Школа им. А.М.Горчакова
Современная версия Царскосельского лицея

А.С.Цветков

Система управления базами данных Microsoft Access

Учебное пособие для 10–11 классов

Санкт-Петербург

2018

Система Управления Базами Данных (СУБД)

Microsoft Access

Глава I. Что такое база данных?

- **База данных** – это набор записей и файлов, организованных особым образом

  Примеры простейших баз данных:

  - документы, сгруппированные по каталогам
  - электронная таблица
  - список электронных писем
  - ярлыки программ в кнопке «Пуск»

- **Реляционная модель базы данных**. В реляционной базе данных все обрабатываемые данные представляются в виде таблиц.

  - *Таблица* – информация об объектах одного типа (по строчкам).
  - *Атрибут* – часть информации об объекте, хранится в виде поля.
  - *Отношение* – способ, которым информация в одной таблице связывается с данными в другой таблице.
  - *Объединение* – объединение информации из нескольких таблиц или запросов на основе совпадающих значений определенных атрибутов.

- **Возможности СУБД**

  - *Определение данных* – ввод, облегчение ввода, контроль ввода информации.
  - *Обработка данных* – выборки, запросы, фильтрация данных.
  - *Управление данными* – указание, каким пользователям разрешено просматривать, модифицировать, добавлять данные.

- **Microsoft Access** – полнофункциональная реляционная СУБД, которая предоставляет средства разработки и управления СУБД. Access может сама организовывать доступ к данным (используя так называемое ядро *Jet-engine*), хранящимся в файлах с расширением *.mdb, при этом все объекты одной базы данных сохраняются в одном файле (некоторые СУБД используют множество файлов для хранения объектов). Access может выступать и в роли надстройки к более мощным системам, таких как серверы баз данных: Microsoft SQL Server, Oracle, MySQL и другие.

- **Запрос** – обращение к базе данных с целью извлечения нужных данных. Результат запроса к базе данных всегда есть некоторая таблица.

- **SQL** – Structured Query Language (язык структурированных запросов) – специальный язык программирования (не являющийся алгоритмическим, подобно Pascal, JavaScript), используемый в Access (и практически во всех других) базах данных для формирования запросов. Алгоритм, по которому будут найдены необходимые данные, «разработает» сама СУБД.

Глава II. Обзор Microsoft Access

Архитектура Microsoft Access

- **Таблица** – объект, использующийся для хранения данных. Каждая таблица содержит информацию о предметах определенного типа. *Поля* (*столбцы*) таблицы служат для хранения различных характеристик предмета, а каждая *запись* (*строка*) содержит сведения о конкретном предмете. Для каждой таблицы можно определить *первичный ключ* – одно или несколько полей, однозначно идентифицирующую каждую запись.

- **Запрос** – объект, позволяющий пользователю получить нужные данные из одной или нескольких таблиц. Для создания запроса можно использовать бланк QBE (Query By Example – запрос по образцу) или написать инструкцию SQL.

- **Форма** – объект, предназначенный для ввода данных, отображения их на экране или управления работой приложения. Формы часто используются для более наглядного представления данных таблиц.

- **Отчет** – объект, предназначенный для форматирования, вычисления и вывода на печать.

- **Страница доступа к данным** – объект, содержащий код HTML, обеспечивающий доступ к данным посредством Internet Explorer'а.

- **Макрос –** объект, представляющий структурированное описание одного или нескольких действий, которые автоматически выполняются в ответ на определенное событие.

- **Модуль –** объект, содержащий программы на языке Visual Basic, который может оперировать любыми другими объектами.

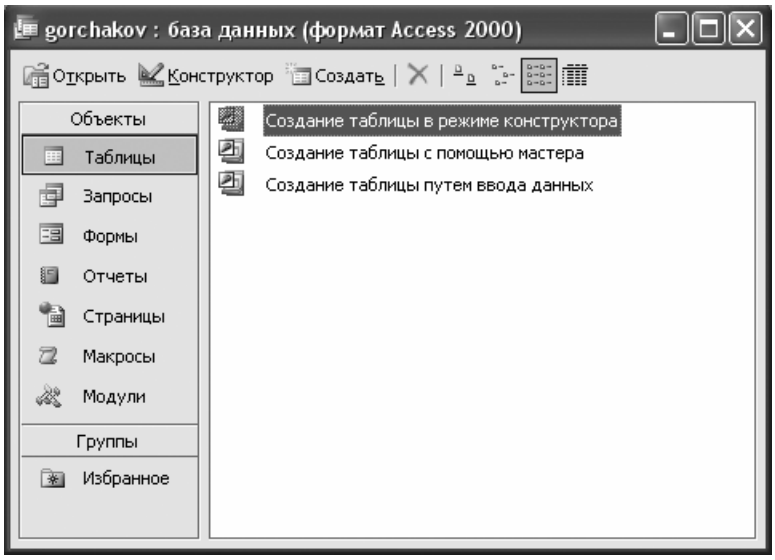


Рис. 2.1 *Создание нового объекта в базе данных*

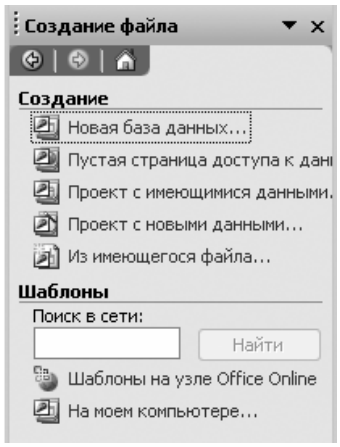


Рис. 2.2 *Создание нового файла*

Создание новой базы данных

Выполнение команды меню «Создать…» (Ctrl-N или первая кнопка на инструментальной панели) приводит к появлению в правой части рабочей области информационного окна, в котором следует выбрать пункт «Новая база данных». MS Access потребует сразу ввести имя файла (в отличие от Word). Все изменения в дальнейшем сразу заносятся в файл. Это необходимо для обеспечения многопользовательского доступа к данным.

Создание новой таблицы

Таблица – основной способ хранения данных в базе данных. Обычно вы всегда должны определить хотя бы одну таблицу. Существует три способа создания новых таблиц.

Создание таблицы в режиме конструктора

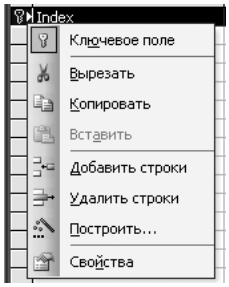


Рис. 2.4. *Задание ключевого поля*

В этом режиме (рис. 2.3) вы задаете структуру таблицы, количество и типы полей. Каждая строка в режиме конструктора определяет одно из полей таблицы. Одно из полей таблицы должно быть задано ключевым полем, которое должно однозначно определять строку таблицы (т.е. в нем не допускаются совпадающие значения для разных записей). Ключевое поле можно задать с помощью щелчка правой клавиши мыши по строке в конструкторе (рис. 2.4).

Создание таблицы в режиме конструктора является наиболее профессиональным подходом и позволяет полностью контролировать создание таблицы. Подробнее этот способ будет описан далее.

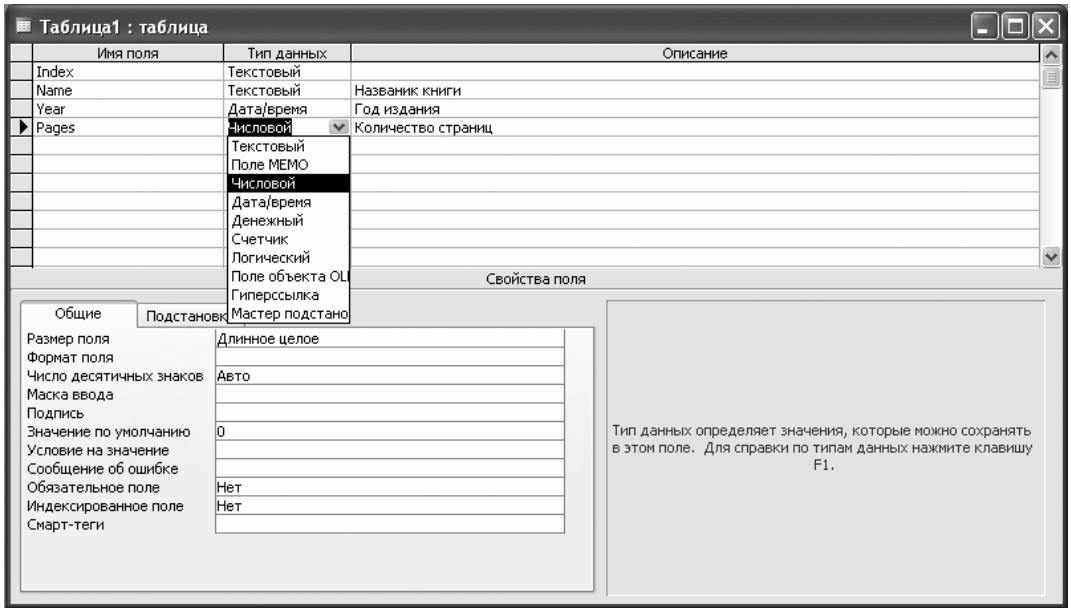


Рис. 2.3. *Создание таблицы в режиме конструктора*

Создание таблицы в режиме мастера

Специальный «мастер» (Wizard) позволяет в диалоговом режиме создать таблицу данных для наиболее часто встречающихся задач. Можно ознакомиться с работой мастера, чтобы лучше представлять возможности Access.

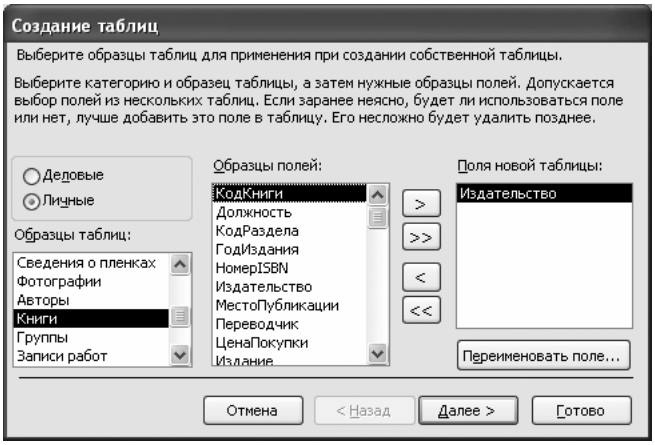


Рис. 2.5. *Создание таблицы в режиме мастера*

Создание таблицы в режиме ввода данных

Этот режим похож на создание таблицы Excel. Microsoft Access сам определяет типы полей на основе вводимых пользователем данных.

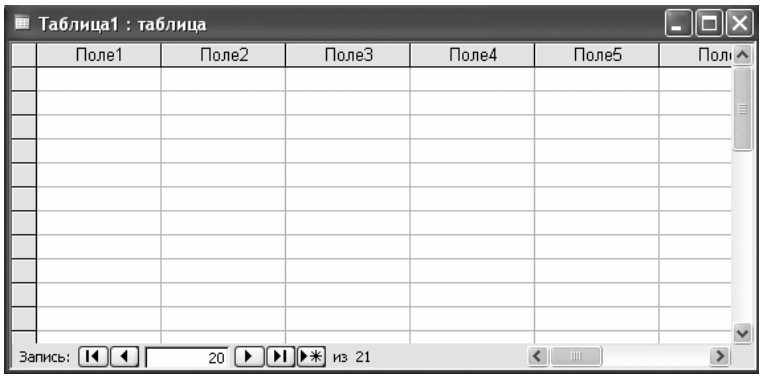


Рис. 2.6. *Создание таблицы в режиме ввода данных*

Просмотр таблицы

tbBooks : таблица

Index	Name	Year	Pages
0001	MS Access	2001	1040
0002	MS Excel	2002	976
0010	MS Word	2000	670
		0	0

Рис. 2.7. *Просмотр таблицы и ввод данных*

После создания таблицы вам будет предложено выбрать для нее имя в базе данных. Двойной щелчок по имени таблицы откроет ее для просмотра и для ввода данных. В простых таблицах ввод данных аналогичен MS Excel.

Задание 1. Создайте в режиме конструктора таблицу с именем tbClass из двух столбцов. Первый столбец текстового типа с именем Name будет содержать фамилию ученика, а второй числового типа с именем Year – год рождения. Определите поле Name как ключевое поле таблицы. Заполните таблицу данными нашего класса. (*1 балл*)

Запросы

Запрос – основной способ получения информации из базы данных. Суть запроса заключается в том, что вы определяете отображаемую информацию (в простейшем случае список полей таблицы) и задаете условия отбора (требования на значения полей). Тем самым вместо просмотра данных всей таблицы удается быстро получить только действительно необходимую информацию. Результат запроса есть также таблица, ее можно использовать в других запросах наравне с обычными таблицами. Простые запросы можно составить по технологии Query By Example, более сложные запросы создаются с помощью языка SQL.

Создание запроса в режиме конструктора

В этом режиме (рис. 2.8) вы задаете список обрабатываемых полей, условие отбора, отображать или нет это поле в результате запроса.

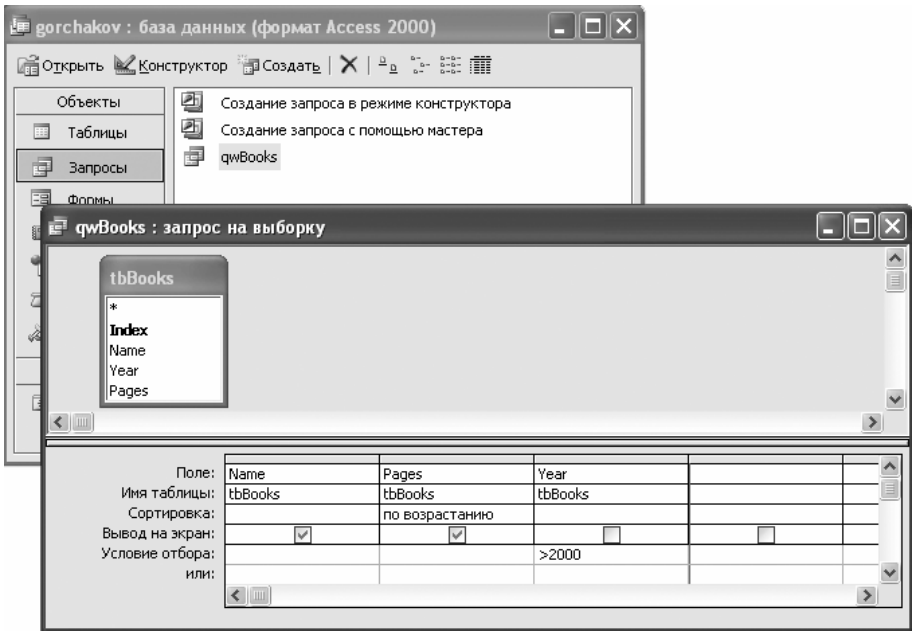


Рис. 2.8. *Окно запроса в режиме конструктора*

Создание запроса в режиме мастера

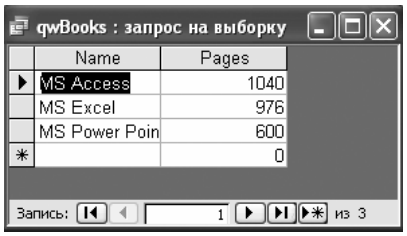


Рис. 2.9. *Результат запроса*

Режим мастера не предоставляет особых дополнительных удобств. Ради интереса можете попробовать создать запрос в режиме мастера, но при изменении макета вы перейдете так или иначе в режим конструктора.

Просмотр результата запроса

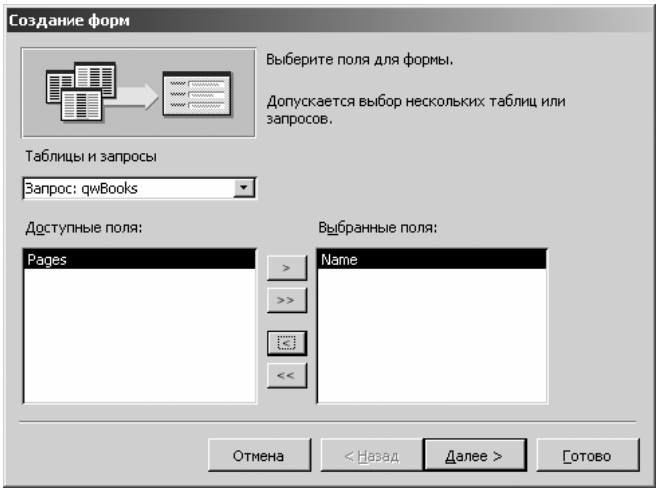
Двойной щелчок по имени запроса в списке или щелчок по кнопке «открыть» вызовет выполнение запроса. Результат запроса – таблица с записями, удовлетворяющих заданным условиям. Отображаются только заданные поля (Рис. 2.9). Обратите внимание на навигационную панель внизу окна. Она присутствует в очень многих окнах, использование ее просто, но эффективно.

Задание 2. Составьте запрос на отбор только фамилий учеников определенного года рождения, отсортируйте по алфавитному порядку. (*1 балл*)

Формы

Формы – диалоговые панели, облегчающие просмотр, ввод и редактирование данных. Конечно, можно вводить и просматривать данные в таблицах, но при большом числе строк или полей это становится крайне неудобным, кроме того при редактировании данных возрастает вероятность ошибки.

Создание формы в режиме мастера



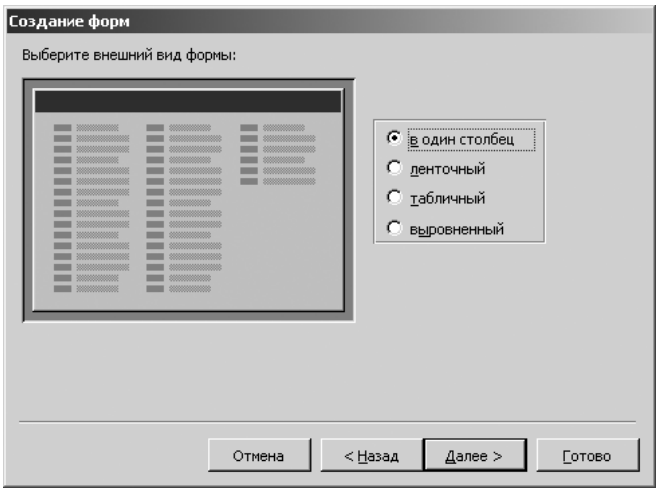


Рис. 2.10 и 2.11. *Создание формы в режиме мастера*

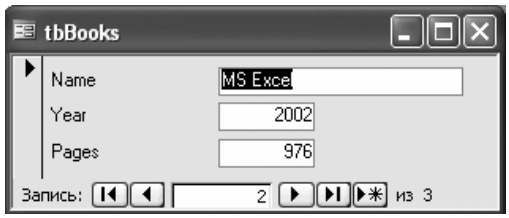


Рис. 2.12. *Окно формы в режиме ввода данных*

Этот режим является достаточно удобным средством. Пользователь выбирает источник данных (имя таблицы или запроса) список полей, доступных для отображения, общий макет формы и она готова. Всегда можно открыть форму в режиме конструктора и поменять ее дизайн, и даже содержание. После создания и выбора имени формы, она появляется в списке форм. Двойной щелчок мыши открывает форму (рис 2.12) для просмотра и редактирования данных. Для добавления новых данных следует нажать последнюю кнопку в инструментальной панели (треугольник со звездочкой).

Создание формы в режиме конструктора

Режим конструктора позволяет полностью управлять всеми свойствами формы. Он будет рассмотрен позже.

Задание 3. Создайте форму в режиме мастера для просмотра данных таблицы tbClass. Поэкспериментируйте с разными видами форм. (*1-2 балла*)

Задание 4. Создайте документ в MS Word, который бы объяснял смысл терминов: СУБД, реляционная модель БД, таблица, запрос, SQL, форма. (*1 балл*).

Отчеты

Если формы предназначены в основном для ввода и просмотра данных на экране, то отчеты удобны для вывода на печать. Отчеты также можно создавать в режиме *конструктора* и *мастера*. На первых этапах мы воспользуемся мастером. Построенный отчет затем можно открыть в режиме конструктора и внести дополнительные изменения. В бланке отчета можно производить форматирование внешнего представления результатов запроса или просто данных таблицы в режиме, похожим на обычный текстовый процессор или издательскую систему.

Алгоритм работы мастера прост. Пользователю необходимо ответить на несколько вопросов: количество полей и источник данных (рис. 2.13), необходимо ли делать группировки (например, по годам издания книг) (рис. 2.14).

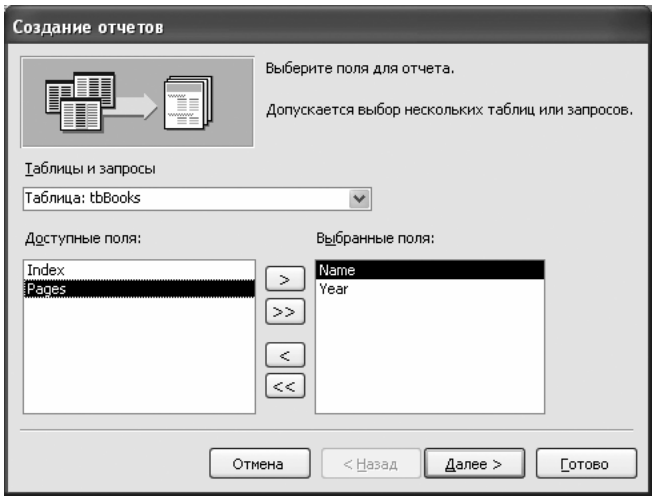
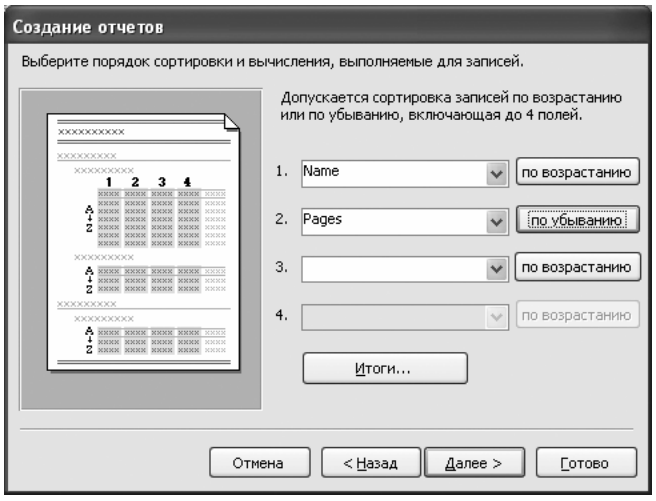


Рис. 2.13 и 2.14. *Создание отчета в режиме мастера*

Затем можно указать порядок сортировки полей (рис. 2.15) и общий вид отчета (рис. 2.16)



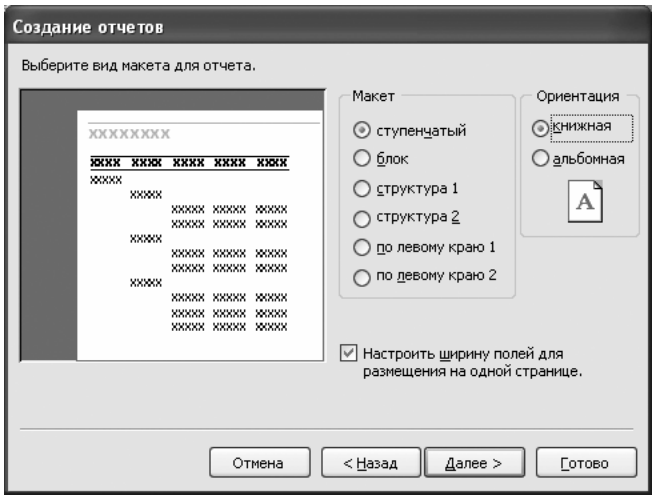
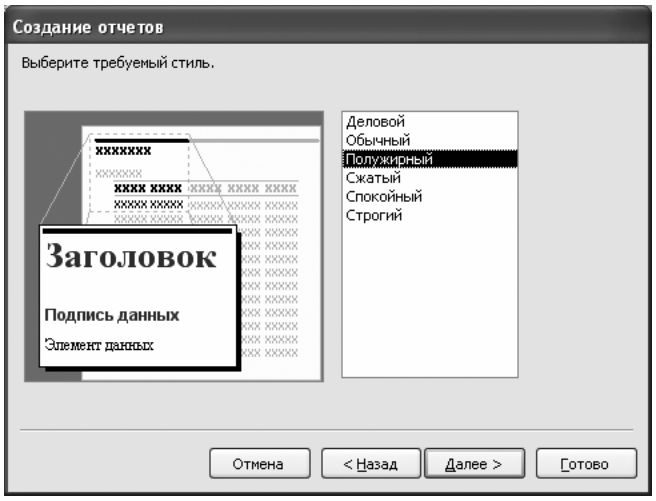


Рис. 2.15 и 2.16. *Создание отчета в режиме мастера*

Последние два вопроса определяют вид заголовка (рис. 2.17) и имя отчета (рис. 2.18).



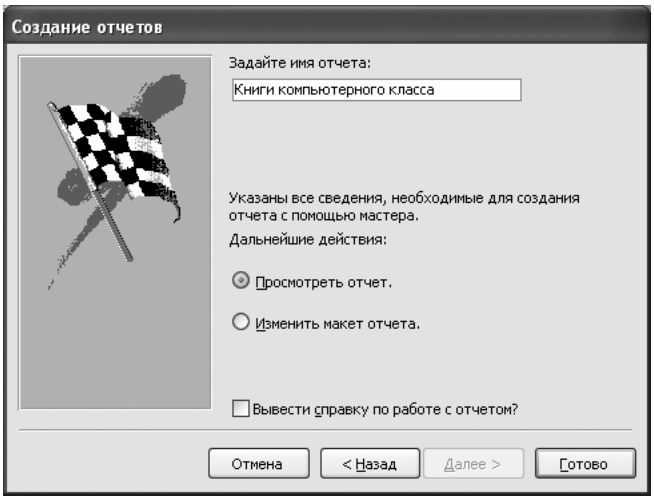
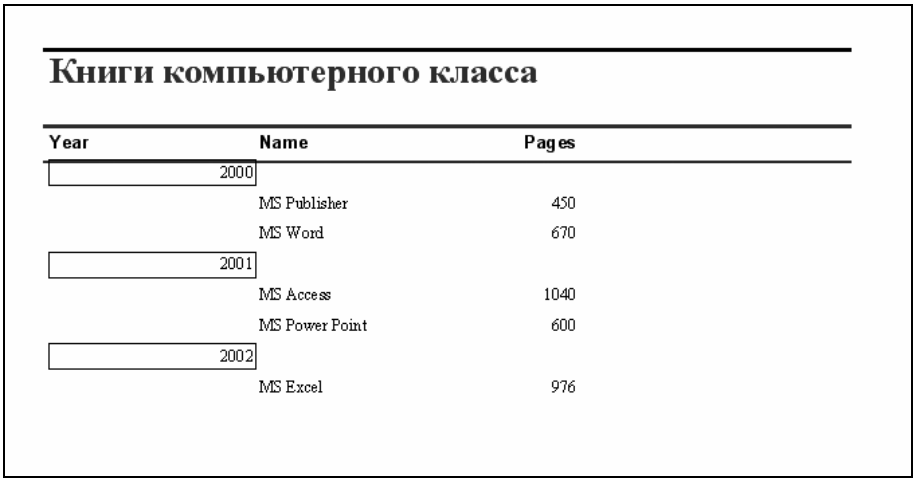


Рис. 2.17 и 2.18. *Создание отчета в режиме мастера*

Готовый отчет представлен на рис. 2.19.



Книги компьютерного класса

Year	Name	Pages
2000		
	MS Publisher	450
	MS Word	670
2001		
	MS Access	1040
	MS Power Point	600
2002		
	MS Excel	976

Рис. 2.19. *Просмотр готового отчета*

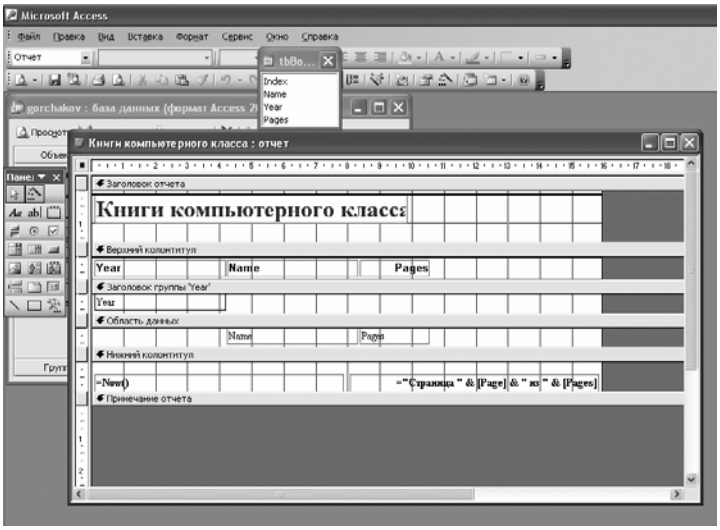


Рис. 2.20. *Отчет в режиме конструктора*

Созданный отчет можно открыть в режиме конструктора, изменить порядок полей, шрифты, добавить иллюстрации и т.п. Подробнее об этом позже.

Задание 5. Изучите все возможные в виды отчетов, предоставляемые мастером. Для этого заполните ваши таблицы данных хотя бы 15-20 записями.

(*1-2 балла*)

Web-страницы доступа к данным

Для пользователей *локальной сети* (не всех пользователей Интеренет – это другая технология, мы рассмотрим ее позже) легко можно создать web-страницу доступа к данным, которая будет правильно работать только в броузере Internet Explorer версии не ниже 5. Это удобный способ доступа к данным, так как обычным пользователям не нужно будет осваивать интерфейс MS Access, вместо этого они воспользуются простым интерфейсом IE. Страницы доступа будут жестко привязаны к местонахождению файла базы данных.

Создание страницы в режиме мастера

Мастер по созданию страниц предложит ответить на четыре стандартных вопроса, аналогичных вопросам по созданию отчета.

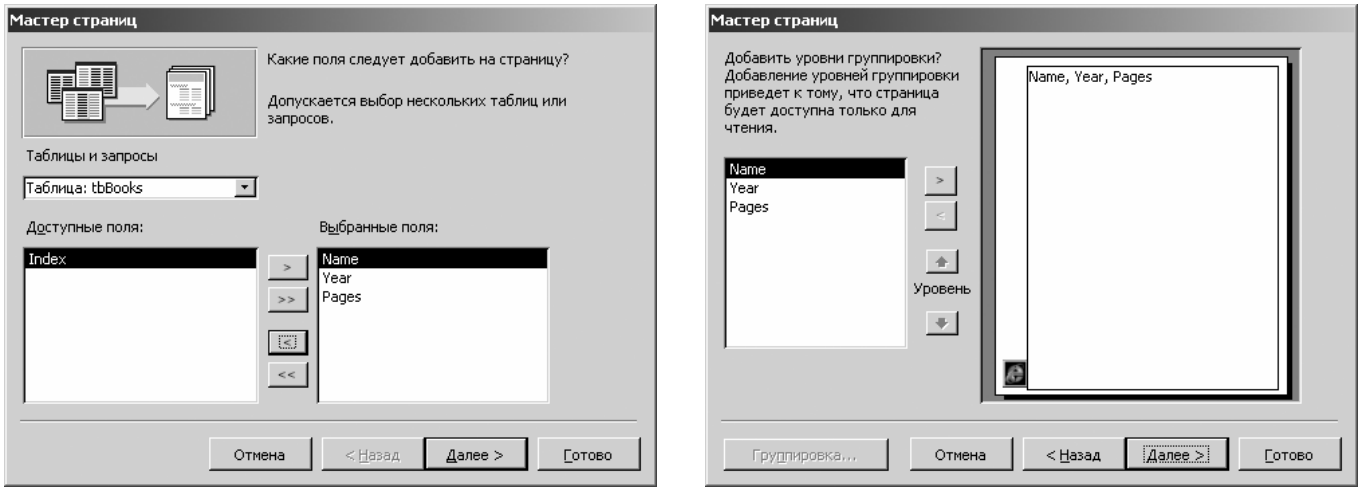


Рис. 2.21 и 2.22 *Создание страницы доступа к данным в режиме мастера*

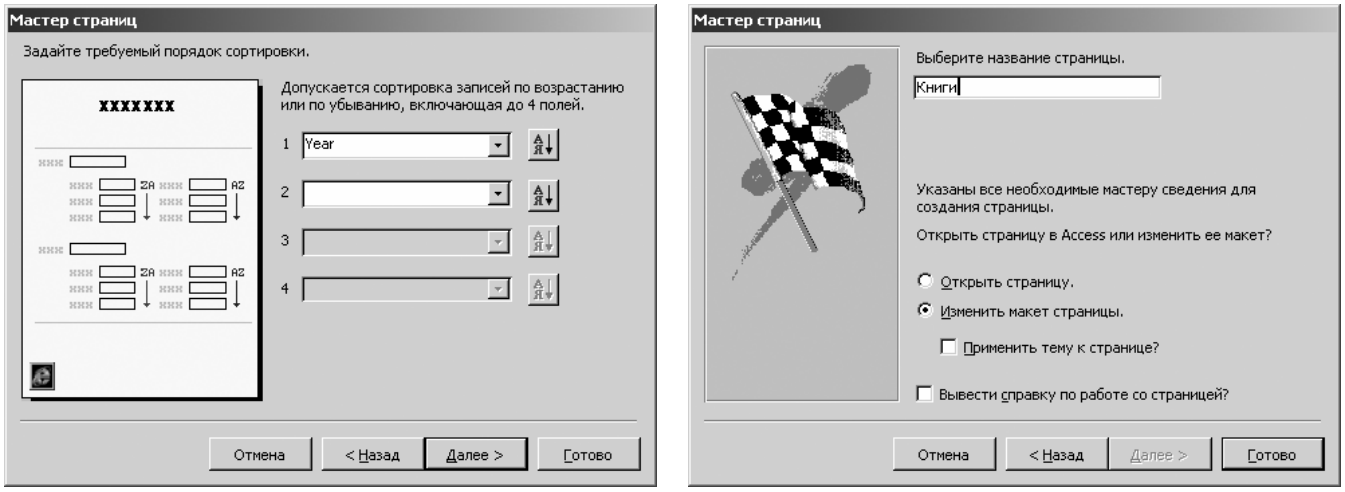


Рис. 2.23 и 2.24 *Завершение создания страницы доступа к данным*

После ответа на последний вопрос (рис. 2.24) можно открыть страницу и отредактировать ее содержание и дизайн в конструкторе страницы (рис. 2.25).

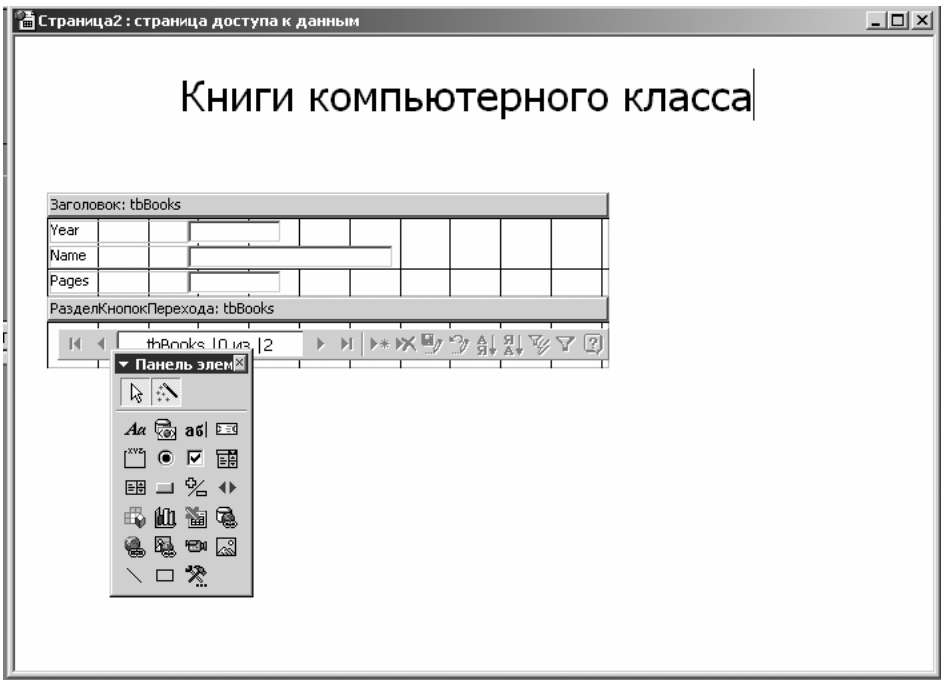


Рис. 2.25. *Страница доступа к данным в режиме конструктора*

После создания страницы и сохранения ее в виде html-файла, ее можно открыть не только в MS Access, но и в броузере Internet Explorer (рис. 2.26).

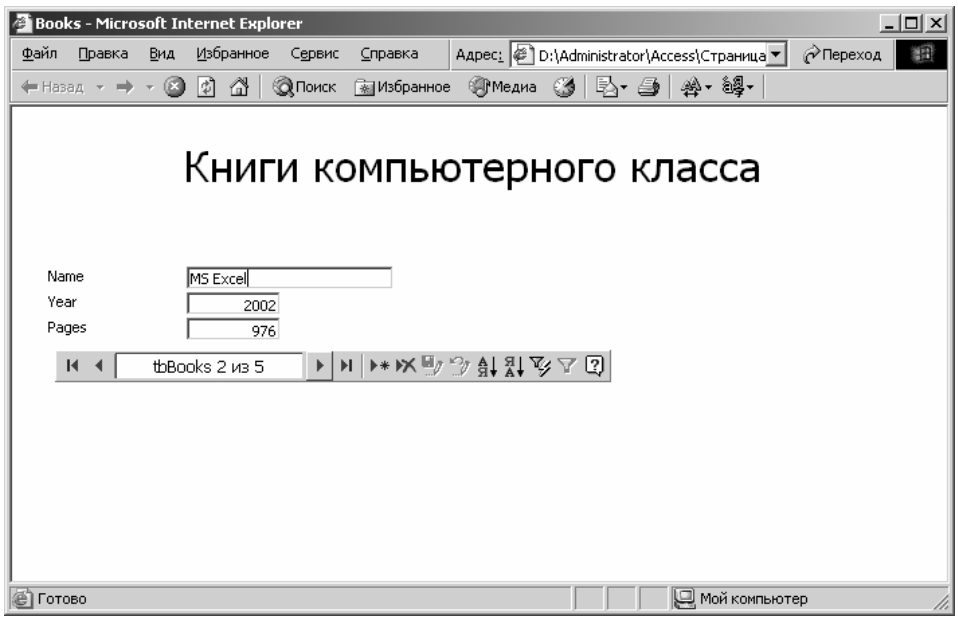


Рис. 2.26. *Страница доступа к данным в окне Internet Explorer'а*

Задание 6. Создайте страницу доступа в режиме мастера для просмотра данных tbClass. (*1 балл*)

Макросы и модули на этапе знакомства с MS Access мы рассматривать не будем.

Задание 7. Найдите информацию о 4-х правилах нормализации таблиц в реляционной базе данных. Запишите их в документ MS Word (*2 балла*).

Глава III. Принципы построения базы данных

Основные этапы разработки приложения

1. **Уточнение задач**. Какие задачи должна решать БД, какие задачи она должна решать в будущем.
2. **Последовательность выполнения задач**. Определить подзадачи. Сгруппировать их по каким-либо принципам.
3. **Анализ данных**. Составление списка всех данных с их типами. Оценить объем данных.
4. **Определение структуры данных**. Группировка данных по таблицам. Выполнение *нормализации* таблицы (см. далее).
5. **Разработка пользовательского интерфейса**. Определить внешний вид форм или web-страниц.
6. **Создание приложения**. Возможно п. 5 окажется недостаточным. Необходимо будет выполнить уточняющие действия.
7. **Тестирование и усовершенствование**.

Задание 8. Вы получили задание на построение БД (если нет, считайте ваша задача – библиотечная БД). Сформулируйте около десятка задач, которые будет решать данная БД. Сохраните список этих задач в документе MS Word (после добавления результатов заданий 9 и 10 этот документ следует распечатать и вложить в папку).

(*1 балл*)

Данные и информация

- Данные – статические значения, хранящиеся в таблицах БД. *Данные хранятся.*
- Информация – сведения, запрашиваемые пользователем, и предоставляемые ему в удобном виде. *Информация запрашивается*.

Отбор необходимых данных

- Для каждой задачи необходимо составить список необходимых данных с пометкой об использовании. Для этого используйте пять букв:
    - I – Input, входной параметр;
    - O – Output, выходной параметр;
    - U – Update, изменяемый;
    - D – Delete, удаляемый;
    - C – Calculate, вычисляемый.

Составление рабочего бланка для задачи

Рабочие бланки – основа первоначальной структуры приложения. С их заполнения начинается разработка БД. Рис. 3.1 представляет пример заполнения такого бланка для задачи занесения информации о книге в библиотеку. В полях должна быть указана следующая информация:

- Элемент данных – название поля в таблице
- Использование – что можно делать с полем (см. выше)
- Описание – подробное описание назначения данного поля
- Объект – где будет размещаться эта информация (имя таблицы, запроса)

Последнее поле можно на первом этапе не заполнять.

Задание 9. Заполните для каждой задачи по соответствующей таблице.

(*1/2 балла за каждую таблицу*)

Рабочий бланк №1 – **Задачи**			
Название задачи: добавление книги в БД			
Краткое описание: внесение данных о новой книге, проверка входных данных			
Связанные задачи: редактирование и удаление записи о книге			
Элемент данных	Использование	Описание	Объект
Автор	I	Фамилия, Имя	табл. Авторы
Название	I	Название книги	табл. Книги
Номер	O	Номер книги по школьному каталогу	табл. Книги
...			

Рис. 3.1. *Заполнение рабочего бланка для задачи «Добавление книги»*

Анализ данных

На следующем этапе необходимо хорошо понять, сколько и каких таблиц будет в БД. Какие связи могут быть между таблицами. Например, неразумно каждый раз вводить имя автора (и всю связанную с этим информацию: годы жизни, национальность, направление, портрет и т.п.) в каждой строке таблицы «Книги». Вместо этого лучше создать отдельную таблицу «Авторы», а в соответствующее поле в таблице «Книги» использовать ссылку на строку в таблице «Авторы».

Типы связей

Надо продумать типы связей между таблицами. Возможные варианты связей:

- **Один-ко-многим** – **наиболее часто используемый тип связи между таблицами**. В такой связи каждой записи в таблице A может соответствовать несколько записей в таблице B (поля с этими записями называют внешними ключами), а запись в таблице B не может иметь более одной соответствующей ей записи в таблице A.

  Пример: у одной книги не может быть больше, чем одно издательство, но у одного издательства наверняка больше, чем одна книга.
- При связи **Один-к-одному** запись в таблице A может иметь не более одной связанной записи в таблице B и наоборот. Этот тип связи используют не очень часто, поскольку такие данные могут быть помещены в одну таблицу. Связь с отношением Один-к-одному применяют для разделения очень широких таблиц, для отделения части таблицы в целях ее защиты, а также для сохранения сведений, относящихся к подмножеству записей в главной таблице.

  Пример: у каждой книги только одна обложка, но держать ее фотографию в основной таблице не выгодно, с точки зрения скорости доступа и размера файла базы.
- При связи **Многие-ко-многим** одной записи в таблице A может соответствовать несколько записей в таблице B, а одной записи в таблице B – несколько записей в таблице A. Такая схема реализуется только с помощью третьей (связующей) таблицы, ключ которой состоит по крайней мере из двух полей, одно из которых является общим с таблицей A, а другое – общим с таблицей B.

  Пример: у одной книги может быть больше, чем один автор. А один автор может написать нескольких книг.

Задание 10. Заполните для каждой таблицы список полей по образцу рис 3.2.
(*1/2 балла за каждую таблицу*)

Бланки таблиц выложены в \\LAMBDA\ACCESS\Постановка задачи.doc

Скопируйте их к себе в «Мои документы», заполните, после проверки преподавателем распечатайте и вложите в папку. Постарайтесь выполнить правила нормализации при проектировании вашей базы данных.

Рабочий бланк №2 – **Объекты**			
Имя объекта: Авторы			
Краткое описание: Список авторов книг			
Связанные объекты:	**Имя**	**Тип связи**	
	Книги	Один	
	Страны	Многие	
Элемент данных	Тип	Описание	Условие на значения
Фамилия	Текстовый (30)	Фамилия Имя (Отчество, если есть)	Обязательное (ключ)
ГодРожд	Числовой	Год рождения	Необязательное
ГодСмерти	Числовой	Год смерти	Необязательное
Страна	Текстовый (40)	Национальность или страна	Обязательное
...			

Рис. 3.2. *Заполнение рабочего бланка для описания объектов*

Ключи

Для того чтобы легко было однозначно идентифицировать запись (строчку) таблицы, часто вводится дополнительное сервисное поле (обычно числовое), уникальное для каждой записи в таблице.

Нормализация

Основная задача, возникающая при проектировании БД, – устранение возможной избыточной информации в таблицах БД. Решение этой задачи позволяет уменьшить размер базы и ускорить поиск по ней. При соблюдении правил нормализации легко организовать связь между таблицами «**один-ко-многим**» и «**многие-ко-многим**»

Нормализацию таблиц можно выполнять до уровня 2, 3, 4. Чем выше уровень, тем лучше структура БД, эффективнее ее хранение и поиск по ней, но сложнее заполнение и изменение записей. Поэтому, зачастую нормализацию проводят не до самого глубокого уровня, а ограничиваются уровнем 2.

1. Таблица находится в **первой нормальной форме** (1НФ) тогда и только тогда, когда ни одна из ее строк не содержит в любом своем поле более одного значения и ни одно из ее ключевых полей не пусто. Обычно в СУБД все таблицы по построению будут находиться, по крайней мере, в 1НФ.
2. Таблица находится во **второй нормальной форме** (2НФ), если она удовлетворяет определению 1НФ и все ее поля, не входящие в первичный ключ, связаны полной функциональной зависимостью с первичным ключом. Это означает, что первичный ключ однозначно определяет значение остальных полей. Запись с разными значениями первичного ключа не могут быть идентичными (у них не могут совпадать полностью все остальные поля).
3. Таблица находится в **третьей нормальной форме** (3НФ), если она удовлетворяет определению 2НФ и не одно из ее неключевых полей не зависит функционально от любого другого неключевого поля. Например, следует исключить одно из двух полей: город или телефонный код, поскольку одно из них однозначно определяет другое. Следует создать отдельную таблицу: «список соответствия городов и телефонных кодов», а в основную таблицу включить что-то одно.

Глава IV. Работа с таблицами

В этой главе мы подробнее рассмотрим все этапы создания таблиц базы данных

Типы данных полей

MS Access поддерживает 9 типов данных с их модификациями

Таблица 4.1 *Типы данных*

Тип данных	Использование	Размер
Текстовый	Алфавитно-цифровые данные	до 255 байт
Поле MEMO	Алфавитно-цифровые данные – предложения, тексты, абзацы	До 65 536 знаков
Числовой	Числовые данные, есть модификации для целых, вещественных разной точности	1, 2, 4, 8, 12, 16 байт
Дата/время	Даты и время	8 байт
Денежный	Данные о денежных суммах, 4 зн. после запятой	8 байт
Счетчик	Уникальное целое, используется для создания ключевого поля	4 байта
Логический	Да/Нет	1 бит
Поле объекта OLE	Картинки, диаграммы и пр. объекты	До 1 Гбайта
Гиперссылка	Адрес в Интернет/Интранет	До 2048 байт

- *Свойство «Размер поля»*. Для текстовых полей указывается максимальная длина (по умолчанию – 50), а для числовых – тип данных:

Байт	Целое от 0 до 255
Целое	Целое от -32768 до +32767
Длинное целое	Целое от -2 147 483 648 до 2 147 483 648
Одинарное с плав. точкой	Аналог типу single в Pascal. от $-3.4\cdot10^{38}$ до $+3.4\cdot10^{38}$. Точность – 7 знаков.
Двойное с плав. точкой	Аналог типу double в Pascal. от $-1.797\cdot10^{308}$ до $+1.797\cdot10^{308}$. Точность – 15 знаков.
Код репликации	16 байтовый год для операции репликации
Действительное	18-значное десятичное число в указанном диапазоне. Занимает 12 байт.

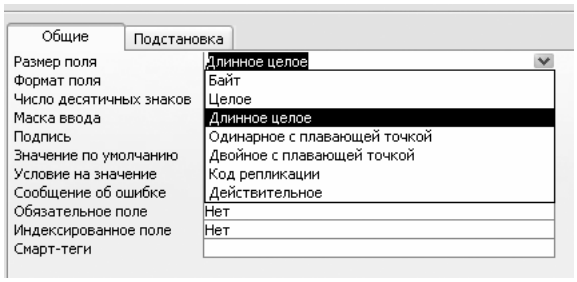


Рис 4.1. *Установка размера поля*

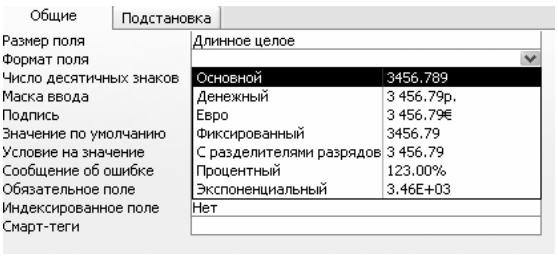


Рис. 4.2 *Выбор формата поля*

- *Свойство «Формат поля»*. Определяет внешний вид данных: денежный, с разделителями разрядов, процентный, экспоненциальный и т.п. (см. рис. 4.2).
- Для вещественных чисел разумно задать свойство «*Число десятичных знаков*».

- Свойство «*Маска ввода*» будет рассмотрено подробнее далее.
- Свойство «*Подпись*» может определить более содержательное название поля, которое Access будет использовать в отчетах и формах.
- Свойство «*Значение по умолчанию*» задает, естественно, значение по умолчанию для поля. По умолчанию это 0 для числовых данных, ложь для логических, а для остальных типов –**Null** – специальное значение, соответствующее неопределенности поля.
- Свойство «*Условие на значение*» будет рассмотрено подробнее далее.
- Свойство «*Сообщение об ошибке*» указывает текст сообщения, которое Access будет выводить, если вводимые данные не отвечают условию на значение.
- Свойство «*Обязательное поле*» устанавливайте в значение Да, если вы не допускаете, чтобы в этом поле хранилось значение Null.
- Свойство «*Пустые строки*» устанавливайте в значение Да, если вы допускаете, чтобы в этом поле хранилось значение пустой строки (“”). Это отличается от Null! Null – не задано, а пустая строка – это текстовая строка длиной в ноль, в ней нет ни одного символа.
- Свойство «*Индексированное поле*» ускоряет доступ к хранящимся в нем данным. Можно также и запретить наличие повторяющихся значений в таком поле.
- Свойство «*Подстановка*» будет рассмотрено подробнее далее.

Задание простых условий на значение полей

Для обеспечения корректности вводимых данных используют задание условий на значение. Оно задается в общем случае выражением. Обычно это знаки отношений (<, <=, > как в Pascal) объединенные знаками логических операций (AND, OR). Например, чтобы не допустить ввода неправильного года издания, в качестве условия можно задать >= 1950 AND <=2005. При задании условия полезно определить и поле «сообщение об ошибке», в этом случае при нарушении значения вместо сообщения (4.5) будет сообщение (4.6). Приведенное условие можно записать в другом синтаксисе: BETWEEN 1950 AND 2005.

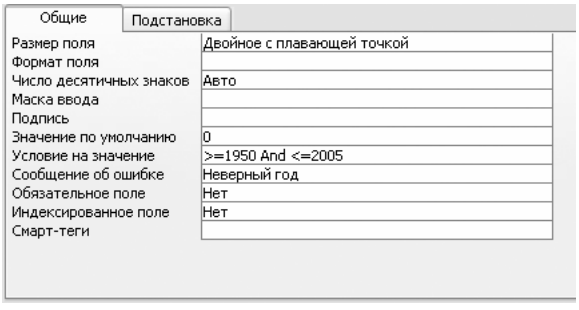


Рис. 4.3. *Определение условия*

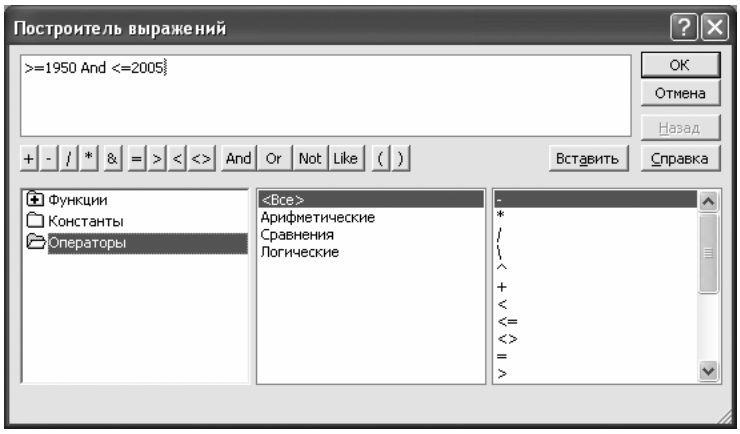


Рис. 4.4. *Построитель выражений*

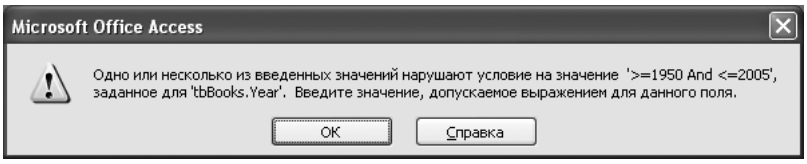


Рис. 4.5. *Сообщение о неверном значении*

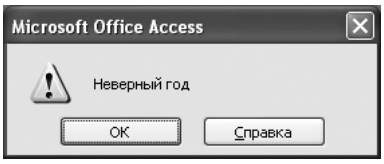


Рис. 4.6. *Пользовательское сообщение*

Для работы с символьными выражениями можно использовать оператор IN для множественного сравнения, например IN (″Сакнт-Петербург″, ″Лодейное поле″, ″Подпорожье″).

Очень удобен оператор LIKE, который позволяет сравнить вводимый текст с некоторым шаблоном.

Таблица 4.2. *Символы шаблона в операторе LIKE*

Символ	Описание
?	Один произвольный символ
*	Любое число (включая нулевое) произвольных символов
#	Одна произвольная цифра

Можно задать условие, чтобы определенная позиция текста содержала только символы, определенные в списке. Для этого используется список символов в квадратных скобках [A-Z] – только большие буквы от A до Z. Можно сделать наоборот: буква не должна входить в список, тогда используется восклицательный знак: [!0-9] – в позиции должна быть не цифра.

Таблица 4.3. *Примеры оператора LIKE*

Условие	Описание
LIKE ″Gor*″	Строка должна начинаться с Gor
LIKE ″??00##″	Строка начинается с двух произвольных символов, затем обязательно идут два нуля, а потом две цифры
LIKE ″[!0-9ABCDEF]*#″	Строка начинается с любого символа, кроме цифры, A, B, C, D, E, F; затем идет любое количество знаков, строка заканчивается цифрой.

Задание 10. Задайте для каждого поля (для которых это возможно) в ваших таблицах условие на значение.

(*1/2 балла за каждое условие*)

Маска ввода

Маска ввода позволяет контролировать ввод данных в таблицу. И не только контролировать, но и упрощать процесс ввода. Маску используют в том случае, когда вводимые данные должны содержать определенные символы в некоторых позициях вводимой строки. Самым простым и ярким примером таких данных являются номера телефонов. Маска ввода должна обеспечить возможность вводить только цифры номера, а остальные символы (скобки вокруг кода города, дефис между цифрами номера) будут добавляться автоматически. Маска задается с помощью символов, приведенных в табл. 4.4. Вводить маску ввода можно непосредственно в строке маски (рис. 4.7).

Для того чтобы сформировать маску ввода, можно использовать Мастера масок ввода. Чтобы ввести маску ввода для этого поля, щелкните мышью по ячейке свойства Маска ввода. Нажмите небольшую кнопку с тремя точками (рис. 4.8), которая появится справа (эта кнопка называется кнопкой Построителя и будет встречаться еще во многих местах), и дальше следуйте указаниям мастера.

Задание 11. Задайте для некоторого поля (для которого это целесообразно) в ваших таблицах маску ввода.

(*1 балл за каждую маску*)

Таблица 4.4. *Символы, используемые для задания маски ввода*

Символ маски	Описание
0	В данную позицию должна быть введена цифра. Знаки плюс (+) и минус (-) не допускаются.
9	В данную позицию должна быть введена цифра или пробел. Знаки плюс (+) и минус (-) не допускаются.
#	В данную позицию должна быть введена цифра, пробел, знаки плюс (+) или минус (-).
L	В данную позицию должна быть введена буква.
?	В данную позицию может быть введена буква или пробел.
A	В данную позицию должна быть введена буква или цифра.
a	В данную позицию должна быть введена буква, цифра или пробел.
&	В данную позицию должен быть введен произвольный символ или пробел.
C	В данную позицию может быть введен произвольный символ или пробел. Если пользователь ничего не введет, Access не занесет в эту позицию никаких данных.
.	Десятичный разделитель (зависит от региональных установок в окне Язык и стандарты Панели управления Windows).
,	Разделитель групп разрядов (зависит от региональных установок в окне Язык и стандарты Панели управления Windows).
:-/	Разделители в значениях даты и времени (зависят от региональных установок в окне Язык и стандарты Панели управления Windows).
<	Преобразует все символы справа к нижнему регистру.
>	Преобразует все символы справа к верхнему регистру.
!	Указывает, что маску нужно заполнять справа налево. Этот символ следует использовать в том случае, когда символы в левой части маски являются необязательными. Его можно помещать в любой позиции маски.
\	Указывает, что следующий символ необходимо рассматривать в качестве постоянного символа, даже если он является специальным символом маски. Например, \A будет выводить в маске букву A.
"литерал"	Вместо того чтобы многократно использовать символ обратного слэша (\), можно просто заключить любой литерал в двойные кавычки.

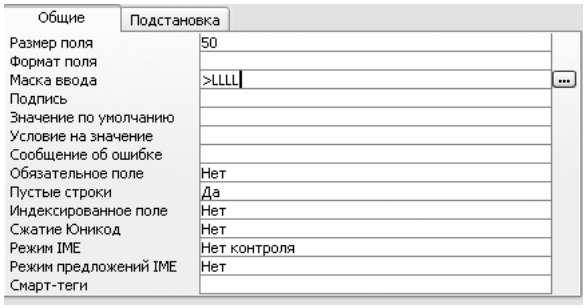


Рис. 4.7. *Маска ввода >LLLL, обеспечивающая ввод только четырех букв, которые будут еще и переводиться в заглавные*

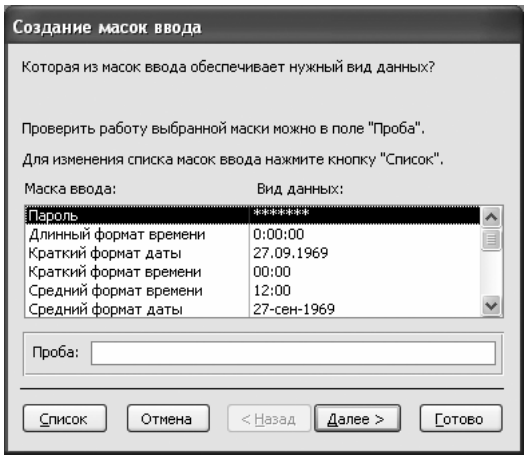


Рис. 4.8. *Окно мастера масок*

Использование мастера подстановок (краткое знакомство)

Если какое-то из полей имеет набор фиксированных значений, то можно поступить таким образом: создать отдельную таблицу их этих значений, например, список издательств (рис. 4.9). В конструкторе таблицы, использующей эту подтаблицу, у поля «Издательство» выбрать тип данных «*Мастер подстановок*». Появившейся мастер подстановок задаст несколько вопросов, ответ на которые приведет к правильному созданию подстановки (рис. 4.10-4.15).

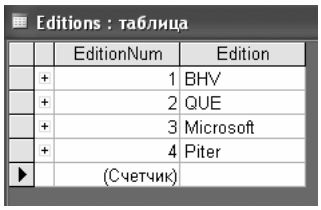


Рис. 4.9. *Таблица со списком издательств, в качестве ключевого поля используется счетчик.*

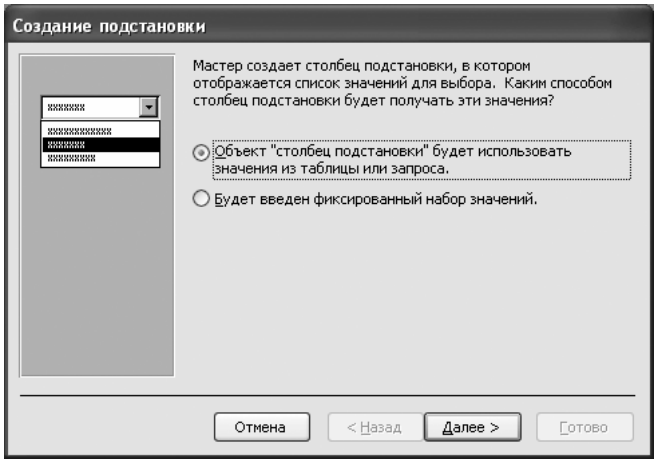


Рис. 4.10. *Выбор объекта подстановки.*

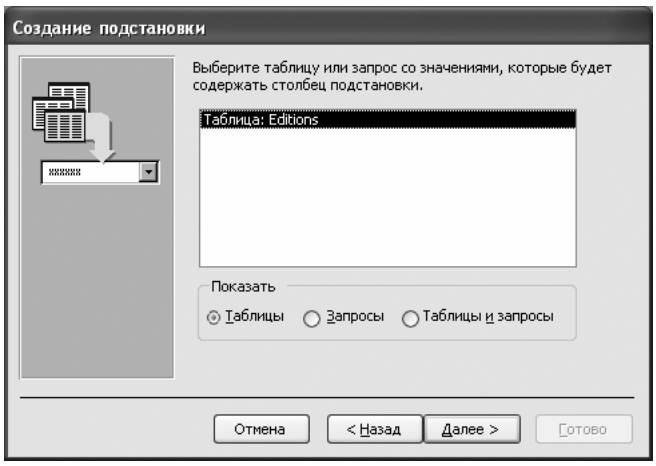


Рис. 4.11. *Выбор имени таблицы, содержащие данные подстановки.*

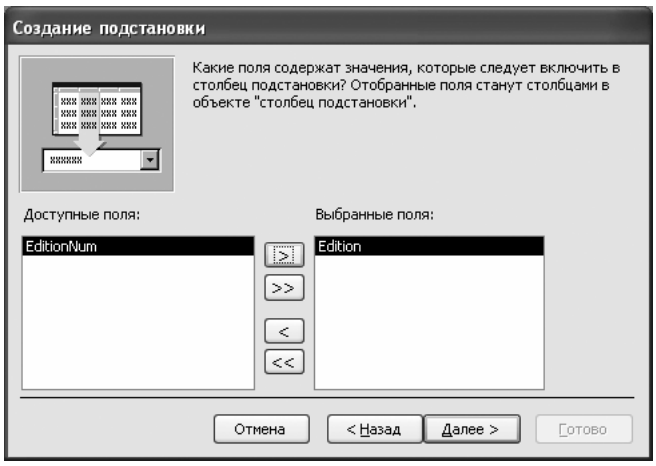


Рис. 4.12. *Выбор полей таблицы, для подстановки (пока одно).*

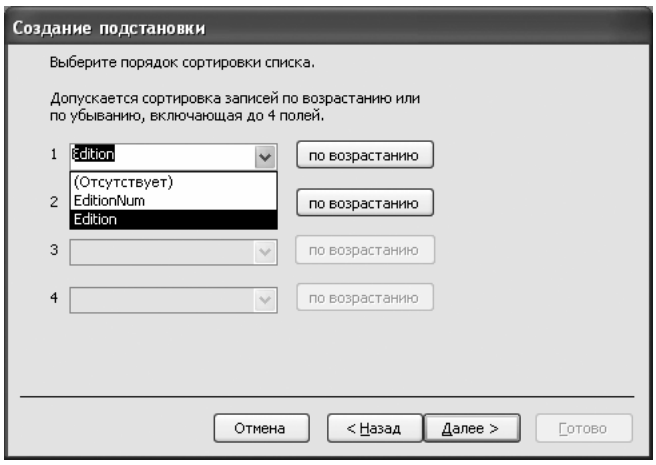


Рис. 4.13. *Выбор порядка сортировки списка.*

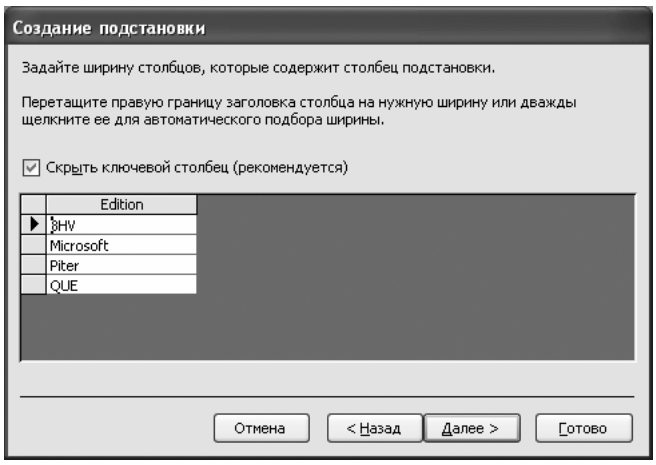


Рис. 4.14. *Установка размера столбцов.*

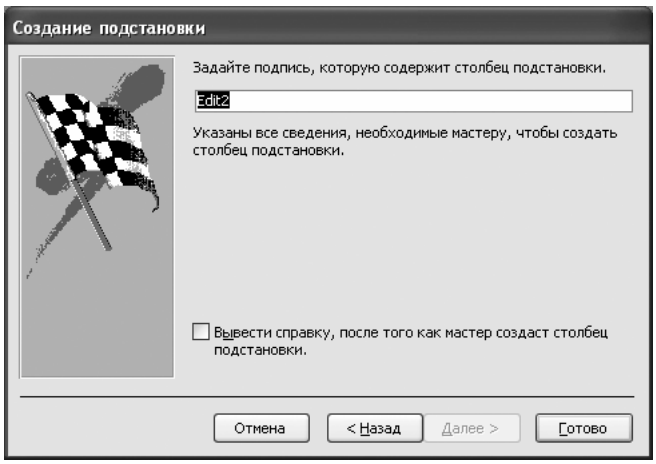


Рис. 4.15. *Выбор имени столбца подстановки.*

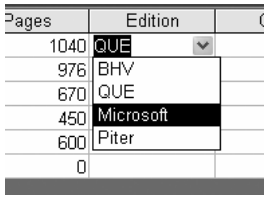


Рис. 4.16. *Появление списка выбора в режиме вода данных в таблицу.*

После определения такой подстановки, при вводе данных в таблицу появится раскрывающийся список, в котором можно сразу выбрать нужное значение поля.

Задание 12. Задайте для поля (для которого это целесообразно) подстановку.
(*1 балл за каждую таблицу подстановки*)

Создание первичного ключа

Выше неоднократно упоминалось понятие ключевого поля. Ключевое поле — это одно или несколько полей, комбинация значений которых однозначно определяет каждую запись в таблице. Если для таблицы определены ключевые поля, то Microsoft Access предотвращает дублирование или ввод пустых значений в ключевое поле. Ключевые поля используются для быстрого поиска и связи данных из разных таблиц при помощи запросов, форм и отчетов.

В Microsoft Access можно выделить три типа ключевых полей: счетчик, простой ключ и составной ключ. Рассмотрим каждый из этих типов.

Для создания ключевого поля типа *Счетчик* необходимо в режиме Конструктора таблиц:

1. Включить в таблицу поле счетчика.
2. Задать для него автоматическое увеличение на 1.
3. Указать это поле в качестве ключевого путем нажатия на кнопку Ключевое поле на панели инструментов Конструктор.

Если до сохранения созданной таблицы ключевые поля не были определены, то при сохранении будет выдано сообщение о создании ключевого поля. При нажатии кнопки Да будет создано ключевое поле счетчика с именем Код и типом данных Счетчик.

Для создания *простого ключа* достаточно иметь поле, которое содержит уникальные значения (например, коды или номера). Если выбранное поле содержит повторяющиеся или пустые значения, его нельзя определить как ключевое. Для определения записей, содержащих повторяющиеся данные, можно выполнить запрос на поиск повторяющихся записей. Если устранить повторы путем изменения значений невозможно, следует либо добавить в таблицу поле счетчика и сделать его ключевым, либо определить составной ключ.

Составной ключ необходим в случае, если невозможно гарантировать уникальность записи с помощью одного поля. Он представляет собой комбинацию нескольких полей. Для определения составного ключа необходимо:

1. Открыть таблицу в режиме Конструктора.
2. Выделить поля, которые необходимо определить как ключевые.
3. Нажать кнопку Ключевое поле на панели инструментов Конструктор таблиц.

Замечание

Для составного ключа существенным может оказаться порядок образующих ключ полей. Сортировка записей осуществляется в соответствии с порядком ключевых полей в окне Конструктора таблицы. Если необходимо указать другой порядок сортировки без изменения порядка ключевых полей, то сначала нужно определить ключ, а затем нажать кнопку Индексы на панели инструментов Конструктор таблиц. Затем в появившемся окне Индексы нужно указать другой порядок полей для индекса с именем Ключевое поле.

Создание и использование индексов

С целью ускорения поиска и сортировки данных в любой СУБД используются индексы. Индекс является средством, которое обеспечивает быстрый доступ киданным в таблице на основе значений одного или нескольких столбцов. Индекс представляет собой упорядоченный список значений и ссылок на те записи, в которых хранятся эти значения. Чтобы найти нужные записи, СУБД сначала ищет требуемое значение в индексе, а затем по ссылкам быстро отбирает соответствующие записи. Индексы бывают двух типов: простые и составные. Простые индексы представляют собой индексы, созданные по одному столбцу. Индекс, построенный по нескольким столбцам, называется составным. Примером составного индекса может быть индекс, построенный по столбцам "Фамилия" и "Имя".

Однако применение индексов приносит не только преимущества, но и недостатки. Главным среди них является тот, что при добавлении и удалении записей или при обновлении значений в индексном столбце требуется обновлять индекс, что при большом количестве индексов в таблице может замедлять работу. Поэтому индексы обычно рекомендуется создавать только для тех столбцов таблицы, по которым наиболее часто выполняется поиск записей. Во многих СУБД (например, FoxPro) индексы хранятся в отдельных файлах и являются предметом заботы разработчиков, т. к. при нарушении индекса поиск данных выполняется некорректно. В Microsoft Access индексы хранятся в том же файле базы данных, что и таблицы и другие объекты Access. Индексировать можно любые поля, кроме MEMO-полей, полей типа Гиперссылка и объектов OLE.

Чтобы создать простой индекс, необходимо:

1. Открыть таблицу в режиме Конструктора.
2. Выбрать поле, для которого требуется создать индекс.
3. Открыть вкладку Общие и выбрать для свойства Индексированное поле значение Да (Допускаются совпадения) или Да (Совпадения не допускаются) (рис. 4.17).

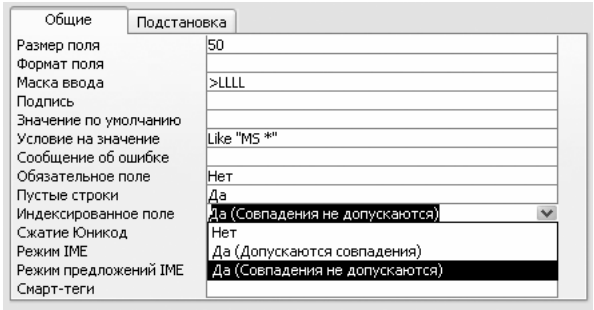


Рис. 4.17. *Задание индексированного поля.*

Задание 13. Продумайте список индексированных полей для вашей базы данных.
(*1 балл за обоснованный индекс*)

Определение связей

Создав несколько таблиц вам нужно сообщить Access, как они связаны друг с другом. В дальнейшем Access сможет автоматически связывать эти таблицы при использовании в запросах, формах, страницах доступа к данным или отчетах.

Связывание таблиц является одной из самых важных операций при построении базы данных

Рассмотрим такую задачу. Построить базу данных, например, фотографий. Запись о каждой фотографии (таблица PHOTOS) должна хранить:

1. Имя файла (FILE)
2. Название фотографии (NAME)
3. Место съемки (PLACE)
4. Дату съемки

Для организации такой таблицы создадим дополнительную таблицу PLACES (рис. 4.19), в которой будем хранить места съемок (их ведь не очень много, по крайней мере, меньше фотографий). В таблице PHOTOS используем Мастер подстановок, чтобы не вводить вручную места съемки (рис. 4.18). Если в последнем поле «Ограничиться списком», указать «Да», то непосредственно во время ввода данных в таблицу PHOTOS будет невозможно указать иные места съемки, нежели указанные в таблице PLACES.

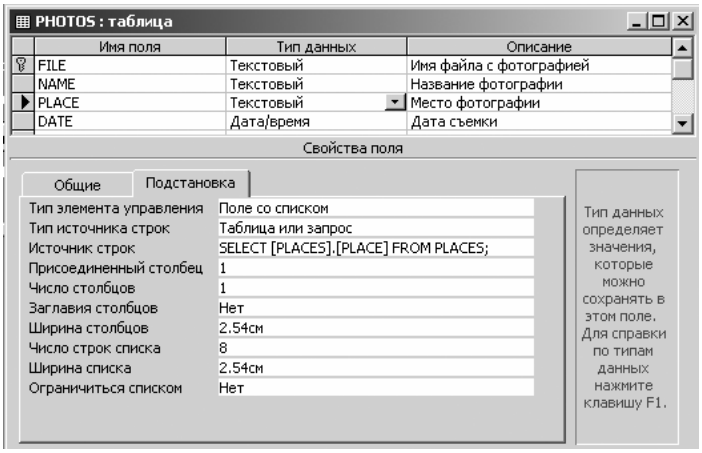


Рис. 4.18. *Определение подстановки для места съемки*

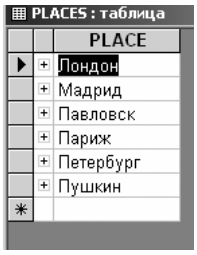


Рис. 4.19. *Таблица мест съемки*

После заполнения таблицы фотографий (рис. 4.20) в таблице мест съемки можно узнать, к каким строкам таблицы PHOTOS ссылаются строки таблицы PLACES.

Рис 4.20. *Заполнение таблицы фотографий*

Рис 4.21. *Раскрытый вид таблицы мест съемки*

Включим режим «Схема данных» (рис. 4.22). На открывшейся схеме данных щелкните два раза мышью по линии, соединяющей таблицы. Отметьте пункты «Обеспечение целостности данных», «Каскадное обновление связанных полей», «Каскадное удаление связанных полей» (рис. 4.23). Будет создана связь между таблицами с типом отношения «один-ко-многим», что будет отмечено знаками 1 и ∞.

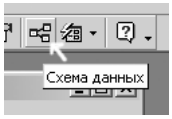


Рис. 4.22.
Кнопка «Схема данных»

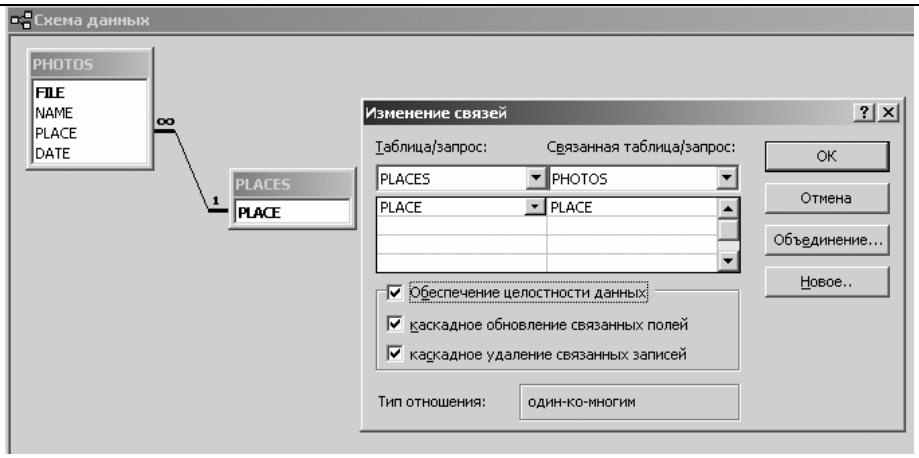


Рис. 4.23. *Связь двух таблиц*

Решим теперь следующую задачу. Дополним базу таблицами, которые позволяли ли бы узнать, кто на какой фотографии изображен. Непосредственно в таблицу PHOTOS эти данные внести нельзя, так как на разных фотографиях может быть изображено весьма различное число людей. Мы не можем предусмотреть одно поле, которое бы хранило эту информацию (нарушение правил нормализации). Заведем таблицу FAMILIES, в которой будем хранить фамилии людей, могущих быть изображенными на фотографиях (рис. 4.24). Следует учесть, что один человек может мыть изображен на многих фотографиях, и наоборот, на одной фотографии может быть изображено много людей, т.е. мы имеет дело с соотношением «многие-ко-многим». Для того, чтобы установить соответствие между списком фамилий и списком фотографий прибегают к построению дополнительной таблицы, мы назовем ее RELATONS. Она будет состоять из трех полей: номер строчки «Код» (счетчик), FAMILY – подстановка из таблицы FAMILIES, FILE – подстановка из таблицы PHOTOS (рис. 4.25).

Рис. 4.24.
Таблица FAMILIES

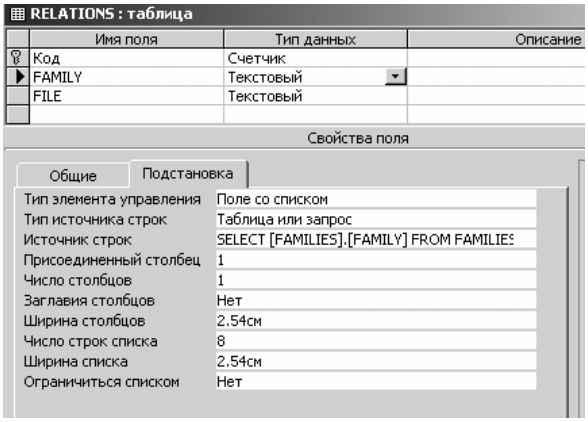


Рис. 4.25. *Структура таблицы RELATIONS*

Рис. 4.26.
Данные таблицы RELATIONS

Можно начать заполнять таблицу RELATIONS, каждая строчка которой будет описывать соотношение между фамилией и именем файла фотографии (рис. 4.26).

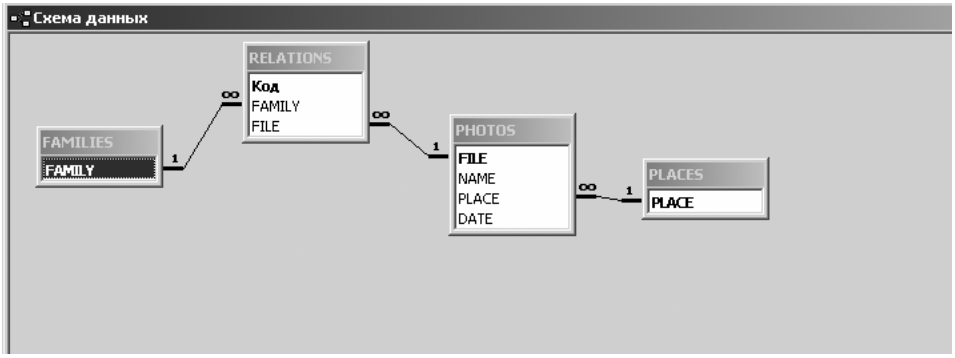


Рис. 4.27. *Схема данных базы фотографий.*

Перейдем к схеме данных. Двойным щелчком мыши по новым связям откройте диалоговые окна и установите флажки, аналогично рис. 4.23. Общая структура базы готова и мы может составить например такой запрос «хочу получить список всех мест, где был Быков», либо «список всех фотографий, на которых изображен Гильденберг». Для этого войдем в Конструктор запросов. Добавим таблицы, которые нам будет нужны для получения данных (рис. 4.28). Укажем поле FAMILY из таблицы FAMILIES с условием значение "Быков" и поле PLACE из таблицы PHOTOS (рис. 4.29). Запрос готов, после назначения ему имени, двойной щелчок по нему выполнит запрос (рис. 4.30).

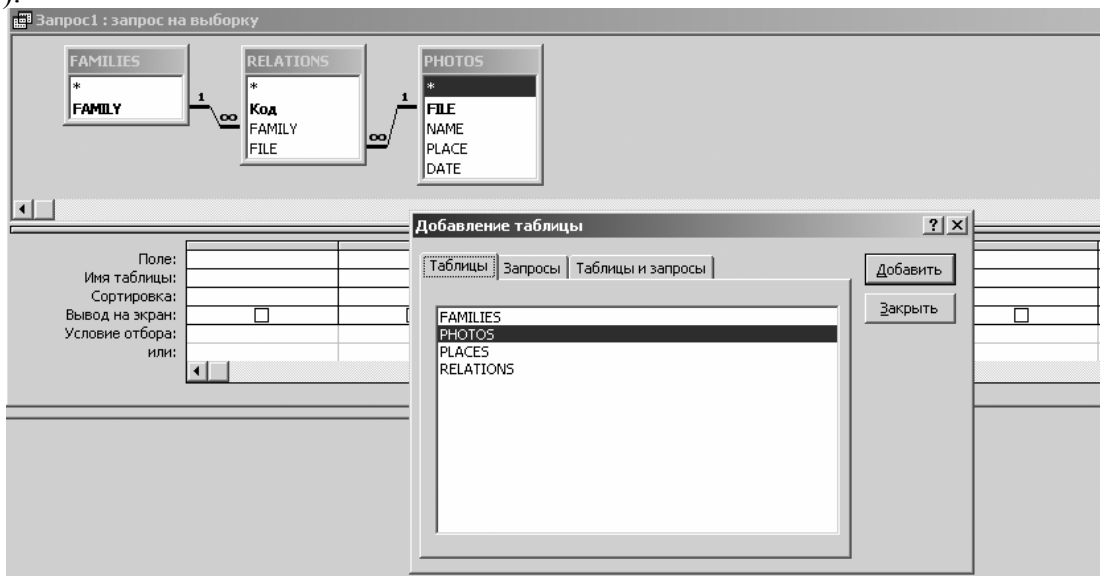


Рис. 4.28. *Добавление таблиц в конструкторе запросов.*

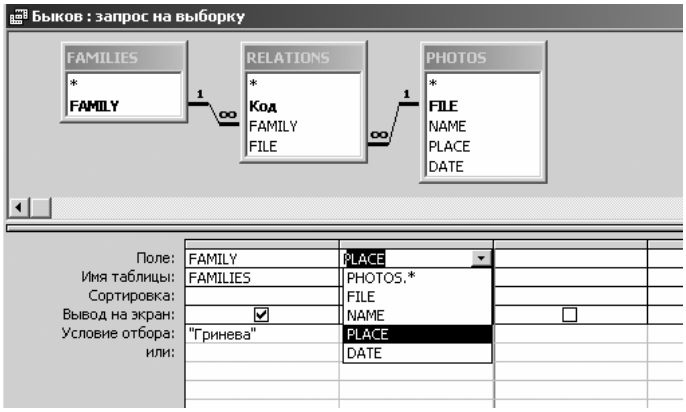


Рис. 4.29. *Построение запроса.*

Рис. 4.30. *Выполнение запроса.*

На самом деле запрос строится на языке SQL, который позволяет делать значительно больше, чем простые запросы, сконструированные визуально. Если в конструкторе запроса нажать правую клавишу мыши и выбрать «Режим SQL» (рис. 4.31), то в открывшемся окне увидите текст запроса (рис. 4.32)

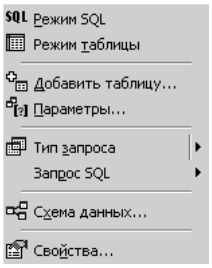


Рис. 4.31. *Установка режима SQL*

```
SELECT FAMILIES.FAMILY, PHOTOS.PLACE
FROM PHOTOS INNER JOIN (FAMILIES INNER JOIN RELATIONS ON
FAMILIES.FAMILY = RELATIONS.FAMILY) ON PHOTOS.FILE =
RELATIONS.FILE
WHERE (((FAMILIES.FAMILY)="Быков"));
```

Рис. 4.32. *Текст запроса на языке SQL*

В ближайшем будущем мы познакомимся с SQL подробнее. Однако уже сейчас можно продемонстрировать всю мощь этого языка. Например, в результате запроса может получиться результат, представленный на (рис 4.33), содержащий повторяющиеся строки, поскольку этот запрос запрашивает места съемок каждой фотографии, на которой изображен Быков. Мы хотим избавится от повторяющихся строк. В режиме SQL постави слово DISTINCT после оператора SELECT (рис. 4.34). После этого в ответе на запрос будут только уникальные строки.

Рис. 4.33. *Ответ с дублированными строками*

```
SELECT DISTINCT FAMILIES.FAMILY, PHOTOS.PLACE
FROM PHOTOS INNER JOIN (FAMILIES INNER JOIN RELATIONS ON
FAMILIES.FAMILY = RELATIONS.FAMILY) ON PHOTOS.FILE =
RELATIONS.FILE
WHERE (((FAMILIES.FAMILY)="Быков"));
```

Рис. 4.34. *Измененный текст запроса на языке SQL*

Определение связей в окне «Схема данных»

Можно определять связи между таблицами непосредственно в окне «Схема данных». Допустим, мы хотим создать таблицу COUNTRIES (рис. 4.35), содержащую поля COUNTRY и CAPITAL (рис. 4.36).

Рис. 4.35. *Таблица COUNTRIES в конструкторе*

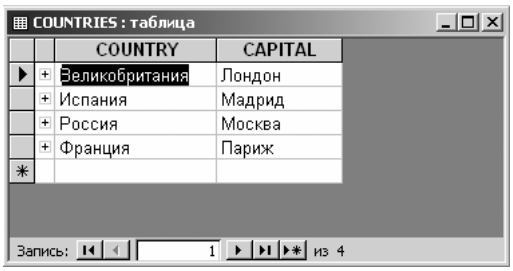


Рис. 4.36. *Заполнение таблицы COUNTRIES*

Добавим в таблицу PLACES в конструкторе поле COUNTRY и пока не будем его заполнять. Откроем окно «Схема данных», перетянем мышкой поле COUNTRY из таблицы PLACES в таблицу COUNTRY. Появится диалог (рис. 4.37), в котором будет предложено создать новую связь. Практически всегда имеет смысл установка флага «Обеспечение целостности данных», в

противном случае будет отсутствовать контроль связи, в поля таблицы можно будет вводить значения, которых нет в связанной таблице. После создания связи, в поле COUNTRY таблицы PLACES будет невозможно вводить данные, отсутствующие в связанной таблице COUNTRIES (рис. 4.38). Таким образом, связь между таблицами может быть установлена через механизм подстановки или в окне «Схема данных», в этом случае не будет выпадающего списка значений.

Рис. 4.37. *Создание связи*

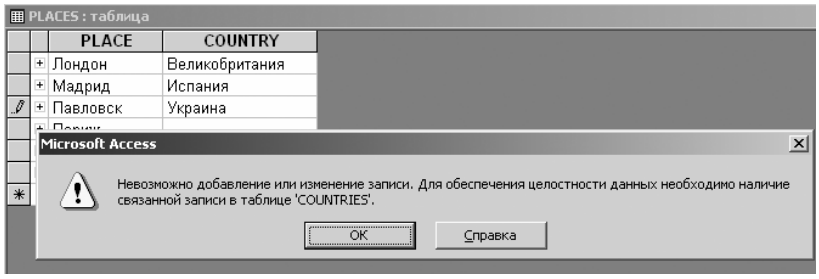


Рис. 4.38. *Сообщение о нарушении связи*

Задание 14 (важное). На основании созданных вами ранее таблиц в Word'е постройте схему вашей собственной базы данных. Определите окончательно, сколько будет таблиц и полей в каждой таблице, как они будут связаны между собой. Вспомните правила нормализации таблиц. Можете использовать только правила подстановки, в большинстве случаев это достаточно для организации связи.

(*5–10 баллов*)

Манипуляция с таблицами

Совет: так как все данные БД хранятся в одном файле, вы можете всегда сделать копию этого файла, чтобы в случае необходимости восстановить БД.

Сжатие БД: В процессе работы с файлом БД он становится фрагментированным и сильно разрастается. Чтобы убрать всю устаревшую информацию из файла БД и уменьшить его размер иногда следует «сжимать» базу данных. Это делает пункт меню *Сжать и восстановить базу данных* из подменю *Служебные программы* в меню *Сервис* (рис. 4.39).

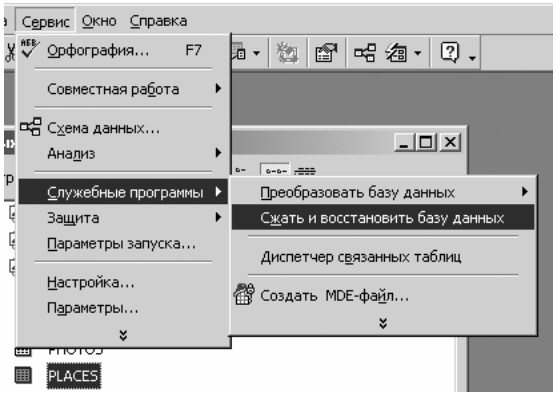


Рис. 4.39. *Сжатие базы данных*

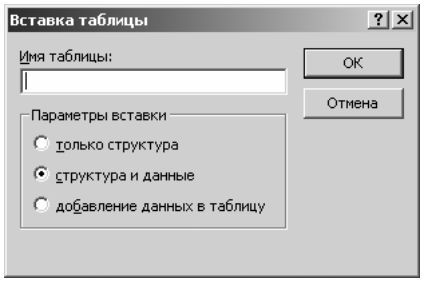


Рис. 4.40. *Вставка сохраненной таблицы*

Копирование таблицы: В случае сохранения одной таблицы, можно скопировать ее (Ctrl-C) и вставить копию (рис. 4.40), будет предложено ввести новое имя для копии таблицы.

Удаление, переименование таблиц не представляет трудностей. Для переименования в Access можно использовать «горячую клавишу» F2.

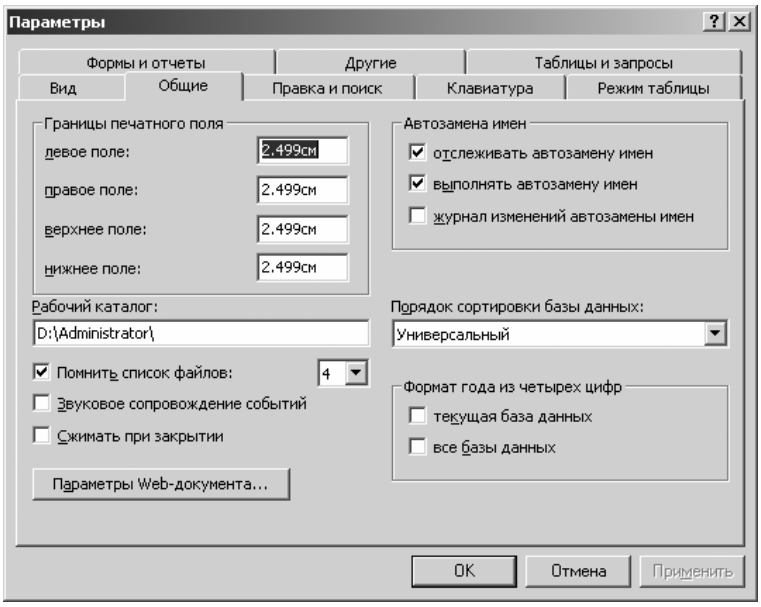


Рис. 4.41. *Включение режима автозамены*

Переименование полей при правильно спроектированной базе данных не представляет особых трудностей. Но при наличии связей переименование связанных полей может повлечь определенные затруднения. В большинстве случаев выручает режим автозамены, который должен быть включен в диалоге *Сервис-Параметры*, закладка *Общие* (рис. 4.41)

Добавление, удаление полей не должно также представлять сложностей.

Для *перемещения поля* в конструкторе щелкните мышью по области выделения (так чтобы появился черный треугольничек) и перетащите поле в желаемую позицию (рис. 4.42).

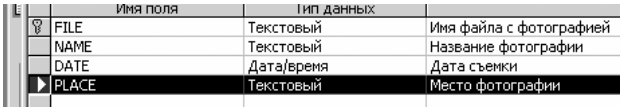


Рис. 4.42. *Отметка строки*

Изменение типа данных поля достаточно просто. Сделать это можно прямо в конструкторе. Если в таблице есть уже данные, то они будут преобразованы в соответствии с новым типом. Результаты этого преобразования иногда могут привести к потере данных в столбце. Тоже самое касается и *изменения длины поля*.

Если в вашей таблице присутствуют повторяющиеся значения поля, это может свидетельствовать о плохой нормализации БД. Можно попробовать запустить Мастер анализа таблиц (рис. 4.43), который поможет разбить одну таблицу на несколько (рис. 4.44). Но в простых случаях, все-таки рекомендуется самостоятельно продумать структуру БД.

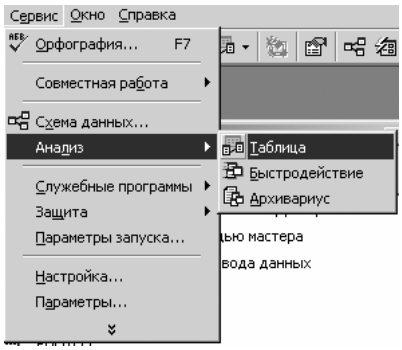


Рис. 4.43. *Запуск мастера анализа таблиц*

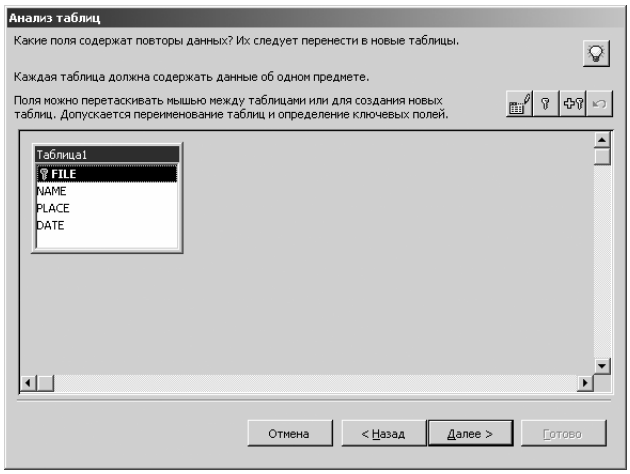


Рис. 4.44. *Мастер анализа таблиц*

Изменение макета таблицы (рис 4.45) бывает необходимо для более наглядного представления данных. Например, очень полезным может оказаться установление ширины столбца в соответствии с шириной данных (рис. 4.46, 4.47).

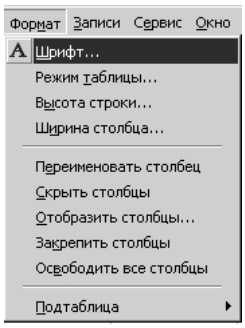


Рис. 4.45. *Меню формат*

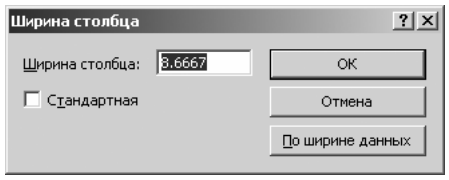


Рис. 4.46. *Установка ширины столбца*

	FILE	NAME	PLACE	DATE
+	P00001	Биг Бен	Лондон	25.09.2004
+	P00002	Тауэр	Лондон	25.09.2004
+	P10021	Двоорец	Павловск	10.11.2003
+	P10023	Прадо	Мадрид	12.10.2003
+	P10045	Стадион	Павловск	23.10.2003
+	P10055	Стадион	Мадрид	21.12.2002

Рис. 4.47. *Вид таблицы, после установления ширины столбца по ширине данных.*

При большом числе полей иногда полезно скрыть часть столбцов (правая клавиша мыши по столбцу) (рис. 4.48) и отобразить скрытые столбцы (меню *Формат*) (рис. 4.49). Можно также настроить общий вид таблицы, наличие линий сетки, фон и т.п. (рис. 4.50).

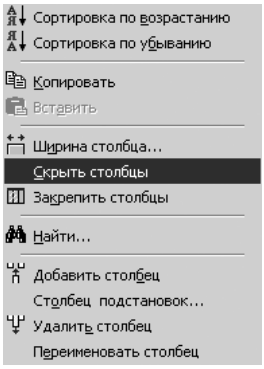


Рис. 4.48. *Локальное меню столбца*

Рис. 4.49. *Диалог отображения столбцов*

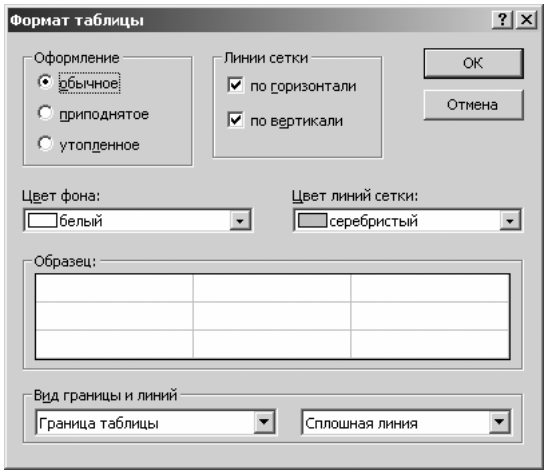


Рис. 4.50. *Диалог формата таблицы*

Access позволяет отображать в текущей таблице данные из других связанных таблиц. Щелкните мышью по знаку +, это приведет к развороту данных из связанной таблицы. Можно развернуть все подтаблицы, используя соответствующий пункт меню *Формат* (рис. 4.51).
При добавлении новой записи полезно знать назначения некоторых сочетаний клавиш (рис. 4.52)

PHOTOS : таблица

	FILE	NAME	PLACE	DATE
+	P00001	Биг Бен	Лондон	25.09.2004
−	P00002	Тауэр	Лондон	25.09.2004

Код	FAMILY
1	Акинтьев
3	Гильденберг
5	Гринева
(Счетчик)	

+	P10021	Двоорец	Павловск	10.11.2003
+	P10023	Прадо	Мадрид	12.10.2003
+	P10045	Стадион	Павловск	23.10.2003
+	P10055	Стадион	Мадрид	21.12.2002

Рис. 4.51. *Работа с подтаблицами*

Ctrl + ;	Ввод текущей даты
Ctrl + :	Ввод текущего времени
Ctrl + Alt + Пробел	Ввод значения по умолчанию
Ctrl + '	Ввод такого же знач., как и в пред. зап.
Ctrl + Enter	Вставка символа новой строки
Ctrl + +	Добавление новой записи
Ctrl + –	Удаление текущей записи

Рис. 4.52. *Сочетания клавиш для ввода данных*

Сортировка и поиск данных в таблице

Сортировка по одному столбцу выполняется элементарно с помощью соответствующих двух кнопок на панели инструментов (найдите самостоятельно). Для сортировки по нескольким столбцам (например, сначала по дате, а по том по фамилии) выберите команду *Расширенный фильтр* из пункта *Фильтр* меню *Записи*. После формирования фильтра (рис. 4.53) нажмите правую клавишу мыши и выберите *Применить фильтр* (или из меню Записи). Фильтр можно сохранить как запрос (также в меню правой клавиши – рис 4.54). В этом случае его можно просто выполнять.

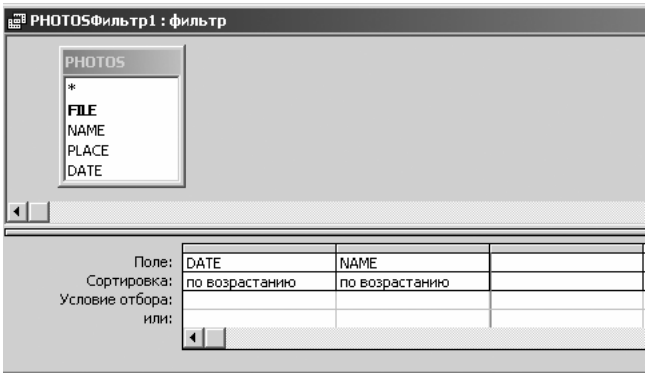


Рис. 4.53. *Построение сложного фильтра*

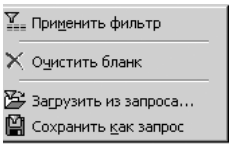


Рис. 4.54. *Локальное меню (правой клавиши) в построении фильтра.*

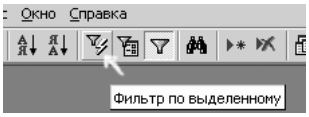


Рис. 4.55. *Применение фильтра по выделенному*

Фильтрация данных может применяться и для других целей, например, вы желаете выделить из таблицы только те строки, которые содержат определенное значение (например, поле PLACE – Мадрид). Выделите это поле, которое имеет такое значение, и нажмите кнопку *Фильтр по выделенному* (рис. 4.55), в таблице будут отображены теперь только строки имеющие значение PLACE – Мадрид. Нажатие кнопки по соседству отменит действие фильтра (рис. 4.56).

PHOTOS : таблица

FILE	NAME	PLACE	DATE
P10055	Стадион	Мадрид	21.12.2002
P10023	Прадо	Мадрид	12.10.2003

Рис. 4.56. *Результат применения фильтра*

PHOTOS : таблица

FILE	NAME	PLACE	DATE
P10055	Стадион	Мадрид	21.12.2002
P10023	Прадо	Мадрид	12.10.2003
P10045	Стадион	Павловск	23.10.2003
P10021	Дворец	Павловск	10.11.2003
P00001	Биг Бен	Лондон	25.09.2004
P00002	Тауэр	Лондон	25.09.2004

Рис. 4.57. *После отмены действия фильтра*

Поиски и замены данных осуществляется с помощью соответствующей команды из меню *Правка*, удобнее воспользоваться сочетанием клавиш Ctrl+F, как и в других приложениях Office (рис 4.58).

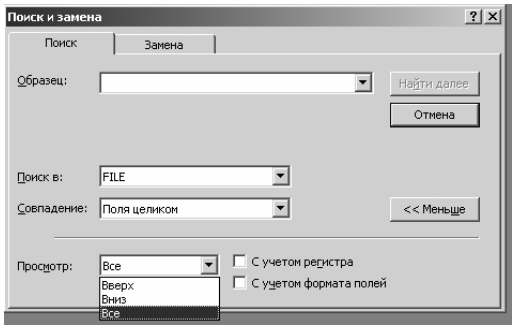


Рис. 4.58. *Поиск и замена данных*

Рис. 4.59. *Предварительный просмотр печати*

Таблицу можно *вывести на печать* также достаточно традиционным способом. Советуем предварительно просмотреть результат, чтобы убедиться в правильном расположении столбцов и количестве страниц.

Вопросы к контрольному опросу №1

1. Сформулируйте три правила нормализации таблиц. Поясните на примерах.
2. Какие типы связей бывают между таблицами? Проиллюстрируйте примером.
3. Назовите девять типов данных, используемых Access.
4. Как задать условие на значение числового поля, текстового поля?
5. Зачем нужна маска ввода? Расскажите о том, как ее установить.
6. Что делает Мастер подстановок?
7. Что такое первичный ключ? Зачем он нужен?
8. Что такое индексированное поле?
9. Какими способами можно установить и настроить связь между таблицами?
10. Как организовать связь «многие-ко-многим». Поясните на примере.
11. Для чего используется язык SQL? Какой оператор в нем применяется чаще всего?
12. Для чего нужно сжатие баз данных?
13. Как переименовать таблицу?
14. Как изменить макет таблицы?
15. Как вставить текущую дату и время в поле?
16. Каким образом произвести сортировку данных таблицы?
17. Как отфильтровать необходимую информацию из таблицы?
18. Как найти нужное значение в полях таблицы?
19. Как распечатать таблицу?
20. Нарисуйте схему данных для предложенной задачи (будет выдана индивидуально). Обоснуйте свое решение.

Глава V. Работа с запросами

Одно из преимуществ запросов состоит в том, что они позволяют достаточно быстро отобрать необходимые данные из нескольких связанных таблиц. Но запросы полезны и при работе с одной таблицей. Все приемы, используемые при работе с единственной таблицей, годятся и для сложных многотабличных запросов, поэтому мы начнем с запросов на выборку данных из одной таблицы.

Выборка данных из одной таблицы

Чтобы создать запрос, мы будем использовать режим конструктора запросов (рис. 2.8, стр. 5). Первое, что будет предложено сделать – выбрать таблицу(ы), по данным которой будет строиться запрос (рис. 5.1). Окно конструктора состоит из двух частей. В верхней размещается список таблиц или запросов, на основании которых будет строится новый запрос. В нижней части располагается бланк запроса, в котором вы будете выполнять всю работу. Каждый столбец представляет одно поле, используемое в запросе. Поле может:

- принадлежать одной из таблиц;
- быть вычисляемым (его значение рассчитывается на основе полей таблиц);
- итоговым (использующим одну из встроенных функций Access).

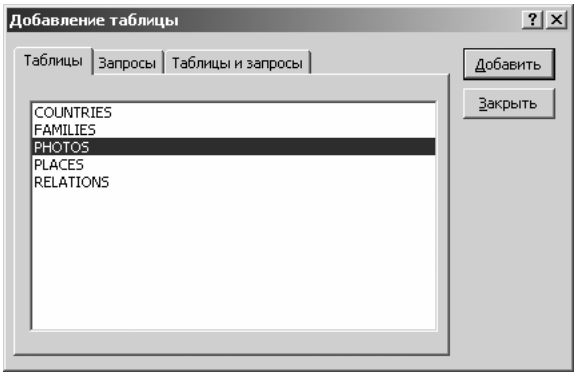


Рис. 5.1. *Добавление таблицы в конструкторе запросов*

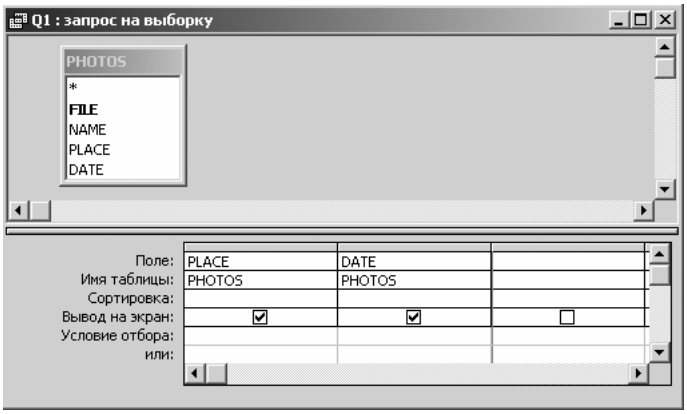


Рис. 5.2. *Построение запроса*

Задание полей

Первым шагом при создании запроса является выбор полей, включаемых в ответ. Это можно сделать несколькими способами. Можно раскрыть список поле и выбрать нужное значение, либо перетащить мышкой имя поля из таблицы (рис. 5.2).

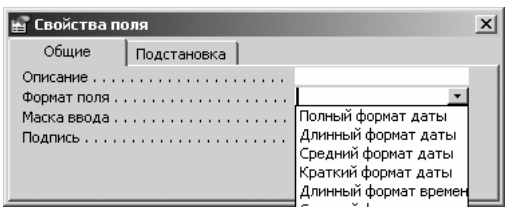


Рис. 5.3. *Установка свойств поля*

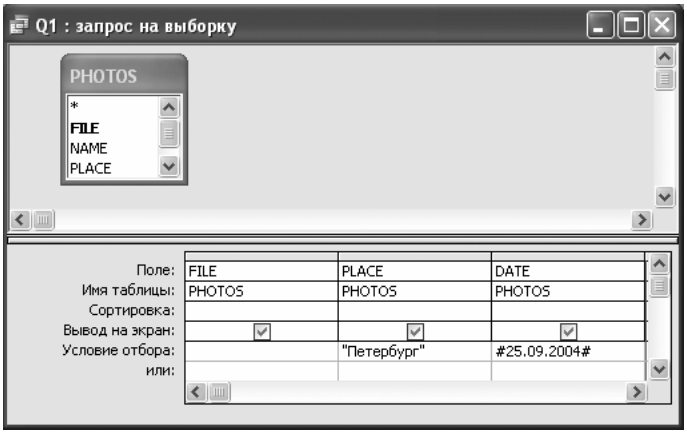


Рис. 5.4. *Установка условия отбора*

Установка свойств полей

Меню правой клавиши мыши будет включать установку дополнительных *свойств поля* (рис. 5.3) В частности можно изменить формат, подпись к столбцу поля в ответе и добавить описание.

***Ввод условий отбора** (важное!)*

Ввод условий отбора аналогичен заданию условия на значения поля таблицы (стр. 16). В ответ запроса будут включены только те записи, которые удовлетворяют условию отбора. Самый простой вид запроса – проверка на совпадение значений полей заданному значению. На рис. 5.4 строится запрос на выборку только тех имен файлов тех фотографий, которые отсняты в Петербурге 25.09.2004. Обратите внимание, что когда вы будете вводить дату в поле условия запроса, она автоматически будет заключена в знаки #, а символьная строка в кавычки. После формирования всех условий запроса конструктор можно закрыть, при этом будет задан вопрос об имени запроса. Двойной щелчок по имени запроса приведет к выполнению запроса (рис. 5.6 а). Обратите внимание на подписи колонок «Место» и «Дата». Они задана в свойствах полей «Подпись» (об именах полей в ответе на запрос см. дальше). Так как в этом запросе будут различаться естественно только имена файлов, а места и даты их одинаковы, то можно опять вернуть к конструктору запроса и снять галочки «Вывод на экран» для полей PLACE и DATE. В этом случае в ответе будут только имена файлов (рис. 5.6 б). Для даты и любых числовых значений можно задавать условие на сравнение, например `>#01.01.2004#`. Диезы при вводе даты можно не указывать, они появятся автоматически, напоминая о том, что это не символьная строка и не число, а дата.

	FILE	NAME	PLACE	DATE
+	P00001	Биг Бен	Лондон	25.09.2004
+	P00002	Тауэр	Лондон	25.09.2004
+	P10021	Дворец	Павловск	10.11.2003
+	P10023	Прадо	Мадрид	12.10.2003
+	P10045	Стадион	Павловск	23.10.2003
+	P10055	Стадион	Мадрид	21.12.2002
+	P1172	Эрмитаж	Петербург	25.09.2004
+	P3200	Летний сад	Петербург	25.09.2004

Рис. 5.5. *Данные исходной таблицы*

Q1 : запрос на выборку

FILE	Место	Дата
P1172	Петербург	25.09.2004
P3200	Петербург	25.09.2004

Q1 : запрос на вы

FILE
P1172
P3200

Рис. 5.6 а, б. *Результат запроса, изображенного на рис. 5.4*

О языке SQL. В ближайшем будущем мы перейдем к составлению запросов на языке SQL. Пока мы будем приводить записи наших запросов на языке SQL. Они настолько просты, что вы поймете их без предварительной подготовки. Постарайтесь обратить внимание на запись.

SQL-запрос, эквивалентный запросу на рис. 5.4.

SELECT FILE, PLACE, DATE **FROM** PHOTOS **WHERE** (PLACE="Петербург") **AND** (DATE=#9/25/2004#);

AND и OR

Для создания более сложных условий можно использовать логические операторы AND и OR. Напоминаем их смысл:

- `a AND b` – истинно тогда, и только тогда, когда истины оба операнда
- `a OR b` – истинно тогда, когда истинен хотя бы один операнд (или оба вместе).

Например, если вы хотите определить дату между первым сентября 2004 г. и 31 декабря 2004 г., необходимо построить условие: `>=#01.09.2004# And <=#31.12.2004#`.

SQL-запрос с использованием логических условий

SELECT FILE, PLACE, DATE **FROM** PHOTOS **WHERE** PLACE="Петербург" **AND** DATE>=#9/1/2004# **AND** DATE<=#12/31/2004#;

BETWEEN, IN и LIKE

Так как очень часто встречается условие именно на принадлежность какому-либо диапазону, то в запросах Access предусмотрен специальный оператор BETWEEN. Форма его записи: `BETWEEN a AND b`, например: `Between #01.09.2004# And #31.12.2004#`. Обращаем внимание, на то, что регистр букв неважен.

SQL-запрос с использованием оператора BETWEEN

SELECT FILE, PLACE, DATE **FROM** PHOTOS **WHERE** PLACE="Петербург" **AND** DATE **BETWEEN** #9/1/2004# **AND** #12/31/2004#;

В тех случаях, когда следует сделать запрос, в котором какое-либо поле должно принимать ряд фиксированных значений, оказывается удобным применение оператора IN. Форма его записи: `IN (знач1; знач2; …)`. Например `In ("Петербург";"Павловск";"Пушкин")`. Обратите, внимание на то, что разделитель полей – точка с запятой.

SQL-запрос с использованием оператора IN

SELECT FILE, PLACE, DATE **FROM** PHOTOS **WHERE** PLACE **IN** ("Петербург","Павловск","Пушкин");

Обратите внимание, что разделитель полей оператора IN в SQL – запятая

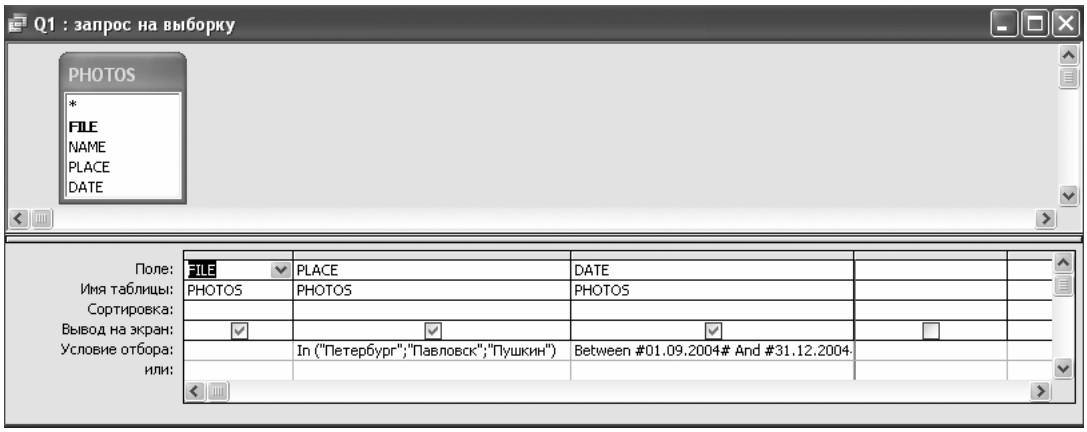


Рис. 5.7. *Запрос со сложными условиями отбора записей*

Если необходимо построить условие на сравнение, частичное совпадение текстовых строк, то очень удобен оказывается оператор LIKE. Оператор производит поиск образцов в текстовых полях. В образцах поиска можно включать следующие символы шаблона:

Таблица 5.1. *Символы шаблона в операторе LIKE*

Символ	Описание
?	Один произвольный символ
*	Любое число (включая нулевое) произвольных символов
#	Одна произвольная цифра
[ ]	Символ из диапазона. Например: [a-z], [2-5].
!	Обратное условие. Например [!0-9] – в этом поле должна быть не цифра.
символ	В этой позиции должен быть указанный символ.

Пример: LIKE "?[a-h]w[0-9]* " означает, что первый символ в строке – любой, второй символ буква от a до h, третий символ – буква w, четвертый символ – цифра, после этого могут идти произвольные символы.

Условия отбора для дат и времени

Чтобы сообщить Access о том, что Вы вводите дату и время, заключайте значение в символы #. Для указания конкретной даты используйте нотацию, которая кажется вам удобной. Например, #15 апреля 2004#, #15.04.2004#, #15/04/04# определяют одну и туже дату. Точно также как и #5:30 PM# и #17:30# определяют одно и тоже время.

Access предоставляет несколько функций, которые могут оказаться полезными при задании отбора. В качестве аргумента многих функций надо указывать в квадратных скобках имя поля. Например, чтобы выбрать только воскресенья, необходимо указать условие отбора `Day([DATE])=1`, если DATE – имя соответствующего поля (столбца).

Таблица 5.2. *Функции работы с датой*

Day(*дата*)	Возвращает значение дня месяца в диапазоне от 1 до 31
Month(*дата*)	Возвращает значение номера месяца в диапазоне от 1 до 12
Year(*дата*)	Возвращает значение года
Weekday(*дата*)	Возвращает значение дня недели от 1 (воскресенье) до 7 (суббота)
Hour(*дата*)	Возвращает час от 9 до 23
Datepart(*код*, *дата*)	Возвращает номер квартала (код – "*q*") или номер недели (код "*ww*")
Date()	Возвращает текущую системную дату

В конструкторе запросов Access после введения условия с функциями даты, как правило, автоматически создаст неотображаемое поле, так как это показано на рис. 5.8.

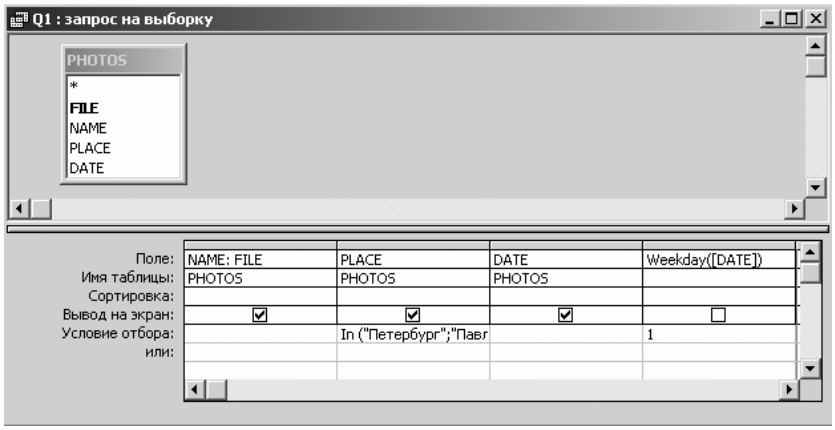


Рис. 5.8. *Использование отбора с функцией работы с датой*

SQL-запрос с использованием функций работы с датой

SELECT FILE, PLACE, DATE **FROM** PHOTOS **WHERE Weekday**([DATE])=1;

Вычисляемые поля

В конструкторе запроса можно задать вычисляемые поля, подобно последнему полю на рис. 5.8. В качестве значения таких полей могут выступать арифметические и символьные выражения с использованием многочисленных встроенных функций Access. В таблице 5.3 представлены некоторые операторы, используемые в выражениях.

Таблица 5.3. *Операторы в числовых выражениях*

Оператор	Описание
+	Складывает два числовых выражения
–	Вычитает из первого числового выражения второе
*	Перемножает два числовых выражения
/	Делит первое числовое выражения на второе
\	Округляет два числовых выражения до целых и делит первое на второе нацело
^	Возводит первое числовое выражение в степень, задаваемое вторым выражением
MOD	Округляет два числовых выражения до целых и возвращает остаток от деления первого на второе.
&	Соединяет две текстовые строки в одну («складывает» строки)

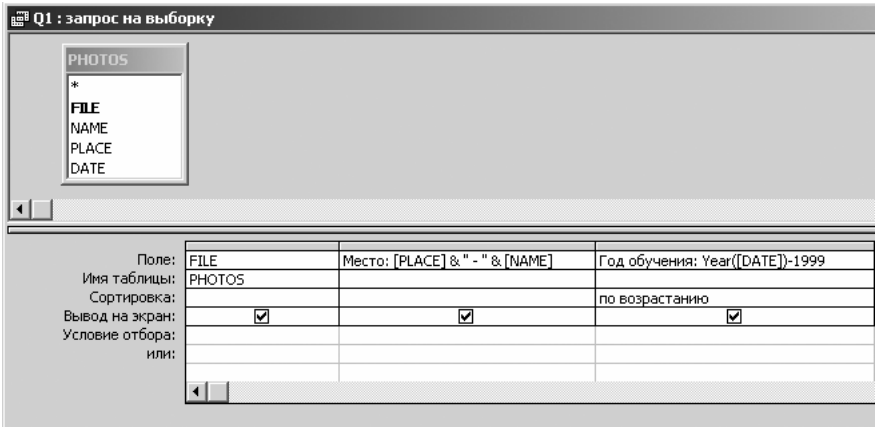


Рис. 5.9. *Использование вычисляемых полей*

На рис. 5.9 представлено создание двух вычисляемых полей с именами «*Место*» и «*Год обучения*». Первое составляется путем конкатенации[1] полей PLACE и NAME в вставкой между ними знака « – », окруженного пробелами, второе вычисляется вычитанием из года даты числа 1999. Результат запроса приведен на рис. 5.10.

Рис. 5.10. *Результат запроса*

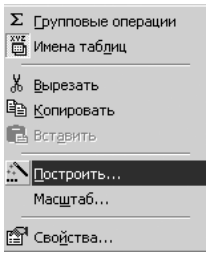


Рис. 5.11. *Выбор команды построителя выражений*

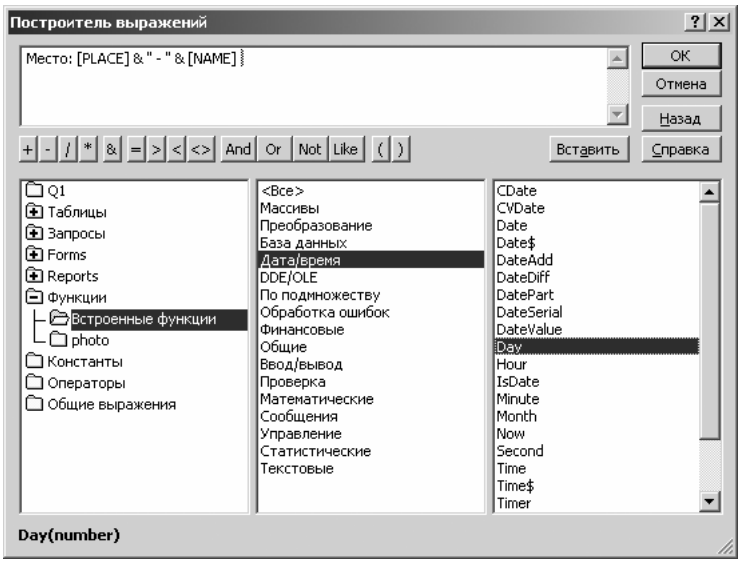


Рис. 5.12. *Построитель выражений*

[1] Объединение (склеивание, сложение) строк.

SQL-запрос с использованием вычисляемых полей

SELECT FILE, [PLACE] & " - " & [NAME] **AS** Место, Year([DATE])-1999 **AS** [Год обучения] **FROM** PHOTOS **ORDER BY** Year([DATE])-1999;

Для создания сложных выражений в Access имеется удобный инструмент «Построитель выражений» (рис. 5.11). В окне построителя выражений (рис. 5.12) имеется список всех полей таблиц, встроенных функций и операторов с примерами их использования.

Задание имен полей

Мы уже говорили о том, как задать имена колонкам в ответе на запрос (с помощью меню правой клавиши мыши в соответствующем поле конструктора запроса). Однако есть еще одна, более фундаментальная возможность. Результат запроса есть таблица. Можно сделать так, чтобы не подписи, а сами поля (колонки таблицы) ответа на запрос имели новые имена, отличные от имен в исходной таблице (в таблицах). Это позволит использовать эти имена в других запросах, базирующихся на текущем запросе. Для этого необходимо вручную отредактировать свойство «Поле» в конструкторе запросов. Перед старым именем поля надо записать новое имя поля и двоеточие (рис. 5.8). Результат запроса показан на рис. 5.9. Заголовок *NAME* в результате замены имени поля, а заголовки *Место* и *Дата* в результате замены свойства «Подпись» соответствующих колонок. Разница заключается в том, что *Место* и *Дата* – это только подписи заголовков, в то время как NAME – имя поля результирующей таблицы. В запросе SQL эта разница будет сразу видна.

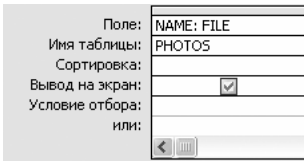


Рис. 5.8. *Замена имени поля FILE на NAME*

Рис. 5.9. *Результат замены имени*

SQL-запрос с использованием замены имени поля ключевым словом AS

SELECT FILE **AS** NAME, PLACE, DATE **FROM** PHOTOS **WHERE** PLACE **IN** ("Петербург","Павловск","Пушкин");

Вы, наверное, уже поняли, как работает основной оператор SQL – оператор SELECT. Его общий синтаксис:

SELECT *список полей* **FROM** *таблица* **WHERE** *условия отбора* **ORDER BY** *порядок сортировки*;

Сортировка данных

Можно заставить Access выводить записи не в том порядке, в котором они находятся в таблицах базы данных. Для этого надо указать порядок сортировки (рис. 5.13). Можно задать сортировку по нескольким полям. В этом случае Access применяет условия сортировки в порядке их расположения в бланке запроса слева направо.

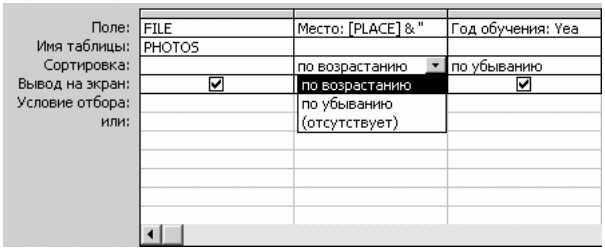


Рис. 5.13. *Построитель выражений*

Задание 15 (также важное).

Создайте не менее 3 запросов к таблицам с использованием правил сортировки, неплохо, если будут вычисляемые поля.

(*по 1 баллу за каждый сложный запрос*)

Итоговые запросы (желательно, но не обязательно)

Существует особый вид запросов – итоговый запрос. В таких запросах нас интересуют не отдельные записи таблицы, а итоговые значения по группам данных. Чтобы создать такой запрос, нажмите правую клавишу мыши в конструкторе запросов и выберите команду «Групповые операции», после ее активации в конструкторе появится дополнительная строка «Групповая операция». По умолчанию значение этого поля «группировка», которое лишь группирует записи, выбирая только уникальные значения. Если раскрыть выпадающий список, то можно обнаружить имена 9 функций, описание которых дано в таблице 5.4. Пример итогового запроса представлен на рис. 5.14. Для поля DATE определено вычисление итоговой функции Min, что для даты равнозначно выборке самой ранней. Второй столбец предполагает выборку уникальных мест съемки. Таким образом, итоговый запрос будет представлять даты первых съемок во всех местах (рис. 5.15). Естественно, можно использовать в итоговых запросах и условия отбора (см. далее).

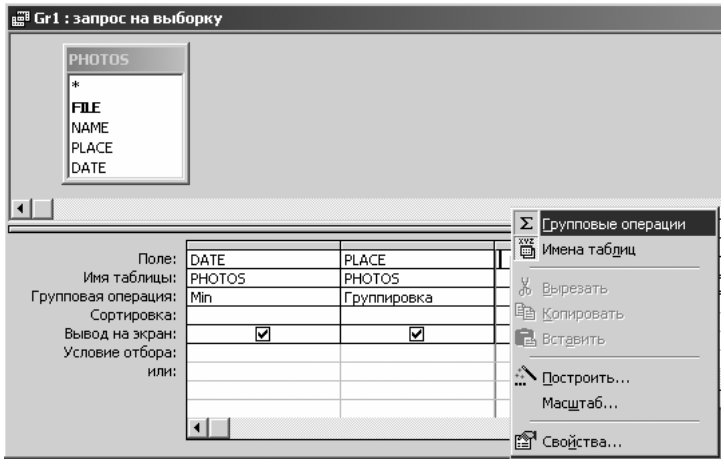


Рис. 5.14. *Создание итогового запроса*

Рис. 5.15. *Результат итогового запроса*

Таблица 5.4. *Итоговые функции*

Функция	**Описание**
Sum	Возвращает сумму всех значений поля в группе (*только числовые и денежные поля*).
Avg	Возвращает среднее арифметическое в группе (*только числовые и денежные поля*).
Min	Возвращает наименьшее значение поля в группе (*все типы*).
Max	Возвращает наибольшее значение поля в группе (*все типы*).
Count	Возвращает число записей, в которых это поле непустое (не Null).
StDev	Возвращает стандартное отклонение всех значений (то, что пишется после ±).
Var	Возвращает дисперсию (степень разброса).
First	Возвращает значение поля из первой записи, обнаруженной в группе.
Last	Возвращает значение поля из последней записи, обнаруженной в группе.

SQL-запрос для групповой операции

SELECT Min(DATE) **AS** [Min-DATE], PLACE **FROM** PHOTOS **GROUP BY** PLACE **ORDER BY** Min(DATE);

Задание 16. Постройте итоговый запрос к вашей БД.

(2 *балла за каждый*)

Перекрестные запросы (необязательно, но очень информативно)

Access поддерживает специальный вид запроса, называемы перекрестным. В нем вы определяете заголовки, как строк, так и столбцов, а в качестве данных используете значения других столбцов. Например, нас интересует какие объекты, в каких городах изображены на фотографиях по годам. Вид такой таблицы изображен на рис. 5.16. Первая колонка – заголовок строк, вторая колонка – итоговая функция по строкам: «сколько всего объектов отснято в данном месте», в качестве значений таблицы используется имя первого объекта в базе данных, соответствующего данному месту и году съемки.


PHOTOS_перекрестный1 : перекрестный запрос

PLACE	Всего объектов	2002	2003	2004
Лондон	2			Биг Бен
Мадрид	2	Стадион	Прадо	
Павловск	2		Стадион	Дворец
Петербург	3		Инженерный замок	Эрмитаж

Запись: 1 из 4

Рис. 5.16. *Результат перекрестного запроса.*

Чтобы построить такой запрос, необходимо, находясь в окне просмотра таблиц, нажать кнопку «новый объект» (рис. 5.17) и выбрать создание перекрестного запроса (рис. 5.18).

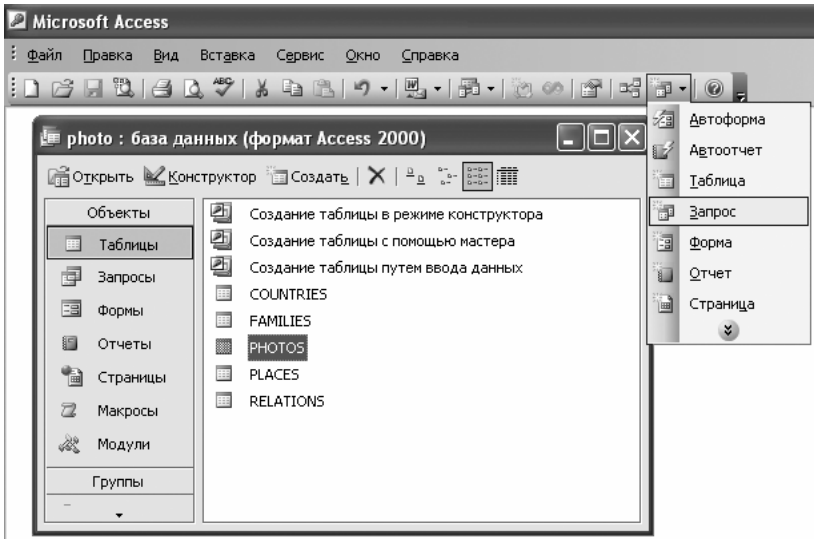


Рис. 5.17. *Выбор команды создания нового запроса.*

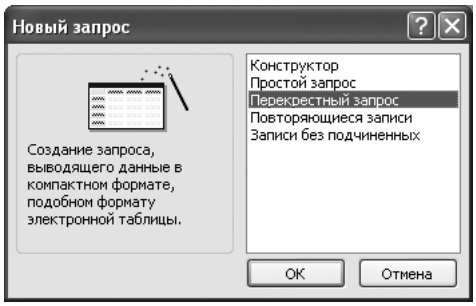


Рис. 5.18. *Выбор перекрестного запроса.*

Далее вас проведен мастер создания запроса. Первое, вы должны определить таблицу (таблицы), к которой будет строиться запрос, затем – поле, которое будет использоваться в качестве заголовков строк (рис. 5.19). После вы определяете заголовки колонок (рис. 5.20), если в качестве этого поля будет использоваться дата, то будет предложено сгруппировать их по месяцам, кварталам или годам (рис. 5.21). Наконец, вы определяете значения ячеек (рис. 5.22). Можно попросить вычислять заодно итоговое значение для каждой строки. Войдя в режим изменения структуры запроса, можно поменять итоговую функцию (рис. 5.23). Запрос готов.

SQL-запрос для перекрестного запроса

```
TRANSFORM First(NAME) AS [First-NAME]
SELECT PLACE, Count(NAME) AS [Итоговое значение NAME] FROM PHOTOS
GROUP BY PLACE PIVOT Format([DATE],"yyyy");
```

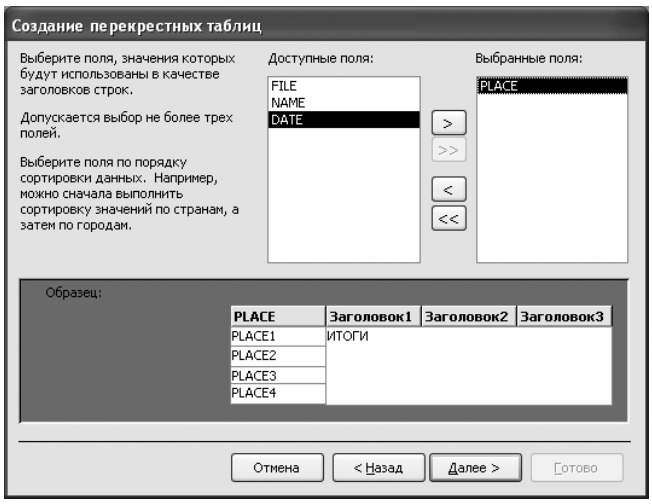


Рис. 5.19. *Выбор заголовков строк.*

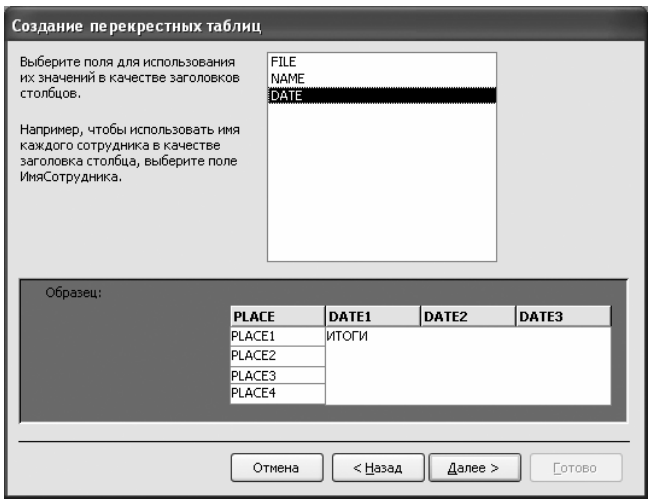


Рис. 5.20. *Выбор заголовков столбцов.*

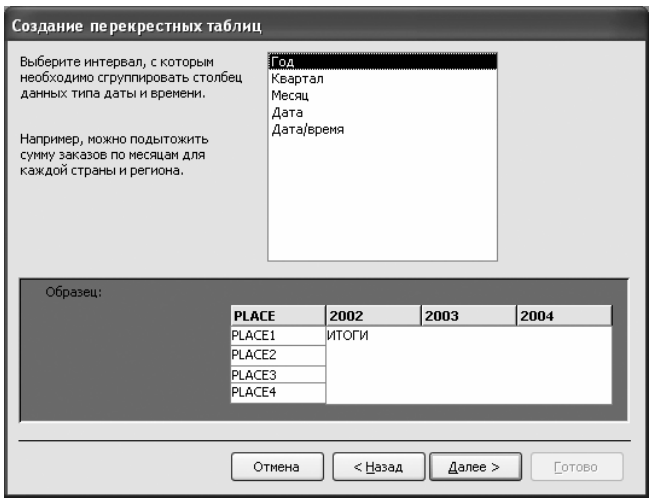


Рис. 5.21. *Группировка по дате.*

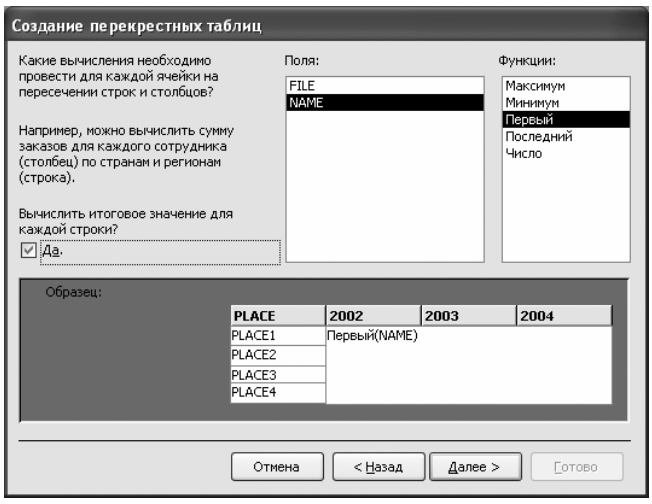


Рис. 5.22. *Выбор значения.*

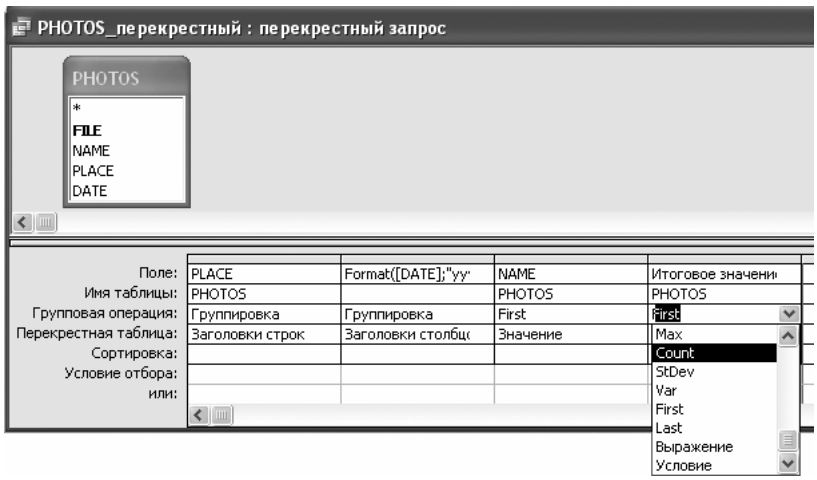


Рис. 5.23. *Изменение макета перекрестного запроса.*

Задание 17. Постройте перекрестный запрос к вашей БД.

(5 *баллов за каждый*)

Использование параметров запроса

До сих пор мы вводили условия отбора непосредственно в бланке запроса. Однако это далеко не всегда удобно. Чтобы определить параметр, используйте в условии запроса фразу, заключенную в квадратные скобки. На рис. 5.24 приведен пример использование параметра в итоговом запросе. Обратите внимание на то, что создается еще одно поле отбора со значением *групповой операции* «*условие*», а в *условии отбора* определено использование параметра. При выполнении запроса появится диалоговое окно с предложением ввести параметр (рис. 5.25).

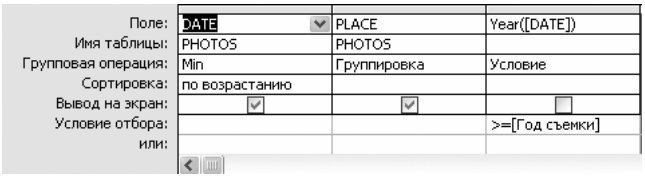


Рис. 5.24. *Использование параметра запроса.*

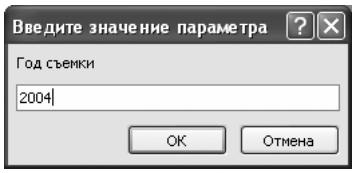


Рис. 5.25. *Запрос ввода параметра.*

В запросе может быть несколько параметров, в этом случае Access попросит ввести их поочередно.

Задание 18. Задайте параметр какому-либо запросу к вашей БД. (*1 балл за каждый*)

Многотабличные запросы (крайне важно)

Наиболее важная черта реляционной базы данных – возможность использовать многотабличные запросы (особенно, если есть связь между таблицами). **Именно для этой цели создавались все наши связи между таблицами и использовались правила нормализации таблиц!**

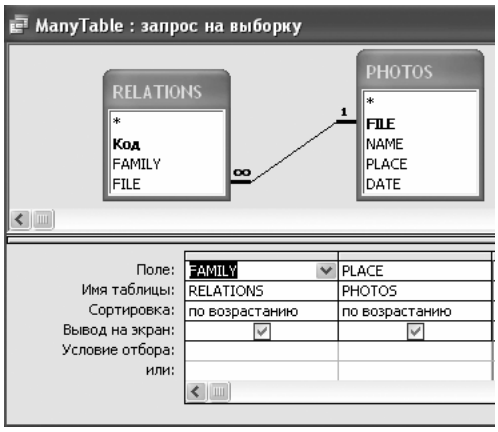


Рис. 5.26. *Создание многотабличного запроса*

Рис. 5.27. *Ответ на многотабличный запрос*

Рис. 5.28. *Редактирование свойств запроса*

Рис. 5.29. *Ответ – только уникальные значения*

SQL-запрос к связанным таблицам (к рис. 5.27)

```
SELECT RELATIONS.FAMILY, PHOTOS.PLACE
FROM PHOTOS INNER JOIN RELATIONS ON PHOTOS.FILE=RELATIONS.FILE
ORDER BY RELATIONS.FAMILY, PHOTOS.PLACE;
```

SQL-запрос на выборку уникальных значений (к рис. 5.29)

```
SELECT DISTINCT RELATIONS.FAMILY, PHOTOS.PLACE
FROM PHOTOS INNER JOIN RELATIONS ON PHOTOS.FILE = RELATIONS.FILE
ORDER BY RELATIONS.FAMILY, PHOTOS.PLACE;
```

После всестороннего изучения возможностей запросов, построенных на основе одной таблицы, составление многотабличных запросов не составит особых трудностей. Поясним это на примере. У нас есть две таблицы PHOTOS, которая содержит поля FILE, PLACE, и RELEATIONS, которая содержит поля FILE и FAMILY. Между соответствующими полями FILE (в обозначениях Access – PHOTOS.FILE и RELATIONS.FILE) установлена связь (неважно как, через подстановку или прямо на схеме данных). Необходимо выбрать из таблицы RELATIONS только те фамилии, для которых есть соответствие (совпадение) между полями FILE связанных таблиц. Это делает запрос, изображенный на рис. 5.26, ответ на него показан на следующем рис. 5.27. Такой способ запроса к связанным таблицами называется симметричным *внутренним объединением* (*Inner join*). Он основан на совпадении связанных полей базовых таблиц.

Обратите внимание, что на рис. 5.27 у нас в выборке оказались одинаковые записи, но они соответствуют разным строками в исходных таблицах (там же разные имена файлов). Чтобы убрать в ответе повторяющиеся строки, измените свойства запроса (меню правой клавиши мыши в конструкторе запроса в любом месте, кроме колонок полей) (рис. 5.28). После этого ваш ответ будет выглядеть так, как, наверное, вы и ожидали. Появится список, позволяющий понять, кто в каких городах бывал (рис. 5.29).

Проанализируйте тексты SQL-запросов.

Вероятно, большинство задач на выборку данных из ваше БД будет решаться как раз многотабличными запросами. Их эффективность будет зависеть от правильной структуры таблиц базы данных и связей между ними.

Задание 19. Постройте все необходимые многотабличные запросы к вашей базе данных.
(*3 балла за каждый*)

Внешнее объединение

Иногда желательно увидеть информацию об объектах, для которых нет связей в другой таблице. Например, нас интересуют места съемок, на которых никто не изображен. Для этого в конструкторе запросов дважды щелкните на связи между таблицами. В появившемся диалоге (рис. 5.30) установите желаемый вид связи: например, объединение *всех* записей из левой таблицы и только тех записей из *правой* таблицы, для которых значения связанных полей совпадают. После этого изменит свой вид стрелка связи (рис. 5.31). В результате запроса (рис. 5.32) вы увидите те записи, для которых нет соответствия в таблице RELATIONS.

Задание 20. Постройте внешнее (левое или правое) объединение для ваше БД.
(*4 балла за каждый запрос*)

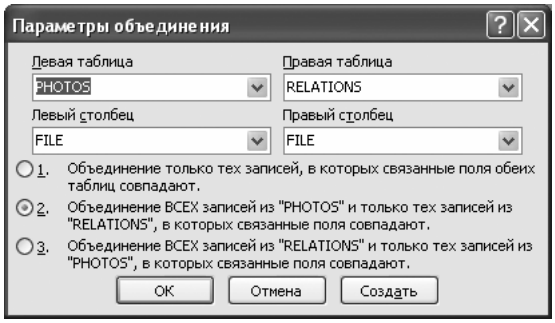


Рис. 5.30. *Установка параметров объединения*

Рис. 5.31. *Новый вид запроса на выборку*

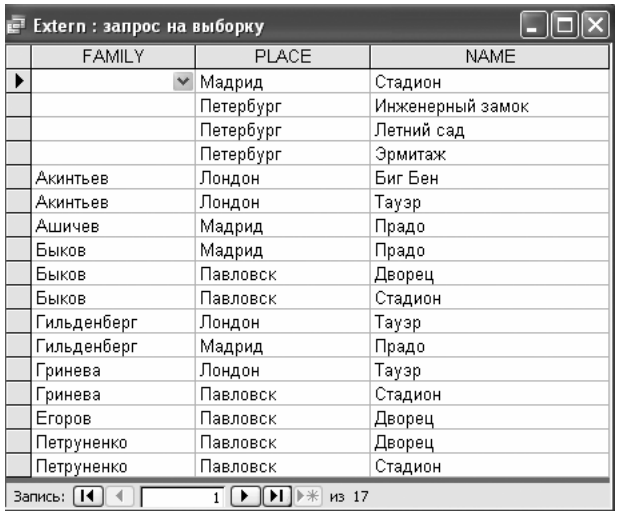


Рис. 5.32. *Результат запроса. Есть строки, для которых нет соответствующих фамилий.*

SQL-запрос на левостороннее объединение

SELECT DISTINCT RELATIONS.FAMILY, PHOTOS.PLACE, PHOTOS.NAME **FROM** PHOTOS **LEFT JOIN** RELATIONS **ON** PHOTOS.FILE = RELATIONS.FILE
ORDER BY RELATIONS.FAMILY, PHOTOS.PLACE;

Запрос к запросу (важно)

Великолепным свойством реляционной базы данных является то обстоятельство, что результат запроса можно использовать в других запросах. Результат запроса, как мы знаем из реляционной модели, является такой же таблицей, как и базовые таблицы. Поэтому в конструкторе запроса вы можете выбрать в качестве исходных данных не таблицы, а запрос (рис. 5.33). Например, выберем из результата предыдущего запроса (рис. 5.32) только те строки, у которых поле FAMILY пустое (и не будем его отображать на экране – нет галочки). Результат представлен на рис 5.34. SQL-текст запроса ничем не отличается от запроса к таблице. Обратите внимание на проверку «*поле пустое*».

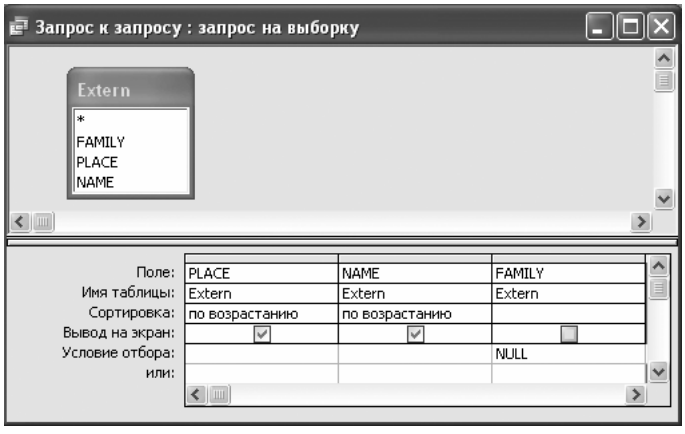


Рис. 5.33. *Построение запроса к запросу*

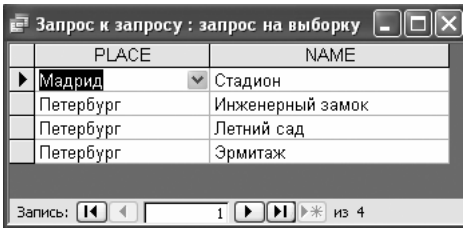


Рис. 5.34. *Результат запроса к запросу*

SQL-запрос к запросу не отличается от запроса к таблице.

SELECT PLACE, NAME **FROM** Extern
WHERE FAMILY Is Null **ORDER BY** PLACE, NAME;

Задание 21. Постройте запрос к запросу в вашей БД.

(2 *балла за каждый запрос*)

Модификация данных с помощью запросов на изменение

Реляционная база данных поддерживает особый вид запросов, цель которых не извлечение, а изменение, добавление и удаление данных в таблицах БД. Существует две причины, по которым становится предпочтительным использование запросов для решения этих задач:

- Обыкновенному пользователю БД не желательно предоставлять доступ к таблицам БД;
- С помощью запросов легко делать различные групповые операции.

Рассмотрим следующий учебный пример: база данных социальных заказов школы и степень их готовности. Пусть пока база данных состоит из двух связанных таблиц: таблица заказов (Orders) и заказчики (cutomers) (рис 5.35). Заполним таблицу, причем для поля флаг будем использовать набор следующих значений: A – заказ активен, D – заказ выполнен, O – заказ просрочен (рис. 5.36). Поставим такую задачу: «изменить поле статуса с A на O для всех заказов, дата исполнения которых раньше текущей». Для начала создадим запрос на выборку данных, отвечающий этим условиям (рис. 5.37). Выберем записи с датой больше или равной текущей, имеющие статус A.

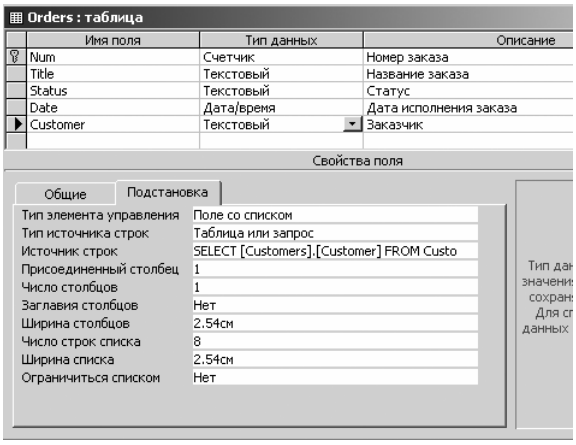


Рис. 5.35. *Таблица Orders в режиме конструктора*

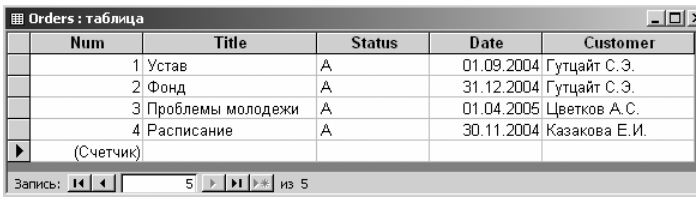


Рис. 5.36. *Таблица Orders в режиме ввода данных*

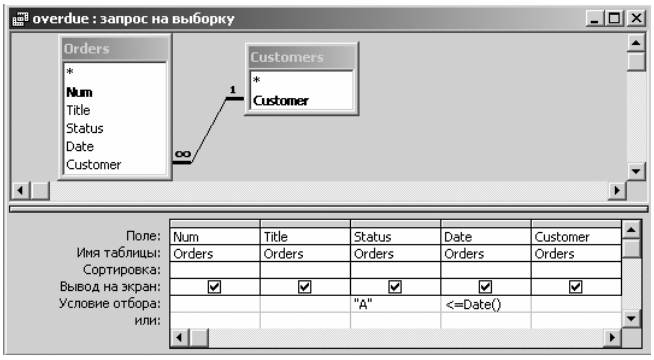


Рис. 5.37. *Таблица Orders в режиме ввода данных*

Теперь можно преобразовать запрос на выборку в запрос на обновление. Для этого надо, находясь в режиме конструктора, раскрыть список кнопки «Тип запроса» и выбрать запрос на обновление (рис 5.38). В появившейся строке «обновление» установим новое значение поля Status в O (рис. 5.39).

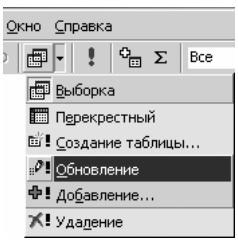


Рис. 5.38. *Преобразование типа запроса*

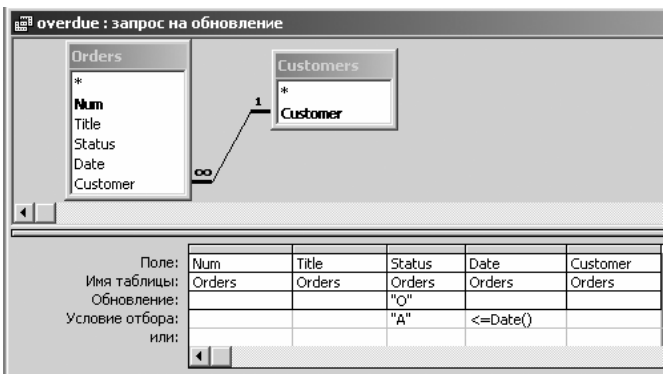


Рис. 5.39. *Построение запроса на обновление данных*

При вызове на исполнение данного запроса вы получите предупреждающее (рис. 5.40) и информационное сообщение (рис 5.41).

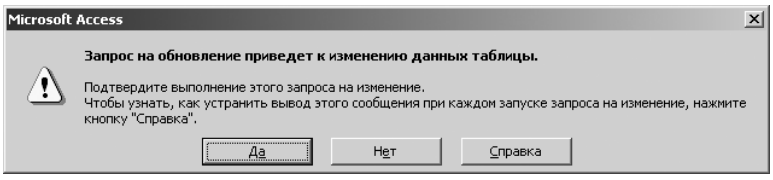


Рис. 5.40. *Предупреждение о запросе на обновление*

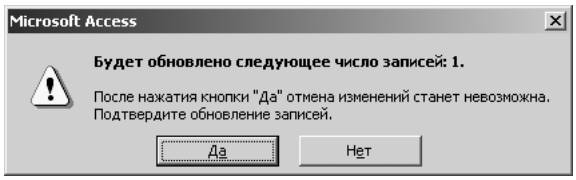
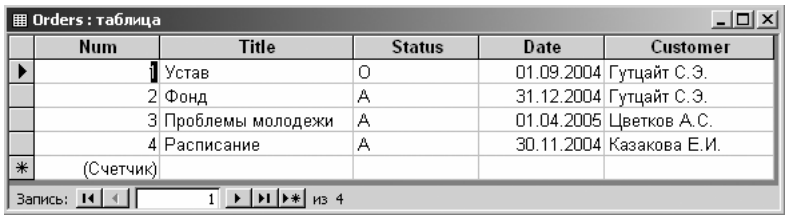


Рис. 5.41. *Информация об обновлении*



SQL-запрос на обновление данных

```
UPDATE Customers
INNER JOIN Orders ON Customers.Customer=Orders.Customer
SET Orders.Status = "O"
WHERE Orders.Status="A" AND Orders.Date<=Date();
```

Рис. 5.42. *Результат выполнения запроса*

В результате выполнения данного запроса автоматически будут обновлены поля записей просроченных заказов (рис. 5.42).

Запрос на создание новой таблицы

Рассмотрим такую задачу. Необходимо из таблицы Orders в таблицу OrdersOld, имеющую такую же структуру, скопировать записи, соответствующие сделанным заказам (т.е. имеющие поле Status “D”). Для этого надо подготовить запрос на выборку (рис 5.43), преобразовать его в запрос на создание новой таблицы, при этом будет запрошено имя целевой таблицы (рис. 5.44). Появится предупреждающее (рис 5.40) и информационное сообщение (рис 5.45), а если таблица уже существовала, сообщение о ее удалении. В результате выполнения запроса будет создана новая таблица (рис. 5.46).

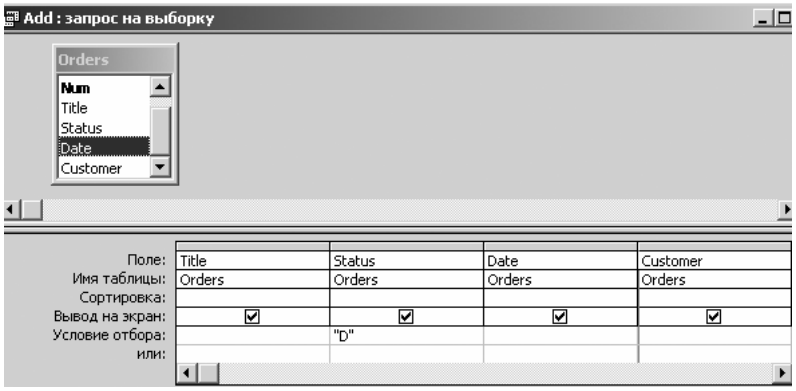


Рис. 5.43. *Подготовка запроса*

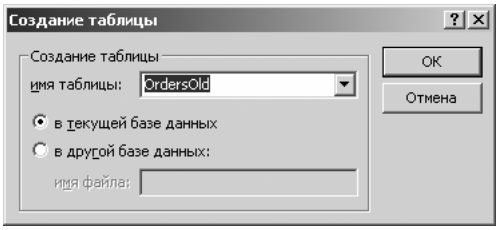


Рис. 5.44. *Преобразование в запрос на создание таблицы*

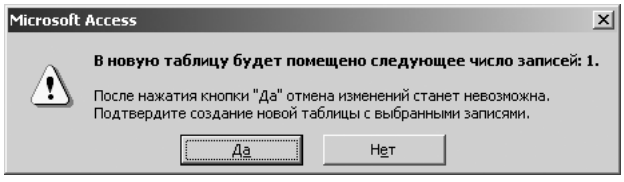


Рис. 5.45. *Информационное сообщение*

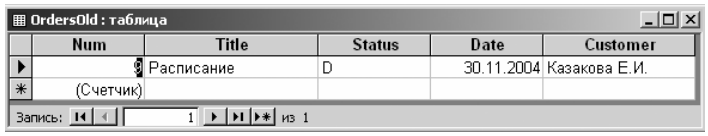


Рис. 5.46. *Результат выполнения*

SQL-запрос на добавление записей в новую таблицу

```
SELECT Title, Status, Date, Customer INTO OrdersOld FROM Orders WHERE Status="D";
```

Запрос на добавление записей в существующую таблицу

Рассмотрим аналогичную задачу, но о добавлении записей к уже существующей таблице. Его создание не отличается сильно от запроса на создание таблицы. Единственно, что вы должны указать тип запроса (рис. 5.47) и выбрать таблицу, в которую будут добавлены записи (рис. 5.48).

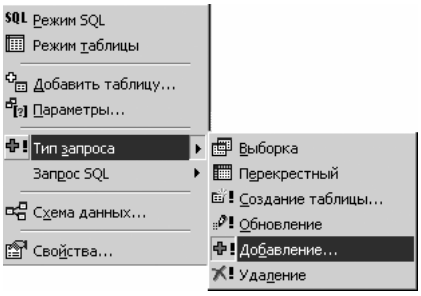


Рис. 5.47. *Выбор запроса на добавление записей*

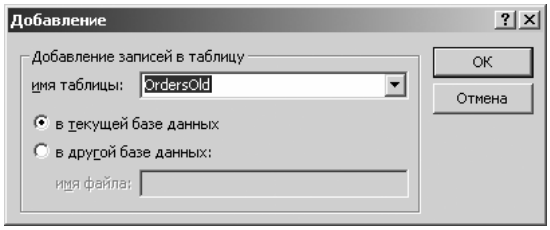


Рис. 5.48. *Запрос имени таблицы для добавления записей*

Запрос на удаление записей из таблицы

Путь необходимо удалить все записи из таблицы со статусом «О». Для начала создайте запрос на выборку записей (рис. 5.49), и проверьте его работу! Убедившись в том, что действительно отбираются нужные записи, преобразуйте запрос к запросу на удаление (рис 5.50). Имя таблицы спрашиваться не будет, подразумевается, что удаление возможно только из записей текущей таблицы запроса.

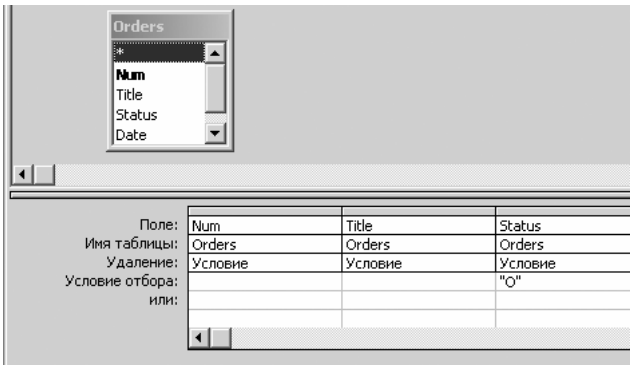


Рис. 5.49. *Подготовка запроса на удаление*

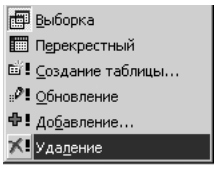


Рис. 5.50. *Преобразование к запросу на удаление записей*

Задание 22. Необходимо построить все запросы на изменение данных в ваших таблицах, которые могут возникнуть при работе с базой данных. Это очень важно, поскольку для конечного пользователя эти запросы будут являться единственно-возможным способом добавлять и изменять записи в таблицах!

(3 *балла за каждый запрос*)

Глава VI. Импорт и экспорт данных

При работе с базами данных очень часто возникают две противоположные задачи: импорт и экспорт таблиц. Например, есть большая электронная таблица с уже набранными данными. Как перенести ее в Access? Обратная задача: я хочу сохранить данные какой-то таблицы или запроса в Access в простой текстовый файл, с тем, чтобы передать его по e-mail. Access имеет весьма развитые возможности по обмену данными в формате различных приложений. Мы рассмотрим только два формата: формат таблицы Excel и текстовый файл.

Импорт электронной таблицы

Пусть у нас имеется электронная таблица, изображенная на рис. 6.1, сохраненная в файле photos.xls. Нам надо импортировать эту таблицу в Access. В разделе базы данных «Таблицы» нажмите правую клавишу мыши и выберите команду «Импорт» (рис. 6.2). В раскрывшемся диалоге обязательно выберите тип фалов Microsoft Excel (*.xls) (рис. 6.3).

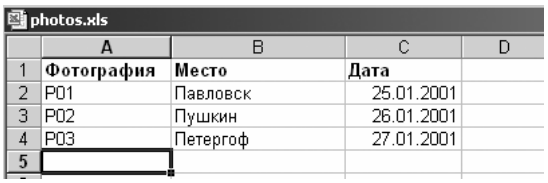


Рис. 6.1. *Исходная Excel-таблица*

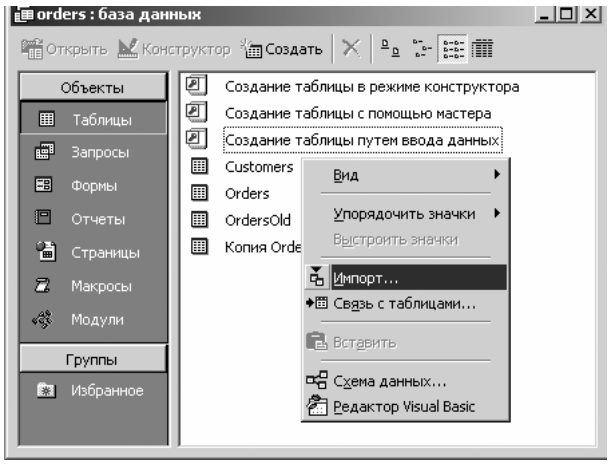


Рис. 6.2. *Начало импорта данных*

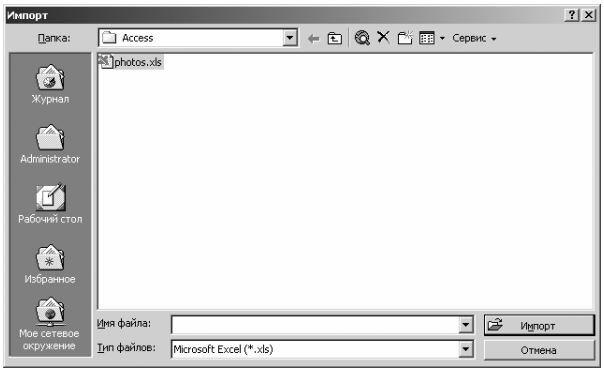


Рис. 6.3. *Диалог выбора xls-файла*

Далее Access проанализирует структуру таблицы и предложит установить соответствие между столбцами таблицы Excel-таблицы и полями таблицы Access (рис. 6.4).

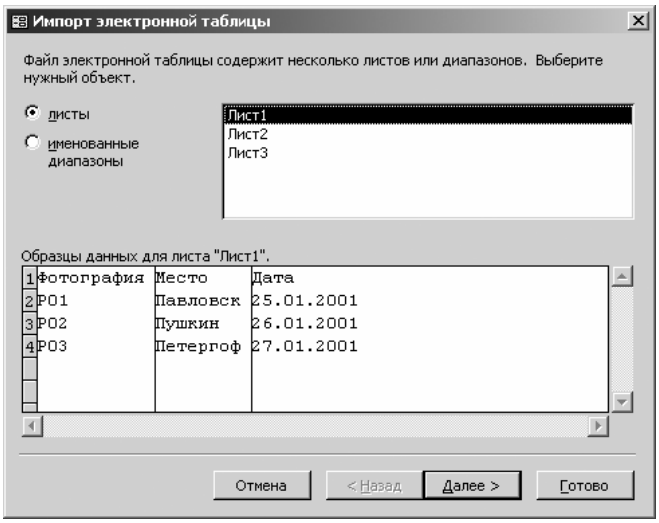


Рис. 6.4. *Определение полей новой таблицы*

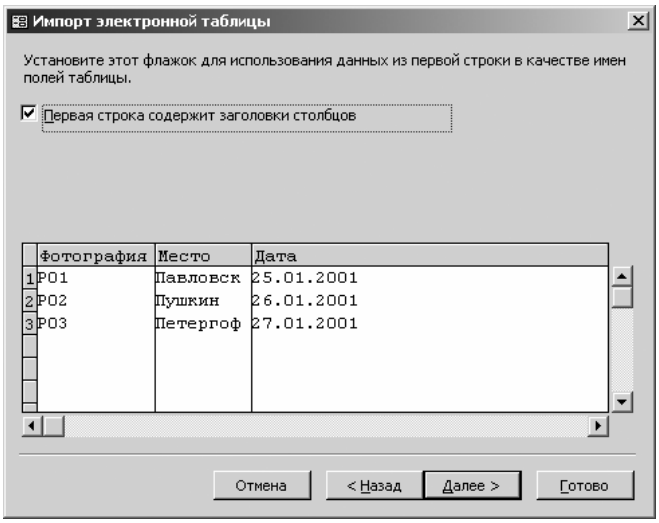


Рис. 6.5. *Определение имен полей*

Мастер импорта таблицы предложит считать первую строку Excel-таблицы строкой заголовков, в Access эти заголовки станут естественно именами полей таблицы (рис. 6.5).

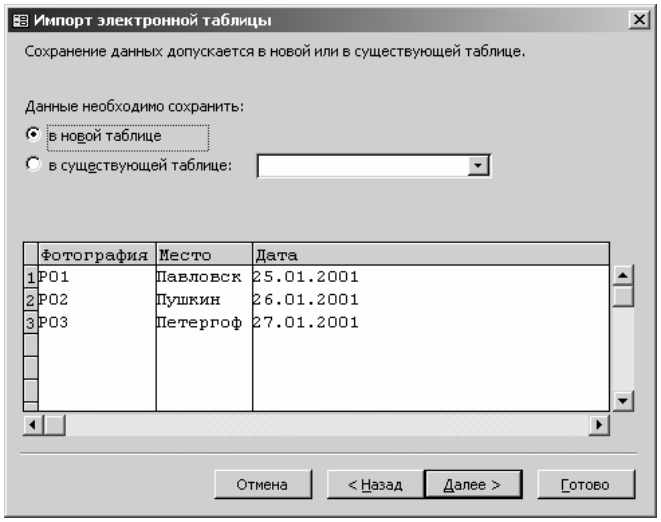


Рис. 6.6. *Определение целевой таблицы*

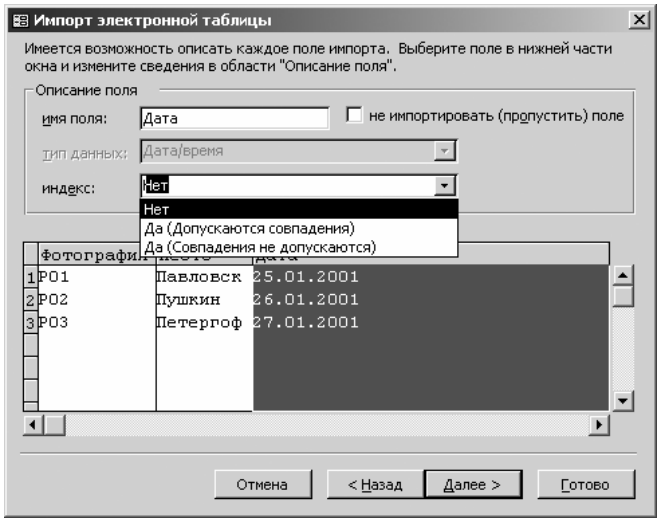


Рис. 6.7. *Определение индексированных полей*

Мастер импорта таблицы предложит считать первую строку Excel-таблицы строкой заголовков, в Access эти заголовки станут естественно именами полей таблицы (рис. 6.5). Далее вы должны выбрать место сохранения данных: в новой или существующей таблице (рис. 6.6), определить наличие индексированных (стр. 21) полей (рис 6.7). Наконец, необходимо назначить ключевое поле (рис. 6.8). Мастер завершает свою работу просьбой указать имя новой таблицы (рис. 6.9).

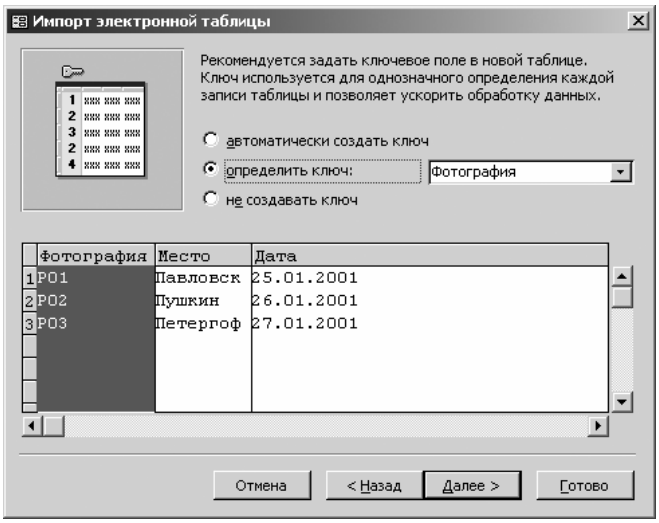


Рис. 6.8. *Определение ключевого поля*

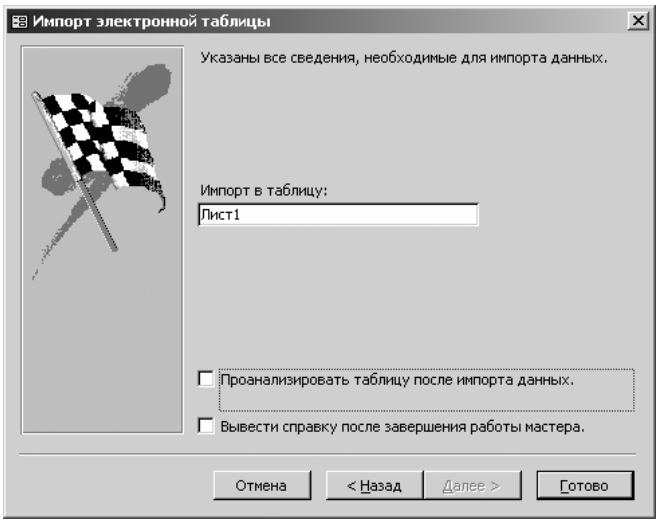


Рис. 6.9. *Определение имени таблицы в Access*

После импорта таблицы необходимо обязательно войти в конструктор вновь созданной таблицы и проверить типы полей, которые назначил Access. В частности, можно уменьшить размер текстовых полей, так как Access назначает им наибольшую длину 255 символов (рис. 6.10).

Важное замечание

Естественно, что для максимально адекватного импорта таблицы необходимо наличие в столбцах значений одного типа (в Excel же это не обязательно), поэтому если таблица сложная, лучше средствами Excel подготовить простую таблицу вида (рис. 6.1), тогда ее данные будут максимально корректно перенесены в Access (рис. 6.11).

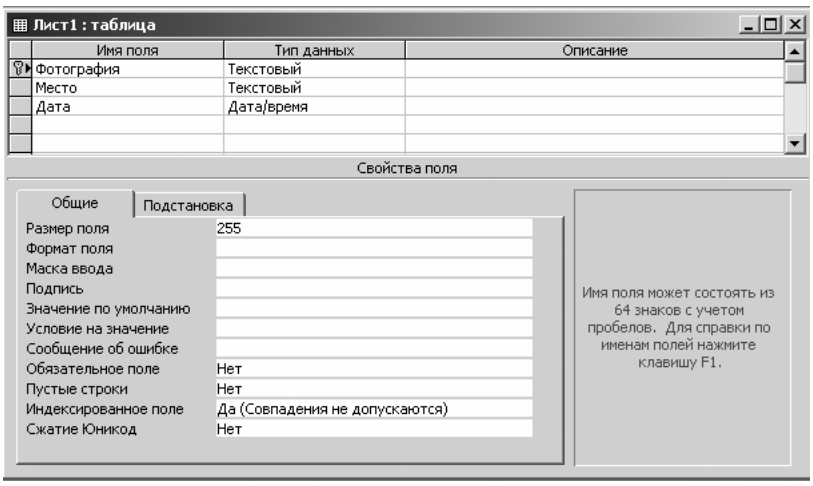


Рис. 6.10. *Созданная таблица в режиме конструктора. Размер текстового поля – 255 символов.*

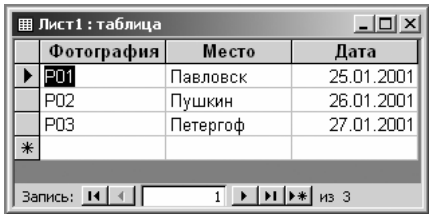


Рис. 6.11. *Импортированные данные в таблице Access*

Экспорт электронной таблицы

Решим обратную задач: данные таблицы или запроса Access перенести в таблицу Excel. В списке таблиц или запросов выделите объект, нажмите правую клавишу мыши и выберите пункт меню «экспорт». Задайте формат файла *Microsoft Excel 97-2000 (*.xls)*, и, возможно, другое имя файла.

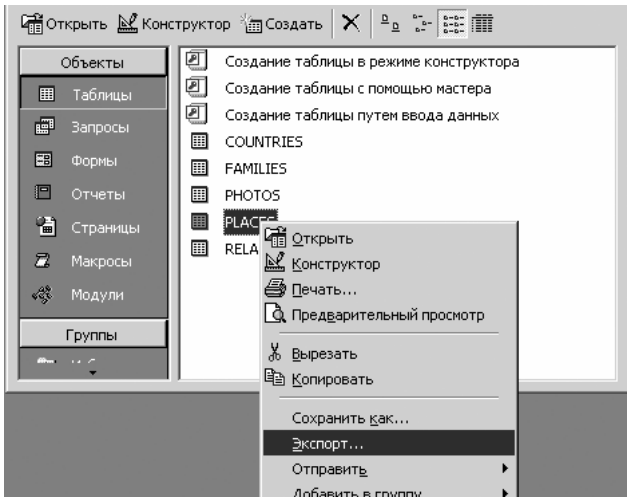


Рис. 6.12. *Экспорт данных*

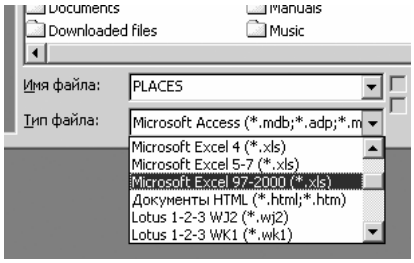


Рис. 6.13. *Выбор формата Excel 97-2000*

Рис. 6.14. *Результат в Excel 97-2000*

Задание 23. Импортируйте в вашу БД данные из электронной таблицы.
(*2 балла*)

Задание 24. Экспортируйте из вашей БД наиболее важные таблицы в Excel.
(*1 балла за каждый экспорт*)

Иногда необходимо произвести импорт/экспорт таблиц в текстовые (не Word!) файлы. Эту задачу рассмотрим при необходимости тем.

Глава VII. Создание запросов с использованием SQL

Любой запрос в Access реализуется с помощью языка SQL. Конструктора запросов является визуальным средством построения предложений на языке SQL и во многих случаях его использование оправдано. Однако для создания подчиненных запросов, сложных запросов, а также запросов из активных web-страниц необходимо знание хотя бы основ SQL.

Синтаксис инструкции SELECT

Инструкция SELECT является ядром языка SQL. Она используется для отбора строк и столбцов из таблиц БД и содержит пять основных предложений:

```
SELECT <список полей>
    FROM <список таблиц>
    [WHERE <условие отбора строк>]
    [GROUP BY <условие группировки>]
    [HAVING <условие отбора групп>]
    [ORDER BY <условие сортировки>];
```

Здесь и далее квадратные скобки обозначают необязательные элементы, угловые скобки – обязательные элементы синтаксиса. Текст внутри угловых скобок характеризует элемент, но не описывает его.

Рассмотрим простой сразу простой пример:

```
SELECT *
  FROM PHOTOS
  WHERE DATE>#1/1/2003#
  ORDER BY DATE, PLACE;
```

Звездочка после SELECT означает, что необходимо выбрать все поля из таблицы PHOTOS, о чем сообщает предложение FROM. На записи (строки) результата накладывается условие: поле DATE должно содержать дату после 1 января 2003 г. Результат необходимо отсортировать сначала по дате, затем по месту (PLACE).

Чтобы попробовать создать запрос в Access прямо на языке SQL. Начните создавать запрос в конструкторе, нажмите правую клавишу мыши и выберите «Режим SQL» (рис. 7.1). Далее введите (желательно без ошибок ☺) текст запроса. При закрытии запроса введите, как обычно, имя. Созданный запрос на SQL по своей работе ничем не отличается от сконструированного. Кстати, если создать аналогичный запрос в режиме конструктора и посмотреть на его SQL-текст (рис. 7.3), то можно заметить, что он выглядит более громоздко: имена всех полей предваряются именем таблицы с точкой (при выборке из одной таблицы это не нужно), есть лишние в данном случае скобки. Т.е. наш запрос по записи является более компактным (и эффективным).

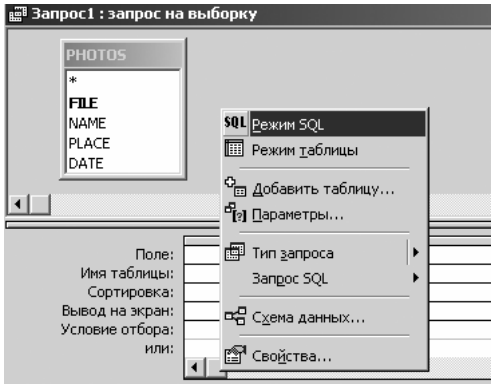


Рис. 7.1. *Установка режима SQL*

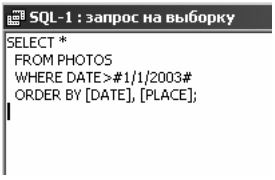


Рис. 7.2. *Ввод текста SQL*

Рис. 7.3. *Ввод текста SQL*

Для выборки только некоторых полей вместо звездочки пишем через запятую их имена, возможно использование конструкции AS для изменения имени поля в ответе запроса. Например, ответ на нижеприведенный запрос будет содержать поля DATE и City (рис. 7.4).

```
SELECT DATE, PLACE AS City
    FROM PHOTOS
    ORDER BY DATE, PLACE;
```

Копия SQL-1 : запрос на выборку

DATE	City
21.12.2002	Мадрид
30.01.2003	Петербург
21.04.2003	Петербург
12.10.2003	Мадрид

Рис. 7.4. *Результат запроса SQL*

Возможно использование функций в операторе SELECT. Подсчитать, сколько фотографий сняты в 2004 г. можно нижеследующей инструкцией, а результат представлен на рис 7.5.

```
SELECT count(*) AS N2004
  FROM PHOTOS WHERE DATE>#1/1/2004#;
```

Копия SQL-1 : запрос на выборку

N2004
5

Рис. 7.5. *Запрос вернул только одно число*

Предложение FROM

Предложение FROM задает имена таблиц или запросов, служащие источниками данных для создаваемого запроса. Простые примеры вы уже видели. Продемонстрируем использование предложения FROM в случае выбора данных из связанных таблиц.

```
SELECT DISTINCT [FAMILY], [PLACE]
  FROM PHOTOS INNER JOIN RELATIONS ON
     [PHOTOS].[FILE]=[RELATIONS].[FILE]
```

ManyTable : запрос на выборку

FAMILY	PLACE
Акинтьев	Лондон
Ашичев	Мадрид
Быков	Мадрид
Быков	Павловск
Гильденберг	Лондон
Гильденберг	Мадрид

Запись: 1 из 10

Рис. 7.6. *Запрос к связанным таблицам*

Следует сказать, что информация о связанности таблиц, вообще говоря, не хранится в базе данных. Это лишь визуальные средства Access рисуют линии и сохраняют эту информацию, на основании которой строятся запросы SQL. В случае непосредственного использования языка SQL связь между таблицами задается предложением INNER JOIN между именами таблиц, а после предлога ON указывается условие связи, данном случае условие на равенство полей из разных таблиц. Обратите внимание, как указывается явно принадлежность поля к таблице: [PHOTOS].[FILE], здесь PHOTOS – имя таблицы, а FILE – имя поля. Наличие квадратных скобок в общем случае необязательно, можно было бы написать и просто PHOTOS.FILE, однако Access всегда сам поставит квадратные скобки около имен. Скобки обязательны, если поле имеет в своем названии пробелы или специальные символы.

Предложение ORDER BY

Задает порядок сортировки строк, возвращаемых инструкцией SELECT. Возможно указание, как имен, так и номеров столбцов. Сначала производится сортировка по первому указанному параметру, затем по следующим. Для сортировки в обратном порядке надо указать суффикс DESC.

```
SELECT * FROM countries
ORDER BY [country] DESC;
```

Q1 : запрос на выборку

COUNTRY	CAPITAL
Франция	Париж
Россия	Москва
Испания	Мадрид
Великобритания	Лондон

Запись: 1 из

Рис. 7.7. *Сортировка в обратном порядке*

Предложение GROUP BY

В инструкции SELECT задает столбцы, используемые для формирования групп выбранных строк. Используется в итоговых запросах. Например, вывести число мест съемок (используем функцию – Count(PLACE)) в разных странах (рис. 7.8). Список итоговых функций приведен в таблице 5.4 на стр. 37.

```
SELECT
    Count(PLACE) AS [Count-PLACE], COUNTRY
FROM PLACES
GROUP BY COUNTRY;
```

groups : запрос на выборку

Count-PLACE	COUNTRY
1	Великобритания
1	Испания
3	Россия
1	Франция

Рис. 7.8. *Итоговый запрос*

Предложение WHERE

Задает условие отбора в инструкции SQL. Применяется не только в операторе SELECT, но и в операторах DELETE и UPDATE. Его синтаксис уже неоднократно рассматривался, но приведем еще несколько примеров:

```
SELECT * FROM photos WHERE DATE>#1/1/2003#;
```

```
SELECT * FROM photos WHERE family LIKE "[A-H]*"
```

```
SELECT * FROM photos WHERE family="Иванов" AND DATE>#1/1/2003#;
```

Следует помнить смысл логических операций AND, OR, NOT (стр. 32), используемых для объединения нескольких условий. Для сравнения текстовых строк с заданными шаблонами широко применяется мощная инструкция LIKE (стр. 17). Следует учесть что в ANSI SQL (международный стандарт языка) в шаблоне LIKE используются другие символы.

Таблица 7.1. Символы шаблона LIKE в разных версиях SQL.

Access	ANSI	Описание
?	_	Один произвольный символ
*	%	Любое число (включая нулевое) произвольных символов
!	^	Обратное условие. Отсутствие в диапазоне

Стандарт ANSI SQL мы будем использовать в Web-страницах доступа к базам данных.

Задание 25. Создайте непосредственно на языке SQL несколько запросов, постарайтесь не пользоваться конструктором запросов. Это очень поможет в подготовке к экзамену по БД.
(*2 балла за каждый запрос*)

Инструкция DELETE

Инструкция DELETE удаляет одну или несколько строк из таблицы. Следует всегда использовать в этой инструкции предложение WHERE, в противном случае вы рискуете удалить все (!) строки из таблицы. Например, следующая инструкция удалит строки из таблицы Orders в которых поле Status есть символ O.

```
DELETE * FROM Orders WHERE [Orders].[Status]="O";
```

Еще раз напомним: будьте аккуратны с применение этой инструкции. При ознакомлении с ее работой сделайте копию таблицы, с которой вы собираетесь работать.

Инструкция INSERT

Инструкция INSERT вставляет в таблицу одну (или несколько) строк. В случае использования ключевого слово VALUES вставляется только одна строка.

```
INSERT INTO COUNTRIES (Country, Capital) VALUES ("Турция", "Стамбул");
```

Этот пример вставляет в таблицу COUNTRIES значения "Турция" и "Стамбул" как полей Country и Capital. В принципе можно не указывать в скобках названия столбцов, но тогда после слова VALUES должны идти значения всех столбцов в таблице, да еще и в том порядке, в котором они в ней находятся. Здесь легко сделать ошибку, поэтому лучше список столбцов писать явно, в этом случае их порядок может и отличаться от порядка в таблице. Инструкция все равно отработает правильно. Мы будем часто применять инструкцию INSERT при модификации базы данных через Web-интерфейс.

Инструкция SELECT ... INTO

Эта инструкция очень похожа на оператор SELECT с той лишь разницей, что результат работы SELECT помещается в новую таблицу. Если таблица с указанным именем существовала, то она удаляется и вместо нее появляется новая.

```
SELECT Title, Date, Customer INTO OrdersOld FROM Orders WHERE Status="D";
```

В данном примере из таблицы Orders извлекаются четыре столбца: Title, Date и Customer, и помещаются в новую таблицу OrdersOld. Причем выбираются только те строки в таблице Orders, у которых значение поля Status есть D. Само поле Status в итоговую таблицу OrdersOld не заносится.

Инструкция UPDATE

Эта инструкция обновляет записи в таблице. Обычно в ней всегда указывается и предложение WHERE. Список, разделенный запятыми, новых значений полей указывается после слова SET в формате *имя поля=значение*.

```
UPDATE OrdersOld SET Status = "O" WHERE Year(Date)<=2004;
```

Это предложение заменяет поле статус на значение O в тех записях таблицы OrdersOld, у которых год даты (используется встроенная функция, см. табл. 5.2 на стр. 34) меньше либо равен 2004.

Задание 26. Создайте непосредственно на языке SQL по одному из запросов с инструкциями Insert, Update, Delete, Select into.

(*2 балла за каждый запрос*)

Вопросы к мини-зачету по языку SQL

1. Опишите синтаксис и семантику инструкции SELECT
2. Как в запросе SQL указать наличие связи между таблицами
3. Для чего служит и где используется ключевое слово DISTINCT
4. Как вывести данные, отсортированные в обратном порядке значений какого-либо столбца
5. Для чего служит предложение Group by
6. Каким образом, и какие условия можно задавать на выборку строк из таблицы
7. Опишите синтаксис и семантику инструкции SELECT ... INTO
8. Опишите синтаксис и семантику инструкции INSERT
9. Опишите синтаксис и семантику инструкции DELETE
10. Опишите синтаксис и семантику инструкции UPDATE

За зачет – до 10 баллов

Глава VIII. Работа с формами в MS Access

Мы уже предварительно знакомились с формами в главе II (стр. 6). В ней разбирался вопрос о создании типовых форм с помощью мастера. С точки зрения удобства и частоты применения формы являются наиболее важными объектами в приложениях Access. Разработка сложных и «умных» по своему поведению форм – непростое дело. Мы познакомимся только с основами создания и разработки форм. Для более подробного изучения рекомендуется обратиться к книгам или многочисленным ресурсам в Интернет.

Создание простой формы для ввода данных

Попробуем создать форму для ввода данных самим разработчиком базы данных, не будем гнаться за эффектностью, а сосредоточимся на функциональности. Войдем в конструктор формы (рис. 8.1). Щелкнем правой клавишей мыши *вне области* данных, в любом месте серого поля. В появившемся диалоге (рис. 8.2) укажите источник записей, с которым будет оперировать форма. После этого всплывет дополнительное окно со списком полей (рис. 8.3). Если почему-то по правой клавише открылось окно свойств не формы, а другого объекта (скорее всего вы щелкнули мышью не на фоне), то вверху диалога свойств всегда можно выбрать объект, свойства которого вы хотите просмотреть или изменить (см. рис 8.3, там раскрыт список объектов).

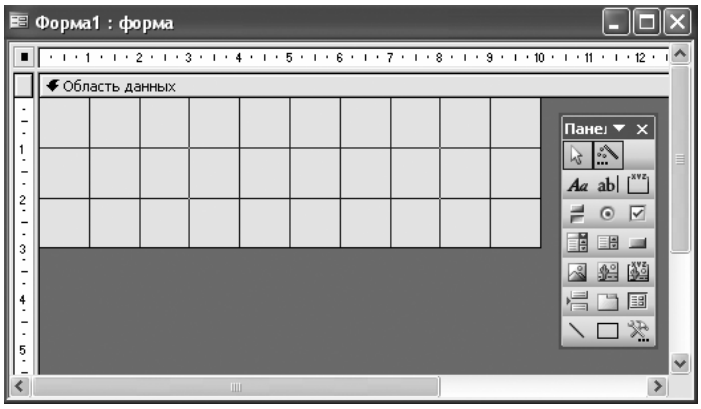


Рис. 8.1. *Общий вид конструктора формы*

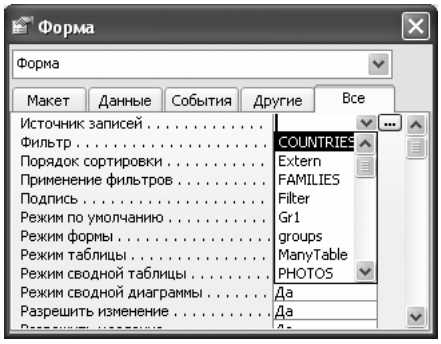


Рис. 8.2. *Выбор источника записей*

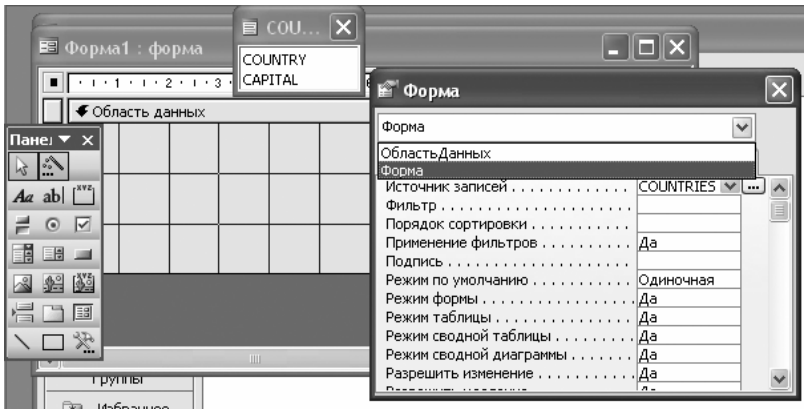


Рис. 8.3. *Список полей вверху рисунка*

Элементы управления добавляются с помощью всплывающей панели инструментов (рис. 8.4). Подробное их назначение описано в таблице 8.1.

Меню правой клавиши мыши для области данных открывает диалог (рис 8.5), в котором можно дополнительные свойства области данных.

Рис. 8.4. *Панель элементов конструктора формы*

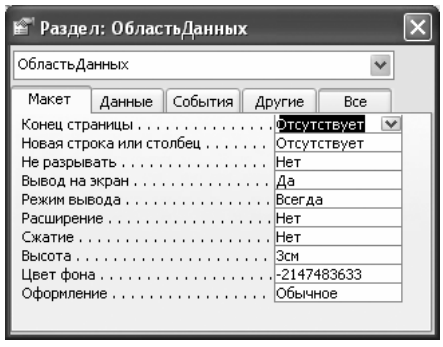


Рис. 8.5. *Свойства области данных*

Таблица 8.1. Назначение элементов *Панели элементов*.

Элемент	Применение	Элемент	Применение
	Выделение объекта базы данных		Активизация Мастеров управляющих элементов
	Создание текста		Отображение полей таблиц
	Создание группы элементов		Создание кнопки, которая в положении "Включено", изображается нажатой
	Создание переключателя (Истина/Ложь)		Устанавливает флажок опции
	Создание списка, для выбора значение или ввода		Создание разворачивающегося списка
	Создание кнопки для запуска макросов		Создание неизменяемого рисунка
	Создание свободной рамки для объекта OLE (неизменяемого от записи к записи)		Создание связывающей (с БД) рамки для объекта OLE (изменяемого от записи к записи)
	Разбиение на страницы		Создание подчиненного отчета
	Рисование линии		Рисование прямоугольника

Самым простым способом размещения элементов управления для ввода и вывода полей таблицы служит перетаскивание мышью полей из списка полей в область данных (рис 8.6). Их можно выровнять, для выравнивания поля данных и подписи раздельно, воспользуйтесь перетаскиванием за большой черный квадрат в левом верхнем углу элемента управления (тот же рис. 8.6).

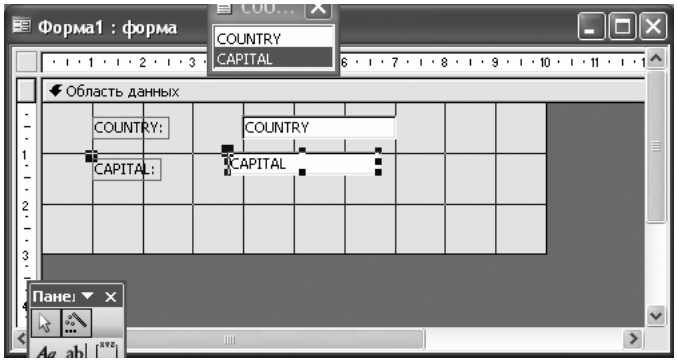


Рис. 8.6. *Перетаскивание и выравнивание полей*

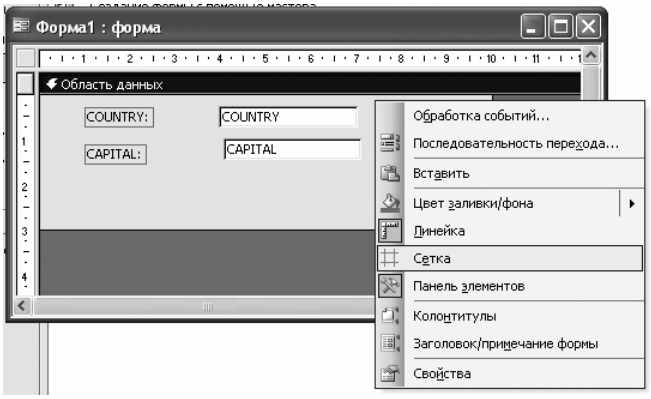


Рис. 8.7. *Включение линейки, сетки, заголовков*

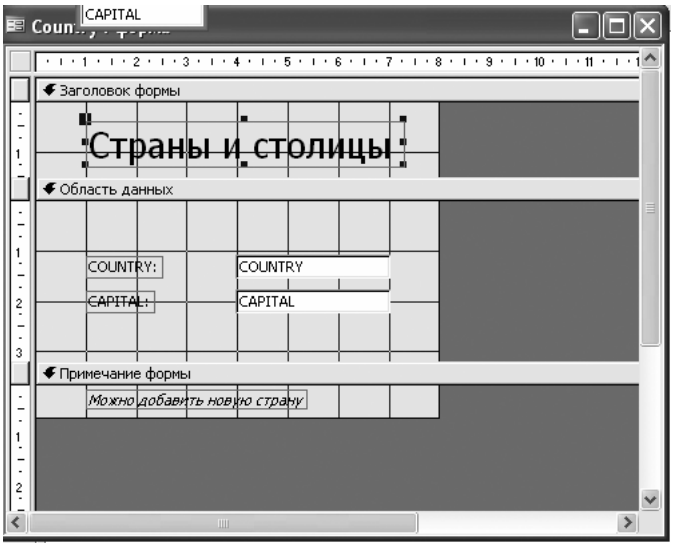


Рис. 8.8. *Окончательный вид формы в режиме конструктора*

Рис. 8.9. *Работа с данными через созданную форму*

Меню правой клавиши в области данных позволяет включить/отключить отображение линейки, сетки, добавить колонтитулы или заголовок (рис. 8.7). Например, можно включить опцию *заголовок/примечание формы*, добавить в них дополнительные текстовые элементы. Можно использовать стандартное для всех приложений MS Office изменение формата надписи (рис. 8.8). После конструирования формы ее нужно сохранить, после этого можно открыть уже в режиме работы с данными (рис. 8.9).

В конструкторе формы двойной щелчок по элементу управления открывает диалог свойств данного элемента. Их очень много, они сгруппированы по признакам, относящихся к внешнему виду элемента управления (макет), источнику данных, обработки событий и прочих свойств. Вкладка *Все* содержит свойства всех закладок в алфавитном порядке (рис 8.10).

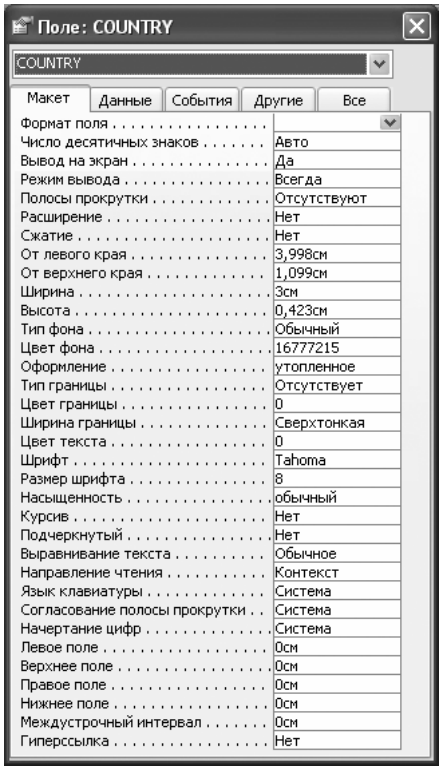


Рис. 8.10. *Окно свойств поля*

Рис. 8.11. *Выпадающие списки для таблицы, связанной соотношениями «многие-ко-многим»*

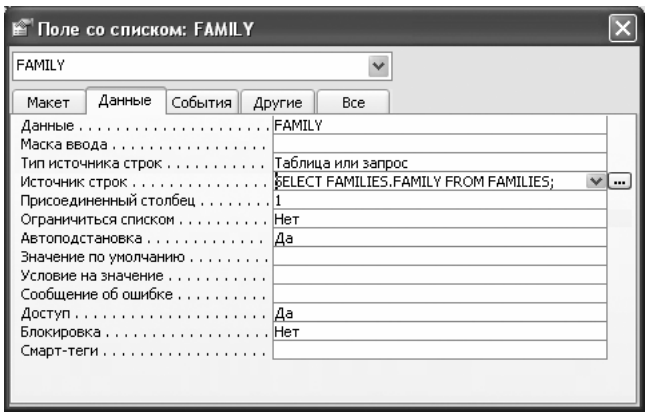


Рис. 8.12. *В свойстве поля есть источник строк выпадающего списка*

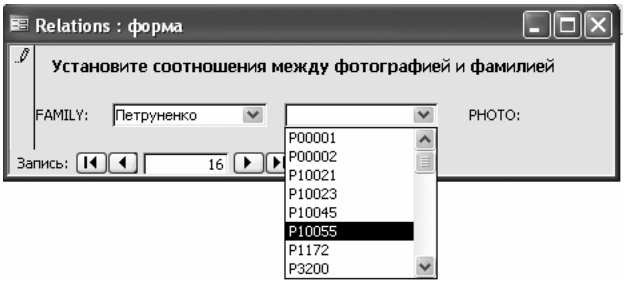


Рис. 8.13. *Работа с формой, содержащей выпадающие списки*

Крайне удобным элементом управления является выпадающий список. Если таблица, к которой строится форма, содержала поля со свойством подстановки значений, то при перетаскивании названий полей в область данных формы (рис. 8.6) будут автоматически созданы *выпадающие списки* (рис. 8.11). В свойствах такого поля вы обнаружите источник строк для этого списка – соответствующий SQL-запрос.

Работа такой формой (рис. 8.13) позволяет очень быстро вводить информацию в таблицы со связями.

Задание 27. Сконструируйте самостоятельно форму ввода и просмотра данных для простой таблицы. Сконструируйте форму с выпадающими списками. (от 2 до 6 *баллов*)

Дополнительные настройки формы

Вы можете модифицировать форму. Например, запретим изменение размеров окна формы (по умолчанию это допустимо), не будем разрешать ввод данных с помощью этой формы, сделаем ее предназначенной только для просмотра. Для этого надо просто установить некоторые свойства в диалоге *свойства формы* (рис 8.14).

Давайте добавим в форму еще элемент управления *кнопку* (рис. 8.15), определим ее внешний вид (рис. 8.16), установим вид всплывающей подсказки и прочие свойства (рис. 8.17) с помощью диалога *свойства объекта*. Вид получившейся рабочей формы показан на рис. 8.18.

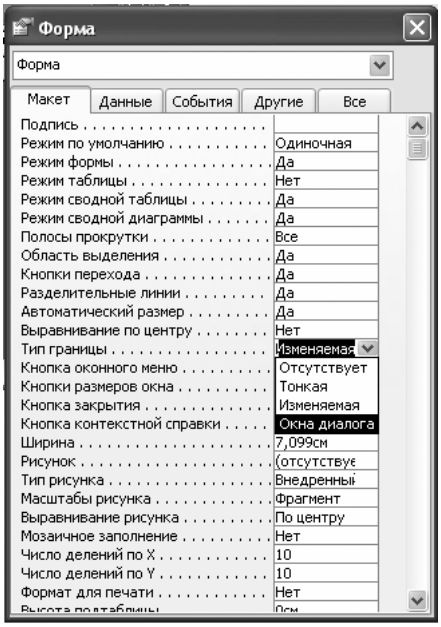


Рис. 8.14. *Настройка свойств формы*

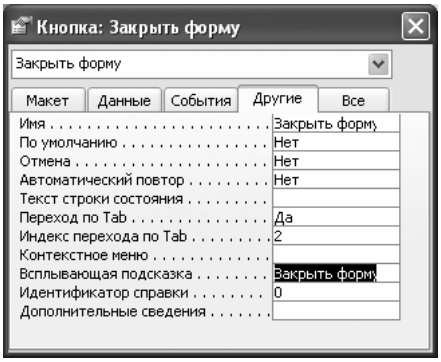


Рис. 8.17. *Оформление кнопки*

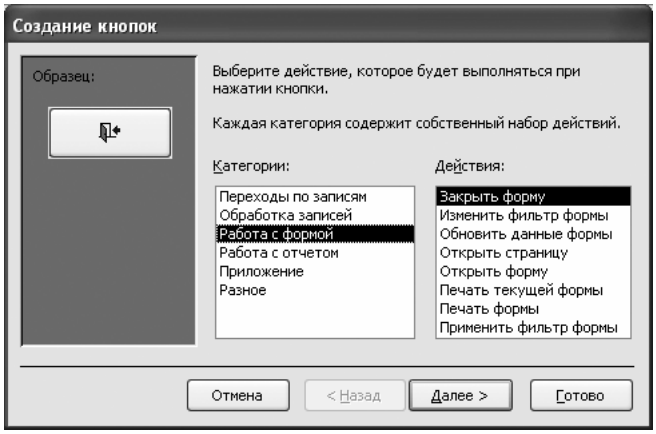


Рис. 8.15. *Добавление кнопки*

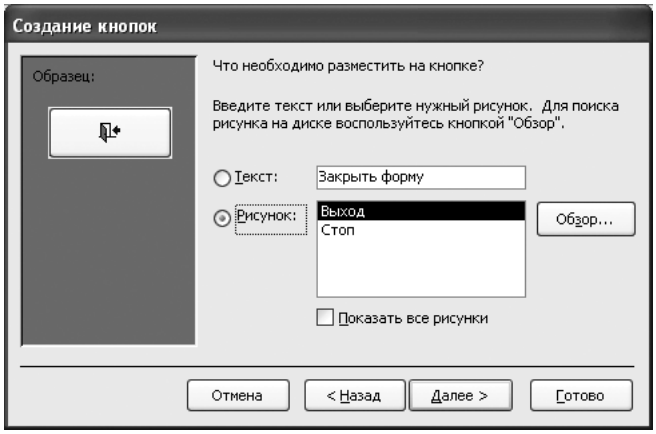


Рис. 8.16. *Оформление кнопки*

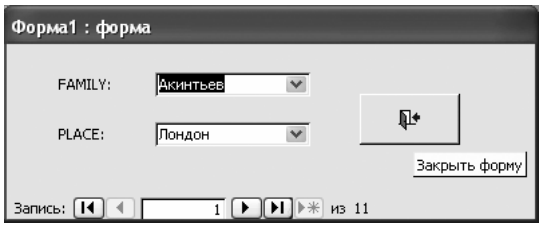


Рис. 8.18. *Форма в действии*

Задание 28. Модифицируйте уже созданные формы, установив дополнительные свойства, максимально улучшив ее интерфейс.

(*от 2 до 4 баллов*)

Задание 29. Разработайте главную кнопочную форму.

(*от 2 до 6 баллов*)

Создание главной кнопочной формы

Очень удобным при работе с базой данной является создание кнопочной формы, управляющей приложением. Разместим на форме только кнопки, и каждой кнопке присвоим действие, связанное с работой с формой, например, открыть другую форму (рис. 8.19). Далее следует выбрать имя формы, которую надо открыть (рис. 8.20), задать рисунок или надпись на кнопке (рис. 8.21). Вид такой формы в конструкторе представлен на рис. 8.22. Форма в действии показана на рис. 8.23. Можно настроить MS Access (меню *Сервис – Параметры запуска*), чтобы при открытии файла БД сразу открывалась главная кнопочная форма, управляющая вашим приложением.

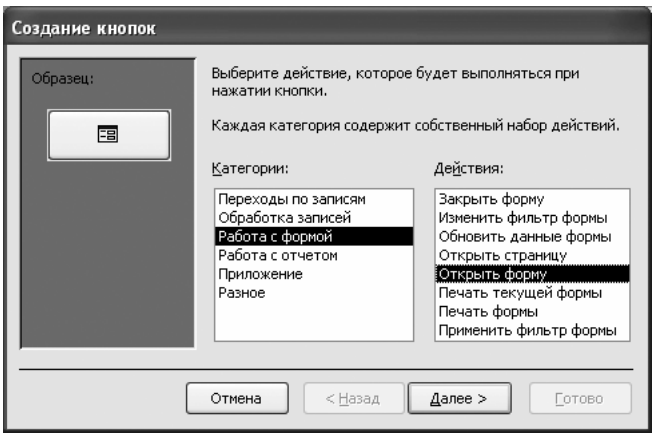


Рис. 8.19. *Создание кнопки управляющей формы*

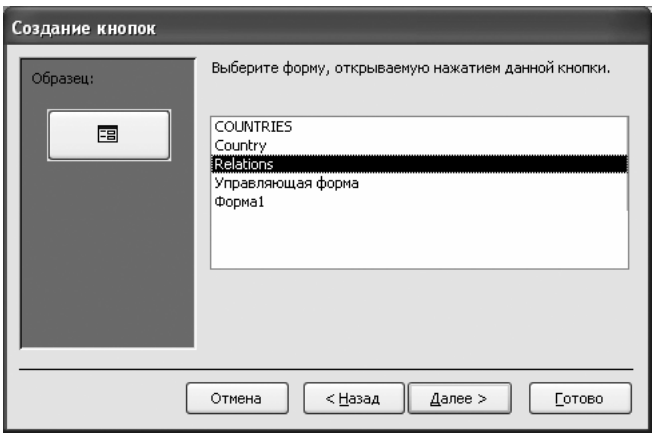


Рис. 8.20. *Уточнение действия формы*

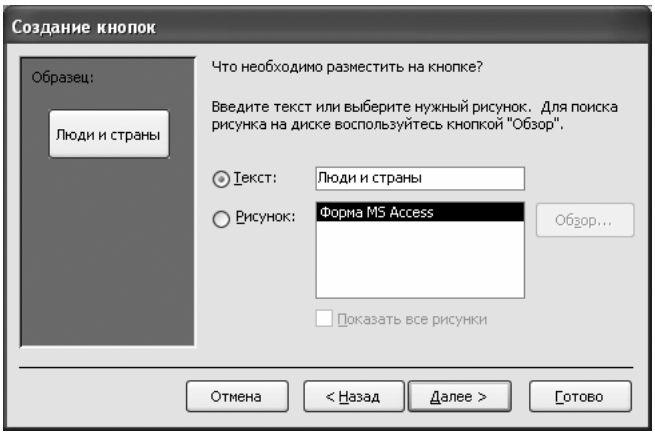


Рис. 8.21. *Определение подписи*

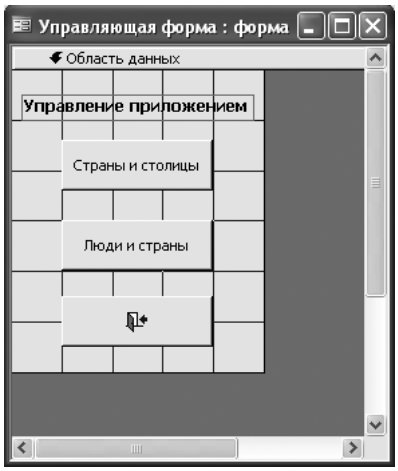


Рис. 8.22. *Кнопочная форма в конструкторе*

Рис. 8.23.
Форма в действии

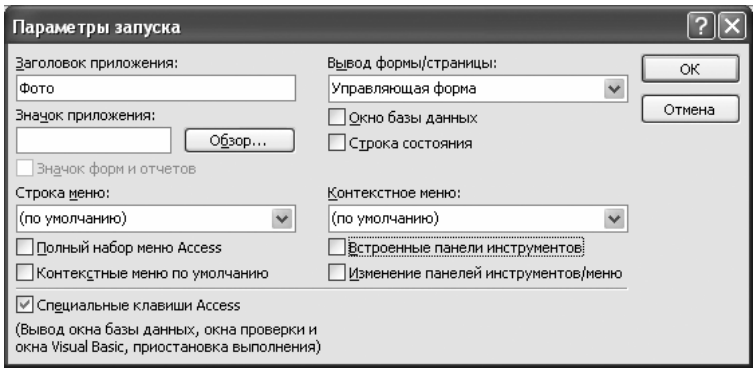


Рис. 8.24. *Определение параметров запуска*

Чтобы отменить этот режим, удерживайте клавишу *Shift* при запуске файла или открытии его через меню *Файл*.

Модификация формы, созданной мастером

Непосредственно создавать форму «с нуля» в конструкторе может быть достаточно трудоемким делом. Поэтому в случаях, близких к стандартным, даже опытные разработчики прибегают к мастеру форм, создают «рыбу», и модифицируют ее в конструкторе форм.

Создадим ленточную форму к таблице COUNTRIES и модифицируем ее. При запуске мастера сразу запрашивается источник данных (рис. 8.25), затем вид формы (рис. 8.26). В конце работы мастера можно перейти сразу в конструктор (рис. 8.27).

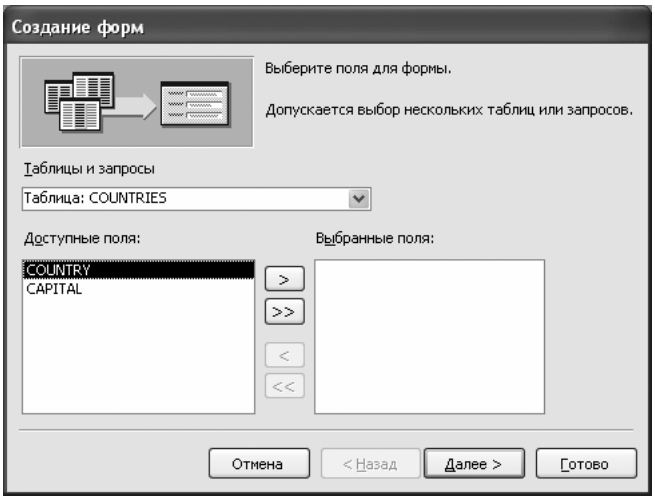


Рис. 8.25. *Определение источника данных в мастере форм*

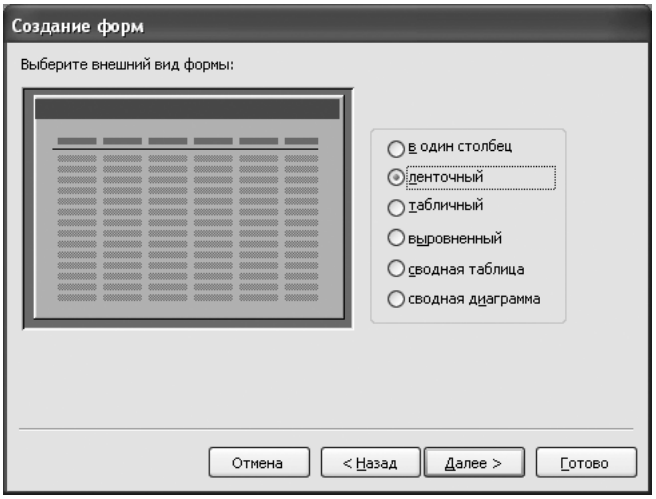


Рис. 8.26. *Выбор внешнего вида формы*

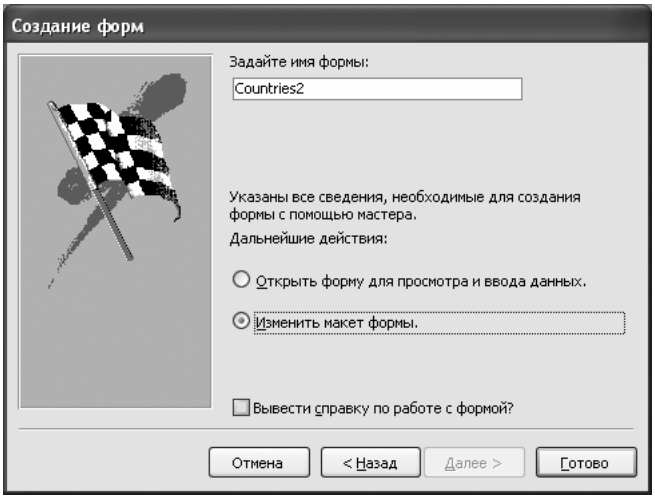


Рис. 8.27. *Переход в конструктор*

Рис. 8.29. *Стандартная ленточная форма в режиме конструктора*

В режиме конструктора вы можете изменять макет формы по своему усмотрению также, как и в случае ее самостоятельной разработки.

Задание 30. Используйте мастер форм и модифицируйте стандартную форму в конструкторе. (*от 2 до 4 баллов*)

Настройка внешнего вида формы

Для конечного пользователя базы данных важен внешний вид элементов управления. продуманность расположения полей, кнопок, правильное выравнивание элементов и т.п. Для автоматического выравнивания элементов управления выделите первый элемент и удерживая клавишу Shift на последующие. Затем воспользуйтесь меню *Формат*, а именно командами *Выровнять*, *Размер*, *Интервал по горизонтали*, *Интервал по вертикали*. (рис. 8.30). Воспользуйтесь элементами оформления *Прямоугольник*, *Линия* для разделения формы на логические части.

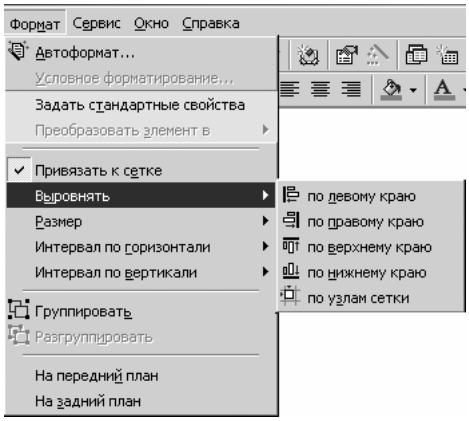


Рис. 8.30. *Меню «Формат»*

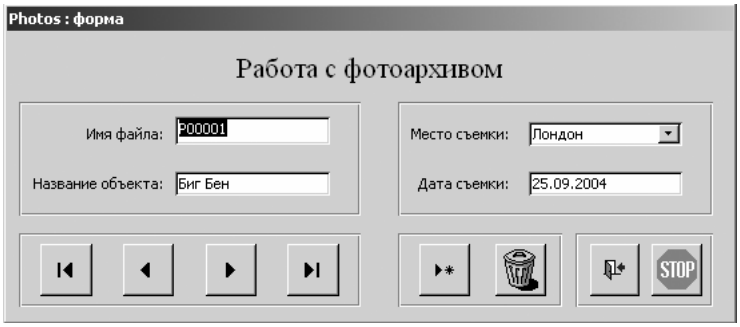


Рис. 8.31. *Форма, разработанная вручную*

Откажитесь от стандартных, предлагаемых по умолчанию элементов управления. Добавьте свои кнопки, разработайте, может быть, свое цветовое оформление (рис. 8.31). Для этого в свойствах формы уберите такие элементы как *Область выделения*, *Кнопки перехода*, *Разделительные линии* и т.п., установите неизменяемую границу (рис 8.33). Создайте свои навигационные кнопки и кнопки управления формой и приложением (рис. 8.34).

Рис. 8.33 *Меню «Свойства формы» с отключенными стандартными элементами*

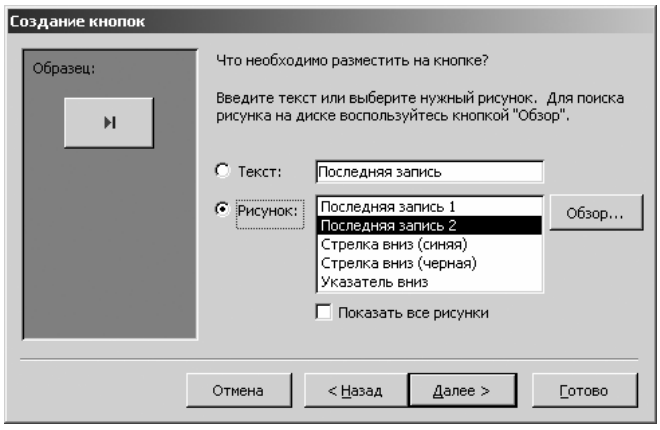


Рис. 8.33 *Создание кнопок*

Задание 31. Разработайте свой дизайн форм вашего приложения.

(*от 2 до 4 баллов*)

Использование связанных форм для выбора определенных записей из другой формы

Вы, наверное, обратили внимание, что при обращении к запросу с параметром всплывают окна, в которые надо вводить значения, что не очень удобно. Существует довольно простой способ задать параметры запроса с помощью формы. Для этого создайте сначала форму, которая выводит все записи из какой-либо таблицы, запроса (рис. 8.34).

Рис. 8.34. *Ленточная форма, вызванная непосредственно*

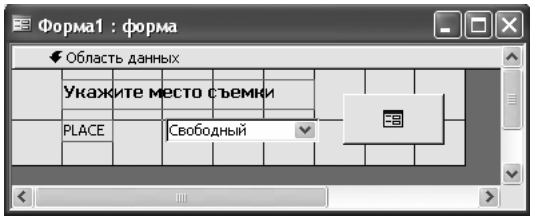


Рис. 8.35. *Форма-запрос в конструкторе*

Далее постройте форму, в которой будет одно или несколько текстовых полей (возможно, выпадающих списков). Для источника данных списков укажите таблицы или список фиксированных значений (рис. 8.35). Затем добавьте кнопку, на которую назначьте действие *Открыть форму*, но отметьте флаг *Открыть форму для отобранных записей* (рис. 8.36).

Для каждого поля управляющей формы и поля подчиненной формы установите соответствие (рис. 8.37). Для более привлекательного вида этого диалога можно задать удобные имена полей в управляющей форме.

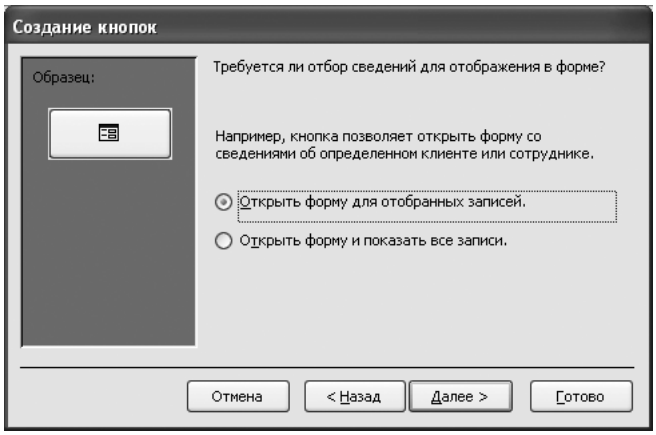


Рис. 8.36. *Выбор отобранных записей*

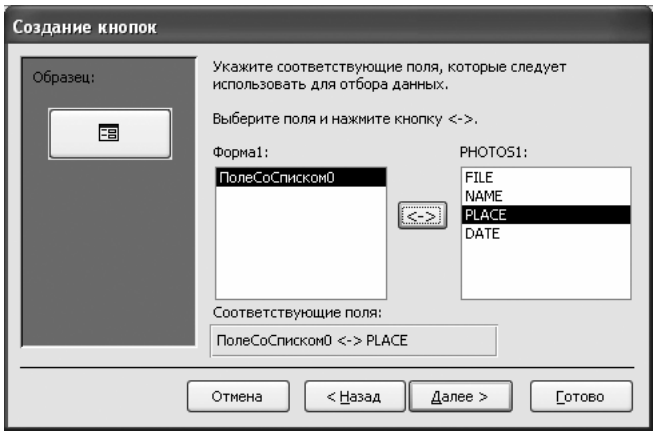


Рис. 8.37. *Установка соответствий полей*

На рис. 8.38 представлен результат совместной работы двух форм.

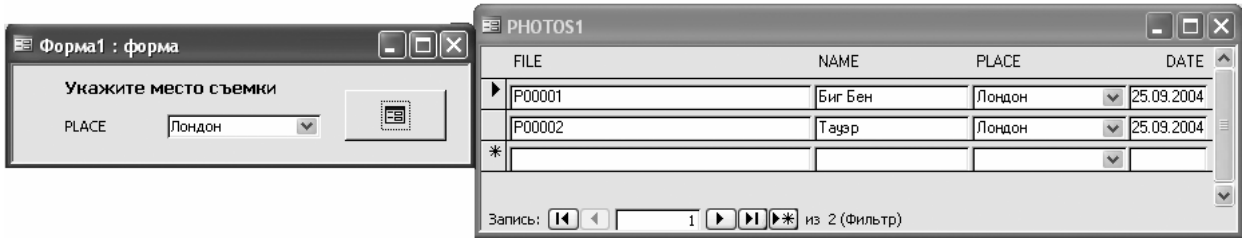
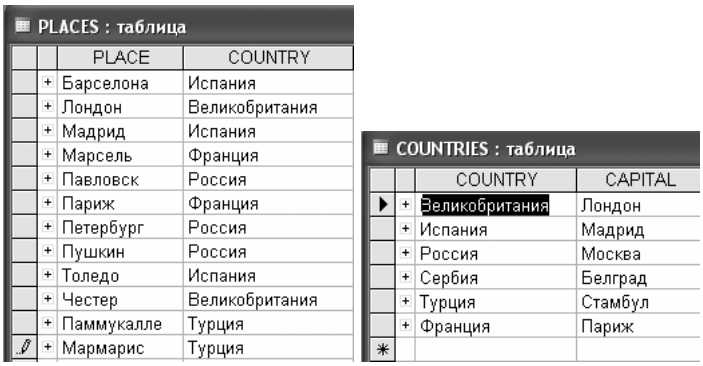


Рис. 8.38 *Ленточная форма, вызванная непосредственно*

Создание и внедрение подчиненных форм

Пусть у нас есть две таблицы, связанные соотношением «один-ко-многим» (рис. 8.39). В этом случае можно построить сложную форму с использованием вложенной формы. Для начала попробуем сделать сложную форму в режиме мастера. Для этого необходимо указать в качестве источника данных не одну, а две связанных между собой таблицы (рис 8.40). В следующем диалоге надо выбрать способ представления: подчиненная (внедренная) форма (рис. 8.41) или связанная форма (рис. 8.42). В последнем случае результат будет такой же, как был уже описан предыдущем параграфе (т.е. связь через кнопку открытия формы). В этот раз мы выбираем подчиненную форму (рис. 8.41).



PLACES : таблица

	PLACE	COUNTRY
+	Барселона	Испания
+	Лондон	Великобритания
+	Мадрид	Испания
+	Марсель	Франция
+	Павловск	Россия
+	Париж	Франция
+	Петербург	Россия
+	Пушкин	Россия
+	Толедо	Испания
+	Честер	Великобритания
+	Паммукалле	Турция
+	Мармарис	Турция

COUNTRIES : таблица

	COUNTRY	CAPITAL
+	Великобритания	Лондон
+	Испания	Мадрид
+	Россия	Москва
+	Сербия	Белград
+	Турция	Стамбул
+	Франция	Париж

Рис. 8.39. *Таблицы PLACES и COUNTRIES*

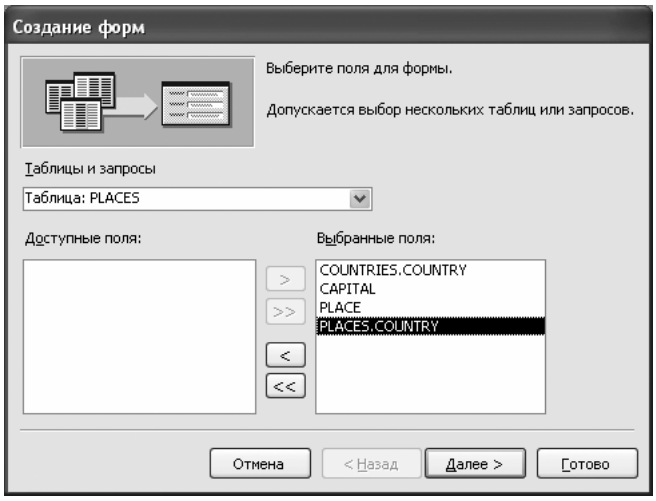


Рис. 8.40. *Таблицы PLACES и COUNTRIES*

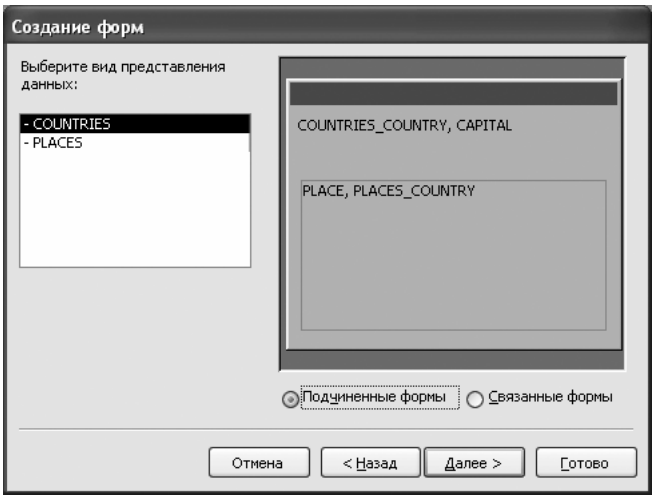


Рис. 8.41. *Выбор подчиненной формы*

Рис. 8.42. *Выбор связанной формы*

Затем следует определить вид подчиненной формы (рис. 8.43). В общем случае оказывается удобен либо табличный, либо ленточный. Окончательный вид сложной формы представлен на рис. 8.44. При переходе к записи новой страны в главной форме, в подчиненной форме отображается список объектов в этой стране.

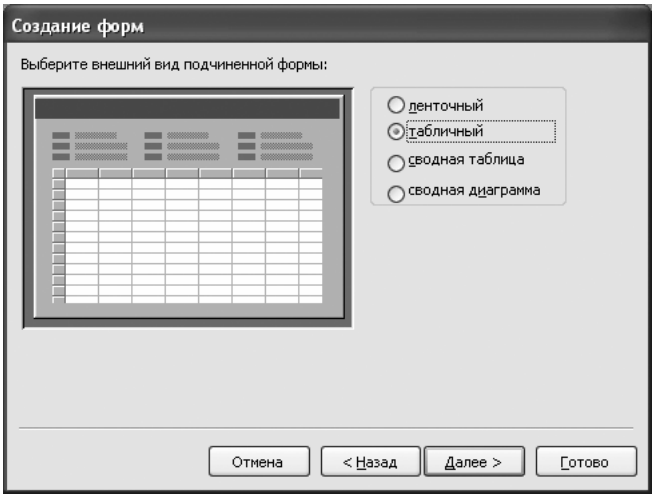


Рис. 8.43. *Выбор вида подчиненной формы*

Рис. 8.44. *Окончательный вид формы с внедренной подчиненной формой*

Создание и внедрение подчиненных форм

Решим теперь задачу о внедренных формах вручную, т.е. построим ее с помощью конструктора форм. Для начала создадим форму (лучше ленточную или табличную) для таблицы, находящейся на стороне «многие» (рис. 8.45). Эта форма в дальнейшем станет внедренной. Потом перейдем к построению главной формы, в качестве источника данных для нее используем таблицу, находящуюся на стороне «один». Вставим в главную форму элемент управления «подчиненная форма» (рис. 8.46). В диалоге выберем в качестве данных подчиненной формы созданную нами ленточную форму (рис. 8.47).

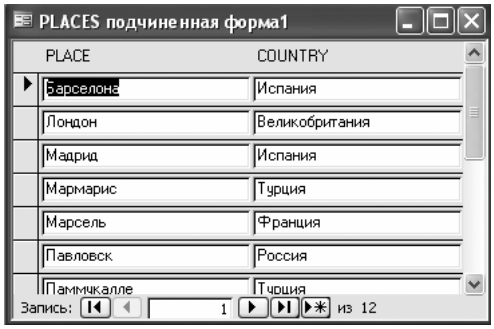


Рис. 8.45. *Форма, которая будет подчиненной*

Рис. 8.46. *Вставка элемента управления «Подчиненная форма»*

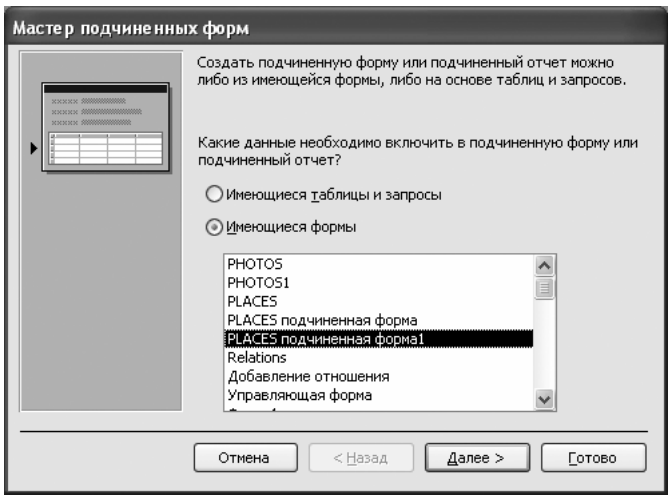


Рис. 8.47. *Выбор объекта для подчиненной формы*

Определим связь между главной и подчиненной формой. Это можно сделать автоматически, на основании связанных полей (рис. 8.48) или вручную, подробно указав связи (рис. 8.49).

Задание 32. Создайте сложную форму для двух связанных таблиц в режиме мастера (*от 1 до2 баллов*)

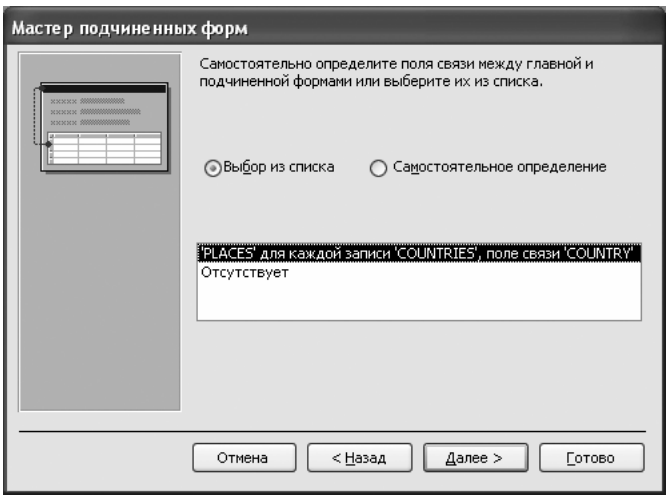


Рис. 8.48. *Определение связи между формами*

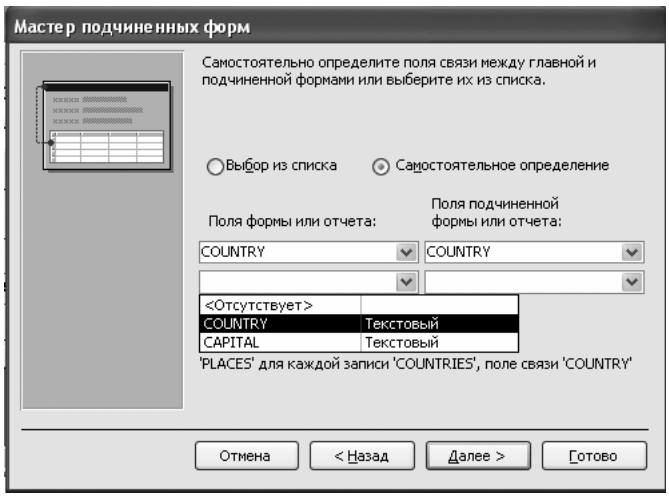


Рис. 8.49. *«Ручное» определение*

Общий вид получившегося элемента управления в режиме конструктора показан на рис. 8.50, а работающая форма представлена на рис. 8.51.

Рис. 8.50. *Внедренная форма в конструкторе*

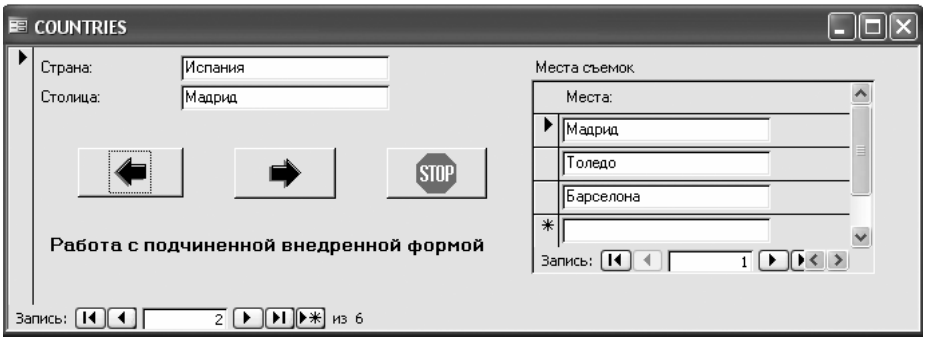


Рис. 8.51. *Внедренная форма в действии*

Задание 33. Создайте сложную форму для двух связанных таблиц.

(*от 3 до 6 баллов*)

Построение форм к запросам с параметрами

Напомним, что в главе V (стр. 40) мы использовали параметры запросов. Для каждого такого параметра всплывало окно, в котором предлагалось ввести значение параметра. Если построить форму к такому запросу, то при вызове формы также будут всплывать окна с параметрами (рис. 5.25). Это, конечно, неудобно. Хотелось бы в специальной форме задать сразу все параметры запроса и построить форму с результатами такого запроса. Чтобы это сделать, постройте форму, содержащую все поля, необходимые для запроса. Задайте какие-либо информативные имена полей, чтобы на них можно было удобно ссылаться в дальнейшем (рис. 8.52). Добавьте кнопку открытия формы для запроса с параметрами (рис. 8.53).

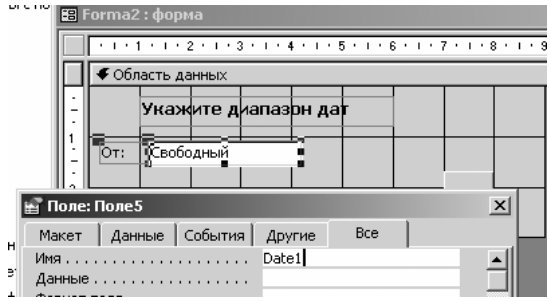


Рис. 8.52. *Определение имени текстового поля*

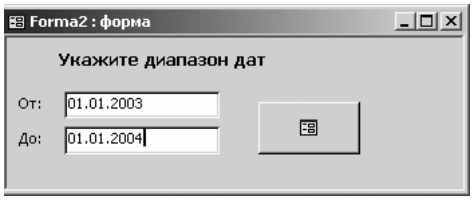


Рис. 8.53. *Форма с параметрами запроса*

Теперь все дело в специальных именах параметров в запросе. Вспомним, что для задания параметра запроса раньше мы просто писали любой текст, заключая его в квадратные скобки. Если мы хотим, чтобы запрос «понял», что недостающие параметры следует брать из формы, надо использовать в нем имена вида [Forms]![*Имя формы*]![*Имя поля*]. Например, в нашем случае это может быть [Forms]![Form2]![Date1] (рис. 8.54). Этот способ, в сочетании со всей мощью SQL-запросов, позволяет строить последовательность форм произвольной сложности.

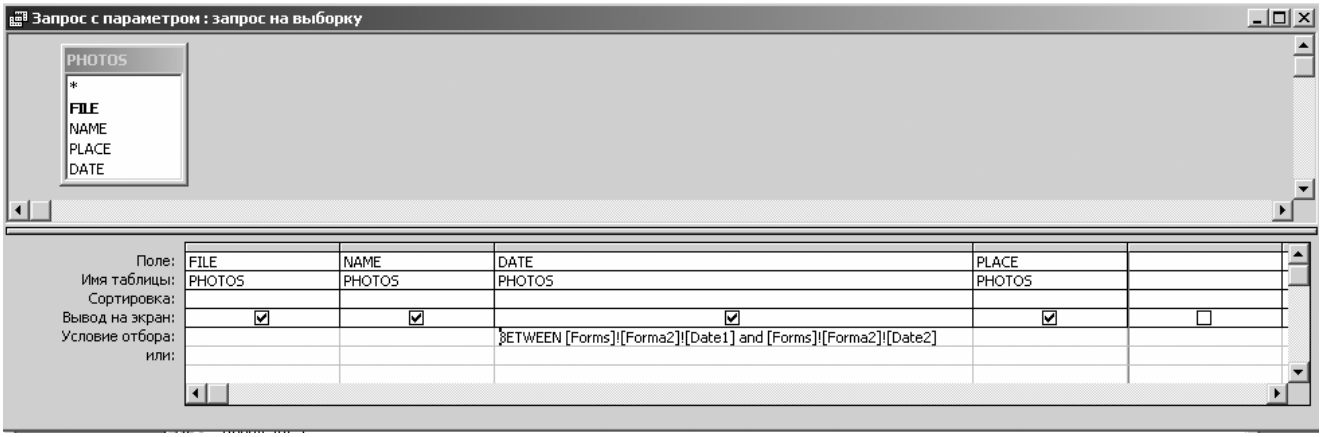
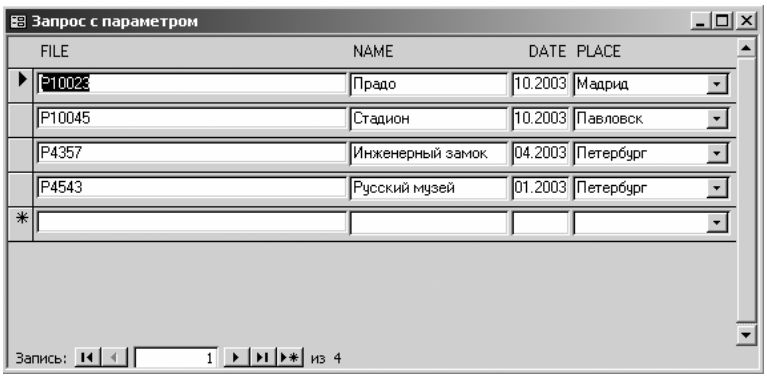


Рис. 8.54. *Построение запроса с параметрами*



Задание 34.
Постройте форму, содержащие параметры запроса, и форму к запросу с параметрами.

(*4-6 баллов*)

Рис. 8.55. *Вызов формы, построенной к запросу с параметрами*

Использование фильтров

Находясь в режиме просмотра данных в форме (рис. 8.56) можно, как это было и в таблицах, построить и применить фильтр к данным с тем, чтобы отобрать не все, а только определенные записи. Для этого нажмите кнопку *Изменить фильтр* в панели инструментов (рис. 8.57). Форма примет следующий вид (рис. 8.58). Введите условия отбора записей так же, как это было описано на стр. 29. После нажмите кнопку *Применить фильтр* (рис. 8.59). Теперь в форме будут отображаться только записи, удовлетворяющие заданным условиям отбора.

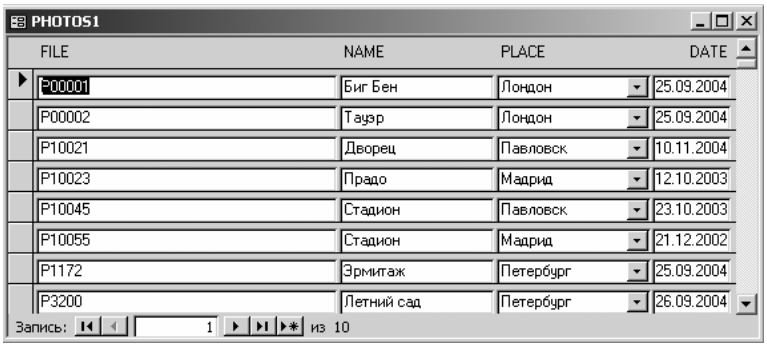


Рис. 8.56. *Форма без применения фильтра*

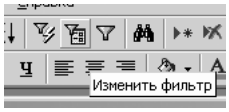


Рис. 8.57. *Кнопки создания и применения фильтра.*

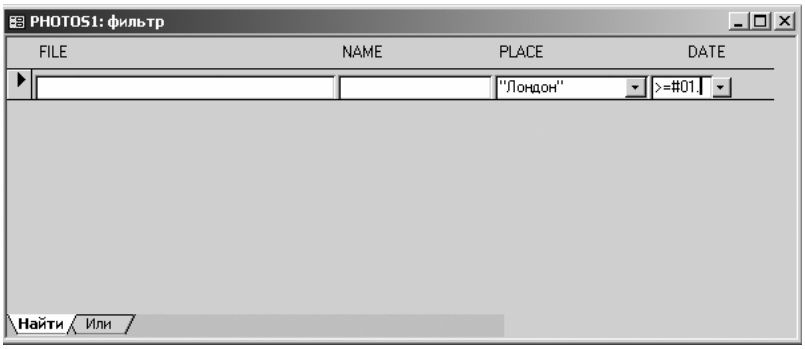


Рис. 8.58. *Создание фильтра*

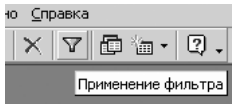


Рис. 8.59. *Вид кнопки применения фильтра в режиме построения фильтра.*

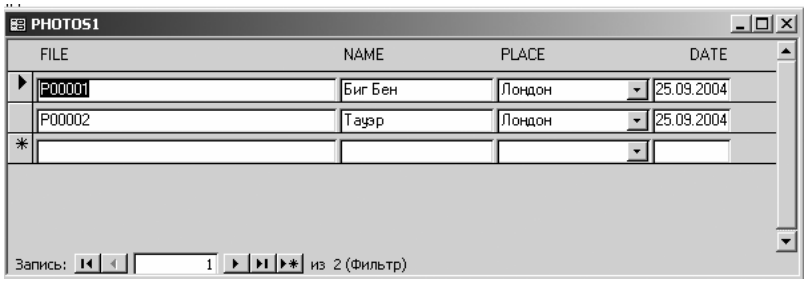


Рис. 8.60. *Форма после применения фильтра*

Улучшение функциональности формы

Определение последовательности перехода

Если пользователь будет использовать клавиатуру для ввода данных в форму, то очень удобно в этом случае перемещаться от одного объекта управления к другому с помощью клавиши *Tab*. Для изменения последовательности перехода выберите команду меню «*Вид–Последовательность перехода…*», мышкой перетащите объекты в списке для выполнения переходов в заданной последовательности (рис. 8.61).

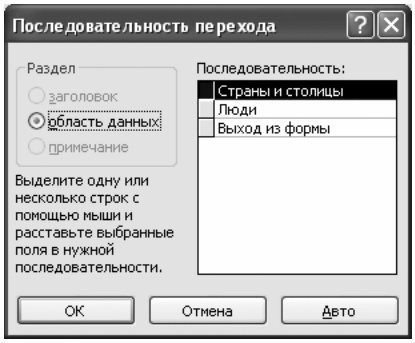


Рис. 8.61. *Определение последовательности перехода*

Использование вкладок

Если ваша форма перегружена элементами (особенно редко используемыми), то имеет смысл использовать такой удобный элемент управления как *вкладка*. Найдите объект управления *вкладка* в панели инструментов и создайте его в форме. Активизируйте определенную закладку и добавьте в нее желаемые поля, измените имя закладки. Если поля уже существовали в форме до применения вкладок, отметьте их, вырежьте (*Shift-Del*) и вставьте в страницу вкладки (рис. 8.62). Полученные результат будет великолепен (рис. 8.63).

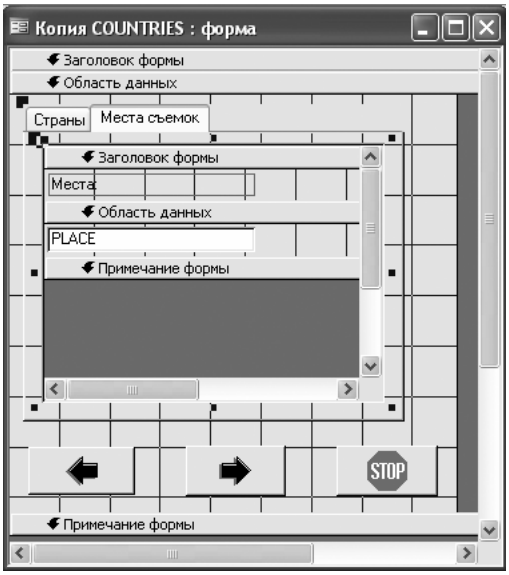


Рис. 8.62. *Конструирование вкладок*

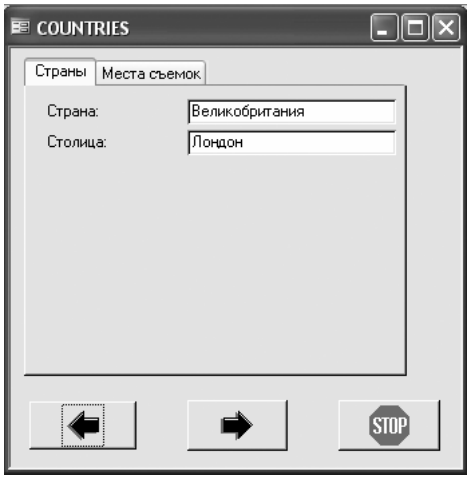


Рис. 8.63. *Вкладки в действии*

Применение условного форматирования

Условное форматирование является весьма выразительным средством улучшения читаемости форм. Выберете команду «*Формат–Условное форматирование…*» (рис. 8.64).

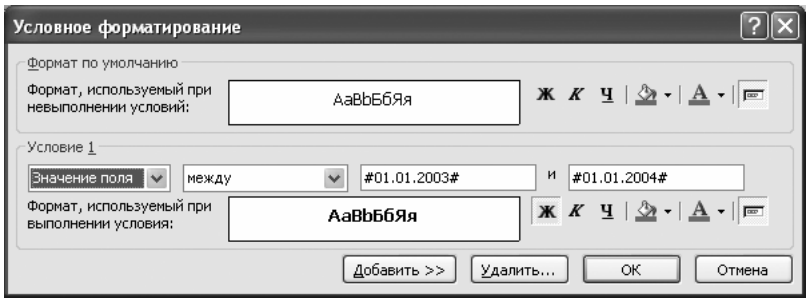


Рис. 8.64. *Вкладки в действии*

Укажите, как следует отформатировать поле при разных условиях. Если одно условия мало, то нажмите кнопку «*Добавить>>*». Посмотрите на результат (рис. 8.65).

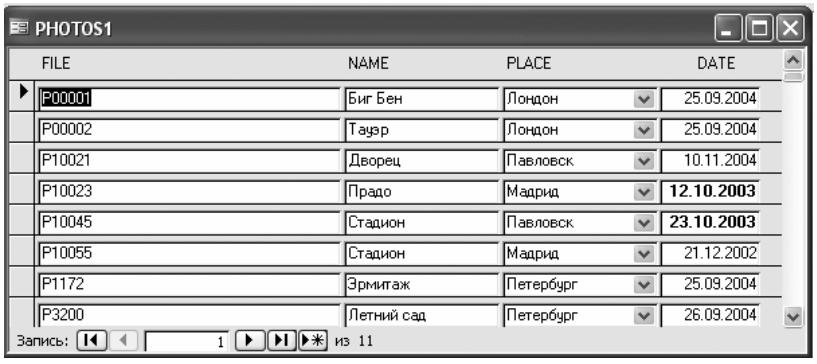


Рис. 8.65. *Вкладки в действии*

Задание 35.
Используйте в формах закладки и/или условное форматирование для улучшения представления данных. Не переусердствуйте!

(*4-6 баллов*)

Зачет по формам

Можно сдавать постепенно на уроках

1. **Главная форма приложения**
   - Оценка функциональности – 5 баллов
   - Оформление и дизайн – 5 баллов

   Всего: 10 баллов

2. **Формы запросов**
   - Оценка каждой формы – 2 балла за каждую
   - Использование запросов с параметрами – 1 балл дополнительно за каждый
   - Использование внедренных форм – 2 балла дополнительно за каждую
   - Использование своего дизайна управляющих элементов – 1 баллу за каждый

   При 5-6 формах возможно набрать: 30 баллов, выше 36 баллов не ставится.

3. **Внесение данных**
   - От 30 до 50 элементов – 3 баллов
   - От 50 до 100 элементов – 6 баллов
   - Свыше 100 элементов – 9 баллов

4. **Использование отчетов**
   - Содержание – 1 балл за отчет
   - Оформление отчета – 2 балла за отчет

   При 3-4 отчетах возможно набрать около 10 баллов, выше 12 баллов не ставится.

Таким образом, к 02.02.05 можно набрать до **58 баллов**

После сдачи этой программы, можно приступать к созданию документации к БД, которая состоит из следующих пунктов:

1.	Постановка задачи (техническое задание)	1-2 стр.	4 балла
2.	Описание главной формы приложения (с иллюстр.)	2-4 стр.	8 баллов
3.	Описание форм запросов (с иллюстрациями)	1 стр. на каждую	3 балла за каждую
4.	Описание структуры БД (с иллюстрацией)	1-2 стр.	5 баллов

За документацию можно заработать до **35 баллов**

Глава IX. Создание отчетов в MS Access

Отчеты предназначены для вывода информации из БД на печать. Разработка отчета очень похожа на разработку формы

Создание простого отчета

Хотя можно создавать самые простые отчеты в режиме мастера, мы перейдем сразу к созданию в режиме конструктора. Как и для формы, необходимо выбрать источник данных: таблицу или запрос (рис. 9.1). Конструктор запроса позволяет задать верхний и нижний колонтитул страницы. В меню «Вставка» есть пункт, позволяющий организовать нумерацию страниц. Данные из запроса или таблицы размещаются в области данных, при этом вы задаете раскладку только одной строки (рис. 9.2). Можно провести аналогию между отчетом и ленточной формой. В отличие от формы, вы всегда можете определить сортировку и группировку отображаемой информации (рис. 9.3). Также как и для формы, используйте меню «Формат – Выровнять» и «Формат – Размер» для придания аккуратного вида документу (рис. 9.4).

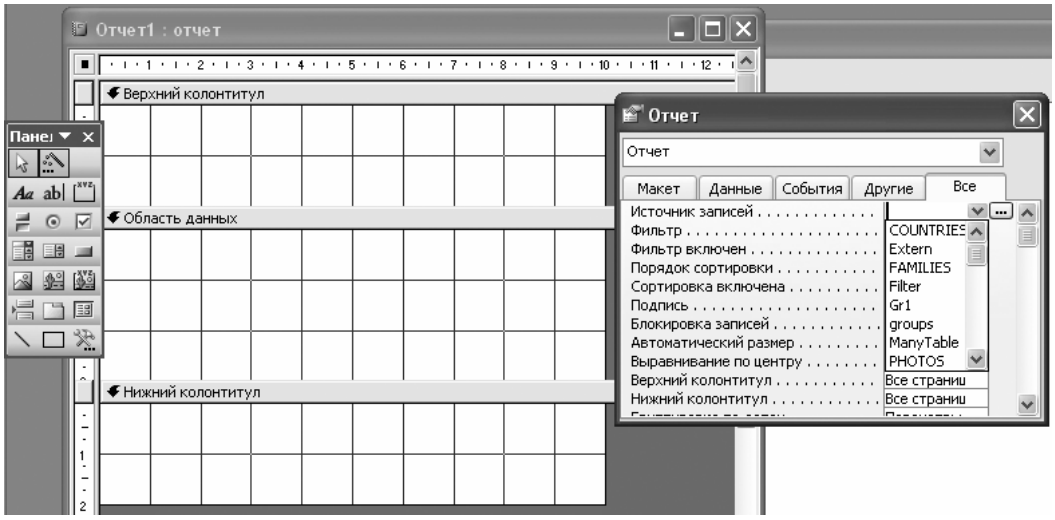


Рис. 9.1. *Конструктор отчета*

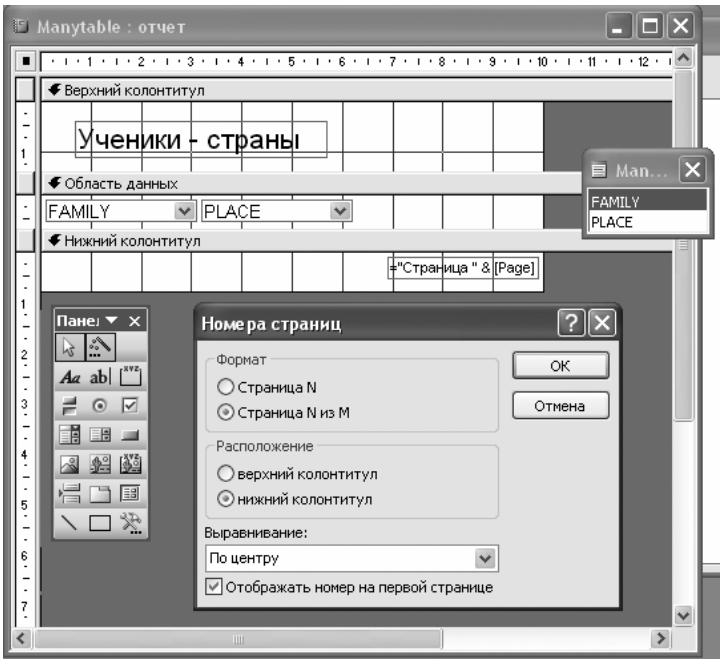


Рис. 9.2. *Оформление верхнего и нижнего колонтитула*

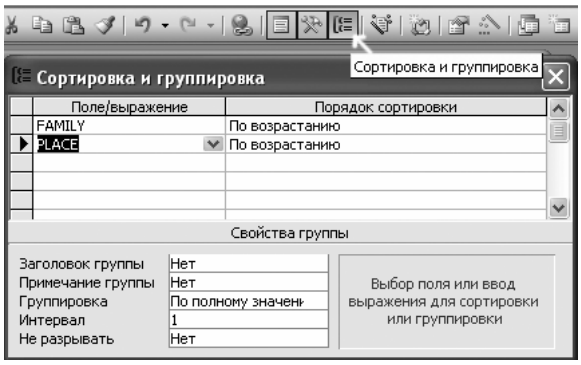


Рис. 9.3. *Сортировка и группировка информации*

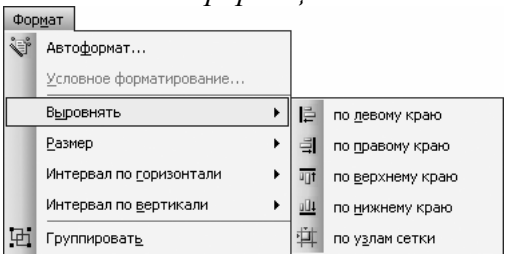


Рис. 9.4. *Выравнивание объектов*

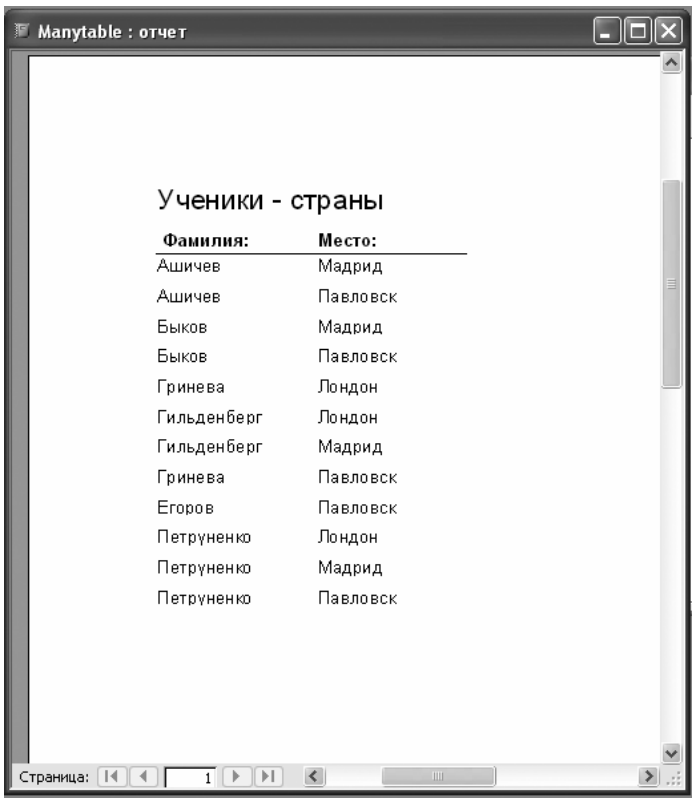


Рис. 9.5. *Готовый отчет*

Готовый отчет представлен на рис. 9.5. Теперь можно его просмотреть и распечатать на принтере (рис. 9.5).

Как вы видите, в этом отчете присутствует избыточная информация: повторяются фамилии. Чтобы этого избежать следует воспользоваться группировкой данных. Сделать это можно в том же окне «Сортировка и группировка», выбрав параметр FAMILY в качестве заголовка группы (рис. 9.6).

После этого удалить параметр FAMILY из области данных (можно вырезать – Shift-Del) и поместите его в заголовок группы (рис. 9.7). Выровняйте элементы в отчете, задайте форматирование разными шрифтами для улучшения читаемости документа.

Вид нового отчета представлен на рис. 9.8. Как мы видим, его наглядность значительно выше по сравнению с отчетом на рис. 9.5.

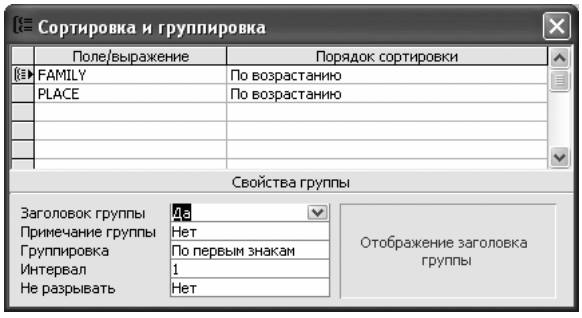


Рис. 9.6. *Создание группировки*

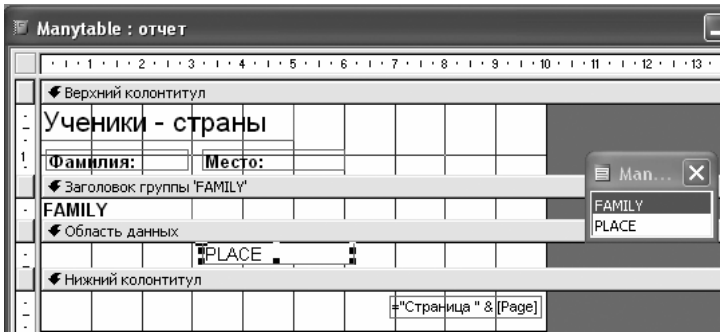


Рис. 9.7. *Определение заголовка группы*

Задание 36. Снабдите вашу БД отчетами. Как уже излагалось на стр. 67 один отчет может быть оценен от 1 до 3 баллов.

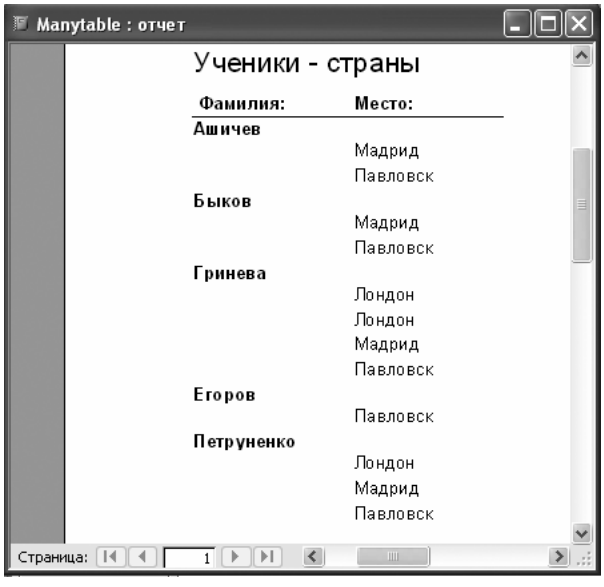


Рис. 9.8 *Вид отчета с группировкой данных*