Chapter 1

An Introduction to Database Development

In this chapter, you learn the concepts and terminology of databases and how to design the tables that your forms and reports will use. Finally, you build the actual tables used by this book's Access Auto Auctions example database.

The fundamental concept underlying Access databases is that data is stored in tables. Tables are comprised of rows and columns of data, much like an Excel worksheet. Each table represents a single entity, such as a person or product.

As you work with Access, you'll spend considerable time designing and refining the tables in your Access applications. Table design and implementation are two characteristics that distinguish database development from most other activities you may pursue.

After you understand the basic concepts and terminology, the next important lesson to learn is good database design. Without a good design, you constantly rework your tables, and you may not be able to extract the information you want from your database. Throughout this book, you learn how to use the basic components of Access applications, including queries, forms, and reports. You also learn how to design and implement each of these objects. The Access Auto Auctions case study provides invented examples, but the concepts are not fictitious.

This chapter is not easy to understand; some of its concepts are complex. If your goal is to get right into Access, you may want to skip to Chapter 2 and read about the process of building tables. If you're fairly familiar with Access but new to designing and creating tables, you may want to read this chapter before starting to create tables.

IN THIS CHAPTER

CROSS-REF To jump right into using Access, skip to Chapter 2.

The Database Terminology of Access

Before examining the actual table examples in this book, it's a good idea to have a firm understanding of the terminology that is used when working with databases — especially Access databases. Microsoft Access follows traditional database terminology. The terms *database, table, record, field,* and *value* indicate a hierarchy from largest to smallest.

Databases

Generally, the word *database* is a computer term for a collection of information concerning a certain topic or business application. Databases help you organize this related information in a logical fashion for easy access and retrieval.

Databases aren't only for computers. There are also manual databases; we simply refer to these as *manual filing systems* or *manual database systems.* These filing systems usually consist of people, papers, folders, and filing cabinets — paper is the key to a manual database system. In a real manual database system, you probably have in/out baskets and some type of formal filing method. You access information manually by opening a file cabinet, taking out a file folder, and finding the correct piece of paper. You use paper forms for input, perhaps by using a typewriter. You find information by manually sorting the papers or by copying information from many papers to another piece of paper (or even into an Excel spreadsheet). You may use a spreadsheet or calculator to analyze the data or display it in new and interesting ways.

An Access database is nothing more than an automated version of the filing and retrieval functions of a paper filing system. Access databases store information in a carefully defined structure. Access tables store data in a variety of forms, from simple lines of text (such as name and address) to complex data such as pictures, sounds, or video images. Storing data in a precise, known format enables a database management system (DBMS) like Access to turn data into useful information.

Tables serve as the primary data repository in an Access database. Queries, forms, and reports provide access to the data, enabling a user to add or extract data, and presenting the data in useful ways. Most developers add macros or Visual Basic for Applications (VBA) code to forms and reports to make their applications easier to use.

A relational database management system (RDBMS), such as Access, stores data in related tables. For instance, a table containing employee data (names and addresses) may be related to a table containing payroll data (pay date, pay amount, and check number). Queries allow the user to ask complex questions (such as "What is the sum of all paychecks issued to Jane Doe in 2007?") from these related tables, with the answers displayed as on-screen forms and printed reports.

In Access, a *database* is the overall container for the data and associated objects. It is more than the collection of tables, however — a database includes many types of objects, including queries, forms, reports, macros, and code modules.

Access works a single database at a time. As you open Access, a single database is presented for you to use. You may open several copies of Access at the same time and simultaneously work with more than one database.

Many Access databases contain hundreds, or even thousands, of tables, forms, queries, reports, macros, and modules. With a few exceptions, all of the objects in an Access 2007 database reside within a single file with an extension of `accdb`, `.accde`, or `.adp`.

The `.adp` file format is a special database format used by Access to act as a front end to work with SQL Server data.

Tables

A table is just a container for raw information (called *data*), similar to a folder in a manual filing system. Each table in an Access database contains information about a single entity, such as a person or product, and the data is organized into rows and columns.

In the section titled "A Five-Step Design Method" later in this chapter, you learn a successful technique for planning Access tables. In Chapters 2 and 3, you learn the very important rules governing relational table design and how to incorporate those rules into your Access databases. These rules and guidelines ensure your applications perform with the very best performance while protecting the integrity of the data contained within your tables.

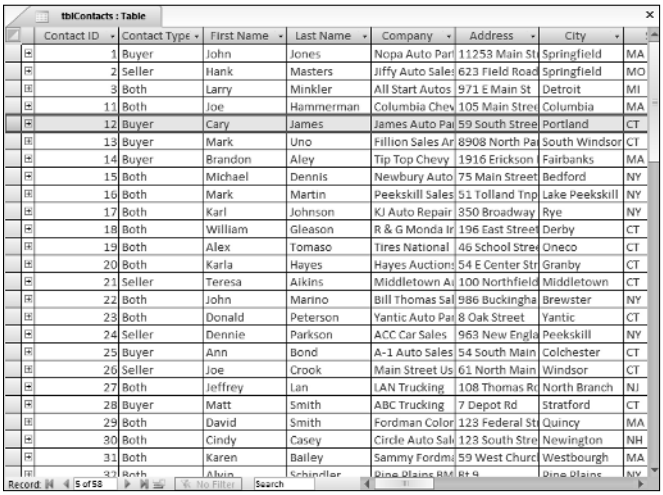
In fact, it is very important that you begin to think of the objects managed by your applications in abstract terms. Because each Access table defines an entity, you must learn to think of the table *as* the entity. As you design and build Access databases, or even when working with an existing application, you must think of how the tables and other database objects represent the physical entities managed by your database.

After you create a table, you view the table in a spreadsheet-like form, called a *datasheet,* comprising rows and columns (known as *records* and *fields,* respectively — see the following section, "Records and fields"). Figure 1-1 shows the datasheet view of the Contacts table in the Access Auto Auction application.

The Contacts table represents people who work with the Auto Auction. Notice how the table is divided into horizontal (left-to-right) rows, and vertical (top-to-bottom) columns of data. Each row (or record) defines a single contact, while each column (or field) represents one type of information known about a contact entity.

FIGURE 1-1

A table displayed as a datasheet



tblContacts : Table

Contact ID	Contact Type	First Name	Last Name	Company	Address	City	
1	Buyer	John	Jones	Nopa Auto Part	11253 Main St	Springfield	MA
2	Seller	Hank	Masters	Jiffy Auto Sales	623 Field Road	Springfield	MO
3	Both	Larry	Minkler	All Start Autos	971 E Main St	Detroit	MI
11	Both	Joe	Hammerman	Columbia Chev	105 Main Stree	Columbia	MA
12	Buyer	Cary	James	James Auto Pa	59 South Stree	Portland	CT
13	Buyer	Mark	Uno	Fillion Sales Ar	8908 North Pa	South Windsor	CT
14	Buyer	Brandon	Aley	Tip Top Chevy	1916 Erickson	Fairbanks	MA
15	Both	Michael	Dennis	Newbury Auto	75 Main Street	Bedford	NY
16	Both	Mark	Martin	Peekskill Sales	51 Tolland Tnp	Lake Peekskill	NY
17	Both	Karl	Johnson	KJ Auto Repair	350 Broadway	Rye	NY
18	Both	William	Gleason	R & G Monda Ir	196 East Street	Derby	CT
19	Both	Alex	Tomaso	Tires National	46 School Stre	Oneco	CT
20	Both	Karla	Hayes	Hayes Auctions	54 E Center Str	Granby	CT
21	Seller	Teresa	Aikins	Middletown Au	100 Northfield	Middletown	CT
22	Both	John	Marino	Bill Thomas Sal	986 Buckingha	Brewster	NY
23	Both	Donald	Peterson	Yantic Auto Par	8 Oak Street	Yantic	CT
24	Seller	Dennie	Parkson	ACC Car Sales	963 New Engla	Peekskill	NY
25	Buyer	Ann	Bond	A-1 Auto Sales	54 South Main	Colchester	CT
26	Seller	Joe	Crook	Main Street Us	61 North Main	Windsor	CT
27	Both	Jeffrey	Lan	LAN Trucking	108 Thomas Rd	North Branch	NJ
28	Buyer	Matt	Smith	ABC Trucking	7 Depot Rd	Stratford	CT
29	Both	David	Smith	Fordman Color	123 Federal St	Quincy	MA
30	Both	Cindy	Casey	Circle Auto Sal	123 South Stre	Newington	NH
31	Both	Karen	Bailey	Sammy Fordma	59 West Churc	Westbourgh	MA
32	Both	Alvin	Schindler	Pine Plains BM	Rt 9	Pine Plains	NY

Record: 5 of 58 No Filter Search

For instance, the top row in tblContacts contains data describing John Jones, including his first name and last name, his address, and the company he works for. Each bit of information describing Mr. Jones is a field (`FirstName`, `LastName`, `Address`, `Company`, and so on). Fields are combined to form a record, and records are grouped to build the table.

Each field in an Access table includes many properties that specify the type of data contained within the field, and how Access should handle the field's data. These properties include the name of the field (`LastName`) and the type of data in the field (`Text`). A field may include other properties as well. For instance, the Size property tells Access how many characters to allow for a person's last name. (You learn much more about fields and field properties in Chapter 2.)

Records and fields

As Figure 1-1 shows, the datasheet is divided into rows (called *records*) and columns (called *fields*), with the first row (the heading on top of each column) containing the names of the fields in the database. Each row is a single record containing fields that are related to that record. In a manual system, the rows are individual forms (sheets of paper), and the fields are equivalent to the blank areas on a printed form that you fill in.

Values

At the intersection of a row (record) and a column (field) is a *value* — the actual data element. For example, John, the name in the first record, represents one data value. You may have a couple questions, such as: What makes this row different from other rows in the table? Is it possible to

have another John Jones in the same table? If there is more than one John Jones, how does the database tell them apart?

Relational Databases

Microsoft Access is a relational database development system. Access data is stored in related tables, where data in one table (such as customers) is related to data in another table (such as orders). Access maintains the relationships between related tables, making it easy to extract a customer and all of the customer's orders, without losing any data or pulling order records not owned by the customer.

Working with multiple tables

Multiple tables simplify data entry and reporting by decreasing the input of redundant data. By defining two tables for an application that uses customer information, for example, you don't need to store the customer's name and address every time the customer purchases an item.

After you've created the tables, they need to be related to each other. For example, if you have a Contacts table and a Sales table, you must relate the Contacts table to the Sales table in order to see all the sales records for a Contact. If you had only one table, you would have to repeat the Contact name and address for each sale record. Two tables let you look up information in the Contact table for each sale by using the related fields Contact ID (in Contacts) and Buyer ID (in Sales). This way, when a customer changes address, for example, the address changes only in one record in the Contact table; when the Sales information is on-screen, the correct contact address is always visible.

Separating data into multiple tables within a database makes the system easier to maintain because all records of a given type are within the same table. By taking the time to segment data properly into multiple tables, you experience a significant reduction in design and work time. This process is known as *normalization.* (You can read about normalization in Chapter 2.)

Later in this chapter in the section titled "A Five-Step Design Method," you have the opportunity to work through a case study for the Access Auto Auctions that consists of five tables.

Knowing why you should create multiple tables

The prospect of creating multiple tables always intimidates beginning database users. Most often, they want to create one huge table that contains all of the information they need — in this case, a Customer table with all the sales performed by the customer and all the items sold or bought for each customer.

So, they create a single table containing a lot of fields, including fields for customer information (contact), sales information (date of sale, salesperson, amount paid, discounts, and so on), and the product information (quantity sold, product description, individual prices, and so on) for each sale. Such a table quickly grows to an unmanageable number of fields and continues growing as new items are added.

As you can see, the table design begins to take on a life of its own. After you've created the single table, it becomes even more difficult to maintain. You begin to realize that you have to input the customer information for every sale a customer makes (repeating the information over and over). The same is true for the items purchased for each sale, which is multiple items for each sale (thus, duplicating information again). This makes the system more inefficient and prone to data-entry mistakes. The information stored in the table becomes inefficiently maintained — many fields may not be appropriate for each record, and the table ends up with a lot of empty fields.

It's important to create tables that hold the minimum of information while still making the system easy to use and flexible enough to grow. To accomplish this, you need to consider making more than one table, with each table containing records with fields that are related only to the focus of that table. Then, after you create the tables, you link them so that you're able to glean useful information from them. Although this process sounds extremely complex, the actual implementation is relatively easy. Again, this process of creating multiple tables from a single table is known as *normalization* (or normalizing your tables).

Access Database Objects and Views

If you're new to databases (or even if you're an experienced database user), you need to understand some key concepts before starting to build Access databases. The Access database contains seven types of top-level objects, which consist of the data and tools that you need to use Access:

- **Table:** Holds the actual data
- **Query:** Searches for, sorts, and retrieves specific data
- **Form:** Lets you enter and display data in a customized format
- **Report:** Displays and prints formatted data
- **Pages:** Publishes data to a corporate intranet
- **Macro:** Automates tasks without programming
- **Module:** Contains programs written in the Visual Basic for Applications (VBA) programming language

Datasheets

Datasheets are one of the many ways by which you can view data in Access. Although not a database object, a datasheet displays a list of records from a table in a format similar to an accounting spreadsheet or Excel worksheet. A datasheet displays data as a series of rows and columns (comparable to an Excel spreadsheet). A datasheet displays a table's information in its raw form. The datasheet view is the default mode for displaying all fields for all records.

You scroll through the datasheet using the directional keys on your keyboard. You can also display related records in other tables while in a datasheet. In addition, you can make changes to the displayed data.

CAUTION **Use caution when making changes or allowing a user to modify data in datasheet format. When a datasheet record is updated, the data in the underlying table is permanently changed.**

Queries

Queries extract information from a database. A query selects and defines a group of records that fulfill a certain condition. Many forms and most reports are based on queries that pre-filter data before it is displayed. Queries are often called from VBA procedures to change, add, or delete database records.

An example of a query is when a person at the Auto Sales office tells the database, "Show me all customers, in alphabetical order by name, who live in Massachusetts and bought something over the past six months, and display them sorted by Customer name," or "Show me all customers who bought cars for a value of $35,000 or more for the past six months and display them sorted by customer name and then by value of the car."

Instead of asking the question in English words, the person uses the query by example (QBE) method. When you enter instructions into the QBE Design window, the query translates the instructions into Structured Query Language (SQL) and retrieves the desired data. Chapter 4 discusses the QBE Design window and building queries.

In the first example, the query first combines data from both the Sales and Contact tables, using the related field Contact ID (the common link between the tables). Next, it retrieves the first name, last name, and any other data you want to see. Access then filters the records, selecting only those in which the value of the sales date is within six months of the current date. The query sorts the resulting records first by contact's last and first names. Finally, the records appear on-screen in a datasheet.

A similar action takes place for the second example — using sales, contacts, invoice items, and products and the criteria applied to the search is where the `Description` field has a car bought whose value in the `Price` field is greater than or equal to $35,000.

After you run a query, the resulting set of records may be used in a form that is displayed on-screen or printed on a report. In this way, user access is limited to the data that meets the criteria in the returned records.

Data-entry and display forms

Data-entry forms help users get information into a database table quickly, easily, and accurately. Data-entry and display forms provide a more structured view of the data than what a datasheet provides. From this structured view, database records can be viewed, added, changed, or deleted. Entering data through the data-entry forms is the most common way to get the data into the database table.

Data-entry forms restrict access to certain fields within the table. Forms also check the validity of your data before it is added to the database table.

Most users prefer to enter information into data-entry forms rather than datasheet views of tables. Data-entry forms often resemble familiar paper documents and can aid the user with data-entry tasks. Forms make data entry self-explanatory by guiding the user through the fields of the table being updated.

Display-only screens and forms are solely for inquiry purposes. These forms allow for the selective display of certain fields within a given table. Displaying some fields and not others means that you can limit a user's access to sensitive data while allowing inquiry into other fields.

Reports

Reports present your data in printed format. Access supports several different types of reports. A report may list all records in a given table (such as a customer table) or may list only the records meeting a certain criterion, such as all customers living in the State of Washington. You do this by basing the report on a query that selects only the records needed by the report.

Your reports can combine multiple tables to present complex relationships among different sets of data. An example is printing an invoice. You access the customer table to obtain the customer's name and address (and other relevant data) and related records in the sales table to print the individual line-item information for the products ordered. You then instruct Access to calculate the totals and print them in a specific format on the form. Additionally, you can have Access output records into an *invoice report,* a printed document that summarizes the invoice.

TIP **When you design your database tables, keep in mind all the types of information that you want to print. Doing so ensures that the information you require in your various reports is available from within your database tables.**

Designing the system's objects

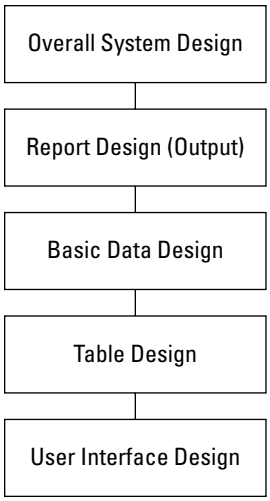
To create database objects, such as tables, forms, and reports, you first complete a series of tasks known as *design.* The better your design is, the better your application will be. The more you think through your design, the faster you can complete any system. The design process is not some necessary evil, nor is its intent to produce voluminous amounts of documentation. The sole intent of designing an object is to produce a clear-cut path to follow as you implement it.

A Five-Step Design Method

Figure 1-2 is a version of the design method that is modified especially for use with Access. This is a top-down approach, starting with the overall system design and ending with the forms design, and it consists of five steps.

FIGURE 1-2

The five-step design flowchart. This design methodology is particularly well-suited for Access databases.



These five design steps, along with the database system illustrated by the examples in this book, teach a great deal about Access and provide a great foundation for creating database applications — including tables, queries, forms, data pages, reports, macros, and simple VBA (Visual Basic for Applications) modules.

The time you spend on each step depends entirely on the circumstances of the database you're building. For example, sometimes the users give you an example of a report they want printed from their Access database, and the sources of data on the report are so obvious that designing the report takes a few minutes. Other times, particularly when the users' requirements are complex, or the business processes supported by the application require a great deal of research, you may spend many days on Step 1.

As you read through each step of the design process, *always* look at the design in terms of outputs and inputs. Although you see actual components of the system (cars, buyers, sellers, and transactions), remember that the focus of this chapter is how to design each step. As you watch the Access Auto Auctions system being designed, pay attention to the design process, not the actual system.

Step 1: The overall design — from concept to reality

All software developers face similar problems, the first of which is determining how to meet the needs of the end user. It's important to understand the overall requirements before zeroing in on the details.

The five-step design method shown in Figure 1-2 helps you to create the system that you need, at an affordable price (measured in time or dollars). The Access Auto Auctions database, for example, allows the client to sell items (vehicles and parts) to customers. The Access Auto Auctions database automates the following tasks:

- Entering and maintaining contact information for customers and sellers (name, address, and financial history)
- Entering and maintaining sales information (sales date; payment method; total amount, including tax; buyer ID; and other fields)
- Entering and maintaining sales line item information (details of items actually purchased)
- Viewing information from all the tables (sales, contacts, sales line items purchased, and payment information)
- Asking all types of questions about the information in the database
- Producing a current contacts directory
- Producing a monthly invoice report
- Producing a customer sales history
- Producing mailing labels and mail-merge reports

These nine tasks that the Access Auto Auctions automates have been expressed by the client. You may need to consider other tasks as you start the design process.

Most of the information that is necessary to build the system comes from the eventual users. This means that you need to sit down with them and learn how the existing process works. To accomplish this you need to do a thorough *needs analysis* of the existing system and how you might automate it.

One way to accomplish this is to prepare a series of questions that give insight to the client's business and how the client uses his data. For example, when considering automating an auto auction business, you may consider asking these questions:

- What reports and forms are currently used?
- How are sales, customer, contacts, and other records currently stored?
- How are billings processed?

As you ask these questions and others, the client will probably remember other things about his business that you should know.

A walkthrough of the existing process is also necessary to get a "feel" for the business. Most likely, you'll have to go back several times to observe the existing process and how the employees work.

When you prepare to follow the remaining steps, keep the client involved — let him know what you're doing and ask for his input as to what you want to accomplish, making sure it is within the scope of his needs.

Step 2: Report design

Although it may seem odd to start with reports, in many cases users are more interested in the printed output from a database than they are in any other aspect of the application. Reports often include virtually every bit of data managed by an application. Because reports tend to be comprehensive, reports are often the best way to gather important information about a database's requirements. In the case of the Access Auto Auctions database, the printed reports contain detailed and summarized versions of most all the data in the database.

After you've defined the Access Auto Auctions' overall systems in terms of what must be accomplished, you can begin report design.

When you see the reports that you will create in this section, you may wonder, "Which comes first — the chicken or the egg?" Does the report layout come first, or do you first determine the data items and text that make up the report? Actually, these items are considered at the same time.

It isn't important how you lay out the fields in a report. The more time you take now, however, the easier it will be to construct the report. Some people go so far as to place gridlines on the report so that they will know the exact location they want each bit of data to occupy. In this example, you can just do it visually.

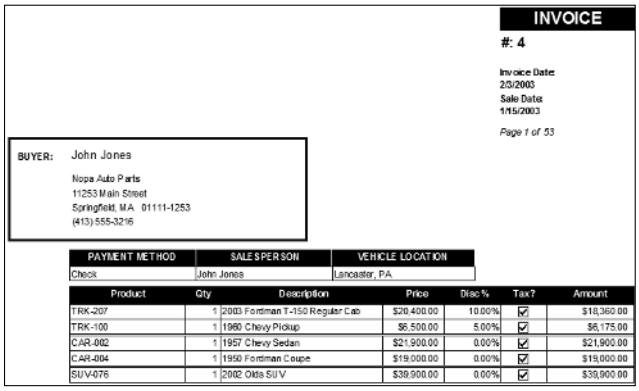
The reports in Figures 1-3 and 1-4 were created with two different purposes. The report in Figure 1-3 displays information about an individual contact (buyer, seller, or both). In contrast, the report in Figure 1-4 is an invoice with billing and customer information. Both of these reports were based on the type of information they use. The design and layout of each report is driven by the report's purpose and the data it contains.

FIGURE 1-3

A contact information report

FIGURE 1-4

A sales invoice report containing sales information



INVOICE

#: 4

Invoice Date: 2/3/2003
Sale Date: 1/15/2003

Page 1 of 53

BUYER: John Jones

Nopa Auto Parts
11253 Main Street
Springfield, MA 01111-1253
(413) 555-3216

PAYMENT METHOD	SALESPERSON	VEHICLE LOCATION
Check	John Jones	Lancaster, PA

Product	Qty	Description	Price	Disc %	Tax?	Amount
TRK-207	1	2003 Fortiman T-150 Regular Cab	$20,400.00	10.00%	☑	$18,360.00
TRK-100	1	1960 Chevy Pickup	$6,500.00	5.00%	☑	$6,175.00
CAR-002	1	1957 Chevy Sedan	$21,900.00	0.00%	☑	$21,900.00
CAR-004	1	1950 Fortiman Coupe	$19,000.00	0.00%	☑	$19,000.00
SUV-076	1	2002 Olds SUV	$39,900.00	0.00%	☑	$39,900.00

CROSS-REF **You can read more about the reports for the Access Auto Auctions system in Chapters 9 and 20.**

Step 3: Data design: What fields are required?

The next step in the design phase is to take an inventory of all the information or data fields that are needed by the reports. One of the best methods is to list the data items in each report. As you do so, take careful note of items that are included in more than one report. Make sure that you keep the same name for a data item that is in more than one report because the data item is really the same item.

Another method is to see whether you can separate the data items into some logical arrangement. Later, these data items are grouped into table structures and then mapped onto data-entry screens (forms). You should enter customer data (buyers and sellers), for example, as part of a contact table process, not as part of a sales entry.

Determining contact information

First, look at each report you have reviewed or attempted to make for the Access Auto Auctions system. For this system, start with the customer data and list the data items, as shown in Table 1-1.

TABLE 1-1

Customer-Related Data Items Found in the Reports

Contacts Report	Invoice Report
Customer Name	Customer Name
Street	Street
City	City
State	State
ZIP Code	ZIP Code
Phone Numbers	Phone Number
Type of Customer	
E-Mail Address	
Web Site Information	
Contact Log Information (four fields)	
Discount Rate	
Customer Since	
Last Sales Date	
Sales Tax Rate	
Credit Information (four fields)	

As you can see by comparing the type of contact (customer) information needed for each report, there are many common fields. Most of the data fields pertaining to the customer are found in both reports. Table 1-1 shows only some of the fields that are used in each report — those related to customer information. Fields appearing on both reports appear on the same rows in the table, which allows you to see more easily which items are in which reports. You can look across a row instead of looking for the same names in both reports. Because the related row and the field names are the same, you can easily make sure that you have all the data items. Although locating items easily is not critical for this small database, it becomes very important when you have to deal with large tables containing many fields.

Determining sales information

After extracting the customer data, you can move on to the sales data. In this case, you need to analyze only the Invoice report for data items that are specific to the sales. Table 1-2 lists the fields in the report that contain information about the sales.

TABLE 1-2

Sales Data Items Found in the Reports

Individual Invoice Report	Line Item Data
Invoice Number	
Sales Date	
Invoice Date	
Payment Method	
Payment Salesperson	
Discount (overall for sale)	
Tax Location	
Tax Rate	
Product Purchased (multiple lines)	Product Purchased
Quantity Purchased (multiple lines)	Quantity Purchased
Description of Item Purchased (multiple lines)	Description of Item Purchased
Price of Item (multiple lines)	Price of Item
Discount for each item (multiple lines)	Discount for Each Item
Taxable? (multiple lines)	Taxable?
Payment Type (multiple lines)	
Payment Date (multiple lines)	
Payment Amount (multiple lines)	
Credit Card Number (multiple lines)	
Expiration Date (multiple lines)	

As you can see when you examine the type of sales information needed for the report, a couple of items (fields) are repeating (for example, the `Product Purchased`, `Number of Items Purchased`, and `Price of Item` fields). Each invoice can have multiple items, and each of these items needs the same type of information — number ordered and price per item. Each sales invoice will probably have more than one item that is sold and being invoiced. Also, each invoice may include partial payments, and it is possible that this payment information will have multiple lines of payment information, so these repeating items can be put into their own grouping.

Determining line item information

You can take all the individual items that you found in the sales information group in the preceding section and extract them to their own group for the invoice report. Table 1-2 shows the information related to each line item.

Looking back at the report in Figure 1-4, you can see that the data from Table 1-2 doesn't list the calculated field amount, but you can re-create it easily in the report.

TIP **Unless a numeric field needs to be specifically stored in a table, simply recalculate it when you run the report (or form). You should avoid creating fields in your tables that can be created based on other fields — these calculation fields can be easily created and displayed in a form or report. As you'll read in Chapter 2, storing calculated values in database tables leads to data maintenance problems.**

Step 4: Table design

Now for the difficult part: You must determine what fields are needed for the tables that make up the reports. When you examine the multitude of fields and calculations that make up the many documents you have, you begin to see which fields belong to the various tables in the database. (You already did much of the preliminary work by arranging the fields into logical groups.) For now, include every field you extracted. You will need to add others later (for various reasons), although certain fields won't appear in any table.

It is important to understand that it isn't necessary to add every little bit of data into the database's tables. For instance, users may express a desire to add vacation and other out-of-office days to the database to make it easy to know which employees are available on a particular day. However, it is very easy to burden an application's initial design by incorporating too many ideas during the initial development phases. Because Access tables are so easy to modify later on, it is probably best to put aside noncritical items until the initial design is complete. Generally speaking, it's not difficult to accommodate user requests after the database development project is under way.

After you've used each report to display all the data, it's time to consolidate the data by purpose (for example, grouped into logical groups) and then compare the data across those functions. To do this step, first you look at the contact information and combine all of its different fields to create one set of data items. Then you do the same thing for the sales information and the line item information. Table 1-3 compares data items from these three groups of information.

TABLE 1-3

Comparing the Data Items from the Contact Information, Sales Information, and Line Item Information

Contacts Data	Invoice Data	Line Items
Customer Name	Invoice Number	Product Purchased
Street	Sales Date	Quantity Purchased
City	Invoice Date	Description of Item Purchased
State	Payment Method	Price of Item

continued

TABLE 1-3 *(continued)*

Contacts Data	Invoice Data	Line Items
ZIP Code	Payment Salesperson	Discount for Each Item
Phone Numbers (two fields)	Discount (overall for this sale)	Taxable?
Type of Customer	Tax Location	
E-Mail Address	Tax Rate	
Web Site Information	Payment Type (multiple lines)	
Contact Log Information (four fields)	Payment Date (multiple lines)	
Discount Rate	Payment Amount (multiple lines)	
Customer Since	Credit Card Number (multiple lines)	
Last Sales Date	Expiration Date (multiple lines)	
Sales Tax Rate		
Credit Information (four fields)		

Consolidating and comparing data is a good way to start creating the individual table definitions for Access Auto Auctions, but you have much more to do.

As you learn more about how to perform a data design, you also learn that the contacts data must be split into two groups. Some of these items are used only once for a contact while other items may have multiple entries. An example is the Contact Log information. Each contact may have multiple log items recorded in the database. This is also true for the Sales column — the payment information can have multiple lines of information.

It is necessary to further break these types of information into their own columns, thus separating all related types of items into their own columns — an example of the *normalization* part of the design process. For example, one customer can have multiple contacts with the company. One customer may make multiple payments toward a single sale. Of course, we've already broken the data into three categories above: contacts, invoices, and sales line items.

Keep in mind that one customer may have multiple invoices, and each invoice may have multiple line items on it. The contact category represents customer (buyer or seller) information, the invoice category contains information about individual sales, and the line items category contains information about each invoice. Notice that these three columns are all related; for example, one customer can have multiple invoices and each invoice may require multiple detail lines (line items).

The relationships between tables can be different. For example, each sales invoice has one and only one customer, while each customer may have multiple sales. A similar relationship exists between the sales invoice and the line items of the invoice.

CROSS-REF **We cover creating and understanding relationships and the normalization process in Chapter 2.**

Assuming that the three groupings represent the main three tables of your system, less additional fields, you need to link tables together. This step, of course, means adding table relationships to the database design.

Database table relationships require a unique field in both tables involved in a relationship. Without a unique identifier in each table, the database engine is unable to properly join and extract related data.

None of the tables in our design has a unique identifier, which means that you need to add at least one more field to each table to serve as the anchor for a relationship to other tables. For example, you could add a `ContactID` field to the Contacts table, then add the same field to the Invoice table, and establish a relationship between the tables through the `ContactID` field in each table. (Creating relationships is explained in Chapter 3.) The database engine uses the relationship between the Contacts and Invoices table to link customers with their invoices. Linking tables is done through special fields, known as *key* fields.

With an understanding of the need for linking one group of fields to another group, you can add the required key fields to each group. Table 1-4 shows two new groups and link fields created for each group of fields. These linking fields, known as *primary keys* and *foreign keys,* are used to link these tables together.

The field that uniquely identifies each row in a table is called the *primary key.* The corresponding field in a related table is called the *foreign key.* In our example, the `ContactID` field in the Contacts table is a primary key, while the `ContactID` field in the Invoices table is a foreign key.

Let's assume a certain record in the Contacts table has 12 in its `ContactID` field. Any records in the Invoices table with 12 in its `ContactID` field is "owned" by contact number 12. As you'll see in Chapters 2 and 3, special rules apply to choosing and managing primary and foreign keys. The notion of primary and foreign keys is the single most important concept behind relational databases. You can read much more about this important concept in Chapters 2 and 3.

TABLE 1-4

Main Tables with Keys

Contacts Data	Invoice Data	Line Items Data	Contact Log Data	Sales Payment Data
ContactID	InvoiceID	InvoiceID	ContactLogID	InvoiceID
Customer Name	ContactID	Line Number	ContactID	Payment Type
Street	Invoice Number	Product Purchased	Contact Date	Payment Date
City	Sales Date	Quantity Purchased	Contact Notes	Payment Amount

continued

TABLE 1-4 *(continued)*

Contacts Data	Invoice Data	Line Items Data	Contact Log Data	Sales Payment Data
State	Invoice Date	Description of Item Purchased	Follow Up?	Credit Card Number
ZIP Code	Payment Method	Price of Item	Follow-Up Date	Expiration Date
Phone Numbers (two fields)	Payment Salesperson	Discount for Each Item		
Type of Customer	Discount (overall for this sale)	Taxable?		
E-Mail Address	Tax Location			
Web Site Information	Tax Rate			
Discount Rate				
Customer Since				
Last Sales Date				
Sales Tax Rate				

With the key fields added to each table, you can now find a field in each table that links it to other tables in the database. For example, Table 1-4 shows a `ContactID` field in both the Contacts table (where it is the table's primary key) and the Invoice table (where it is a foreign key).

You have identified the core of the three primary tables for your system, as reflected by the first three columns in Table 1-4. This is the general, or first, cut toward the final table designs. You have also created two additional tables (columns) from fields shown in Table 1-3.

Taking time to properly design your database and the tables contained within it is arguably the most important step in developing a database-oriented application. By designing your database efficiently, you maintain control of the data — eliminating costly data-entry mistakes and limiting your data entry to essential fields.

Although this book is not geared toward teaching database theory and all of its nuances, this is a good point to briefly describe the art of database normalization. You'll read the details of normalization in Chapter 3, but in the meantime you should know that normalization is the process of breaking data down into constituent tables. Earlier in this chapter you read about how many Access developers add dissimilar information, such as contacts, invoice data, and invoice line items, into one large table. A large table containing dissimilar data quickly becomes unwieldy and hard to keep updated. Because a contact's phone number appears in every row containing that customer's data, multiple updates must be made when the contact's phone number changes.

Normalization is the process of breaking data into smaller, more manageable tables. Each table defines one and only one *entity,* such as a contact or an invoice, but not both. The contact and invoice tables are related through a primary key (`ContactID` in the customers table) and a foreign key (also named `ContactID`) in the invoices table.

There is much more involved in the normalization process, but, in the meantime, we'll leave that for Chapter 3.

Step 5: Form design: Input

After you've created the data and established table relationships, it's time to design your forms. *Forms* are made up of the fields that can be entered or viewed in Edit mode. If at all possible, your screens should look much like the forms that you use in a manual system. This setup makes for the most user-friendly system.

When you're designing forms, you need to place three types of objects on-screen:

- Labels and text box data-entry fields (the fields on Access forms and reports are usually called *controls*)
- Special controls (multiple-line text boxes, option buttons, list boxes, check boxes, business graphs, and pictures)
- Graphical objects to visually enhance them (colors, lines, rectangles, and three-dimensional effects)

When designing a form, place your fields (text boxes, check boxes, list boxes, and radio buttons) just where you want them on the form. Ideally, if the form is being developed from an existing printed form, the Access data-entry form should resemble the printed form. The fields should be in the same relative place on the screen as they are in the printed counterpart.

Labels display messages, titles, or captions. Text boxes provide an area where you can type or display text or numbers that are contained in your database. Check boxes indicate a condition and are either unchecked or checked (selected). Other types of controls available with Access include list boxes, combo boxes, option buttons, toggle buttons, and option groups.

CROSS-REF **Chapter 7 covers the various types of controls available in Access.**

In this book, you create several basic data-entry forms:

- **Contact Log:** A simple data-entry form
- **Contacts:** A slightly more complex data-entry form, containing several different types of controls
- **Sales:** Combines data from multiple tables
- **Products:** Data-entry form for adding products to the Access Auto Auction database.

You'll encounter each of these forms as you read through the following chapters. Although the Access Auto Auction is but one type of database application built with Microsoft Access, the principles you learn building the Access Auto Auction tables, queries, forms, reports, and other database objects are applicable to virtually any other Access project.

Summary

This chapter introduces the concepts and considerations driving database development. There is no question that data is important to users. Most companies simply cannot operate without their customer and product lists, accounts receivable and accounts payable, and payroll information. Even very small companies must efficiently manage their business data.

Good database design means much more than sitting down and knocking together a few tables. Very often, poor database design habits come back to haunt developers and users in the form of missing or erroneous information on screens and printed reports. Users quickly tire of re-entering the same information over and over again, and business managers and owners expect database applications to *save* time and money, not contribute to a business's overhead.