Joining and relating tables

In an RDBMS and in a GIS, tables are commonly combined using a **join**, in order to bring different sets of information together. The tables are combined using a common field called a **key**. The key field must be of the same data type in both tables. When a join is performed, the two separate tables become one and contain the information from both tables (Fig. 6.2). The join is a temporary relationship and may be removed when it is no longer needed.

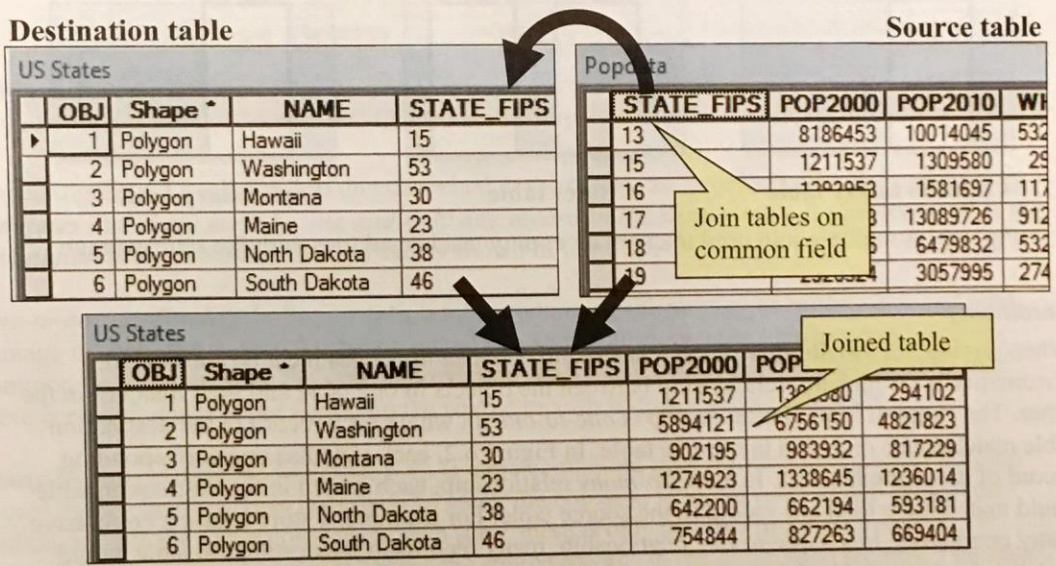

Fig. 6.2. Joining on a common field. These two tables are joined into one using the common field STATE_FIPS. The data from the source table are placed into the destination table.

Joins have a direction. The table containing the information to be appended is called the **source table**. The table that receives the appended information is called the **destination table**. In Figure 6.2, the destination table, US States, is a feature attribute table. The source table containing demographic data is a standalone table. When the two tables are joined, the

demographic data are appended to the US States attribute table. Often joins are used to bring data from standalone tables into an attribute table for mapping or analysis.

Join direction matters. Once the demographics are joined to the table of the states shapefile, then the information could be used to make a map of the population data. If, however, the join had been performed in the opposite direction, with the demographics as the destination and the states as the source, then the resulting table would be a standalone table. In this case, a map showing the demographics could not be made because the demographics table is not a feature class.

Multiple joins can combine three or more tables. In ArcGIS, the destination table can have multiple source tables joined to it; each join is initiated with the same destination table but a different source. Source fields in the first join can be used as key fields in the next join. However, a table cannot simultaneously be a source for one join and a destination for another.

Suppose that you wish to compare the average teacher salary in counties to the state average (Fig. 6.3) for all counties in the United States. You have a table of county salaries and one of state salaries, but one has a state name field and the other has a state abbreviation field. You can't join them directly. However, the states table has both a name and abbreviation field, so you do a multiple join. The counties table is the destination both times. First you join the states to the counties table using the StateName field as the key; then you can join the teacher salary table to the counties using the StateAbbr field.

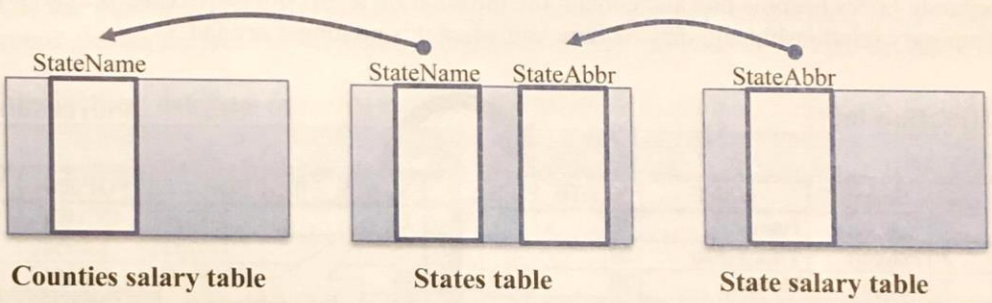

Fig. 6.3. A multiple join used to compare county teacher salaries with the state average

Cardinality

When joining, the **cardinality** of the relationship between the tables must be considered. Cardinality is the numeric relationship between the objects in one table and their matches in the other. The simplest kind of relationship is *one-to-one*, in which each record in the destination table matches one record in the source table. In Figure 6.2, each state has one corresponding record of demographic data. In a *one-to-many* relationship, each record in the destination table could match more than one record in the source table. For example, a store location could have many employees. In a *many-to-one* relationship, many records in the destination table would match a single record in the source table, such as many cities falling within one state. Finally, a *many-to-many* relationship indicates that multiple records can appear in both tables. For example, a student may take more than one class, and most classes have more than one student.

The direction of the join must be taken into account when ascertaining the cardinality of a relationship. The destination table is the point of reference and comes first; that is, the relationship cardinality is reported as {destination} to {source}. Imagine two tables containing *states* and *counties*. If one performs a join with *states* as the destination table and *counties* as the

source, the cardinality is one-to-many because each state contains many counties (and a county may belong to only one state). If the join is reversed and *counties* is the destination table, then the cardinality becomes many-to-one because there are many counties in one state.

TIP: Putting the destination table first when stating cardinality is not a universal convention and may be found reversed in other data management applications. It is used here to match the convention adopted by ESRI in its publications and Help documents. It may help to always imagine the destination table on the left, as in Figure 6.4.

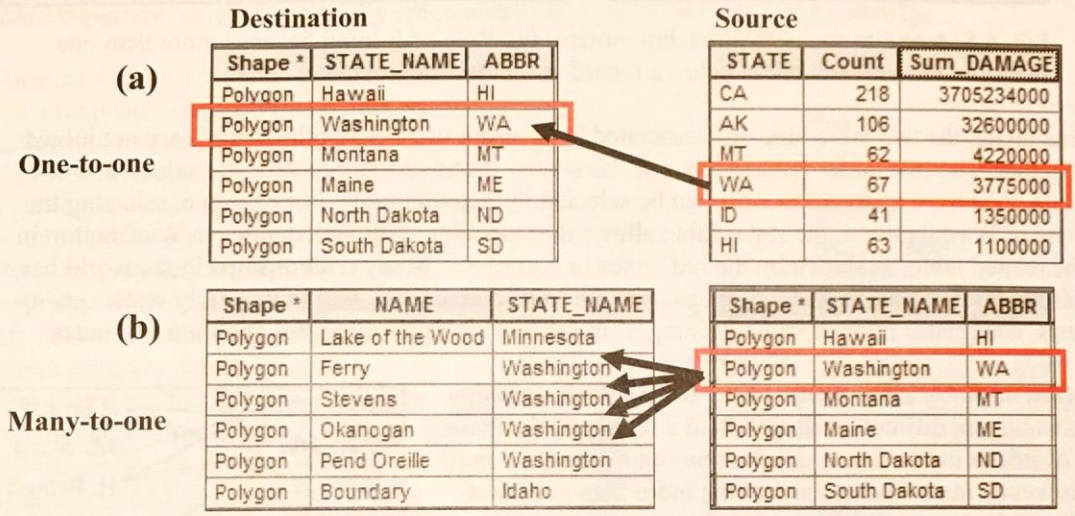

Fig. 6.4. A cardinality of one-to-one or many-to-one permits tables to be combined without violating the Rule of Joining. (a) One-to-one cardinality. (b) Many-to-one cardinality.

The cardinality of a relationship dictates whether the tables can be joined. ***The* Rule of Joining *stipulates that there must be one and only one record in the source table for each record in the destination table.*** Consider the four tables shown in Figure 6.4.

One-to-one cardinality. In Figure 6.4a, a table containing the number of earthquakes and total damage in each state (source) is being joined to a states attribute table (destination), using the common field provided by the state abbreviation. Each state occurs once in each table so that there is no ambiguity about matching the source records to the destination records.

Many-to-one cardinality. In Figure 6.4b, a table containing state information (source) is being joined to a table of counties (destination) using the common field provided by the state name. Although there are many counties in each state, there is only one record for each state in the source table and no ambiguity in linking the source records to the destination records. The state record does get used more than once, but it still does not violate the Rule of Joining.

One-to-many cardinality. Figure 6.5 shows the reverse of the join in Figure 6.4b. Now states are the destination table and counties are the source table. Many county records in the source table match each state record in the destination table. The Rule of Joining is violated, and it becomes ambiguous which county record should be matched to the state. One cannot perform a join if a one-to-many relationship is present. Instead, we perform a different operation called a **relate**.

Destination

states

Shape *	STATE_NAME	STATE_FIPS
Polygon	Hawaii	15
Polygon	Washington	53
Polygon	Montana	30
Polygon	Maine	23
Polygon	North Dakota	38
Polygon	South Dakota	46
Polygon	Wyoming	56

?

Source

counties

Shape *	NAME	STATE_NAME	FIPS
Polygon	Lake of the Wood	Minnesota	27077
Polygon	Ferry	Washington	53019
Polygon	Stevens	Washington	53065
Polygon	Okanogan	Washington	53047
Polygon	Pend Oreille	Washington	53051
Polygon	Boundary	Idaho	16021
Polygon	Lincoln	Montana	30053

Fig. 6.5. A one-to-many relationship violates the Rule of Joining because more than one record in the source table matches a record in the destination table.

In a relate, the two tables are still associated by a common field, but the records are not joined together. The two tables remain separate. However, if one or more records are selected in one table, then the associated records can be selected in the other table. For example, selecting the state of Washington in the states table allows the selection of all the counties in Washington in the related table, as shown by the red boxes in Figure 6.5. Many relationships in the world have cardinalities of one-to-many, such as a school to its classes or a well to its yearly water quality tests, and relates provide valuable support in dealing with those features and their attributes.

Many-to-many cardinality. Many-to-many relationships exist but are difficult to deal with in a relational database. Consider a university. Each class has many students in it. However, most students are taking more than one class (Fig. 6.6). In practice, one would have to use multiple relates to represent the relationships and work with them one at a time. A one-to-many relate could be set up from the classes table to the students to allow all the students in each class to be found. Another one-to-many relate would allow the list of classes for each student to be generated.

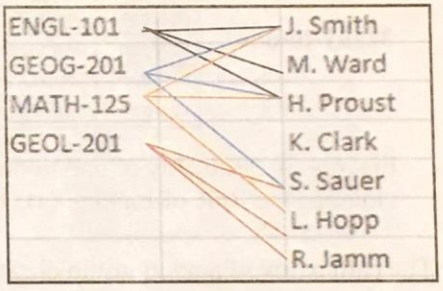

Fig. 6.6. A many-to-many relationship

Sometimes a join is performed in which most records have matches, but some do not. Occasionally, a user makes a fundamental error in crafting the join, and none of the records have matches. In these cases, the join will still be performed, but the records without matches will be given zero or <Null> values in the attribute fields from the source table. In Figure 6.7, an earthquakes table (highlighted) was joined to a states table (white) based on the state abbreviation and a many-to-one cardinality. Earthquakes in Puerto Rico had no match in the states table, and the source fields are given <Null> values in the joined table. It is always a good idea to view the table after a join and confirm that it went as intended. If missing values are encountered, a plan can be made

Earthquakes table fields		States table fields		
STA	LOCATION	STATE_	STATE	POP201
PR	Puerto Rico Region	<Null>	<Null>	<Null>
PR	Puerto Rico Region	<Null>	<Null>	<Null>
PR	Northwest of Puerto Ri	<Null>	<Null>	<Null>
AL	Irondale, Alabama	Alabama	AL	473559
AL	Near Birmingham, Alab	Alabama	AL	473559
AL	Near Birmingham, Alab	Alabama	AL	473559

Fig. 6.7. Result of no match for the destination records (highlighted) in the source table (white)

to handle them. In this case, one might choose to exclude the earthquakes outside the 50 states from further consideration or to find a different states feature class that includes territories.