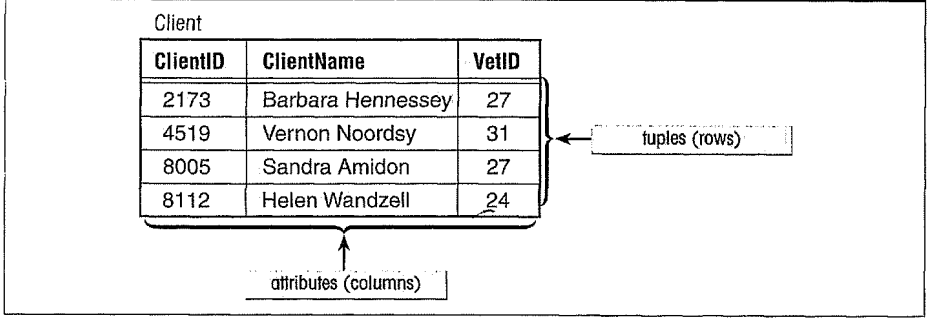

Adamski, J.J., Finnegan, K.T., Hommel, C., 2001, *Microsoft 2000, Comprehensive Enhanced,* Course Technology Thomson Learning, USA Massachusetts.

Pages copied: RD2-RD8

Relations

A relational database stores its data in tables. A **table** is a two-dimensional structure made up of rows and columns. The terms table, row, and column are the popular names for the more formal terms **relation** (table), **tuple** (row), and **attribute** (column), as shown in Figure 1.

Figure 1 A RELATION CONSISTING OF TUPLES AND ATTRIBUTES



The Client table shown in Figure 1 is an example of a relation, a two-dimensional structure with the following characteristics:

- Each row is unique. Because no two rows are the same, you can easily locate and update specific data. For example, you can locate the row for ClientID 8005 and change the ClientName value, Sandra Amidon, or the VetID value, 27.
- The order of the rows is unimportant. You can add or view rows in any order. For example, you can view the rows in ClientName order instead of ClientID order.
- Each table entry contains a single value. At the intersection of each row and column, you cannot have more than one value. For example, each row in Figure 1 contains one ClientID, one ClientName, and one VetID.
- The order of the columns is unimportant. You can add or view columns in any order.
- Each column has a unique name called the **attribute name**. The attribute name allows you to access a specific column without needing to know its position within the relation.
- The entries in a column are from the same domain. A **domain** is a set of values from which one or more columns draw their actual values. A domain can be broad, such as "all legitimate names of people" for the ClientName column, or narrow, such as "24, 27, or 31" for the VetID column. The domain of "all legitimate dates" could be shared by the BirthDate, StartDate, and LastPayDate columns in a company's employee relation.
- Each row in a relation describes, or shows the characteristics of, an entity. An **entity** is a person, place, object, event, or idea for which you want to store and process data. For example, ClientID, ClientName, and VetID are characteristics of the clients of a pet-sitting company. The Client relation represents all the client entities and their characteristics. That is, the sets of values in the rows of the tblClient relation describe the different clients of the company. The Client

relation includes only characteristics of a client. Other relations would exist for the company's other entities. For example, a Pet relation might describe the clients' pets and an Employee relation might describe the company's employees.

Knowing the characteristics of a relation leads directly to a definition of a relational database. A **relational database** is a collection of relations.

Keys

Primary keys ensure that each row in a relation is unique. A **primary key** is an attribute, or a collection of attributes, whose values uniquely identify each row in a relation. In addition to being *unique*, a primary key must be *minimal* (that is, contain no unnecessary extra attributes) and must not change in value. For example, in Figure 2 the State relation contains one record per state and uses StateAbbrev as its primary key.

Figure 2 A RELATION AND ITS KEYS

alternate key

primary key

State

StateAbbrev	StateName	EnteredUnionOrder	StateBird	StatePopulation
CT	Connecticut	5	American robin	3,287,116
MI	Michigan	26	robin	9,295,297
SD	South Dakota	40	pheasant	696,004
TN	Tennessee	16	mockingbird	4,877,185
TX	Texas	28	mockingbird	16,986,510

Could any other attribute, or collection of attributes, be the primary key of the State relation?

- Could StateBird serve as the primary key? No, because the column does not have unique values (for example, the mockingbird is the state bird of more than one state).
- Could StatePopulation serve as the primary key? No, because the column values change periodically and are not guaranteed to be unique.
- Could StateAbbrev and StateName together serve as the primary key? No, because the combination is not minimal. Something less, StateAbbrev by itself, can serve as the primary key.

- Could StateName serve as the primary key? Yes, because the column has unique values. In a similar way, you could select EnteredUnionOrder as the primary key for the State relation. One attribute, or collection of attributes, that can serve as a primary key is called a **candidate key**. The candidate keys for the State relation are StateAbbrev, StateName, and EnteredUnionOrder. You choose one of the candidate keys to be the primary key, and the remaining candidate keys are called **alternate keys**.

Figure 3 shows a City relation containing the attributes StateAbbrev, CityName, and CityPopulation.

Figure 3 A RELATION WITH A COMPOSITE KEY

City

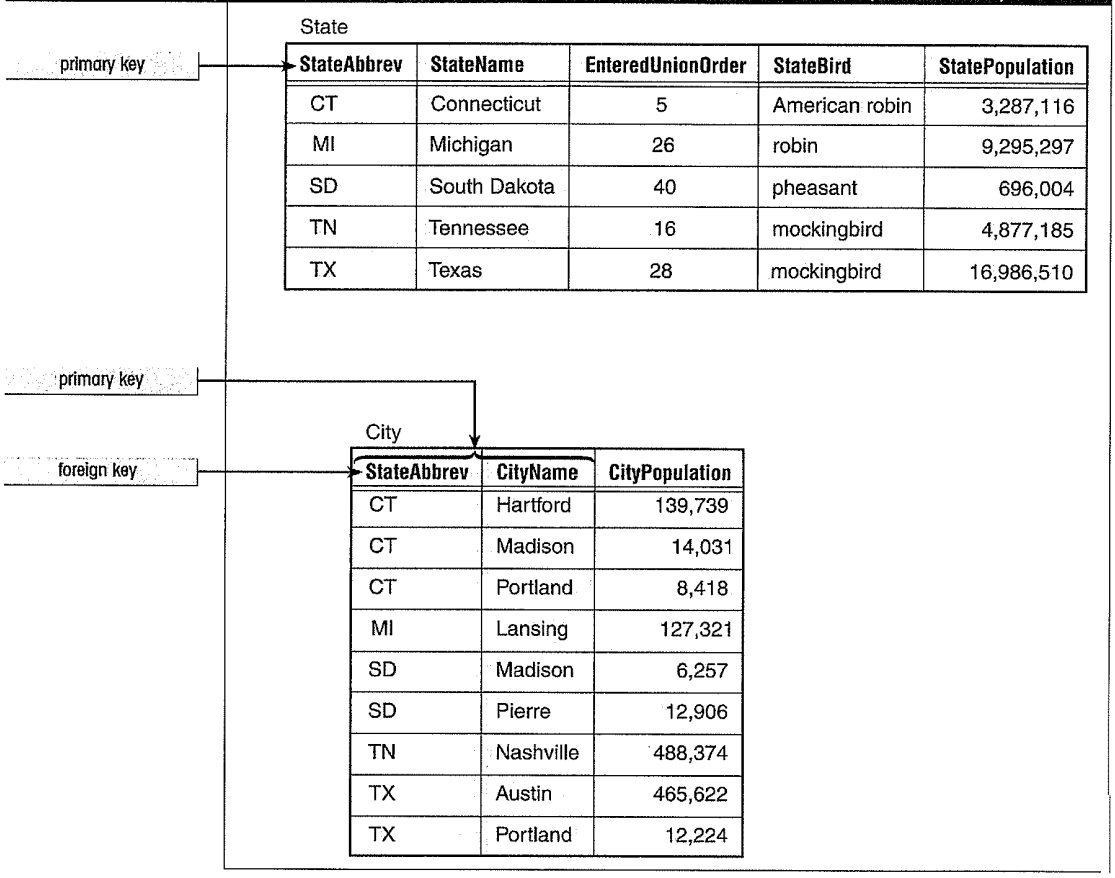
primary key

StateAbbrev	CityName	CityPopulation
CT	Hartford	139,739
CT	Madison	14,031
CT	Portland	8,418
MI	Lansing	127,321
SD	Madison	6,257
SD	Pierre	12,906
TN	Nashville	488,374
TX	Austin	465,622
TX	Portland	12,224

What is the primary key for the City relation? The values for CityPopulation periodically change and are not guaranteed to be unique, so CityPopulation cannot be the primary key. Because the values for each of the other two columns are not unique, StateAbbrev alone cannot be the primary key and neither can CityName (for example, there are two Madisons and two Portlands). The primary key is the combination of StateAbbrev and CityName. Both attributes together are needed to identify, uniquely and minimally, each row in the City relation. A multiple-attribute primary key is called a **composite key** or a **concatenated key**.

The StateAbbrev attribute in the City relation is also a **foreign key**. A **foreign key** is an attribute, or a collection of attributes, in one relation whose values must match the values of the primary key of some relation. As shown in Figure 4, the values in the City relation's StateAbbrev column match the values in the State relation's StateAbbrev column. Thus, StateAbbrev, the primary key of the State relation, is a foreign key in the City relation. Although the attribute name StateAbbrev is the same in both relations, the names could be different. Most people give the same name to an attribute stored in two or more tables to broadcast clearly they are really the same attribute.

Figure 4 STATEABBREV AS A PRIMARY KEY (STATE RELATION) AND A FOREIGN KEY (CITY RELATION)



State

StateAbbrev	StateName	EnteredUnionOrder	StateBird	StatePopulation
CT	Connecticut	5	American robin	3,287,116
MI	Michigan	26	robin	9,295,297
SD	South Dakota	40	pheasant	696,004
TN	Tennessee	16	mockingbird	4,877,185
TX	Texas	28	mockingbird	16,986,510

City

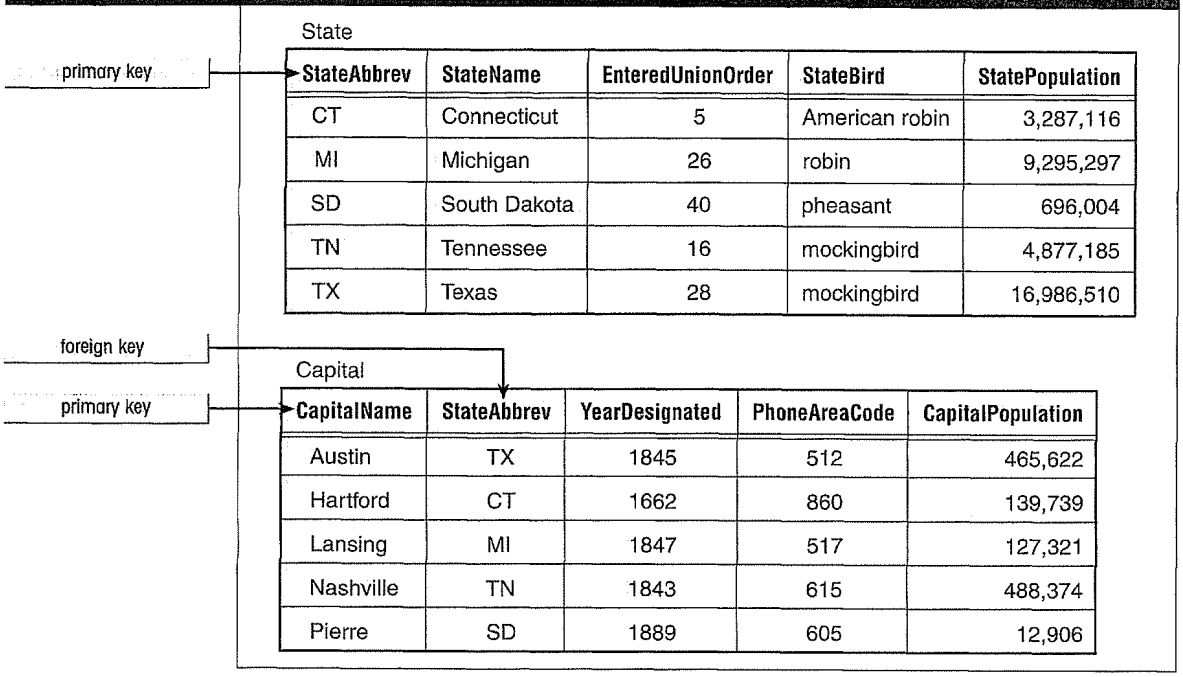
StateAbbrev	CityName	CityPopulation
CT	Hartford	139,739
CT	Madison	14,031
CT	Portland	8,418
MI	Lansing	127,321
SD	Madison	6,257
SD	Pierre	12,906
TN	Nashville	488,374
TX	Austin	465,622
TX	Portland	12,224

A **nonkey attribute** is an attribute that is not part of the primary key. In the two relations shown in Figure 4, all attributes are nonkey attributes except StateAbbrev in the State and City relations and CityName in the City relation. *Key* is an ambiguous word because it can refer to a primary, candidate, alternate, or foreign key. When the word key appears alone, however, it means primary key and the definition for a nonkey attribute consequently makes sense.

Relationships

The Capital relation, shown in Figure 5, has one row for each state capital. The CapitalName and StateAbbrev attributes are candidate keys; selecting CapitalName as the primary key makes StateAbbrev an alternate key. The StateAbbrev attribute in the Capital relation is also a foreign key, because its values match the values in the State relation's StateAbbrev column.

Figure 5 A ONE-TO-ONE RELATIONSHIP



One-to-One

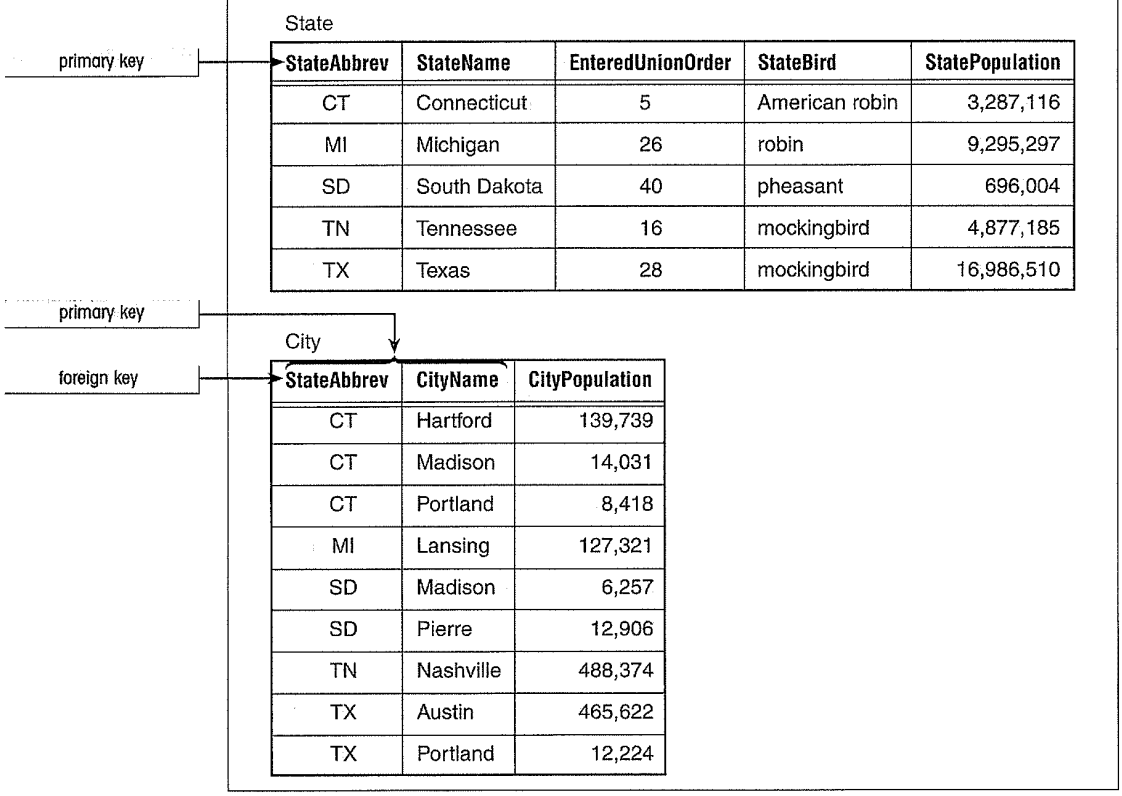
The State and Capital relations, shown in Figure 5, have a one-to-one relationship. A **one-to-one relationship** (abbreviated 1:1) exists between two relations when each row in one relation has at most one matching row in the other relation. StateAbbrev, which is a foreign key in the Capital relation and the primary key in the State relation, is the common field that ties together the rows of each relation.

Should the State and Capital relations be combined into one relation? Although the two relations in any 1:1 relationship can be combined into one relation, each relation describes different entities and should usually be kept separate.

One-to-Many

The State and City relations, shown once again in Figure 6, have a one-to-many relationship. A **one-to-many relationship** (abbreviated 1:M) exists between two relations when one row in the first relation matches many rows in the second relation and one row in the second relation matches only one row in the first relation. Many can mean zero rows, one row, or two or more rows. StateAbbrev, which is a foreign key in the City relation and the primary key in the State relation, is the common field that ties together the rows of each relation.

Figure 6 A ONE-TO-MANY RELATIONSHIP



State

StateAbbrev	StateName	EnteredUnionOrder	StateBird	StatePopulation
CT	Connecticut	5	American robin	3,287,116
MI	Michigan	26	robin	9,295,297
SD	South Dakota	40	pheasant	696,004
TN	Tennessee	16	mockingbird	4,877,185
TX	Texas	28	mockingbird	16,986,510

City

StateAbbrev	CityName	CityPopulation
CT	Hartford	139,739
CT	Madison	14,031
CT	Portland	8,418
MI	Lansing	127,321
SD	Madison	6,257
SD	Pierre	12,906
TN	Nashville	488,374
TX	Austin	465,622
TX	Portland	12,224

Many-to-Many

In Figure 7, the State relation with a primary key of StateAbbrev and the Crop relation with a primary key of CropName have a many-to-many relationship. A **many-to-many relationship** (abbreviated as M:N) exists between two relations when one row in the first relation matches many rows in the second relation and one row in the second relation matches many rows in the first relation. In a relational database, you must use a third relation to serve as a bridge between the two M:N relations; the third relation has the primary keys of the M:N relations as its primary key. The original relations now each have a 1:M relationship with the new relation. The StateAbbrev and CropName attributes represent the primary key of the Production relation that is shown in Figure 7. StateAbbrev, which is a foreign key in the Production relation and the primary key in the State relation, is the common field that ties together the rows of the State and Production relations. Likewise, CropName is the common field for the Crop and Production relations.

Figure 7 A MANY-TO-MANY RELATIONSHIP

primary key

State

StateAbbrev	StateName	EnteredUnionOrder	StateBird	StatePopulation
CT	Connecticut	5	American robin	3,287,116
MI	Michigan	26	robin	9,295,297
SD	South Dakota	40	pheasant	696,004
TN	Tennessee	16	mockingbird	4,877,185
TX	Texas	28	mockingbird	16,986,510

Crop

CropName	Exports	Imports
Corn	$4,965.8	$68.5
Cotton	$2,014.6	$11.4
Soybeans	$4,462.8	$15.8
Wheat	$4,503.2	$191.1

primary key

foreign key

Production

StateAbbrev	CropName	Quantity
MI	Corn	241,500
MI	Soybeans	47,520
MI	Wheat	35,280
SD	Corn	377,200
SD	Soybeans	63,000
SD	Wheat	119,590
TN	Corn	79,360
TN	Soybeans	33,250
TN	Wheat	13,440
TX	Corn	202,500
TX	Cotton	3,322
TX	Soybeans	12,870
TX	Wheat	129,200