
6. То же, что запрос 5, но только для тех покупателей, у кого объем прошлых приобретений больше 1000.
7. То же, что запрос 3, но для компаний, начинающихся с «Tr» (замечание: наряду со способом, примененным в запросе рис. 9, при сравнении с подстрокой можно использовать конструкцию вида «Tr*»).

5. ЗАЩИТА РАБОТЫ

Письменный отчет по работе не делается. Защита работы производится у компьютера в форме демонстрации результатов и отдельных операций, а также ответов на контрольные вопросы из следующего перечня (на вопрос модульного контроля в письменной форме):

1. Вопрос из блока №2 модульного контроля (в письменной форме)
2. Определение запроса
3. Правила создания запроса с помощью конструктора
4. Правила создания запроса с помощью мастера
5. Структура SQL-оператора SELECT
6. Сортировка; особенности сортировки по нескольким полям
7. Создание запросов без повторяющихся строк
8. Создание запросов с условиями отбора

ЛАБОРАТОРНАЯ РАБОТА № 3
РАЗРАБОТКА ЭКРАННЫХ ФОРМ

1. ЦЕЛЬ РАБОТЫ

Цель работы – изучить и практически освоить пользовательский интерфейс системы управления базами данных (СУБД) настольного типа Access, научиться создавать экранные формы Access.

2. ОБЪЕКТ ИССЛЕДОВАНИЯ

Объектом исследования является СУБД Access фирмы Microsoft. Инструментом исследования является программный пакет Access.

3. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

3.1. Понятие формы и средства разработки форм в пакете Access

Экранные формы или просто формы предназначены для удобного отображения таблиц (запросов) баз данных, а также навигации по таблицам. Под навигацией понимают все манипуляции, связанные с перемещением по таблице – «прыжки» к началу и концу, переход к предыдущей или следующей записи. В более сложных случаях с помощью форм производится любое интерактивное взаимодействие между конечным пользователем и базой данных. В этом случае на форму могут быть помещены командные кнопки, вычисляемые поля, списки, подчиненные формы, закладки и другие элементы современного интерфейса.

Для создания форм в Access предусмотрены два основных средства – конструктор и мастер. Как и при создании таблиц и запросов, мастер проще в использовании, но может применяться в стандартных случаях. В остальных случаях применяется конструктор. Имеются еще и автоформы, вызываемые из окна базы данных путем нажатия инструментальной кнопки «Создать». Автоформы предназначены для автоматизированного создания простейших форм, не связанных с каким-либо преобразованием информации БД.

3.2. Создание форм с помощью мастера

Для выхода в данный режим необходимо в окне базы данных установить световой маркер на опцию «Формы» и выбрать опцию «Создание формы с помощью мастера».

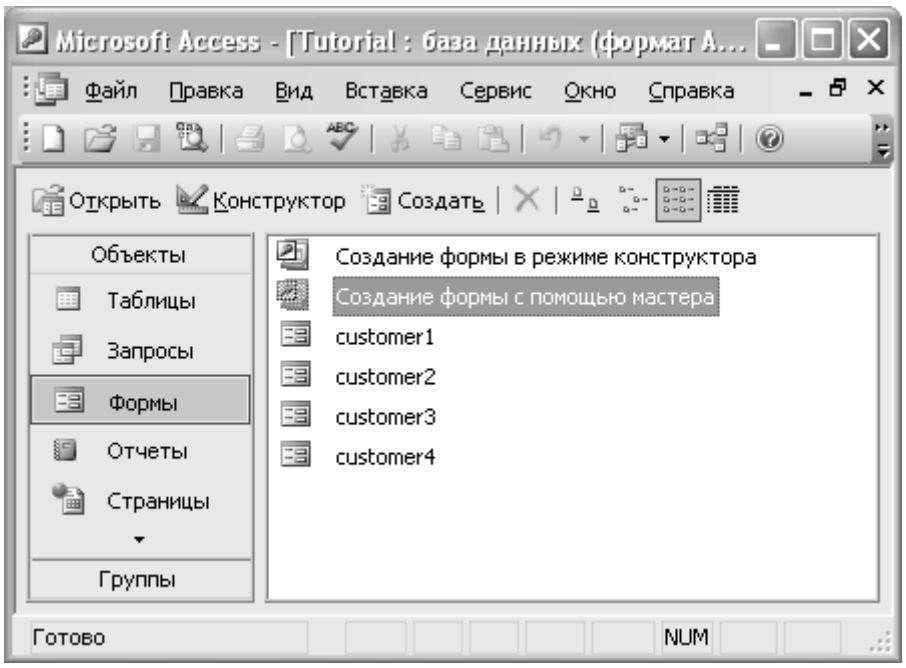


Рисунок 1 – Переход в режим создания формы

После выхода в режим открывается окно вида:

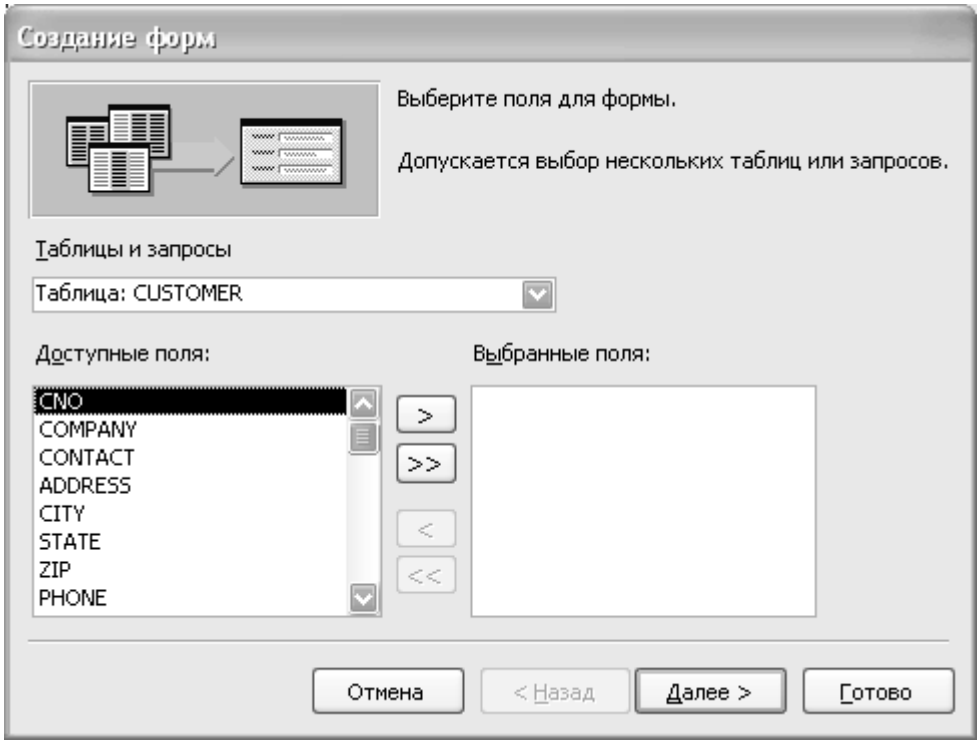


Рисунок 2 – Окно создания формы

Из этого окна (поле ввода «Таблицы и запросы») необходимо выбрать источник данных (записей), на основе которого будет создаваться форма. Далее с помощью кнопок со стрелками перемещаем в область «Выбранные поля» требуемые поля и нажимаем кнопку «Далее».

Следующее окно предназначено для выбора вида формы:

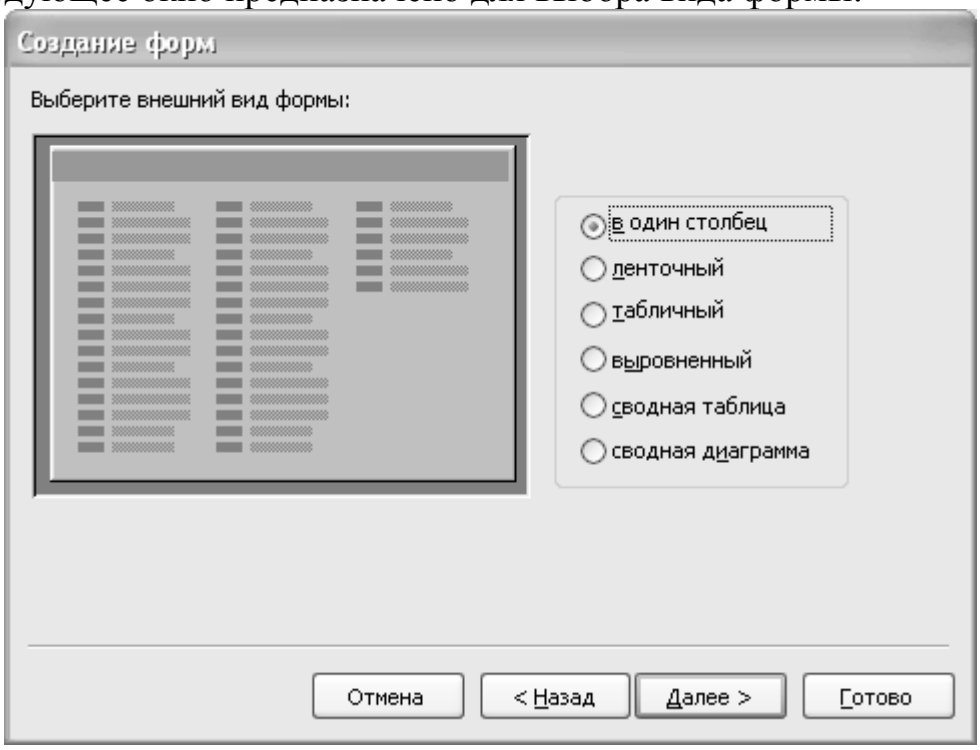


Рисунок 3 – Выбор вида формы

После выбора вида формы нажимаем кнопку «Далее» и переходим к окну:

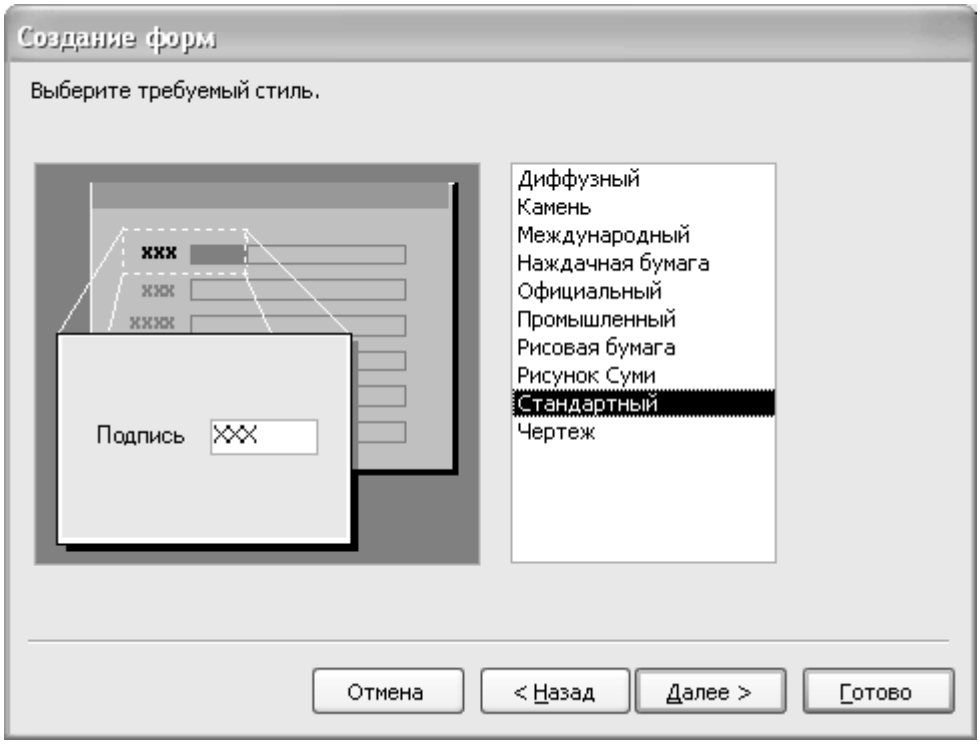


Рисунок 4 – Выбор стиля

В этом окне выбирается стиль оформления формы.

И, наконец, нажав последний раз кнопку «Далее», переходим к последнему окну мастера:

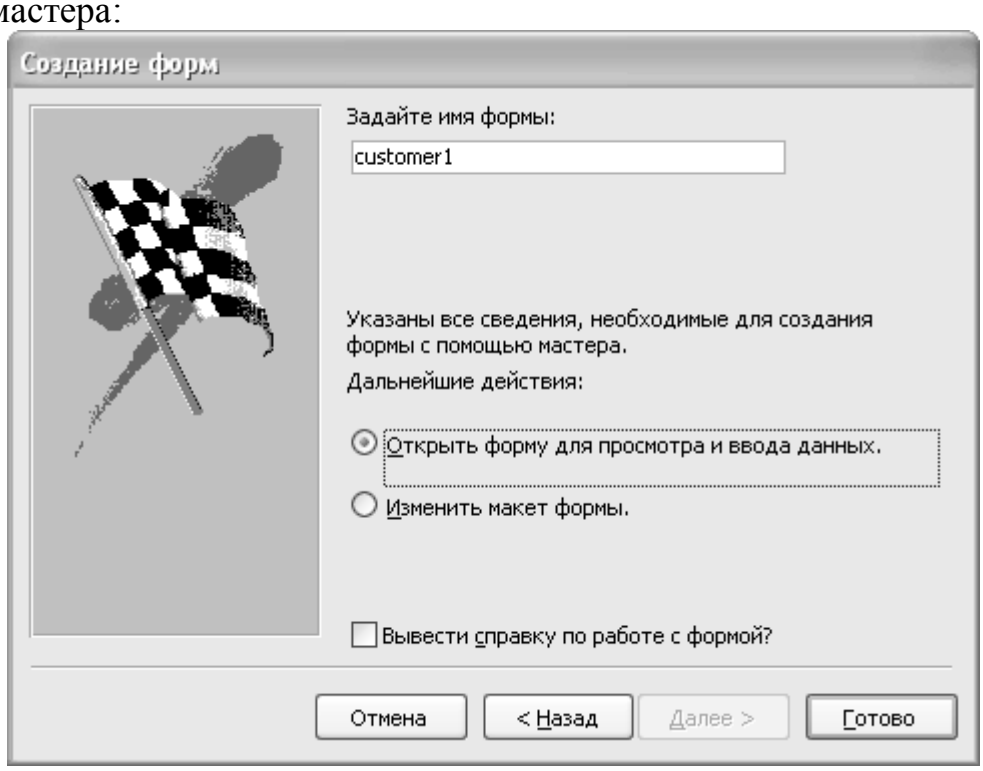


Рисунок 5 – Последнее окно мастера форм

В этом окне, прежде всего, задают имя формы, под которым она будет сохранена в базе данных. Кроме того, здесь же с помощью переключателя выбирается дальнейший режим работы с нею – просмотр и ввод данных либо переход к конструктору формы с целью ее модификации (после нажатия кнопки «Готово»).

Таким образом, построение формы с помощью мастера не требует особенных знаний и доступно любому конечному пользователю.

На рисунках 6 – 9 помещены несколько внешних видов формы для таблицы CUSTOMER (см. рисунок 3).

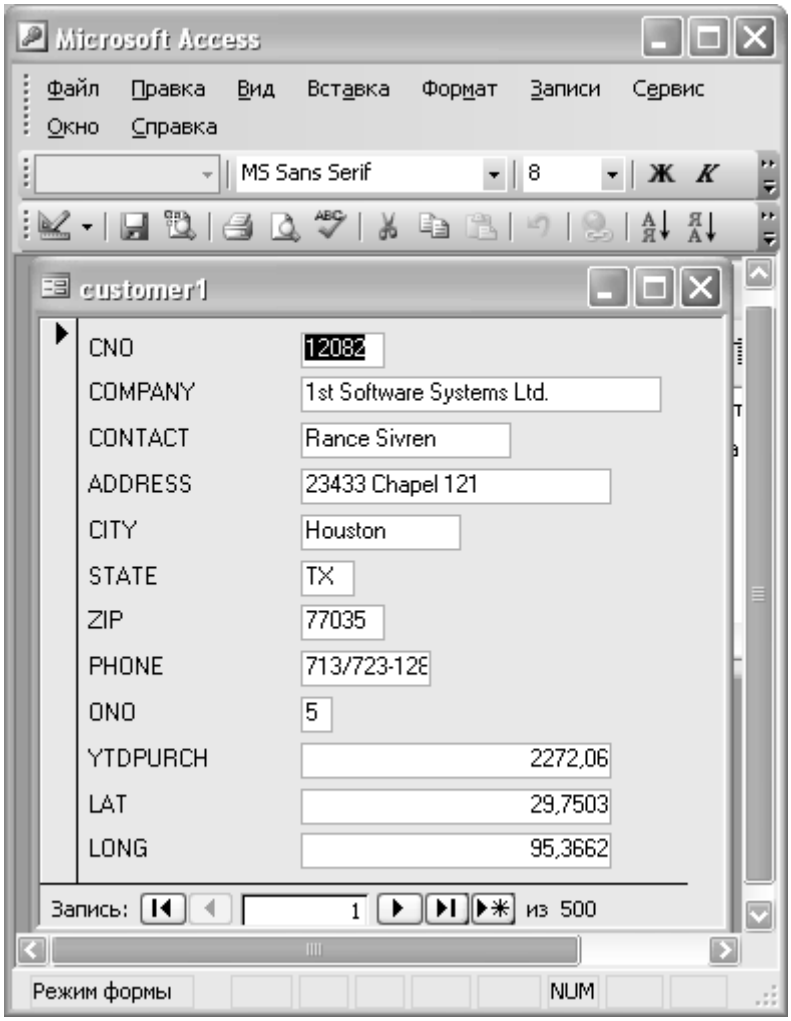


Рисунок 6 – Вид формы «в один столбец»

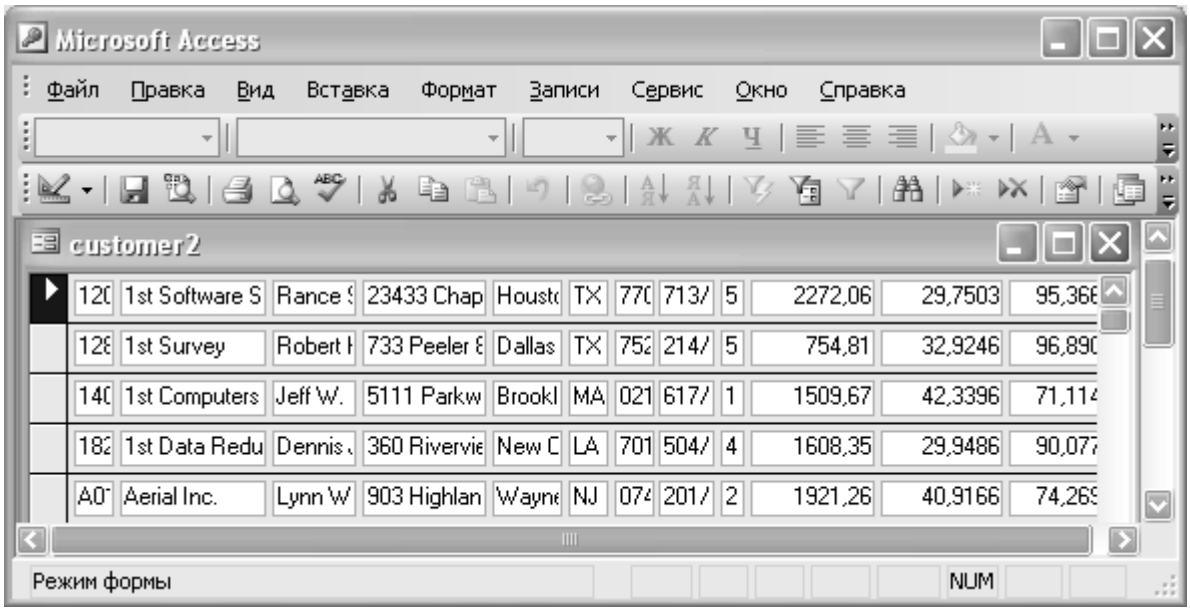


Рисунок 7 – Ленточный вид формы

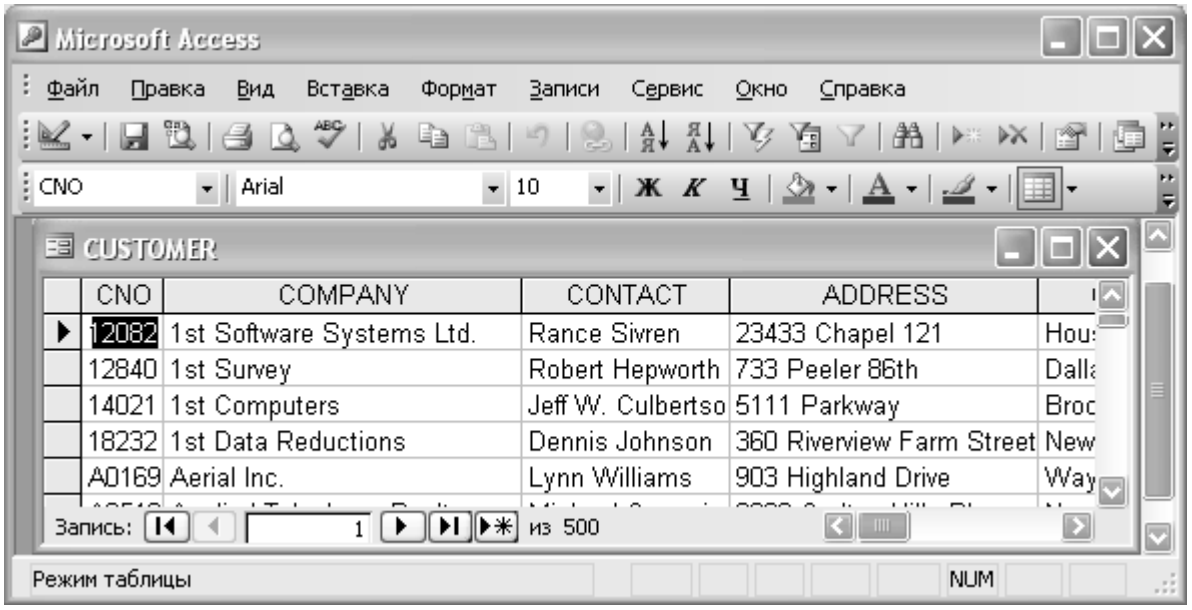


Рисунок 8 – Табличный вид формы

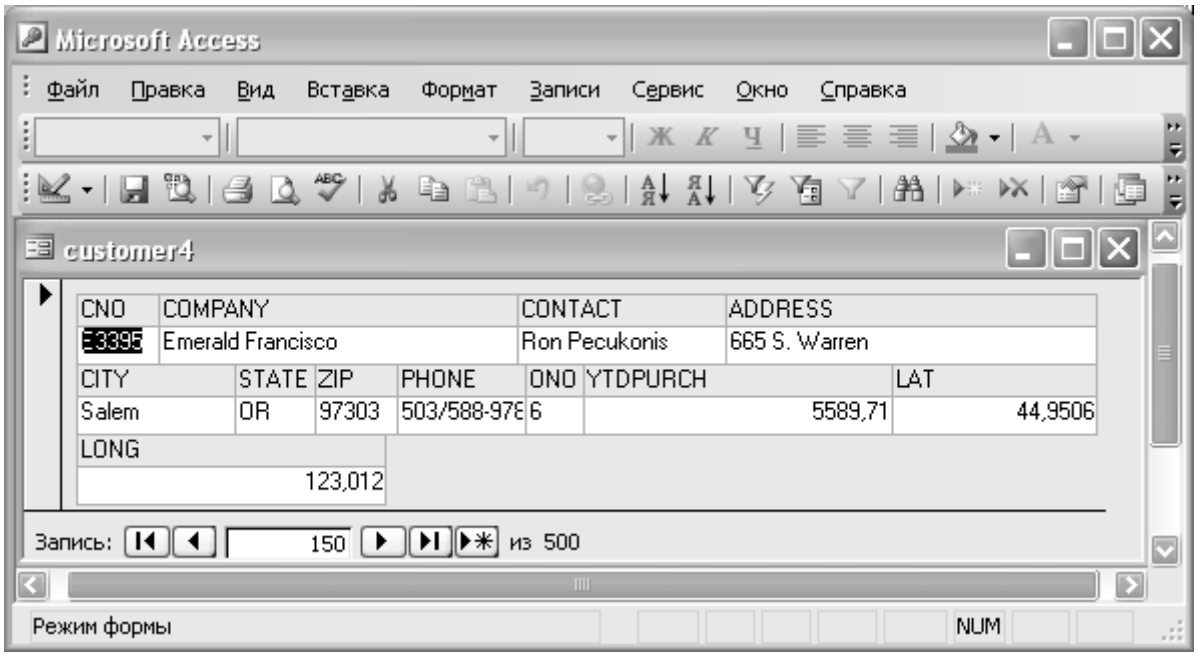


Рисунок 9 – Выровненный вид формы

3.3. Создание форм с помощью конструктора

Конструктор форм предназначен для создания сложных форм, включающих различные элементы управления (поля ввода, кнопки, списки, переключатели и др.), выбираемые из инструментальной панели, а также событий и обработчиков событий. Для грамотного конструирования форм необходимо знать, из каких частей она состоит.

Сведения в форме могут быть разбиты на разделы. Все формы содержат основной раздел – область данных. Этот раздел обычно содержит элементы управления, присоединенные к полям в источнике записей. Однако он может также содержать свободные элементы управления, такие как надписи, определяющие содержимое полей. Форма также может включать разделы заголовка формы, верхнего колонтитула, нижнего колонтитула, примечания формы. Верхний колонтитул используется для вывода названия, заголовков столбцов, даты или номера страницы сверху на каждой странице формы или отчета. В формах верхние колонтитулы отображаются только при печати. Нижний колонтитул используется для вывода итоговых значений по странице, даты или номера страницы снизу на каждой странице формы или отчета. В формах нижние колонтитулы отображаются только при печати. Каждый раздел формы имеет свое предназначение и печатается в определенном порядке.

При вызове конструктора форм выводится два окна и инструментальная панель, которую можно перетащить мышью в окно базы данных, как это сделано на рисунке 10 (нижняя инструментальная панель):

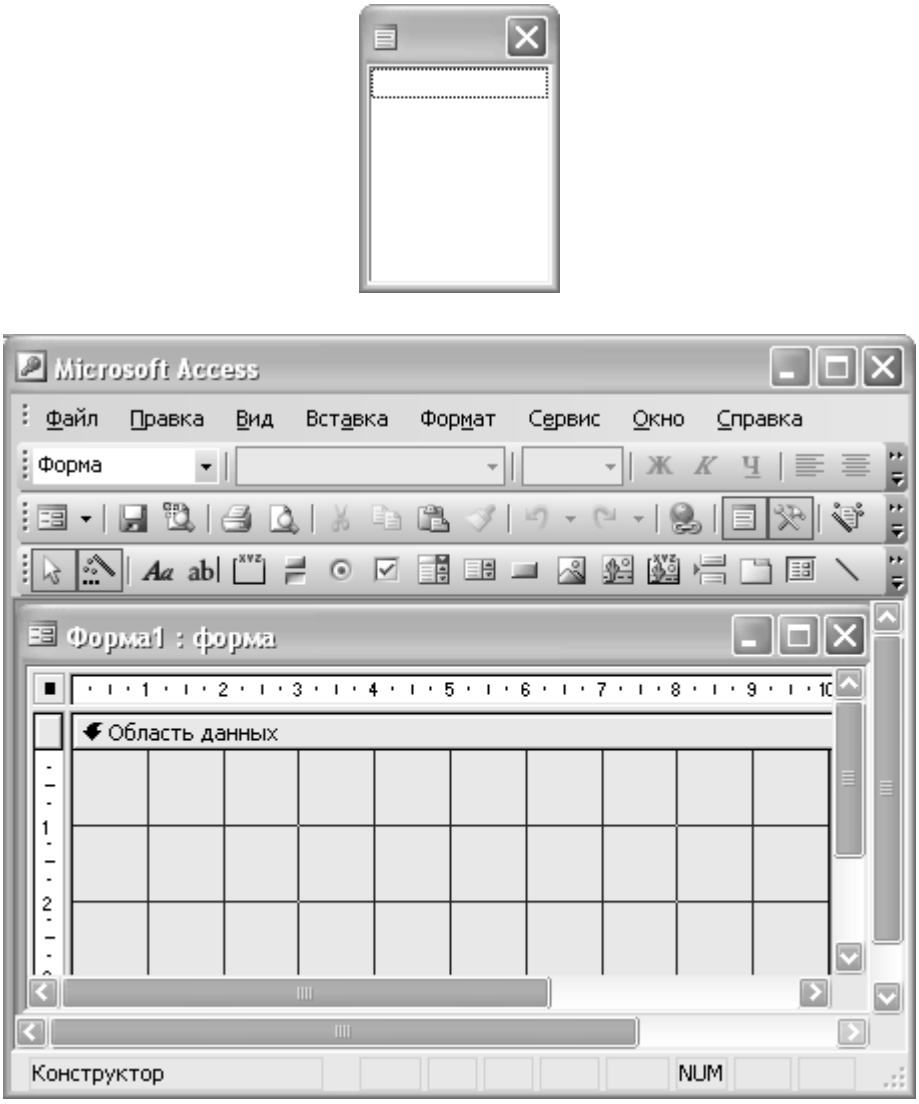


Рисунок 10 – Окна конструктора форм

На рисунке 10 по умолчанию загрузилась только область данных. При необходимости подключения других разделов следует войти в опцию «Вид» главного меню окна базы данных и установить флажки у соответствующих пунктов («Колонтитулы» и/или «Заголовок/примечание формы»). Верхнее окно на рисунке 10 предназначено для отображения полей подключенной таблицы или запроса. При открытии окна конструктора форм оно может и не появиться, однако обязательно появится при выборе таблицы или запроса, присоединенного к форме. Указанный выбор производится следующим образом: надвигаем курсор мыши на верхнюю часть формы (на рисунке 10 в ней расположен текст «Форма1: форма»); правой кнопкой мыши вызываем контекстное меню; выбираем пункт «Свойства» контекстного меню; в открывшемся окне переходим к закладке «Данные» и в нем выбираем источник записей – таблицу или запрос (см. рисунок 11).

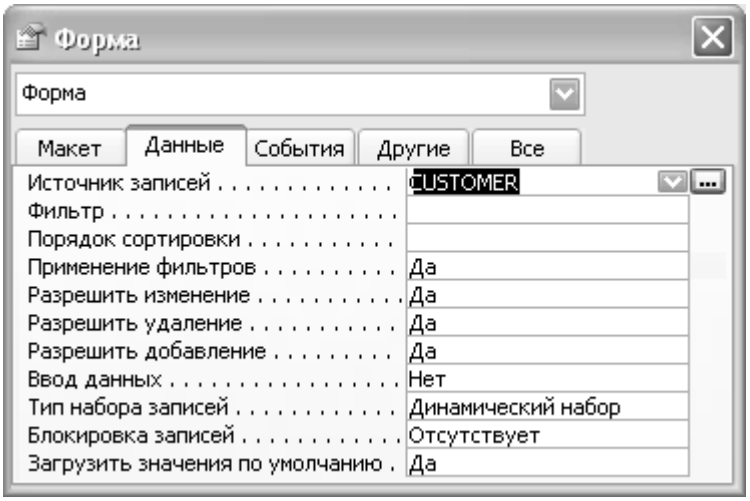


Рисунок 11 – Выбор присоединенной таблицы или запроса

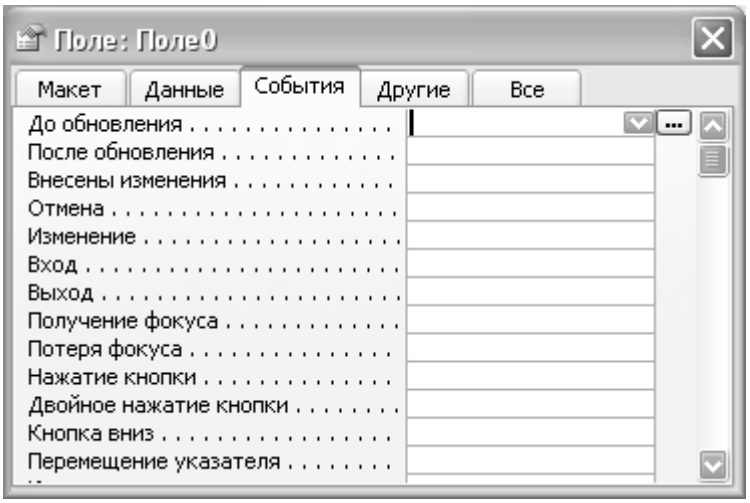
После выбора таблицы или запроса в указанном выше окне появятся соответствующие поля:

После выбора вновь появляется окно конструктора форм, с которым производится дальнейшая работа. Обязательными элементами формы являются элементы управления, обеспечивающие навигацию по таблице. Эти элементы видны на рисунках 6, 9 и представляют собой набор кнопок, поле для отображения номера записи и вспомогательной текстовой информации. Конструктор

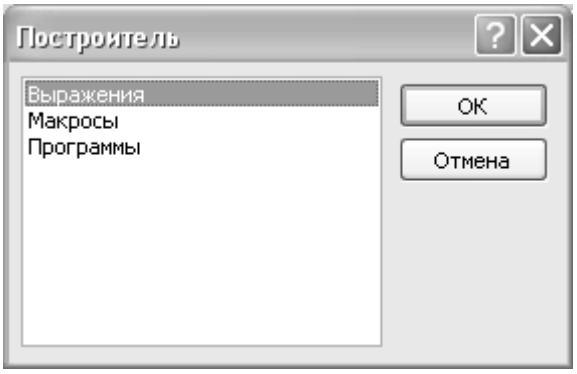
эти элементы помещает на форму автоматически. Это можно увидеть, перейдя из режима конструктора в режим формы (опция «Вид – Режим формы» меню окна конструктора формы на рисунке 10). Например, для построения формы вида «в один столбец», в раздел данных необходимо поместить 12 элементов управления типа «поле» (на инструментальной панели помечен кнопкой с символами «ab|»). Для переноса элемента на панель следует левой кнопкой мыши нажать на кнопку «ab|», затем кнопку отпустить, переместить курсор мыши на требуемую позицию области данных и вновь нажать левую кнопку мыши. В результате появится разделенное на две части поле – левая часть для его названия, а правая – для значения.

Дальнейшие действия над элементом управления «поле» следующие: надвигаем курсор мыши на левую половину, вызываем правой кнопкой мыши контекстное меню, выбираем в нем пункт «Свойства» и задаем свойству «Подпись» значение CNO; надвигаем курсор мыши на правую половину, вызываем правой кнопкой мыши контекстное меню, выбираем в нем пункт «Свойства», переходим к закладке «Данные» и выбираем из списка полей свойства «Данные» значение CNO. Аналогично формируются остальные поля формы рисунка 6. Если левая часть поля с названием не нужна, ее можно удалить, выделив мышью и нажав кнопку «Delete» на клавиатуре компьютера (либо выбором пункта «Вырезать» контекстного меню, вызываемого правой кнопкой мыши). Можно левое и правое поля располагать в разных местах области данных путем перетаскивания мышью, а также изменять их размеры.

Более сложные функции реализуются с помощью событий и обработчиков событий. Для некоторого элемента управления это делается с помощью выхода через его «Свойства» на закладку «События», где выбирается подходящее событие и в поле ввода справа нажатием кнопки с «галочкой» устанавливается «Процедура обработки» (если кнопки не видны, нужно щелкнуть правой кнопкой по полю):



После этого нажимаем кнопку с многоточием и переходим в окно выбора вида обработчика:



При выборе опции «Выражения» откроется построитель выражения, результат которого отобразится в редактируемом поле. При выборе опции «Макросы» отроется окно редактора макросов, а при выборе опции «Программы» – окно редактора Visual Basic. Результаты работы последних сохраняются в виде отдельных объектов базы данных – макросов и модулей.

4. ПРОГРАММА РАБОТ

8. Создать экранные формы для работы с таблицей CUSTOMER, пользуясь мастером форм, в видах «в один столбец», «ленточном», «табличном», «выровненном». Для отображения отобрать следующие поля и их порядок расположения: CNO, COMPANY, ADRESS, STATE, CITY, PHONE, YTDPURCH; каждую форму сохранить в базе данных под именами «форма1», «форма2», «форма3», «форма4».
9. Создать экранную форму для работы с таблицей INVOICES, пользуясь конструктором форм. В основу взять вид «в один столбец» и преобразовать его, заменив имена полей их содержательными заголовками на русском языке: INO – Номер счета; CNO – Номер покупателя; IDATE – Дата выписки счета; SALESMAN – Продавец. Форму сохранить под именем «форма5».
10. Создать запрос из связанных таблиц CUSTOMER, INVOICES, SALESMAN, аналогичный по структуре таблице INVOICES, но в которой значения поля INVOICES.CNO с кодами компаний заменены наименованиями компаний из таблицы CUSTOMER (CUSTOMER.COMPANY), а значения поля с кодами продавцов INVOICES.SALESMAN – соответствующими именами продавцов из SALESMAN.NAME. Запрос сохранить под именем «Счета». Условия соединения таблиц CUSTOMER, INVOICES, SALESMAN можно задать различными способами:
    - в виде логического выражения: «CUSTOMER.CNO=INVOICES.CNO and INVOICES.SALESMAN=SALESMAN.SALESMAN»);
    - в виде частей логического выражения «CUSTOMER.CNO=INVOICES.CNO», «INVOICES.SALESMAN=SALESMAN.SALESMAN», каждая из которых помещена в отдельную клетку строки условий отбора;

- в виде созданных дополнительных полей CUSTOMER.CNO, INVOICES.SALESMAN окна конструктора запросов, под которыми в строке условий отбора помещены соответственно INVOICES.CNO и SALESMAN.SALESMAN; при этом флажок печати у дополнительных полей снимается.

11. Создать табличный вид экранной формы на основе полученного запроса, заменив имена полей содержательными заголовками на русском языке (так, как в пункте 2 программы работ, где вместо CNO теперь будет Покупатель). Форму сохранить под именем «форма6».

5. ЗАЩИТА РАБОТЫ

Письменный отчет по работе не делается. Защита работы производится у компьютера в форме демонстрации результатов и отдельных операций, а также ответов на контрольные вопросы из следующего перечня (на вопрос модульного контроля в письменной форме):

1. Вопрос из блока №3 модульного контроля (в письменной форме)
2. Назначение форм
3. Правила создания форм с помощью мастера
4. Виды форм «в один столбец», «ленточный», «табличный», «выровненный»
5. Правила создания форм с помощью конструктора: присоединение таблиц, работа с элементами управления
6. Особенности создания запросов из связанных таблиц

ЛАБОРАТОРНАЯ РАБОТА № 4
ФОРМИРОВАНИЕ ОТЧЕТОВ

1. ЦЕЛЬ РАБОТЫ

Цель работы – изучить и практически освоить пользовательский интерфейс системы управления базами данных (СУБД) настольного типа Access, научиться формировать отчеты Access.

2. ОБЪЕКТ ИССЛЕДОВАНИЯ

Объектом исследования является СУБД Access фирмы Microsoft. Инструментом исследования является программный пакет Access.

3. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

3.1. Понятие отчета и средства формирования отчетов в пакете Access

Отчеты предназначены для эффективного представления данных в печатном формате. Между экранными формами и отчетами имеется много общего. Различие же в том, что формы ориентированы на экранное представление данных, а отчеты – на печатное. Для печатных форматов характерно представление данных в преобразованном (по сравнению с БД) виде. Под преобразованным видом здесь понимается следующее: наличие характерных для отчетов верхних и нижних колонтитулов (т.е. предваряющей и заключительной информации); наличие вычисляемых полей для хранения промежуточных и заключительных итогов по данным числового типа; возможное наличие иллюстраций. Имеются и другие отличия форм и отчетов. Тем не менее, указанные отличия не являются принципиальными, поскольку в формы можно ввести все указанные выше элементы. Основное отличие проявляется в автоматизированных средствах создания форм и отчетов (автоформ и автоотчетов, мастеров форм и отчетов). Что же касается конструкторов форм и отчетов, то это очень похожие инструментальные средства, различающиеся тем, что в формах осуществляется навигация по присоединенной таблице или запросу, а в отчете – по экранному представлению страниц отчета. Кроме того, из меню конструктора форм можно без выхода из конструктора просмотреть форму в табличном виде, а из меню конструктора отчетов этого сделать нельзя (нужно предварительно выйти из конструктора и затем открыть полученный отчет).

Для создания отчетов в Access предусмотрены два основных средства – конструктор и мастер. Как и при создании таблиц, запросов и форм, мастер проще в использовании, но может применяться в стандартных случаях. В ос-

тальных случаях применяется конструктор. Имеются еще и автоотчеты, вызываемые из окна базы данных путем нажатия инструментальной кнопки «Создать». Автоотчеты предназначены для автоматизированного создания простейших отчетов, не связанных с каким-либо преобразованием информации БД.

Перед формированием отчета необходимо задать источник данных – таблицу или запрос. Для примера создадим запрос из связанных таблиц CUSTOMER и INVOICES, оператор SELECT которого имеет вид:

```
SELECT CUSTOMER.COMPANY, CUSTOMER.PHONE,
CUSTOMER.CONTACT, INVOICES.IDATE, INVOICES.ITOTAL
FROM CUSTOMER, INVOICES
WHERE (((INVOICES.CNO)=[customer].[cno]))
ORDER BY CUSTOMER.COMPANY, INVOICES.IDATE;
```

Цель данного запроса – получить информацию обо всех компаниях-покупателях, сделавших заказы. При этом информация должна быть отсортирована по компаниям, а внутри каждой компании – по датам. Запрос сохраним под именем «Laba4».

3.2. Создание отчетов с помощью мастера

Для выхода в данный режим необходимо в окне базы данных установить световой маркер на опцию «Отчеты» и выбрать опцию «Создание отчета с помощью мастера».

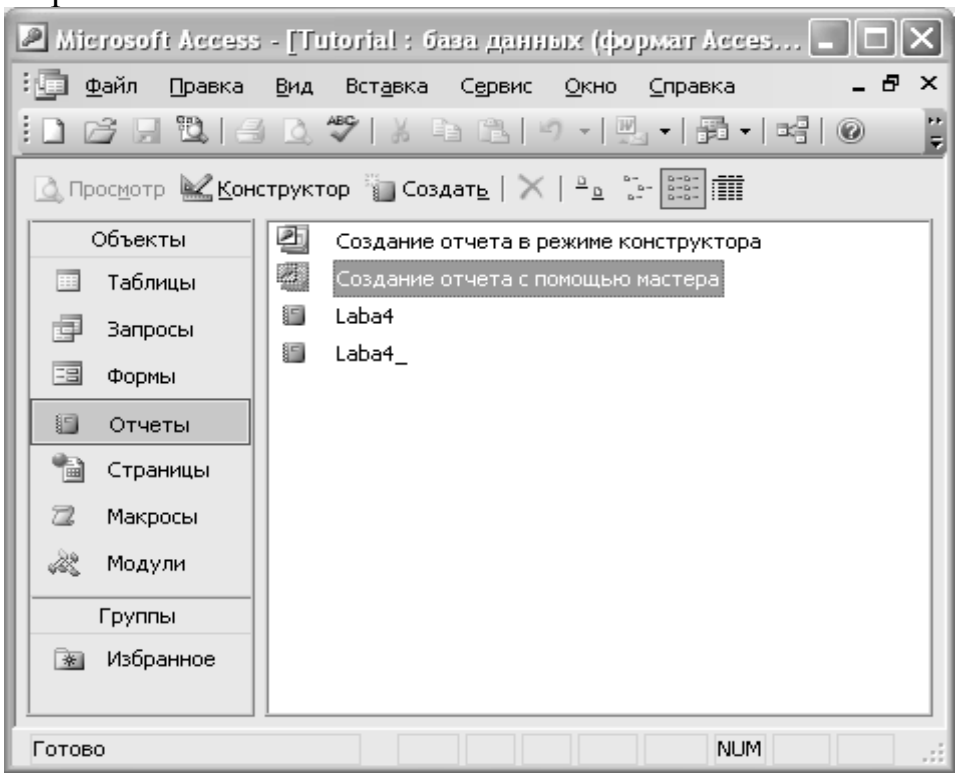


Рисунок 1 – Переход в режим создания отчета

После выхода в режим открывается окно вида:

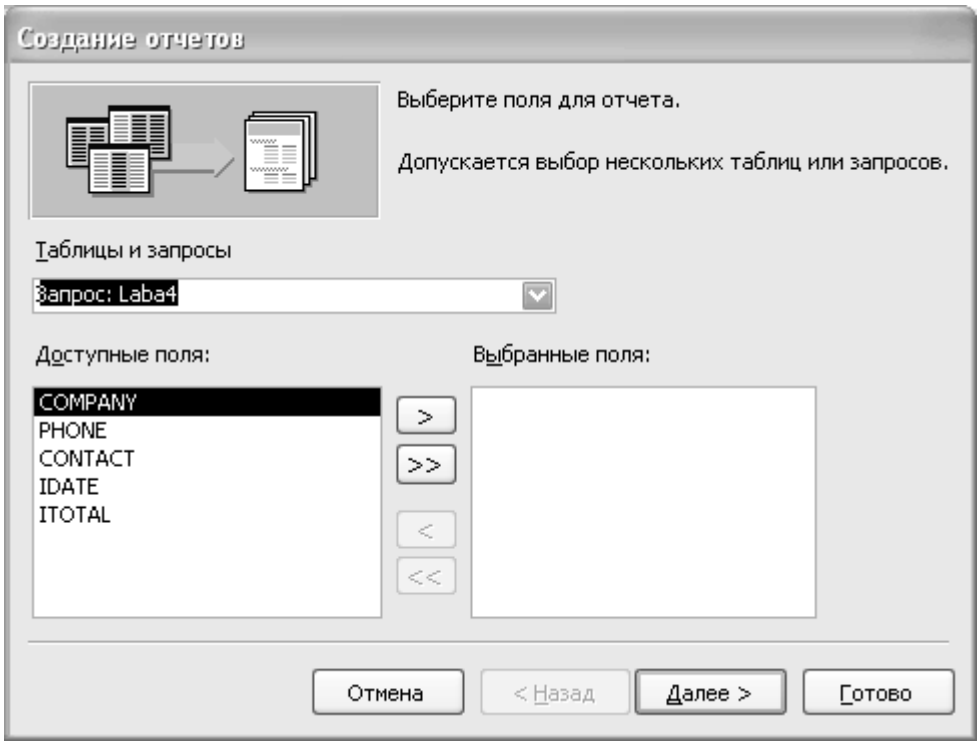


Рисунок 2 – Окно создания отчета

Из этого окна (поле ввода «Таблицы и запросы») необходимо выбрать источник данных (записей), на основе которого будет создаваться отчет. Далее с помощью кнопок со стрелками перемещаем в область «Выбранные поля» требуемые поля и нажимаем кнопку «Далее».

Следующее окно предназначено для выбора вида представления данных:

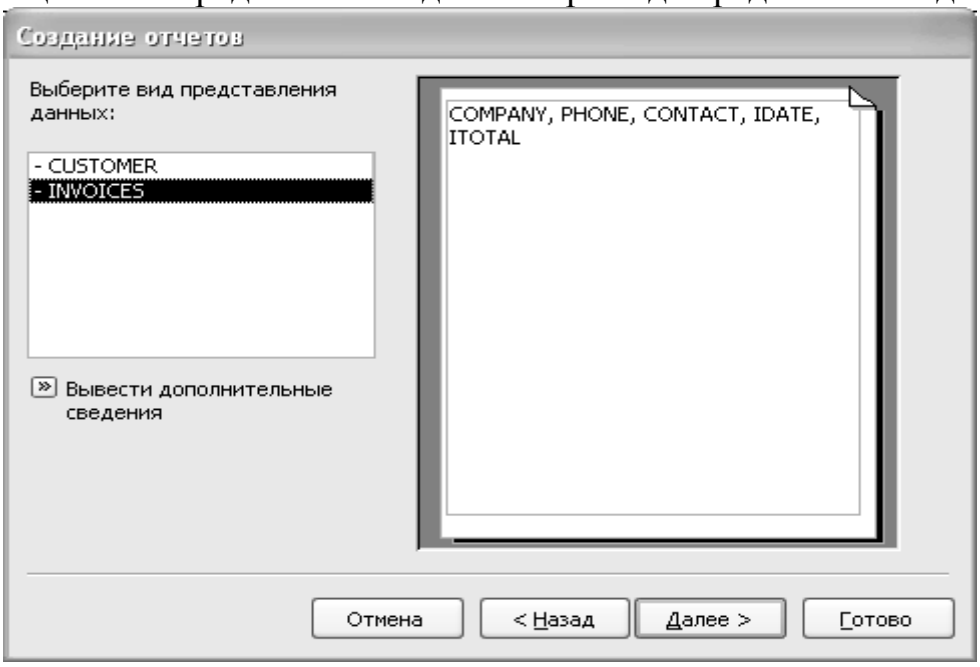


Рисунок 3 – Выбор вида представления данных

Мастер проверяет связи между выбранными таблицами, и в зависимости от выбранного представления группирует записи в отчете на основе этих связей. Пользователь имеет возможность сгруппировать данные или отказаться от группировки.

Мы здесь впервые сталкиваемся с понятием «группировка», которое означает такое перемещение строк источника данных, которое удобно для пользователя. Например, для данных запроса из рисунка 3 можно сгруппировать данные по компаниям (т.е. подряд помещаются данные вначале для одной компании, затем другой и т.д.), по датам. А, например, данные по ITOTAL (сумме заказа) группировать бессмысленно, так как вряд ли встретятся несколько заказов на одну и ту же сумму (с точностью до цента). Группировка данных важна не только для удобства восприятия информации, но и для получения итогов по группе данных (например, сумме заказов каждой из компаний). После выбора вида формы нажимаем кнопку «Далее» и переходим к окну группировки данных:

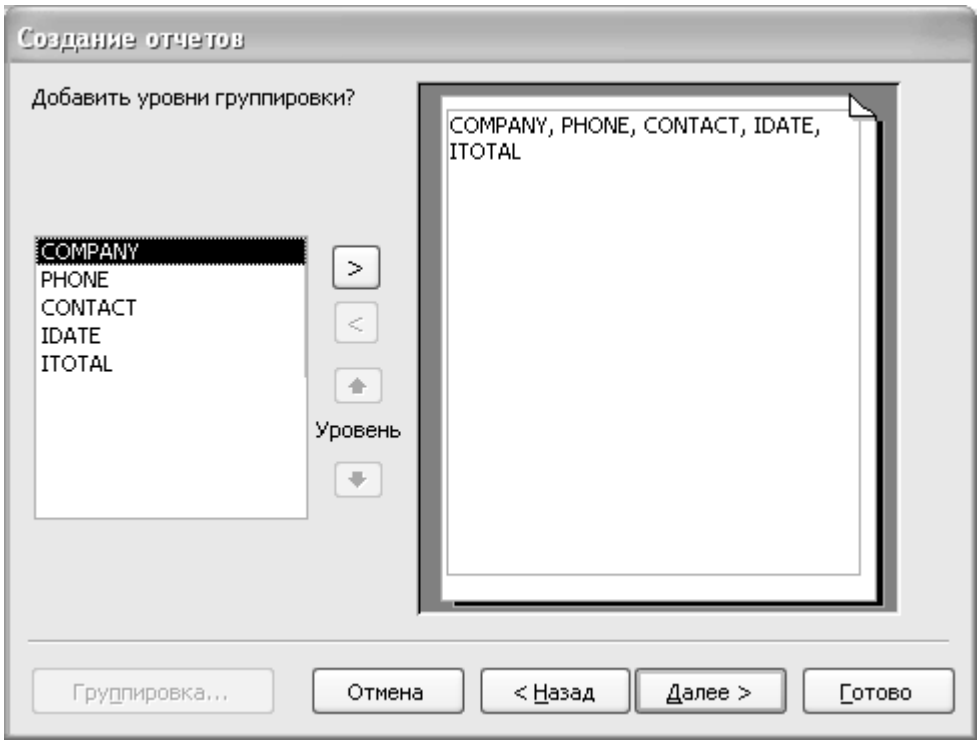


Рисунок 4 – Окно группировки данных

Предположим, что нам нужна группировка по компаниям. В этом случае выделяем COMPANY и нажимаем кнопку «>». В результате получается окно:

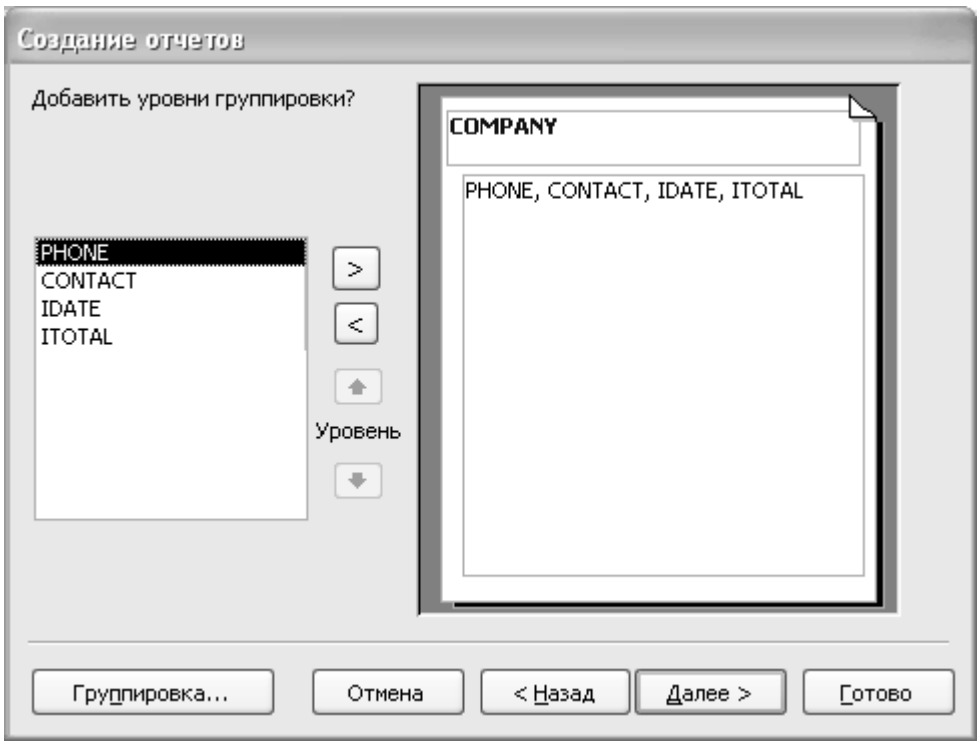


Рисунок 5 – Окно группировки с выбранным полем

Если для группировки выбрать еще одно поле, то получится случай вложенной группировки. Тогда необходимо указать, какая группировка в какую вложена. По умолчанию вложенность определяется порядком выбора полей и переноса их вправо. Если же необходимо переопределить уровни вложенности, пользуются инструментальными кнопками «стрелка вверх» и «стрелка вниз» (см. рисунок 5).

Нажав кнопку «Далее», переходим к окну сортировки и итогов:

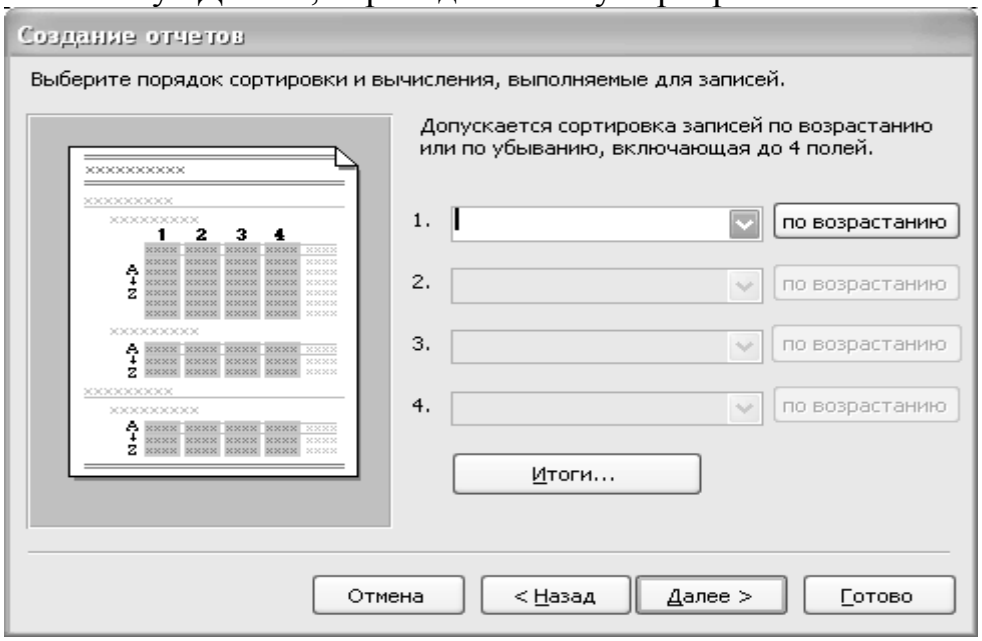


Рисунок 6 – Окно сортировки и итогов

В этом окне задают признаки сортировки, а также итоги по полям числового типа, если они нужны. Итоги вычисляются с помощью набора агрегатных функций (функций над колонками таблицы). При этом, если введены группировки, итоги рассчитываются не только по всей колонке, но и по группам. Например, в рассмотренном выше примере при группировке по компаниям будут выведены не только окончательные итоги, но и итоги для каждой компании. Предусмотрены четыре агрегатные функции sum, avg, min, max, подсчитывающих соответственно сумму, среднее, минимальное и максимальное значения числового поля. Пусть, например, требуется получить средний объем заказа по компании. Нажав кнопку «Итоги…», открываем окно выбора агрегатных функций:

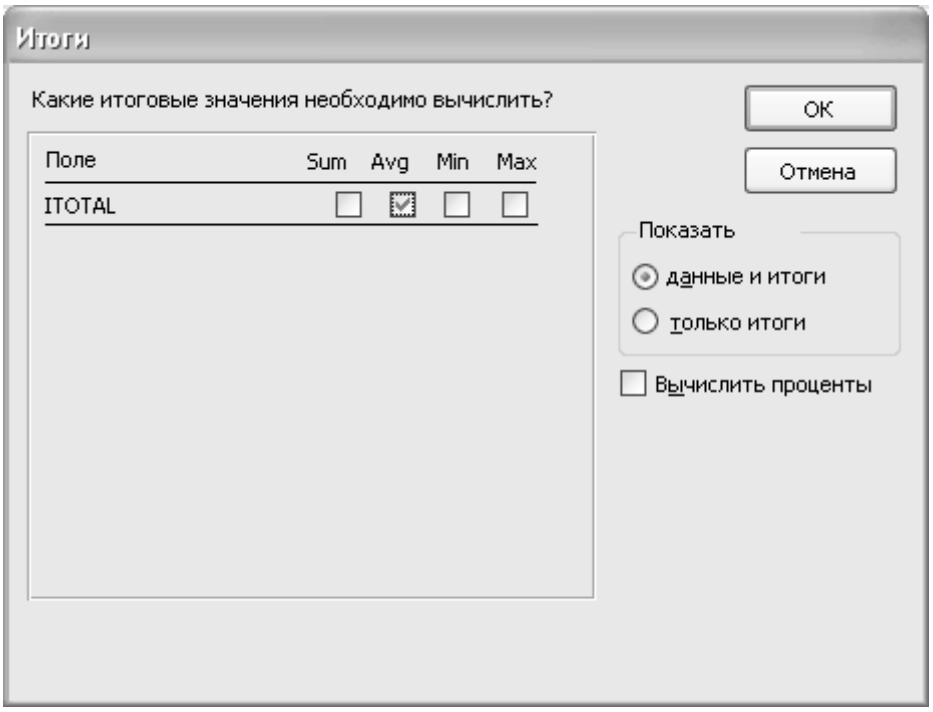


Рисунок 7 – Окно выбора агрегатных функций

Заметим, что в окне рисунка 7 будут выведены только числовые поля, против каждого из которых будет по четыре флажка. Если же числовых полей в источнике записей нет, кнопки «Итоги…» на форме рисунка 6 не будет.

После выбора функций нажимаем инструментальную кнопку «ОК» и возвращаемся в окно сортировки и итогов, откуда посредством кнопки «Далее» переходим к окну выбора макета:

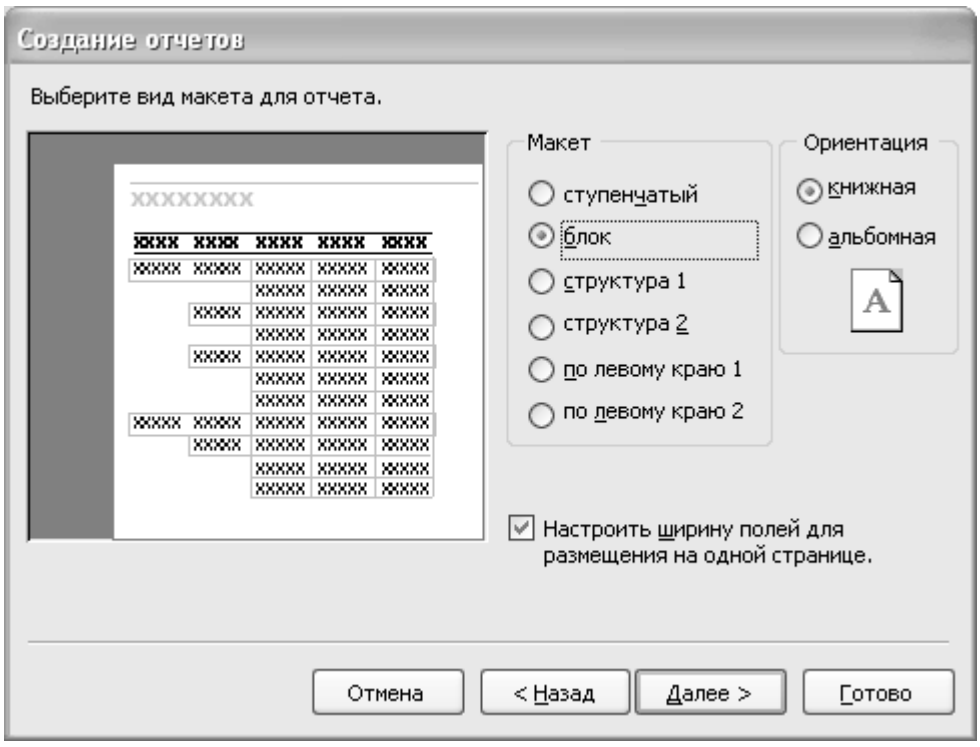


Рисунок 8 – Окно выбора макета

Выбрав макет и нажав кнопку «Далее», переходим к окну выбора стиля (такому же, как у мастера форм):

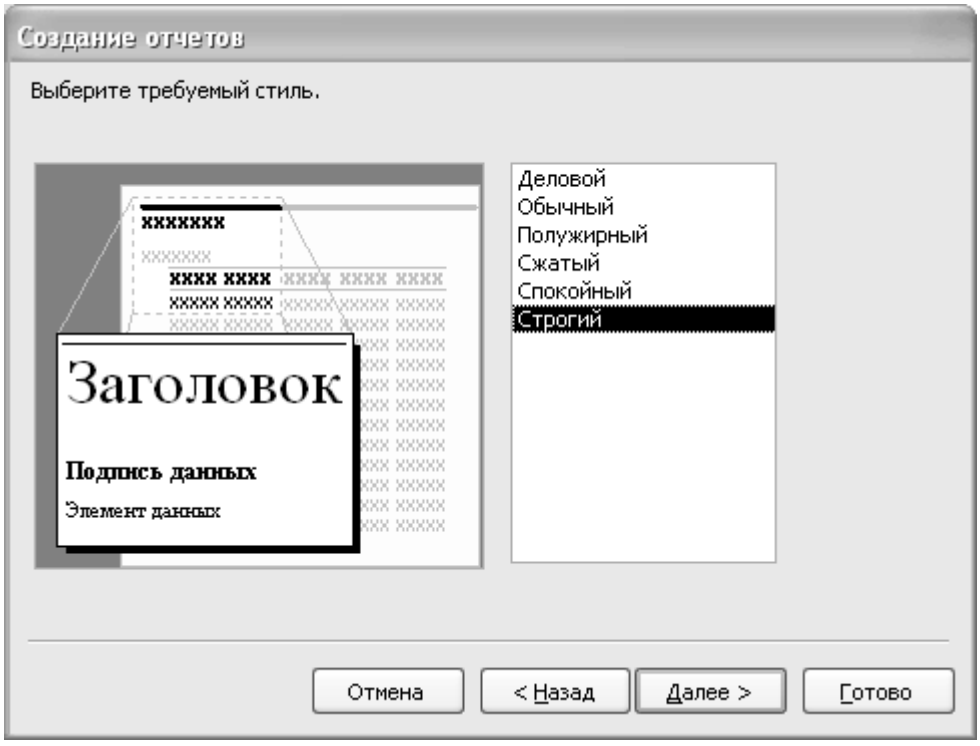


Рисунок 9 – Окно выбора стиля

После выбора стиля еще раз нажимаем кнопку «Далее» и переходим к последнему окну мастера отчетов:

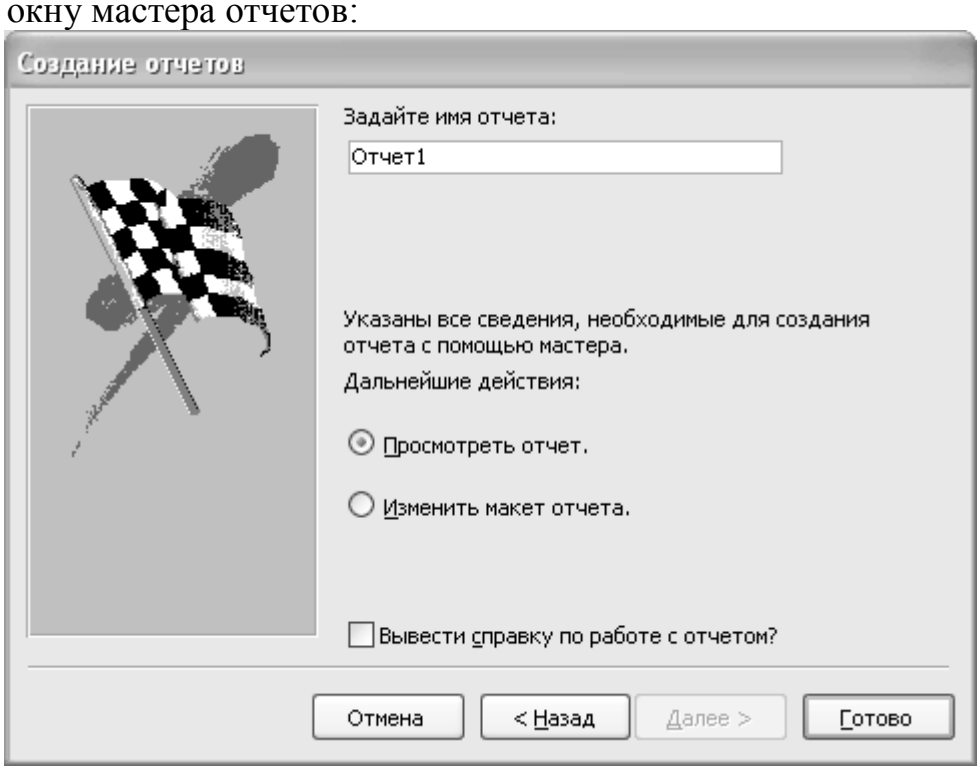


Рисунок 10 – Последнее окно мастера отчетов

В этом окне, прежде всего, задают имя отчета, под которым он будет сохранен в базе данных. Кроме того, здесь же с помощью переключателя выбирается дальнейший режим работы с ним – просмотр отчета либо переход к конструктору отчета с целью его модификации (после нажатия кнопки «Готово»).

Таким образом, построение отчета с помощью мастера не требует особенных знаний и доступно любому конечному пользователю.

На рисунке 11 помещена часть первой страницы экранного представления отчета, построенного с помощью мастера отчетов (всего в отчете 12 страниц). Отчет может быть распечатан либо экспортирован с возможной потерей отдельных элементов форматирования (опции «Файл – Печать», «Файл – Экспорт» главного меню Access).

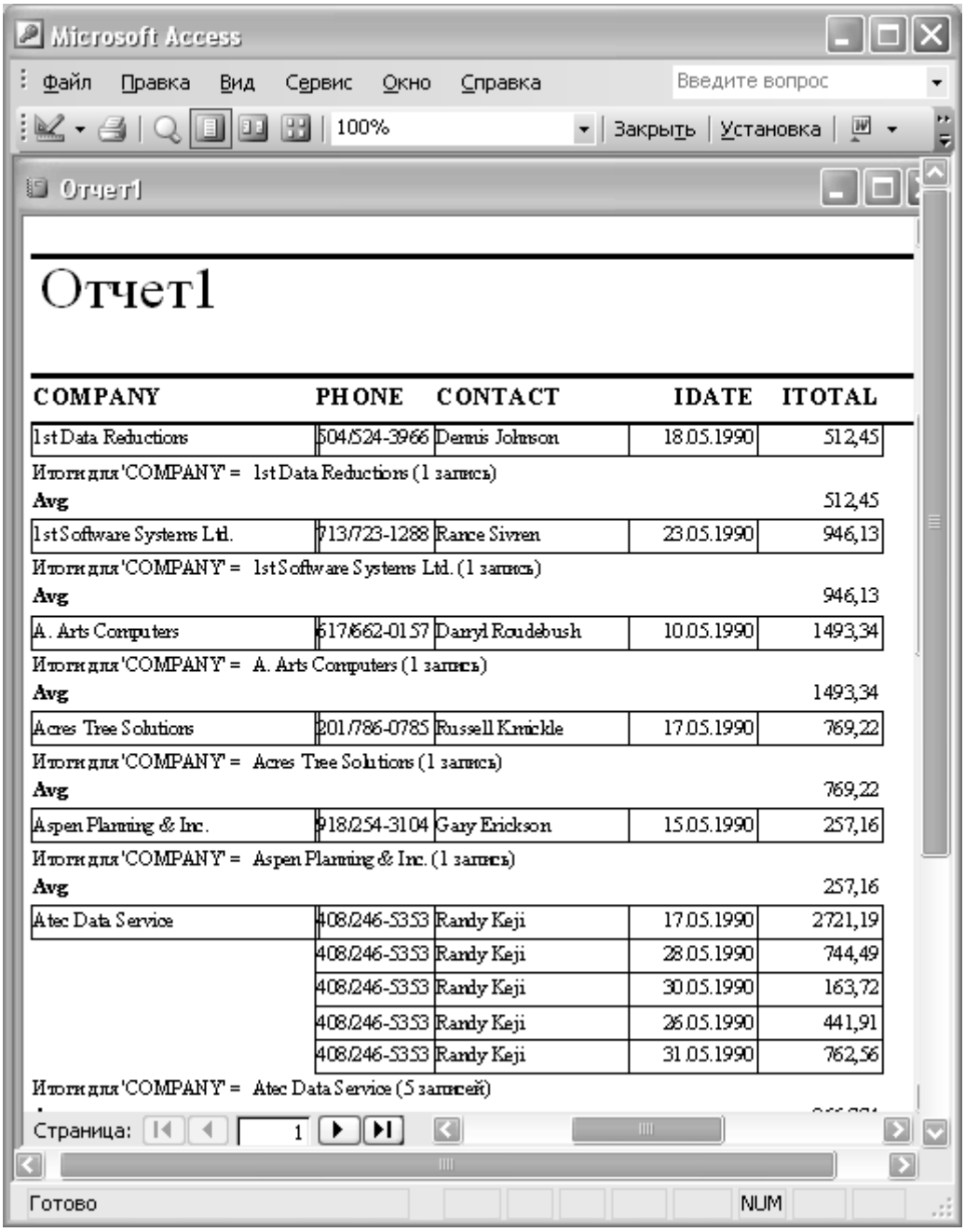


Рисунок 11 – Пример экранного представления отчета

3.3. Создание отчетов с помощью конструктора

Конструктор отчетов предназначен для создания нестандартных отчетов, отражающих индивидуальные требования потребителя, и может включать различные элементы управления (поля ввода, кнопки, списки, переключатели и др.), выбираемые из инструментальной панели, а также события и обработчики событий. Для грамотного конструирования отчетов необходимо знать, из каких частей он состоит.

Так же, как и в формах [4], сведения в отчете разбиты на разделы. Но в отличие от форм, содержащих, как правило, только область данных, отчеты практически всегда содержат верхний и нижний колонтитулы, а иногда и раз-

делы заголовка и примечаний. Область данных содержит элементы управления, присоединенные к полям в источнике записей. Однако он может также содержать свободные элементы управления, такие как надписи, определяющие содержимое полей. Верхний колонтитул используется для вывода заголовков столбцов, даты или номера страницы сверху на каждой странице отчета. Нижний колонтитул используется для вывода итоговых значений по странице, даты или номера страницы снизу на каждой странице отчета. Заголовок обычно содержит наименование отчета, а примечание – вспомогательную информацию, помещаемую в конце отчета. Заголовок и примечание, в отличие от колонтитулов, выводятся однажды – соответственно в начале и конце отчета. Колонтитулы же выводятся на каждой странице отчета.

При вызове конструктора отчетов появляются два окна и инструментальная панель, которую можно перетащить мышью в окно базы данных, как это сделано на рисунке 10 (нижняя инструментальная панель):

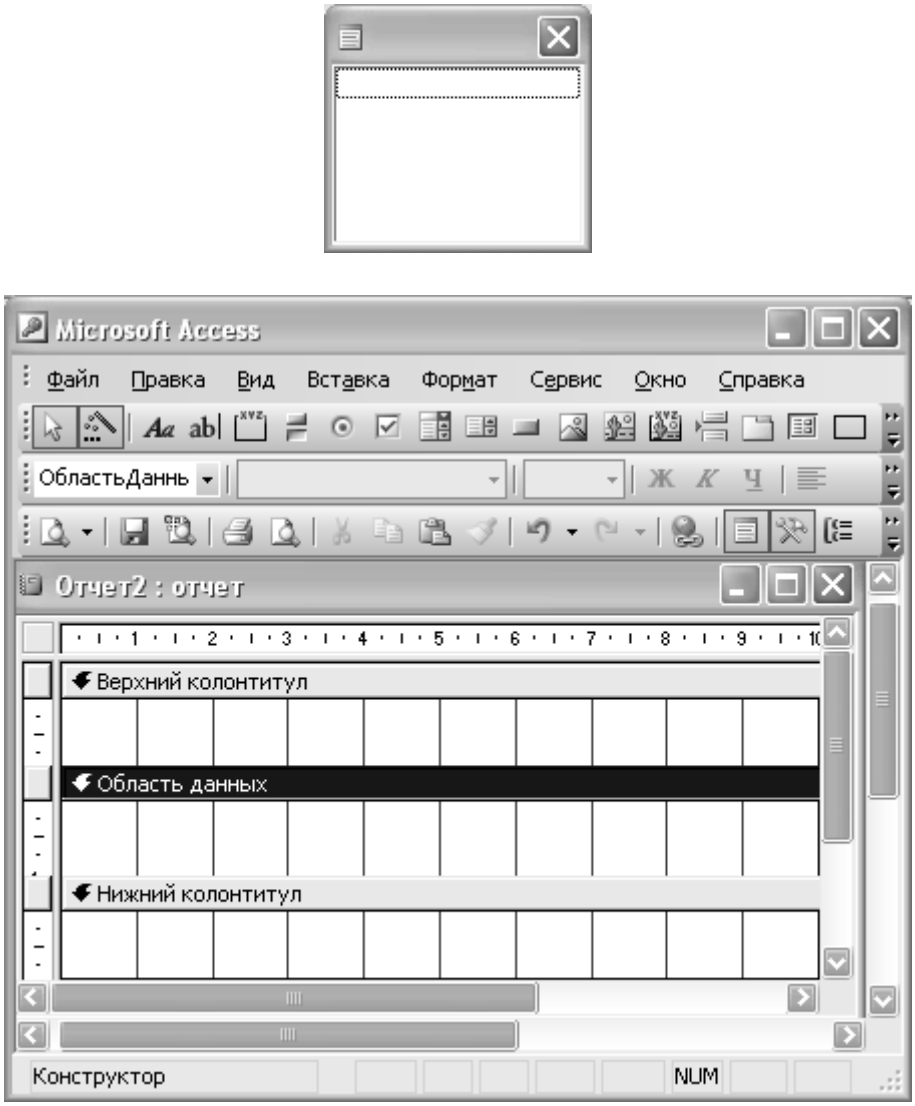


Рисунок 12 – Окна конструктора отчетов

На рисунке 12 по умолчанию загрузились область данных и колонтитулы. При необходимости подключения разделов «Заголовок» и «Примечание» следует войти в опцию «Вид» главного меню окна базы данных и установить флажок у пункта «Заголовок/примечание отчета». Верхнее окно на рисунке 12 предназначено для отображения полей подключенной таблицы или запроса. При открытии окна конструктора отчетов оно может и не появиться, однако обязательно появится при выборе таблицы или запроса, присоединенного к отчету. Указанный выбор производится следующим образом: надвигаем курсор мыши на верхнюю часть формы (на рисунке 12 в ней расположен текст «Отчет2: отчет»); правой кнопкой мыши вызываем контекстное меню; выбираем пункт «Свойства» контекстного меню; в открывшемся окне переходим к закладке «Данные» и в нем выбираем источник записей – таблицу или запрос (см. рисунок 13).

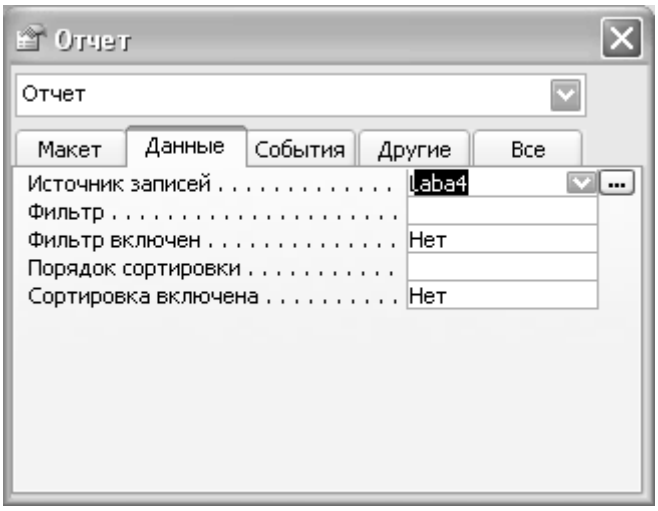


Рисунок 13 – Выбор присоединенной таблицы или запроса

После выбора таблицы или запроса в указанном выше окне появятся соответствующие поля:

После выбора вновь появляется окно конструктора отчетов, с которым производится дальнейшая работа. Обязательными элементами отчета являются элементы управления, обеспечивающие навигацию по экранному представлению отчета. Эти элементы видны на рисунке 11 и представляют собой набор кнопок, поле для отображения номера страницы и вспомогательной текстовой информации. Конструктор эти элементы помещает в отчет автоматически.

Отчет для примера, сформированного мастером отчетов (см. рисунок 11), имеет следующее окно конструктора отчетов:

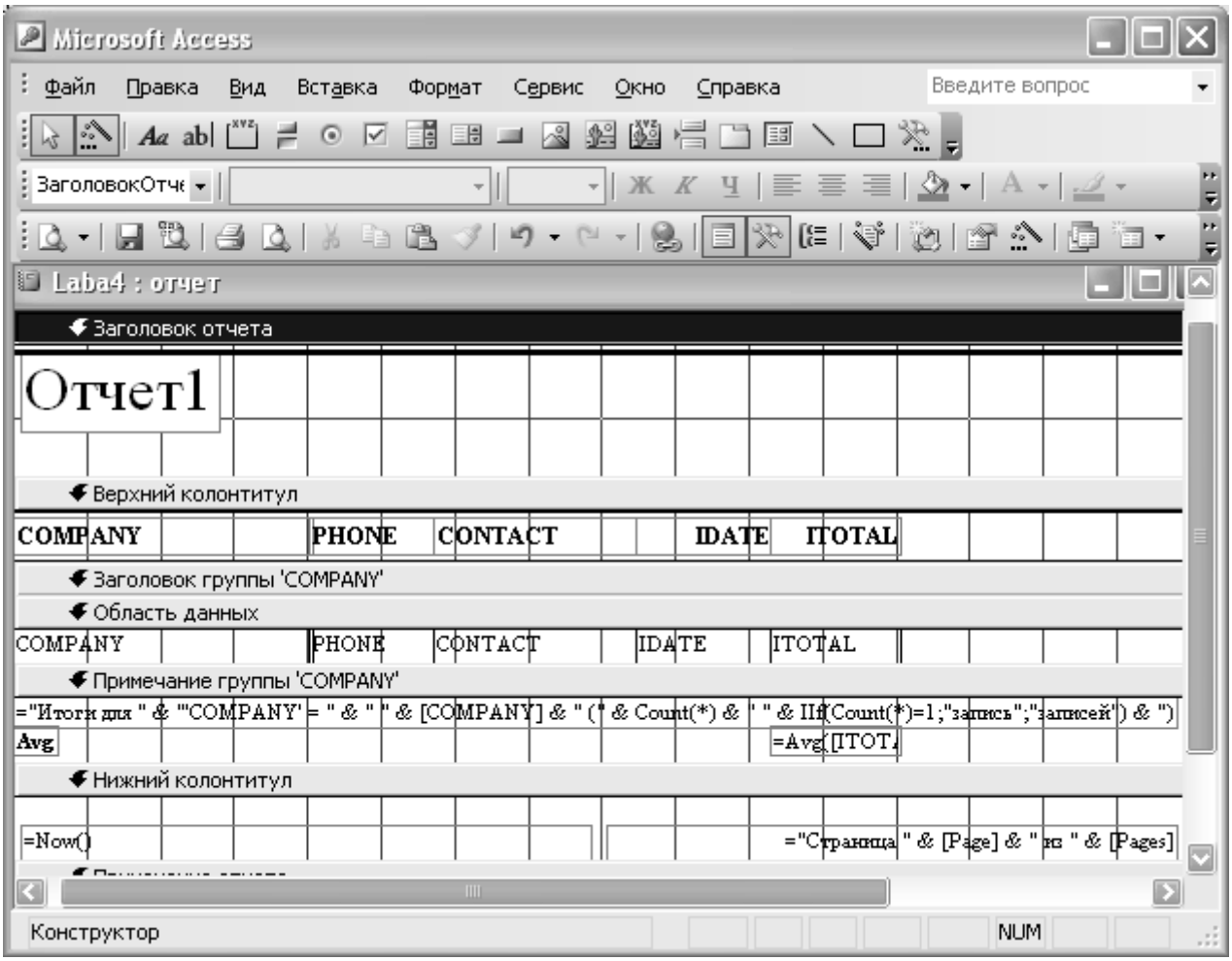


Рисунок 14 – Пример сформированного окна конструктора отчетов

Раскрывая поочередно правой кнопкой мыши на элементах управления контекстные меню и выбирая в них пункт «Свойства», можно установить типы и свойства каждого элемента. Как видно, конструктор внес в раздел «Заголовок отчета» элемент управления типа «Надпись» со значением свойства «Подпись», равному «Отчет1». В верхний колонтитул помещены левые части элементов типа «Поле» с названиями полей. В область данных помещены правые части элементов типа «Поле» с данными. Разделы «Заголовок группы ‘COMPANY’», «Примечание группы ‘COMPANY’» внесены автоматически, поскольку при создании отчета по этому полю заказана группировка. Раздел «Заголовок группы ‘COMPANY’» мастер оставил пустым, а в раздел «Примечание группы ‘COMPANY’» внес операторы подсчета итогов. В нижний колонтитул попали оператор вычисления текущей даты и текущей страницы.

С помощью конструктора можно модифицировать полученный отчет. Так, для изменения имен полей на их содержательные названия (COMPANY – Компания, PHONE – Телефон, CONTACT – Менеджер, IDATE – Дата заказа, ITOTAL – Сумма заказа) надвигаем курсор мыши на наименование соответст-

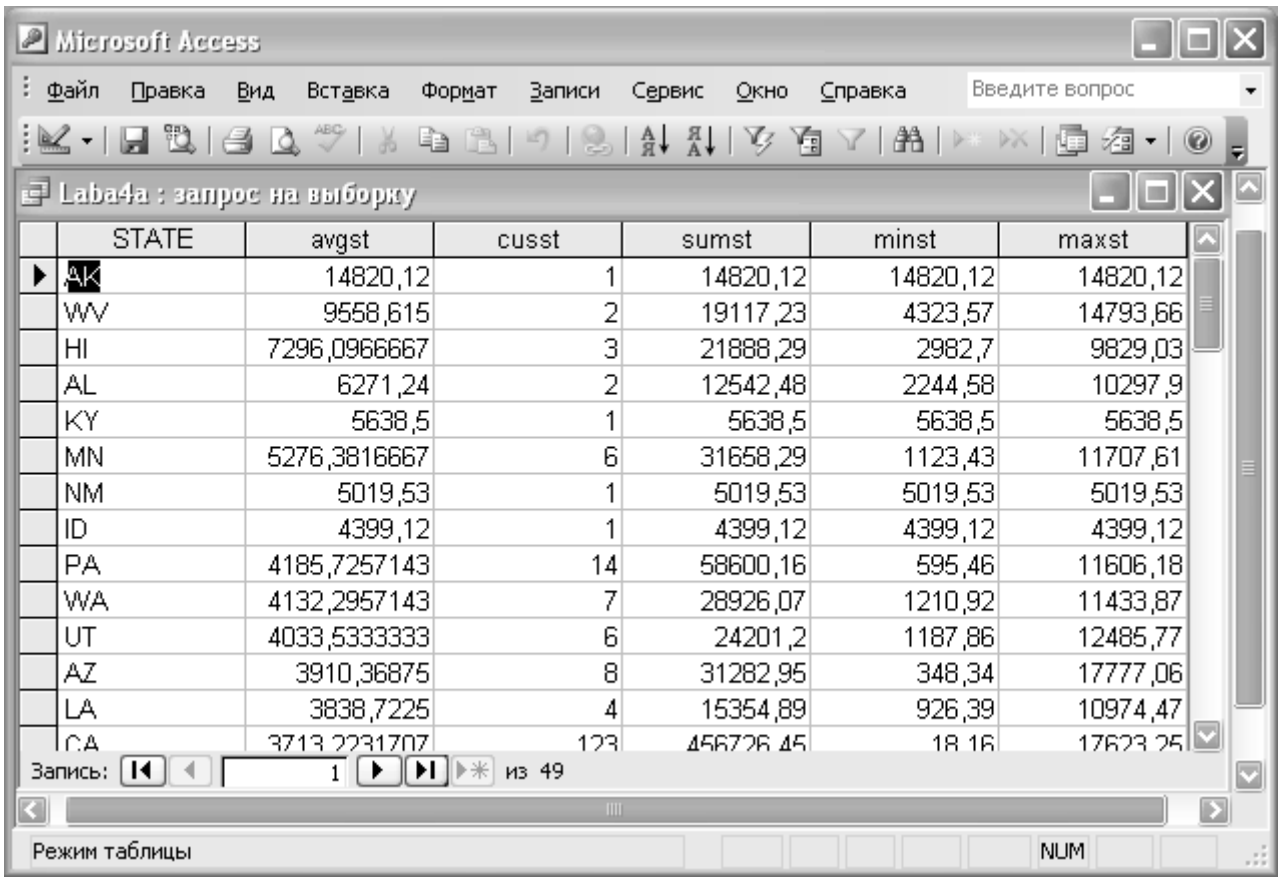
вующего поля из верхнего колонтитула, вызываем правой кнопкой мыши контекстное меню, выбираем в нем пункт «Свойства» и задаем свойству «Подпись» соответствующее значение.

4. ПРОГРАММА РАБОТ

1. Создайте запрос из таблицы, SQL-оператор SELECT которого имеет вид:

```
SELECT CUSTOMER.STATE,
Avg(CUSTOMER.YTDPURCH) AS [avgst],
Count(*) AS [cusst],
Sum(CUSTOMER.YTDPURCH) AS [sumst],
Min(CUSTOMER.YTDPURCH) AS [minst],
Max(CUSTOMER.YTDPURCH) AS [maxst]
FROM CUSTOMER
GROUP BY CUSTOMER.STATE
ORDER BY Avg(CUSTOMER.YTDPURCH) DESC;
```

   Сохраните запрос под именем «Laba4a». Данный запрос наберите в окне запроса, перейдя в режим SQL (опция «Вид – Режим SQL») окна базы данных. После набора перейдите в режим конструктора (опция «Вид – Конструктор») и посмотрите, каким получилось его окно. Далее перейдите в режим таблиц (опция «Вид – Режим таблиц») и посмотрите результат запроса. Он должен быть следующим:



STATE	avgst	cusst	sumst	minst	maxst
AK	14820,12	1	14820,12	14820,12	14820,12
WV	9558,615	2	19117,23	4323,57	14793,66
HI	7296,0966667	3	21888,29	2982,7	9829,03
AL	6271,24	2	12542,48	2244,58	10297,9
KY	5638,5	1	5638,5	5638,5	5638,5
MN	5276,3816667	6	31658,29	1123,43	11707,61
NM	5019,53	1	5019,53	5019,53	5019,53
ID	4399,12	1	4399,12	4399,12	4399,12
PA	4185,7257143	14	58600,16	595,46	11606,18
WA	4132,2957143	7	28926,07	1210,92	11433,87
UT	4033,5333333	6	24201,2	1187,86	12485,77
AZ	3910,36875	8	31282,95	348,34	17777,06
LA	3838,7225	4	15354,89	926,39	10974,47
CA	3713,2231707	123	456726,45	18,16	17623,25

2. Создайте с помощью мастера отчет по запросу «Laba4a», который должен выводить все поля запроса и рассчитывать итоги: среднее значение по avgst (куплено в среднем по штату); количество покупателей в каждом штате cusst; сумму по штатам купленного в каждом штате sumst; минимум от minst (покупки на минимальную сумму в штате); максимум от maxst (покупки на максимальную сумму в штате); все указанные расчетные величины в данном случае формируются непосредственно в запросе с помощью агрегатных функций и группировки, а в отчете необходимо лишь предусмотреть получение итогов по всей таблице. Кроме того, результат запроса будет отсортирован (ORDER BY) по убыванию (DESC) средней суммы покупок (Avg(CUSTOMER.YTDPURCH)) в штате. Имена avgst, cusst, sumst, minst, maxst называются псевдонимами (они в Access могут быть записаны и кириллицей, включая пробелы).
3. С помощью конструктора модифицируйте отчет «Laba4a», перенеся итоги по штатам и общие итоги из отдельных строк для каждой агрегатной функции в одну строку. Имена агрегатных функций удалите, а освободившуюся область уберите путем перетаскивания ее нижней границы вплоть до оставляемой строки со значениями итогов. Просмотрите отчет и сохраните его под именем «Laba4a1».
4. В отчете «Laba4a1» данные представлены так, что их трудно сопоставлять и анализировать, так как числа имеют разное число десятичных знаков после запятой. Для устранения этого недостатка выделите мышью соответствую-

щее поле данных (см. рисунок 15), войдите с помощью правой кнопки мыши в контекстное меню, выберите там пункт «Свойства» и в закладке «Макет» установите свойство «Формат поля» на «Денежный». При этом во все поля с этим свойством будут выводиться числа с двумя знаками после запятой и обозначением валюты. Тип валюты определяется в настройках Windows («Пуск – настройка – панель управления – язык и региональные стандарты»). Преобразуйте таким способом все промежуточные и итоговые поля данных, за исключением cusst. Результат сохраните в отчете под именем «Laba4a2». Тип валюты не меняйте.

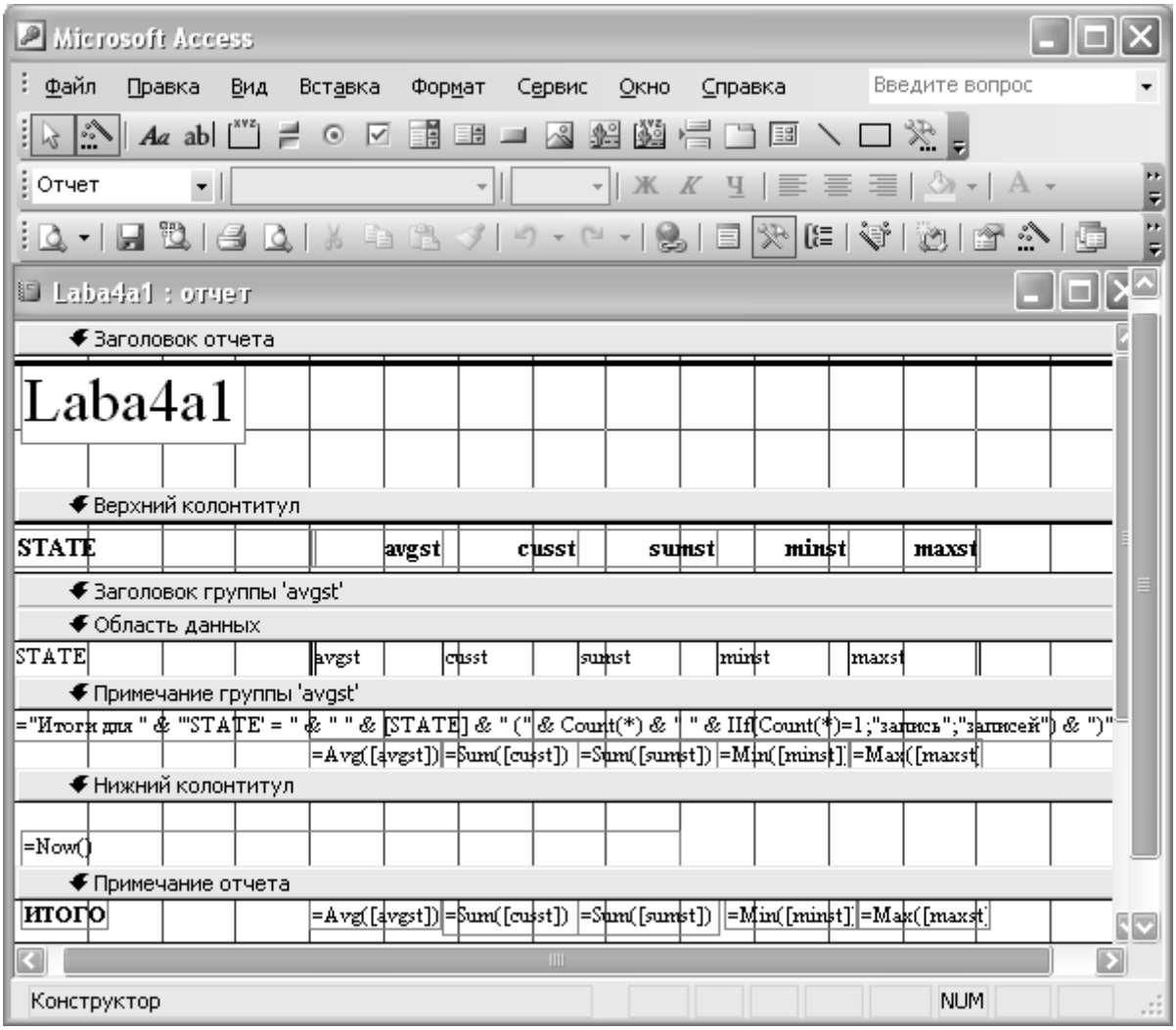


Рисунок 15 – Окно конструктора отчета с выделенным полем данных

5. Просмотрите внимательно полученный отчет, обратив особое внимание на то, по тому ли полю отсортированы данные и все ли итоговые поля попали в раздел примечаний отчета. Если мастер сформировал отчет с отклонениями от требуемого вида, вызовите конструктор и скорректируйте отчет. В частности, для исправления сортировки войдите в контекстное меню и выберите опцию «Сортировка и группировка». В открывшемся окне выберите требуемое поле и порядок сортировки либо отключите сортировку. Если какое-либо поле (например, «=Avg([avgst])»)

не попало в строку «Итого», скопируйте (через опцию «Копировать» контекстного меню) аналогичное поле (оно выделено на рисунке 15) в буфер и вставьте его в строку «Итого».

5. ЗАЩИТА РАБОТЫ

Письменный отчет по работе не делается. Защита работы производится у компьютера в форме демонстрации результатов и отдельных операций, а также ответов на контрольные вопросы из следующего перечня (на вопрос модульного контроля в письменной форме):

1. Вопрос из блока №4 модульного контроля (в письменной форме)
2. Назначение отчетов Access
3. Правила создания отчетов с помощью мастера
4. Правила создания отчетов с помощью конструктора (присоединение таблиц или запросов, работа с элементом управления типа «поле»)
5. Особенности создания запросов с использованием агрегатных функций
6. Сортировка и группировка в запросе
7. Сортировка и группировка в отчете

ЛАБОРАТОРНАЯ РАБОТА № 5
ФОРМИРОВАНИЕ СХЕМ БАЗ ДАННЫХ

1. ЦЕЛЬ РАБОТЫ

Цель работы – изучить и практически освоить роль и принципы использования схем баз данных в системе управления базами данных (СУБД) настольного типа Access.

2. ОБЪЕКТ ИССЛЕДОВАНИЯ

Объектом исследования является СУБД Access фирмы Microsoft. Инструментом исследования является программный пакет Access.

3. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

3.1. Роль и принципы использования схем баз данных

При проектировании баз данных (т.е. «реализованных решений по объемам, размещению и формам существования информации» из определения информационного обеспечения автоматизированных систем) используются несколько подходов. Один из подходов, приобретших в последнее время особенное распространение для проектирования так называемых реляционных баз данных, называется «ER-подход», и основан на понятиях «сущность» и «связь». Аббревиатура ER происходит от слов Entity (сущность) и Relationship (связь), а сам метод основан на выявлении сущностей, их атрибутов и связей между сущностями.

Сущность - это некоторый объект, представляющий интерес для конкретного приложения. Этот объект должен иметь экземпляры, отличающиеся друг от друга и допускающие однозначную идентификацию. Сущности обычно выражаются существительными, например «служащие», «подразделения».

Атрибут - свойство сущности. Например, атрибутами сущности «служащие» могут быть «табельный номер», «фамилия», «имя», «отчество», «год рождения» и.т.д. Тот атрибут или те атрибуты, которые идентифицируют каждый экземпляр сущности, называется ключом сущности (например, «табельный номер», но не «фамилия», «имя», «отчество» ввиду возможности наличия или последующего появления на данном предприятии полных однофамильцев). Ключ может состоять из нескольких атрибутов (составной из К атрибутов) и должен обладать двумя свойствами: уникальностью (в произвольный заданный момент времени никакие два различных элемента сущности не имеют одних и тех же значений для своих атрибутов) и минимальностью (ни один из атрибутов не может быть исключен из К без нарушения условия уникальности).

Связь представляет собой соединение между двумя или более сущностями. Связь обычно выражается глаголом, например «работают».

Сочетание типа «служащие»-«работают»-«подразделения» является элементом так называемой диаграммы ER-типа. Сама же диаграмма состоит из множества сущностей и всевозможных связей между ними.

Связи характеризуются порядками (арностями), степенями и классами принадлежности. Арность показывает, сколько таблиц связывается между собой данной связью. Наиболее распространены бинарные связи, которые связывают две таблицы. Для бинарных связей степени могут иметь один из типов: «один к одному», «один ко многим», «много ко многим». Степени показывают, в каком соответствии находятся строки одной и другой таблиц. Например, таблица «Студент» и таблица «Дисциплины» имеют степень связи «много ко многим», поскольку каждый студент изучает много дисциплин, а каждая дисциплина изучается многими студентами. Класс принадлежности связи определяется для каждого ее конца и показывает, обязательно ли должен быть связан каждый из элементов соответствующей таблицы. Например, если существуют дисциплины, которые никто из студентов не изучает, то класс принадлежности того конца связи, который примыкает к таблице «Дисциплины», необязателен.

Итогом процесса проектирования реляционной базы данных является набор таблиц-отношений (базовых таблиц), связанных между собой через специальные связные поля (внешние ключи – foreign key) либо отдельные связные таблицы, также состоящие из внешних ключей. Например, в базе данных Tutorial из нашего цикла лабораторных работ имеется пять связанных с соответствующими сущностями базовых таблиц – CUSTOMER, INVOICES, OFFICES, PARTS, SALESMAN, – и одна связная таблица DETAIL. Связь между таблицами CUSTOMER и INVOICES осуществляется через внешний ключ CNO, мигрировавший из CUSTOMER (где он является первичным ключом – primary key) в INVOICES. Анализ таблиц Tutorial позволяет также установить остальные связи. Каким образом связывать таблицы – через внешние ключи в базовых таблицах или через внешние ключи в отдельных связных таблицах, – зависит от степеней и классов принадлежностей связей, и составляют один из этапов методики проектирования баз данных. Совокупность базовых и связных таблиц-отношений вместе с определенными первичными ключами, а также связями в виде внешних ключей в базовых и связных таблицах образуют «схему реляционной базы данных».

При создании запросов к базе данных с помощью оператора SELECT языка SQL связи используются в разделе оператора, называемом «условия соединения». В условиях соединения проверяются равенства значений полей из первичных и внешних ключей соединяемых таблиц. При этом, в общем случае, условия соединения формируются «вручную». Если же каким-либо явным образом задать схему базы данных, условия соединения можно формировать автоматически, облегчая тем самым работу пользователя. Это одно из возможных применений схем баз данных, реализованных в конструкторе запросов Access.

Другое, более традиционное применение схем – поддержка аппарата сохранения целостности данных при их первоначальном вводе или обновлении. Различают три стандартных вида целостности: целостность домена, целостность по сущностям и целостность по ссылкам.

Первый вид целостности (целостность домена) связан с принадлежностью данных какой-либо колонки (атрибута) определенному типу (числовому, символьному и др.). Тип данных колонки таблицы называют доменом, откуда и происходит название данного вида целостности. Реализация проверки целостности домена сводится к контролю типа вводимых данных.

Второй вид целостности (по сущностям) контролируется на отсутствие в таблице-отношении двух и более записей (запись – альтернативное наименование строк таблиц, принятое в теории баз данных) с одинаковым значением первичного ключа. Проверка целостности по сущностям реализуется в виде объявления первичного ключа для каждой базовой таблицы и последующей проверки при вводе новой записи, не совпадает ли значение ее первичного ключа со значениями первичных ключей всех других записей, существующих в таблице. Кроме того, здесь же проверяется, не является ли значение первичного ключа неопределенным (Null).

Третий вид целостности (по ссылкам) связан с понятием внешнего ключа и означает корректность связей таблиц: проверяется, совпадает ли значение внешнего ключа некоторой записи с каким-либо значением первичного ключа другой таблицы, с которой связана данная. При этом допускается, чтобы значение внешнего ключа было полностью неопределенным, т.е. неопределенными должны быть все атрибуты составного ключа. Для реализации данного вида проверки СУБД должна иметь средства объявления некоторого набора атрибутов внешним ключом.

3.2. Способ задания схем в Access

Построение схемы производится посредством опции «Сервис – Схема данных» главного меню Access (имеется также соответствующая кнопка на инструментальной панели). Эта опция активна, когда загружено окно базы данных и в ней имеются таблицы. После исполнения указанной опции открывается окно «Схема данных». Если схема еще не создавалась, окно схемы данных пустое:

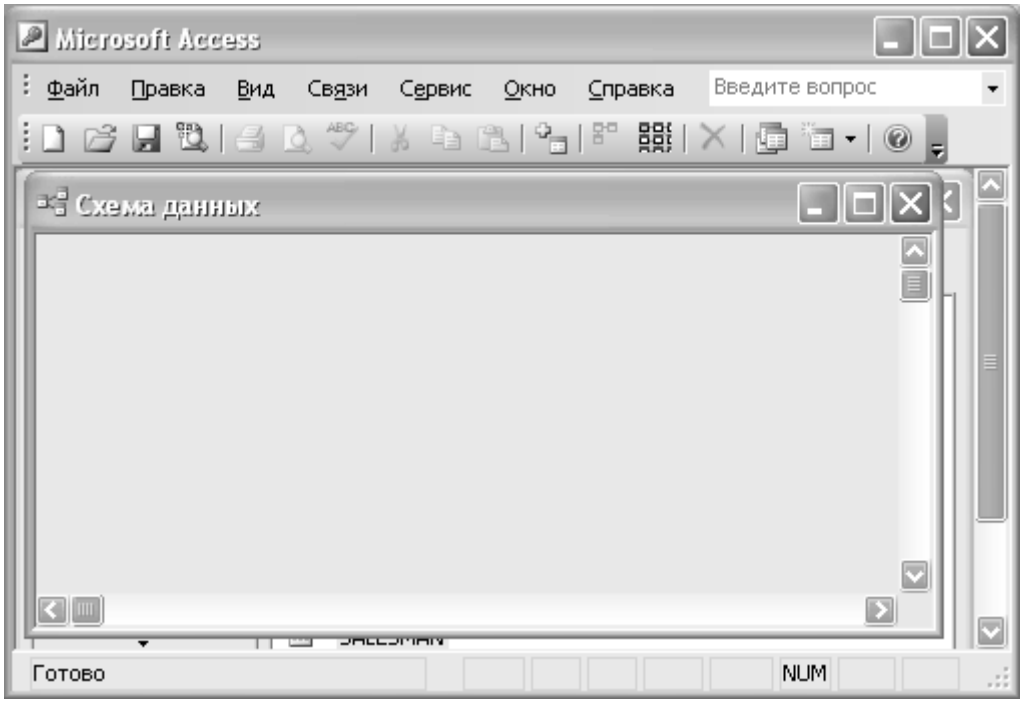


Рисунок 1 – Окно схемы данных

Кроме того, в этом случае автоматически открывается окно добавления таблиц и запросов (на рисунке 2 – базы данных Tutorial):

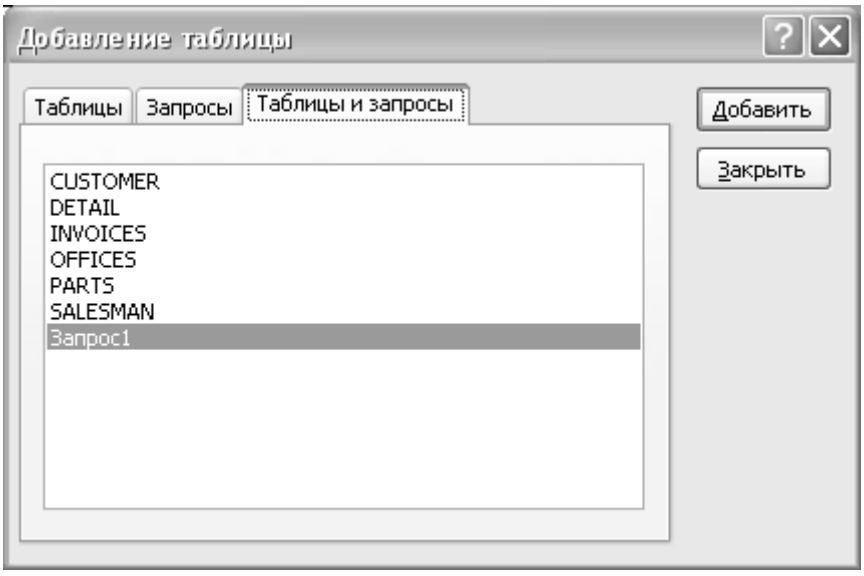


Рисунок 2 – Окно добавления таблиц и запросов

Для добавления таблицы или запроса в схему выделяем поочередно требуемые элементы световым маркером и нажимаем командную кнопку «Добавить». При этом в окне схемы данных появляются выбранные элементы, равномерно размещенные в окне. При необходимости элементы можно разместить так, как вы пожелаете, путем перетаскивания элементов мышью и изменения их размеров:

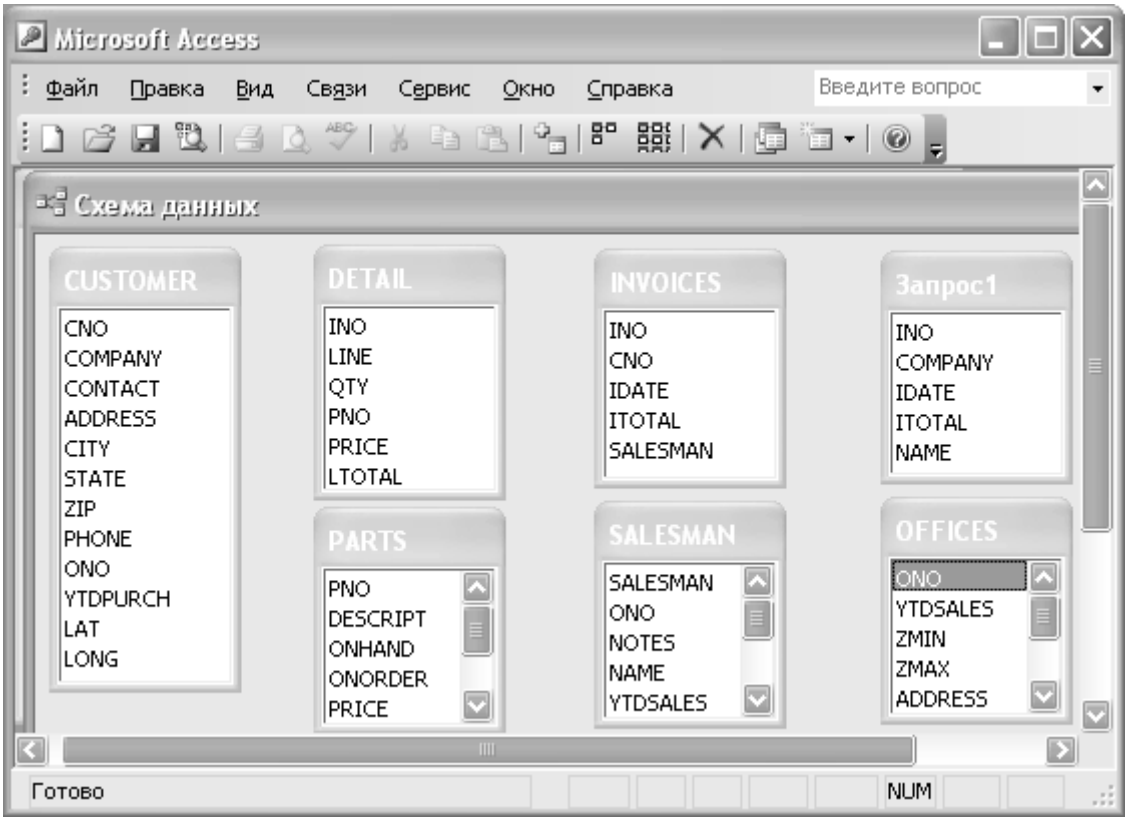


Рисунок 3 – Окно с таблицами схемы данных

Следующим этапом создания схемы является назначение первичных ключей базовых таблиц. Для приведенных таблиц базы данных Tutorial первичными ключами будут: для CUSTOMER – CNO; для INVOICES – INO; для PARTS – PNO; для OFFICES – ONO; для SALESMAN – SALESMAN. Кроме этих таблиц, в базу попал и «Запрос1», отличающийся от таблицы INVOICES только двумя полями, которые взяты из таблиц CUSTOMER и SALESMAN [5]. Для запросов первичные ключи установить нельзя. Для назначения первичных ключей входим в контекстное меню соответствующей таблицы (нажимаем правую кнопку мыши при совмещении курсора мыши с таблицей) и выбираем пункт «Конструктор таблиц». Появляется окно конструктора таблиц:

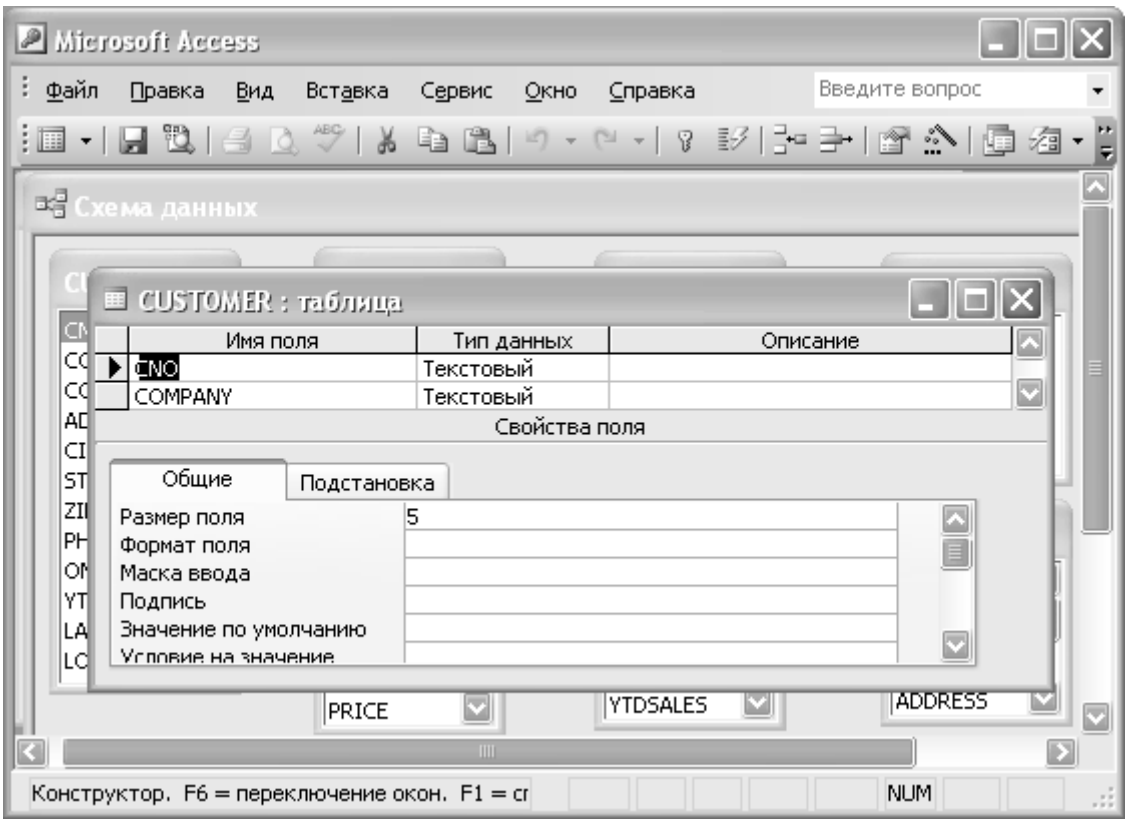


Рисунок 4 – Вызов конструктора таблиц для задания первичного ключа

Выбираем поле и в контекстном меню данного поля устанавливаем опцию «Ключевое поле» (другой способ – посредством опции главного меню «Правка – Ключевое поле»). Если ключ составной, т.е. состоит из нескольких полей, такие операции проделываем для каждого из них. Ключевые поля в окне конструктора таблиц отмечены изображением ключика, а в окне схемы данных они выделяются жирным шрифтом. После установки каждого из ключей появляется окно с запросом о сохранении изменений:

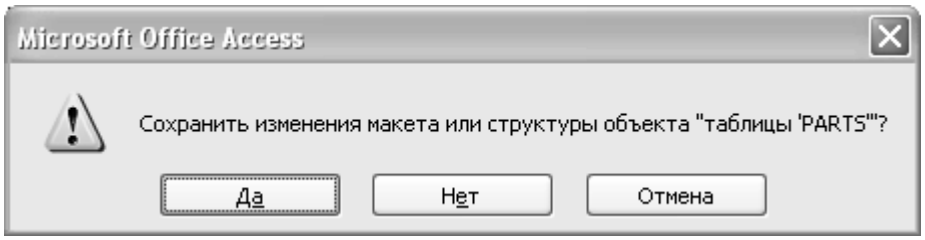


Рисунок 5 – Окно сохранения изменений

В итоге получаем схему с определенными первичными ключами:

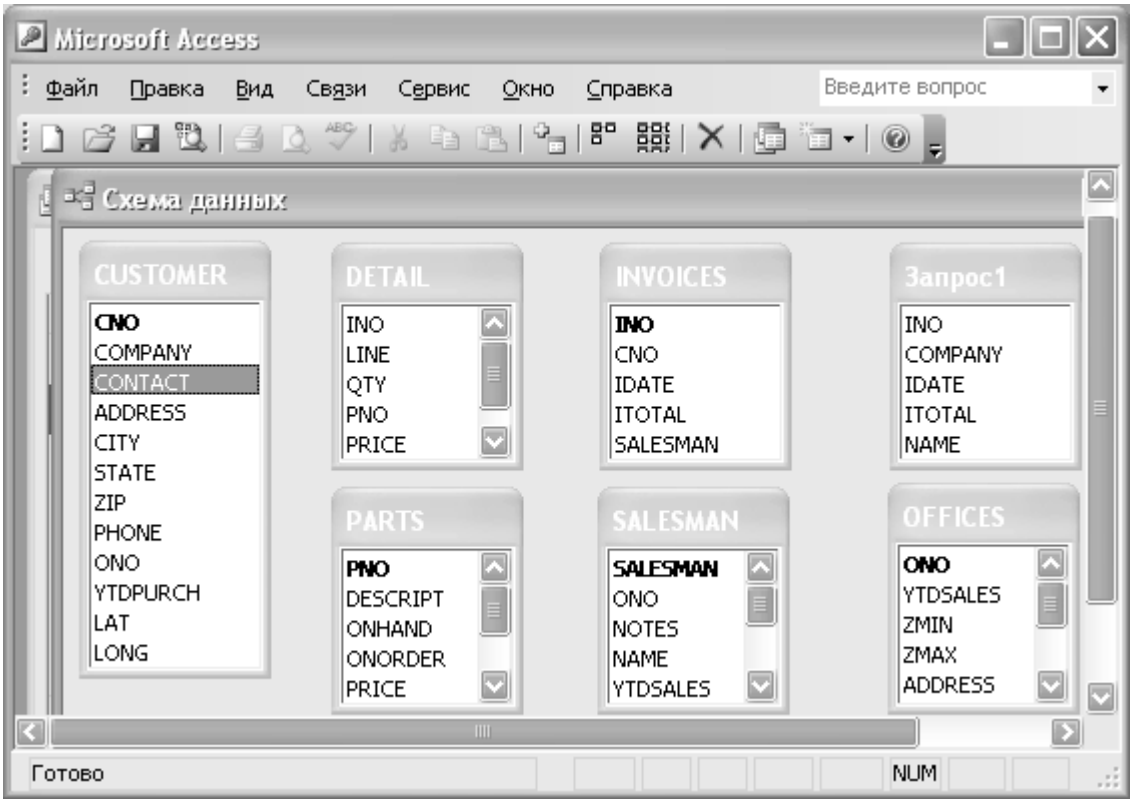


Рисунок 6 – Схема базы данных с определенными первичными ключами

Следующий этап составления схемы – создание связей. В базе Tutorial связные поля (внешние ключи) имеют те же имена, что и первичные ключи, выделенные на рисунке 6 жирным шрифтом. В общем случае это может быть и не так (см. ниже в таблицах «Отделы» и «Сотрудники»). Для создания связи входим в опцию главного меню «Связи – Изменить связь…», в результате чего появляется окно:

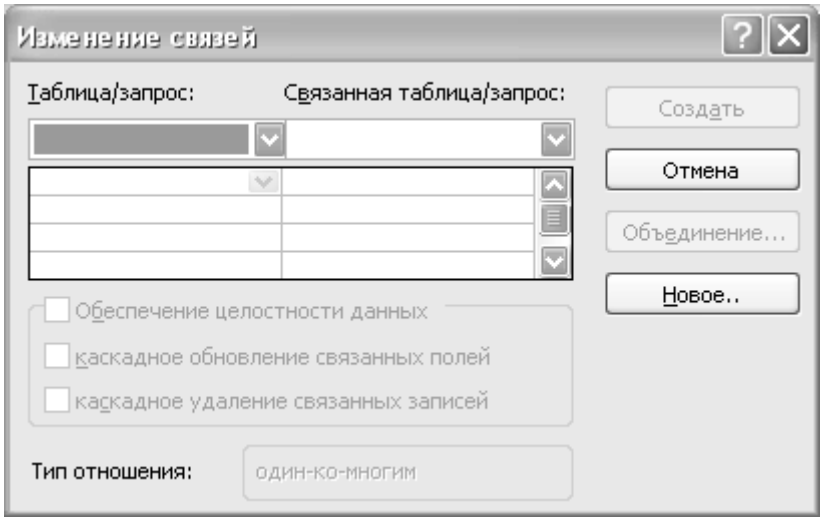


Рисунок 7 – Окно изменения связей

В окне изменения связей, если связей еще нет, нажимаем командную кнопку «Новое…» и в открывшемся окне:

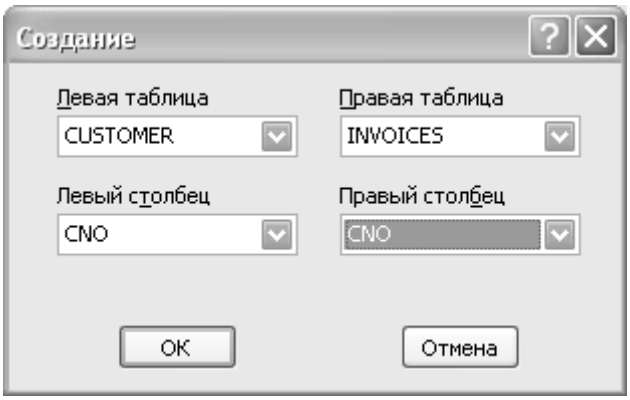
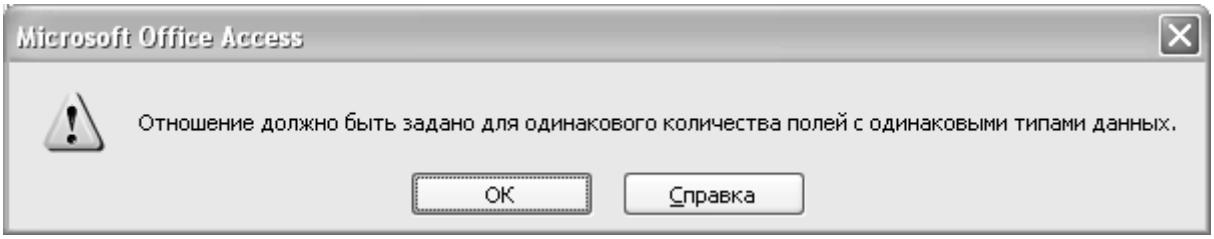


Рисунок 8 – Окно создания связи

выбираем связываемые объекты и нажимаем «ОК». В результате в окне изменения связи (рисунок 7) появится заполненная строка. При нажатии кнопки «Создать» связь создается и в окне схемы данных визуализируется в виде линии. Если при этом установить флажок в поле «Обеспечение целостности данных», то на линии появится обозначение степени связи и станут доступными для изменения флажки «Каскадное обновление связанных полей» и «Каскадное удаление связанных записей». Кроме того, при этом может появиться сообщение типа:



Это сообщение, скорее всего, свидетельствует о несовпадении типов первичного и соответствующего внешнего ключа. В нашем примере такая ситуация имеет место при связи таблиц INVOICES и DETAIL, где связное поле INO имеет разные типы (так получилось при импорте таблиц из FoxPro): INO как первичный ключ в таблице INVOICES имеет числовой тип со значением «целое» для свойства «Размер поля», а INO как внешний ключ в таблице DETAIL – числовой тип со значением этого свойства «двойное с плавающей точкой». Это обстоятельство необходимо иметь в виду, как при импорте таблиц, так и при их создании.

Относительно опций о каскадном удалении и обновлении: существует две стратегии удаления и обновления – через промежуточные сообщения пользователю и каскадные. В первом случае попытка удаления или обновления связанного поля или записи приводит к предупреждающему сообщению о том, что вначале требуется удалить или обновить те записи или поля, на которые ссылаются данное поле или запись. Таким образом, при такой стратегии последовательность удалений или обновлений возлагается на пользователя. Во втором

случае производится автоматическое каскадное удаление (обновление), начиная с записей или полей на дальнем конце цепочки ссылок. Второй способ более быстрый, но и более опасный с точки зрения ошибочного удаления данных.

После создания всех связей, исключая связи с «Запрос1», схема базы данных Tutorial имеет вид:

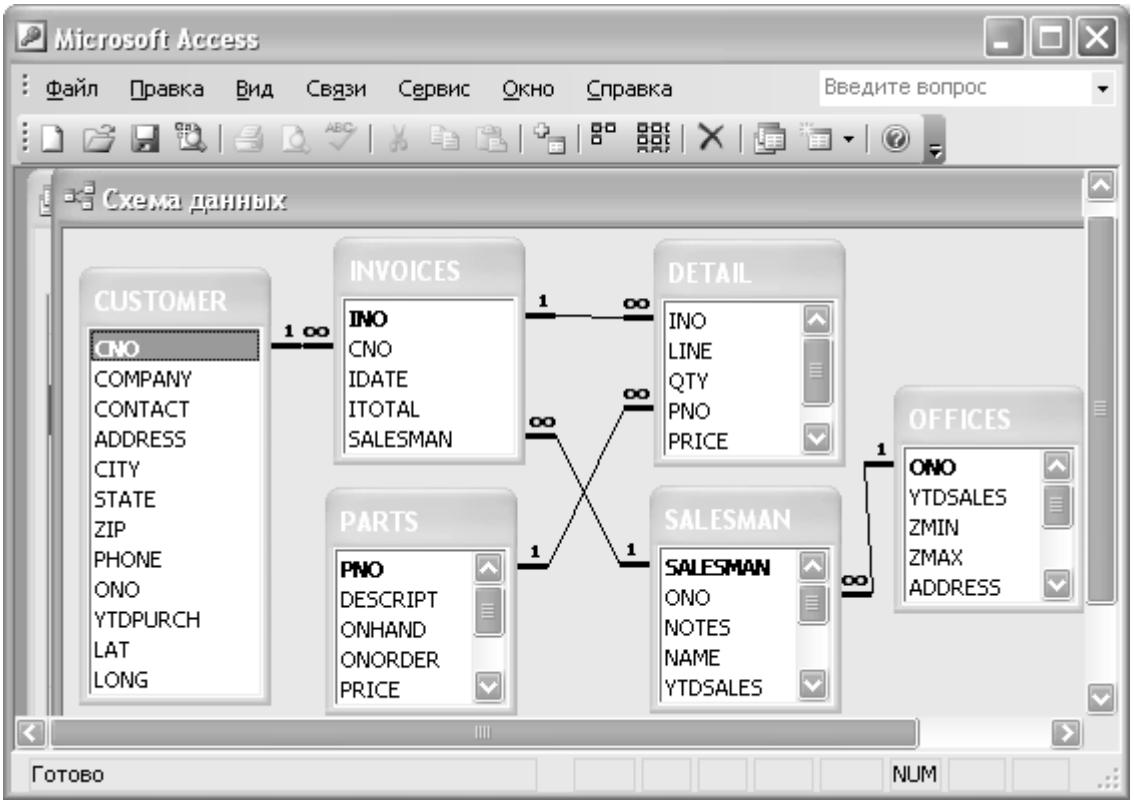


Рисунок 9 – Схема базы данных Tutorial

4. ПРОГРАММА РАБОТ

4.1. Создание запросов с использованием схемы данных

Наличие схемы данных облегчает создание запросов к базе данных. Перед освоением облегченной процедуры создания запросов сформируйте схему базы данных Tutorial. Для этого войдите в Access, загрузите базу данных Tutorial и сформируйте ее схему так, как это сделано в разделе 3.

С помощью конструктора создайте запрос, аналогичный запросу «Счета» из лабораторной работы №3, т.е. запрос из связанных таблиц CUSTOMER, INVOICES, SALESMAN, аналогичный по структуре таблице INVOICES, но в которой значения поля INVOICES.CNO с кодами компаний заменены наименованиями компаний из таблицы CUSTOMER (CUSTOMER.COMPANY), а значения поля с кодами продавцов INVOICES.SALESMAN – соответствующими именами продавцов из SALESMAN.NAME [5].

Для этого из окна базы данных перейдите в окно конструктора запросов и создайте запрос следующего вида:

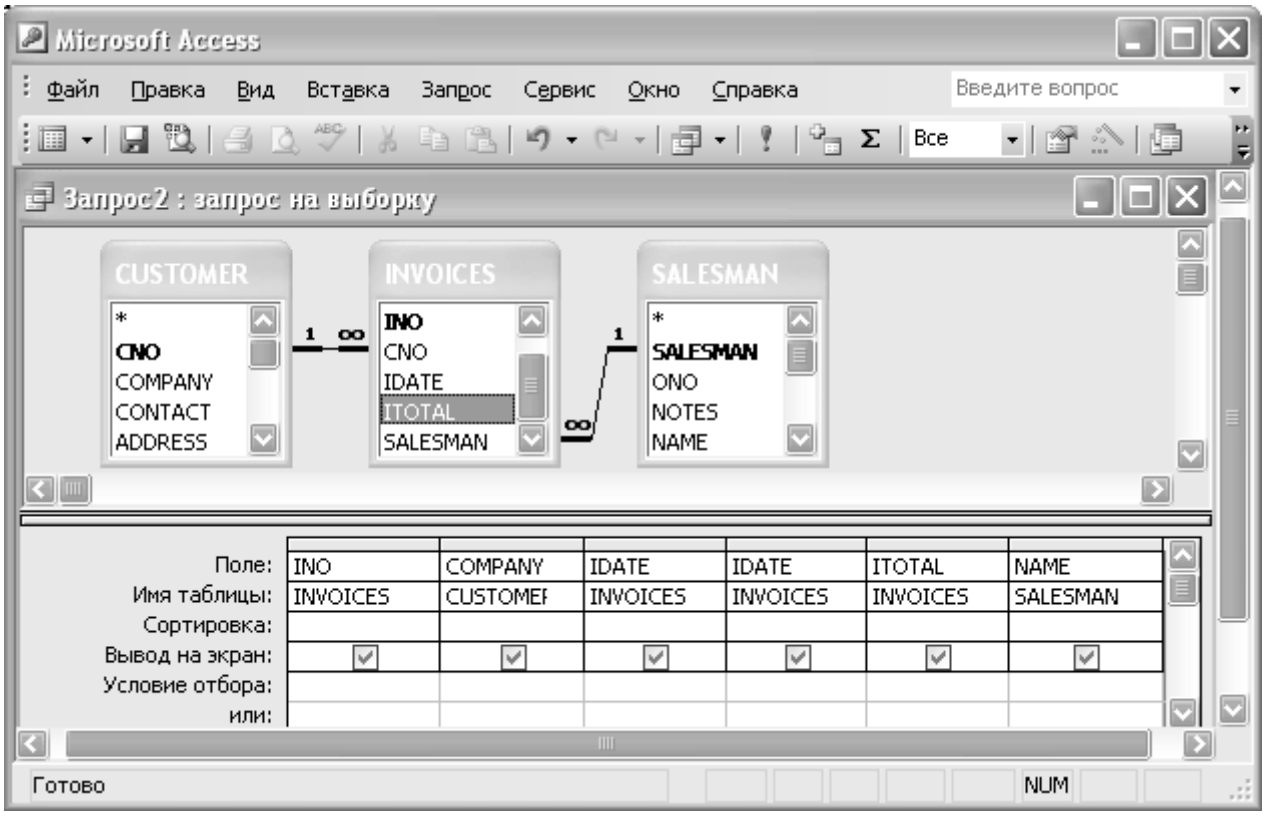


Рисунок 10 – Запрос с использованием схемы данных

Сохраните запрос под именем «Счета1» и сравните окно рисунка 10 с аналогичным окном запроса «Счета». Обратите внимание на то, что в окне запроса «Счета1» нет условий соединения таблиц по полям CNO и SALESMAN. На самом деле соединение есть, в чем нетрудно убедиться как по результатам запроса (получите их, перейдя в режим таблиц посредством опции «Вид – Режим таблицы»), так и по сформированному SQL-оператору SELECT (посмотрите его, перейдя в режим SQL посредством опции «Вид – Режим SQL»).

4.2. Использование схемы данных для контроля целостности базы данных

4.3.1 Проверка целостности домена

Откройте таблицу «Отделы», созданную в лабораторной работе №2 и попытайтесь ввести в поле «Количество сотрудников» любые сочетания символов, не подпадающих под понятие целого числа. Убедитесь, что такие данные отвергаются при попытке сохранить изменения. Обратите внимание на то, что числовой тип вида «длинное целое» (четырехбайтовый тип с максимальным значением 2147483647) для данного поля нерационален, так как допускает бессмысленные значения. Здесь уместен байтовый тип (максимальное значение 255), что более подходит для количества сотрудников в отделе. Внесите это исправление и сохраните результат в таблице «Отделы1».

4.3.2 Проверка целостности по сущностям

Определите посредством конструктора таблиц, какой из атрибутов таблицы является первичным ключом (он помечен изображением ключика). Попытайтесь ввести в таблицу «Сотрудники» строку со значением первичного ключа, совпадающим со значением первичного ключа какой-либо из существующих строк. Убедитесь, что это сделать невозможно. Попытайтесь также в какой-либо строке задать неопределенное значение ключевого атрибута, удалив все его символы, и убедитесь, что это также невозможно сделать.

4.3.3 Проверка целостности по ссылкам

Откройте таблицу «Сотрудники» и измените тип элемента управления для поля «Код отдела» с типа «Список» на тип «Поле» (посредством перехода в режим конструктора таблиц и перехода к закладке «Подстановка»). Попытайтесь в какой-либо из строк изменить значение поля «Код отдела» на значение, отсутствующее в таблице «Отделы», и убедитесь в том, что это можно сделать. Новую таблицу сохраните под тем же именем «Сотрудники».

Далее откройте схему данных, установите связь между таблицами «Отделы» и «Сотрудники» без подключения средств контроля целостности и сохраните схему. Откройте таблицу «Сотрудники» и вновь попытайтесь в какой-либо из строк изменить значение поля «Код отдела» на значение, отсутствующее в таблице «Отделы». Убедитесь, что по-прежнему это можно сделать.

Модифицируйте схему, подключив средства контроля целостности. Для этого с помощью однократным щелчком левой или правой кнопки мыши выделите линию, связывающую таблицы «Отделы» и «Сотрудники», и выберите опцию «Вид – Изменение связей». В открывшемся окне изменения связей установите флажок «Обеспечение целостности данных» (рисунок 11) и попытайтесь закрыть окно изменения связей.

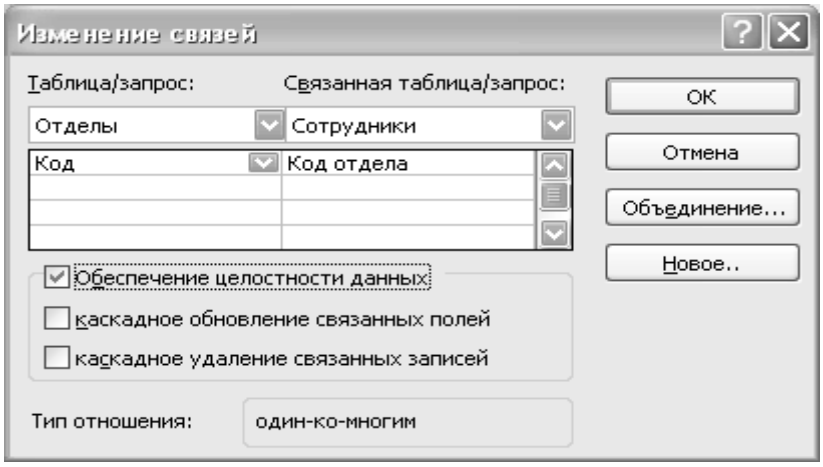


Рисунок 11 – Модифицированное окно изменения связей

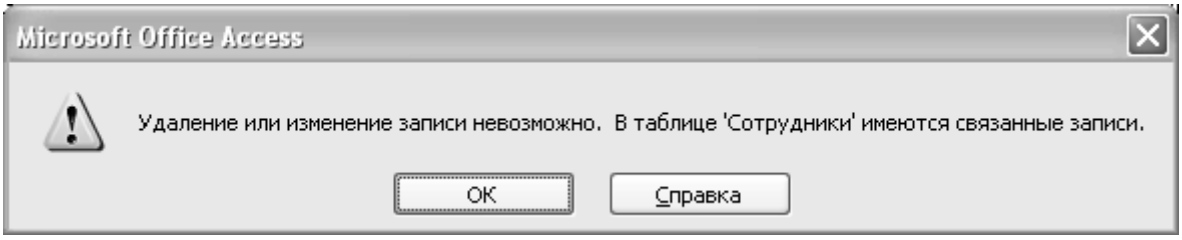
Появится сообщение о нарушении условий целостности и необходимости изменения данных. Нажмите командную кнопку «Отмена», закройте окно схе-

мы данных, откройте таблицу «Сотрудники» и исправьте в ней ранее введенное неправильное значение в поле «Код отдела» на одно из существующих в таблице «Отделы» (поле «Код»). Далее вновь подключите средства обеспечения целостности. Открыв таблицу «Сотрудники», вновь попытайтесь модифицировать одно из значений поля «Код отдела» на недопустимое. Убедитесь, что при попытке сохранения таблицы появится сообщение о необходимости наличия связанной записи в таблице «Отделы».

Проверим еще одну составляющую целостности по ссылкам: возможность неопределенных значений внешнего ключа. Примером такой ситуации является наличие в базе данных сотрудника, не принадлежащего к какому-либо отделу (например, только что принятого на работу и уже занесенного в базу данных, но еще не назначенного на конкретное место работы). Войдите в таблицу «Сотрудники» и удалите все символы из поля «Код отдела» одной или нескольких строк. Убедитесь, что при этом таблицу можно закрыть с сохранением сделанных изменений, т.е. такая операция допустима.

4.3.4. Проверка режимов удаления и обновления данных

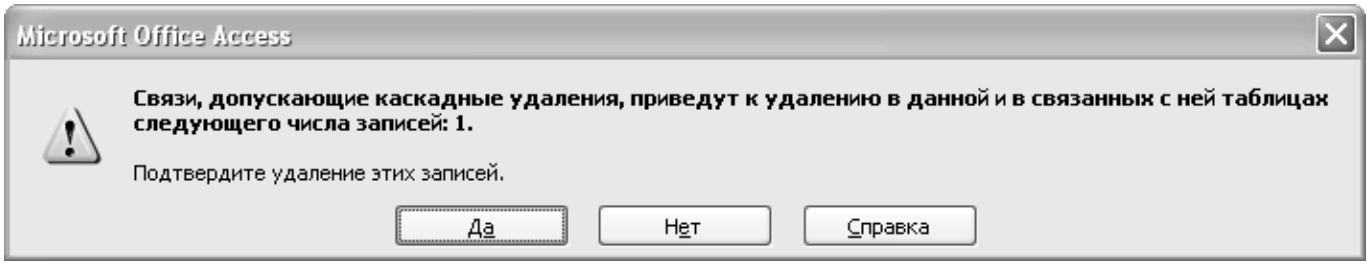
Проверим режимы «ручного» и каскадного удаления и обновления данных на примере связанных таблиц «Отделы» и «Сотрудники». Попробуем вначале удалить какую-либо запись из таблицы «Сотрудники». Для этого откроем эту таблицу, выделим запись (например, первую) и нажмем на клавиатуре клавишу «Delete» (альтернативные способы – посредством опции «Правка – Удаление» меню окна базы данных либо посредством опции «Удалить запись» контекстного меню). Появляется окно с запросом о подтверждении удаления, после чего запись необратимо удаляется. Теперь попробуем удалить такую запись из таблицы «Отделы», на которую есть ссылка в таблице «Сотрудники» (т.е. уберем данные об отделе, а сотрудников этого отдела в базе оставим). Для этого войдем в таблицу «Отделы», выберем подходящую запись и попробуем ее удалить. Появится сообщение:



А теперь попробуем из таблицы «Отделы» удалить ту запись, на которую в таблице «Сотрудники» нет ссылок (такая запись появилась в результате выполнения первого эксперимента настоящего пункта). Убедитесь, что теперь удаление возможно. Таким образом, мы проверили режим «ручного» удаления записей.

Теперь проверим режим каскадного удаления записей. Для этого откроем окно схемы данных, выделим линию связи между таблицами и выберем опцию

«Связи – Изменить связь…». В открывшемся окне изменения связей установим флажок «Каскадное удаление связанных записей», выйдем с сохранением из окна схемы данных и откроем таблицу «Отделы». Вновь попробуем удалить запись, на которую есть ссылка в таблице «Сотрудники». Появится сообщение:



После нажатия командной кнопки «Да» будут удалены записи с соответствующим номером отдела в обеих таблицах. Проделайте описанные действия и убедитесь в этом, посмотрев содержимое таблиц.

Режим «ручного» обновления полей требует, чтобы сначала обновилось соответствующее поле в корневой таблице «Отделы», а уже потом менялось поле в таблице «Сотрудники». В данном случае ключевое поле типа «Счетчик» изменить нельзя из-за особенностей этого типа данных. Для проверки режима можно завести новую запись, у которой значение этого поля установится автоматически. После этого войдем в таблицу «Сотрудники» и у какой-либо записи изменим значение поля «Код отдела» на значение поля «Счетчик» указанной новой записи. Убедитесь, что обновление произошло.

И, наконец, проверим режим каскадного обновления поля. Для этого таблица «Отделы» не подходит по той причине, что ее ключевое поле «Код» изменять нельзя (см. выше). Поэтому проделаем следующее:

- в окне схемы данных выделим линию связи между таблицами «Отделы» и «Сотрудники» и в контекстном меню, вызываемом правой кнопкой мыши, исполним опцию «Удалить»; линия связи в окне схемы исчезнет;
- войдем в таблицу «Отделы», перейдем в режим конструктора и изменим тип поля «Счетчик» на тип «Числовой – Длинное целое», таблицу закроем;
- войдем в окно схемы данных и вновь установим связь между таблицами (см. раздел 3), установив в окне изменения связей все флажки;
- изменим значение ключевого поля одной из записей таблицы «Отделы» и сохраним таблицу;
- откроем таблицу «Сотрудники» и убедимся, что измененное значение автоматически заменило все значения поля «Код отдела», которые ссылались на соответствующую запись таблицы «Отделы».

5. ЗАЩИТА РАБОТЫ

Письменный отчет по работе не делается. Защита работы производится у компьютера в форме демонстрации результатов и отдельных операций, а также

ответов на контрольные вопросы из следующего перечня (на вопрос модульного контроля в письменной форме):

9. Вопрос из блока № 5 модульного контроля (в письменной форме)
10. Понятия сущности, атрибутов сущности, связей между сущностями
11. Характеристики связей
12. Понятия первичного и внешнего ключа
13. Понятие схемы базы данных
14. Целостность данных и ее составляющие (целостность домена, целостность по сущностям, целостность по ссылкам)
15. Использование схем для конструирования запросов
16. Использование схем для контроля целостности данных

ЛАБОРАТОРНАЯ РАБОТА № 6
ЯЗЫК СТРУКТУРИРОВАННЫХ ЗАПРОСОВ SQL

ЦЕЛЬ РАБОТЫ

Данная лабораторная работа предназначена для изучения и практического освоения разнообразных возможностей команды SELECT языка SQL. В результате выполнения этой работы студенты должны в соответствии с индивидуальным заданием сконструировать SQL-запросы к учебной базе данных и выполнить эти запросы в системе Access.

ОПИСАНИЕ УЧЕБНОЙ БАЗЫ ДАННЫХ

Запросы, рассматриваемые в этой работе, используют учебную базу данных, состоящую из шести таблиц: CUSTOMER, INVOICES, DETAIL, SALESMAN, OFFICES и PARTS, содержащих данные по работе воображаемого предприятия. Эти таблицы расположены в директории TUTORIAL.

Ниже описаны некоторые атрибуты этих таблиц, которые придется в соответствии с вариантами заданий использовать в запросах:

Таблица CUSTOMER ("ЗАКАЗЧИКИ") – каждая запись содержит информацию о фирме – заказчике продукции, в том числе:

– customer.cno – идентификационный номер фирмы–заказчика;

– customer.company – название данной фирмы;

– customer.city – название города, в котором находится данная фирма;

– customer.state– сокращенное название штата, в котором находится данная фирма;

– customer.lat – географическая широта местонахождения данной фирмы;

– customer.long– географическая долгота местонахождения данной фирмы.

Таблица INVOICES ("СЧЕТА") – каждая запись содержит информацию о счете, выписанном по конкретному заказу, в том числе:

– invoices.ino – номер счета (номер заказа);

– invoices.cno – номер фирмы – заказчика (для связи с таблицей CUSTOMER);

– invoices.idate – дата выполнения заказа;

– invoices.itotal – общая сумма заказа

– invoices.salesman – номер продавца (для связи с таблицей SALESMAN).

Таблица SALESMAN ("ПРОДАВЦЫ") – каждая запись содержит информацию о торговом агенте, занимающемся сбытом продукции, в том числе:

– salesman.salesman – номер продавца; – salesman.name – ф.и.о. продавца;

– salesman.ytdsales – объем продаж продавца с начала года;

– salesman.ono – номер торгового представительства (для связи с таблицей OFFICES).

Таблица OFFICES ("КОНТОРЫ") – каждая запись содержит информацию о торговом представительстве, занимающемся сбытом, в том числе:

– offices.ono – номер торгового представительства;

– offices.city – название города, в котором находится данное представительство;

– offices.state – сокращенное название штата, в котором находится данное представительство.

Таблица DETAILS ("ПОДРОБНОСТИ") – каждая запись содержит информацию о поставке конкретного вида товара, входящего в заказ, в том числе:

– detail.ino – номер счета (для связи с таблицей INVOICES);

– detail.qty – количество единиц данного товара в заказе;

– detail.price – стоимость единицы данного товара;

– detail.pno – номер детали, входящей в поставку (для связи с таблицей PARTS).

Таблица PARTS ("ДЕТАЛИ") – каждая запись содержит информацию о выпущенной детали конкретного типа, в том числе:

– parts.pno – идентификационный номер детали;

– parts.descript – описание данной детали.

Схема данной базы данных показана на рисунке 9 лабораторной работы №5 на с.67.

Таблица CUSTOMER ("ЗАКАЗЧИКИ") может объединяться с таблицей INVOICES ("СЧЕТА") при помощи поля CNO (номер заказчика), а с таблицей OFFICES ("КОНТОРЫ") – по полю ONO (номер конторы).

Таблица INVOICES может объединяться с таблицей DETAIL ("ДЕТАЛИ") при помощи поля INO (номер счета). Таблица DETAIL может объединяться с таблицей PARTS ("ПАРТИИ") при помощи поля PNO (номер партии). Таблица INVOICES может объединяться также с таблицей SALESMAN ("ПРОДАВЦЫ") при помощи поля SALESMAN (продавец). Таблица SALESMAN может объединяться с таблицей OFFICES при помощи поля ONO.

СОЗДАНИЕ SQL-ЗАПРОСОВ В СУБД MS ACCESS

Все запросы, которые мы рассматривали до сих пор, создавались либо с помощью Мастера, либо с помощью Конструктора запросов. Конструктор запросов представляет собой графический инструмент для создания запросов по образцу (QBE – Query By Example). Однако на самом деле любой запрос хра-

нится в базе данных в формате SQL – Structured Query Language (Язык структурированных запросов) [1]. Основное достоинство этого языка состоит в том, что он является стандартом для большинства реляционных СУБД. SQL имеет унифицированный набор инструкций, которые можно использовать во всех СУБД, поддерживающих этот язык. Действующим стандартом языка SQL является принятая Американским национальным институтом стандартов (American National Standards Institute – ANSI) версия SQL. Фирмы-разработчики СУБД при реализации языка SQL могут вносить в него расширения, но обязаны реализовать базовый набор команд ANSI SQL.

Процессор обработки данных Jet (составная часть Access) выполняет инструкции языка Access SQL (Jet SQL), которые могут отличаться от ANSI SQL. Основные различия между языками ANSI SQL и Microsoft Jet SQL приведены в приложении А.

Для того чтобы создать инструкцию SQL, необходимо [1]:

1) создать новый запрос с использованием Конструктора запросов;
2) закрыть диалоговое окно **Добавление таблицы**, не добавляя таблиц;
3) выбрать команду **Вид, Режим SQL** для вывода окна SQL;
4) удалить весь текст, который выведен в окне SQL;
5) ввести в окне инструкцию SQL;
6) нажать на панели инструментов кнопку **Запуск**, чтобы вывести результирующее множество записей.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Оператор SELECT без преувеличения является сердцем SQL[2]. Он позволяет выполнять поиск и представлять данные самыми разными способами. Большинство инструкций SQL являются либо инструкциями SELECT, либо инструкциями SELECT...INTO.

Инструкция SELECT

Инструкция SELECT предписывает ядру базы данных Microsoft Jet вернуть сведения из базы данных в виде набора записей.

Синтаксис команды SELECT имеет следующий вид [2].

SELECT [*предикат*] { * | *таблица*.* | [*таблица*.]*поле1* [AS *псевдоним1*]
[, [*таблица*.]*поле2* [AS *псевдоним2*] [, ...]]}
FROM *выражение_таблицы* [, ...] [IN *внешняя_база_данных*]
[WHERE...]
[GROUP BY...]
[HAVING...]
[ORDER BY...]
[WITH OWNERACCESS OPTION]

где *предикат* – один из следующих предикатов: ALL, DISTINCT, DISTINCTROW или TOP. Предикаты используются для ограничения числа возвращаемых записей. По умолчанию используется ALL;

* – указывает, что из заданной таблицы или таблиц выбираются все поля;

Таблица – имя таблицы, из полей которой отбираются записи;

поле1, *поле2* – имена полей, содержащих данные, которые необходимо извлечь. Если задано несколько полей, данные извлекаются из них в указанном порядке;

псевдоним1, *псевдоним2* – имена, используемые как заголовки столбцов вместо исходных названий столбцов *таблицы*;

выражение_таблицы – имя таблицы (таблиц), содержащей данные, которые необходимо извлечь;

*внешняя_база_*данных – имя внешней базы данных, содержащей таблицы, указанные в компоненте *выражение_таблицы*, если они не находятся в текущей базе данных.

Чтобы выполнить эту операцию, ядро базы данных Microsoft Jet выполняет поиск указанной таблицы (таблиц), извлекает соответствующие столбцы, выбирает строки, отвечающие заданным условиям, и сортирует полученные строки в заданном порядке.

Инструкции SELECT не вносят изменений в содержимое базы данных.

Если имя поля включено в несколько таблиц в предложении FROM, поместите перед ним имя таблицы и оператор **.** (точку).

Ядро базы данных Microsoft Jet использует имя поля таблицы как имя поля в результирующем наборе записей. Если требуется назначить другое имя поля, используйте зарезервированное слово AS. При использовании статистических функций или запросов, возвращающих неоднозначные или одинаковые имена полей, необходимо использовать предложение AS для создания другого имени поля.

В инструкции SELECT можно использовать дополнительные предложения для дальнейшего ограничения и упорядочения полученных данных. Дополнительные сведения см. в дальнейших подразделах методических указаний.

Инструкция SELECT...INTO

Инструкция SELECT...INTO осуществляет запрос на создание таблицы.

Синтаксис инструкции SELECT...INTO имеет вид:

```
SELECT поле1[, поле2[, ...]]
INTO новая_таблица [IN внешняя_база_данных]
FROM источник
```

где *поле1*, *поле2* – имена полей, которые будут скопированы в новую таблицу;

новая_таблица – имя создаваемой таблицы. Имя должно удовлетворять стандартным соглашениям по именам. Если задаваемое имя *новой_таблицы* уже существует, возникнет сообщение об ошибке;

источник – имя существующей таблицы, из которой отбираются записи. Может представлять собой одну или несколько таблиц, а также запрос.

При создании таблицы ее поля наследуют типы данных и размеры полей базовых таблиц запроса. Другие параметры полей не переносятся. Может, например, понадобиться определить первичный ключ новой таблицы.

Предложение FROM

Предложение FROM задает таблицы и запросы, содержащие выводящиеся в инструкции SELECT поля. В контексте оператора SELECT предложение GROUP BY имеет следующий синтаксис:

```
SELECT список_полей
FROM выражение_таблицы [IN внешняя_база_данных]
```

где *список_полей* – имена полей, загружаемых вместе с любыми псевдонимами, статистическими функциями SQL, предикатами (ALL, DISTINCT, DISTINCTROW или TOP) или с любым другим параметром инструкции SELECT;

выражение_таблицы – выражение, определяющее одну или несколько таблиц для извлечения данных. Выражение может представлять собой имя таблицы, сохраненное имя запроса или результирующее выражение, составленное с использованием операций INNER JOIN, LEFT JOIN или RIGHT JOIN.

Наличие предложения FROM после инструкции SELECT обязательно.

Порядок таблиц, входящих в компонент *выражение_таблицы*, не имеет значения.

Предложение WHERE

Предложение WHERE определяет, какие записи из таблиц, перечисленных в предложении FROM, обрабатываются инструкцией SELECT:

```
SELECT список_полей
FROM выражение_таблицы
WHERE условия_отбора
```

где *список_полей* – имя поля или полей, загружаемых вместе с любыми псевдонимами, предикатами (ALL, DISTINCT, DISTINCTROW или TOP) или с любым другим параметром инструкции SELECT;

условия_отбора – выражение, которому должны соответствовать записи, включаемые в результаты запроса.

Ядро базы данных Microsoft Jet выбирает записи, отвечающие условиям, указанным в предложении WHERE. Если предложение WHERE не задано, запрос возвращает все строки таблицы. Если в запросе указано несколько таблиц, но не задано предложение WHERE или JOIN, запрос генерирует декартово произведение таблиц.

Задавая аргумент *условия_отбора*, литералы даты следует представлять в американском формате, даже если американская версия ядра базы данных Microsoft® Jet не используется. Например, дата «10 мая 1996» записывается как 10/5/96 в Великобритании и как 5/10/96 в США. Литералы даты необходимо заключать в знаки номера (#) (например, `#5/10/96#`).

В SQL имеется ряд операторов и ключевых слов для задания условий отбора, которые будут описаны далее в этом разделе.

Операторы сравнения

В большинстве диалектов SQL имеются следующие операторы сравнения: = (равно), > (больше), < (меньше), >= (больше равно), <= (меньше равно), != (не равно), <> (не равно).

Логические операторы

Если в предложение WHERE нужно поместить несколько условий, то для их соединения можно использовать логические (булевы) операторы AND, OR и NOT.

Диапазоны (BETWEEN и NOT BETWEEN)

Другим средством реализации условий являются диапазоны. Определить диапазон можно двумя способами:

- С помощью операторов сравнения > и <;
- С помощью ключевого слова BETWEEN.

Ключевое слово BETWEEN можно использовать для задания включающего диапазона, если в искомые значения должны включаться и границы диапазона.

При использовании конструкции NOT BETWEEN находятся все строки, не входящие в указанный диапазон.

Списки (IN и NOT IN)

Ключевое слово IN позволяет выбрать значения, совпадающие со значениями из заданного списка. Оператор IN имеет следующий синтаксис:

выражение [**NOT**] **IN(**значение1; значение2; . . .**)**

Оператор IN возвращает значение True, если выражение найдено в списке значений, и False, если оно не найдено. Чтобы вычислить значение выражения с обратным условием (выражения нет в списке значений), можно добавить логический оператор NOT.

Возможно, наиболее полно преимущества ключевого слова IN проявляются во вложенных запросах, также называемых подзапросами.

Оператор LIKE

Оператор LIKE сравнивает строковое выражение с шаблоном в выражении SQL, и имеет следующий синтаксис:

выражение **LIKE** "*шаблон*"

где *выражение* – *в*ыражение SQL, используемое в предложении WHERE;

шаблон – *с*трока или литерал текстовой строки, с которым сравнивается *выражение*.

Оператор LIKE можно использовать для поиска значений в полях, соответствующих указанному шаблону. В *шаблоне* можно либо указывать значение целиком (например, LIKE "Иванов"), либо использовать подстановочные знаки, чтобы найти значения в некотором интервале (например, LIKE "Ив*").

Оператор LIKE не работает с числовыми полями.

Предложение GROUP BY

Предложение GROUP BY объединяет записи с одинаковыми значениями в указанном списке полей в одну группу. Если в инструкцию SELECT включена статистическая функция SQL, такая как **Sum** или **Count**, для каждой записи создается сводное значение.

В контексте оператора SELECT предложение GROUP BY имеет следующий синтаксис:

```
SELECT список_полей
FROM таблица
WHERE условие_отбора
[GROUP BY список_полей_группы]
```

где *список_полей_группы* – имена полей, используемых для группировки записей. Порядок следования имен в *списке_полей_группы* определяет уровень группировки – от самого высокого до самого низкого.

Значения полей GROUP BY, равные **Null**, группируются и не опускаются. Однако значения, равные **Null**, не оцениваются ни одной из статистических функций SQL.

Предложение WHERE используется для исключения строк, которые не нужно группировать. Предложение HAVING используется для фильтрации записей после группировки.

Все поля в списке полей SELECT должны либо содержаться в предложении GROUP BY, либо быть аргументами статистической функции SQL.

Основными статистическими (агрегатными) функциями являются следующие:

– AVG(<искомый_элемент>) – среднее значение по столбцу числовых данных

– COUNT(<искомый_элемент>) – число искомых элементов в столбце; COUNT(*) определяет число строк в выводе запроса.

– MIN(<искомый_элемент>) – определяет наименьшее значение искомого элемента в столбце.

– MAX(<искомый_элемент>) – определяет наибольшее значение искомого элемента в столбце.

– SUM(<искомый_элемент>) – определяет суммарный итог по столбцу числовых данных.

Функции работы с полями не могут быть вложенными.

Предложение HAVING

Предложение HAVING определяет групповую запись, отображаемую в инструкции SELECT с предложением GROUP BY. После того как GROUP BY сгруппирует записи, HAVING покажет те записи, которые отвечают условиям предложения HAVING. В контексте оператора SELECT предложение HAVING имеет следующий синтаксис:

```
SELECT список_полей
FROM таблица
WHERE выбор_условия
GROUP BY список_полей_ группы
[HAVING условие_ группы]
```

где *условие_группы* – выражение, определяющее отображаемую запись.

В общем смысле, предложение HAVING работает аналогично предложению WHERE, но применяется к группам. WHERE накладывает ограничения на строки, а HAVING – на группы. Как правило, предложение HAVING применяется ко всему запросу в целом, после вычисления значений функций и разбиения на группы. Чтобы не запутаться в этих тонкостях, нужно просто запомнить порядок предложений в операторе SELECT. Предложение WHERE всегда стоит после предложения FROM, а предложение HAVING – после предложения GROUP BY. С точки зрения синтаксиса условного выражения, предложения HAVING и WHERE идентичны, отличие состоит лишь в том, что в условии предложения WHERE не могут находиться агрегирующие функции [3].

Предложение HAVING работает следующим образом: сначала GROUP BY разделяет строки на наборы, затем на полученные группы накладываются условия предложения HAVING. Предложение WHERE отсеивает строки до группировки, а предложение HAVING – после.

Предложение ORDER BY

Предложение ORDER BY выполняет сортировку записей, полученных в результате работы запроса. Сортировка выполняется по возрастанию или по убыванию значений указанного поля (полей). В контексте оператора SELECT предложение ORDER BY имеет следующий синтаксис:

SELECT *список_полей*
FROM *таблица*
WHERE *условие_отбора*
[ORDER BY *поле1* [ASC | DESC][, *поле2* [ASC | DESC]][, ...]]]

где *поле1, поле2* – имена полей, по которым выполняется сортировка записей.

По умолчанию используется порядок сортировки «по возрастанию» (зарезервированное слово ASC). Сортировку по убыванию можно осуществить с помощью зарезервированного слова DESC.

Предложение ORDER BY обычно является последним пунктом в инструкции SQL.

В предложение ORDER BY можно включать дополнительные поля. Сначала записи сортируются по полю, указанному первым в предложении ORDER BY. Затем для записей с равными значениями первого поля выполняется сортировка по полю, указанному вторым, и так далее.

Предикаты ALL, DISTINCT, DISTINCTROW, TOP

Предикаты ALL, DISTINCT, DISTINCTROW, TOP указывают записи, выбранные для запросов SQL. Синтаксис использования предикатов:

SELECT [ALL | DISTINCT | DISTINCTROW | [TOP *n* [PERCENT]]]
FROM *таблица*

где предикат ALL подразумевается, если один из предикатов не включен. Ядро базы Microsoft Jet выбирает все записи, отвечающие условиям инструкции SQL;

DISTINCT исключает записи, содержащие повторяющиеся данные в отдельных полях. В результаты запроса включаются только уникальные значения каждого из полей, находящегося в списке инструкции SELECT. Если в списке выбора находится несколько элементов, при использовании ключевого слова DISTINCT выбираются строки с уникальными комбинациями значений этих элементов;

DISTINCTROW исключает данные полностью повторяющихся записей, а не содержащих отдельные поля с одинаковыми данными;

TOP n [PERCENT] возвращает некоторое количество записей, находящихся в числе первых или последних записей диапазона, заданного предложе-

нием ORDER BY. Можно также использовать зарезервированное слово PERCENT для получения некоторого процента записей, находящихся в числе первых или последних записей диапазона, заданного предложением ORDER BY.

Структурированные запросы и подзапросы

Подзапрос – это дополнительный метод манипуляций с несколькими таблицами, представляющий собой оператор SELECT, вложенный:

– в предложение WHERE, HAVING или SELECT другого оператора SELECT;

– в оператор INSERT,UPDATE или DELETE;

– в другой подзапрос [2].

Именно возможность вложения операторов SQL друг в друга является причиной, по которой SQL был назван Structured Query Language (язык структурированных запросов) [2].

Подзапросы возвращают результаты внутреннего запроса во внешнее предложение и имеют две основные формы: некоррелированную и коррелированную. Первая реализуется (концептуально) «изнутри наружу», т.е. внешний запрос выполняет то или иное действие, основываясь на результатах выполнения внутреннего запроса. Вторую (коррелированную) форму подзапроса можно представить себе как обратное действие: внешний оператор SQL предоставляет значения для внутреннего подзапроса, которые будут использоваться при его выполнении. Затем результаты выполнения подзапроса возвращаются во внешний запрос [2].

Пример 0.1 **Выбрать названия всех фирм, совершавших сделки с продавцом 434.**

Некоррелированный запрос:

```
SELECT company
FROM customer
WHERE cno  IN
   (SELECT  cno
    FROM invoices
    WHERE salesman LIKE 434);
```

Коррелированный запрос:

```
SELECT company
FROM customer
WHERE '434'  IN
   (SELECT  salesman
    FROM invoices
    WHERE  invoices.cno= customer.cno);
```

Этот же запрос можно сформулировать как запрос на соединение:

```
SELECT DISTINCT company
FROM customer, invoices
WHERE customer.cno=invoices.cno
AND salesman LIKE 434;
```

Окончательное решение о том, каким же средством воспользоваться – подзапросом или соединением таблиц – обычно определяется стилем, в котором Вы предпочитаете работать. Тем не менее, необходимо использовать подзапросы, когда нужно сравнивать значения агрегатных функций с другими значениями, а соединения необходимо использовать для отображения информации из нескольких таблиц.

Как коррелированные, так и некоррелированные подзапросы бывают трех типов, в зависимости от элементов в предложении WHERE внешнего запроса:

1) подзапросы, которые не возвращают ни одного или возвращают несколько элементов (начинаются с IN или с оператора сравнения, содержат ключевые слова ANY или ALL);

2) подзапросы, которые возвращают единственное значение (начинаются с простого оператора сравнения);

3) подзапросы, которые представляют собой тест на существование (начинаются с EXISTS).

Подзапросы, возвращающие несколько элементов

Эта группа включает подзапросы, начинающиеся с IN, NOT IN или оператора сравнения с ключевыми словами ANY или ALL.

Подзапросы, начинающиеся с ключевого слова IN имеют следующую общую форму:

```
SELECT…
WHERE выражение [NOT] IN (подзапрос)
```

Результатом внутреннего подзапроса является список, включающий от нуля до нескольких значений. После того, как подзапрос возвратит результаты, к их обработке приступит внешний запрос.

Запросы с ключевым словом IN приведены в примере Пример 0.1.

В другом виде подзапросов, которые не возвращают или возвращают несколько строк, используется оператор сравнения, модифицированный ключевыми словами ANY или ALL. Такие подзапросы имеют форму вида:

```
SELECT….
WHERE выражение оператор_сравнения [ANY | ALL] (подзапрос)
```

Посредством ALL задается условие соответствия <выражения> условиям сравнения для ВСЕХ значений, сгенерированных подзапросом, чтобы записи, где содержатся эти поля, попали бы в результат запроса.

Посредством ANY задается условие для соответствия <выражения> по крайней мере одному из значений условий сравнения, сгенерированных подзапросом.

Пример 0.2 Вывод всех записей из таблицы CUSTOMER, у которых значение объема покупок YTDPURCH меньше, чем любое значение объема покупок в штате Мичиган "MI":

```
SELECT * FROM customer a WHERE a.ytdpurch < ANY
(SELECT ytdpurch FROM customer WHERE state = "MI");
```

Подзапросы, возвращающие единственное значение

Подзапрос, начинающийся с немодифицированного оператора сравнения (оператор сравнения, не сопровождаемый ключевыми словами ANY или ALL), должен возвращать единственное значение. (В противном случае Вы получите сообщение об ошибке, и запрос не будет обработан.) Такие запросы имеют общую форму вида:

```
SELECT…
WHERE выражение оператор_сравнения (подзапрос)
```

Чтобы воспользоваться этим видом подзапроса, Вы должны достаточно хорошо знать свои данные и суть решаемой задачи, чтобы быть уверенными в том, что подзапрос возвратит одно и только одно значение. Если же Вы рассчитываете получить несколько значений воспользуйтесь ключевым словом IN или модифицированным оператором сравнения.

Пример 0.3 Найти названия всех деталей, с ценами выше текущей средней цены.

```
SELECT pno
FROM parts
WHERE price >
        (SELECT AVG(price)        FROM parts);
```

Подзапросы, выполняющие проверку на существование

Ключевое слово EXISTS (квантор существования) в предложении WHERE выполняет проверку на существование (или несуществование) данных, которые удовлетворяют критериям соответствующего подзапроса [2]. Подзапрос, который начинается с EXISTS, имеет общую форму следующего вида:

```
SELECT…
WHERE [NOT] EXISTS (подзапрос)
```

EXISTS выполняет проверку на наличие или отсутствие пустого набора строк. Если подзапрос возвращает хотя бы одну строку, этот результат оценивается как «истина». Если подзапрос возвращает пустой набор (строк нет) этот результат оценивается как «ложь».

Пример 0.4 Найти названия деталей, которые еще никто не покупал.

```
SELECT descript
FROM parts
WHERE NOT EXISTS
        (SELECT *  FROM detail WHERE detail.pno = parts.pno);
```

РЕАЛИЗАЦИЯ ОПЕРАЦИЙ РЕЛЯЦИОННОЙ АЛГЕБРЫ СРЕДСТВАМИ ЯЗЫКА SQL

Специальные реляционные операции

Селекцией (выбором) называется унарная операция над отношением, позволяющая построить “горизонтальное” подмножество отношения, т.е. подмножество кортежей внутри отношения, удовлетворяющих определенному предикату.

Пример 0.1 Получить информацию о компаниях, входящих в базу данных CUSTOMER.DBF, название которых содержит слово "Computer".

```
SELECT *
FROM customer
WHERE company LIKE "*Computer*";
```

Селекция : запрос на выборку

CNO	COMPANY	CONTACT	ADDRESS	CITY	STATE	ZIP	PHC
14021	1st Computers	Jeff W. Culberts	5111 Parkway	Brookline	MA	02146	617/232
A8818	A. Arts Comput	Darryl Roudebu	3305 Plantation	Melrose	MA	02176	617/662
A3061	Advantage Corr	Duane Marshall	3784 Van Dyke	San Jose	CA	95131	408/946
A0887	American Comp	Dick W Guyton	199 E Main	Lubbock	TX	79413	806/799
B0777	Belmar Tronixs	Andy Rigney	2239 University	Santa Monica	CA	90405	213/452
B2546	BoreMax Comp	Greg Sanderso	2225 Wisconsir	Columbus	GA	31904	404/327
B5945	Business State	Bob Reetz	2 Faar W. 4U	Allentown	PA	18105	215/481
C5575	City Computer I	Vince Eng	1501 Mulberry I	Commack	NY	11725	516/499
C1977	Computer Bud	Phil Linak	1281 Ventura B	Concord	CA	94519	415/825
C3386	Computerized C	Al Reozan	1852 California	Nashville	TN	37208	615/327
C1134	Computer Direc	Boris Southwor	910 Stock Rd.	Longview	TX	75604	214/759
C6690	Computer Flash	Mark L Bosted	17610 5th St.	New York	NY	10022	212/708
C8550	Computer Inc	Phil Yang	20051 Middle G	Spring Grove	IL	60081	312/587

Запись: 1 из 49

Рисунок 0.1

Проекцией называется унарная операция, позволяющая получать “вертикальное” подмножество заданного отношения, т.е. такое подмножество, которое получается выбором специфицированных атрибутов с последующим исключением (!!), если это необходимо, избыточных дубликатов кортежей, состоящих из значений выбранных атрибутов.

Пример 0.2 Получить список тех штатов, где имеется хотя бы один покупатель.

```
SELECT DISTINCT state
 FROM customer ;
```

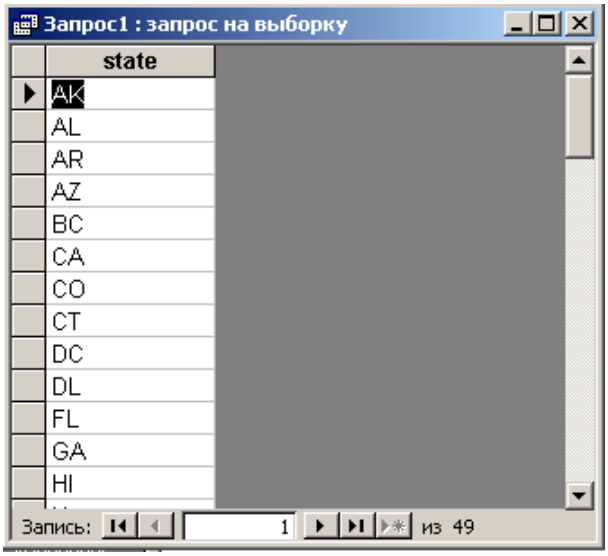


Рисунок 0.2

Соединение. Пусть θ - любой допустимый оператор сравнения скаляров (=, <, и т.д.). Тогда θ-соединением отношения R по атрибуту A с отношением T по атрибуту D называется множество всех кортежей “b” таких, что “b” является конкатенацией какого-либо кортежа “r”, принадлежащего R, и какого-либо кортежа t, принадлежащего T, и предикат r.A θ t.D принимает значение “истина”. При этом R.A и T.D должны быть определены на одном и том же домене, а оператор θ должен иметь смысл для этого домена.

Пример 0.3 Получить список всех пар номеров партий и их описаний, обе части которых были заказаны одним и тем же покупателем.

```
SELECT a1.pno, a1.descript, a2.pno, a2.descript
FROM parts a1, parts a2, invoices b1, invoices b2,  detail c1, detail c2
WHERE b1.ino = c1.ino AND c1.pno = a1.pno
 AND b2.ino = c2.ino AND c2.pno = a2.pno
 AND b1.cno = b2.cno
 AND a1.pno < a2.pno;
```

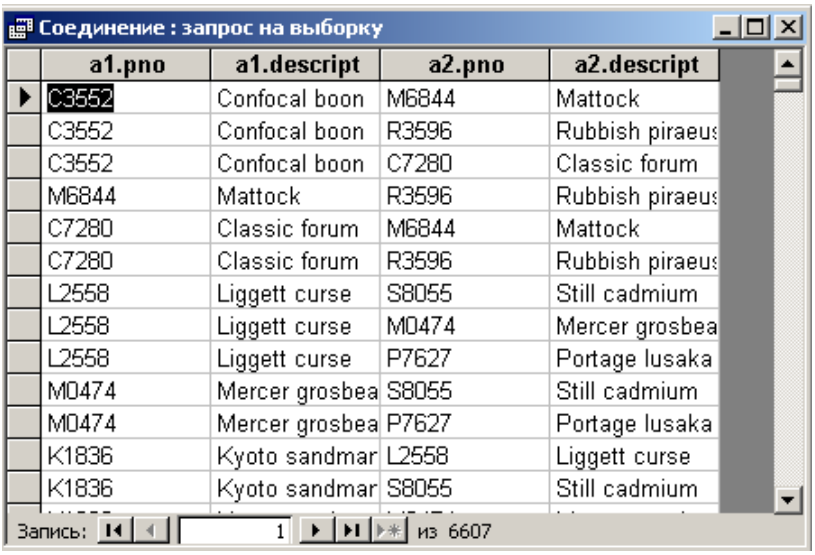


Рисунок 0.3

Примечания. Этот запрос порождает более 6000 результирующих строк. Если Вы не будете достаточно аккуратны, Вы можете получить более 12000 строк. Трюк, с помощью которого получается только 6000 строк, заключен в последней строке запроса: каждая пара выбирается только однажды. В противном случае каждая пара выбиралась бы дважды - как (x, y) и как (y, x).

Рассмотрим отдельно соединение таблицы со своей копией. Соединение таблицы с самой собой является другим вариантом соединения на основе равенства, при этом сравниваются значения внутри столбца одной таблицы.

Пример 0.4 Поиск покупателей, живущих в г. Даллас, и имеющих одинаковые почтовые индексы.

Так как в этом запросе таблица CUSTOMER объединяется сама с собой, то для различения этих двух ролей в списке таблиц предложения FROM ей нужно задать два разных псевдонима.

```
SELECT DISTINCT c1.cno, c1.company, c1.zip
FROM customer AS c1, customer AS c2
WHERE c1.city LIKE 'Dallas'
AND c1.cno<>c2.cno
AND c1.zip=c2.zip;
```

Рисунок 0.4

Теоретико-множественные операции

Бинарные операции над множествами, к которым относятся объединение, пересечение и разность множеств, применимы только к тем отношениям, которые являются совместимыми по объединению (т.е. имеют идентичные структуры). Поэтому, чтобы продемонстрировать работу этих операций на примере имеющихся таблиц, сначала будем производить операции селекции и проекции для того, чтобы сформировать отношения, совместимые по объединению.

Операция **объединения** позволяет получить отношение, состоящее из всех строк, входящих в одно или оба объединяемых отношения. При этом исходные отношения или их объединяемые проекции должны быть совместимыми по объединению. Для SQL это означает, что две таблицы можно объединять тогда и только тогда, когда:

- они имеют одинаковое число столбцов, например, m;
- для всех i (i = 1, 2, ..., m) i-й столбец первой таблицы и i-й столбец второй таблицы имеют в точности одинаковый тип данных.

Оператор UNION позволяет объединить результаты нескольких запросов. Этот оператор полезен, когда вы хотите одновременно увидеть аналогичные данные из различных таблиц. Синтаксис оператора UNION имеет следующий вид:

```
[TABLE] запрос1 UNION [ALL] [TABLE] запрос2
[UNION [ALL] [TABLE] запросN [ ... ]]
```

Пример 0.5 Получить адреса и телефоны всех покупателей и продавцов, находящихся в г. Сан-Франциско. Воспользуемся операцией объединения с предшествующими операциями селекции и проекции.

Создадим два отношения, совместимых по объединению.

Отношение, содержащее адреса и телефоны всех покупателей находящихся в г. Сан-Франциско, можно создать с помощью следующего запроса:

```
SELECT address, phone INTO sel1
FROM customer
WHERE city like 'San Francisco';
```

Отношение, содержащее адреса и телефоны всех продавцов находящихся в г. Сан-Франциско, можно создать с помощью следующего запроса:

```
SELECT address, phone INTO sel2
FROM salesman
WHERE city LIKE 'San Francisco';
```

Для полученных отношений sel1 и sel2 выполним операцию объединения:

```
table sel1
UNION
table sel2;
```

Результат операции объединения представлен на рисунке 0.5.

Эту задачу можно было решить также с помощью одной инструкции:

```
SELECT address
FROM customer
WHERE city like 'San Francisco'
UNION
SELECT address
FROM salesman
WHERE city LIKE 'San Francisco';
```

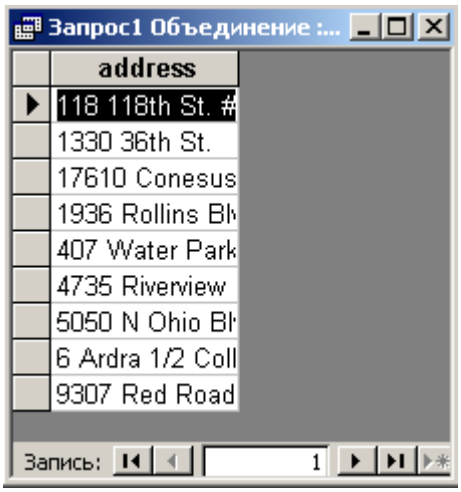


Рисунок 0.5

Пересечением двух отношений R, S называется множество всех кортежей, каждый из которых принадлежит как R, так и S.

Пример 0.6 Получить список штатов, в которых расположены и покупатели, и торговые представительства. Воспользуемся пересечения объединения с предшествующей операцией проекции.

С помощью операции проекции создадим два отношения, совместимых по объединению.

Отношение, содержащее список всех штатов, где расположены компании покупателей, можно создать с помощью следующего запроса:

```
 SELECT DISTINCT state INTO sel_con1
FROM customer;
```

Отношение, содержащее список всех штатов, где расположены торговые представительства, можно создать с помощью следующего запроса:

```
SELECT DISTINCT state INTO sel_con2
FROM offices;
```

Выполним операцию пересечения полученных отношений sel_con1 и sel_con1:

```
SELECT DISTINCT sel_con1.*
FROM sel_con1, sel_con2
WHERE sel_con1.state=sel_con2.state;
```

Результат выполнения операции пересечения представлен на рисунке 0.6.

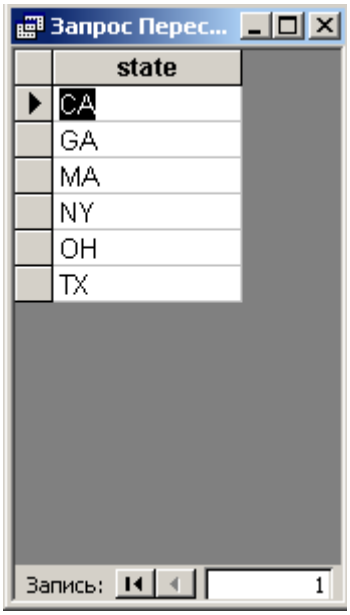


Рисунок 0.6

Этот же результат можно получить и по-другому:

```
SELECT DISTINCT customer.state
FROM CUSTOMER, OFFICES
WHERE offices.state = customer.state;
```

или

```
SELECT DISTINCT state
FROM CUSTOMER
WHERE state IN (SELECT DISTINCT state FROM offices);
```

Также возможен вариант

```
SELECT DISTINCT state
FROM customer
WHERE EXISTS (SELECT * FROM offices
WHERE offices.state=customer.state);
```

Разностью между двумя отношениями R, S называется множество всех кортежей, каждый из которых принадлежит R и не принадлежит S.

Пример 0.7 Получить список штатов, в которых расположены конторы покупателей, но нет торговых представительств. Воспользуемся операцией разности с предшествующей операцией проекции.

Создадим отношение, содержащее список всех штатов, где расположены конторы покупателей, и отношение, содержащее список всех штатов, где расположены торговые представительства, выполнив соответствующие операции проекции, как в примере 0.6.

Выполним операцию разности над полученными отношениями:

```
SELECT DISTINCT *
FROM sel_con1 AS s1
WHERE s1.state NOT  IN
(SELECT DISTINCT sel_con2.state FROM sel_con2);
```

Результат выполнения операции разности представлен на рисунке 5.7

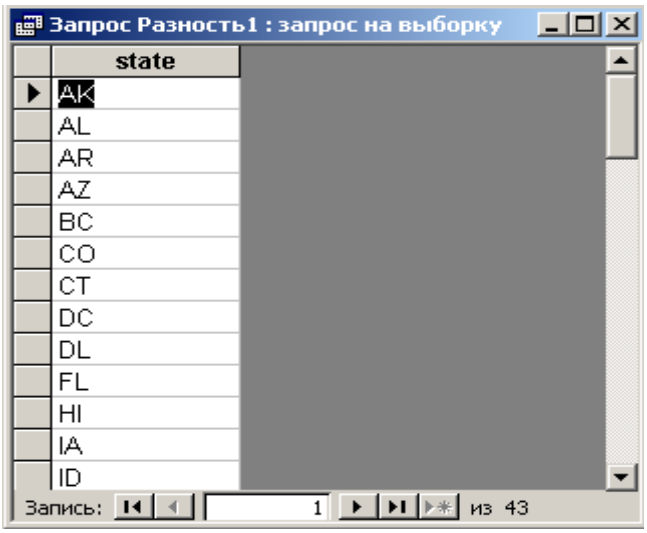


Рисунок 0.7

Эту задачу можно решить и с помощью одной инструкции:

```
SELECT DISTINCT state
FROM customer
WHERE state NOT IN  (SELECT DISTINCT state FROM offices);
```

Такой же результат можно получить следующим образом:

```
SELECT DISTINCT state
FROM customer
WHERE NOT EXISTS (SELECT * FROM offices
WHERE offices.state=customer.state);
```

Следующий вариант даст НЕПРАВИЛЬНЫЙ результат:

```
SELECT DISTINCT customer.state
FROM customer, offices
WHERE offices.state <> customer.state;
```

Декартовым произведением двух отношений R, S называется множество всех кортежей таких, что t является конкатенацией некоторого кортежа r, принадлежащего R и какого-либо кортежа s, принадлежащего S.

Для получения декартова произведения нескольких таблиц надо указать во фразе FROM перечень перемножаемых таблиц, а во фразе SELECT – все их столбцы.

Пример 0.8 Декартово произведение таблиц DETAIL и PARTS.

```
SELECT detail.*, parts.*
FROM detail, parts;
```

Запрос Декартово произведение : запрос на выборку

INO	LINE	QTY	DETAIL.PNO	DETAIL.PRICE	LTOTAL	PARTS.PNO	DESCRIPT	ONHAND
1054	1	2	C7280	278,25	556,5	A9155	Abashed goat	18
1054	1	2	C7280	278,25	556,5	A3858	Abated operativ	17
1054	1	2	C7280	278,25	556,5	A6064	Abrogate obsta	75
1054	1	2	C7280	278,25	556,5	A6162	Absentee donki	5
1054	1	2	C7280	278,25	556,5	A2330	Achievers nally	52
1054	1	2	C7280	278,25	556,5	A1953	Adequate pile	68
1054	1	2	C7280	278,25	556,5	A3028	Adieu divider	118
1054	1	2	C7280	278,25	556,5	A6302	Airtight injury	16
1054	1	2	C7280	278,25	556,5	A6803	Albeit soothsay	23
1054	1	2	C7280	278,25	556,5	A9814	Alert poole	36
1054	1	2	C7280	278,25	556,5	A7825	Allyn kennedy	108
1054	1	2	C7280	278,25	556,5	A5441	Alpaca roll	118
1054	1	2	C7280	278,25	556,5	A6191	Alto scrolls	65

Запись: 1 из 395270

Рисунок 0.8

ЗАДАНИЕ НА РАБОТУ

1) Изучить структуру учебной базы данных в соответствии с разделом 2.
2) Ознакомиться к теоретическими сведениями из разделов 3,4.

3) Выполнить запросы, представленные в примерах раздела 5.
4) Выполнить два упражнения из раздела 0, согласно выданному преподавателем варианту заданий.

Таблица 7.1 – Варианты заданий

№ варианта	1	2	3	4	5	6	7	8	9	10	11	12
№№ упражнений	1,7	2,8	3,9	4,10	5,11	6,12	6,7	5,8	4,9	3,10	2,11	1,12

УПРАЖНЕНИЯ

При решении упражнений 1–6 рекомендуется использовать вложенные запросы. При решении упражнений 7–12 рекомендуется использовать группировку записей.

1. (Использование одной и той же таблицы во внешнем и вложенном подзапросе.) Выдать номера и объем продаж для продавцов, которые имеют объем продаж больше среднего. Отсортировать по возрастанию объема продаж.

2. Найти фамилии и телефоны продавцов, проживающих в том же городе, где расположена фирма «Azimuth Bavis & Systems».

3. Получить список компаний, не сделавших ни одного заказа.

4. Получить все конторы, их города и штаты, объем продаж у которых больше, чем у любого индивидуального продавца. (Использовать квантор ALL)

5. Получить данные обо всех конторах, расположенных в штате Калифорния (CA), совершивших сделки в период между 17 мая 1990г. и 10 июня 1990г.

6. Вывести номера продавцов, совершавших сделки по продажи детали с шифром A9155. (Использовать квантор EXISTS).

7. Подсчитать для каждой детали общее количество сделок по ее продаже и объем продаж в денежном выражении.

8. Выбрать общую и среднюю сумму сделок для продавцов, совершивших более пяти сделок. Отсортировать по средней сумме сделки.

9. Получить список номеров срочно требующихся деталей и для каждой из них общее количество нужных единиц товара и сумму поставки. Под срочно требующейся понимается деталь, заказанная в количестве более 50 единиц.

10. Узнать общий объем продаж, призведенных каждым продавцом каждой конторы. (группировка по двум полям).

11. Получить список всех компаний, сделавших более одного заказа.

12. Получить список всех контор и для каждой из них общую сумму заказов, расположив список по убыванию общей суммы заказов. (Это требует соединения трех таблиц).

ПОРЯДОК ЗАЩИТЫ РАБОТЫ И КОНТРОЛЬНЫЕ ВОПРОСЫ

Письменный отчет по работе не делается. Защита работы производится у компьютера в форме демонстрации результатов и отдельных операций, а также ответов на контрольные вопросы из следующего перечня (на вопрос модульного контроля в письменной форме):

17. Общий алгоритм ER-проектирования реляционной базы данных. Пример
18. Генерация предварительных отношений для бинарных связей степени 1:1
19. Генерация предварительных отношений для бинарных связей степени 1:n и m:n
20. Генерация предварительных отношений для трехсторонних связей
21. Использование ролей для генерации отношений

ПРИЛОЖЕНИЕ А
ОСНОВНЫЕ РАЗЛИЧИЯ ЯЗЫКОВ JET SQL И ANSI SQL

(справочное)

Ядро базы данных Microsoft Jet SQL в основном совместимо с ANSI-89 Level 1. Однако некоторые возможности языка запросов ANSI SQL не реализованы в Microsoft Jet SQL. Аналогично, язык запросов Microsoft Jet SQL содержит зарезервированные слова и возможности, не поддерживаемые языком ANSI SQL.

Основные различия:

– Языки Microsoft Jet SQL и ANSI SQL имеют разные зарезервированные слова и типы данных.

– Другие правила применяются для конструкции Between...And, которая имеет следующий синтаксис:

выражение1 [NOT] Between значение1 And значение2

В языке Microsoft Jet SQL значение1 может быть больше, чем значение2; в языке ANSI SQL значение1 должно быть меньше значения2 или равно ему.

– В языке Microsoft Jet SQL при использовании оператора Like поддерживаются как подстановочные знаки языка ANSI SQL, так и подстановочные знаки, относящиеся к Microsoft Jet (см. таблицу А.1). Невозможно совместно использовать подстановочные знаки ANSI и Microsoft Jet. Допускается использовать только один набор знаков, их нельзя смешивать. При попытке использования подстановочных знаков языка ANSI SQL с приложением Microsoft они будут трактоваться как литералы.

Таблица А.1 – Соответствие подстановочных знаков

Символ соответствия	Microsoft Jet SQL	ANSI SQL
Любой текстовый символ	?	знак подчеркивания (_)
Любое количество символов	*	%

– В языке Microsoft Jet SQL меньше ограничений. Например, разрешается производить группировку и сортировку по выражениям.

– Язык Microsoft Jet SQL поддерживает более производительные выражения.

Язык Microsoft Jet SQL обладает следующими **расширенными возможностями:**

– Инструкция TRANSFORM обеспечивает поддержку перекрестных запросов.

– Существуют дополнительные статистические функции SQL, такие как StDev и VarP.

– Для определения параметров запроса используется описание PARAMETERS.

Язык Microsoft Jet SQL **не поддерживает** следующие возможности языка ANSI SQL.

– Ссылки в статистической функции DISTINCT. Например, в языке Microsoft Jet SQL невозможно использовать конструкцию SUM(DISTINCT *имя_столбца*).

– Использование предложения LIMIT TO *nn* ROWS для ограничения числа строк, возвращаемых запросом. Для подобного ограничения можно использовать только предложение WHERE.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Справочная система пакета Microsoft Access.
2. Боуман Дж. Практическое руководство по SQL/Дж. Боуман, С. Эмерсон, М. Дарновски. – Изд-во «Вильямс», 2001.– 336с.
3. Конспект лекций по дисциплине «Информационное обеспечение систем управления» [Электронный ресурс].– Режим доступа: Локальная сеть кафедры ТК, f:\polsov\common\Infob\LInfOb1.doc, f:\polsov\common\Infob\LinfOb2.doc.

ЛАБОРАТОРНЫЙ ПРАКТИКУМ ПО СИСТЕМЕ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ ACCESS

Методические указания

Составитель: **Шушляпин** Евгений Андреевич

В авторской редакции

Изд. № 158/19. Объём 5,75 п.л.
РИИЦМ ФГАОУ ВО "Севастопольский государственный университет"