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 Access application's forms and reports will use.

Database development is quite unlike most other ways you work with computers. Unlike Microsoft Word or Excel, where the approach to working with the application is easy to understand, good database development requires prior knowledge. A beginning user opening Access for the first time likely has no idea where to start. Although the opening user interface helps you create your first database, from that point on, you're pretty much on your own. Unlike Word or Excel, you can't just start typing things in at the keyboard and see any results.

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. In a properly designed database, each table represents a single entity, such as a person or product. Each row within a table describes a single instance of the entity, such as one person or an individual product. Each column in an Access table contains a single type of data, such as text or date/time.

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 processes that distinguish database development from most other computer activities you may pursue. Unlike a word processor, where you can dive right in and start typing words and sentences, building a database table requires some prior knowledge of how databases work.

IN THIS CHAPTER

Examining the differences between databases, tables, records, fields, and values

Discovering why multiple tables are used in a database

Creating Access database objects

Designing a database system

On the CD-ROM

All the examples presented in this chapter can be found in the sample database `CollectibleMiniCars.accdb` on this book's CD-ROM. If you haven't yet copied this database to your hard drive, please do so now.

After you understand the basic concepts and terminology, the next important lesson to learn is good database design. Without a good design, you may have to constantly rework your tables, queries will be difficult to write, 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. Although the Collectible Mini Cars case study provides invented examples, the concepts illustrated by this simple application are not fictitious.

Some of this chapter's concepts are somewhat complex, especially to people new to Access or database development.

Cross-Reference

If your goal is to get right into Access, you might want to skip to Chapter 2 and read about building tables. If you're fairly familiar with Access but new to designing and creating tables, read the current chapter before starting to create tables.

The Database Terminology of Access

Before examining the table examples in this book, it's a good idea to have a firm understanding of the terminology used when working with databases — especially Access databases. Microsoft Access follows most, but not all, traditional database terminology. The terms *database, table, record, field,* and *value* indicate a hierarchy from largest to smallest. These same terms are used with virtually all database systems, so you should learn them well.

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. Some older database systems used the term *database* to describe individual tables. Current use of *database* applies to all elements of a database system.

Databases aren't only for computers. There are also manual databases; we sometimes 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. Users fill out paper forms for input, perhaps by using a keyboard to input information that is printed on forms. 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 a variety of different kinds of data, 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 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 Access applications easier to use.

A relational database management system (RDBMS), such as Access, stores data in *related* tables. For example, a table containing employee data (names and addresses) may be related to a table containing payroll information (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 2012?") from these related tables, with the answers displayed as onscreen forms and printed reports.

In fact, one of the fundamental differences between a relational database and a manual filing system is that, in a relational database system, data for a single individual person or item may be stored in separate tables. For example, in a patient management system, the patient's name, address, and other contact information is likely to be stored in a different table than the table holding patient treatments. In fact, the treatment table holds all treatment information for all patients, and a patient identifier (usually a number) is used to look up an individual patient's treatments in the treatment table.

In Access, a *database* is the overall container for the data and associated objects. It's 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 an Access database, the objects (tables, queries, and so on) in the database are presented for you to work with. You may open several copies of Access at the same time and simultaneously work with more than one database, if needed.

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

Cross-Reference

The `.adp` file format is a special database format used by Access to act as a front end to work with SQL Server data. Chapter 37 covers Access Data Projects in detail.

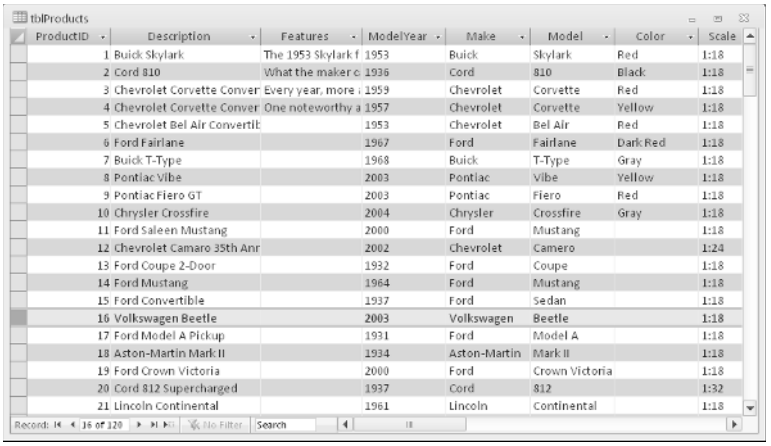
Tables

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 in the table is organized into rows and columns.

Figure 1.1 shows the Products table from the Collectible Mini Cars database application. The Products table is typical of the tables found in Access applications. Each row defines a single product. In Figure 1.1, the row containing information on the die-cast model of a 2003 Volkswagen Beetle is selected.

FIGURE 1.1

The Collectible Mini Cars products table



In the "A Five-Step Design Method" section, later in this chapter, I show you a successful technique for planning Access tables.

Cross-Reference

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's 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 have to 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 and how the entities relate to one another.

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 1.2 shows the Datasheet view of the customers table in the Collectible Mini Cars application. Although a datasheet and a spreadsheet are superficially similar, a datasheet is a very different type of object. Chapter 6 discusses Access datasheets, and the differences between datasheets and spreadsheets are explained in detail.

FIGURE 1.2

A table displayed as a datasheet

The customers table represents people who work with Collectible Mini Cars. 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 customer, while each column (or *field*) represents one type of information associated with customers.

For example, the top row in `tblCustomers` contains data describing Fun Zone, including the address, and phone number. Each bit of information describing Fun Zone is a field (`CompanyName`, `Address`, `Phone`, and so on). Fields are combined to form a record, and records are grouped to build the table. (Each row in a table constitutes a record.)

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 (`Company`) and the type of data in the field (`Text`). A field may include other properties as well. For example, the Address field's `Size` property tells Access the maximum number of characters allowed for the address.

Cross-Reference

You learn much more about fields and field properties in Chapter 2.

Records and fields

As Figure 1.2 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. In Figure 1.2, the fields are named `CustomerID`, `Company`, `Address`, `City`, `State`, and so on. 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.

Note

When working with Access, the term field is used to refer to an attribute stored in a record. In many other database systems, including SQL Server, column is the expression you'll hear most often in place of field. Field and column mean the same thing. The exact terminology used relies somewhat on the context of the database system underlying the table containing the record.

Values

At the intersection of a record and a field is a *value* — the actual data element. For example, Fun Zone, the company name in the first record, represents one data value. Certain rules (discussed in Chapters 2 and 3) govern how data is contained in an Access table. For example, in a properly designed database, the Fun Zone record occurs only once because each row in a table must be unique in some way. A table may contain more than one company named Fun Zone, but *something* about each company (such as the address) must be different. If rows in a table are not unique, Access has no way to distinguish between the duplicate rows, and the data can't be trusted or managed properly.

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 the customer's orders, without losing any data or pulling order records not owned by the customer.

Note

In the following sections (in fact, in the rest of this book), you'll see references to things such a "the customers table" or "the `tblCustomers` table." In the former, "the customers table" refers to the database table containing customer data, while "the `tblCustomers` table" (or just "`tblCustomers`") refers to the database table named `tblCustomers`. Different developers have different ways of naming things. For example, in my database, I may use `tblCustomers` as the name of the customers table, while another person might use Customers as the name for the same table. When working with a database it's very important to understand exactly which object is referenced by a name or description.

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 customers table (`tblCustomers`) and a sales table (`tblSales`), you must relate `tblCustomers` to `tblSales` in order to see all the sales records for a customer. If you had only one table, you would have to repeat the customer name and address for each sale record. Two tables let you look up information in `tblCustomers` for each sale by using the related fields `CustomerID` (in `tblCustomers`) and `CustomerID` (in `tblSales`). This way, when a customer changes address, for example, the address changes only in one record in `tblCustomers`. When sales information is onscreen, 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 properly segment data into multiple tables, you experience a significant reduction in design and work time. This process is known as *normalization.*

Cross-Reference

You can read about normalization in Chapter 3.

Later in this chapter, in the section titled "A Five-Step Design Process," you can work through a case study for Collectible Mini Cars that consists of five tables.

Why create multiple tables?

The prospect of creating multiple tables almost always intimidates beginning database users. Most often, beginners want to create one huge table that contains all the information they need — for example, a customer table with all the sales placed by the customer and the customer's name, address, and other information. After all, if you've been using Excel to store data so far, it may seem quite reasonable to take the same approach when building tables in Access.

A single large table for all customer information quickly becomes difficult to maintain. You have to input the customer information for every sale a customer makes (repeating the name and address information over and over again in every row). The same is true for the items purchased for each sale when the customer has purchased multiple items as part of a single purchase. This makes the system more inefficient and prone to data-entry mistakes. The information in the table is inefficiently stored — certain fields may not be needed for each sales record, and the table ends up with a lot of empty fields.

You want 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 fields that are only related 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.

Access Database Objects

If you're new to databases (or even if you're an experienced database user), you need to understand a few key concepts before starting to build Access databases. The Access database contains six 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.
- **Macro:** Automates tasks without programming.
- **Module:** Contains programming statements 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 permanent database object, a datasheet displays a table's content in a row-and-column format similar to a Microsoft Excel worksheet. A datasheet displays a table's information in a raw form, without transformations or filtering. The Datasheet view is the default mode for displaying all fields for all records. (Figures 1.1 and 1.2 earlier in this chapter are Datasheet views of Access tables.)

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

Be careful when you're making changes or allowing a user to modify data in Datasheet view. 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. Most forms and reports are based on queries that combine, filter, or sort data before it's displayed. Queries are often called from macros or VBA procedures to change, add, or delete database records.

An example of a query is when a person at the sales office tells the database, "Show me all customers, in alphabetical order by name, who are located in Massachusetts and bought something over the past six months." or "Show me all customers who bought Chevrolet car models within the past six months and display them sorted by customer name and then by sale date."

Instead of asking the question in English words, a person uses the query by example (QBE) method. When you enter instructions into the Query Designer window and run the query, the query translates the instructions into Structured Query Language (SQL) and retrieves the desired data.

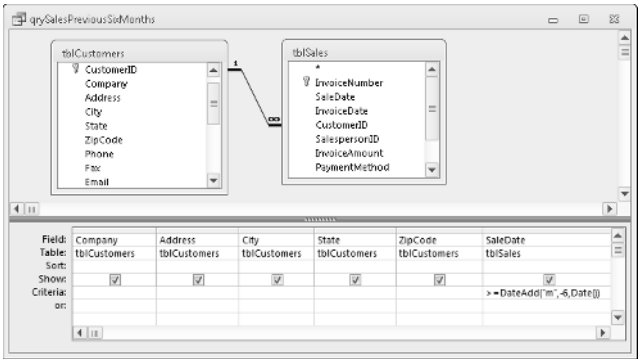
Cross-Reference

Chapter 4 discusses the Query Designer window and building queries.

In the first example, the query first combines data from `tblSales` and `tblCustomers`, using `CustomerID` as a link between the tables. Next, it retrieves the customer name, address, and any other data you want to see. Access then filters the records, selecting only those in which the sales date is within six months of the current date. The query sorts the resulting records by the customer's name. Finally, the resulting records are displayed as a datasheet. Figure 1.3 shows just such a query in Design view. In this figure, the user is requesting all customers from Connecticut who've placed an order in the previous six months.

FIGURE 1.3

A typical Access query



After you run a query, the resulting set of records may be used in a form that is displayed onscreen 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 can also check the validity of your data before it's added to the database table.

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

Read-only screens and forms are often used for inquiry purposes. These forms display certain fields within a table. Displaying some fields and not others means that you can limit a user's access to sensitive data while allowing access to other fields within the same table.

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 customers table) or may contain only the records meeting certain criteria, such as all customers living in Arizona. You do this by basing the report on a query that selects only the records needed by the report.

Reports often combine multiple tables to present complex relationships among different sets of data. An example is printing an invoice. The customers table provides 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 each product ordered. The report also calculates the sales totals and prints them in a specific format. 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.

Database objects

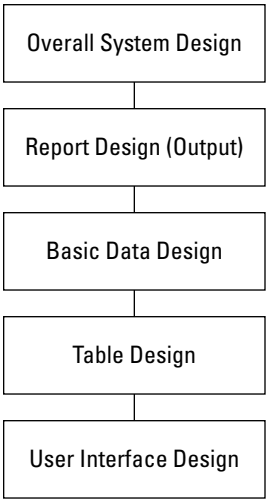
To create database objects, such as tables, forms, and reports, you first complete a series of *design* tasks. The better your design is, the better your application will be. The more you think through your design, the faster and more successfully you can complete any system. The design process is not 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 1.4 is a version of a common design method modified especially for use with Microsoft Access. This five-step method is a top-down approach, starting with the overall system design and ending with the forms design.

FIGURE 1.4

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, reports, macros, and simple VBA modules.

The time you spend on each step depends entirely on the circumstances of the database you're building. For example, sometimes 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 (products, customers, and transactions), remember that the focus of this chapter is how to move through each step. As you watch the Collectible Mini Cars database being designed, pay particular 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 user requirements before zeroing in on the details.

The five-step design method shown in Figure 1.4 helps you to create the system that you need, at an affordable price (measured in time or dollars). The Collectible Mini Cars database, for example, allows the client to sell items (vehicles and parts) to customers and supports the following tasks:

- Entering and maintaining customer information (name, address, and financial history)
- Entering and maintaining sales information (sales date, payment method, total amount, customer identity, and other fields)
- Entering and maintaining sales line-item information (details of items purchased)
- Viewing information from all the tables (sales, customers, sales line items, and payments)
- Asking all types of questions about the information in the database
- Producing a monthly invoice report
- Producing a customer sales history
- Producing mailing labels and mail-merge reports

These eight tasks have been described by the users. 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 users. This means that you need to sit down with them and learn how the existing process works. To accomplish this you must 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 any type of business, you may consider asking these questions:

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

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

A walkthrough of the existing process is also helpful to get a feel for the business. You may have to go back several times to observe the existing process and how the employees work.

As you prepare to complete the remaining steps, keep the client involved — let the users know what you're doing and ask for input on what to accomplish, making sure it's within the scope of the user's 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 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 Collectible Mini Cars database, the printed reports contain detailed and summarized versions of most of the data in the database.

After you've defined the Collectible Mini Cars' overall systems in terms of what must be accomplished, you can begin report design.

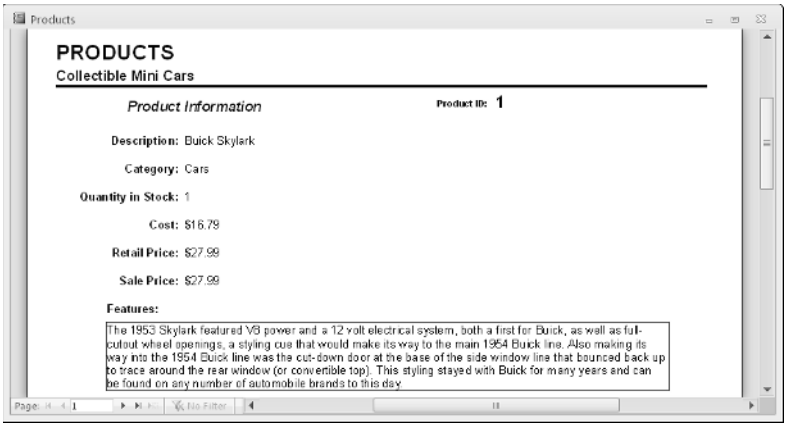
When you see the reports that you'll create in this section, you may wonder, "Which comes first — the chicken or the egg?" Does the report layout come first, or do you first determine the data items and text that make up the report? Actually, these items are considered at the same time.

It isn't important how you lay out the data 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 know exactly where they want each bit of data to be.

The reports in Figures 1.5 and 1.6 were created with two different purposes. The report in Figure 1.5 displays information about the Collectible Mini Cars products while the report in Figure 1.6 is an invoice with billing and customer information. The design and layout of each report is driven by the report's purpose and the data it contains.

FIGURE 1.5

A product information report

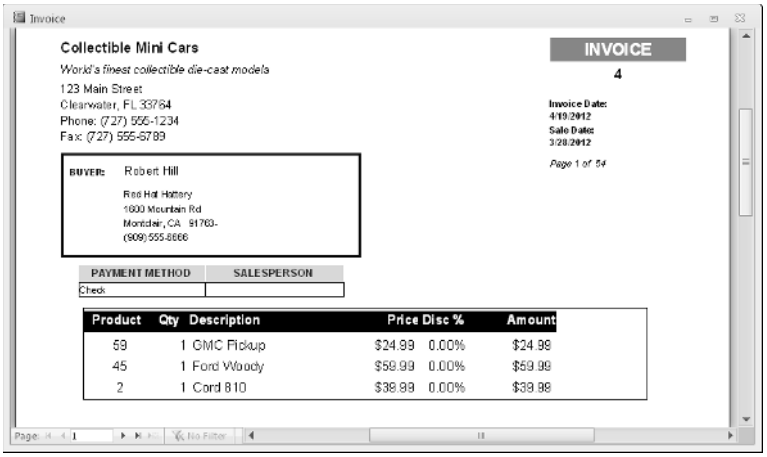


Cross-Reference

You can read more about the reports for the Collectible Mini Cars database in this book's introduction and in Chapters 9 and 20.

FIGURE 1.6

A sales invoice report containing sales information



Step 3: Data design

The next step in the design phase is to take an inventory of all the information 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 separate the data items into a logical arrangement. Later, these data items are grouped into table structures and then mapped onto data-entry screens (forms). You should enter customer data, for example, as part of a customers table process, not as part of a sales entry.

Customer information

First, look at each report you've roughed out for your database. For the Collectible Mini Cars database, 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

Customers Report	Invoice Report
Customer Name	Customer Name
Street	Street
City	City
State	State

Customers Report	Invoice Report
ZIP Code	ZIP Code
Phone Numbers	Phone Number
E-Mail Address	
Web Site Information	
Discount Rate	
Customer Since	
Last Sales Date	
Sales Tax Rate	
Credit Information (four fields)	

As you can see by comparing the type of customer information needed for each report, there are many common fields. Most of the customer data fields 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. 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.

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 sales.

TABLE 1.2

Sales Data Items Found in the Reports

Invoice Report	Line Item Data
Invoice Number	
Sales Date	
Invoice Date	
Payment Method	
Salesperson	
Discount (overall for sale)	
Tax Location	
Tax Rate	

continued

TABLE 1.2 *(continued)*

Invoice Report	Line Item Data
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
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 few items (fields) are repeating (for example, the `Product Purchased`, `Quantity 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. Many sales have more than one purchased item. Also, each invoice may include partial payments, and it's possible that this payment information will have multiple lines of payment information, so these repeating items can be put into their own grouping.

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 1.6, you can see that the data from Table 1.2 doesn't list the calculated field amount. The amount is dynamically calculated as the report prints, rather than storing the value in the database.

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 — calculated data can be easily created and displayed in a form or report.

Cross-Reference

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'll need to add others later (for various reasons), although certain fields won't appear in any table.

It's important to understand that you don't need to add every little bit of data into the database's tables. For example, users may want 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's 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's 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 look at the customer information and combine all its different fields to create a single 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

Customer Data	Invoice Data	Line Items
Customer Company Name	Invoice Number	Product Purchased
Street	Sales Date	Quantity Purchased
City	Invoice Date	Description of Item Purchased
State	Payment Method	Price of Item
ZIP Code		Discount for Each Item
Phone Numbers (two fields)	Discount (overall for this sale)	Taxable?
E-Mail Address	Tax Rate	
Web Site	Payment Type (multiple lines)	
	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 Collectible Mini Cars, but you have much more to do.

As you learn more about how to perform a data design, you also learn that the customer data must be split into two groups. Some of these items are used only once for each customer, while other items may have multiple entries. An example is the Sales column — the payment information can have multiple lines of information.

You need 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, I've already broken the data into three categories: customers, 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 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-Reference

I cover creating and understanding relationships and the normalization process in Chapter 3.

Database table relationships require a unique field in both tables involved in a relationship. A unique identifier in each table helps the database engine to properly join and extract related data.

Only the sales table has a unique identifier (`InvoiceNumber`), which means that you need to add at least one field to each of the other tables to serve as the link to other tables. For example, adding a `CustomerID` field to `tblCustomers`, adding the same field to the invoice table, and establishing a relationship between the tables through `CustomerID` in each table. The database engine uses the relationship between customers and invoices to connect customers with their invoices. Relationships between tables is done through *key* fields.

Cross-Reference

Creating relationships is explained in Chapter 3.

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 the *primary key*. The corresponding field in a related table is the *foreign key*. In our example, `CustomerID` in `tblCustomers` is a primary key, while `CustomerID` in `tblInvoices` is a foreign key.

Let's assume a certain record in `tblCustomers` has 12 in its `CustomerID` field. Any records in Invoices with 12 as its `CustomerID` is "owned" by customer 12.

Cross-Reference

As you'll see in Chapters 2 and 3, special rules apply to choosing and managing keys. The notion of primary and foreign keys is the single most important concept behind relational databases.

TABLE 1.4

Tables with Keys

Customers Data	Invoice Data	Line Items Data	Sales Payment Data
CustomerID	InvoiceID	InvoiceID	InvoiceID
Customer Name	CustomerID	Line Number	Payment Type
Street	Invoice Number	Product Purchased	Payment Date
City	Sales Date	Quantity Purchased	Payment Amount
State	Invoice Date	Description of Item Purchased	Credit Card Number
ZIP Code	Payment Method	Price of Item	Expiration Date
Phone Numbers (two fields)	Salesperson	Discount for Each Item	
E-Mail Address			
Web Site Information			
Discount Rate			
Customer Since			
Last Sales Date			
Sales Tax Rate	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 `CustomerID` in both the customers table (where it's the primary key) and the Invoice table (where it's a foreign key).

You've 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've also created an additional table to hold the sales payment data. Normally, payment details (such as the credit card number) are not part of a sales invoice.

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 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 customers, 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 customer's phone number appears in every row containing that customer's data, multiple updates must be made when the phone number changes.

Step 5: Form design

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. Generally speaking, your Access screens should look a lot like the forms used in a manual system.

When you're designing forms, you need to place three types of objects onscreen:

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

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. Other types of controls available with Access include list boxes, combo boxes, option buttons, toggle buttons, and option groups.

Cross-Reference

Chapter 7 covers the various types of form controls available in Access.

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

- **Clusters:** Contains several different types of controls
- **Sales:** Combines data from multiple tables
- **Products:** Adds products to the Collectible Mini Cars database

You'll encounter each of these forms as you read through the following chapters. Although Collectible Mini Cars is just one small example of an Access database application, the principles you learn building the Collectible Mini Cars 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 can't 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 reentering 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.