8

Keys

A fact in itself is nothing. It is valuable only for the idea attached to it, or for the proof which it furnishes.
—CLAUDE BERNARD

Topics Covered in This Chapter

By now you've identified all the subjects that the database will track and defined the table structures that will represent those subjects. Furthermore, you've put the structures through a screening process to control their makeup and quality. In this next stage of the database-design process, you'll begin the task of assigning *keys* to each table. You'll soon learn that there are different types of keys, and each plays a particular role within the database structure. All but one key is assigned during this stage; you'll assign the remaining key later (in Chapter 10) as you establish relationships between tables.

Why Keys Are Important

Keys are crucial to a table structure for the following reasons:

- *They ensure that each record in a table is precisely identified.* As you already know, a table represents a singular collection of similar objects or events. (For example, a CLASSES table represents a *collection* of classes, not just a single class.) The complete set of records within the table constitutes the collection, and each record represents a unique instance of the table's subject within that collection. You must have some means of accurately identifying each instance, and a key is the device that allows you to do so.
- *They help establish and enforce various types of integrity.* Keys are a major component of table-level integrity and relationship-level integrity. For instance, they enable you to ensure that a table has unique records and that the fields you use to establish a relationship between a pair of tables always contain matching values.
- *They serve to establish table relationships.* As you'll learn in Chapter 10, you'll use keys to establish a relationship between a pair of tables.

Always make certain that you define the appropriate keys for each table. Doing so will help you guarantee that the table structures are sound, that redundant data within each table is minimal, and that the relationships between tables are solid.

Establishing Keys for Each Table

Your next task is to establish keys for each table in the database. There are four main types of keys: *candidate*, *primary*, *foreign*, and *non-keys*. A key's type determines its function within the table.

Candidate Keys

The first type of key you establish for a table is the *candidate* key, which is a field or set of fields that uniquely identifies a single instance of the table's subject. Each table must have *at least one* candidate key. You'll eventually examine the table's pool of available candidate keys and designate one of them as the official primary key for the table.

Before you can designate a field as a candidate key, you must make certain it complies with *all* of the Elements of a Candidate Key. These elements constitute a set of guidelines you can use to determine whether the field is fit to serve as a candidate key. You cannot designate a field as a candidate key if it fails to conform to *any* of these elements.

Elements of a Candidate Key

- *It cannot be a multipart field.* You've seen the problems with multipart fields, so you know that using one as an identifier is a bad idea.
- *It must contain unique values.* This element helps you guard against duplicating a given record within the table. Duplicate records are just as bad as duplicate fields, and you must avoid them at all costs.
- *It cannot contain null values.* As you already know, a null value represents the *absence* of a value. There's absolutely no way a candidate key field can identify a given record if its value is null.
- *Its value cannot cause a breach of the organization's security or privacy rules.* Values such as passwords and Social Security Numbers are not suitable for use as a candidate key.
- *Its value is not optional in whole or in part.* A value that is optional implies that it may be null at some point. You can infer, then, that

an optional value automatically violates the previous element and is, therefore, unacceptable. (This caveat is especially applicable when you want to use two or more fields as a candidate key.)

- *It comprises a minimum number of fields necessary to define uniqueness.* You can use a combination of fields (treated as a single unit) to serve as a candidate key, so long as each field contributes to defining a unique value. Try to use as few fields as possible, however, because overly complex candidate keys can ultimately prove to be difficult to work with and difficult to understand.
- *Its values must uniquely and exclusively identify each record in the table.* This element helps you guard against duplicate records and ensures that you can accurately reference any of the table's records from other tables in the database.
- *Its value must exclusively identify the value of each field within a given record.* This element ensures that the table's candidate keys provide the only means of identifying each field value within the record. (You'll learn more about this particular element in the section on primary keys.)
- *Its value can be modified only in rare or extreme cases.* You should never change the value of a candidate key unless you have an absolute and compelling reason to do so. A field is likely to have difficulty conforming to the previous elements if you can change its value arbitrarily.

Establishing a candidate key for a table is quite simple: Look for a field or set of fields that conforms to all of the Elements of a Candidate Key. You'll probably be able to define more than one candidate key for a given table. Loading a table with sample data will give you the means to identify potential candidate keys accurately. (You used this same technique in the previous chapter.)

See if you can identify any candidate keys for the table in Figure 8.1.

Employees

Employee ID	Social Security Number	EmpFirst Name	EmpLast Name	EmpStreet Address	EmpCity	EmpState	EmpZipcode	EmpHome Phone
1000	856-91-9938	Kendra	Bonnicksen	1204 Bryant Road	Seattle	WA	98157	363-9948
1001	886-11-2231	Katherine	Erlich	101 C Street, Apt. 32	Bellevue	WA	98046	322-6992
1002	901-48-0039	Timothy	Ennis	7402 Kingman Drive	Redmond	WA	98115	527-4992
1003	816-93-1299	Shannon	McLain	4141 Lake City Way	Seattle	WA	98136	336-5992
1004	978-02-1129	Susan	McLain	2100 Mineola Avenue	Seattle	WA	98115	572-9948
1005	955-92-5583	Estela	Pundt	101 C Street, Apt. 32	Bellevue	WA	98046	322-6992
1006	801-22-1734	Timothy	Sherman	66 NE 120th	Bothell	WA	98216	522-3232

Figure 8.1. *Are there any candidate keys in this table?*

You probably identified Employee ID, Social Security Number, EmpLast Name, EmpFirst Name *and* EmpLast Name, EmpZipcode, and EmpHome Phone as potential candidate keys. But you'll need to examine these fields more closely to determine which ones are truly eligible to become candidate keys. Remember that you must automatically disregard any field(s) failing to conform to even *one* of the Elements of a Candidate Key.

Upon close examination, you can draw the following conclusions:

- *Employee ID is eligible.* This field conforms to every element of a candidate key.
- *Social Security Number is ineligible because it could contain null values and will most likely compromise the organization's privacy rules.* Contrary to what the sample data shows, this field could contain a null value. For example, there are many people working in the United States who do not have Social Security numbers because they are citizens of other countries.

❖ **Note** Despite its widespread use in many types of databases, I would strongly recommend that you refrain from using SOCIAL SECURITY NUMBER as a candidate key (or a primary key, for that matter) in any of your database structures. In many instances, it doesn't conform to the Elements of a Candidate Key. You can learn some very interesting facts about Social Security numbers (which will shed some light on why they make poor candidate/primary keys) by visiting the Social Security Adminstration's Web site at http://www.ssa.gov.

- *EMPLAST NAME is ineligible because it can contain duplicate values.* As you've learned, the values of a candidate key must be unique. In this case there can be more than one occurrence of a particular last name.
- *EMPFIRST NAME and EMPLAST NAME are eligible.* The combined values of both fields will supply a unique identifier for a given record. Although multiple occurrences of a particular first name or last name will occur, the combination of a given first name and last name will always be unique. (Some of you are probably saying, "This is not necessarily always true." You're absolutely right. Don't worry; we'll address this issue shortly.)
- *EMPZIPCODE is ineligible because it can contain duplicate values.* Many people live in the same zip code area, so the values in EMPZIPCODE cannot possibly be unique.
- *EMPHOME PHONE is ineligible because it can contain duplicate values and is subject to change.* This field will contain duplicate values for either of these reasons:
    1. One or more family members work for the organization.
    2. One or more people share a residence that contains a single phone line.

You can confidently state that the EMPLOYEES table has two candidate keys: Employee ID and the combination of EmpFirst Name and EmpLast Name.

Mark candidate keys in your table structures by writing the letters "CK" next to the name of each field you designate as a candidate key. A candidate key composed of two or more fields is known as a *composite candidate key*, and you'll write "CCK" next to the names of the fields that make up the key. When you have two or more composite candidate keys, use a number within the mark to distinguish one from another. If you had two composite candidate keys, for example, you would mark one as "CCK1" and the other as "CCK2."

Apply this technique to the candidate keys for the EMPLOYEES table in Figure 8.1. Figure 8.2 shows how your structure should look when you've completed this task.

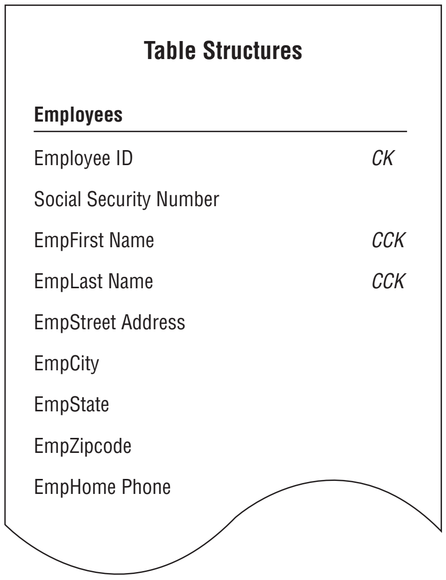

Table Structures

Employees	
Employee ID	*CK*
Social Security Number	
EmpFirst Name	*CCK*
EmpLast Name	*CCK*
EmpStreet Address	
EmpCity	
EmpState	
EmpZipcode	
EmpHome Phone	

Figure 8.2. *Marking candidate keys in the EMPLOYEES table structure.*

Now, try to identify as many candidate keys as you can for the PARTS table in Figure 8.3.

Parts

Part Name	Model Number	Manufacturer Name	Retail Price
Shimka XT Cranks	XT-113	Shimka Incorporated	199.95
Faust Brake Levers	BL / 45	Faust USA	53.79
MiniMite Pump		MiniMite	35.00
Hobo Fanny Pack		Hobo Bike Company	59.00
Diablo Bike Pedals	Mtn-A26	Diablo Sports	129.50
Shimka Truing Stand	SP-100		37.95
Faust Brake Levers	BL / 60	Faust USA	79.95

Figure 8.3. *Can you identify any candidate keys in the PARTS table?*

At first glance, you may believe that PART NAME, MODEL NUMBER, the combination of PART NAME and MODEL NUMBER, and the combination of MANUFACTURER and PART NAME are potential candidate keys. After investigating this theory, however, you come up with the following results:

- *PART NAME is ineligible because it can contain duplicate values.* A given part name will be duplicated when the part is manufactured in several models. For example, this is the case with Faust Brake Levers.
- *MODEL NUMBER is ineligible because it can contain null values.* A candidate key value must exist for each record in the table. As you can see, some parts do not have a model number.
- *PART NAME and MODEL NUMBER are ineligible because either field can contain null values.* The simple fact that MODEL NUMBER can contain null values instantly disqualifies this combination of fields.
- *MANUFACTURER and PART NAME are ineligible because the values for these fields seem to be optional.* Recall that a candidate key value

cannot be optional in whole or in part. In this instance, you can infer that entering the manufacturer name is optional when it appears as a component of the part name; therefore, you cannot designate this combination of fields as a candidate key.

It's evident that you don't have a single field or set of fields that qualifies as a candidate key for the PARTS table. This is a problem because each table must have at least *one* candidate key. Fortunately, there is a solution.

Artificial Candidate Keys

When you determine that a table does not contain a candidate key, you can create and use an *artificial* (or *surrogate*) candidate key. (It's artificial in the sense that it didn't occur "naturally" in the table; you have to manufacture it.) You establish an artificial candidate key by creating a new field that conforms to all of the Elements of a Candidate Key and then adding it to the table; this field becomes the official candidate key.

You can now solve the problem in the PARTS table. Create an artificial candidate key called Part Number and assign it to the table. (The new field will automatically conform to the Elements of a Candidate Key because you're creating it from scratch.) Figure 8.4 shows the revised structure of the PARTS table.

When you've established an artificial candidate key for a table, mark the field name with a "CK" in the table structure, just as you did for the EMPLOYEES table in the previous example.

You may also choose to create an artificial candidate key when it would be a stronger (and thus, more appropriate) candidate key than any of the existing candidate keys. Assume you're working on an EMPLOYEES table and you determine that the only available candidate key is the combination of the EmpFirst Name and EmpLast Name fields. Although this may be a valid candidate key, using a single-field candidate key might

Parts

Part Number	Part Name	Model Number	Manufacturer Name	Retail Price
41000	Shimka XT Cranks	XT-113	Shimka Incorporated	199.95
41001	Faust Brake Levers	BL / 45	Faust USA	53.79
41002	MiniMite Pump		MiniMite	35.00
41003	Hobo Fanny Pack		Hobo Bike Company	59.00
41004	Diablo Bike Pedals	Mtn-A26	Diablo Sports	129.50
41005	Shimka Truing Stand	SP-100		37.95
41006	Faust Brake Levers	BL / 60	Faust USA	79.95

Figure 8.4. *The PARTS table with the artificial candidate key* PART NUMBER.

prove more efficient and may identify the subject of the table more easily. Let's say that everyone in the organization is accustomed to using a unique identification number rather than a name as a means of identifying an employee. In this instance, you can choose to create a new field named EMPLOYEE ID and use it as an artificial candidate key. This is an absolutely acceptable practice—do this without hesitation or reservation if you believe it's appropriate.

❖ **Note** I commonly create an ID field (such as EMPLOYEE ID, VENDOR ID, DEPARTMENT ID, CATEGORY ID, and so on) and use it as an artificial candidate key. It always conforms to the Elements of a Candidate Key, makes a great primary key (eventually), and, as you'll see in Chapter 10, makes the process of establishing table relationships much easier.

Review the candidate keys you've selected and make absolutely certain that they thoroughly comply with the Elements of a Candidate Key. Don't be surprised if you discover that one of them is not a candidate key after all—incorrectly identifying a field as a candidate key happens occasionally. When this does occur, just remove the "CK" designator from the field name in the table structure. Deleting a candidate key won't pose a problem as long as the table has more than one candidate key. If you discover, however, that the only candidate key you identified for the table is *not* a candidate key, you *must* establish an artificial candidate key for the table. After you've defined the new candidate key, remember to mark its name with a "CK" in the table structure.

Primary Keys

By now, you've established all the candidate keys that seem appropriate for every table. Your next task is to establish a *primary* key for each table, which is the most important key of all.

- A *primary key field* exclusively identifies the table throughout the database structure and helps establish relationships with other tables. (You'll learn more about this in Chapter 10.)
- A *primary key value* uniquely identifies a given record within a table and exclusively represents that record throughout the entire database. It also helps to guard against duplicate records.

A primary key must conform to the exact same elements as a candidate key. This requirement is easy to fulfill because you select a primary key from a table's pool of available candidate keys. The process of selecting a primary key is somewhat similar to that of a presidential election. Every four years, several people run for the office of president of the United States. These individuals are known as "candidates" and they have all of the qualifications required to become president. A national election is held, and a single individual from the pool of available presidential candidates is elected to serve as the country's official president. Similarly, you identify each qualified candidate key in the table, run your own election, and select one of them to become the official primary key of the table. You've already identified the candidates, so now it's election time!

Assuming that there is no other marginal preference, here are a couple of guidelines you can use to select an appropriate primary key:

1. *If you have a simple (single-field) candidate key and a composite candidate key, choose the simple candidate key.* It's always best to use a candidate key that contains the least number of fields.
2. *Choose a candidate key that incorporates part of the table name within its own name.* For example, a candidate key with a name such as SALES INVOICE NUMBER is a good choice for the SALES INVOICES table.

Examine the candidate keys and choose one to serve as the primary key for the table. The choice is largely arbitrary—you can choose the one that you believe most accurately identifies the table's subject or the one that is the most meaningful to everyone in the organization. For example, consider the EMPLOYEES table again in Figure 8.5.

Either of the candidate keys you identified within the table could serve as the primary key. You might decide to choose EMPLOYEE ID if everyone in the organization is accustomed to using this number as a means of identifying employees in items such as tax forms and employee benefits programs. The candidate key you ultimately choose becomes the primary key of the table and is governed by the Elements of a Primary Key. These elements are exactly the same as those for the candidate key, and you should enforce them to the letter. For the sake of clarity, here are the Elements of a Primary Key:

Elements of a Primary Key

- It cannot be a multipart field.
- It must contain unique values.
- It cannot contain null values.

- Its value cannot cause a breach of the organization's security or privacy rules.
- Its value is not optional in whole or in part.
- It comprises a minimum number of fields necessary to define uniqueness.
- Its values must uniquely and exclusively identify each record in the table.
- Its value must exclusively identify the value of each field within a given record.
- Its value can be modified only in rare or extreme cases.

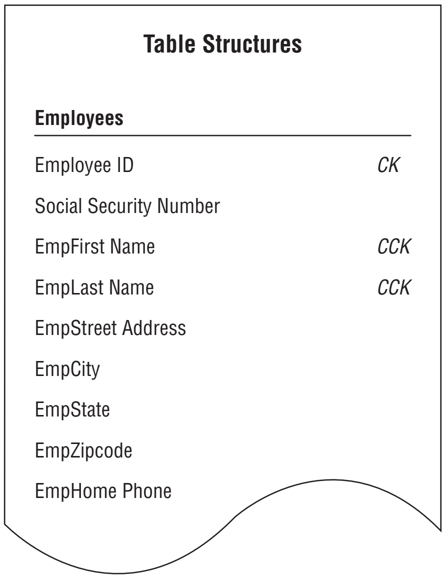

Table Structures

Employees	
Employee ID	*CK*
Social Security Number	
EmpFirst Name	*CCK*
EmpLast Name	*CCK*
EmpStreet Address	
EmpCity	
EmpState	
EmpZipcode	
EmpHome Phone	

Figure 8.5. *Which candidate key should become the primary key of the EMPLOYEES table?*

Before you finalize your selection of a primary key, it is imperative that you make absolutely certain that the primary key fully complies with this particular element:

- Its value must exclusively identify the value of each field within a given record.

Each field value in a given record should be unique throughout the entire database (unless it is participating in establishing a relationship between a pair of tables) and should have *only one* exclusive means of identification—the specific primary key value for that record.

You can determine whether a primary key fully complies with this element by following these steps:

1. Load the table with sample data.
2. Select a record for test purposes and note the current primary key value.
3. Examine the value of the first field (the one immediately after the primary key) and ask yourself this question:

   Does this primary key value *exclusively* identify the current value of <*fieldname*>?

   a. If the answer is yes, move to the next field and repeat the question.

   b. If the answer is no, *remove the field from the table*, move to the next field and repeat the question.
4. Continue this procedure until you've examined every field value in the record.

A field value that the primary key *does not* exclusively identify indicates that the field itself is *unnecessary* to the table's structure; therefore, you should remove the field and reconfirm that the table complies with the Elements of the Ideal Table. You can then add the field you just removed to another table structure, if appropriate, or you can discard it completely because it is truly unnecessary.

Here's an example of how you might apply this technique to the partial table structure in Figure 8.6. (Note that Invoice Number is the primary key of the table.)

Sales Invoices

Invoice Number	Invoice Date	CustFirst Name	CustLast Name	EmpFirst Name	EmpLast Name	EmpHome Phone
13000	06/15/02	Frank	DeSoto	Estela	Pundt	363-9948
13001	06/15/02	Gregory	Mattson	Katherine	Erlich	322-6992
13002	06/15/02	Caroline	Coie	Kendra	Bonnicksen	527-4992
13003	06/16/02	David	Cunningham	Kendra	Bonnicksen	336-5992
13004	06/16/02	Caroline	Coie	Shannon	McLain	572-9948
13005	06/17/02	Frank	DeSoto	Estela	Pundt	322-6992

Figure 8.6. *Does the primary key exclusively identify the value of each field in this table?*

First, you load the table with sample data. You then select a record for test purposes—we'll use the third record for this example—and note the value of the primary key (13002). Now, pose the question above for each field value in the record.

> Does this primary key value *exclusively* identify the current value of . . .

Invoice Date?	Yes, it does. This invoice number will always identify the specific date that the invoice was created.
CustFirst Name?	Yes, it does. This invoice number will always identify the specific first name of the particular customer who made this purchase.

CustLast Name?	Yes, it does. This invoice number will always identify the specific last name of the particular customer who made this purchase.
EmpFirst Name?	Yes, it does. This invoice number will always identify the specific first name of the particular employee who served the customer for this sale.
EmpLast Name?	Yes, it does. This invoice number will always identify the specific last name of the particular employee who served the customer for this sale.
EmpHome Phone?	*No*, it doesn't! The invoice number *indirectly* identifies the employee's home phone number via the employee's name. In fact, it is the *current value* of both EmpFirst Name and EmpLast Name that exclusively identifies the value of EmpHome Phone—change the employee's name and you *must* change the phone number as well. You should now remove EmpHome Phone from the table for two reasons: The primary key does not exclusively identify its current value and (as you've probably already ascertained) it is an unnecessary field. As it turns out, you can discard this field completely because it is already part of the EMPLOYEES table structure.

After you've removed the unnecessary fields you identified during this test, examine the revised table structure and make sure it complies with the Elements of the Ideal Table.

The primary key should now exclusively identify the values of the remaining fields in the table. This means that the primary key is truly sound and you can designate it as the official primary key for the table. Remove the "CK" next to the field name in the table structure and replace it with a "PK." (A primary key composed of two or more fields is known as a *composite primary key*, and you mark it with the letters

"CPK.") Figure 8.7 shows the revised structure of the SALES INVOICE table with INVOICE NUMBER as its primary key.

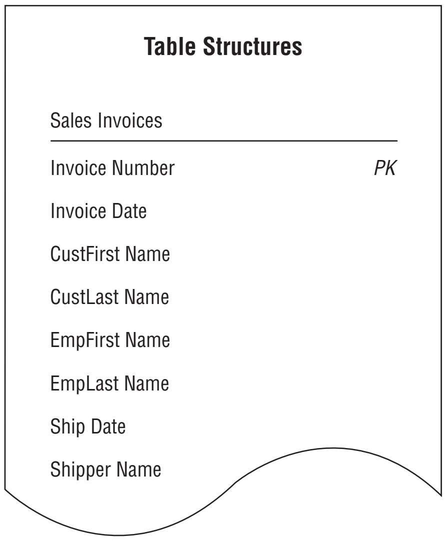

Table Structures

Sales Invoices	
Invoice Number	*PK*
Invoice Date	
CustFirst Name	
CustLast Name	
EmpFirst Name	
EmpLast Name	
Ship Date	
Shipper Name	

Figure 8.7. *The revised SALES INVOICES table with its new primary key.*

As you create a primary key for each table in the database, keep these two rules in mind:

Rules for Establishing a Primary Key

1. *Each table must have one—and only one—primary key.* Because the primary key *must* conform to each of the elements that govern it, only one primary key is necessary for a particular table.
2. *Each primary key within the database must be unique—no two tables should have the same primary key unless one of them is a subset table.* You learned at the beginning of this section that the primary key exclusively identifies a table throughout the database structure; therefore, each table must have its own *unique* primary

key in order to avoid any possible confusion or ambiguity concerning the table's identity. A subset table is excluded from this rule because it represents a more specific version of a particular data table's subject—both tables *must* share the same primary key.

Later in the database-design process, you'll learn how to use the primary key to help establish a relationship between a pair of tables.

Alternate Keys

Now that you've selected a candidate key to serve as the primary key for a particular table, you'll designate the remaining candidate keys as *alternate* keys. These keys can be useful to you in an RDBMS program because they provide an alternative means of uniquely identifying a particular record within the table. If you choose to use an alternate key in this manner, mark its name with "AK" or "CAK" (composite alternate key) in the table structure; otherwise, remove its designation as an alternate key and simply return it to the status of a normal field. You won't be concerned with alternate keys for the remainder of the database-design process, but you will work with them once again as you implement the database in an RDBMS program. (Implementing and using alternate keys in RDBMS programs is beyond the scope of this work—our only objective here is to designate them as appropriate. This is in line with the focus of the book, which is the logical design of a database.)

Figure 8.8 shows the final structure for the EMPLOYEES table with the proper designation for both the primary key and the alternate keys.

Non-keys

A *non-key* is a field that does not serve as a *candidate*, *primary*, *alternate*, or *foreign* key. Its sole purpose is to represent a characteristic of the table's subject, and its value is determined by the primary key.

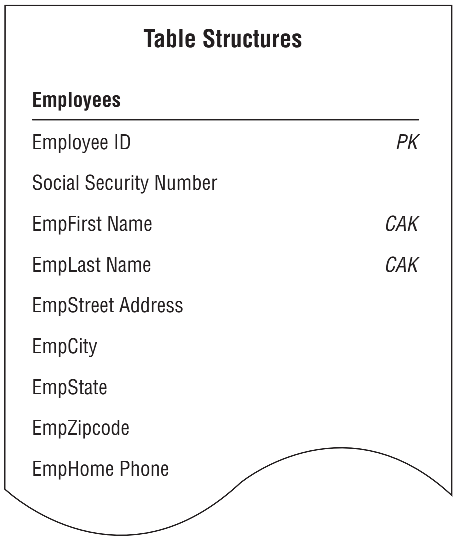

Table Structures

Employees	
Employee ID	*PK*
Social Security Number	
EmpFirst Name	*CAK*
EmpLast Name	*CAK*
EmpStreet Address	
EmpCity	
EmpState	
EmpZipcode	
EmpHome Phone	

Figure 8.8. *The EMPLOYEES table with designated primary and alternate keys.*

There is no particular designation for a non-key, so you don't need to mark it in the table structure.

Table-Level Integrity

This type of integrity is a major component of overall data integrity, and it ensures the following:

- There are no duplicate records in a table.
- The primary key exclusively identifies each record in a table.
- Every primary key value is unique.
- Primary key values are not null.

You began establishing table-level integrity when you defined a primary key for each table and ensured its enforcement by making absolutely certain that each primary key fully complied with the Elements of a Primary Key. In the next chapter, you'll enhance the table's integrity further as you establish *field specifications* for each field within the table.

Reviewing the Initial Table Structures

Now that the fundamental table definitions are complete, you need to conduct interviews with users and management to review the work you've done so far. This set of interviews is fairly straightforward and should be relatively easy to conduct.

During these interviews, you will accomplish these tasks:

- *Ensure that the appropriate subjects are represented in the database.* Although it's highly unlikely that an important subject is missing at this stage of the database-design process, it can happen. When it does happen, identify the subject, use the proper techniques to transform it into a table, and develop it to the same degree as the other tables in the database.
- *Make certain that the table names and table descriptions are suitable and meaningful to everyone.* When a name or description appears to be confusing or ambiguous to several people in the organization, work with them to clarify the item as much as possible. It's common for some table names and descriptions to improve during the interview process.
- *Make certain that the field names are suitable and meaningful to everyone.* Selecting field names typically generates a great deal of discussion, especially when there is an existing database in place. You'll commonly find people who customarily refer to a particular field by a certain name because "that's what it's called on my screen." When you change a field name—you have good reasons for doing so—you must diplomatically explain to these folks that you

renamed the field so that it conforms to the standards imposed by the new database. You can also tell them that the field can appear with the more familiar name once the database is implemented in an RDBMS program. What you've said is true; many RDBMSs allow you to use one name for the field's physical definition and another name for display purposes. This feature, however, does not change, reduce, or negate the need for you to follow the guidelines for creating field names that you learned in Chapter 7.

- *Verify that all the appropriate fields are assigned to each table.* This is your best opportunity to make certain that all of the necessary characteristics pertaining to the subject of the table are in place. You'll commonly discover that you accidentally overlooked one or two characteristics earlier in the design process. When this happens, identify the characteristics, use the appropriate techniques to transform them into fields, and follow all the necessary steps to add them to the table.

When you've completed the interviews, you'll move to the next phase of the database-design process and establish *field specifications* for every field in the database.

CASE STUDY

It's now time to establish keys for each table in the Mike's Bikes database. As you know, your first order of business is to establish candidate keys for each table. Let's say you decide to start with the CUSTOMERS table in Figure 8.9.

As you review each field, you try to determine whether it conforms to the Elements of a Candidate Key. You determine that Status, CustHome Phone, and the combination of CustFirst Name and CustLast Name are potential candidate keys, but you're not quite certain whether any of

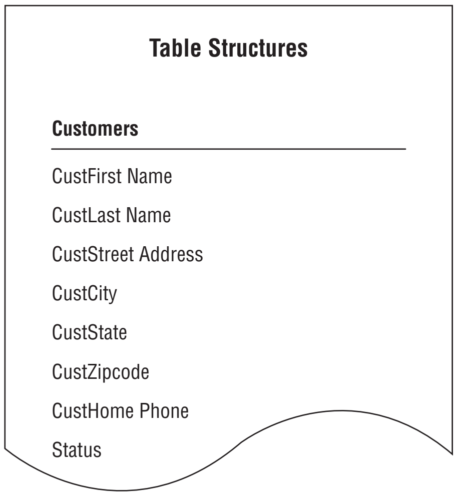

Table Structures

Customers

CustFirst Name
CustLast Name
CustStreet Address
CustCity
CustState
CustZipcode
CustHome Phone
Status

Figure 8.9. *The CUSTOMERS table structure in the Mike's Bikes database.*

them will completely conform to all of the elements. So you decide to test the keys by loading the table with sample data as shown in Figure 8.10.

Customers

CustFirst Name	CustLast Name	CustStreet Address	CustCity	CustState	CustZipcode	CustHome Phone	Status
Bridget	Berlin	2121 NE 35th	Bellevue	WA	98004	422-4982	Valued
Phillip	Bradley	101 9th Avenue	Kent	WA	98126	322-1178	
Kel	Brigan	7525 Taxco Lane	Redmond	WA	98225	363-9360	Valued
Barbara	Carmichael	7525 Taxco Lane	Redmond	WA	98225	363-9360	Preferred
Daniel	Chavez	750 Pike Street	Bothell	WA	98001	441-3987	Valued
Daniel	Chavez	301 N Main	Seattle	WA	98115	365-7199	
Sandi	Cooper	115 Pine Place	Seattle	WA	98026	332-0499	Preferred

Figure 8.10. *Testing candidate keys in the CUSTOMERS table.*

Always remember that a field must comply with *all* of the Elements of a Candidate Key in order to qualify as a candidate key. You must immediately disqualify the field if it does not fulfill this requirement.

As you examine the table, you draw these conclusions:

- *Status is ineligible because it will probably contain duplicate values.* As business grows, Mike is going to have many "Valued" customers.
- *CustHome Phone is ineligible because it will probably contain duplicate values.* The sample data reveals that two customers can live in the same residence and have the same phone number.
- *CustFirst Name and CustLast Name are ineligible because they will probably contain duplicate values.* The sample data reveals that the combination of first name and last name can represent more than one distinct customer.

These findings convince you to establish an artificial candidate key for this table. You then create a field called Customer ID, confirm that it complies with the requirements for a candidate key, and add the new field to the table structure with the appropriate designation.

Figure 8.11 shows the revised structure of the CUSTOMERS table.

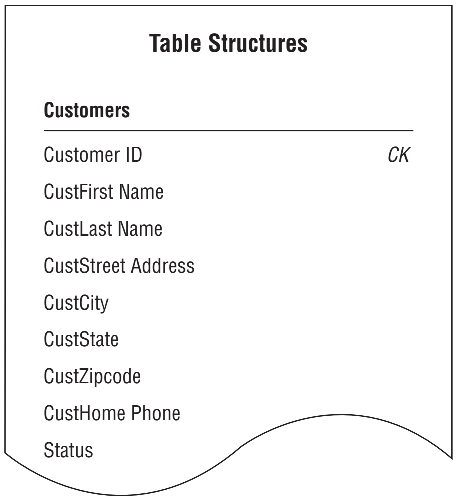

Table Structures

Customers

Customer ID	*CK*
CustFirst Name	
CustLast Name	
CustStreet Address	
CustCity	
CustState	
CustZipcode	
CustHome Phone	
Status	

Figure 8.11. *The CUSTOMERS table with the new artificial candidate key, Customer ID.*

Now you'll repeat this procedure for each table in the database. Remember to make certain that every table has at least *one* candidate key.

The next order of business is to establish a primary key for each table. As you know, you select the primary key for a particular table from the table's pool of available candidate keys. Here are a few points to keep in mind when you're choosing a primary key for a table with more than one candidate key:

- Choose a simple (single-field) candidate key over a composite candidate key.
- If possible, pick a candidate key that has the table name incorporated into its own name.
- Select the candidate key that best identifies the subject of the table or is most meaningful to everyone in the organization.

You begin by working with the EMPLOYEES table in Figure 8.12. As you review the candidate keys, you decide that EMPLOYEE NUMBER is a much better choice for a primary key than the combination of EMPFIRST NAME and EMPLAST NAME because Mike's employees are already accustomed to identifying themselves by their assigned numbers. Using EMPLOYEE NUMBER makes perfect sense, so you select it as the primary key for the table.

Now you perform one final task before you designate EMPLOYEE NUMBER as the official primary key of the table: You make absolutely certain that it exclusively identifies the value of each field within a given record. So, you test EMPLOYEE NUMBER by following these steps:

1. Load the EMPLOYEES table with sample data.
2. Select a record for test purposes and note the current value of EMPLOYEE NUMBER.
3. Examine the value of the first field (the one immediately after EMPLOYEE NUMBER) and ask yourself this question:

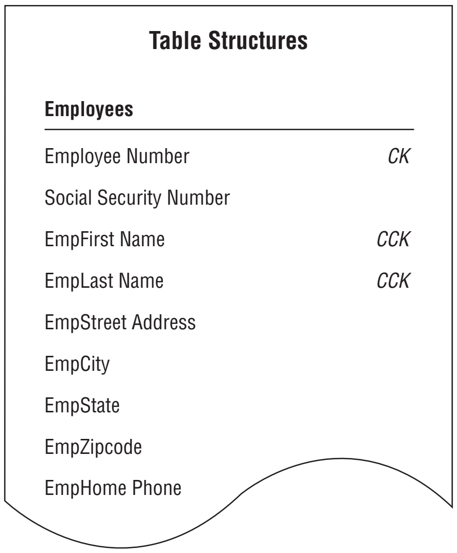

Table Structures

Employees	
Employee Number	*CK*
Social Security Number	
EmpFirst Name	*CCK*
EmpLast Name	*CCK*
EmpStreet Address	
EmpCity	
EmpState	
EmpZipcode	
EmpHome Phone	

Figure 8.12. *The EMPLOYEES table structure in the Mike's Bikes database.*

Does this primary key value *exclusively* identify the current value of <*fieldname*>?

a. If the answer is yes, move to the next field and repeat the question.

b. If the answer is no, *remove the field from the table*, move to the next field and repeat the question. (Be sure to determine whether you can add the field you just removed to another table structure, if appropriate, or discard it completely because it is truly unnecessary.)

4. Continue this procedure until you've examined every field value in the record.

You know that you'll have to remove any field containing a value that EMPLOYEE NUMBER *does not* exclusively identify. EMPLOYEE NUMBER does exclusively identify the value of each field in the test record, however, so you use it as the official primary key for the EMPLOYEES table and mark its name with the letters “PK” in the table structure. You then repeat this process with the rest of the tables in Mike’s new database until every table has a primary key.

Remember to keep these rules in mind as you establish primary keys for each table:

- Each table must have one—and only one—primary key.
- Each primary key within the database should be unique—no two tables should have the same primary key (unless one of them is a subset table).

As you work through the tables in Mike’s database, you remember that the SERVICES table is a subset table. You created it during the previous stage of the design process (in Chapter 7), and it represents a more specific version of the subject represented by the PRODUCTS table. The PRODUCT NAME field is what currently relates the PRODUCTS table to the SERVICES subset table. You now know, however, that a subset table *must* have the same primary key as the table to which it is related, so you’ll use PRODUCT NUMBER (the primary key of the PRODUCTS table) as the primary key of the SERVICES table. Figure 8.13 shows the PRODUCTS and SERVICES tables with their primary keys.

The last order of business is to conduct interviews with Mike and his staff and review all the work you’ve performed on the tables in the database. As you conduct these interviews, make certain you check the following:

- That the appropriate subjects are represented in the database
- That the table names and descriptions are suitable and meaningful to everyone
- That the field names are suitable and meaningful to everyone
- That all the appropriate fields are assigned to each table

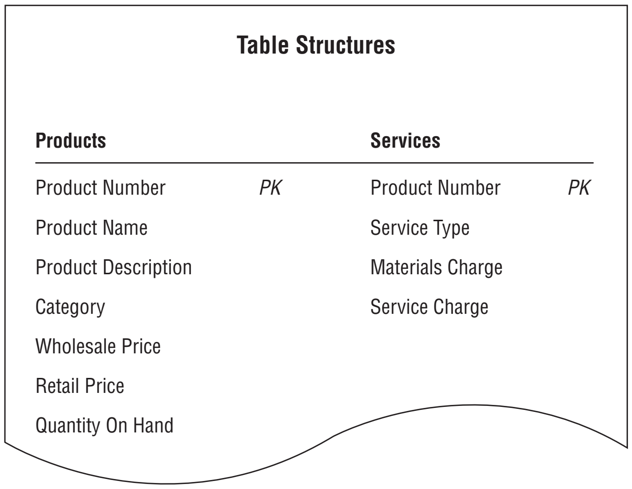

Table Structures

Products		**Services**	
Product Number	*PK*	Product Number	*PK*
Product Name		Service Type	
Product Description		Materials Charge	
Category		Service Charge	
Wholesale Price			
Retail Price			
Quantity On Hand			

Figure 8.13. *Establishing the primary key for the SERVICES subset table.*

By the end of the interview, everyone agrees that the tables are in good form and that all the subjects with which they are concerned are represented in the database. Only one minor point came up during the discussions: Mike wants to add a CALL PRIORITY field to the VENDORS table. There are instances in which more than one vendor supplies a particular product, and Mike wants to create a way to indicate which vendor he should call first if that product is unexpectedly out of stock. So, you add the new field to the VENDORS table and bring the interview to a close.

Summary

The chapter opened with a discussion of the importance of *keys*. You learned that there are different types of keys, and each type plays a different role within the database. Each key performs a particular function, such as uniquely identifying records, establishing various types of integrity, and establishing relationships between tables. You now know that you can guarantee sound table structure by making certain that the appropriate keys are established for each table.

We then discussed the process of establishing keys for each table. We began by identifying the four main types of keys: *candidate*, *primary*, *foreign*, and *non-keys*. First, we looked at the process of establishing candidate keys for each table. You learned about the Elements of a Candidate Key and how to make certain that a field (or set of fields) complies with these elements. Then you learned that you can create and use an artificial candidate key when none of the fields in a table can serve as a candidate key or when a new field would make a stronger candidate key than any of the existing candidate key fields.

The chapter continued with a discussion of *primary keys*. You learned that you select a primary key from a table's pool of candidate keys and that the primary key is governed by a set of specific elements. We then covered a set of guidelines that help you determine which candidate key to use as a primary key. Next, you learned how to ensure that the chosen primary key exclusively identifies a given record and its set of field values. When the primary key does not exclusively identify a particular field value, you know that you must remove the field from the table in order to ensure the table's structural integrity. You also know that each table must have a single, unique primary key.

You then learned that you designate any remaining candidate keys as *alternate keys*. These keys will be most useful to you when you implement the database in an RDBMS program because they provide an alternate means of identifying a given record. We then discussed the *non-key* field, which is any field not designated as a candidate, primary, alternate, or foreign key. You now know that a non-key field represents a characteristic of the table's subject and that the primary key exclusively identifies its value.

Table-level integrity was the next subject of discussion, and you learned that it is established through the use of primary keys and enforced by the Elements of a Primary Key.

The chapter closed with some guidance on conducting further *interviews* with users and management. You now know that these interviews provide you with a means of reviewing the work you have performed on the tables and help you to verify and validate the current database structure.

Review Questions

1. State the three reasons why keys are important.
2. What are the four main types of *keys*?
3. What is the purpose of a *candidate key*?
4. State four items of the *Elements of a Candidate Key*.
5. True or False: A candidate key can be composed of more than one field.
6. Can a table have more than one candidate key?
7. What is an *artificial* candidate key?
8. What is the most important key you assign to a table?
9. Why is this key important?
10. How do you establish a *primary key*?
11. State four items of the *Elements of a Primary Key*.
12. What must you do before you finalize your selection of a primary key?
13. What is an *alternate key*?
14. What do you ensure by establishing table-level integrity?
15. Why should you review the initial table structures?