Тема 6. ТЕХНОЛОГИИ БАЗ ДАННЫХ

6.1. Базы данных: понятие, назначение и архитектура.

6.2. Модели данных.

6.3. Системы управления базами данных (СУБД). Обзор современных СУБД и направления их развития.

6.4. Функциональные возможности СУБД на примере MS Access.

6.1. Базы данных: понятие, назначение и архитектура

База данных (БД) – это именованная совокупность данных, отображающая состояние объектов, их свойства и взаимоотношения в некоторой предметной области.

Предметная область – это часть реального мира, которая описывается и моделируется с помощью БД.

Объект – предмет, вещество, событие, лицо, явление, абстрактное понятие, т. е. все то, что может характеризоваться набором значений некоторой совокупности атрибутов.

Атрибут – это информационное отображение свойства объекта. *Преимущества БД* по сравнению с файловой организацией данных:

1) БД является *информационной моделью объекта*, от обоснованности, точности и достоверности которой во многом зависит эффективность управления объектом;

2) в БД информация *хранится централизованно*. Многие пользователи могут иметь возможность доступа, просмотра и изменения данных в одно и то же время, при этом пользуясь самой последней версией информации. Легче изменять и согласовывать данные. Экономится дисковое пространство;

3) организация данных в базах дает возможность *быстрого поиска, отображения и анализа данных*;

4) БД представляет собой новый подход к организации данных. Возможно *обращение к данным без знания физического расположения их в памяти компьютера*, вследствие чего доступ к данным и их обработка более просты. Разработка прикладных программ, использующих БД, становится проще, быстрее, дешевле.

Современные БД имеют большие объемы, измеряемые в Терабайтах (1 Тб = 1024 Гб) и даже Петабайтах (1 Пб = 1024 Тб). Для их названия появился специальный термин – «сверхбольшие БД».

Приложения БД:

запросы – требования пользователя на отбор данных из базы и (или) на выполнение определенных действий;

формы – чаще всего шаблоны, применяемые для ввода, просмотра и редактирования данных БД;

отчеты – представления информации из БД в виде, удобном для ее восприятия и анализа пользователем;

web-страницы – средства для публикации БД в сети Интернет;

прикладные программы – программы, работающие с БД, и написанные на языке программирования, встроенном в СУБД.

Компоненты БД:

1) *данные пользователей*;

2) *метаданные*. СУБД производит описание структуры БД (метаданные). ***Метаданные*** – данные о данных (хранятся в системных таблицах). По-другому метаданные называют *словарем данных* или *каталогом данных*;

3) *данные, призванные улучшить производительность и доступность БД*. Они состоят главным образом из индексов;

4) *метаданные приложений*. Это описания структуры и формата пользовательских запросов, форм, отчетов и других приложений, выполненные СУБД.

Пользователь БД – лицо или прикладная программа, которые могут обращаться с командами и (или) запросами к БД и получать от нее результаты обращений. Люди, работающие с БД, – это конечные пользователи БД и обслуживающий персонал. ***Конечные пользователи БД*** – это специалисты предметной области, которым требуется информация из БД для выполнения прямых служебных обязанностей (например, бухгалтер, руководитель, менеджер отдела продаж). ***Обслуживающий персонал*** – люди, ответственные за работу БД (*администраторы БД*) и разработку соответствующего прикладного программного обеспечения (*разработчики прикладного ПО*).

Структуры данных. Исследования в области БД прежде были направлены на разработку способов структуризации данных.

Структура данных – программная единица, позволяющая хранить и обрабатывать множество однотипных и (или) логически связанных данных.

Данные бывают двух типов:

основной (простой) – форма представления определяется архитектурой компьютера. Данные простого типа – это символы, числа, т. е. элементы, дробление которых не имеет смысла;

сложный – конструируемый пользователем для решения конкретных задач. Сложные типы данных (структуры данных) формируются из элементарных данных.

Некоторые структуры: ***массив*** – простая совокупность элементов данных одного типа; ***запись*** – совокупность элементов данных разного типа. В простейшем случае записи содержат одинаковое количество элементов, называемых полями. Записи имеют идентификатор (уникальное имя или номер), называемый ***ключом***. Совокупность записей одинаковой структуры называется ***файлом***.

Массив, запись – это *статические* структуры, так как занимают в памяти компьютера постоянный объем.

Динамические структуры – это дерево, список, ссылка. Существует большое количество данных, которые могут быть представлены как ***деревья*** – для их хранения требуется нелинейное адресное пространство.

Список – это упорядоченный набор значений, в котором некоторое значение может встречаться более одного раза.

Ссылка – это объект, указывающий на определенные данные, но не хранящий их (это может быть переменная, содержащая адрес ячейки).

Более сложные структуры могут включать массив, дерево, запись в качестве элементов данных.

Существует большое разнообразие сложных типов данных.

6.2. Модели данных

Модель данных – формальная теория представления и обработки данных в системах управления базами данных (СУБД), которая включает, по меньшей мере, аспекты:

1) *структуры:* методы описания типов и логических структур данных в БД (точка зрения пользователя на представление данных);

2) *манипуляции*: методы манипулирования данными (набор допустимых операций, выполняемых на структуре данных);

3) *целостности*: методы описания и поддержки целостности БД (механизм поддержания соответствия данных предметной области на основе формального описания правил).

Модель данных – это совокупность принципов организации БД. Важнейший из них – это принцип связи объектов в БД.

Классическими моделями данных являются иерархическая, сетевая и реляционная модели данных.

Иерархическая модель. В иерархической модели связи между данными можно представить с помощью дерева (рис. 4.1).

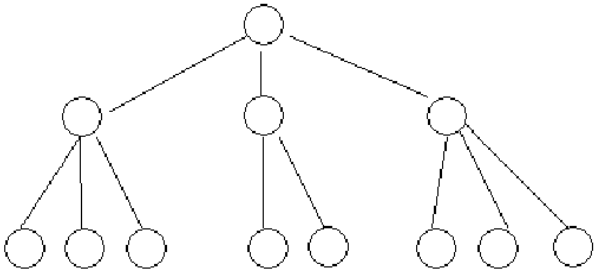

Рис. 4.1. Пример организации данных по иерархической модели

Данные расположены на разных иерархических уровнях и называются ***сегментами***. Самый высокий сегмент – *корневой.* Сегменты на более низком уровне – *сегменты-потомки.* Сегменты на более высоком уровне – *сегменты-предки.*

Каждый сегмент может иметь только одного предка на более высоком уровне и одного или несколько потомков на более низком уровне.

Доступ к определенному сегменту осуществляется по цепочке, от сегмента-предка к сегменту-потомку, начиная слева.

Достоинства иерархической модели: эффективное использование памяти компьютера; неплохие показатели времени выполнения основных операций над данными.

Недостатки: громоздкость для обработки данных со сложными логическими связями; сложность понимания обычным пользователем; медленный доступ к данным нижних уровней иерархии; четкая ориентация на определенные типы запросов.

Сетевая модель. Сетевая модель – развитие иерархической модели. В ней потомок может иметь любое количество предков. Сегменты – наборы записей об объектах – связываются между собой не только по принципу сверху вниз, но и по горизонтали с помощью наборов записей о связях. Любой сегмент может быть связан с любым другим. На рис. 4.2 показан пример организации данных в сетевой модели.

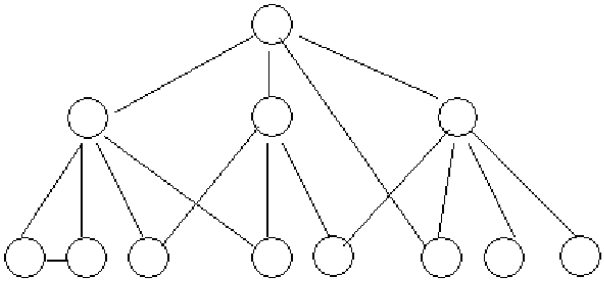

Рис. 4.2. Пример организации данных в сетевой модели

Достоинство сетевой модели: возможность эффективной реализации БД: по показателям затрат памяти; оперативности доступа к данным.

Недостатки: сложность и жесткость схемы БД, построенной на ее основе; сложность понимания и обработки информации в БД обычным пользователем; ослаблен контроль целостности связей вследствие допустимости установления произвольных связей между записями.

Реляционная модель. В основе реляционной модели данных (РМД) лежит понятие отношения. Отношение отображает некоторый объект. Объект характеризуется набором атрибутов x_1, x_2, …, x_n, а каждый атрибут – набором допустимых значений, называемым ***доменом***. Пусть

$$D_1 = \{x_1, x_2, \ldots, x_k\};$$
$$D_2 = \{y_1, y_2, \ldots, y_l\};$$
$$\ldots\ldots\ldots\ldots$$
$$D_n = \{z_1, z_2, \ldots, z_m\}.$$

Список имен атрибутов (x_1, x_2, …, x_n) называется ***схемой отношения***, а количество атрибутов в отношении – ***степенью отношения***.

Отношение – это подмножество R декартова произведения

$$D_1 \times D_2 \times \cdots \times D_n, \text{ т. е. } R \subseteq D_1 \times D_2 \times \cdots \times D_n.$$

Декартово произведение – это набор всевозможных сочетаний из n значений, где каждое значение берется из своего домена.

Пусть атрибут *Номер заказа* имеет домен $D1 = \{1021, 1022, 1023\}$, атрибут *Код клиента* – домен $D2 = \{AA, AC\}$, атрибут *Стоимость заказа*, млн. руб. – домен $D3 = \{100, 300, 120\}$.

Тогда отношение R есть декартово произведение $D_1 \times D_2 \times D_3$ – набор из 18 троек значений. Его можно представить как таблицу вида:

Номер заказа	Код клиента	Стоимость заказа, млн. руб.
1021	AA	100
1021	AA	300
1021	AA	120
1021	AC	100
1021	AC	300
1021	AC	120
1022	AA	100
1022	AA	300
1022	AA	120
1022	AC	100
1022	AC	300
1022	AC	120
1023	AA	100
1023	AA	300
1023	AA	120
1023	AC	100
1023	AC	300
1023	AC	120

Термин «отношение» – синоним слова «таблица».

Столбцы таблицы соответствуют атрибутам. Строки называются ***кортежами***. Количество кортежей в отношении – ***мощность отношения.***

Реляционная модель данных (РМД) – модель данных, основанная на математическом понятии отношения и представлении отношений в форме таблиц.

Реляционная таблица должна обладать *следующими свойствами.*

1. Каждое значение атрибута на пересечении строки и столбца должно быть *атомарным.*
2. Значения в столбце должны быть однородными.
3. Каждая строка уникальна.
4. Каждый столбец имеет уникальное имя.
5. Последовательность столбцов в таблице несущественна.

6. Последовательность строк в таблице несущественна.

Пример реляционной таблицы

КЛИЕНТ

Код клиента	Клиент	Адрес
АА	МАЗ	Минск, пр. Партизанский, 76
АБ	Элема	Минск, ул. Тростенецкая, 5
АС	Коммунарка	Минск, ул. Аранская, 61
АД	МТЗ	Минск, ул. Долгобродская, 6

В реляционной таблице *столбцы* называют ***полями***, а *строки* – ***записями***. Одно или несколько полей, значения которых в каждой записи таблицы однозначно ее идентифицируют, называют ***ключевым полем***.

В таблице КЛИЕНТ таковым может быть поле «Код клиента» или поле «Клиент».

В реляционной БД между таблицами устанавливаются связи. ***Связь*** устанавливается посредством связи ключевых полей, содержащих общую информацию для обеих таблиц.

Одна запись главной таблицы может быть связана с одной или несколькими записями подчиненной таблицы. При этом значения первичного ключа уникальны, а внешнего – могут повторяться.

В общем случае РМД представляет собой ***множество взаимосвязанных таблиц***. Графическое изображение связи между таблицами называется ***схемой данных***.

Пусть имеется еще таблица ЗАКАЗ вида:

ЗАКАЗ

Номер заказ	Код клиента	Дата заказ	Стоимость заказ, млн. руб.
1020	АБ	01.02.15	400
1021	АА	01.02.15	100
1022	АС	12.02.15	300
1023	АА	20.06.15	120
1024	АБ	28.05.15	600

Нужно связать таблицы КЛИЕНТ и ЗАКАЗ. Это можно сделать по полю «Код клиента». В таблице КЛИЕНТ оно играет роль *первич-*

ного ключа, а в таблице ЗАКАЗ – *внешнего ключа*. На рис. 4.3 показана схема данных.

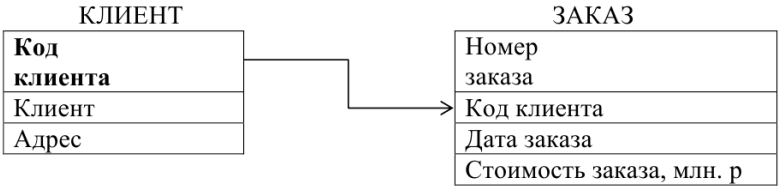

Рис. 4.3. Схема данных

В РМД должны выполняться условия целостности данных:

1) *целостность таблиц* накладывает ограничения на значения первичного ключа – они должны быть *уникальными* и *непустыми*.

2) *ссылочная целостность* предполагает, что *каждое значение внешнего ключа должно совпадать с одним из значений первичного ключа*.

В процессе обработки реляционных таблиц СУБД выполняет операции реляционной алгебры, такие как:

– традиционные операции над множествами: объединение, пересечение, декартово произведение, разность;

– специальные операции: проекция, выборка, соединение, деление.

Достоинства РМД: проста для понимания; наглядна; имеет строгое математическое обоснование.

Недостатки: не допускает представления объектов со сложной структурой, так как в ее рамках возможно моделирование лишь с помощью двумерных таблиц; данные об объектах содержатся, как правило, во многих таблицах. Извлечение информации о каждом объекте требует выполнения многих операций соединения с помощью первичных и внешних ключей, что значительно замедляет обработку данных.

В последнее время активно используются при разработке БД такие модели, как *постреляционная, объектно-ориентированная, объектно-реляционная* и *многомерная модели*.

Постреляционная модель – в общем случае представляет собой расширенную реляционную модель, снимающую ограничение неделимости значений полей, т. е. допускаются многозначные поля, значения которых состоят из подзначений. В случае постреляцион-

ной модели данные о заказах клиентов из двух связанных реляционных таблиц КЛИЕНТ и ЗАКАЗ можно представить одной таблицей:

Код клиента	Номер заказа	Клиент	Адрес	Дата заказ	Вес заказа
АА	1021	МАЗ	Минск, пр. Партизанский, 76	01.02.15	100
	1023			20.06.15	120
АБ	1020	Элема	Минск, ул. Тростенецкая, 5	01.02.15	400
	1024			28.05.15	600
АС	1022	Коммунарка	Минск, ул. Аранская, 61	12.02.15	300

Достоинства постреляционной модели данных: возможность представления связанных реляционных таблиц одной постреляционной таблицей, что обеспечивает высокую наглядность представления данных и повышение эффективности их обработки; отсутствие ограничений на длину полей и их количество в записях таблицы. *Недостаток* – сложность в обеспечении целостности данных.

Объектно-ориентированная и объектно-реляционная модели – используются для преодоления ограниченных возможностей РМД по хранению и обработке сложных объектов, как, например, документ, звук, видео, графическое изображение и др.

Объектно-ориентированная модель (ООМД) – представляет структуру, которую можно изобразить графически в виде дерева, узлами которого являются объекты. Каждый объект характеризуется уникальным *идентификатором, состоянием и поведением*.

Состояние объекта – определяется множеством значений его атрибутов.

Поведение объекта описывают *методы*, называемые ***процедурами***, т. е. составной частью описания объекта являются процедуры, способные производить действия над атрибутами объекта в случае наступления тех или иных событий.

Объекты могут объединяться в ***классы***. Экземпляры одного класса отличаются лишь значениями своих свойств. Методы устанавливаются для класса.

Для выполнения действий над объектами применяются объектно-ориентированные механизмы – *наследование, инкапсуляция, полиморфизм*.

Наследование – на основе существующего класса можно образовать новый класс объектов, наследующий свойства родительского класса.

Инкапсуляция – доступ к данным осуществляется только лишь в соответствии с правилами поведения объекта, описываемыми методами.

Полиморфизм – способность объектов по-разному реагировать на одно и то же событие в окружающем мире. Полиморфизм используется для унификации обработки разнородных объектов. Например, метод «Печать результата» может быть определен для многих классов объектов, но работать по-разному, в зависимости от класса, к которому он применяется.

Достоинство ООМД – способность отображать информацию о сложных объектах с исчерпывающим описанием взаимосвязей между ними и их динамического поведения. ООМД обычно применяется для сложных предметных областей, при моделировании которых не хватает функциональности РМД.

Недостаток – сложность понятийного аппарата, что усложняет ее применение и отрицательно сказывается на накоплении опыта создания и эксплуатации объектно-ориентированных БД.

Объектно-реляционная модель данных (ОРМД) – является гибридной моделью, сочетающей возможности реляционной модели с объектными свойствами данных.

Существует два подхода к ее созданию.

1. Объекты, видимые на внешнем интерфейсе, отображаются в таблицы реляционной БД. И наоборот, объекты воспроизводятся из их представления в табличной среде хранения, когда они запрашиваются пользователями или приложениями (*гибридный подход*).

2. Внутренние реляционные механизмы СУБД для управления данными расширяются объектно-ориентированными возможностями (*расширенный реляционный подход*). Этот подход технологически более продвинутый и предпочитаемый в настоящее время большинством разработчиков реляционных СУБД.

Отличительная особенность ОРМД от ООМД состоит в том, что она основана на стратегии реляционной модели.

Многомерная модель – означает *многомерное логическое представление структуры информации*, а не многомерность визуализации данных. Многомерная модель предназначена *для аналитической* обработки информации, в отличие от предыдущих моделей. В ней используются такие понятия, как агрегируемость, историчность, прогнозируемость данных.

Агрегируемость данных – возможность их рассмотрения с различным уровнем обобщения.

Историчность – высокий уровень статичности (неизменяемости) данных и их взаимосвязей, а также обязательная привязка данных к временным точкам.

Прогнозируемость данных – применение функций их прогнозирования к различным интервалам времени.

Представление данных о продажах автомобилей:

по реляционной модели:

Марка автомобиля	Месяц	Объем продаж, шт.
Ауди	Январь	12
Ауди	Февраль	24
Ауди	Март	5
Пежо	Январь	2
Пежо	Февраль	18
BMW	Февраль	19

многомерной модели:

Марка автомобиля	Январь	Февраль	Март
Ауди	12	24	5
Пежо	2	18	0
BMW	0	19	0

Основные понятия в многомерной модели – *измерение и ячейка.*

Измерение – это множество однотипных данных, образующих одну из граней многомерного гиперкуба. Наиболее часто используются *временные* измерения – дни, месяцы, кварталы и годы. В качестве *географических* измерений широко употребляются города, районы, регионы и страны.

Ячейка – это поле, значение которого однозначно определяется фиксированным набором измерений. В вышеприведенном примере значение ячейки объема продаж однозначно определяется комбинацией временного измерения Месяц и измерения Марка автомобиля.

Многомерную модель, отображающую объемы продаж автомобилей менеджерами по годам, можно представить уже в виде трехмерного куба (рис. 4.4).

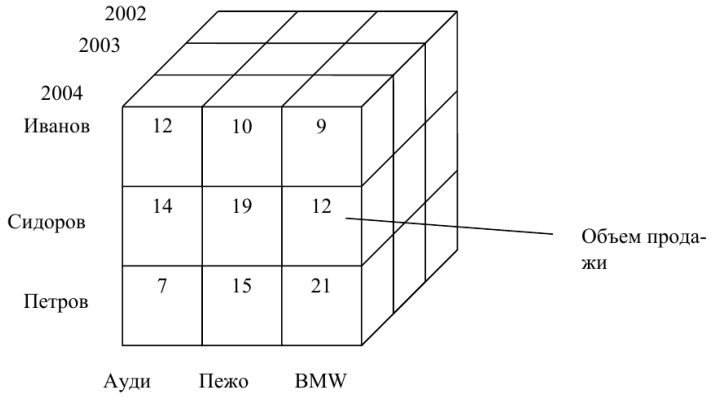

Рис. 4.4. Трехмерная модель данных о продажах автомобилей

В многомерной модели данных используется два варианта организации данных – гиперкубическая и поликубическая. В ***гиперкубической*** все кубы определяются одним и тем же набором измерений (максимально возможным). В ***поликубической*** – несколько гиперкубов различной размерности и различными измерениями в качестве граней.

Для извлечения данных из БД, организованной по многомерной модели, применяются специальные операции: срез, вращение, агрегация и детализация.

Срез – данные, полученные в результате фиксации одного или нескольких измерений.

Вращение – процедура изменения порядка следования измерений. Применяется в основном при двухмерном представлении данных.

Агрегация и ***детализация*** – соответственно переход к более или менее детальному представлению информации из гиперкуба.

Достоинство многомерной модели – удобство и эффективность *аналитической* обработки больших объемов данных, связанных с временными интервалами.

Недостаток – громоздкость для простейших задач оперативной обработки информации.

6.3. Системы управления базами данных (СУБД). Обзор современных СУБД и направления их развития

СУБД существуют практически для всех аппаратных платформ (от карманных компьютеров до суперкомпьютеров) и используются во всех автоматизированных информационных системах как средство работы с БД.

Система управления базами данными (СУБД) – это совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

Языковые средства: язык описания данных (ЯОД), язык манипулирования данными (ЯМД), язык запросов, язык программирования. ЯОД – средства для описания структуры данных, их типов, свойств и ограничений для данных. ЯМД позволяет вставлять, обновлять, удалять и извлекать данные.

Современные языки запросов позволяют выполнять все операции над данными БД. Пример – язык SQL.

Некоторые СУБД располагают языками, которые используются и для обращения к данным БД, и для написания прикладных программ. Например, xBase. Для глубокой аналитической обработки данных могут создаваться программы на традиционных языках программирования, для которых конкретная СУБД обеспечивает доступ к БД.

Программные средства СУБД обеспечивают работу с физической БД (поиск, изменение, обработка, извлечение данных и др.) и выполнение всех ее функций.

Функциональные возможности СУБД.

Современные СУБД предоставляют пользователям следующие возможности:

– создание БД;

– обновление хранящихся в ней данных;

– быстрое извлечение из БД необходимых данных по запросам пользователей; выполнение вычислений над данными;

– создание форм для удобства работы с данными БД;

– создание отчетов на основе информации БД для ее анализа пользователями; разработка приложений;

– экспорт/импорт данных в БД из других приложений;

– публикация БД в Интернете;

– управление БД;

– защита данных от несанкционированного доступа, от сбоев в работе компьютерной системы;

– восстановление БД в случае ее повреждения.

Архитектура СУБД – это совокупность функциональных компонентов системы и их взаимосвязей. СУБД включает 3 компонента:

подсистему средств проектирования БД и ее приложений;

подсистему обработки компонентов приложений;

ядро СУБД, которое является посредником между данными подсистемами и БД, участвует в управлении транзакциями, блокировке, резервном копировании и восстановлении. ***Транзакция*** – совокупность операций с БД, которые должны быть выполнены обязательно до конца, чтобы БД осталась в непротиворечивом состоянии.

Классификация СУБД.

По степени универсальности различают СУБД:

– общего назначения;

– специального назначения.

***По типу поддерживаемой модели данных*:**

– *иерархические*. Типичным представителем является Information Management System (IMS) фирмы IBM;

– *сетевые*. Типичным представителем является Integrated Database Management System (IDMS) компании General Electric;

– *реляционные*. Первые коммерческие реляционные СУБД – от компаний IBM, Oracle Corporation и др.;

– *постреляционные* – uniVers, Bubba, Dasdb и др.;

– *объектно-ориентированные* – ORION, IRIS, Vbase, PDM и др.;

– *объектно-реляционные* – Informix Universal Server (Informix Software), DB2 Universal Database (IBM) и др.;

– *многомерные* – Oracle Express Server (Oracle), Cache (InterSystems) и др.

***По принципу обработки запросов к БД*:**

– настольные;

– серверные;

– распределенные.

Режимы работы пользователя в СУБД. Пользователи могут работать с БД в СУБД:

– через *меню системы*;

– в *командном режиме*;

– *программном режиме*.

Работа с локальной БД осуществляется в *однопользовательском* режиме, с централизованной БД – в *многопользовательском.*

СУБД является программной основой автоматизированной информационной системы. При ее выборе учитывают следующие факторы:

– скорость выполнения операций корректировки данных;

– время выполнения запросов (время отклика);

– время генерации отчета;

– максимальное число одновременных обращений к БД в многопользовательском режиме и др.

В 1988 г. была создана международная организация TPC (Transaction Processing Performance Council), которая разработала ряд *эталонных тестов* для сравнительной оценки производительности коммерческих СУБД.

Функции СУБД. СУБД, работая с физической БД (с БД на устройстве внешней памяти), выполняют:

– управление:

• *данными во внешней памяти*. Эта функция дает возможность пользователям выполнять основные операции с данными – сохранение, извлечение, обновление;

• *транзакциями*. **Транзакция** – логическая единица работы, включающая несколько команд вставки, удаления или модификации, которая переводит БД из одного завершенного состояния в другое завершенное состояние, которое не нарушает целостности этих данных (все данные в таблицах БД правильны, а ссылки между таблицами корректны). В зависимости от того, какие команды используются, транзакции есть следующих *типов*: только для записи; только для модификации; только для чтения; только для удаления. Транзакции только для чтения называют *запросом*;

• *параллельным доступом*, чтобы избежать конфликта с нежелательными последствиями при попытке обновления данных;

• *буферами оперативной памяти*;

– поддержку:

• *языков баз данных*;

• *обмена данными* (экспорт/импорт);

• *целостности данных*;

• *независимости прикладных программ от данных.*

– ведение системного каталога;

– контроль доступа к данным.

Должен быть только санкционированный доступ.

Направления развития СУБД. В настоящее время можно говорить о следующих направлениях развития СУБД:

– расширение множества типов обрабатываемых данных;

– комбинирование технологий WWW и технологий БД;

– переход к активным БД.

Настольные СУБД. Архитектура файл-сервер.

Настольные СУБД использовались для работы с локальными БД на ПК, а с появлением компьютерных сетей – для работы с централизованными БД в архитектуре файл-сервер.

Архитектурой файл-сервер называется архитектура с совместным использованием файлов. В ней компьютеры объединены в сеть. На *файловом сервере* сети размещается БД, а на *рабочих станциях* устанавливаются настольные СУБД. СУБД на рабочей станции запрос пользователя или прикладной программы посылает к БД на сервере. По запросу из БД выбираются необходимые для его выполнения *таблицы целиком* и пересылаются на рабочую станцию, где СУБД выполняет запрос.

Недостатки архитектуры файл-сервер:

– по сети передается гораздо больший объем данных, чем реально нужно для выполнения запроса, и сеть сильно перегружается;

– выполнение запросов и управление целостностью БД ложится на СУБД пользователей, что является причиной невысокой безопасности работы. Секретность и конфиденциальность информации обеспечить также трудно; пользователи могут формировать запросы и на внесение изменений в БД. При этом блокируются записи, которые изменяются одним из пользователей, чтобы в это время другой пользователь не внес изменений в те же данные. Из-за этого настольные СУБД редко используются для обработки БД больших объемов, ориентированных на транзакции.

Характеристика настольных СУБД.

Популярные настольные СУБД: dBase, Paradox, MS FoxPro, MS Access. СУБД семейства **dBase** благодаря простоте в использовании, нетребовательности к ресурсам компьютера, комфортной среде разработки приложений БД, популярному формату данных и популярному языку программирования xBase имели немалую популярность. Последние версии dBase имеют средства, присущие современным СУБД. Тенденция такова, что dBase превращается в некоммерческий продукт с доступными исходными текстами программ.

Ранние версии **Paradox** предоставляли более широкие возможности, чем аналогичные версии dBase, – средства статистического анализа данных, создание приложений на языке PAL с возможностью визуального построения пользовательских интерфейсов. Windows-версии этой СУБД использовались как универсальное средство управления различными БД. Последние версии являются компонентом Corel Office Professional и содержит средства, свойственные современным СУБД. Популярность Paradox в настоящее время снизилась, хотя в мире эксплуатируется еще много информационных систем на ее основе.

MS FoxPro происходит от настольной СУБД FoxBase (Fox Software), которая впоследствии была приобретена Microsoft. Последние версии имеют средства визуального моделирования объектов, средства публикации данных в Интернете и др. С каждой новой версией этот продукт все более интегрируется с другими продуктами Microsoft. Тенденция развития этой СУБД – из настольной СУБД превращается в средство разработки приложений в архитектуре клиент/сервер и распределенных приложений.

MS Access была первой настольной реляционной СУБД для 16-разрядной версии Windows. Ее популярность значительно возросла после включения в состав MS Office. Эта СУБД ориентирована на непрофессиональных пользователей MS Office. В ней вся информация, относящаяся к конкретной БД, хранится в одном файле, что удобно для начинающих пользователей. Последние версии могут быть использованы, с одной стороны, в качестве настольной СУБД, а с другой – в качестве клиента MS SQL Server.

Достоинства настольных СУБД: просты для освоения и использования; имеют дружественный пользовательский интерфейс; ориентированы на класс самых распространенных компьютеров – персональных и на самую широкую категорию пользователей – непрофессионалов; обеспечивают хорошее быстродействие при работе с небольшими БД.

Недостатки: с увеличением объемов БД и числа их пользователей снижается производительность СУБД, и они дают сбои при обработке данных.

Серверные СУБД. Архитектура клиент-сервер.

Серверные СУБД используются в архитектуре клиент/сервер. Базовый принцип этой архитектуры – централизация хранения и обработки данных. На *сервере сети* размещается база данных и

устанавливается мощная серверная СУБД или ***сервер БД*** – программный компонент, обеспечивающий хранение больших объемов информации, ее обработку и представление пользователям в сетевом режиме. На *клиентских компьютерах* устанавливаются клиентские приложения.

Выполнение запроса к БД в архитектуре клиент-сервер представлено на рис. 4.5.

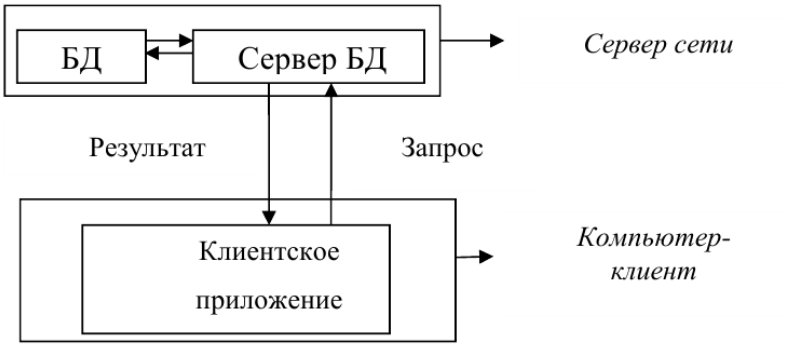

Рис. 4.5. Выполнение запроса к БД в архитектуре клиент-сервер

Функции сервера БД:

– интерпретирует, выполняет запрос, формирует его результат и пересылает по сети на клиентский компьютер;

– управляет целостностью БД;

– обеспечивает одновременную безопасную, отказоустойчивую многопользовательскую работу с одними и теми же данными;

– выполняет хранение и резервное копирование данных;

– реализует бизнес-правила (установки, принятые в предметной области).

Преимущества архитектуры клиент-сервер:

– уменьшается сетевой трафик;

– уменьшается потребность клиентских приложений в оперативной памяти;

– существенно повышается степень безопасности БД.

Современные серверы БД. *Характеристики современных серверов БД*:

– многоплатформенные;

– наличие нескольких версий для решения различных по масштабу задач; высокие показатели надежности и доступности;

– параллельная обработка данных в многопроцессорных системах; распределенные запросы и транзакции;

– стандартные механизмы доступа к данным (ODBC, JDBC, OLE DB, ADO.NET);

– средства разработки клиентских приложений;

– простота администрирования;

– несколько сценариев репликаций;

– создание хранилищ данных и OLAP.

***Популярные серверные СУБД*:**

ORACLE:

– имеет *большое количество различных версий и типов*;

– поддерживает *свыше 80 вариантов операционной среды;*

– функционирует *на большинстве компьютерных платформ* (мэйнфреймы, Unix-серверы, ПК и др.);

– поддерживает *все варианты архитектур компьютеров,* в том числе многопроцесорные системы;

– *хранит и обрабатывает различные типы данных* (текстовых документов, графических образов более 20 форматов, видео- и аудио-информации);

– имеет язык запросов – *PL-SQL*;

– имеет *множество различных компонентов и модулей.*

MS SQL Server:

– реляционная СУБД;

– имеет язык запросов – *Tranzact-SQL*;

– используется *для работы с БД от персональных до крупных масштабов предприятия*.

СУБД My SQL:

– свободно *распространяемая реляционная СУБД с открытой архитектурой*. Поддержку осуществляет корпорация Oracle;

– обычно используется *в качестве сервера*, к которому обращаются локальные и удаленные клиенты;

– поддерживает *большое количество таблиц*, различающихся системой хранения данных;

– благодаря открытой архитектуре *постоянно появляются новые типы таблиц.*

Распределенные БД и СУБД.

Распределенная база данных (РаБД) – совокупность логически взаимосвязанных БД, распределенных в компьютерной сети. В РаБД данные распределены по узлам компьютерной сети. Каждый узел имеет собственную БД и может обращаться к данным, хранящимся на других узлах. Пользователь распределенной БД не обязан знать, каким образом ее компоненты размещены в узлах сети и представляет себе эту БД как единое целое.

РаБД могут быть:

– *однородными* (гомогенными), имеющими в основе одну СУБД, обычно с единственным языком БД;

– *неоднородными* (гетерогенные), имеющими в основе две или более существенно различающихся СУБД.

6.4. Функциональные возможности СУБД на примере MS Access

Microsoft Access 2007 – профессиональная программа управления базами данных. С ее помощью можно накапливать и систематизировать разнообразную информацию, искать и сортировать объекты согласно выбранным критериям, конструировать удобные формы для ввода данных и генерировать на основании имеющихся записей прекрасно оформленные отчеты.

Типы данных. Для того чтобы обеспечить возможность хранения в базе данных разнообразной информации, Access предлагает большой набор типов данных, представленных в табл. 4.1.

Таблица 4.1. **Типы данных MS Access**

Название типа	Назначение
Текстовый	Текст длиной до 255 символов
Поле MEMO	Текст длиной до 65 000 символов
Числовой	Числа различных форматов
Дата/время	Дата и (или) время
Денежный	Денежные значения различных форматов
Счетчик	Счетчик, который автоматически увеличивается на единицу с добавлением каждой новой записи
Логический	Величины, способные принимать только два значения: да/нет
Поле объекта OLE	Поля, позволяющие вставлять рисунки, звуки и данные других типов
Гиперссылка	Ссылки, дающие возможность открывать объект Access (таблицу, форму, запрос и т. д.), файл другого приложения или web-страницу

Основные понятия СУБД Microsoft Access. База данных под управлением СУБД Access – это файл с расширением *.accdb*, содержащий следующие объекты: таблицы для хранения данных, формы для ввода и редактирования данных в интерактивном режиме, запросы для обработки таблиц и других запросов, отчеты для вывода результатов обработки данных, макросы и модули для автоматизации выполнения рутинных действий.

Интерфейс пользователя. Окно Microsoft Access (рис. 4.6) содержит элементы, стандартные для приложений Microsoft Office 2007: строку заголовка, ленту, рабочую область, строку состояния.

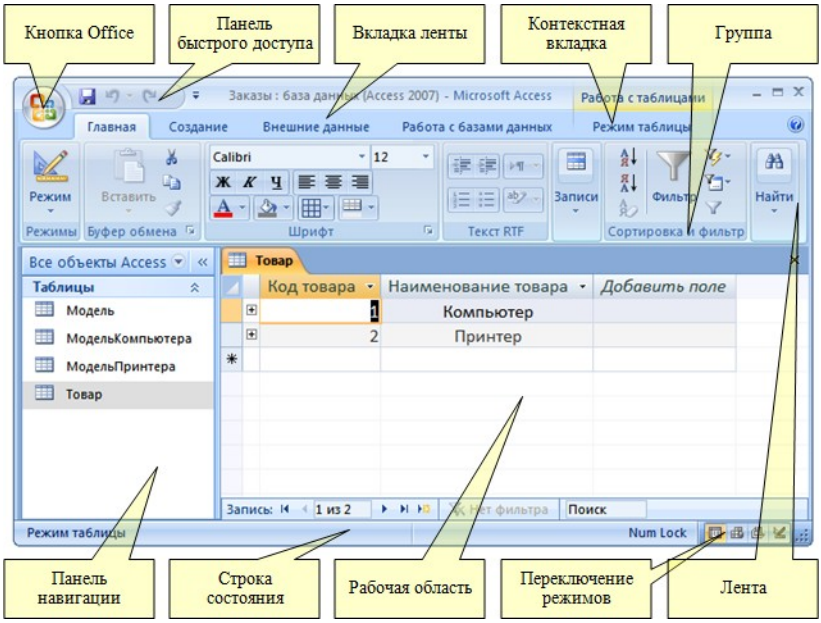

Рис. 4.6. Интерфейс Microsoft Access

В строке заголовка отображается имя открытой базы данных (дополнительно показываются ее формат и режим), а также кнопки управления окном программы. Строка заголовка расположена в верхней части окна и немного смещена вправо. В левой части расположены панель быстрого доступа и кнопка «Office».

Кнопка «Office» – открывает меню для выполнения основных операций с файлом базы данных, а также список последних документов (баз данных), с которыми работал пользователь.

Панель быстрого доступа – содержит кнопки для выполнения часто используемых команд. Эта панель настраиваемая. С помощью кнопки со стрелкой, направленной вниз, открывается ее меню, состоящее из команд, которые можно в нее добавить или убрать из нее.

Лента – элемент интерфейса, содержащий все элементы управления, с помощью которых выполняются команды и задания в Access 2007. Элементы расположены на тематических вкладках, вкладки бывают основные, или постоянные, и дополнительные, или контекстные.

1. Основные вкладки – **Главная**, **Создание**, **Внешние данные**, **Работа с базами данных** – содержат постоянно доступные команды и инструменты.

2. Контекстные вкладки – **Конструктор**, **Режим таблицы**, **Формат**, **Упорядочить** и др., появляются на ленте по мере необходимости. Количество отображаемых контекстных вкладок и их названия связаны с текущим объектом базы данных, его режимом или задачами, которые можно выполнить.

Панель сообщения – отображает важные сообщения системы, например, предупреждения системы безопасности.

Панель навигации (Область переходов) – отображает настраиваемые списки всех объектов базы данных, с ее помощью можно открывать объекты и выполнять определенные действия. Панель навигации прикреплена к левому краю окна Access. При необходимости ее можно свернуть, освобождая дополнительное место для объектов базы данных.

Рабочая область – в ней располагаются открытые объекты базы данных, с которыми может работать пользователь. Несколько открытых объектов располагаются на вкладках, которые «наложены» друг на друга. Для перехода между вкладками (объектами) используют ярлыки, воспроизводящие названия объектов.

Строка состояния – в ней отображается текущее состояние программы, а также содержатся кнопки для быстрого переключения режимов отображения активного объекта базы данных.

Контекстная панель инструментов – содержит кнопки для выполнения команд форматирования выделенного фрагмента текста.

Режимы работы. Приложение поддерживает три режима работы:

1. Режим запуска, в котором можно выполнять сжатие, преобразование, шифрование/дешифрование и другие операции без открытия базы данных.

2. Режим конструктора, в котором создаются и модифицируются объекты базы данных.

3. Режим выполнения, в котором отображаются окна объектов базы данных. При работе с таблицей этот режим называется – режимом таблицы, с формой – режимом формы, с отчетом – режимом предварительного просмотра.

Создание базы данных. При наличии проекта физической модели базы данных можно приступить к ее созданию в Microsoft Access. Сначала создается файл базы данных, затем – все входящие в нее объекты и, в первую очередь, таблицы. Все остальные объекты базы данных будут основаны на таблицах и установленных между ними связях.

СУБД Access позволяет создавать БД, используя различные средства: новый файл, новый файл на основе шаблона, конвертирование внешней базы данных (текстового формата, формата электронной таблицы, реляционной базы данных и т. п.).

1. *Создание новой базы данных.*

1.1. Запускаем Microsoft Access, выполнив команду меню Пуск > Программы > Microsoft Office > Microsoft Office Access 2007.

1.2. В открывшемся окне (рис. 4.7) нажимаем на значке **Новая База данных**.

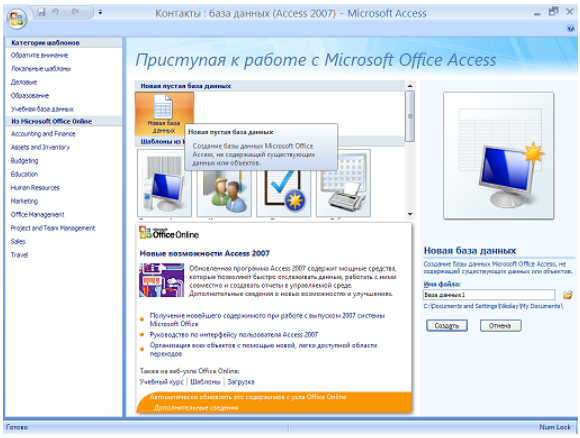

Рис. 4.7. Окно Microsoft Access

1.3. В поле **Имя файла** пишем название создаваемой базы, например «Библиотека».

1.4. По умолчанию база данных создается в папке **Мои документы** (My Documents), для изменения пути необходимо кликнуть на иконке  слева от поля **Имя файла**.

1.5. Нажимаем кнопку **Создать**.

2. *Создание базы данных с помощью шаблонов.*

2.1. Запускаем Microsoft Access, выполнив команду меню Пуск → Программы → Microsoft Office →Microsoft Office Access 2007.

2.2. В открывшемся окне (рис. 4.8), в области **Локальные шаблоны** выбираем нужный шаблон.

2.3. Нажимаем кнопку **Создать**.

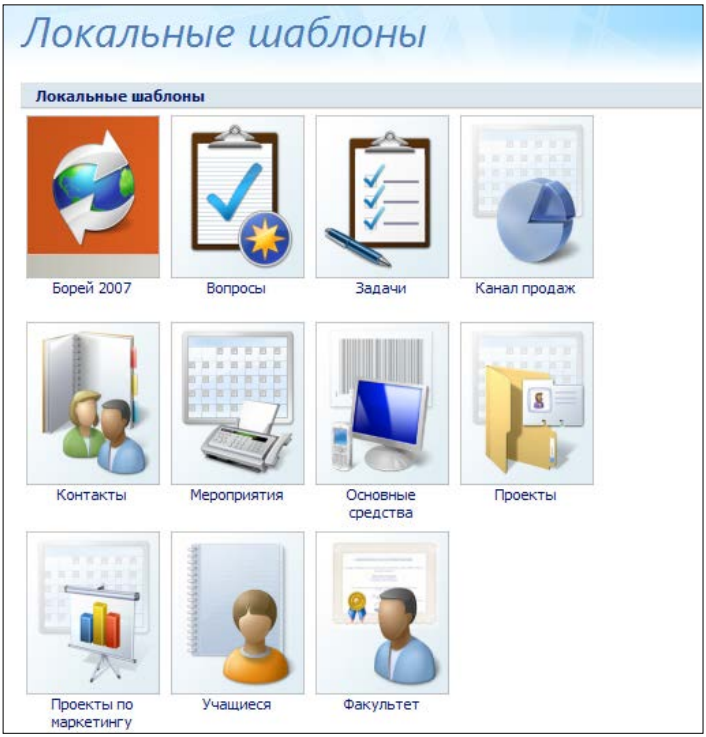

Рис. 4.8. Локальные шаблоны

Таблицы. После создания новой базы открывается окно **База данных** (рис. 4.9).

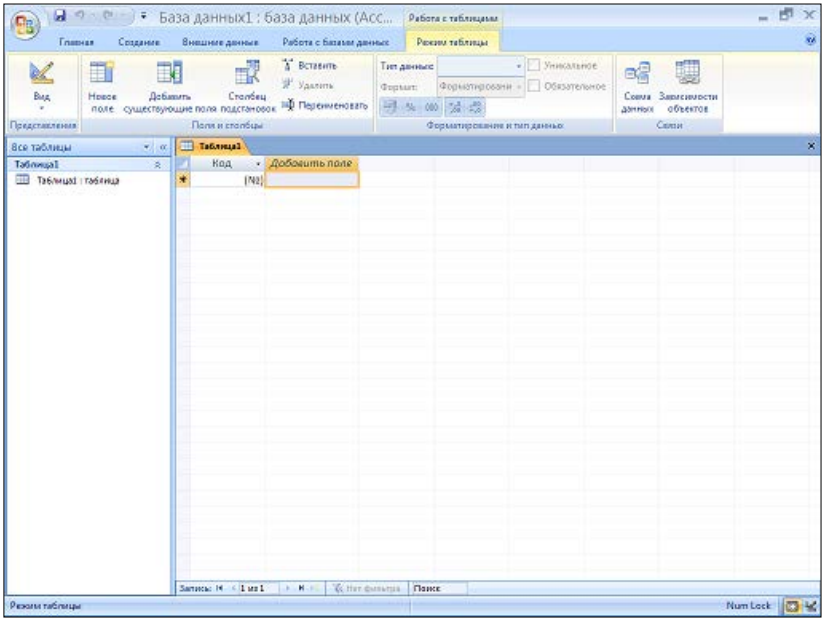

Рис. 4.9. Окно **Базы данных**

3. *Создание таблицы с помощью шаблонов таблиц.*

3.1. Переходим на закладку **Создание** (рис. 4.10).

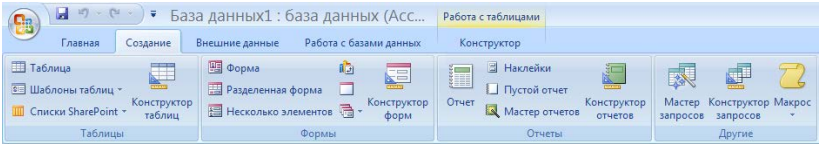

Рис. 4.10. Верхнее меню

3.2. Нажимаем **Шаблоны таблиц**, выбираем **Контакты**.

3.3. В результате получили таблицу с готовыми именами столбцов, при необходимости можно добавить свои столбцы (двойной клик ЛКМ на последнем столбце **Добавить поле** и ввести имя нового столбца)

или удалить лишние столбцы (клик ПКМ на лишнем столбце, пункт контекстного меню – **Удалить столбец**) (рис. 4.11).

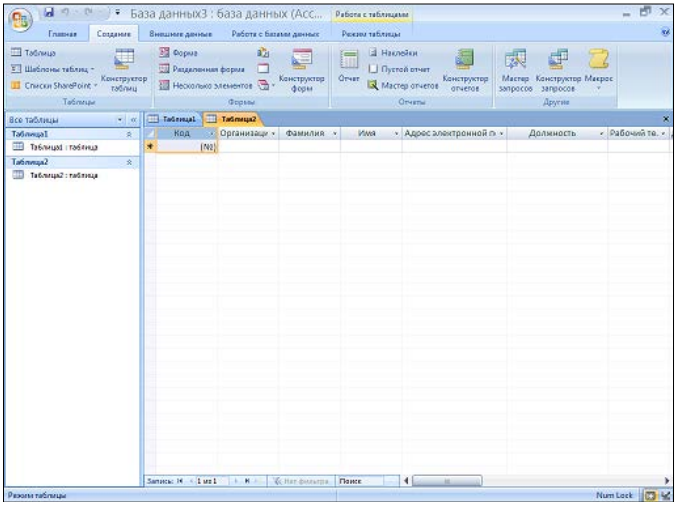

Рис. 4.11. Окно таблицы

3.4. Кликнем в ячейку под названием **Организация**, введем название организации – **Школа**, тогда значению **Код** автоматически присваивается значение.

3.5. Заполним остальные ячейки таблицы (рис. 4.12).

3.6. Сохраняем таблицу: ПКМ на закладке **Таблица 2** вводим имя **Контакты**, нажимаем **ОК**.

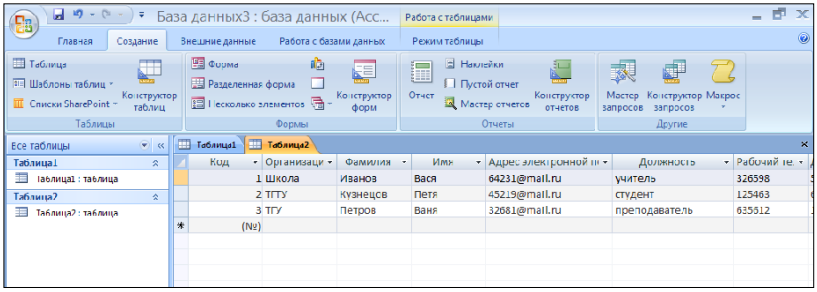

Рис. 4.12. Таблица данных

4. *Создание таблицы с помощью конструктора таблиц.*

4.1. Кликнем на закладку **Таблица 1**.

4.2. Переходим в режим конструктора (рис. 4.13).

4.3. Предлагается сохранить таблицу, вводим имя **Книги** и нажимаем **ОК** (рис. 4.14).

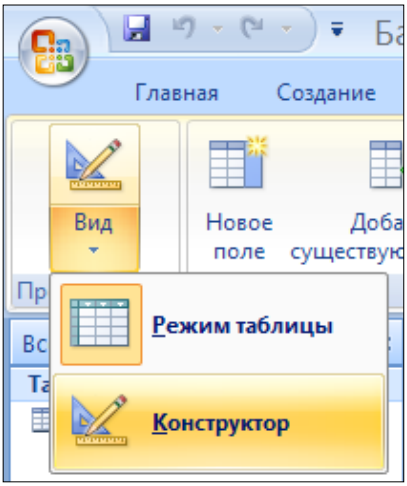

Рис. 4.13. Выбор режима отображения

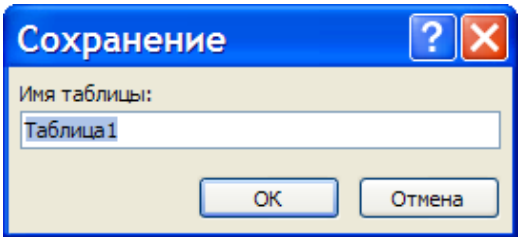

Рис. 4.14. Ввод имени таблицы

4.4. В открывшемся окне конструктора таблиц можно вводить имена полей новой таблицы или добавлять поля в уже созданную таблицу (рис. 4.15).

4.5. Кликнем ниже ячейки **Код**, введем имя столбца **Название**, тип данных – текстовый. Заполняем еще несколько ячеек (рис. 4.16).

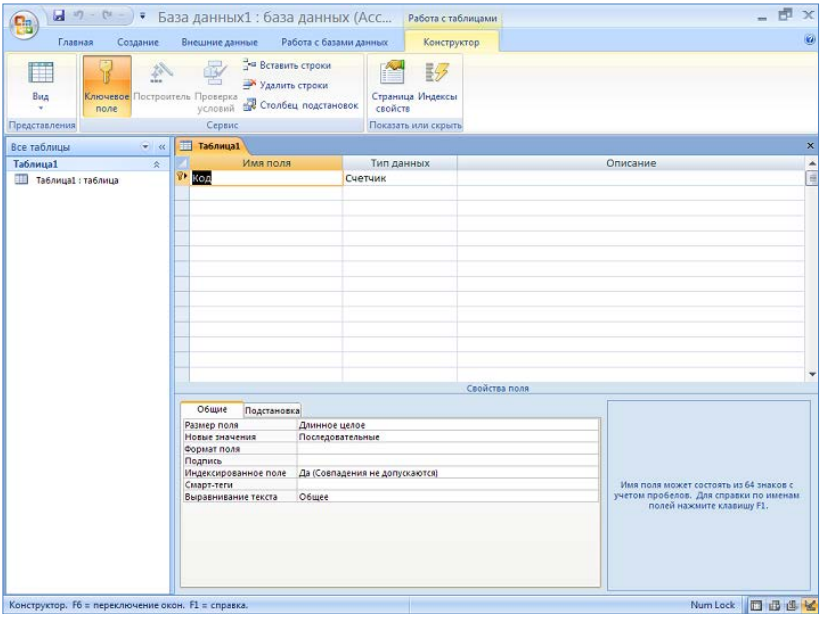

Рис. 4.15. Окно конструктора таблиц

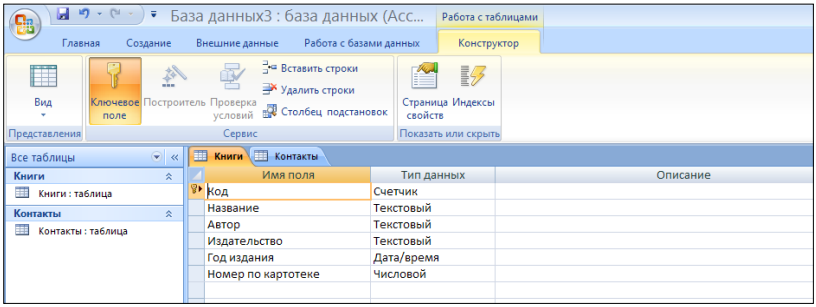

Рис. 4.16. Заполнение полей таблицы

4.6. Переходим в режим таблицы: **Вид → Режим таблицы**.
4.7. Заполняем таблицу аналогично п. 4.4. и 4.5 (рис. 4.17).

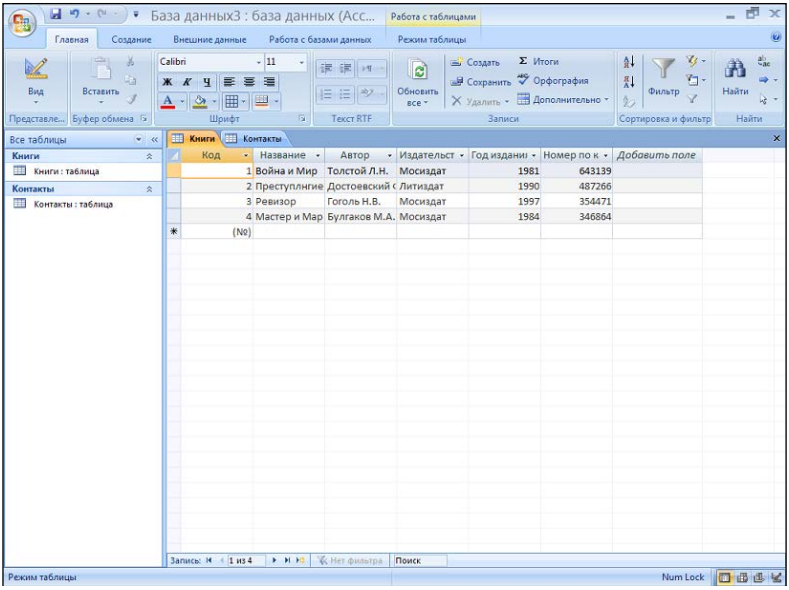

Рис. 4.17. Таблица данных

Связь таблиц. Связь позволяет установить правила взаимодействия между таблицами. Различают два типа связей: один ко многим и многие ко многим. В нашем случае при создании базы данных библиотеки подходит связь – многие ко многим, т. е. у одного человека может быть несколько книг и экземпляры одной книги могут быть у разных людей.

При создании связи **один ко многим** ключ первой таблицы прописывается отдельным столбцом во второй таблице.

При создании связи **многие ко многим** ключи обеих таблиц прописывается в третьей таблице (вспомогательной).

Рассмотрим на примере.

5. *Создание связи.*

5.1. Переходим на закладку **Создание** и создаем новую таблицу.

5.2. Затем переходим в режим конструктора, при этом сохраняем таблицу под именем **Записи**.

5.3. Вписываем названия столбцов по примеру (рис. 4.18).

5.4. Переходим в режим таблицы и заполняем ее по примеру (рис. 4.19).

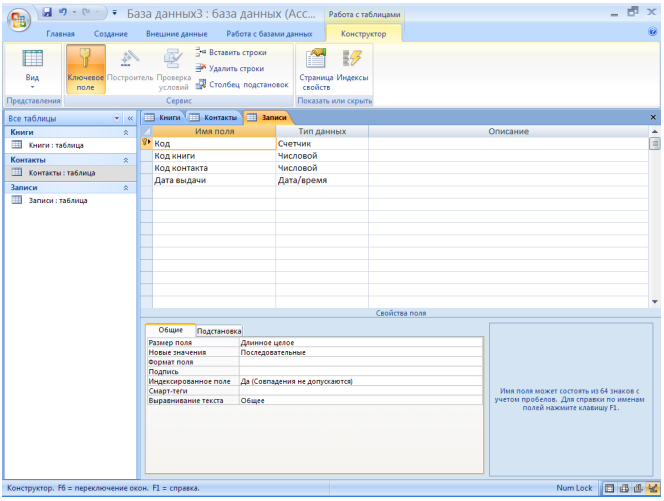

Рис. 4.18. Заполнение полей таблицы

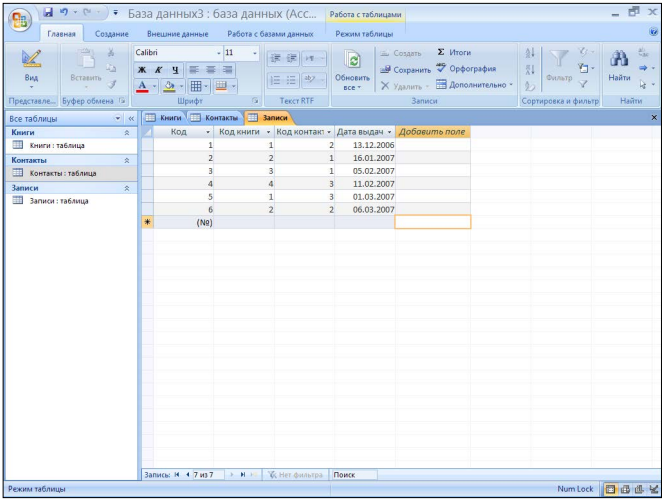

Рис. 4.19. Таблица данных

5.5. Переходим на закладку **Работа с базами данных** и нажимам на кнопку **Схема данных**.

5.6. В открывшемся окне последовательно добавляем все три таблицы, в результате получим в окне **Схема данных** 3 таблицы (рис. 4.20).

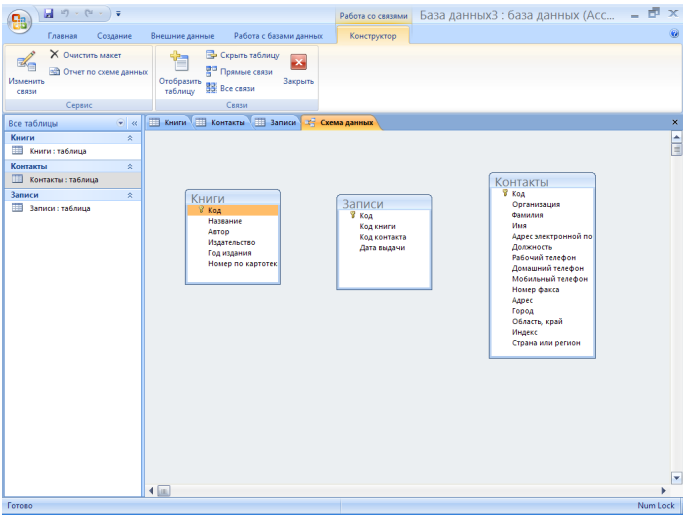

Рис. 4.20. Окно схемы данных

5.7. Создаем связь – помещаем указатель на пункт **Код** из таблицы **Книги**, нажимаем **ЛКМ** и, не отпуская ее, перетаскиваем в пункт **Код книги** из таблицы **Записи**.

5.8. В появившемся окне ставим флажок *Обеспечение целостности данных*. Этот режим не позволит Access оставлять в таблице **Записи** записи, для которых нельзя подобрать запись таблицы **Книги** с подходящим значением поля **Код**.

Установка флажка *Каскадное удаление связанных записей* приведет к тому, что при удалении записи таблицы **Книги** будут удалены все соответствующие записи таблицы **Записи**. Если указанный флажок сброшен, удаление тех записей таблицы **Книги**, на которые ссылается хотя бы одна запись таблицы **Записи**, запрещено.

Установка флажка *Каскадное обновление связанных* полей приведет к тому, что при обновлении поля **Код** таблицы **Книги** будут автоматически обновляться одноименные поля в соответствующих записях таблицы **Записи** (рис. 4.21).

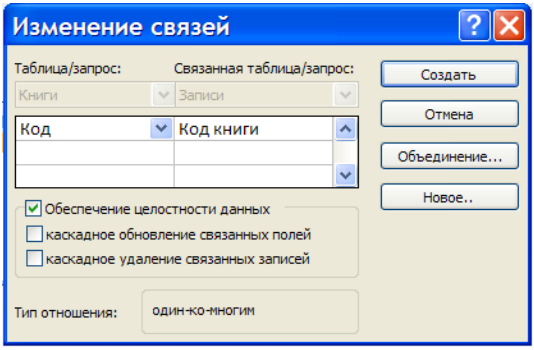

Рис. 4.21. Параметры связи

5.9. Нажимаем кнопку **Создать**.

5.10. Повторяем пункты 5.7, 5.8, 5.9 для таблицы **Контакты** (рис. 4.22).

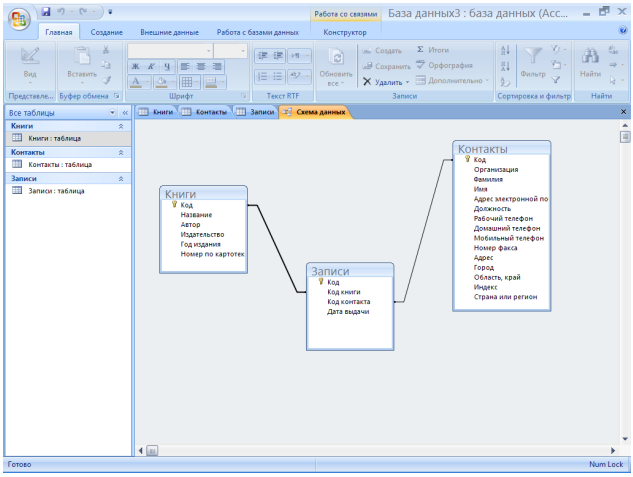

Рис. 4.22. Схема данных

6. *Список подстановки.*

При заполнении таблицы **Записи** новыми данными использование для этого **Кода** очень неудобно, для облегчения заполнения таблицы в Access есть список подстановки.

6.1. Откроем таблицу **Записи** в режиме конструктора.

6.2. Для имени поля **Код книги** в поле **Тип данных** выбираем пункт – **Мастер подстановок**.

6.3. В отрывшемся окне **Создание подстановки** оставляем переключатель, как показано на рис. 4.23 и нажимаем **Далее >**.

6.4. Выбираем таблицу **Книги** (рис. 4.24) и нажимаем **Далее >**.

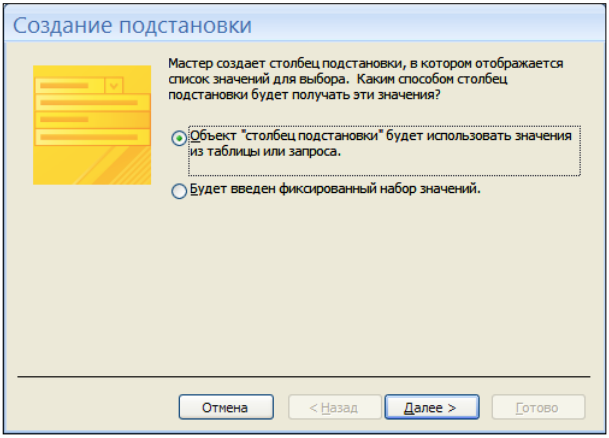

Рис. 4.23. Мастер подстановок (шаг 1)

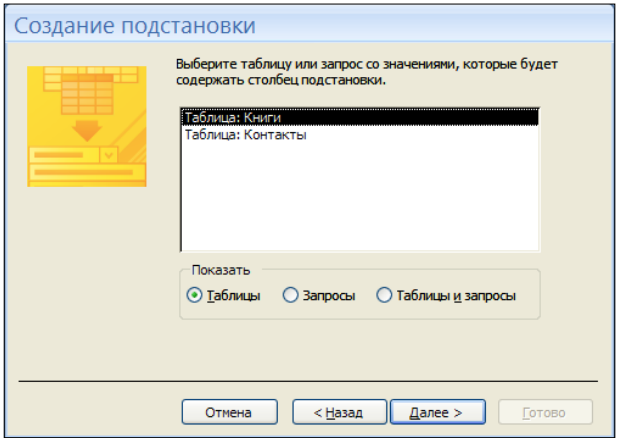

Рис. 4.24. Мастер подстановок (шаг 2)

6.5. Добавляем поля **Название** и **Автор** с помощью кнопки **«>»** и нажимаем **Далее >** (рис. 4.25).

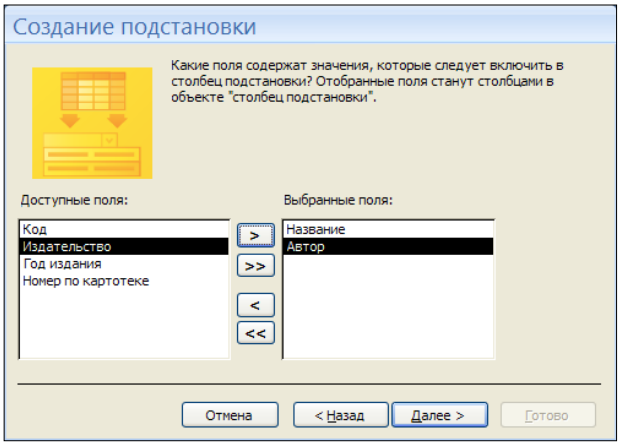

Рис. 4.25. Мастер подстановок (шаг 3)

6.6. Можно выполнить сортировку записей по возрастанию или по убыванию (рис. 4.26), выбрав название поля из списка, нажимаем **Далее >**.

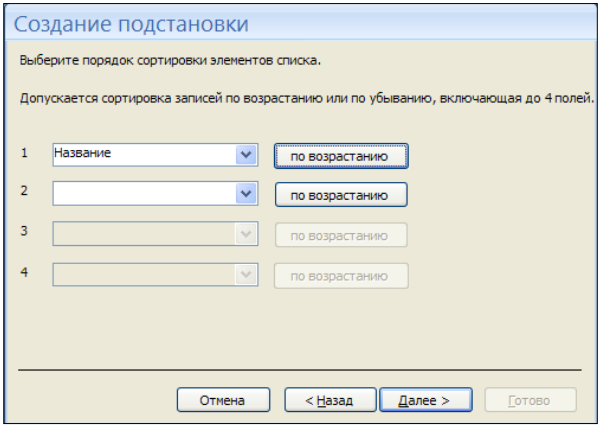

Рис. 4.26. Мастер подстановок (шаг 4)

6.7. Для создания подстановки нажимаем **Готово** (рис. 4.27).

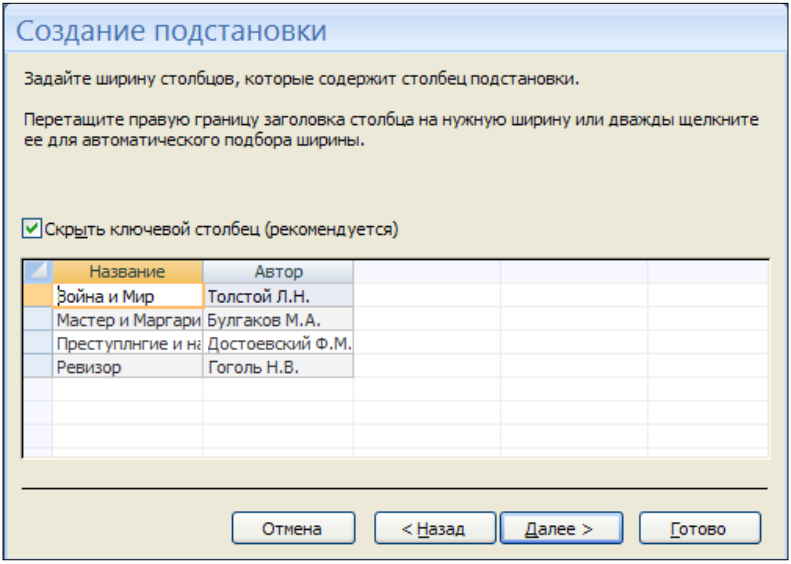

Рис. 4.27. Мастер подстановок (шаг 5)

Теперь в таблице **Записи** рис. 4.28. вместо **Кода книги** будет отображаться название книги и ее автор. Аналогичные действия проделайте со столбцом **Код контакта**, добавив из таблицы **Контакты** поля **Имя** и **Фамилия**.

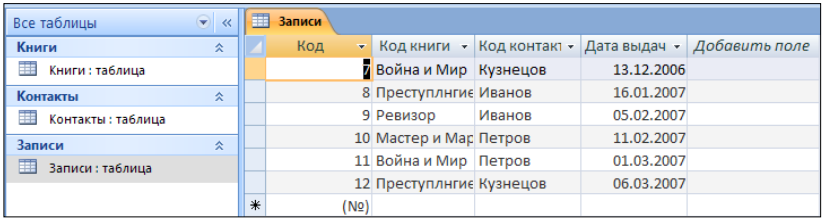

Рис. 4.28. Таблица **Записи**

А при добавлении новой записи будет появляться список книг или контактов (рис. 4.29).

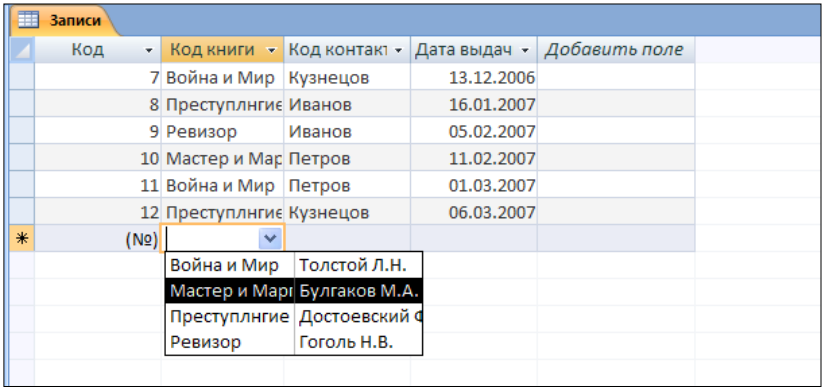

Рис. 4.29. Заполнение таблицы с помощью **Мастера**

Запросы. Запросы являются инструментом поиска и структурирования данных. Запрос, адресованный одной или нескольким таблицам, инициирует выборку определенной части данных и их передачу в таблицу, формируемую самим запросом. В результате вы получаете подмножество информационного множества исходных таблиц, сформированное по определенному закону. Если обрабатываемый объем информации велик, выделение необходимых данных в такое подмножество позволяет существенно сократить время их обработки. В системах типа клиент-сервер, где основные базы данных хранятся на файловом сервере, система запросов позволяет уменьшить объем информации, передаваемой через локальную сеть.

7. *Мастер запросов.*

Чтобы упростить задачу пользователя, в состав Access включен **Мастер запросов**, позволяющий автоматизировать процесс построения запроса.

7.1. Переходим на закладку **Создание** и нажимаем кнопку **Мастер запросов**.

7.2. Выбираем **Простой запрос** и нажимаем **OK**.

7.3. В раскрывающемся списке **Таблицы и запросы** (рис. 4.30) выбираем таблицу **Контакты**, из списка **Доступные поля** выбираем: **Фамилия**, **Имя** и **Рабочий телефон**. Затем из таблицы **Книги** выбираем: **Название** и **Автор**, а из таблицы **Записи** – **Дата выдачи**. И нажимаем **Далее >**.

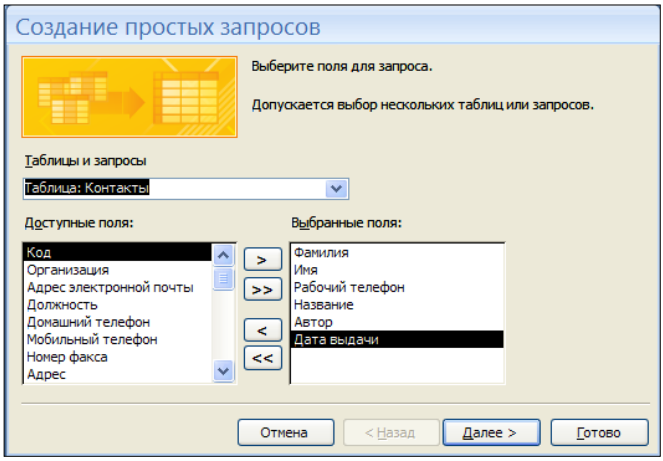

Рис. 4.30. Мастер запросов (шаг 1)

7.4. Выбираем **подробный отчет** (рис. 4.31) и нажимаем **Далее >**.

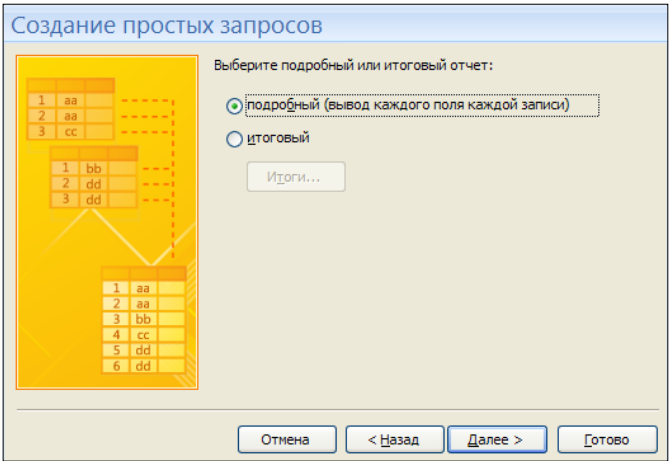

Рис. 4.31. Мастер запросов (шаг 2)

7.5. Вводим имя запроса, ставим переключатель на **Открыть запрос для просмотра данных** (рис. 4.32) и нажимаем **Готово**.

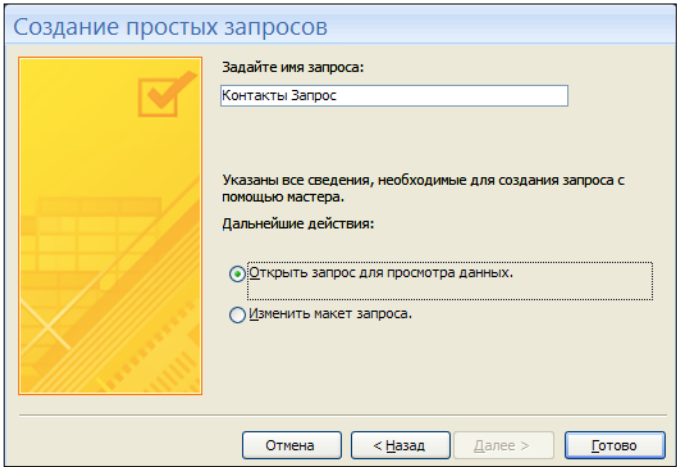

Рис. 4.32. Мастер запросов (шаг 3)

7.6. В результате получаем таблицу (рис. 4.33).

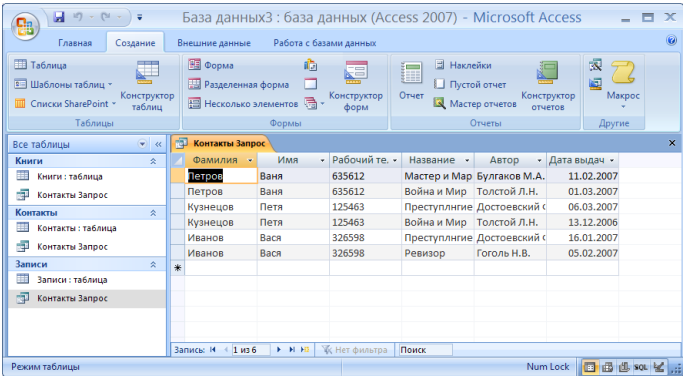

Рис. 4.33. Таблица запроса

8. *Конструктор запросов.*

Мастер запросов умеет конструировать только простые условия отбора. Чтобы наложить дополнительные ограничения, следует пользоваться конструктором запросов, обеспечивающим полное управление параметрами запроса и построение сложных условий отбора данных.

8.1. Переходим на закладку **Создание** и нажимаем кнопку **Конструктор запросов**.

8.2. Появляется диалоговое окно **Добавление таблицы** (рис. 4.34). Добавляем все три таблицы и закрываем окно.

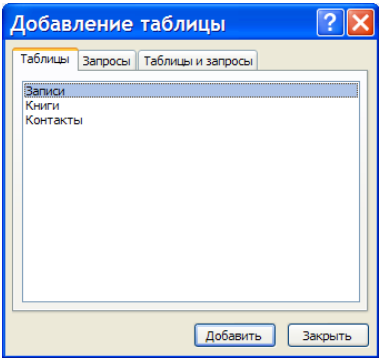

Рис. 4.34. Добавление таблицы

8.3. Из раскрывающегося списка выбираем таблицы и поля для отображения в запросе (рис. 4.35).

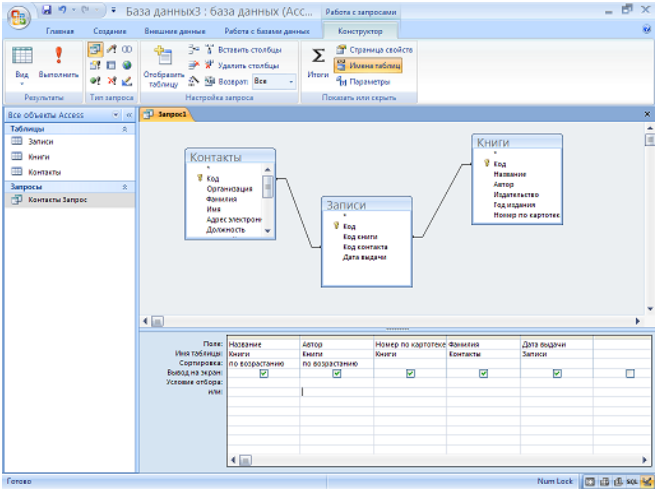

Рис. 4.35. Окно **Конструктор запросов**

9. *Условие отбора.*

Добавление в запрос условия отбора позволяет выбирать из таблицы не все записи, а лишь те, которые удовлетворяют определенным критериям. Например, нам нужны записи, приходящиеся на февраль 2007 г.

9.1. В бланке запроса щелкаем на ячейке **Условие отбора пятого столбца** правой кнопкой мыши и выбираем в контекстном меню команду **Построить**. Откроется окно построителя выражений (рис. 4.36).

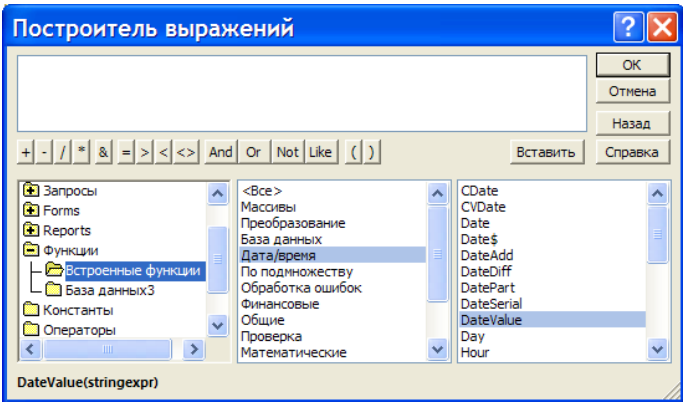

Рис. 4.36. Окно построителя выражений

9.2. В левом списке построителя щелкаем на папке **Операторы**.

9.3. В среднем списке выбираем категорию **Сравнения**.

9.4. В правом списке дважды щелкаем на пункте **Between**.

9.5. Щелчком выделяем в поле формулы первый местозаполнитель **Выражение**.

9.6. В левом списке построителя выражений двойным щелчком открываем папку **Функции**.

9.7. Щелкаем на папке **Встроенные функции**, содержащей стандартные функции Access.

9.8. В среднем списке построителя выражений щелкаем на пункте **Дата/время**.

9.9. В правом списке дважды щелкаем на функции **DateValue**, чтобы заменить ею местозаполнитель **Выражение**.

9.10. Выделив местозаполнитель **stringexpr**, вводим текст **01.02.2007**.

9.11. Повторяя шаги, заменяем второй местозаполнитель **Выражение** на выражение **DateValue** (**31.02.2007**).

В результате у нас получилась формула Between DateValue (**01.02.2007**) and DateValue (**31.02.2007**). Она проверяет условие нахождения даты в интервале от 1 до 31 февраля 2007 г.

9.12. Нажимаем **OK**.

9.13. Для выполнения запроса нажимаем 

. Результат выполнения запроса представлен на рис. 4.37.

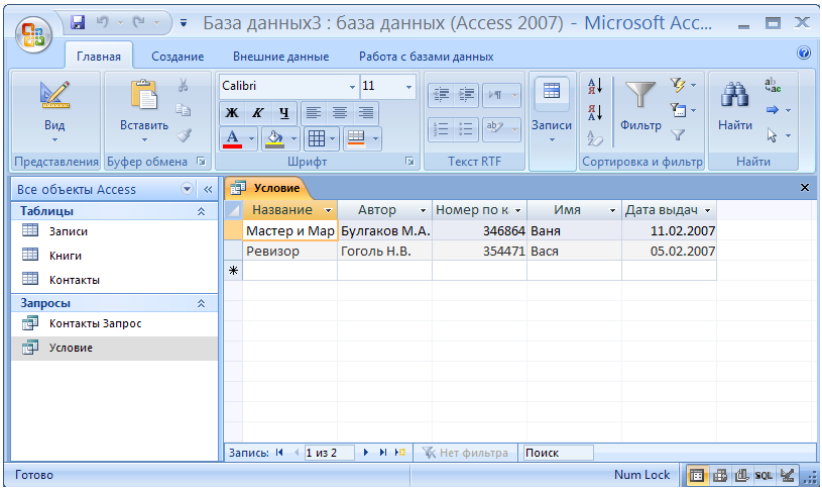

Рис. 4.37. Результат выполнения запроса

10. *Запрос действия.*

Запросы могут применяться также для добавления, удаления и обновления группы записей таблицы. Такие запросы являются мощным инструментом преобразования данных, они называются запросами действия. К примеру, нам нужно изменить в таблице **Книги** номер по картотеке. Заменить первую цифру – 4 на 6. Подобную операцию трудно провести вручную, если в таблице больше тысячи записей.

10.1. Создаем новый запрос в режиме конструктора.

10.2. В окне конструктора открываем таблицу **Книги**.

10.3. В значение поле выбираем **Номер по картотеке** (рис. 4.38).

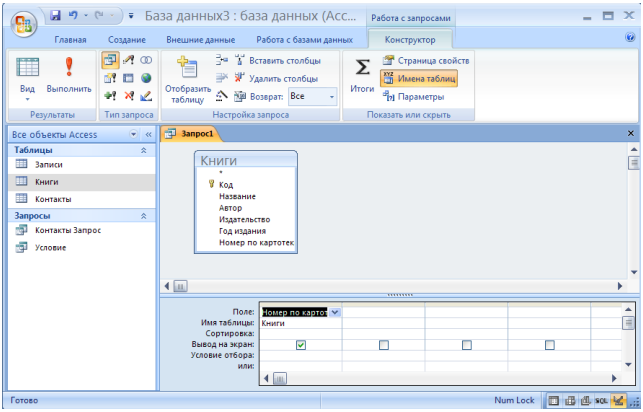

Рис. 4.38. Окно конструктора запроса

10.4. В поле **Тип запроса** выбираем запрос на обновление, при этом в бланке запроса появится еще одно поле **Обновление**, в которое нужно ввести новое значение поля.

10.5. В поле **Обновление** вызываем контекстное меню, щелкаем на пункте **Построить**.

10.6. В окне построителя выражений пишем формулу (рис. 4.39).

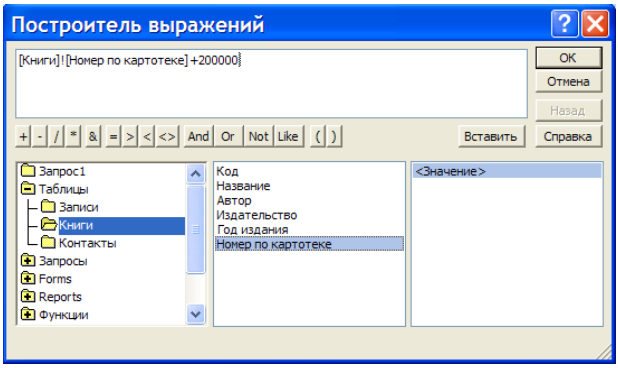

Рис. 4.39. Окно построителя выражений

10.7. В поле **Условие отбора** вызываем контекстное меню, щелкаем на пункте **Построить**.

10.8. В окне построителя выражений пишем формулу (рис. 4.40).

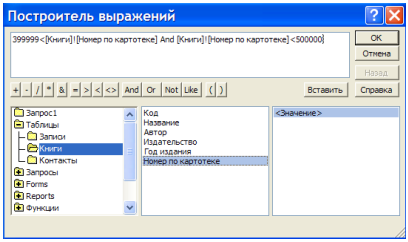

Рис. 4.40. Окно построителя выражений

10.9. Закрываем запрос, сохранив под именем **Обновление**.

10.10. Двойным кликом выполняем запрос **Обновление**, Access проинформирует об обнаружении одной записи, удовлетворяющей условию отбора, и попросит подтвердить необходимость ее изменения.

Виды запросов действия представлены в табл. 4.2.

Таблица 4.2. **Виды запросов действия**

Тип		Описание
Выбор		Выполняет отбор записей из базы данных и показывает их
Обновление		Обновляет данные в существующей таблице
Объединение		Запрос в режиме SQL
Создание таблицы		Выполняет отбор записей из базы данных и сохраняет их как новую таблицу
Перекрестный		Выполняет сведение данных по двум наборам значений, один из которых отображается в левой части таблицы, а другой – в верхней ее части
Запрос к серверу		SQL запрос на сервер
Добавление		Добавляет данные в существующую таблицу
Удаление		Удаляет данные, соответствующие указанному условию из указанной таблицы
Управление		

Формы. В то время как таблицы и запросы позволяют отобразить на экране длинные списки записей, формы дают возможность сосредоточиться на конкретной записи. Они облегчают ввод, редактирование и восприятие информации, могут содержать вспомогательные подписи и элементы оформления.

11. *Мастер форм*.

11.1. Переходим на закладку **Создание** и запускаем **Мастер форм** (рис. 4.41).

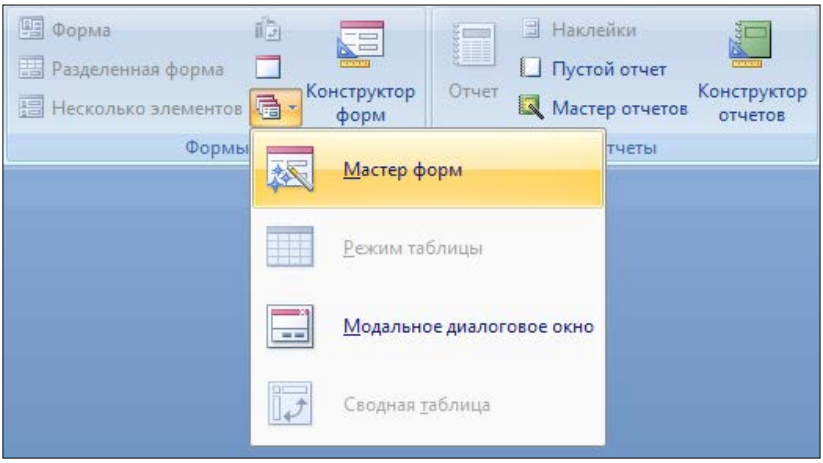

Рис. 4.41. Запуск **Мастера форм**

11.2. В списке **Таблицы и запросы** выберите **Таблица: Контакты**.

11.3. Кликом на кнопке **«>>»** добавьте все поля в список выбранных полей.

11.4. Выделите поле **Код** и кликом на кнопке **«<»** уберите из списка выбранных полей, так как значение этого поля является кодом, оно неважно для пользователя и поэтому его не следует включать в форму (рис. 4.42).

11.5. Кликните на кнопке **Далее >**.

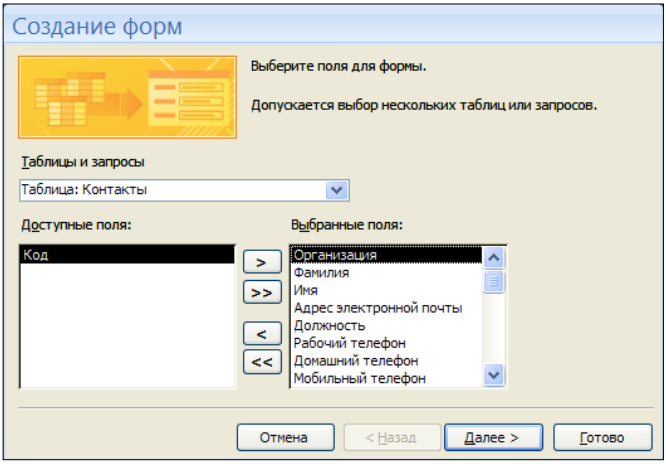

Рис. 4.42. Мастер форм

11.6. В следующем окне выберите внешний вид формы – выровненный – и кликните на кнопке **Далее >**.

11.7. В третьем окне мастера выберите стиль оформления формы (какая понравится) и нажмите на кнопку **Далее >**.

11.8. В четвертом окне нажмите на кнопку **Готово**, ничего не изменяя. В результате получим форму, представленную на рис. 4.43.

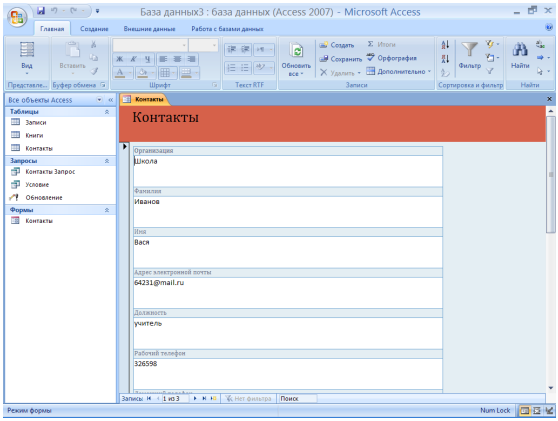

Рис. 4.43. Форма для таблицы **Контакты**

В окне формы можно редактировать уже существующие записи, а также создавать новые.

Конструктор форм. Недостатком форм, создаваемых мастером, является то, что они однообразны и не содержат пояснительных надписей. Чтобы приукрасить форму и расположить поля более удобным образом, следует воспользоваться конструктором форм, который позволяет передвигать и масштабировать элементы формы, связывать их с источником данных и настраивать любые другие параметры формы.

12. *Конструктор форм*.

12.1. Создадим, как показано в подразделе 4.1, форму на таблицу **Книги** (рис. 4.44).

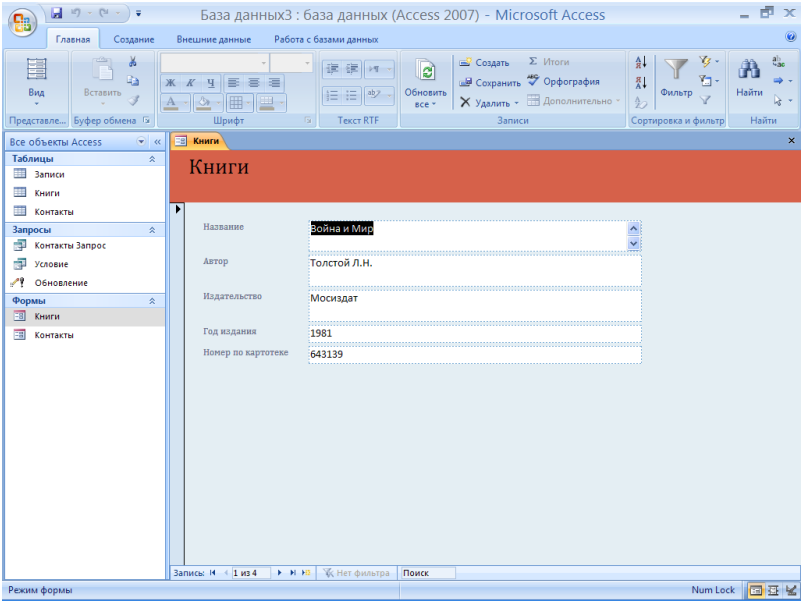

Рис. 4.44. Форма для таблицы **Книги**

12.2. Нажимаем кнопку **Вид** и переходим в режим конструктора (рис. 4.45).

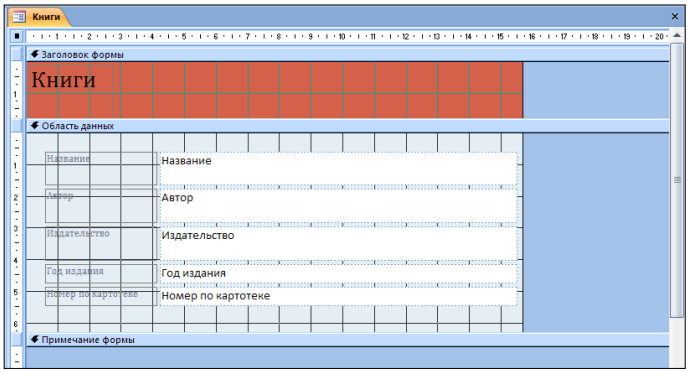

Рис. 4.45. Окно конструктора форм

В окне конструктора присутствует разметочная сетка с вертикальной и горизонтальной линейкой, помогающая позиционировать объекты.

При нажатии ПКМ в поле **Заголовок формы** или **Область данных**, пункт **Цвет заливки/фона** можно изменить цвет поля.

Кликнув на любом элементе формы, вокруг него появляется рамка, потянув за края которой можно изменить форму и размер элемента или переместить его в другое место формы.

Каждый элемент формы имеет свои свойства, окно свойств открывается нажатием ПКМ на элементе, пункт **Свойства** (рис. 4.46).

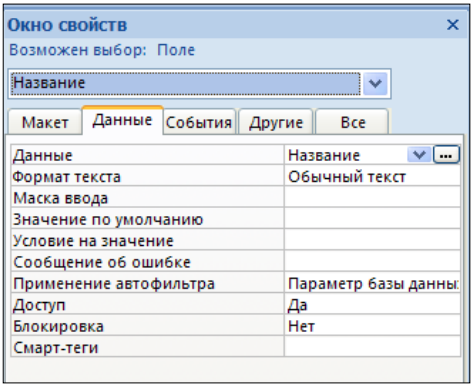

Рис. 4.46. Окно свойств

При открытии конструктора форм на экране появляется панель элементов. С помощью кнопок панели элементов в форму можно добавлять различные объекты, типы которых перечислены в табл. 4.3.

Таблица 4.3. **Элементы управления форм**

Название элемента		Назначение
1	2	3
Поле		Используется для ввода и отображения информации полей таблиц и запросов, а также для вывода результатов вычислений
Подпись		Подписи создаются автоматически вместе с элементами типа текстовое поле, поле со списком и т. п. Они разъясняют смысл поля ввода. Дополнительные надписи могут использоваться для заголовков и пояснений
Кнопка		Щелчок на такой кнопке выполняет команду, с помощью которой можно перейти в другую форму, переместиться по записям и выполнить многие другие операции
Поле со списком		Разрешает как непосредственный ввод значения в поле, так и его выбор в раскрывающемся списке предопределенных значений
Список		Позволяет выбирать данные из набора пунктов и не разрешает непосредственный ввод значений
Подчиненная форма/отчет		Вставляет в форму (или отчет) область с другой формой (или отчетом), связанной с главной
Линия		Добавляет прямую линию
Прямоугольник		Рисует прозрачный или непрозрачный прямоугольник с цветной границей любой толщины
Присоединенная рамка объекта		Объект OLE, связанный с полем данных таблицы или запроса
Группа переключателей		Группа переключателей ссылается на некоторое поле таблицы. Каждый переключатель группы соответствует определенному целочисленному значению этого поля

Окончание табл. 4.3

1	2	3
Флажок		Предназначен для представления полей типа да/нет. Отмеченный квадратик соответствует величине «да», а пустой – «нет»
Переключатель		Элемент группы переключателей, в которой может быть отмечен только один из них (его кружок выглядит зачерненным)
Выключатель		Позволяет вводить информацию типа да/нет. Величине «да» или «истина» соответствует утопленное положение выключателя
Вкладка		Позволяет разместить на одном и том же пространстве экрана несколько наборов элементов управления. Удобен в тех случаях, когда элементы легко разделяются на логические группы
Вставить диаграмму		
Свободная рамка объекта		Объект OLE, не связанный ни с каким источником данных
Рисунок		Рисунок любого графического формата с рамкой
Разрыв страниц		Линия, по которой формируется перевод страницы при выводе формы или отчета на принтер
Вставить гиперссылку		Создание ссылки на web-страницу, рисунок, адрес электронной почты или программу
Вложение		Присоединенные файлы

Отчеты. В целом отчеты похожи на формы, но они, как правило, предназначаются для вывода информации из базы данных на принтер. Поэтому в отчетах данные форматируют так, чтобы их было удобно размещать на отдельных страницах. Отчеты поддерживают самые разнообразные способы оформления и позволяют группировать данные, разбивая их на логически цельные блоки.

13. *Мастер отчетов.*

Чтобы облегчить работу пользователя, в Access имеется специальный мастер, который при недостатке времени позволяет быстро создавать довольно привлекательные отчеты.

13.1. Переходим на закладку **Создание** и запускаем **Мастер отчетов** (рис. 4.47).

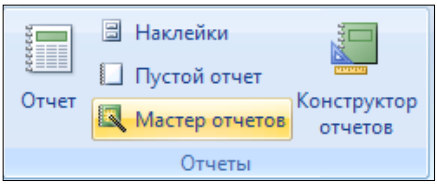

Рис. 4.47. Запуск **Мастера отчетов**

13.2. В открывшемся окне **Мастера**, выбираем **Таблица: Контакты**, добавляем все поля кроме поля **Код**, нажимаем **Далее >** (рис. 4.48).

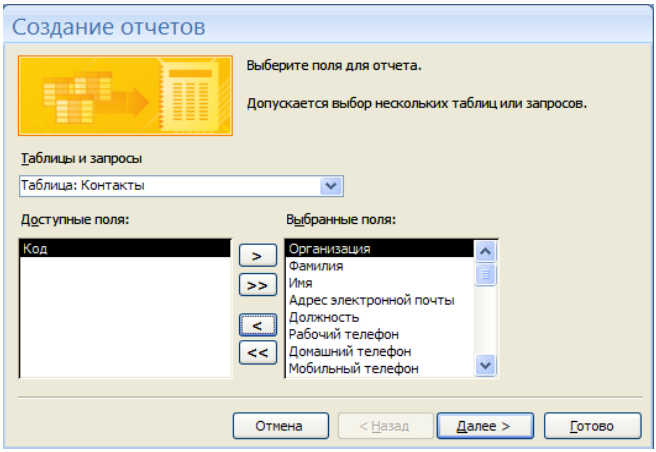

Рис. 4.48. Мастер отчетов (шаг 1)

13.3. В следующем окне **Мастера** оставляем все без изменения и нажимаем **Далее >.**

13.4. Выбираем сортировку по фамилии и нажимаем **Далее >** (рис. 4.49).

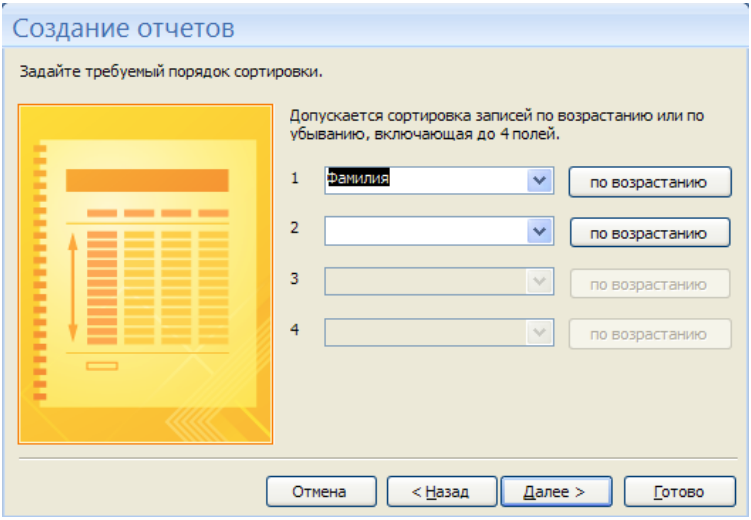

Рис. 4.49. Мастер отчетов (шаг 2)

13.5. Ставим переключатели, как показано на рис. 4.50, и нажимаем **Далее >**.

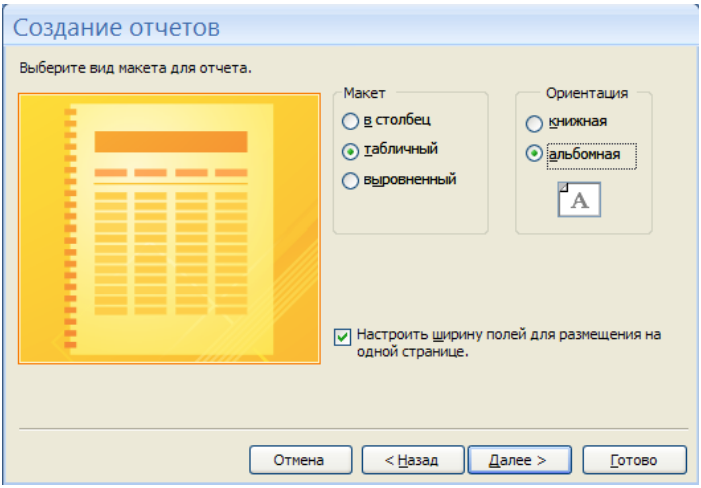

Рис. 4.50. Мастер отчетов (шаг 3)

13.6. Выбираем стиль оформления отчета – Access 2007 – и нажимаем **Готово** (рис. 4.51).

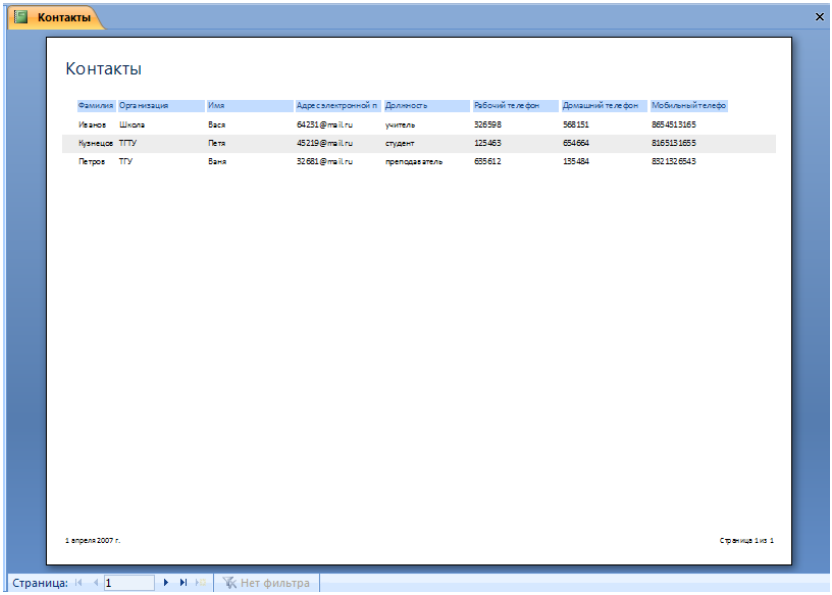

Рис. 4.51. Отчет **Контакты**

14. *Конструктор отчетов.*

Отчет, представленный на рис. 4.52, имеет ряд недостатков, таких как: одни поля занимают слишком много места, а другие поля не умещаются; отчет занимает больше одной страницы и др.

Эти недостатки можно исправить с помощью **Конструктора отчетов**.

14.1. Переходим в режим конструктора, щелкая ПКМ на отчете, пункт – Конструктор (рис. 4.52).

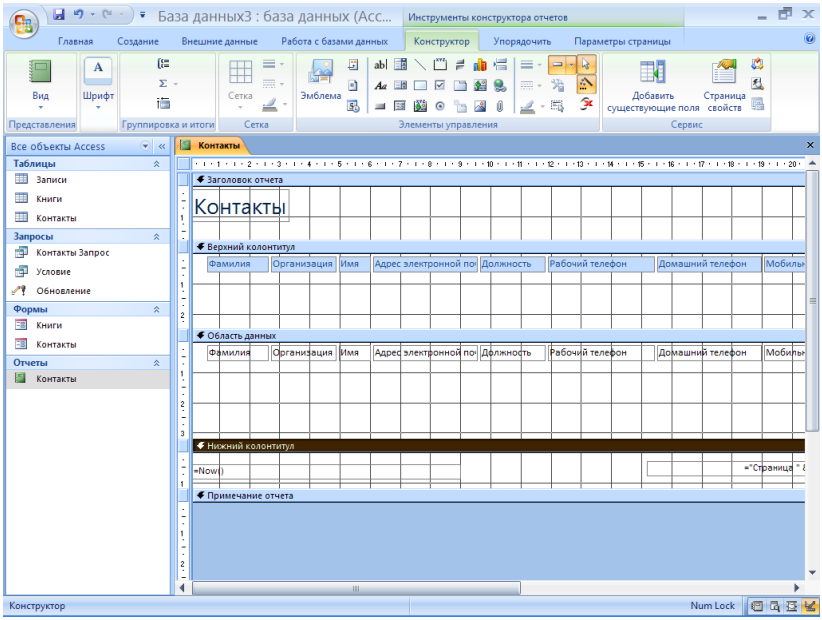

Рис. 4.52. Конструктор отчетов

14.2. Конструктор отчетов напоминает конструктор форм, в нем также можно изменять размеры, форму и положение элементов. Переместим элементы так, чтобы они не выходили за пределы одного листа и занимали место соответственно содержимому. После редактирования переходим в режим предварительного просмотра.

Контрольные вопросы

1. Что представляет собой база данных?
2. В чем преимущества БД по сравнению с файловой организацией данных?
3. Назовите приложения и компоненты БД.
4. Кто является пользователем БД?
5. Дайте определение понятию «структура данных».
6. Назовите статические и динамические структуры данных.
7. Какие три аспекта включает понятие «модели данных»?

8. Каков принцип связи данных в иерархической и сетевой моделях данных?

9. Как определяют реляционную модель данных?

10. Какими свойствами должна обладать реляционная таблица?

11. Какие условия целостности должны выполняться в реляционной модели данных?

12. Дайте определение постреляционной модели данных.

13. Назовите основные понятия, на которых базируется объектно-ориентированная модель данных.

14. Как определяют объектно-реляционную модель данных?

15. Для чего предназначена многомерная модель данных?

16. Дайте определение понятию «система управления базами данных».

17. Каковы функциональные возможности современных СУБД?

18. Какие три компонента включает архитектура СУБД?

19. Дайте классификацию СУБД по принципу обработки запросов к БД.

20. Каковы направления развития СУБД?

21. Как выполняется запрос к БД в архитектуре файл-сервер?

22. Какие популярные настольные СУБД вы знаете?

23. Дайте определение понятию «сервер БД».

24. Каковы преимущества архитектуры клиент-сервер?

25. Охарактеризуйте современные серверы БД.

26. Дайте краткую характеристику СУБД Oracle, MS SQL Server, My SQL.

27. Дайте определение понятиям «распределенная БД» и «распределенная СУБД».

ЛИТЕРАТУРА

1. Компьютерные информационные технологии. Система управления базами данных: метод. указания по выполнению лабораторных работ в СУБД Access / М. С. Латушкина [и др.]. – Горки: БГСХА, 2014. – 40 с.