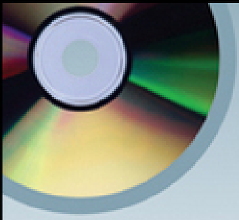


Ready-to-load database models on CD-ROM

MVP
Microsoft®
Most Valuable
Professional

Microsoft® Access® Small Business Solutions

State-of-the-Art Database Models for Sales, Marketing, Customer Management, and More Key Business Activities

Teresa Hennig, Truitt Bradly, Larry Linson, Leigh Purvis, and Brent Spaulding
Foreword by Luke Chung

CHAPTER

4

People, Organizations, Addresses

What information do you need that spans all, or almost all, of the functions of your business or organization? Of course, that is information about people. Businesses and commercial organizations need to track, relate, and report information about employees, customers, suppliers, and vendors. Noncommercial organizations deal with information about members, sponsors, donors, and others.

So it is quite understandable that people and organizations are a key component of many database applications, regardless of the main business objectives. Managing the data can be quite challenging because the data often has multiple uses and can easily become fragmented or copied into multiple locations, and the more places that data is stored, the more difficult it is to update and ensure that it is consistent and accurate.

To complicate the matter even more, you will frequently discover additional kinds of data that you want to store. For example, in most business functions you don't need to know about the family of a person, but to enhance customer, employee, or member communications, that is useful information. You have several options for handling how you add this data to existing tables. An easy way to consistently use field properties is to copy an existing field—say, FavoriteColor—and rename it to SpouseName. Alternatively, add a new field named SpouseName, or you might add a separate FamilyMember table that includes spouses, children, significant others, and whatever other relations that you choose.

Consider the many places and tools you personally use to store information about people that you know. What would it take to have a central storage system that could manage all of the data relevant to each contact and to make it easy to look up the information by filtering on a characteristic or field? This simple exercise will help you visualize what might be involved when you set about designing a database for this purpose.

If you use an e-mail program, open it and look at the contact information. Although you will be looking at forms, not the underlying tables, you can deduce a great deal about the table design by considering the information and the ways that it can be entered and displayed. It will soon be obvious that most people have multiple accounts, and you may have part of their data stored on your computer, and other parts of their data online.

If you use more than one program for tracking personal information, you can see that they each store some of the same information, but that each program has some custom options. The list of similarities and differences is too extensive to cover here, but they indicate how important it is for you to consider not only what data you need to store, but also how it will be used.

You may be familiar with computer applications for personal information management (PIM) and for handling address book (AB) or contact information, so we use them as examples in this chapter. You will learn how to arrange the subjects into tables, and the details into fields that make up the records of those tables. We will show you how and when to create additional tables to incorporate related information. The tables are designed to store data that is common to a variety of organizational structures.

How This Chapter Is Organized

Two database structures are presented in this chapter. After looking at the questions to ask and the factors to consider, you can select the structure most appropriate for your needs, based on the characteristics of the real-world environment that you are modeling and how you need to use the data. Our focus in this chapter is to help you determine and create a solid foundation for storing, managing, and using data pertaining to people and organizations, but the examples we provide are not intended to represent the core of a full enterprise-level customer relationship management system. We definitely do not suggest that you use these sample tables to create your own PIM or address book database. As mentioned, there are many packages, both commercial and open source, for those purposes that cover all but the most unique or unusual needs.

Although the examples provided can be used as the foundation for a solution strictly focused on people and organizations, we assume that you will use these or similar tables in conjunction with other examples in the book. You might incorporate them as reusable parts of full business applications or you could share one set of people and organization information between your other business database applications. We hope you will find this chapter to be a useful tool for learning how to gather requirements for the data you include in your database applications, as well for designing those databases.

However, that is not to say that our examples would not support a more robust application. If you are seeking a contact management solution but are finding that the commercial packages do not fit your needs, then our examples will be of great benefit. By following the processes in this chapter, you will soon have the skills to customize these tables for use in conjunction with such an application, and we will cheer you on. In fact, even if you already have a PIM, AB, or contact management application, creating such an application might be a useful learning exercise before you leap directly into work on a database application for one of your vital business functions.

In this chapter, you learn how to store data specific to individual people or to particular organizations, along with other data associated with both people and organizations. You will review and consider options for using related tables to store information such as e-mail addresses and telephone numbers, and learn how to use a junction table to manage many-to-many relationships between the people and the organizations.

First you'll examine a simplified design for people and organization information that will work for many situations. Then you'll look at a slightly more complex design that provides additional flexibility. Of course, everything comes at a price, but don't be intimidated. In this case the price you pay will be the additional time and effort needed to both create the structure and then design the user interface, but you may be surprised to find that it is not burdensome.

In addition to explaining the models and how they correspond to various business functions, this chapter also includes some bonus tables. You will find a table for looking up the correct postal abbreviations for U.S. states, territories, and military ''States'' and Canadian provinces and territories. In the example database for this chapter on the CD that accompanies the book, this table is pre-populated with names and abbreviations. You may need to do nothing more than import it into your own application. It can be an invaluable tool for creating mailing and marketing solutions.

We've also provided a table for month, month abbreviation, and month number. Date manipulations are built into the VBA programming language used by Access, and into Access's expressions, but in many cases, especially when creating queries, you may find it simpler and easier to use this table instead of using date and time functions to perform calculations.

Now that you have a general idea of what you will be doing, we'll spend a little time reviewing some information about some of the fields.

Basic Field Information

To give you a start on creating the tables, we thought that you'd find it helpful to know a little about the fields that we use in nearly every table. So, we've compiled a series of notes that you can easily refer back to.

NOTE **Item, subject, and detail are not, in general use, precise definitions. In this chapter, however, when we use the words without additional qualification they have the following meanings: *item* refers to a list of things to consider for inclusion in tables, *subject* refers to the topic of a table, and *detail* refers to something that will likely become a field in a table.**

STANDARD FIELDS USED IN MOST TABLES

Primary and Natural Keys

Each table in this chapter, as with most tables in this book, uses an Access AutoNumber data type for the primary key. An AutoNumber field is a surrogate key, an arbitrarily defined value that will uniquely identify the record. As you'll recall from Chapters 2 and 3, using a single-field primary key makes it easier to create, understand and work with the relationships between tables and the joins used in queries.

Using an AutoNumber for the primary key does not preclude the table from having a different field (or combination of fields) that represents a natural key—real world data that uniquely identifies the record. Natural keys typically have the field properties of Required and Indexed with no duplicates allowed. This ensures that a value is stored in the field and that two or more records cannot have the same value.

Multi-Field and Unique Indexes

Indexes can include more than one field. You can use a multi-field index to ensure that any combination of field values does not occur more than one time by setting the index property to make it unique.

Junction tables often use a multi-field index on the two foreign keys. You can use the index property to ensure that the combination of these two fields will always be unique. In most cases, you also need to use the field property to make the field Required, so that each Record has to store a value.

In other cases, a single field (such as the main subject of the table) can be indexed with no duplicates allowed to insure that the content of the field is not duplicated. You will typically also make this a required field. Single field indexes can be created using either the field property or the index property.

Appendixes A and B provide an extensive discussion on field properties and relationships. A brief review of those two appendixes will give you a strong foundation for working on the examples throughout the book.

Audit Trail

Each data table may also contain fields that indicate who updated the record and when the record was updated. It may seem overkill to include this, but there may often be legal requirements or business rules that call for some form of audit trails. Before omitting these fields, you should make certain that your accountant and legal counsel concur.

In most Access database applications, you will have to use code or macros to collect audit trail data because the database engine does not provide this capability—at least not to the extent desired. However, many server databases, such as Microsoft SQL Server, provide support for logging all changes and thereby creating an audit trail.

Although these two fields do not provide a complete audit trail, they can identify when a record was created and if it is active. By setting the default value of the field to Now(), a Date/Time field can be used to automatically capture when a record was initially created.

You can also use a Yes/No field or a Date/Time field to indicate if a record is active or if it has been flagged as no longer used. Flagging a record is typically preferable to deleting the record, which may be illegal or cause data continuity problems.

Note Fields

Finally, transaction tables will typically have a "Note" field for optional comments about the record. Users may find it helpful to record a note if they're interrupted while entering data, and developers may use them when creating or testing the application. They can also store information that seems pertinent to the actual content of the record-something not specified in processing but perhaps useful to someone examining the record in detail.

No additional definition is required for these standard fields, so they will not be discussed each time they occur in the chapter.

MEMO FIELDS

You often hear advice against using Memo fields in transaction tables, primarily because the fields have been "subject to data corruption." If a table contains a Memo field that becomes corrupted, it is possible to lose all the data in that record, not just the data in the Memo field. Fortunately, the frequency of such corruptions has diminished significantly as network technology has matured and we are less likely to experience momentary outages and system interruptions that lead to data corruption.

Continued

MEMO FIELDS *(continued)*

An easy alternative is to move the Memo field to a related table, using either a one-to-one or many-to-one relationship. Chapter 7 provides a detailed discussion of how this can be implemented.

DATA RETENTION

You should exercise care and consult with your attorney on the nature of the information you keep on file. You are required to store certain information, but it is also prohibited by law and regulations to keep other information. Preserving information that is not required and that is not clearly useful to your business can, in fact, work to your disadvantage in future legal proceedings. Having too many records can also place an undue burden on systems and people trying to store and retrieve data.

Now that you know what we'll be covering and some of the basic structure that we'll be working with, it's time to get started with the project. The first step is to determine what data your database needs to store.

People and Organization Information Processing and Storage

Now we move on to dealing with the business functions and structuring the data to address them.

Personal Information Manager (PIM) or Address Book (AB) Basics

The basic functionality for a PIM or AB has the same or similar requirements as you'll find in other business functions that need to use people and organization data. Those requirements are as follows:

- You need to be able to view, add, change, and delete the information you decide you maintain about the people and organizations.
- You need to be able to view, add, change, and delete information that is related to the people and organizations but not stored with the people and organization data. Examples include telephone numbers, e-mail addresses, and in some cases postal address information.
- You need to be able to search by either person or organization to retrieve information to be displayed or printed as a report.

- You want to be able to search or group by categories of person (minor, adult, customer, vendor, etc.) or type and category of organization when viewing or reporting information.
- You want to be able to look up reference information used, such as the state and province postal abbreviations, categories for people, types of and categories for organizations, types of phone numbers, and types of e-mail accounts. Some of this reference information, such as state and province postal abbreviations, changes so infrequently that you won't want to expend a great deal of effort making it easy for the general user to view, add, change, and delete it. Other information, such as categories for people and organizations, should be made relatively simple for your application administrator to view, add, change, and delete. For example, with an age range category such as pre-school, children, teens, adult, and elderly, it might be desirable to change ''elderly'' to ''senior.''
- You should consider whether there is any detail information that may change frequently and in a manner that would warrant a feature that allows users to perform a mass change. One example might be governmental categorizations, such as the type of business number and description used by the U.S. Census Bureau and Internal Revenue Service. This particular information isn't included in our examples, but it might be useful in your database(s) to support the business functions particular to your organization.

Questions to Ask

Collecting and storing information about people and organizations (and often the relationships between them) is a common need, but the amount of that information useful to a given business function varies. To determine how much and what type of information needs to be maintained, you need to ask key people associated with the business function (the sponsor or requester of the database application, knowledgeable users, and perhaps yourself) several leading questions such as:

- ***Do you need different information about people and organizations?*** Is information about both people and organizations required, or only one or the other? If both, are both handled similarly, as in the case where either a person or an organization could be a customer? Or is information for the business function primarily about people? If so, is there a need to know which organizations a person is affiliated with? If the information you need is primarily about organizations, is contact information needed for people in various roles within the organization?
- ***How Will the Person Information Be Used?*** How will the information about people be used for this business function? For example, if the business function simply requires contact information, you can choose

items of information (called *fields* in an Access table) of a professional nature; but if the function has to do with maintaining a relationship with the person, you may choose more personal information about family, birthdays, or accomplishments. For more details about managing business relationships, see also Chapter 5, ''Customer Relationship Management,'' and Chapter B1, ''Knowledge: Intellectual Property, Structural Capital, and Intellectual Capital.''

- ***How will organization information be used?*** For this business function, how will the information about organizations be used? Different types of information may be needed for organizations that represent your vendors versus your customers, or you may want to keep certain information simply to be aware of the individual's association with an organization.
- ***Will people and organizations be treated the same?*** Do you make the same types of contacts with people as you do with organizations? For example, are your customers only people, only organizations, or can they be both? Are people included only in their role within an organization, or does the business function primarily relate to people?

  The answers to these questions will determine the structure of the tables that you need. If contact with an organization is to always be through the people who are affiliated with it, then most of the contact details will only be required to be associated with people. However, if you need to have complete sets of contact information for organizations as well as for people, then you will also need to establish direct links between the contact details and the organization.
- ***Are details of related tables frequently shared?*** If multiple people, organizations, or a combination thereof might have the same telephone number, e-mail address, or physical address, then the contact details can be shared so that they only need to be maintained in one location. We explain how to do that by using a *many-to-many relationship* between the tables.

Items to Store

You should first consider the subjects that you want to store, retrieve, and manipulate information about—these subjects will serve as the basis for the tables you create for your database. For each subject, you will identify the details of interest. The details will become fields within the tables.

At this point, we can identify two key subjects: people and organizations. You may also think of related subjects, such as telephone numbers, e-mail addresses, delivery addresses, and other physical addresses. As you look into these, you'll likely think of using lists that provide state abbreviations, types of phone numbers, and even categories for contacts.

Once you have determined the key subjects, you need to identify all of the details that you want to record or track, commonly known as *requirements gathering*. This typically involves interviewing the key players and examining existing processes and reports. You can gain some valuable insight and tips from Appendix BB, ''Gathering Requirements.''

Looking at the universe of details that you need to manage leads to identifying the tables and how the information in each table will be integrated with other tables. In the following discussions, we italicize the items that we are considering as prospective tables or fields.

The Subject and Details of People

We'll start by identifying the details that you may need to record for people. In your business dealings with a person, one of the first requirements is for users to be able to identify the specific person—not just anyone named Larry. You may wish to consider including a person code if your organization assigns one—this might be an employee number or a customer number. You will probably find it useful to include a category such as friends, family, clients, and professional groups. Creating a category enables you to select and work with groups of people. For example, you might want to send holiday greetings to friends and family, or a newsletter to all of your clients.

You should also keep the person's name in whatever format is appropriate for your data. In English-speaking countries this would typically include separating the *first, middle, and last names* (even though some people have more than three). The separation is appropriate because in different scenarios, you may want to display or print the names individually or in a specific order, such as last name first. In most cases you will also want to keep address information, which is separated into the distinct parts we are familiar with—street, city, state, and country. Based on your scenario, you may also need to use other information, such as date of birth or date of first contact, last purchase date, client status, and even preferred method of contact.

The Subject and Details of Organizations

Just as with people, you must first be able to identify the organization. In your environment you might use an organization code—this could be a vendor number, a customer number, or another designation. You should also keep the organization's name, of course. You may also find it useful to include an organization type, as well as a category. This will enable you to split categories of organizations into subsets that share common characteristics.

Address information will again require several detail fields, and you will likely have several details based on your scenario, such as date of first contact with the organization. The list of details will grow as you review the activities involved to identify other information your business functions will require.

Other Subjects and Details

More often than not, your contact with either a person or an organization will be via e-mail or phone, rather than physical address. Therefore, these are good candidates for additional subjects, with the detail being a way to link the e-mail address or telephone number to a specific person or organization.

In addition to the actual data, another type of subject is the reference or lookup information. This type of information is provided in lists that can make it easier to enter, store, and retrieve data, and make data entry more efficient by limiting user options. We will store these lists in lookup tables.

In this example, we've already mentioned several items that can be useful as lookup tables, including a category of person and the type and category of organization. You may also want to use two other lookup tables, one for postal code abbreviations and one for country name.

Arranging the Data into Tables

In discussing the subjects, we've identified some key topics and how they relate to each other, as illustrated in Figure 4-1. The topics are the basis for creating the tables, and the discussion of details should help you decide which fields to include.

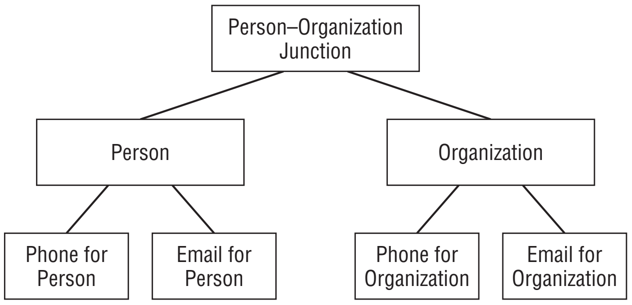


Figure 4-1: Overview of the Subjects

The subject of this database application is people and organizations, so it is logical for you to start by creating a table to contain information about people, which we'll name tblPerson, and another about organizations, named tblOrganization.

The Person Table

Information about the subject ''people'' can be collected in a table where each record represents a person and the information about that person. In our example, we name this tblPerson and translate the details of our earlier discussion into fields. The detail ''person code'' will become the field PersonCode. We will handle category by creating a field PersonCategoryID, which will contain the value of the primary key of the selected category from tblPersonCategory. (Primary keys are discussed in the ''Standard Fields Used in Most Tables'' sidebar earlier in this chapter.)

The details first name, middle name, and last name will translate directly to three separate fields for the name of the Person: NameLast, NameMiddle, and NameFirst. Each field may contain one or more words, as a first name can be Bobbie Jo, or Billy Bob. It may seem tempting, especially in a simple model, to combine the names and only store and use a full name. *Resist that temptation*. Sooner or later, you will need to sort the names by last name, or extract the first name for use in the salutation of an e-mail. The pain and difficulty involved in deconstructing a full name into its component parts outweighs any benefits you feel you will gain by having only one name field. Besides, you can easily combine, or concatenate, the name fields as needed. The Access expression to create a full name from the three components is simple (and can also be used in Access's VBA code programming language):

```
NameFull = [NameFirst] & (" " + [NameMiddle]) & " " & [NameLast]
```

In the simple model for people and information (described later in this chapter), you will also store address information in several fields. We've described that in detail in the documentation for the tables in the example database. If you decide you need the details ''date of birth'' and ''date of first contact,'' you might name them PersonBirthDate and PersonDateOfFirst-Contact. You will find detailed descriptions of the fields in the actual table documentation for our sample database, which follows in the sections headed ''The Simple Model'' and ''The Complex Model,'' with descriptions of fields and screen captures of table design.

The Organization Table

Because you have separated people from organizations, you also need to create a table with records to represent each organization. An appropriate name for the table is tblOrganization. The detail ''organization code'' was intentionally omitted from our example, although we included a ''person code.'' We did this to illustrate that an organization might use one but not the other. Should you need to include an identifier, an appropriate name for the field might be OrganizationCode. In the interest of readability and space, you could decide to abbreviate Organization as Org in all field names, but we'll use the complete word to provide continuity between the narrative and the sample file.

You can translate the detail ''name'' to a single field named Organization-Name. You can create a field OrganizationTypeID as a foreign key to the lookup table tblOrganizationType. In the example, the name of the category would also be selected from the tblOrganizationType, but since we are only providing one level of detail, we did not include a field for category. Having fields for both *types* and *categories,* where one is a subset of the other, would enable you to treat subsets of the organizations in your database in a hierarchy of groups with common characteristics.

Details about address information will be handled in fields in the same manner as we described for the People table. To include the detail ''date of first contact,'' create a field named OrganizationDateOfFirstContact.

The Organization-Person Table

A person may be involved with multiple organizations, and an organization may be related to multiple people; this is a many-to-many relationship and requires a junction (or intersection) table to support it. Because this is ''infrastructure,'' it was not discussed earlier as a subject or detail. The specific fields for this junction table will be explained when we start creating the tables. But as is common to all junction tables, it will contain a foreign key relating to a record in tblOrganization and a foreign key relating to a record in tblPerson.

The Phone and Email Tables

In the simple model of people and organizations, there is a Phone table for people, tblPhoneForPerson, and a Phone table for organizations, tblPhoneForOrg. Similarly, there are separate tables to store the e-mail addresses: tblEMailForPerson and tblEMailForOrg. This structure was chosen to simplify diagramming and to make it easier to understand the structure. In each of the Phone and Email tables of the simple model of person and organization data, there is a foreign key field that relates the record to the specific person in the Person table or the organization in the Organization table. In the complex model, Person and Organization are related to Phone and EMail using a junction table.

There are several viable options for managing contact information. To give you experience with other approaches, we will treat these tables differently in the complex model of people and organization information, which will be discussed later.

Reference or Lookup Tables

You can create a lookup table so that users can select data from a list of options. Lookup tables typically have one field that contains the unique record identifier for that table (the ID), and another field containing the information being looked up. For our example, you should create lookup tables for the category of person (tblCategoryOfPerson), and both the type and the category of organization (tblOrganizationType and tblOrganizationCategory). Other lookup information and tables which you might want to use are postal code abbreviation (tblStatesRef), and country name or abbreviation (tblCountryRef).

Determining the Level of Detail

To provide you with design options and learning opportunities, we consider two views or models based on the real world scenarios of people and organizations. The simple model serves well for many database applications, so that is our primary example. This model may be all that is needed for applications in which people and organizations are not the primary focus of the business function. It also serves well as a database to provide simple contact functions.

Later in this chapter we also discuss features useful where people and organizations are a primary focus of the business function. We explain and provide features to handle circumstances in which the additional complexity would be warranted.

The Simple Model

Addresses, in the simple model, are kept in the records for individual people and organizations. This works well when you only have one address per person or organization, which is a very common situation. Most people and organizations will have a telephone number or e-mail address, but they will frequently have multiple phone numbers or e-mail addresses, not just one. In the simple model, the phone number and e-mail address tables each have a *one-to-many* relationship with people and organizations.

In addition, although we are providing a simple model, we also include an example of a more complex relationship that is also quite common. Because one person can be associated with multiple organizations, and vice-versa, you need a way to represent this in your tables. This is known as a *many-to-many relationship*, which is resolved by creating a junction table.

The Person Table

The table tblPerson, shown in Figure 4-2, contains the standard fields, plus those listed below. Most fields are self-explanatory so there are minimal comments. For the purpose of clarity, we have included a table identifier in fields that may also appear in other tables. This is not a requirement but avoids confusion down the road.

PersonCode is a Text field containing the assigned code for this person (if used in your organization or useful in your database application). If this needs to be a unique value, you should set the field properties to have this field indexed and not allow duplicates.

PersonCategoryID is a Long Integer field that is used as a foreign key to tblPersonCategory. This is called a foreign key because it stores the value of the primary key of the selected record.

NOTE You will typically find it easy to recognize foreign keys, even if you are not familiar with the database or the person who designed it. In the majority of cases, the foreign key will have the same name as the primary key from the original table, so it is derived from the name of the table and has a suffix of ID. In addition, they will typically be a Long Integer data type. You can learn more about relationships and key values in Appendix B, "Relationships, Joins, and Nulls."

NameLast is a Text field containing the person's last name.

NameMiddle is a Text field containing the person's middle name.

NameFirst is a Text field containing the person's first name.

PersonAddress1 is a Text field that stores the first line of the person's address.

PersonAddress2 is a Text field that stores a possible second line of the person's address.

PersonCity is a Text field containing the name of the city of the person's address.

PersonState is a Text field containing the state, province, territory, and military ''States'' postal abbreviations for the U.S. and Canada of the person's address. Note that even if you are using a lookup table (also known as a reference or domain table) for state abbreviations, it makes perfect sense to store the abbreviation itself, rather than the value of the primary key. This logic applies to future instances of State—and other similar fields. You can find more information about lookup tables in Chapter 2, ''Elements of a Microsoft Access Database,'' and in Appendix B, ''Relationships, Joins, and Nulls.''

If you choose to use the lookup table, and store just the abbreviation, you can still display the full name of the state when necessary by creating a join back to the table containing the state names and abbreviations. In that case, the abbreviations would need to be unique, and fortunately for us, they are.

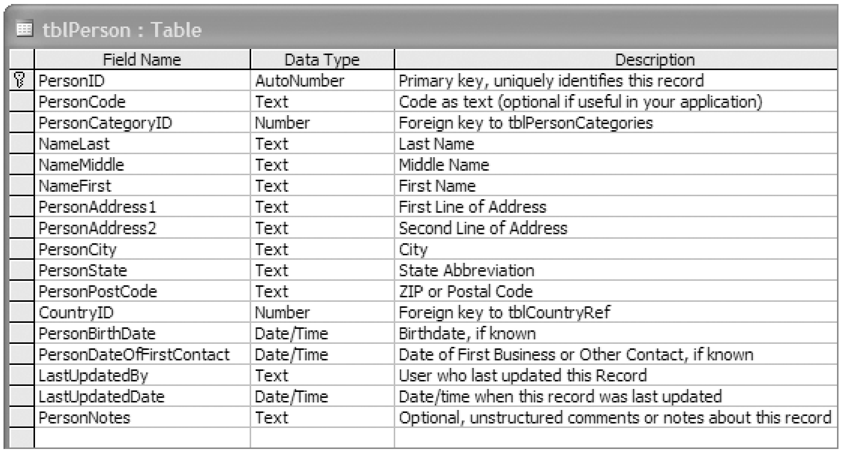
PersonPostCode is a Text field storing the zip or postal code of the person's address.

CountryID is a Long Integer field that is used as a foreign key to the Country Reference table. In most cases, this is done equally well by storing the abbreviation for the country, as we did with state. This address structure is useful in the United States and a good many other countries, but not everywhere; if it is not suitable in your location, you may need to modify the fields defining an address.

PersonBirthDate is a Date/Time field that records the person's birth date, if known.

PersonDateOfFirstContact is a Date/Time field that records the date (and possibly time) of the first business or other contact, if known.

Next, we create the tables that are required to store multiple values related to the same record, such as e-mail addresses and phone numbers.



tblPerson : Table

Field Name	Data Type	Description
PersonID	AutoNumber	Primary key, uniquely identifies this record
PersonCode	Text	Code as text (optional if useful in your application)
PersonCategoryID	Number	Foreign key to tblPersonCategories
NameLast	Text	Last Name
NameMiddle	Text	Middle Name
NameFirst	Text	First Name
PersonAddress1	Text	First Line of Address
PersonAddress2	Text	Second Line of Address
PersonCity	Text	City
PersonState	Text	State Abbreviation
PersonPostCode	Text	ZIP or Postal Code
CountryID	Number	Foreign key to tblCountryRef
PersonBirthDate	Date/Time	Birthdate, if known
PersonDateOfFirstContact	Date/Time	Date of First Business or Other Contact, if known
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
PersonNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-2: The Person table

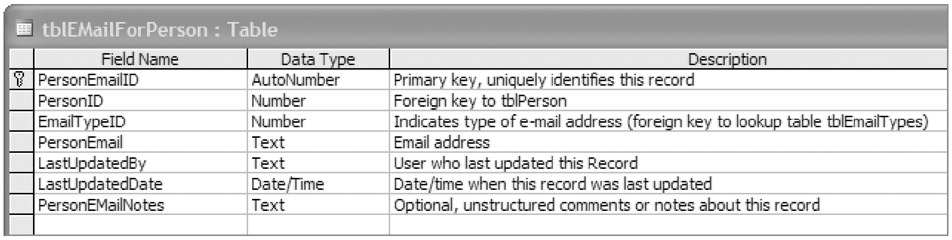
The Email For Person Table

The table tblEMailForPerson, shown in Figure 4-3, includes the following fields.

PersonID is a Long Integer field that is used to link to the person's record to which this e-mail is related, a foreign key to tblPerson.

EmailType is a Long Integer field that is used to indicate the type of e-mail address based on a selection from the lookup table, tblEmailType. This is called a foreign key because it stores the value of the primary key of the selected record.

Email is a Text field containing the person's e-mail address as text—for example, `benbrown@acmewidgets.com`.



tblEMailForPerson : Table

Field Name	Data Type	Description
PersonEmailID	AutoNumber	Primary key, uniquely identifies this record
PersonID	Number	Foreign key to tblPerson
EmailTypeID	Number	Indicates type of e-mail address (foreign key to lookup table tblEmailTypes)
PersonEmail	Text	Email address
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
PersonEMailNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-3: The Email For Person table

The Phone For Person Table

The table tblPhoneForPerson (shown in Figure 4-4) contains the fields required to store the phone numbers for individuals in tblPerson. In addition to the standard fields, you will need the fields described next.

PersonID is a Long Integer field that is used as a foreign key that identifies in tblPerson the record of the person who owns this phone number.

PersonPhoneType is a Text field that describes the type of phone: home, work, mobile, or pager. Alternatively, if you expect to be performing analysis using this field, and the consistency of the types would be beneficial, you can create a lookup table of types, and replace the Text field with a numeric foreign key to the lookup table.

PersonPhoneDisplayOrder is a Long Integer field that is used to specify the order in which to display the phone, if there are multiple phones for the related record.

PersonPhoneCountryCode is a Text field that contains the country code (all formats), a variable number of digits depending on the region. A value of 1 is the code for the U. S., Canada, and the Caribbean.

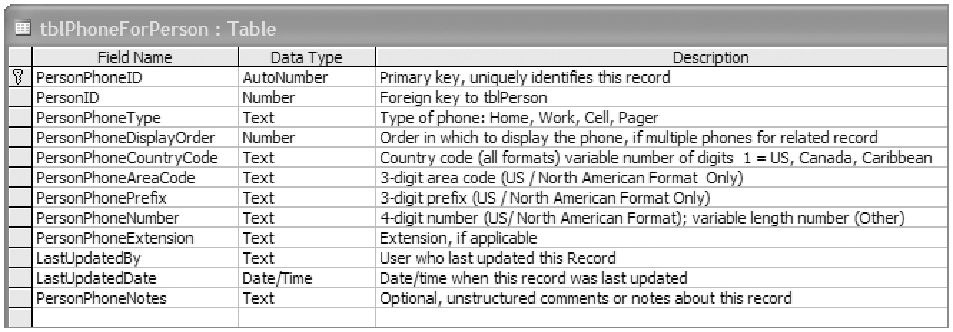
PersonPhoneAreaCode is a Text field that contains the three-digit area code (U.S./North American format only).

PersonPhonePrefix is a Text field that contains the three-digit prefix (U.S./North American format only).

PersonPhoneNumber is a Text field that contains the four-digit number (U.S./North American format only) or a variable-length number (other phone number formats).

The separate fields making up the phone number may be useful if you later use this information for detailed demographic analysis. For many applications, however, the area code, prefix, and phone number can be stored as one Text field instead of three. Whichever approach you choose, we recommend that you apply it consistently within an application.

PersonPhoneExtension is a Text field that contains the extension, if applicable.


tblPhoneForPerson : Table

Field Name	Data Type	Description
PersonPhoneID	AutoNumber	Primary key, uniquely identifies this record
PersonID	Number	Foreign key to tblPerson
PersonPhoneType	Text	Type of phone: Home, Work, Cell, Pager
PersonPhoneDisplayOrder	Number	Order in which to display the phone, if multiple phones for related record
PersonPhoneCountryCode	Text	Country code (all formats) variable number of digits 1 = US, Canada, Caribbean
PersonPhoneAreaCode	Text	3-digit area code (US / North American Format Only)
PersonPhonePrefix	Text	3-digit prefix (US / North American Format Only)
PersonPhoneNumber	Text	4-digit number (US/ North American Format); variable length number (Other)
PersonPhoneExtension	Text	Extension, if applicable
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
PersonPhoneNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-4: The Phone for Person table

The Organization Table

Now that you have the table required to store information specific to people, we will create similar tables to store information about organizations.

Based on the scenario described earlier, we built tblOrganization, shown in Figure 4-5, to store the core information about the organization. Following the same process that you just completed, you can create the fields described next.

OrganizationName is a Text field that stores the name of this organization.

OrganizationAddress is a Text field that contains the first line of the organization's address.

OrganizationAddress2 is a Text field that contains a possible second line of the organization's address.

OrganizationCity is a Text field that contains the city.

OrganizationState is a Text field that records the state, province, or other political subdivision. Even if you are using a lookup table for state abbreviations, it makes perfect sense to store the abbreviation itself, rather than the value of the primary key.

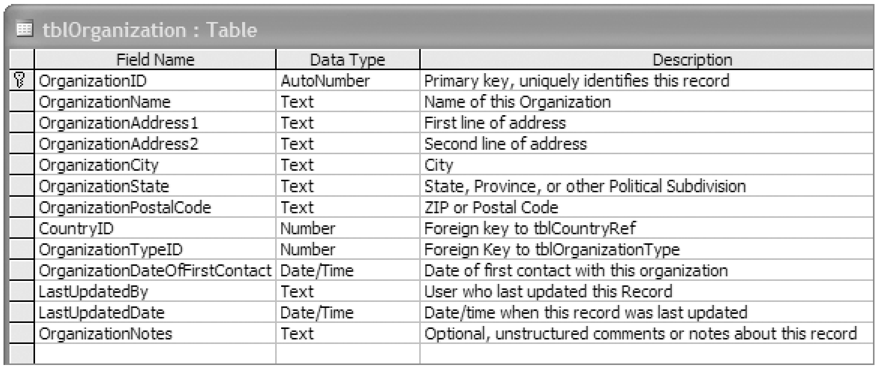
OrganizationPostalCode is a Text field that holds the zip or postal code.

CountryID is a Long Integer field that is used as a lookup key to the Country Reference table. In most cases, this could be done equally well by storing the abbreviation for the country, as we did with states in the State Reference table.

OrganizationTypeID is a Long Integer field that is used as a lookup key to tblOrganizationType.

OrganizationDateOfFirstContact is a Date/Time field that records the date (and possibly time) of the first contact with this organization.

Once you have the central table to store the direct details about each organization, you can create the tables to store the information for fields that may have several values, such as email.


tblOrganization : Table

Field Name	Data Type	Description
OrganizationID	AutoNumber	Primary key, uniquely identifies this record
OrganizationName	Text	Name of this Organization
OrganizationAddress1	Text	First line of address
OrganizationAddress2	Text	Second line of address
OrganizationCity	Text	City
OrganizationState	Text	State, Province, or other Political Subdivision
OrganizationPostalCode	Text	ZIP or Postal Code
CountryID	Number	Foreign key to tblCountryRef
OrganizationTypeID	Number	Foreign Key to tblOrganizationType
OrganizationDateOfFirstContact	Date/Time	Date of first contact with this organization
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
OrganizationNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-5: The Organization table

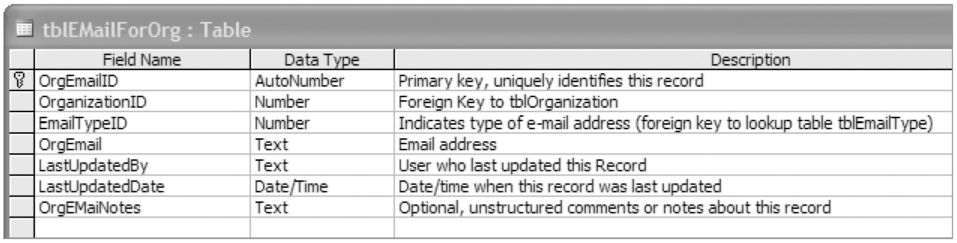
The Email for Organization Table

The table tblEMailForOrg (shown in Figure 4-6) is similar to the table for the e-mail addresses of individuals. You use the fields described next in this table.

OrganizationID is a Long Integer field that is used as a lookup field to store values from the table tblOrganization.

EmailType is a Long Integer field that indicates type of e-mail address. (This is a foreign key to lookup table tblEmailType.)

Email is a Text field that records an e-mail address.



tblEMailForOrg : Table

Field Name	Data Type	Description
OrgEmailID	AutoNumber	Primary key, uniquely identifies this record
OrganizationID	Number	Foreign Key to tblOrganization
EmailTypeID	Number	Indicates type of e-mail address (foreign key to lookup table tblEmailType)
OrgEmail	Text	Email address
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
OrgEMaiNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-6: The Email for Organization table

The Phone for Organization Table

Finally, you need a table to manage the numerous phone numbers that may be associated with a given organization.

The table tblPhoneForOrg contains the following fields, as shown in Figure 4-7.

OrganizationID is a Long Integer field that is used as the foreign key to a related record in tblOrganization.

OrgPhoneType is a Text field that describes the type of phone: home, work, mobile, or pager.

OrgPhoneDisplayOrder is a Long Integer field that is used to indicate the order in which to display the phone number, if multiple phone numbers exist for the related record.

OrgPhoneCountryCode is a Text field that contains a telephone's country code, which varies in number of digits in different regions.

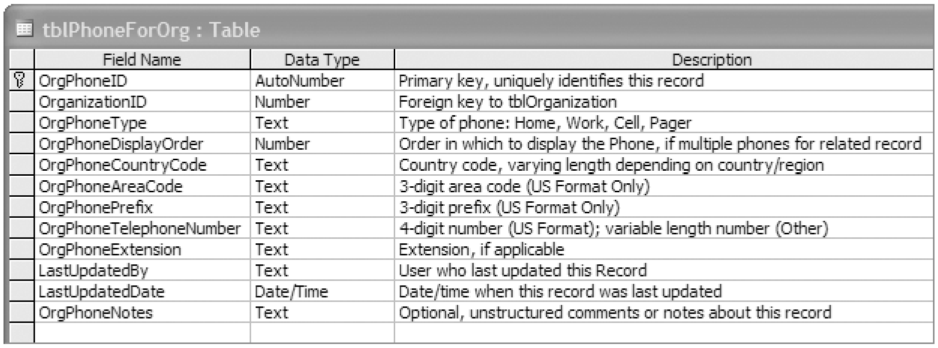
OrgPhoneAreaCode is a Text field that contains a three-digit area code (U.S. /North American format only).

OrgPhonePrefix is a Text field that holds a three-digit prefix (U.S. /North American format only).

OrgPhoneTelephoneNumber is a Text field that stores a four-digit number (U.S. /North American format) or a variable-length number (other).

As noted earlier, in relation to the Phone for Person table, the area code, prefix, and phone number can be stored as ''phone number'' in a single Text field, unless you expect to later perform detailed demographic analysis.

OrgPhoneExtension is a Text field that records an extension, if applicable.



tblPhoneForOrg : Table

Field Name	Data Type	Description
OrgPhoneID	AutoNumber	Primary key, uniquely identifies this record
OrganizationID	Number	Foreign key to tblOrganization
OrgPhoneType	Text	Type of phone: Home, Work, Cell, Pager
OrgPhoneDisplayOrder	Number	Order in which to display the Phone, if multiple phones for related record
OrgPhoneCountryCode	Text	Country code, varying length depending on country/region
OrgPhoneAreaCode	Text	3-digit area code (US Format Only)
OrgPhonePrefix	Text	3-digit prefix (US Format Only)
OrgPhoneTelephoneNumber	Text	4-digit number (US Format); variable length number (Other)
OrgPhoneExtension	Text	Extension, if applicable
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
OrgPhoneNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-7: The Phone for Organization table

NOTE In looking at the fields for tblPhoneForOrg, you will notice an anomaly in the name field for the primary key. When working with tables from other databases, you will often find that the table and field names do not exactly match the conventions that you have established for your own files. In some cases, it is easy to change, but in many cases, you will need to work with inherited names. In the table tblPhoneForOrg, the field PhoneID is an example of how you might retain the field name to facilitate exchanging data with the original source.

That completes the core tables that store the details specific to an individual or an organization. Next, we will create the lookup tables that store the lists of data from which users select, such as categories and states.

Lookup Tables

Lookup tables, as described earlier, provide an efficient way of obtaining faster and more accurate data entry, limiting the values in a field to a defined list, and increasing data integrity. We've used some in our sample database, and have also provided some reference tables that you can download as a bonus, so to speak. You might choose to use other lookup tables in your solutions.

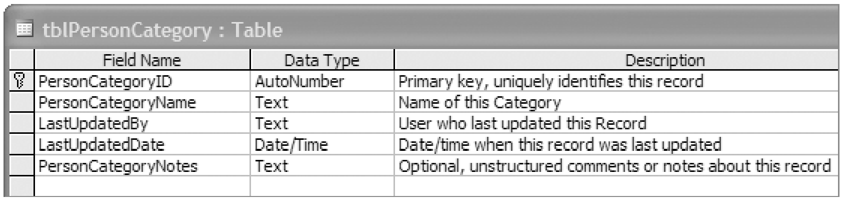
Most lookup tables have a similar structure, so we only provide a brief explanation of each table. Following the guidelines for good relational database models, each of our lookup tables includes an AutoNumber primary key, whose name is derived from the table name with the suffix of ID.

NOTE The values are from a lookup table, and you can store either the primary key value or the text value. Deciding which value to store is typically based on how the data might change, how the data will be used, and the developer's personal preference. You can learn more about lookup tables in Chapter 3 and Appendix B.

The Person Category Table (a Lookup Table)

The list of categories in which you want to group people is included in tblPersonCategory. As shown in Figure 4-8, it has only one field that is unique.

PersonCategoryName is a Text field that contains the name of this category. Categories are whatever you find useful for your database application, e.g., member, vendor, or contractor.



tblPersonCategory : Table

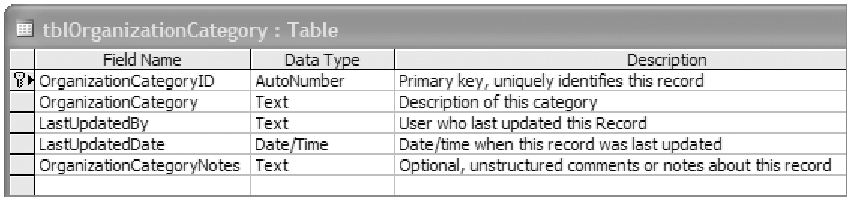
Field Name	Data Type	Description
PersonCategoryID	AutoNumber	Primary key, uniquely identifies this record
PersonCategoryName	Text	Name of this Category
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
PersonCategoryNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-8: The Person Category table

The Organization Category Table (a Lookup Table)

Next, we create a similar table for organizations. This table, named tblOrganizationCategory, is shown in Figure 4-9. It also has a single unique field.

OrganizationCategory is a Text field that specifies this organization's category. Examples of organization category could include commercial, non-profit, government, and religious.



tblOrganizationCategory : Table

Field Name	Data Type	Description
OrganizationCategoryID	AutoNumber	Primary key, uniquely identifies this record
OrganizationCategory	Text	Description of this category
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
OrganizationCategoryNotes	Text	Optional, unstructured comments or notes about this record

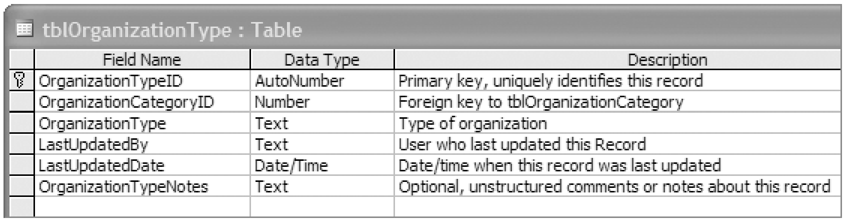
Figure 4-9: The Organization Category table

The Organization Type Table (a Lookup Table)

Within a category of organizations, such as services, you may have several types, such as office repair, printing, or delivery. We're using tblOrganizationType to store the list of specific types that are appropriate for our needs. Then, as shown in Figure 4-10, we can associate each type with a specific category by including the field from the category table. This table contains only two unique fields.

OrganizationCategoryID is a Long Integer field that is used as a lookup key to tblOrganizationCategory.

OrganizationType is a Text field that describes the organization type.



tblOrganizationType : Table

	Field Name	Data Type	Description
🔑	OrganizationTypeID	AutoNumber	Primary key, uniquely identifies this record
	OrganizationCategoryID	Number	Foreign key to tblOrganizationCategory
	OrganizationType	Text	Type of organization
	LastUpdatedBy	Text	User who last updated this Record
	LastUpdatedDate	Date/Time	Date/time when this record was last updated
	OrganizationTypeNotes	Text	Optional, unstructured comments or notes about this record

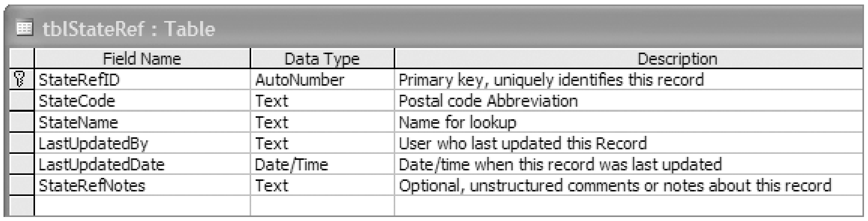
Figure 4-10: The Organization Type table

The State Reference Table (a Lookup Table)

Now we are ready to review some reference tables that can be helpful for managing contact information. We start with the table named tblStateRef. We have provided this in the example databases for this chapter on the CD that comes with this book. It is pre-populated with a list of U.S states, territory, and military postal locations, Canadian provinces and territories and their official abbreviations. The table, shown in Figure 4-11, has only two unique fields.

StateCode is a Text field that stores the postal code abbreviation for the state.

StateName is a Text field that contains the name of the state, province, or other locale for lookup.



tblStateRef : Table

	Field Name	Data Type	Description
🔑	StateRefID	AutoNumber	Primary key, uniquely identifies this record
	StateCode	Text	Postal code Abbreviation
	StateName	Text	Name for lookup
	LastUpdatedBy	Text	User who last updated this Record
	LastUpdatedDate	Date/Time	Date/time when this record was last updated
	StateRefNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-11: The State Reference table

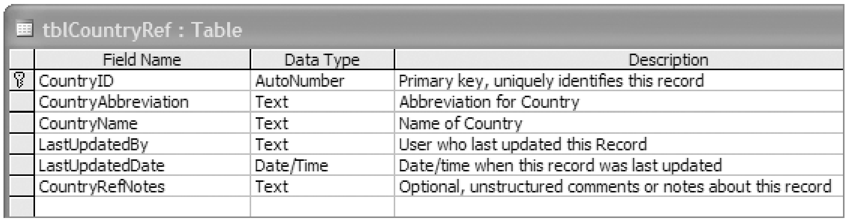
The Country Reference Table (a Lookup Table)

As we rely on the Internet and e-mail, it is becoming increasingly common to have contacts from around the world, so you may need to identify the country.

Having the official abbreviation for a country in a lookup table can help you quickly complete online forms, group contacts based on country, or even group countries into regions. We've provided a table, named tblCountryRef, complete with a few countries and their official abbreviations, as examples; the content is not intended to be nor represented as a complete list. If you make a lot of international calls, you may want to add a column for country or region code as used by the phone service. This table, shown in Figure 4-12, contains two unique fields.

CountryAbbreviation is a Text field that contains the abbreviation for the country's name.

CountryName is a Text field that contains the name of the country.



tblCountryRef : Table

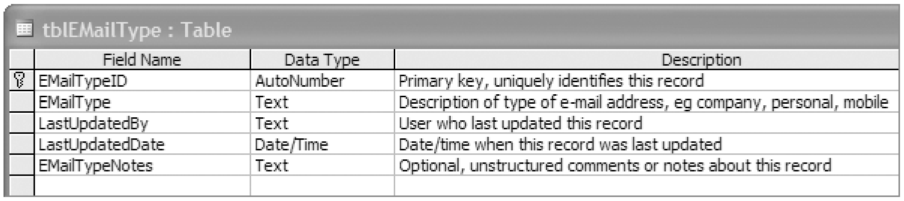
	Field Name	Data Type	Description
🔑	CountryID	AutoNumber	Primary key, uniquely identifies this record
	CountryAbbreviation	Text	Abbreviation for Country
	CountryName	Text	Name of Country
	LastUpdatedBy	Text	User who last updated this Record
	LastUpdatedDate	Date/Time	Date/time when this record was last updated
	CountryRefNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-12: The Country Reference table

The EMail Type Table (a Lookup Table)

The table tblEMailType is shown in Figure 4-13. It contains the values used to classify e-mail by type, if you choose to do so in your database application. There is one field in addition to the standard fields.

EMailType is a Text field that describes the type of e-mail, which could include values such as work, personal, school, or mobile.



tblEMailType : Table

	Field Name	Data Type	Description
🔑	EMailTypeID	AutoNumber	Primary key, uniquely identifies this record
	EMailType	Text	Description of type of e-mail address, eg company, personal, mobile
	LastUpdatedBy	Text	User who last updated this record
	LastUpdatedDate	Date/Time	Date/time when this record was last updated
	EMailTypeNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-13: The Email Type table (a lookup table)

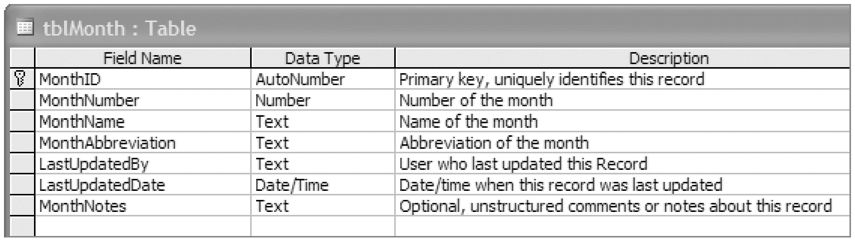
The Month Table (a Lookup Table)

As a matter of convenience, some developers like to use a table to store the various ways of displaying the month. We've provided tblMonth, which contains fields for the number, the full name, and the abbreviation, as shown in Figure 4-14.

MonthNumber is a Long Integer field that contains the number (1–12) of the month.

MonthName is a Text field that records the name of the month.

MonthAbbreviation is a Text field that stores the abbreviation for the month's name.



tblMonth : Table

Field Name	Data Type	Description
MonthID	AutoNumber	Primary key, uniquely identifies this record
MonthNumber	Number	Number of the month
MonthName	Text	Name of the month
MonthAbbreviation	Text	Abbreviation of the month
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
MonthNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-14: The Month table

That completes our list of lookup tables, but we have one more type of table to complete our example: a junction table.

Junction Tables

Simply described, a *junction table* is used to relate several items from one table with several items in another table. For example, consider classes and students. Each student attends several classes, and each class has multiple students. In Access, we resolve this by using a third table to store the foreign key to the related record. We also include a third field to store the primary key for the junction table itself.

You'll see how this works as you build the next table. One of the first things that you might notice is that when we refer to the junction table, we use a hyphen between the names of the two tables. This is a common way of indicating a junction table. Conversely, we do not use a hyphen referring to the tables in a one-to-many relationship. We've provided a detailed discussion in Appendix B, ''Relationships, Joins, and Nulls.''

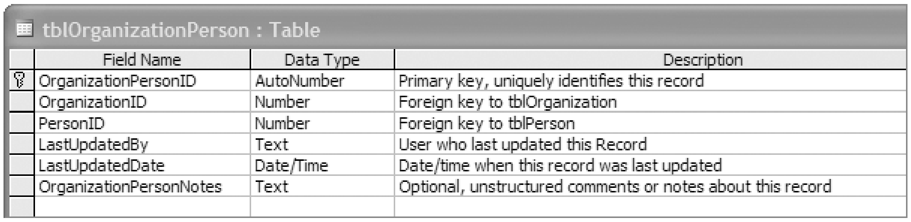
The Organization-Person Table (a Junction Table)

Because each person can belong to several organizations and each organization is likely to be represented by several people, this creates a many-to-many relationship. We use a junction table to resolve this into two one-to-many relationships. Each person can have many records in the junction table, and each organization can have many records in the junction table; but each record from the junction table is associated with only one person and one organization.

For our database, we use tblOrganizationPerson. Each record in this table will contain the value that uniquely identifies a specific organization and a person, as shown in Figure 4-15. These will both be required fields.

OrganizationID is a Long Integer field that is used as a foreign key to the related organization's record in tblOrganization.

PersonID is a Long Integer field that is used as foreign key to the related person's record in tblPerson.



tblOrganizationPerson : Table

	Field Name	Data Type	Description
🔑	OrganizationPersonID	AutoNumber	Primary key, uniquely identifies this record
	OrganizationID	Number	Foreign key to tblOrganization
	PersonID	Number	Foreign key to tblPerson
	LastUpdatedBy	Text	User who last updated this Record
	LastUpdatedDate	Date/Time	Date/time when this record was last updated
	OrganizationPersonNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-15: The Organization-Person table

Putting the Tables Together

Now that you have created all of the tables, you can see how they are related by looking at the relationship view, or schema. In database terminology, a schema is a diagram that shows how the tables are related. In the schema for the simple model, shown in Figure 4-16, you can see how relationships are formed by storing the primary key from one table in another table.

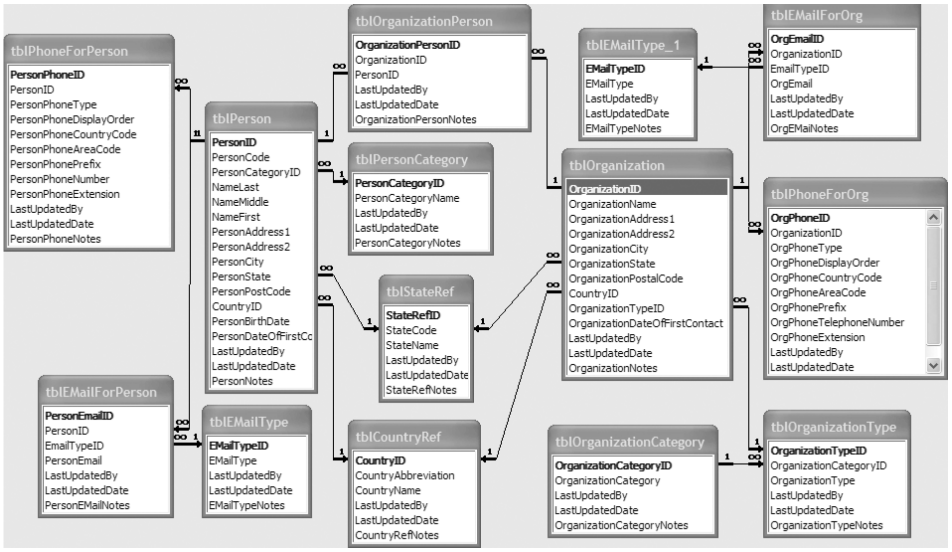


Figure 4-16: Relationship diagram of the simple model

Just to make it clear, the EMail For Person table and the EMail For Org table are both linked to the same EMail Type table, but two copies of the EMail Type table are shown (one with its name followed by ''_1'' because otherwise the relationship lines would be confusing).

Even a relatively simple database can start to look complicated as you add more tables, but understanding how the tables are related is fundamental to understanding how the data is managed in your application. Therefore, if you are faced with a complex situation, like our next example, you might find it helpful to break it into sections, and look at the tables and their relationships, one function at a time. If you'd like a refresher on creating and working with table relationships, we suggest that you review Chapter 3 and Appendix B.

Differences between the Simple and Complex Models

The primary difference between our simple example and the complex example is that some tables—the tables for people and the tables for organizations—have more information in common. Specifically, people and organizations may share email, phone, and address information. In the simple model that you just completed, we made the following assumptions:

- The person or organization using the database interacts with other individuals and small-to-modest-size organizations. For such an audience, it is reasonable to assume that a person or an organization has just a single address, and that address would not be shared with another person or organization. This enabled the simplicity of storing the address information directly in the tables for both the individual and the organization.
- Similarly, a person or an organization might have multiple e-mail addresses and multiple phone numbers and types, but for design purposes they do not share an e-mail address or a phone number with another person or organization. Please note that the design does not preclude more than one person using the same phone number or e-mail address. This results in a design for which e-mail addresses and phone numbers are kept in separate tables related one-to-many to the Person table and the Organization table.

NOTE The most common way to implement a one-to-many relationship is by including a foreign key (a field that has a unique value) from the table on the "one" side of the relationship (often referred to as the *parent* table) in the table on the "many" side of the relationship (often referred to as the *child* table). This is typically accomplished by storing the value of the primary key from the parent record in the child record. This enables you to store the same parent value in multiple child records, while limiting each child record to having only one parent, ergo the term "one-to-many."

For a complex model, you will see how to manage data when the people and organizations have more data in common. For example, if several people are employed by the same organization, you can expect to have multiple people sharing the same postal addresses, e-mail addresses (such as an alias of `info@xyz`), or phone numbers. This creates the many-to-many relationship that you recently encountered. Even though the same data is involved, it requires a different table structure. To help you understand and apply this concept, we will explain it a little differently this time.

NOTE **You implement a many-to-many relationship in Access by creating a third table called a *junction table*. The junction table stores a foreign key from the two tables in the relationship. In addition, there are no differences between the "sides" of the relationship; the two tables are treated equally. For example, if you have tens or hundreds of people in your database related to one organization, each of those people may share one or more addresses. Furthermore, if the organization has subsidiaries in the same location, and each one is included in the Organization table, multiple organizations could easily share those same addresses. Each combination of address and organization or address and person will be represented by a record in a junction table.**

In order to support the many-to-many relationships among e-mail, people, and organizations, and between phones, people, and organizations, you need to create several junction tables. Because the foreign keys will be stored in the junction table, you won't need a field to store a foreign key in the EMail or Phone tables, as you did when you created the one-to-many relationships in the simple model. Therefore, the tables for e-mail and phone numbers will be slightly different in this model.

Because address information is likely to be shared, you also need to create a separate table for addresses, and provide junction tables between the newly created Address table and the Person table, as well as between the Address table and the Organization table.

The Complex Model

Because the ''business'' data is similar, and only the structure of the tables is changed, we will look at an overview of the complex model. Using the basic structure of the simple model that you've just completed, we'll provide a full description of the tables that are different. However, for tables that are the same—such as all of the lookup tables in the simple model—you can find the full description in the previous example.

Figure 4-17 shows an overview of how data is related in the complex model. From this simple diagram, you can easily recognize the need for seven junction tables and three detail tables. The following sections explain the fields necessary for our two core tables—Person and Organization—and then explain the detail tables for address, e-mail, and phone. With those established, we can methodically create the junction tables.

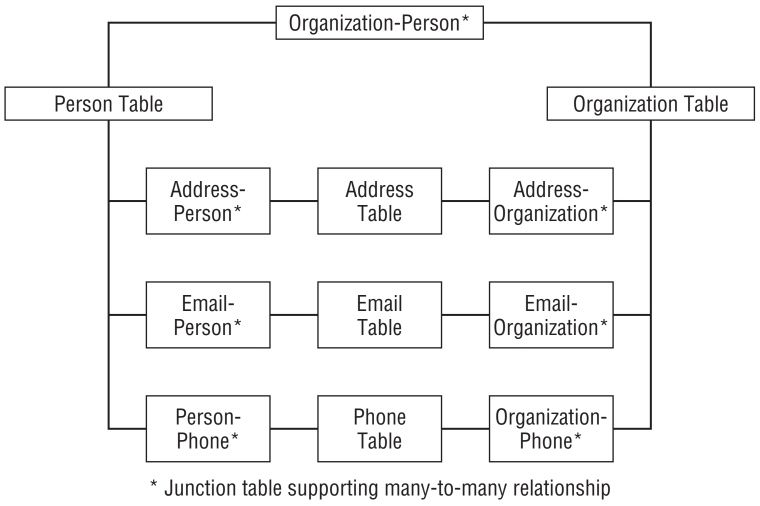
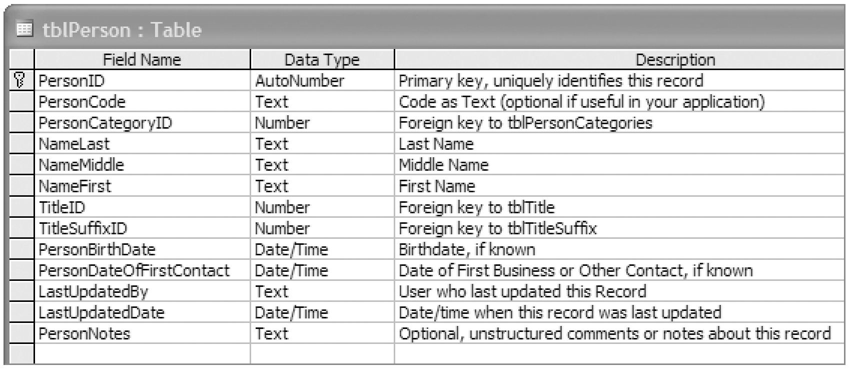


Figure 4-17: Overview of subjects (complex model)

The Person Table

The table named tblPerson, as shown below in Figure 4-18, identifies the person, the person's contact information, and additional information about the person required to meet the business needs of the application. This might include details such as position or to whom the person reports. These items of information are stored in the following fields in the table:



tblPerson : Table

Field Name	Data Type	Description
PersonID	AutoNumber	Primary key, uniquely identifies this record
PersonCode	Text	Code as Text (optional if useful in your application)
PersonCategoryID	Number	Foreign key to tblPersonCategories
NameLast	Text	Last Name
NameMiddle	Text	Middle Name
NameFirst	Text	First Name
TitleID	Number	Foreign key to tblTitle
TitleSuffixID	Number	Foreign key to tblTitleSuffix
PersonBirthDate	Date/Time	Birthdate, if known
PersonDateOfFirstContact	Date/Time	Date of First Business or Other Contact, if known
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
PersonNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-18: The Person table

PersonCode is an optional field used to store a code, typically as text, to identify the person. In order to uniquely identify the person, this needs to be a unique value. Because you want to avoid inadvertent duplication, you should set the field properties to have this field indexed and not allow duplicates.

PersonCategoryID is a Long Integer field that is used as a foreign key to the Person Category table. This might be the classification of worker, title, department or any other aspect that you need to record. You may require several similar fields.

NameLast is a Text field containing the person's last name.

NameMiddle is a Text field containing the person's middle name.

NameLast is a Text field containing the person's first name.

Resist the temptation to combine the name parts and only store and use a full name. As we explained when we were creating the Person table for the simple example, it is much better to have individual fields and then combine the data in any manner that you need.

PersonBirthDate is a Date/Time field that records the person's birth date, if known.

PersonDateOfFirstContact is a Date/Time field that records the date (and possibly time) of the first business or other contact, if known.

If you compare this to the simple model, you'll notice that all of the address fields have been removed, but you'll also see several fields that are the same in both the simple and complex versions of this table.

The Organization Table

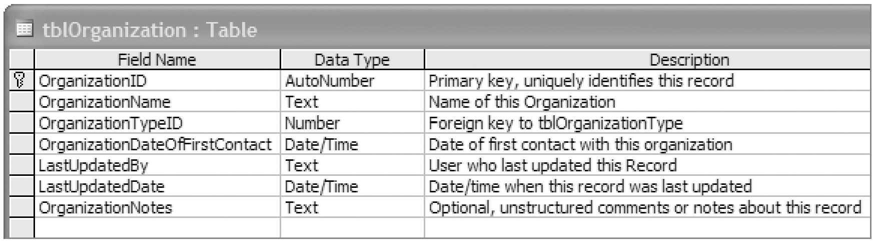
Next, we created a similar table for organizations, tblOrganization, shown in Figure 4-19. Like the Person table, it no longer contains address information, as it did in the simple model. Because all of the contact information can be shared with persons and other organizations, this table does not contain any of the related foreign keys, so it only has three new fields.

OrganizationName is a Text field that contains the name of this organization.

NOTE **Because you can anticipate several organizations having the same name, you cannot require this to be a unique value. Instead, you include features in the user interface, typically an Access Combo Box control, to display enough information to select the specific organization that you want. The Combo Box will likely become one of your favorite controls, when you begin designing and working with forms (which is a topic beyond the scope of this book).**

OrganizationTypeID is a Long Integer field that is used as a foreign key to look up values in the Organization Type table.

OrganizationDateOfFirstContact is a Date/Time field that records the date (and possibly time) of the first contact with this organization.



tblOrganization : Table

Field Name	Data Type	Description
OrganizationID	AutoNumber	Primary key, uniquely identifies this record
OrganizationName	Text	Name of this Organization
OrganizationTypeID	Number	Foreign key to tblOrganizationType
OrganizationDateOfFirstContact	Date/Time	Date of first contact with this organization
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
OrganizationNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-19: The Organization table

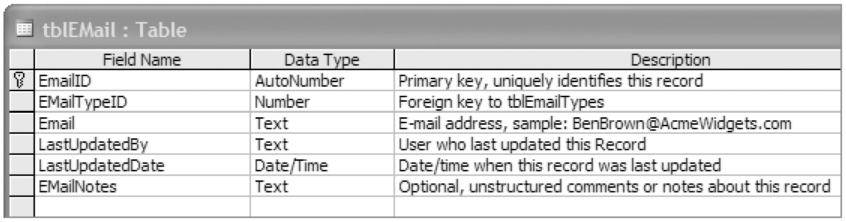
Now that you have the two core tables, you can create the tables for storing specific contact information that can be shared by people and organizations.

The EMail Table

tblEMail, as shown in Figure 4-20, contains the email addresses for people, organizations, or both.

EmailTypeID is a Long Integer field that is used to indicate the type of e-mail address (a foreign key to the lookup table tblEMailType). This could indicate whether the address is primary, home, work, and so on.

Email is a Text field that stores an e-mail address. There are options to store this as a hyperlink data type instead of text, but most programs will automatically convert a text version of an e-mail address to a hyperlink when you use the address.



tblEMail : Table

Field Name	Data Type	Description
EmailID	AutoNumber	Primary key, uniquely identifies this record
EMailTypeID	Number	Foreign key to tblEmailTypes
Email	Text	E-mail address, sample: BenBrown@AcmeWidgets.com
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
EMailNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-20: The EMail table

The Phone Table

The table tblPhone, shown in Figure 4-21, it differs from the Phone table in the simple model in that it doesn't contain a foreign key to either the Person table or the Organization table. Those relationships are accommodated by two junction tables, one between Organization and Phone and one between Person and Phone. We'll describe those shortly. For now, we'll create the fields for tblPhone.

PhoneType is a Text field that indicates the type of phone: home, work, mobile, or pager. The value is stored because the types of phones rarely change. If, in your environment, they do change more rapidly, replace this with a foreign key to a lookup table for phone types.

PhoneCountryCode is a Text field that records country code, a varying number of digits depending on region, for all formats. A value of 1 indicates U.S., Canada, and certain Caribbean countries.

PhoneAreaCode is a Text field that contains a three-digit area code (U.S./North American format only).

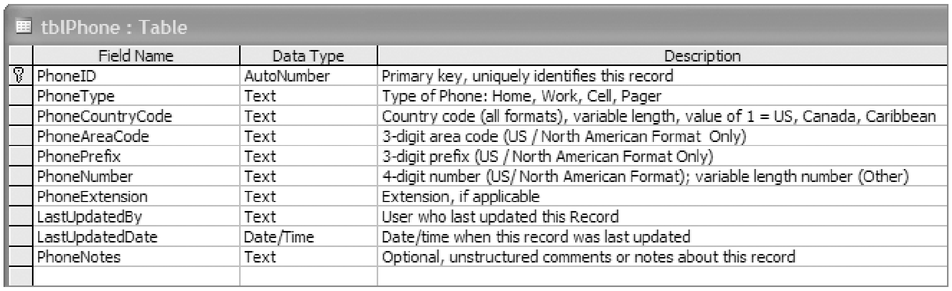
PhonePrefix is a Text field that holds a three-digit prefix (U.S./North American format only).

PhoneNumber is a Text field that stores a four-digit number (U.S./North American format) or a variable-length number (other formats).

As discussed when creating the tables for the simple model, the area code, prefix, and phone number can be stored in a single Text field, as ''phone number.'' Having separate fields can be helpful if you need to perform detailed demographic analysis. Single fields may also be a benefit if you frequently receive large lists of contact data to clean and import into your system.

PhoneExtension is a Text field that contains an extension, if applicable.

As with the simple model, you could also include a field to set the display order.


tblPhone : Table

Field Name	Data Type	Description
PhoneID	AutoNumber	Primary key, uniquely identifies this record
PhoneType	Text	Type of Phone: Home, Work, Cell, Pager
PhoneCountryCode	Text	Country code (all formats), variable length, value of 1 = US, Canada, Caribbean
PhoneAreaCode	Text	3-digit area code (US / North American Format Only)
PhonePrefix	Text	3-digit prefix (US / North American Format Only)
PhoneNumber	Text	4-digit number (US/ North American Format); variable length number (Other)
PhoneExtension	Text	Extension, if applicable
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
PhoneNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-21: The Phone table

The Address Table

The table, named tblAddress, has no counterpart table in the simple model because the address data was stored directly in the tables for Person and Organization.

You might start by creating the primary key and the field to identify the address type. Then, save some time by opening tblPerson in design view to copy and paste the address fields into the new table; then remove the prefix of people to end up with the field names shown in Figure 4-22.

AddressTypeID is a Long Integer field that is a foreign key to address type. Users will select from a list such as shipping, home, work, main, and so on.

Address1 is a Text field that contains first line of the address.

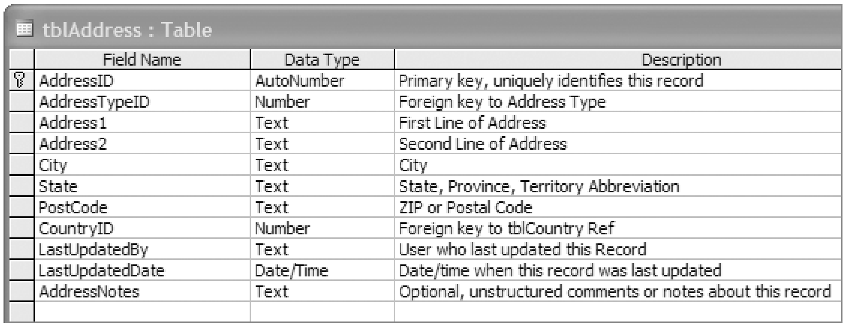
Address2 is a Text field that contains second line of the address.

City is a Text field that contains the city where the address is located.

State is a Text field that stores the state, province, or territory abbreviation. For validation that the abbreviation is typed correctly, this should be selected from a Combo Box in the user interface that has the State Ref table values as its Row Source.

PostCode is a Text field that holds the zip or postal code.

CountryID is a Long Integer field that is used as a foreign key to the Country Reference table.



tblAddress : Table

Field Name	Data Type	Description
AddressID	AutoNumber	Primary key, uniquely identifies this record
AddressTypeID	Number	Foreign key to Address Type
Address1	Text	First Line of Address
Address2	Text	Second Line of Address
City	Text	City
State	Text	State, Province, Territory Abbreviation
PostCode	Text	ZIP or Postal Code
CountryID	Number	Foreign key to tblCountry Ref
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
AddressNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-22: The Address table

Now that you have the tables for storing the information, you need to create the lookup tables that provide the lists for users to make selections.

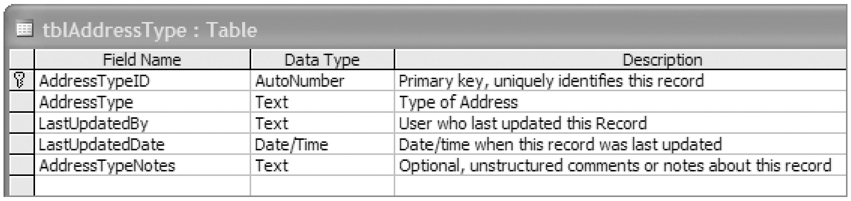
Lookup Tables

There are four new lookup tables in the complex model that were not used in the simple model. The other lookup tables are identical to those in the simple model, so you use the tables that you've already created. Again, you can incorporate other lookup tables for example to obtain state abbreviations.

The Address Type Table (a Lookup Table)

This new lookup table is for listing the types of addresses. As shown in Figure 4-23, tblAddressType has only one unique field.

AddressType is a Text field that specifies the type of address. For a business, appropriate types might be billing or receiving (ship-to). For an individual, types might include office, home, shipping.



tblAddressType : Table

Field Name	Data Type	Description
AddressTypeID	AutoNumber	Primary key, uniquely identifies this record
AddressType	Text	Type of Address
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
AddressTypeNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-23: The Address Type table

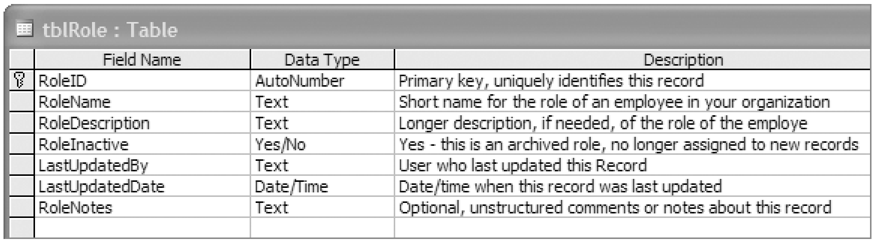
The Role Table (a Lookup Table)

The table tblRole is shown in Figure 4-24 and allows lookup of the name of the role that a person has in an organization. It has three table-specific fields.

RoleName is a Text field that contains the short name for the role of an employee in your organization.

RoleDescription is a Text field that is optionally used for a longer description of the role.

RoleInactive is a Yes/No field in which Yes means the role is archived for historical purposes and no longer available to be assigned.



tblRole : Table

Field Name	Data Type	Description
RoleID	AutoNumber	Primary key, uniquely identifies this record
RoleName	Text	Short name for the role of an employee in your organization
RoleDescription	Text	Longer description, if needed, of the role of the employe
RoleInactive	Yes/No	Yes - this is an archived role, no longer assigned to new records
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
RoleNotes	Text	Optional, unstructured comments or notes about this record

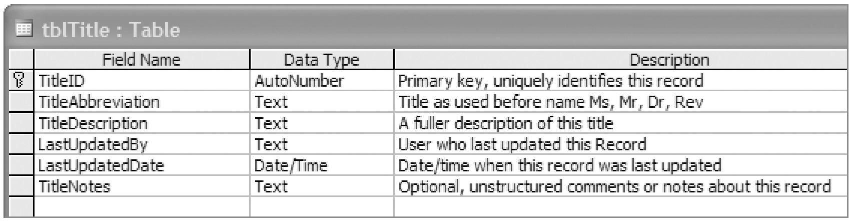
Figure 4-24: The Role table

The Title Table (a Lookup Table)

The table tblTitle, shown in Figure 4-25, contains the title by which the person is addressed in conversation, written communication, and postal address. It has two table unique fields.

TitleAbbreviation is a Text field containing the abbreviation of the title, such as Ms, Mr, Dr, or Rev.

TitleDescription is a Text field containing a longer description of the title, such as ''Ms is for a woman (irrespective of marital status)'' and ''Doctor of Medicine.''



tblTitle : Table

	Field Name	Data Type	Description
🔑	TitleID	AutoNumber	Primary key, uniquely identifies this record
	TitleAbbreviation	Text	Title as used before name Ms, Mr, Dr, Rev
	TitleDescription	Text	A fuller description of this title
	LastUpdatedBy	Text	User who last updated this Record
	LastUpdatedDate	Date/Time	Date/time when this record was last updated
	TitleNotes	Text	Optional, unstructured comments or notes about this record

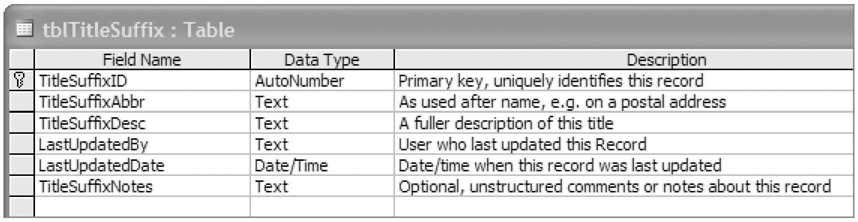
Figure 4-25: The Title table

The Title Suffix Table (a Lookup Table)

The table tblTitleSuffix, shown in Figure 4-26, contains the title suffix by which the person is addressed in conversation, written communication, and postal address. It has two fields in addition to the standard fields.

TitleSuffixAbbr is a Text field containing the abbreviation of the title, such as PhD, CEO, MA, or Col. USAF (Ret).

TitleDescription is a Text field containing a longer description of the title, such as ''Doctor of Philosophy is the most advanced post-graduate degree in many fields of study,'' ''Chief Executive Officer is the executive responsible for corporate policy,'' ''Master of Arts is a post-graduate degree'' and ''Colonel, USAF (Retired) is a title for a retired United States Air Force officer.''



tblTitleSuffix : Table

	Field Name	Data Type	Description
🔑	TitleSuffixID	AutoNumber	Primary key, uniquely identifies this record
	TitleSuffixAbbr	Text	As used after name, e.g. on a postal address
	TitleSuffixDesc	Text	A fuller description of this title
	LastUpdatedBy	Text	User who last updated this Record
	LastUpdatedDate	Date/Time	Date/time when this record was last updated
	TitleSuffixNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-26: The Title Suffix table

Junction Tables

As illustrated in the Figure 4-16 earlier in this chapter, the complex model relies on seven junction tables. You will use the junction table that we previously created to associate people with organizations, and create six new tables to support or resolve the other many-to-many relationships. They all have a similar structure, so you will quickly become familiar with creating and using them.

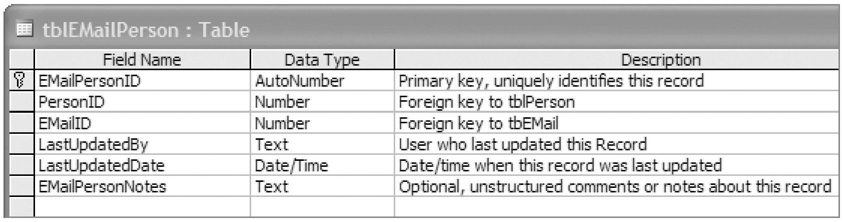
The EMail-Person Table (a Junction Table)

We'll start by creating a table named tblEMailPerson to associate people with e-mail addresses. As shown in Figure 4-27, there are really no ''new'' fields in this table. The first field is the AutoNumber primary key, and the table has three standard fields.

The remaining two fields are required to represent the many-to-many relationship between persons (tblPerson) and e-mail addresses (tblEmail). These two fields store the primary key for the selected record that represents a person and an e-mail address.

PersonID is a Long Integer field that stores the primary key from tblPerson as a foreign key.

EMailID is a Long Integer field that stores the primary key from tblEMail as a foreign key.


tblEMailPerson : Table

Field Name	Data Type	Description
EMailPersonID	AutoNumber	Primary key, uniquely identifies this record
PersonID	Number	Foreign key to tblPerson
EMailID	Number	Foreign key to tbEMail
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
EMailPersonNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-27: The EMail-Person table

You can apply the same explanation about the fields to the rest of the junction tables, so we will just list the field names without the details.

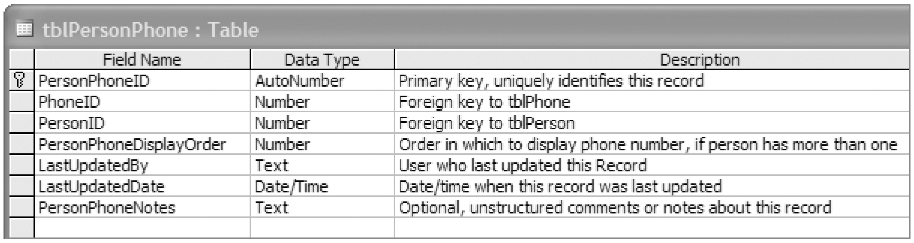
The Person-Phone Table (a Junction Table)

Similar to e-mail addresses, each person may have several phone numbers and a phone number may be used by multiple persons. You use the junction table, tblPersonPhone, between tblPerson and tblPhone. We have included a field that can be used to determine the order in which numbers will appear, so this table has three unique fields, as shown in Figure 4-28.

PhoneID is a Long Integer field that stores the primary key from tblPhone as a foreign key.

PersonID is a Long Integer field that stores the primary key from tblPerson as a foreign key.

PersonPhoneDisplayOrder is a Long Integer field that determines the order in which this phone number will be displayed, if the person has more than one phone number. This allows each Person to set an individual display order, even if the phone is shared.



tblPersonPhone : Table

	Field Name	Data Type	Description
🔑	PersonPhoneID	AutoNumber	Primary key, uniquely identifies this record
	PhoneID	Number	Foreign key to tblPhone
	PersonID	Number	Foreign key to tblPerson
	PersonPhoneDisplayOrder	Number	Order in which to display phone number, if person has more than one
	LastUpdatedBy	Text	User who last updated this Record
	LastUpdatedDate	Date/Time	Date/time when this record was last updated
	PersonPhoneNotes	Text	Optional, unstructured comments or notes about this record

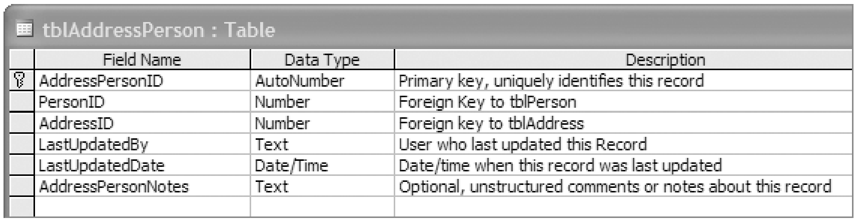
Figure 4-28: The Person-Phone table

The Address-Person Table (a Junction Table)

Because we've already established that a person may have multiple addresses, such as a post office box and home address, and that several people can frequently have the same address, we again use a junction table to associate the two records. This table is named tblAddressPerson (see Figure 4-29). It contains the standard fields plus a field that uniquely identifies the person, and a field to uniquely identify the address. You might also include a field to store the display order if that will not be determined by the address type, which is stored in the Address table.

PersonID is a Long Integer field that stores the primary key from tblPerson as a foreign key.

AddressID is a Long Integer field that stores the primary key from tblAddress as a foreign key.



tblAddressPerson : Table

	Field Name	Data Type	Description
🔑	AddressPersonID	AutoNumber	Primary key, uniquely identifies this record
	PersonID	Number	Foreign Key to tblPerson
	AddressID	Number	Foreign key to tblAddress
	LastUpdatedBy	Text	User who last updated this Record
	LastUpdatedDate	Date/Time	Date/time when this record was last updated
	AddressPersonNotes	Text	Optional, unstructured comments or notes about this record

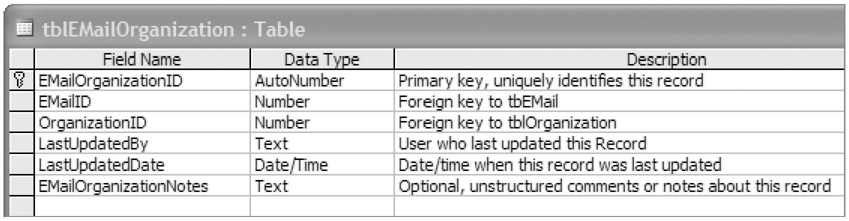
Figure 4-29: The Address-Person table

The EMail-Organization Table (a Junction Table)

At this point, you've created the junction tables required to manage the data for people. Your next step is to create similar tables for organizations. Like people, an organization may have several e-mail addresses. You might also find that one e-mail address can be associated with multiple records in your Organization table—for example, if a company has several subsidiaries co-located and using the same computer systems (sharing would be more common with physical addresses, but may occur with email addresses, too). To support this type of flexibility, we created a junction table named tblEMailOrganization, shown in Figure 4-30. Like the e-mail table for people, this table has only two unique fields.

EMailID is a Long Integer field that stores the primary key from tblEMail as a foreign key.

OrganizationID is a Long Integer field that stores the primary key from tblOrganization as a foreign key.


tblEMailOrganization : Table

Field Name	Data Type	Description
EMailOrganizationID	AutoNumber	Primary key, uniquely identifies this record
EMailID	Number	Foreign key to tbEMail
OrganizationID	Number	Foreign key to tblOrganization
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
EMailOrganizationNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-30: The EMail-Organization table

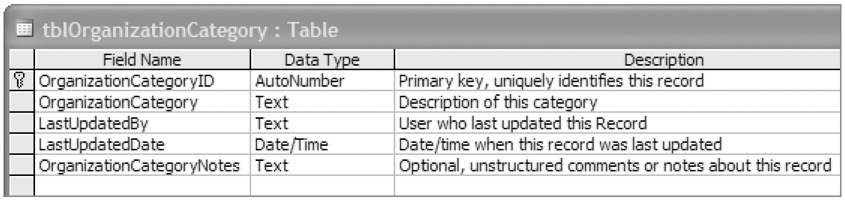
The Organization-Phone Table (a Junction Table)

Again, in the same manner that you created the junction table for managing phone listings for people, you create tblOrganizationPhone to associate organizations with phone listings. As shown in Figure 4-31, we have included a field to stipulate the order in which numbers will appear, so there are three unique fields in this table.

PhoneID is a Long Integer field that stores the primary key from tblPhone as a foreign key.

OrganizationID is a Long Integer field that stores the primary key from tblOrganization as a foreign key.

OrganizationPhoneDisplayOrder is a Long Integer field that determines the order in which this phone number will be displayed if the organization has more than one phone number.


tblOrganizationCategory : Table

Field Name	Data Type	Description
OrganizationCategoryID	AutoNumber	Primary key, uniquely identifies this record
OrganizationCategory	Text	Description of this category
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
OrganizationCategoryNotes	Text	Optional, unstructured comments or notes about this record

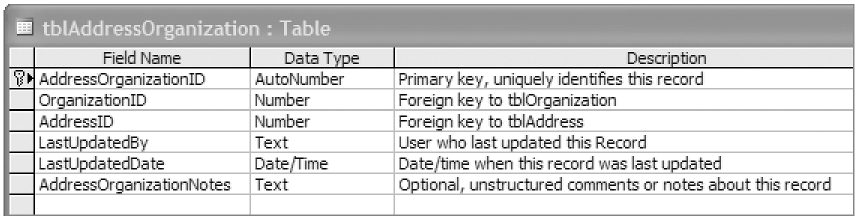
Figure 4-31: The Organization-Phone table

The Address-Organization Table (a Junction Table)

The final junction table that you'll need for organization contact information is used to associate the addresses with each organization. Using the same approach as we did for people, create a table tblAddressOrganization, as shown in Figure 4-32. Because the address type is in the Address table, you do not need to include that here. However, similar to the management of phone numbers, you may want to include a field to specify the display or sort order. Our table includes two fields that are unique to this table.

OrganizationID is a Long Integer field that stores the primary key from tblOrganization as a foreign key.

AddressID is a Long Integer field that stores the primary key from tblAddress as a foreign key.


tblAddressOrganization : Table

Field Name	Data Type	Description
AddressOrganizationID	AutoNumber	Primary key, uniquely identifies this record
OrganizationID	Number	Foreign key to tblOrganization
AddressID	Number	Foreign key to tblAddress
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
AddressOrganizationNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-32: The Address-Organization table

NOTE When you are reviewing this table you will notice that the order of the subjects is reversed in the name of the primary key field. This illustrates some of the dilemma and challenges that you will encounter in many of your projects. It would seem logical to use organization as the prefix for the table and field names, but we also wanted to follow our convention of using alphabetical order for naming junction tables. As we were pulling all of the files together, we recognized the benefit of listing junction table names in alphabetical order to provide consistency as you copy tables from one database to another. In this case, the table name was revised but the name of the primary key was not. We encourage you to name the tables and fields to suit your needs, whether that is using abbreviations, reversing the names, or working with the files as-is.

The Organization-Person Table (a Junction Table)

Like the other junction tables, this one, named tblOrganizationPerson (shown in Figure 4-33), creates an association between two subjects, a person and an organization; but this table also provides additional information that is specific to that unique relationship. In our scenario, we are also identifying the title of a person and the role that he or she has within the selected organization.

NOTE The values are from a lookup table, and you can store either the primary key value or the text value. Deciding which value to store is typically based on how

the data might change and the developer's personal preference. You can learn more about lookup tables in Chapter 3 and Appendix B.

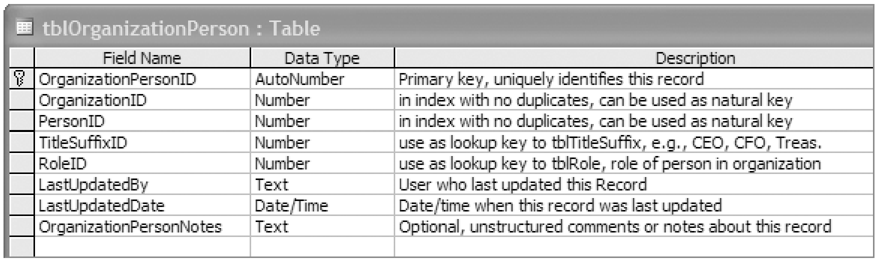
For our example, tblOrganizationPerson requires the four fields described below in addition to the three standard fields.

OrganizationID is a Long Integer field that stores the primary key from tblOrganization as a foreign key.

PersonID is a Long Integer field that stores the primary key from tblPerson as a foreign key.

TitleSuffixID is a Long Integer field that is used as the foreign key to a lookup table (sometimes known as a ''lookup key'') called tblTitleSuffix, in reference to the person and organization. Some examples are CEO, CFO, Treas, and Secy.

RoleID is a Long Integer field that is used as a lookup key to tblRole, for the role of this person in this organization.



tblOrganizationPerson : Table

Field Name	Data Type	Description
OrganizationPersonID	AutoNumber	Primary key, uniquely identifies this record
OrganizationID	Number	in index with no duplicates, can be used as natural key
PersonID	Number	in index with no duplicates, can be used as natural key
TitleSuffixID	Number	use as lookup key to tblTitleSuffix, e.g., CEO, CFO, Treas.
RoleID	Number	use as lookup key to tblRole, role of person in organization
LastUpdatedBy	Text	User who last updated this Record
LastUpdatedDate	Date/Time	Date/time when this record was last updated
OrganizationPersonNotes	Text	Optional, unstructured comments or notes about this record

Figure 4-33: The Organization-Person table

Putting the Tables Together

Now that we've described all of the tables, we are ready to show you how they fit together. At this point, you've created several tables that each store pieces of information about a person or an organization, as well as junction tables required to connect the records. You've also created several lookup tables that make it easy for users to select values from lists of data; the lists can also be beneficial by limiting options to a predetermined set of values . All of those tables and relationships are illustrated in Figure 4-34.

If you follow the lines between the tables, you can see that the contact data is linked to each person and each organization. You can also see how tblOrganizationPeople links people with organizations.

You can use Access to draw a relationship diagram. Of course, you will have to choose the tables to be included and establish the relationships between them. The relationship diagram identifies and represents the relationships based on the field names and types. You can learn more about relationships in Chapter 3 and Appendix B.

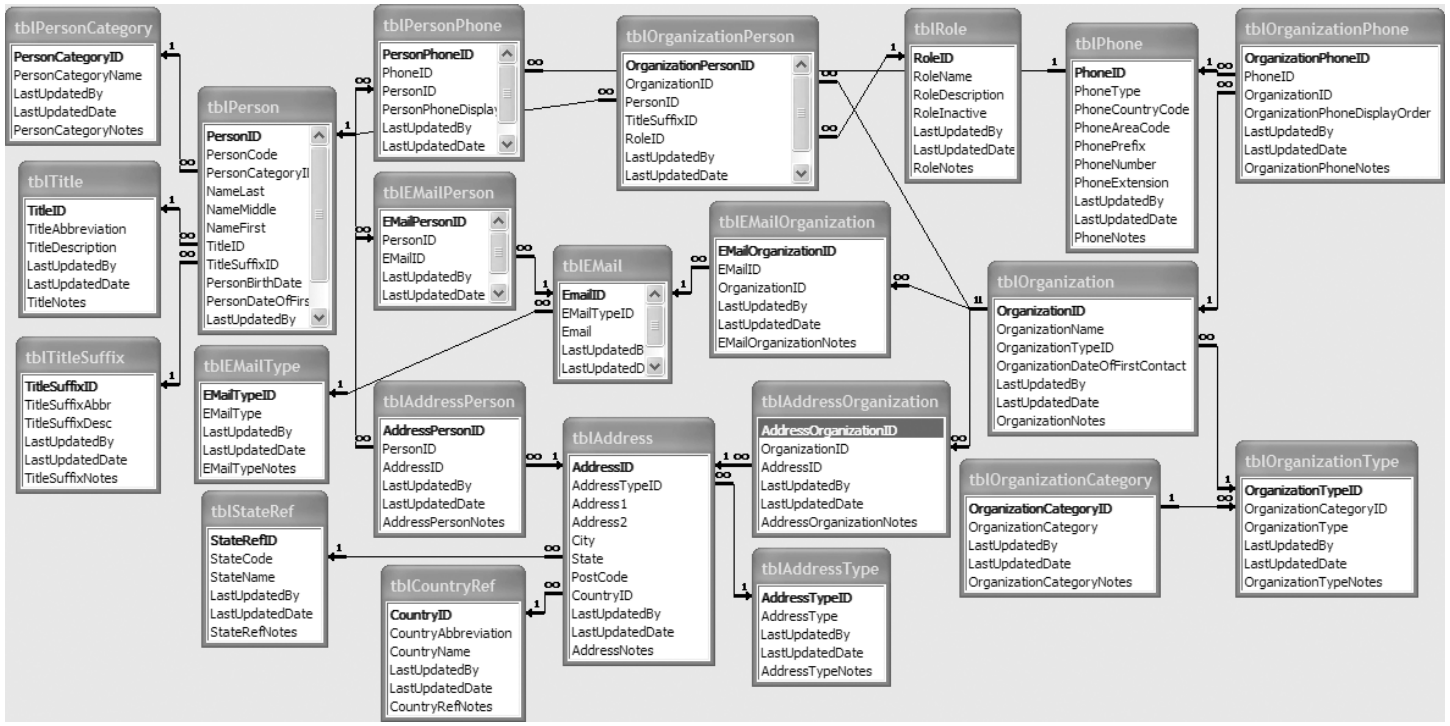


Figure 4-34: Relationship diagram of the Complex Model

RELATED TABLES REQUIRED BY BUSINESS FUNCTIONS

You would have an unusual business if there were no business functions that needed to use not only the information stored in the People and Organization tables in this chapter, but also other related information, such as Customer Relationship Management or Managing Memberships. You can find the tables you need and information about their relationships to the tables in this chapter by reading the chapters in this book devoted to specific business functions.

Adding More Information about People and Organizations

If you need to record additional information about a person or organization, you can insert additional fields into the Person table or the Organization table. As an alternative, a separate table of business function–specific information can be created and related using a one-to-one relationship to the Person or Organization table. This type of table structure should only be used if the tables for Person and Organization are shared by multiple business functions and if, in your judgment, storing the data in a separate table will simplify managing and sharing it.

Information that can have several records associated with a person or an organization, such as appointments, events, or contact history, needs to be stored in separate tables. These would require additional junction tables similar to those used for contact information.

Summary

In this chapter, you learned how to create two databases to manage contact information for people and organizations. We started by discussing the purpose of the database and what we wanted to accomplish. We then considered sample questions that you can use to determine what level of detail you need to record. For example, you might include separate, unique pieces of data, such as a job title, or it may require allowing several similar items, such as having multiple phone numbers. In that case, you can also provide a way to specify the display order.

Based on the details needed and how much information is shared, you may be able to use a simple table structure, such as when there is limited data in common between people and organizations. Otherwise, if people and organizations might both use the same contact information, you can provide a more complex table structure.

Even in the simple model, you have the opportunity to create a many-to-many relationship to create the ties between records about a person and records about an organization. We explained how the data is related and managed using a junction table. This is a valuable model because you will frequently find scenarios that include this type of relationship.

You have examined tables describing people and organizations. In the simple model these tables include address information, but in the complex model people and organizations are each related many-to-many to a separate, shared Address table. You also examined a case for which there are separate Phone and E-Mail tables for people and for organizations (related one-to-many, in the simple model); and a case in which phone and e-mail information may be shared, and are therefore kept in separate tables related many-to-many to each of the People and Organization tables.

You have also seen that you can apply a category to people and to organizations, should you find that useful, as it enables you to group by category for searching or reporting in your database application.

As a bonus, we also discussed and provided some additional tables, not specifically and directly related to people and organizations. We've included tables with some official data that is commonly needed for addresses, such as the State Reference table, and another for referring to countries. We've also provided a table with month names, numbers, and abbreviations. The latter table can be used to eliminate some calculations with date-and-time functions that are built into Access's expression feature and its Visual Basic for Applications (VBA) language.

By working through the examples, you have not only created table structures that can serve as standalone solutions, you have had the opportunity to learn a lot about designing a database. As you read the other chapters, you can concentrate on the business functionality that they are describing and implementing; and as you complete other examples, you can incorporate the appropriate tables from this chapter to support your needs for contact information—a nontrivial aspect of many database solutions.

Contents