CHAPTER 1

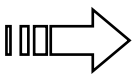
An Introduction to Database Development

IN THIS CHAPTER

- Examining the differences between databases, tables, records, fields, and values
- Discovering why multiple tables are used in a database
- Exploring Access database objects
- Designing a database system

Database development is unlike most other ways you work with computers. Unlike Microsoft Word or Excel, where the approach to working with the application is relatively intuitive, good database development requires prior knowledge. You have to learn a handful of fundamentals, including database terminology, basic database concepts, and database best practices.

Throughout this chapter, we cover the fundamentals of database development.



If your goal is to get right into Access, you might want to skip to Chapter 2.

The Database Terminology of Access

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.

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.

NOTE

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 manual database systems, you typically have in and 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's 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 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 from 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.

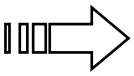
In Access, a *database* is the overall container for the data and associated objects. It's more than the collection of tables, however — a database includes many types of objects, including queries, forms, reports, macros, and code modules.

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, MDB, MDE, or ADP.

Tables

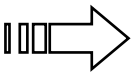
A table is just a container for raw information (called *data*), similar to a folder in a manual filing system. Each table in an Access database contains information about a single entity, such as a person or product, and the data in the table is organized into rows and columns.



In Chapters 3 and 4, 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 that your applications perform well while protecting the integrity of the data contained within your tables.

In Access a table is an 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 can 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"). Although a datasheet and a spreadsheet are superficially similar, a datasheet is a very different type of object.



Chapter 5 discusses Access datasheets and the differences between datasheets and spreadsheets. You can find much more about fields and field properties in Chapter 3.

Records and fields

A 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.

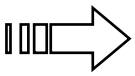
Each column is a field that includes many properties that specify the type of data contained within the field, and how Access should handle the field's data. These properties include the name of the field (Company) and the type of data in the field (Text). A field may include other properties as well. For example, the Address field's Size property tells Access the maximum number of characters allowed for the address.

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 Microsoft SQL Server, *column* is the expression you'll hear most often in place of *field*. *Field* and *column* mean the same thing. The 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, if you have a field called Company, a company name entered into that field would represent one data value. Certain rules govern how data is contained in an Access table.



See Chapters 3 and 4 for more on these rules.

Relational Databases

Access is a relational database management 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.

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 Customer table and a Sales table, you can relate the two tables using a common field between them. In this case, Customer Number would be a good field to have in both tables. This will allow you to see sales in the Sales table where the Customer Number matches the Customer table.

The benefit of this model is that you don't have to repeat key attributes about a customer (like customer name, address, city, state, zip) each time you add a new record to the Sales table. All you need is the customer number. When a customer changes address, for example, the address changes only in one record in the Customers table.

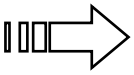
Why Create Multiple Tables?

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 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 a 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 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.

Separating data into multiple tables within a database makes a 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.*



You can read about normalization in Chapter 4.

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 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 an 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.

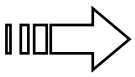
You can scroll through the datasheet using the directional keys on your keyboard. You can also display related records in other tables while in a datasheet. In addition, you can make changes to the displayed data.

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 plain English, 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.



Chapter 8 discusses the Query Designer window and building queries.

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 be enhanced with data validation rules or VBA code to 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 into 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 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 allows for an extraordinary amount of flexibility when creating reports. For instance, you can configure a report to list all records in a given table (such as a Customers table) or you can have the report 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

The five design steps described in this section provide a solid 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.

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.

For example your users may ask for a database that 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

When reviewing these eight tasks, you may need to consider other peripheral tasks that weren't mentioned by the user. Before you jump into designing, sit down 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.

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 might ask 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, they're often the best way to gather important information about a database's requirements.

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 to identify exactly where they want each bit of data to be.

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.

For example, you can start with all the customer data you'll need for each report, 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
ZIP Code	ZIP Code
Phone Numbers	Phone Numbers
E-Mail Address	
Web Address	
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 field names are the same, you can easily make sure that you have all the data items. Although locating items easily isn't critical for this small database, it becomes very important when you have to deal with large tables containing many fields.

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	Product Purchased
Sales Date	Quantity Purchased
Invoice Date	Description of Item Purchased

Invoice Report	Line Item Data
Payment Method	Price of Item
Salesperson	Discount for Each Item
Discount (overall for sale)	
Tax Location	
Tax Rate	
Product Purchased (multiple lines)	
Quantity Purchased (multiple lines)	
Description of Item Purchased (multiple lines)	
Price of Item (multiple lines)	
Discount for each item (multiple lines)	
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.

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.

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 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 (overall for this sale)	Discount for Each Item
Phone Numbers (two fields)	Tax Rate	Taxable?
E-Mail Address	Payment Type (multiple lines)	
Web Address	Payment Date (multiple lines)	
Discount Rate		
Customer Since	Payment Amount (multiple lines)	
Last Sales Date	Credit Card Number (multiple lines)	
Sales Tax Rate	Expiration Date (multiple lines)	
Credit Information (four fields)		

Consolidating and comparing data is a good way to start creating the individual table, 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.

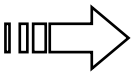
You need to further break these types of information into their own columns, thus separating all related types of items into their own columns — an example of the *normalization* part of the design process. For example, one customer can have multiple contacts with the company or make multiple payments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

Keep in mind that one customer may have multiple invoices, and each invoice may have multiple line items on it. The invoice-data 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 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.

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 (Invoice Number), 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 Customer ID field to the Customer table, adding the same field to the Invoice table, and establishing a relationship between the tables through Customer ID 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.



We cover creating and understanding relationships and the normalization process in Chapter 4.

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, Customer ID in the Customers table is a primary key, while Customer ID in the Invoices table is a foreign key.

Let's assume a certain record in the Customers table has 12 in its Customer ID field. Any record in the Invoices table with 12 as its Customer ID is "owned" by customer 12.

TABLE 1.4 Tables with Keys

Customers Data	Invoice Data	Line Items Data	Sales Payment Data
Customer ID	Invoice ID	Invoice ID	Invoice ID
Customer Name	Customer ID	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	Tax Rate		
Web Address			
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 Customer ID 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 three core 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 fact 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 place to briefly describe the art of database normalization. You'll read the details

of normalization in Chapter 4, 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

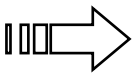
After you've created the data and established table relationships, it's time to design your forms. *Forms* are made up of the fields that can be entered or viewed in Edit mode. Generally speaking, your Access screens should look a lot like the forms used in a manual system.

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.



Starting with Chapter 17, we cover in detail the topic of creating forms.