CHAPTER 5

Attribute data handling

In chapter 2 you have seen the basic concepts of ILWIS for Windows. One of the topics of that chapter dealt with the relationship between spatial data (maps) and attribute data (tables). This chapter is completely dedicated to the use of tabular data in ILWIS.

Before you start it is good to repeat some of the conclusions that were obtained from the first chapter (see figure 5.1):

– An attribute table is linked to a map through its domain.

– An attribute table can only be linked to maps with a class or ID domain.

– An attribute table may contain several columns. Each of these columns can have a class, ID or value domain (or other special domains, such as color, string, bool etc.)

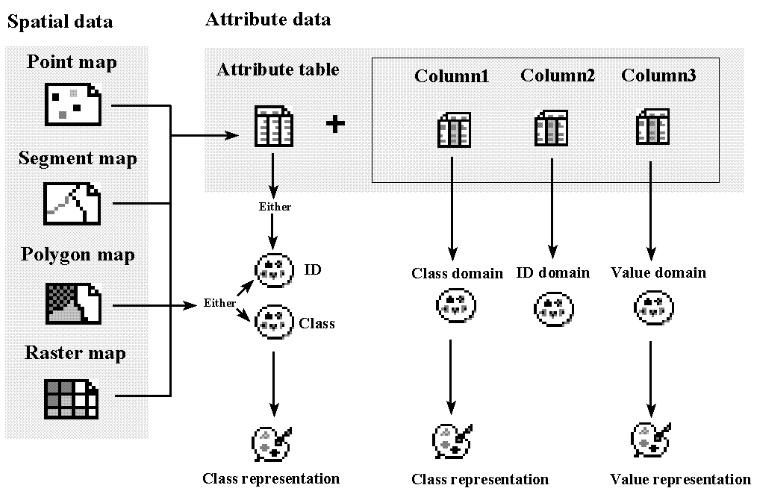


Figure 5.1: Schematic relationship between spatial data (point- segment-, polygon- or raster maps), and attribute data (tables, containing various columns)

In this chapter you will see how you can create a table, how to import it from another software package, how to edit it, and various ways to display table data. The main part of this chapter is dedicated to calculating with table data.

Before you can start with the exercises, you should start up ILWIS and change to the subdirectory c:\ilwis21\data\usrguide\chap05, where the data files for this chapter are stored.

☞

- Double-click the ILWIS program icon in the ILWIS program group.
- Change the working drive and the working directory until you are in the directory c:\ilwis21\data\usrguide\chap05.

The data set: city blocks

Throughout this chapter you will work on a dataset dealing with the city blocks of the central part of the city of Cochabamba, in Bolivia. The city blocks have been digitized from a paper map, and for each building block attribute information is available in a table.

☞

- Open polygon map `Citybl` and click a few different city blocks to know their contents.
- Open domain `Citybl` and look at the contents.
- Close the domain and the map window and open the table `Citybl`.

You will see that each city block is coded by a unique number, so that it can be identified separately and information from each individual block can be obtained from the table connected to the map. The polygon map `Citybl` has a domain *ID (Identifier)*.

The table `Citybl` contains the following 4 columns:

– `Area`: a column with a value domain, containing the area of each city block in square meters.

– `Landuse`: a column with a class domain containing the prevailing land use type within the city block.

`Residential`: city blocks used primarily for housing.

`Commercial`: city blocks primarily containing buildings used for commercial activities such as shops, offices, banks, hotels, restaurants etc.

`Institutional`: city blocks that contain buildings such as schools, universities, hospitals, churches, museums, governmental offices etc.

`Industrial`: city blocks with buildings primarily dedicated to processing, manufacturing, and storage of goods.

`Recreational`: city blocks used for any community or socio-cultural activity, indoor or outdoor, such as sport fields, stadiums, parks etc.

`Transport`: city blocks used for transport related services such as bus station, railway station etc.

`Vacant`: city blocks within the urban area that are not used for any urban activity, such as abandoned buildings, undeveloped plots, bare hills, forested areas etc.

`Water`: includes all water bodies such as lakes and rivers.

`Airport`: the airport of Cochabamba.

– `District`: the city of Cochabamba is divided into a number of districts. Each district contains a number of city blocks. Since the districts have unique codes, the `District` column has also an ID domain.

– `Population`: this is a column with a value domain which contains the number of persons living in each city block. Note that some city blocks have a population of 0 because there are no houses in that block.

☞

- Double-click the titles of the columns and look at the Column properties dialog boxes.
- Close the table `Citybl`.

The data used for this exercise is partly invented. In the table `Citybl` the location of the generalized city blocks, the land use and the cadastral districts reflect the actual situation, but the population column and the data related to the districts that will be shown later have been made up for this exercise.

In the exercises of this chapter you will evaluate the following:

– The population density per building block.

– The total population per district.

– Land use proportions per district.

– The number of schools in relation to the number of schoolchildren in each district.

5.1 Creating a table

Tables can be obtained in two ways: either by creating them in ILWIS and entering the values manually from the keyboard, or by importing existing data in other file formats. You will first see how to create a table in ILWIS: a table linked to the Identifier domain `District`, with information regarding the cadastral districts of the city of Cochabamba, such as:

– `Housecon`: the average dwelling condition, or building quality, within the district. This will be a class domain with the classes bad, moderate and good.

– `Schools`: the number of schools within the district. This is a value domain.

– `Pchildren`: the percentage of the population in the district that consists of school-children., between 4 and 18 years old. This is also a value domain.

The full table is listed below:

Table 5.1: Information per district of the city of Cochabamba

District	Housecon	Schools	Pchildren
nr 1	Moderate	8	31
nr 2	Good	7	22
nr 3	Good	2	24
nr 4	Good	10	18
nr 5	Good	10	15
nr 6	Moderate	4	29
nr 7	Moderate	5	34
nr 8	Moderate	5	33
nr 9	Moderate	8	32
nr 10	Bad	5	42
nr 11	Moderate	5	35
nr 12	Bad	1	40
nr 13	Bad	7	35

☞

- Double-click the New Table item in the Operation-list. The Create Table dialog box is opened.
- Type `District` in the text box Table Name.
- Type the description: `Information related to the cadastral districts`.
- Select the domain `District` from the list box Domain.
- Click OK.

Now the table window is opened. The table has 13 records with the identification codes of the 13 districts (`nr 1` up to `nr 13`).

The first column will be `Housecon`. This column will have a class domain with three classes (`Bad`, `Moderate`, and `Good`).

☞

- Open the Columns menu and select the Add Column command. The Add Column dialog box is opened.
- Type `Housecon` in the Column Name text box.
- Click the Create Domain button (the small button to the right of the arrow of the Domain list box). The Create Domain dialog box is opened. Type the name `Housecon` in the text box, and make sure that the option Class is selected.
- Type the description: `Housing condition`. Click OK.
- The Domain Editor is opened. Open the Edit menu and select the Add Item command. The Add Domain Item dialog box is opened.
- Enter the first class name: `Bad`. There is also an option to enter a code, which is useful when the class names are large. In this case no code is entered. Click OK.
- Then add the second class by pressing INS. Type for the name of the second class: `Moderate`. Then add the third class: `Good`.
- From the Edit menu, select the option Manual Sorting. Select the word `Moderate` and place it above `Good`.
- Close the Domain Editor.
- Type the description in the Add Column dialog box: `Average housing condition per district`.
- Click OK in the Add Column dialog box.

Column `Housecon` is added to the table `District`.

☞

- Click the first field of column `Housecon`. A drop-down list is opened. If you click the small arrow, the three classes of the domain of this column are shown. Select the class `Moderate`.
- Press the arrow down (↓) key on the keyboard, to go to the next field. You can use the left (←) and right (→) arrows to move through the items in the domain list. Select `Good` from the list for district nr 2.
- Enter the rest of the data for this column `Housecon` according to table 5.1.

Then create the second column for this table: `Schools` (number of schools in each district).

☞

- Double-click the gray column header to the right of the column `Housecon`. This is another way of adding a column.
- The Add Column dialog box is opened. Enter the Column Name: `Schools`. Select the default domain: Value, and enter for the value range: `0` to `50`, and the precision: `1.0`.
- Type the description: `Number of schools per district`. Click OK.
- Enter the values for the number of schools as given in table 5.1.

The last column to add is called: `Pchildren` (the percentage of the population consisting of schoolchildren).

☞

- Add the column: `Pchildren`. For the domain of this column select the system domain Perc, with a value range of `0` to `100`, and a precision of `1.0`.
- Type the description: `Percentage school children per district`.
- Click OK.
- Enter the values as shown in table 5.1.
- Close the table window

Editing a column

You can edit any field in a table by clicking it and typing the new contents. You can use the following keys on the keyboard to edit a table:

– The UP or DOWN ARROW keys, to move one field up or down.

– The TAB key, to move one field to the right.

– The SHIFT + TAB keys, to move one field to the left.

– The ESC key to stop editing.

For editing columns you will work with the table `Citybl`. First you will add one extra column, called `Tourism`, containing the most important touristic sites within the city center of Cochabamba.

☞

- Open the table `Citybl`.
  Add a new column `Tourism`. Create for this column a new domain `Tourism`, in which you add the items shown in table 5.2.
- Go to the records indicated in table 5.2 and type the text shown in the table.
- Close the table window.

Table 5.2: Some touristic sites in Cochabamba

city block	**Tourism**
017	Cathedral
018	Central Park
075	Post office
267	University
553	Stadium
700	Memorial hill
707	Statue

Summary: Creating, importing and editing tables

– Tables can be obtained in two ways: either by creating them in ILWIS and entering the values manually from the keyboard, or by importing existing data in other file formats.

– An attribute table can be linked to other data objects that have either a class or an ID domain.

– It is also possible to create tables with a domain `None`. In that case you cannot link the table to other objects.

– An attribute table may contain several columns. Each of these columns can have a class, ID or value domain (or other special domains, such as color, string, perc, bool etc.).

5.2 Importing digital attribute data

In many GIS projects you don't have to start from scratch. Often a lot of data is readily available, also in digital format. This is especially true for attribute data such as the data we are using in this exercise. These data have to be imported into ILWIS and adapted so that they can be used in combination with maps.

In this small exercise we will import a Dbase file which contains the number of houses for each cityblock in Cochabamba.

☞

- In the main window, open the File menu, and select the Import command.
- The Import dialog box is opened.
- Select for the Import Format: `dBase table .DBF`.
- Select the file `Houses`. The output file name should also be `Houses`.
- Click OK.
- The Importing dBase III/IV table dialog box is opened. In this dialog box you can indicate which of the columns is defining the domain of the table, and the domain type of the columns. As you can see the column `Name` is used to define the domain of the table, and the column `Houses` is a value domain.
- Click OK.

Now the table is being imported into ILWIS. For tables that are imported from other software packages, it is important to select which one of the columns is used to define the domain of the table.

☞

- Open table `Houses`.
- In the table window, open the Edit menu and select Properties.

As you can see in the Properties dialog box, the table has the domain `Houses`, while it should have the domain `Citybl` in order to be able to use it in connection with the table `Citybl`. To solve this you need to create a new column in the table, which has the domain `Citybl`. The contents of this new column can be the same as the domain items used for the table (the items in the left gray column). The new column can be created with a table calculation formula (which will be treated extensively in the next exercise).

☞

- Type the following formula on the command line of the table window:
  `Citybl:=%K`

The %K in the formula is a pre-defined variable, which refers to the domain items in the left gray column of the table.

☞

- Press Enter. The Column properties dialog box is opened.
- Select the Domain: `Citybl`. Click OK.
- Close the table window.

Now the table houses has a (key)column (`Citybl`) through which you can join it with the table `Citybl`. Table joining will be explained in exercise 5.6.

5.3 Calculating with columns

After some introductory exercises dealing with the creation and conversion of a table, you are now starting with the main theme of this chapter: *table calculations*. These calculations will be done with *Table Calculation formulas*.

The formulas can be typed on the command line of a table window. A table calculation formula consists of an output column that will contain the result of the calculation, the definition symbol, or the assignment symbol, and an expression:

```
Outputcolumn=Expression
```

or

```
Outputcolumn:=Expression
```

When the Output column does not exist, a new column is created in the table. When the Output column exists in the table, the content of the column is replaced by the result of the formula. Column names should start with a non-numeric character and there is no limit to its length.

– When the assignment symbol (`:=`) is used, a column is created, that is not depending on other columns (a source data object, see section 2.5). You can edit the data stored in such a column directly, e.g., by double-clicking on the field to be edited.

– When the definition symbol (`=`) is used and the output column does not exist yet, a *dependent column* is created. The data in a dependent column depend on data from other columns via the formula which was used. When data in one of the source columns is changed, the result also changes, when you make the column up to date. Fields of a dependent column cannot be edited directly to protect the result of a calculation.

The expression usually contains operators and/or functions to specify the calculation to be performed. The expression calculates results for all records of the output column. The available TabCalc operators and functions are listed in the On-line Help and in the Reference Guide. You can press F1 at any time in the table window to access detailed information about them.

A formula can be very long. You can move within the formula using the left and right arrow keys, in combination with the CTRL key, or by using the mouse pointer. To retrieve a previous formula, use the arrow up key on the keyboard: this is called the *history*.

! You can use the following short-cut keys:
- Ctrl+V (to paste information from the Clipboard to the Command line),
- Ctrl+C (to copy from the Command line to the Clipboard).

This is extremely useful when you have to edit complicated and long formulas. For an overview of the keyboard shortcuts see the Appendix I of the Reference Guide.

Calculations with value columns

You will now practice with calculations using columns. There are quite a lot of differences in the way you calculate with value columns on the one hand, and class and ID columns on the other. Let's start by using some operators on columns with a value domain. We use the table `Citybl` to calculate the area of each city block in hectares, by dividing the column `Area` with 10000. (1 hectare = 100*100 = 10,000 square meters).

☞

- Open the table `Citybl`.
- In the table window, position the mouse pointer on the command line and type the following formula:
  `Areaha=Area/10000`
- Press Enter.

Now the Column properties dialog box is opened. Since the column `Areaha` did not exist in the table, it now has to be defined. This dialog box provides possibilities to define the new column: domain type, range, precision, position of the column, width of the column, number of decimals and description. The value range defines the range of possible output values in the output column. Calculated values outside this range are assigned the Undefined value (indicated by a question mark: ?). It is thus important to select a range which encompasses all the possible output values.

In the Column properties dialog box, the value range is defined in two text boxes: the minimum value in the first text box, and the maximum value in the other one. The precision text box is used to define the resolution of output values in the output column. A precision of 1 means that output values will be rounded to whole numbers. A precision of 0.1 means that the output values will have 1 decimal. The description text box can be used to enter a text explaining what the column represents. The use of description text boxes is optional, but it is highly recommended, as it helps to remind you what the data means.

☞

- Type some text in the Description text box, e.g.:
  `Area of the city blocks in hectares`.
- The rest of the properties we will leave as they are; accept the default values suggested by ILWIS by clicking OK.

The dialog box is closed, and the program executes the formula. The results are displayed in the column `Areaha`.

Operators for value columns

There are several types of operators and functions that can be used in expressions with value columns.

! You can get an overview of all operators and functions that can be used for value columns in the On-Line Help. Open the Help menu, and select Help on this Window. The Help is opened on the Table window page. Select the hypertext link Command line. Select the hypertext link Table Calculation. Select the hypertext link Operators and functions on value columns. Now you will see an overview of the operators and functions on value columns. Click the hypertext links to get more information and examples of the various operators.

Arithmetic operators

These are the most simple operators, which are used for the multiplication, division, subtraction or addition of columns. Another example of these arithmetic operators will be used to calculate the population density per city block.

☞

- In the table window, position the mouse pointer on the command line and type the following formula:
  `Popdens=Population/Areaha ↵`

  From now on the symbol ↵ will be used to indicate that you have to press Enter.

  The Column properties dialog box is opened.
- Type the following description in the Column properties dialog box: `Population density (persons per hectare)`.
- Click OK.

The table window now also contains the column `Popdens` with the population density for each city block.

Relational operators

Relational operators (=, <, <=, >, >=, <>) test whether one expression is equal, smaller, smaller or equal, larger, larger or equal, or different than another expression. Let us find out, for example, which city blocks have a population density of more than 200 persons per hectare.

☞

- Type the following formula on the command line:
  `Highdens=Popdens>200 ↵`
- The Column properties dialog box is opened.
- Type the following description: `Population density higher than 200 persons per hectare`.

Note that the suggested domain in the Column Properties dialog box is Bool. A Bool domain (abbreviation of *Boolean*) has only three possible values: True, False and undefined.

☞

- Click OK.

Logical operators

Logical operators such as AND, OR, XOR and NOT, compare two expressions and check if both are true (AND), at least one is true (OR), or only one is true (XOR). The NOT operation checks if an expression is true or false. If the expression is true, the NOT operation will result in false and vice versa. As an example, the AND operator is used to find the city blocks that have a population density between 200 and 300 persons per hectare.

☞

- Type the following formula on the command line:
  `Highdens1=(Popdens>200)and(Popdens<300) ↵`
  The Column properties dialog box is opened.
- Type the following description: `Population density between 200 and 300 persons per hectare`.
- Click OK.

The same result could be obtained with the INRANGE function.

☞

- Type the following formula on the command line:
  `Highdens2=INRANGE(Popdens,200,300)↵`
- Click OK in the Column properties dialog box.

Conditional function

The examples that we have used for the relational and logical operators all give output values which are either true or false or undefined. In practice we use these operators mostly in combination with the *conditional IFF function*, which has the form of an IF...THEN..ELSE statement. The format for a conditional statement in ILWIS is: IFF(*a*,*b*,*c*). If condition *a* is true, then return expression *b*, otherwise return expression *c*.

! In the previous DOS versions of ILWIS this function was written as IF(*a*,*b*,*c*), with only one F. Now it is defined as a conditional IF Function (IFF).

☞

- Type the following formula on the command line:
  ```
  Highdens3=IFF((Popdens>200)and(Popdens<300),
  Popdens,?)↵
  ```
- Click OK in the Column properties dialog box.

The resulting column (`Highdens3`) contains values for those records where the condition `(Popdens>200)and(Popdens<300)` was true. The other fields contain a question mark (?). These are so- called *undefined values*.

Conditional functions are used very extensively in ILWIS. You will find that 3 out of 4 formulas you create will contain an IFF statement.

Conditional functions can also be part of another conditional function. In that case we call them nested IFF functions. The calculation performed above can also be written as such a nested IFF function.

☞

- Type the following formula on the command line:
  ```
  Highdens4=IFF(Popdens>200,IFF(Popdens<300,
  Popdens,?),?)↵
  ```
- Click OK in the Column properties dialog box.

When you use nested IFF functions, you should be careful in determining how many closing brackets you must use (always as many closing brackets as there are IFF's in the formula), and where you put them.

Before continuing it is better to delete the columns that were made in this exercise, except for the column `Popdens`.

☞

- Select column Highdens by clicking the name of the column.
- From the Edit menu choose Delete. Confirm the deletion.
- Select column Highdens1, and press the Delete key on the keyboard. Confirm the deletion.
- Delete the columns Highdens2, Highdens3, and Highdens4.
- Close table Citybl.

Operators used for class or ID columns

Columns that have a class or ID domain can also be used in formulas. When using class names or IDs within an expression, these class names and IDs should be put between double quotes, e.g. Residential.

The number of operators and functions that you can use for class or ID columns is quite limited. Of course, arithmetic operators make no sense (you can hardly divide one word by another). Relational operators are only limited to two: = (check whether two names are equal) and <> (check if they are unequal). Logical operators and conditional functions, however, are used extensively on class or ID columns.

Some examples are shown below. The first example is finding out which city blocks have a land use Residential and occur within the district nr 7.

☞

- Open table Citybl.
- Type the following formula on the command line:
  ```
  Resnr7=(Landuse=“Residential”)and(District=
  “nr 7”)↵
  ```
- Click OK in the Column properties dialog box.

The result of this formula is a column, using domain Bool.

Now let us combine several operators and an IFF function. We will find out the city blocks with either a residential or commercial land use in district nr 7.

☞

- Type the following formula on the command line:
  ```
  Rescomnr7=IFF(((Landuse=“Residential”)or(Landuse=
  “Commercial”))and(District=“nr 7”),Landuse,”?”)↵
  ```

Note that in the Column properties dialog box the suggested domain of the output column is Cityluse. This is because the possible results of the formula are either

Residential or Commercial, which are both items of the domain Cityluse, or ? (undefined).

☞

- Click OK in the Column Properties dialog box.

How to define domains for IFF functions with class or ID

When we work with IFF functions that give class or ID results, we may have four different situations:

– 1. *IFF(expression, domain1,"?")*. The result of the expression fits in an existing class or ID domain (mostly one of the input columns), or is the undefined value. This is illustrated by the example shown above;

– 2. *IFF(expression, domain1,"name")*. The result of the expression fits in a existing domain or is a name which is not yet in the domain. In this case you must add one item Name to the existing domain. A similar situation is *IFF(expression,"name",domain1)*;

– 3. *IFF(expression, domain1, domain2)*. The result of the expression fits either in domain1 or in domain2. In this case you must merge the two domains;

– 4. *IFF(expression, "name","name")*. The result of the expression is two names which fit, or may not fit, in an existing domain.

In situations 2, 3, and 4 you cannot simply use an existing domain for the output column. You should either add items to the existing domain, or you should create a new domain, depending on the THEN and ELSE parts of the IFF function. In case 2 you can add an extra name to the existing domain. In case 3 you can merge the two domains, either by adding the items of one domain to the other or by making a new domain which contains the items of both. In case 4 you can make a new domain which contains both names.

When ILWIS encounters in an IFF function one of the first three possibilities, the program will suggest that you use an existing domain. When you decide to use that existing domain and press OK in the Column properties dialog box, you will get a warning, and you are asked whether you want to add the missing items to that domain. However, it is not advisable to generate large domains with a mixture of information. In many cases it is better to generate a new domain. To do so, you will have to press the Create domain button in the Column properties form.

Note that it is always possible to use the output domain String for all four situations given above. However, it is not advised as a string column can not be used in combination with a map.

Let us now look at the situations 2, 3, and 4 with some examples.

☞

- Type the following formula on the command line:
  `Newlanduse=IFF(Landuse<>“Residential”,Landuse,”Houses”)↵`

This statement attempts to replace the class `Residential` with `Houses`. This is an example of situation 2, *IFF(expression, domain1,”name”)*. If the land use in the city block is not `Residential` then take the names from the column `Landuse`, otherwise use the word `Houses`. So in fact we replace the word `Residential` with the word `Houses`. This is a situation in which you could still use the existing domain `Cityluse` (belonging to the column `Landuse`) and add the item `Houses` to it.

☞

- Click OK in the Column properties dialog box.
- The Merge domain dialog box appears, asking you to Add string `Houses` to domain `Cityluse`. Answer with Yes.

The new column `Newlanduse` shows the names of the non-residential city blocks, as in the column `Landuse`, and the new name `Houses` for the residential city blocks.

Let us now look at situation 3: *IFF(expression, domain1, domain2)*. If there is information in column `Tourism` (which you generated in exercise 5.1), then we will use that information, otherwise we will take the names of the land use classes.

☞

- Type the following formula on the command line:
  `Newlanduse2=IFF(ISUNDEF(Tourism),Landuse,Tourism)↵`

What we have done here, is to look at whether fields in the column `Tourism` have undefined values, using a special function ISUNDEF. If that is the case, the names from the `Landuse` column are used, otherwise the names from the column `Tourism`. Since the items in the column `Tourism` also refer to land use, we can add the items of the existing domain `Cityluse` to it, so there is no need to create a new domain.

☞

- Press OK in the Column properties dialog box. The Merge domain dialog box appears, asking you to `Merge strings of domain ‘Cityluse’ into domain ‘Tourism’`. Answer with Yes.

Finally, let us give an example of situation 4, *IFF(Expression,"name","name")*. We will evaluate which city blocks are located in the city center. The city center consists of the district numbers 4, 5, 6, 7, 8, and 9. If we would create a formula for that it would be quite long:

```
Center=IFF((District="nr 4")or(District="nr 5")or(District="nr 6") or
(District="nr 7")or(District="nr 8")or(District="nr 9"),"Center","Not
in center")
```

Since this is quite a long formula we can do some tricks to shorten the equation. We need to evaluate whether the district is larger then 3 and smaller than 10. Unfortunately you cannot perform such relational operators on ID or class columns (the column district is an ID column). So what we have to do is to change the names of the column `District`, which appear to be numbers but are in fact IDs, to true numbers. There are special functions for this.

☞

- Type the following formula on the command line:
  `Distrnr=VALUE(RIGHT(District,2))↵`

In this formula two functions are combined. The formula first selects the last two characters of the strings in column `District` with the function `RIGHT(District,2)`, and then these characters are converted to values with the function `Value()`.

☞

- Select the default domain Value in the Column properties dialog box, with a minimum 1 and maximum 13, and a precision of 1. Click OK. Now you have values and can perform the actual calculation.
- Type the following formula on the command line:
  ```
  Center=IFF(INRANGE(Distrnr,4,9),"Center","Not
     in center") ↵
  ```

Now the calculation formula is much shorter, and the INRANGE function, which works on value columns, can be used to see if the district number of the district is between 4 and 9.

In this case it would make no sense to add the new names `Center` and `Not in center` to an existing domain. They contain a different type of information than that of the domains `Cityluse` or `District`.

Since the ELSE and THEN parts of the IFF function contain names that ILWIS cannot identify as being part of an existing domain, the program will display the

String domain as the default domain for the output column. In this case, however, we would like to make a new domain: `Center`.

☞

- Click the Create domain button (located next to the arrow of the domain list box). The Create Domain dialog box is opened.
- Enter the Domain Name: `Center`. Click OK.
- The Domain Editor is opened. Add the two items: `Center` and `Not in center`. Close the Domain Editor. Now you are back in the Column properties dialog box. The domain `Center` is now selected. Click OK.

! Note that you could have also calculated the entire formula directly, by combining a series of functions in one expression, without the need to generate a value column first:

```
Center=IFF(INRANGE(VALUE(RIGHT(District,2)),4,9),”Center”,
”Not in center”)
```

There are many more operators and functions that can be used for class or ID domains than the ones we have treated in this exercise.

! You can get an overview of all operators and functions that can be used for class or ID columns in the On-Line Help and the Reference Guide.
There is a series of functions that allows you to calculate with coordinates of maps, and a group of functions to calculate with colors.

Before you continue with the next exercise, it is better to delete most of the columns created in this exercise.

☞

- Select column `Resnr7` by clicking the name of the column, and press the Delete key on the keyboard. Confirm the deletion.
- Delete also the columns `Rescomnr7`, `Newlanduse`, and `Newlanduse2`.
- When you try to delete the column `Distrnr`, you will get a message that the column is still in use. It is used to calculate the dependent column `Center`. So before we can delete it, the dependency link should be broken.

- Double-click the column header `Center`. The Column properties dialog box is opened. Press the Break Dependency Link button. Confirm this by clicking Yes in the next window.
- Delete the columns `Center` and `Distrnr`.
- Close the table `Citybl`.

Summary: Table Calculation

– Calculations are typed on the command line of the table window with formulas. Each formula contains the name of an output column, a definition or assignment symbol (which determines whether the output column will be dependent or not) and an expression.

– An expression contains operators, functions and operands (names of columns, strings and values).

– There is a different set of operators and functions on value columns and on class, ID, and string columns. You cannot mix values and strings in an output column. The result column should be either value or string.

– On value columns we can use arithmetic, relational, and conditional operators, and a whole set of functions, of which the IFF function is the most important.

– *IFF(expression, then, else)* statements are very common in ILWIS. IFF functions can be combined into nested IFF statements.

– On class, ID or string columns we can use a smaller set of operators and functions (some relational and conditional operators).

– When the result of an IFF statement is a string, you can have 4 possibilities:

IFF(expression, domain1,”?”), IFF(expression, domain1,”name”),

IFF(expression, domain1, domain2), and *IFF(expression, “name”,”name”).*

5.4 Classifying data in a column

In the previous exercise you have seen a number of different operators and functions that can be used in calculations with columns. In this exercise you will learn another useful function: *classification of values in a column*.

The table `Citybl`, containing information on the city blocks of Cochabamba, will be used again as an example.

☞

- Open table `Citybl`.

One of the columns in this table that was created in the previous exercise is called `Popdens` (population density of each city block, expressed as number of persons per hectare).

! If you don't have this column you can generate it by typing the following command on the command line of the table window:

```
Popdens:=10000*Population/Area
```

Domain Group

The column `Popdens` shows different values for each city block. To simplify this information it would be handy if we could classify it into a number of classes. This is done with the *CLFY function*. This function classifies values into a number of classes. The classes should be defined beforehand, and are stored in a so-called *domain Group*. A Group domain is a special type of class domain, in which for each class a boundary value is given.

Table 5.3: Boundary values for classifying population density

Upper Bound	**Name**
0	0 persons/ha
50	1 - 50 persons/ha
100	50 - 100 persons/ha
200	100 - 200 persons/ha
10000	> 200 persons/ha

You will first create a group domain, called `Popdensc`.

☞

- Activate the main window of ILWIS, open the File menu and select the Create, and the Create Domain commands. The Create Domain dialog is opened.
- Type for the Domain Name: `Popdensc`. Click the option: Class and the check box Group. Click OK. The Domain Editor is opened.
- From the Edit menu, select the Add Item command. The Add Domain Item dialog box is opened.
- Type for the Upper Bound: `0`. Name: `0 persons/ha`. Click OK.
- Press the INS key. The Add Domain Item dialog box is opened.
- Type for the Upper Bound: `50`. Name: `0 - 50 persons/ha`. Click OK.
- Repeat the procedure for the other classes listed in the table.
- Close the Domain Editor.

Now that the group domain is created we can classify the population density values in the table.

☞

- Activate the Table window of ILWIS with the table `Citybl`, and type the following formula on the command line:
  `Popdensc=CLFY(Popdens,Popdensc)↵`
- Click OK in the Column properties dialog box. Now you will see a new column with the description for each city block.
- Close the table `Citybl`.

Group domains can also be used on value maps, while using the SLICING operation (see chapter 7).

Summary: Classifying data in a column

This exercise dealt with the classification of values in a column.

– The classification is done with the CLFY function.

– A classification uses a so-called *Group domain*, which is a special type of class domain. For each class name, there is a boundary value to be used in the classification.

5.5 Aggregate functions

In the previous exercises, the use of several operators and functions that can be used in the calculation of columns were presented. All of these operators and functions work on each field individually, without looking at other fields in the same column.

In several cases, you may want to take into account the value for the same unit in the whole column or in sub-sets of records in a column. For example, you may want to calculate the population density, not for each city block individually, but for an entire district of the city. The functions that allow you to do such kind of operations are called *aggregate functions* (see figure 5.2). Aggregations are performed on:

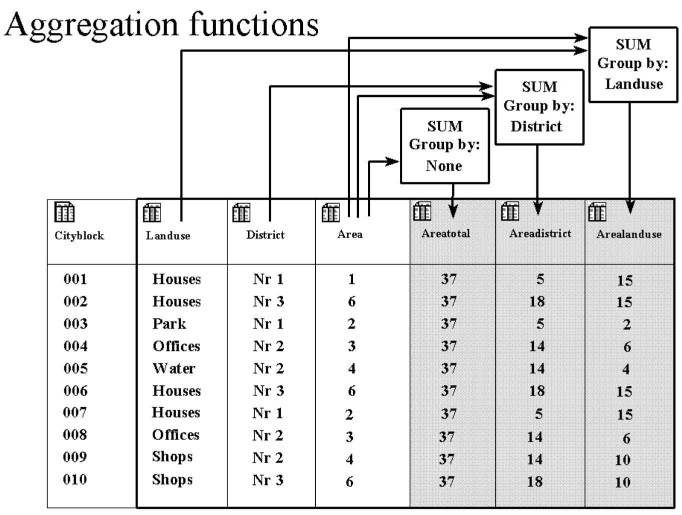
– all records of a column, or

– all records that belong to the same group of records as determined by a key or grouping column,

Aggregate functions can be performed on value columns as well as on class and ID columns.

– In case of value columns, all available statistical functions might be meaningful. In short, the following aggregations can be performed: *average, weighted average, count, minimum, median, weighted median, maximum, predominant, weighted predominant, standard deviation, weighted standard deviation,* and *sum*.

– In case of class or ID columns, only the *median*, the *count* and the *predominant* aggregate functions are meaningful.



Cityblock	Landuse	District	Area	Areatotal	Areadistrict	Arealanduse
001	Houses	Nr 1	1	37	5	15
002	Houses	Nr 3	6	37	18	15
003	Park	Nr 1	2	37	5	2
004	Offices	Nr 2	3	37	14	6
005	Water	Nr 2	4	37	14	4
006	Houses	Nr 3	6	37	18	15
007	Houses	Nr 1	2	37	5	15
008	Offices	Nr 2	3	37	14	6
009	Shops	Nr 2	4	37	14	10
010	Shops	Nr 3	6	37	18	10

Figure 5.2: An example of the use of the aggregation function SUM for three situations. Group by: none: all values are summed up. Group by the column `District`: all values are summed up for each `District`. Group by: column `Landuse`

Aggregation commands can be selected from the Columns menu in a table window, or by typing a statement on the command line of a table window. The first method is shown here. The use of the command line for operations will be explained in chapter 12. The result of an aggregation can be written into a new column of the same table, in a new column of a new table, or in a new column of another existing table.

Some examples will be shown, again with the table `Citybl`. The first example is the calculation of the total area of all city blocks.

☞

- Open table `Citybl`.
- Open the Columns menu, and select the Aggregation command. The Aggregate Column dialog box is opened.
- Select for the Column: `Area`, the function: Sum, and type as Output Column: `Totalarea`. Click OK.
- Type the description: `Total area of all city blocks` in the Column properties dialog box. Click OK.

You will see that the column `Totalarea` contains the value 10678499.792 for all records. That is because you have calculated the total area summing up all records. It is evident that the column `Totalarea` is not useful in combination with a map. Since the column only contains 1 value, it says nothing about the different units in the map.

Instead, you can also calculate the total area per district. Then areas are summed up only for those records that have the same name in the column `District`. The column `District` is used to group the records. This way we can calculate the total area of each district, or of each land use type, or any other group column.

☞

- Open the Columns menu and select the Aggregation command. The Aggregate Column dialog box is opened.
- Select for the Column: `Area`, the function: Sum.
- Select the check box Group by, and select the column `District`.
- Type as Output Column: `Areadistrict`. Click OK.
- Type the description: `Total area of each district` in the Column properties dialog box. Click OK.

The values in the column `Areadistrict` now show different areas for different districts.

☞

- Check this by opening the Columns menu, and choosing the Sort command. The Sort dialog box is opened. Click Column, and select the column `District`. Click OK. The table is sorted according to the column `District`.
- In the table window, use the scroll button on the right-hand side and check whether the values in column `Areadistrict` change for each district.
- Sort the table again on the domain.

It is rather uneconomical to store the data from the column `Areadistrict` within the table `Citybl`. Out of 717 different records, there are only 13 different values in this column. These values do not relate to the city blocks, but to the districts. To reduce the *data redundancy* it would be better to store the results in the `District` table (the one that you created in section 5.1).

☞

- Perform the same aggregation; use the same name for the output column, but choose to write the output in the table: `District`.
- Open the table `District` and look at the result.

Now the data is stored in a more useful way. The table has 13 records (13 districts), and each district has a value for `Areadistrict`.

You will now use another aggregation function: Average.

☞

- Close the table `District`.
- In the table `Citybl`, calculate the average number of persons per building block for each district. Store the result in the column: `Avgpopbl` in the table `District`.
- Open the table `District` and examine the result.
- Close all table windows.

A more complex problem

Now you will learn how to use aggregate functions for a more complicated problem: what is the percentage of the land use types residential, commercial and institutional per district. You cannot obtain this information directly by an aggregate function, since the table has 717 records, one for each city block. For each city block it is indicated what is the district and what is the land use. The area of the districts is calculated already in the column `Areadistrict`. Now you need to know the area of each combination of `District` and `Landuse`. In order to do that you will have to combine the two columns `District` and `Landuse`. This can be done with the so-called *concatenation operation*. With a concatenation operation you can glue two strings together.

☞

- Open the table `Citybl`.
- Type the following formula on the command line:
  `Distrlanduse=District+Landuse ↵`
- The Column properties dialog box is now displayed. The output column will have a String domain. Click OK.

The new column `Distrlanduse` now contains the unique combinations of the district numbers and the land use types. So now we can calculate with an aggregate function the area occupied by each type of land use in each district. However, you should first change the domain for the column `Distrlanduse` from string to a class domain.

☞

- Double-click the column header `Distrlanduse`. The Column Properties dialog box is opened. Press the button Create Domain. The Create Domain dialog box is opened.
- Change the name of the domain to `Dlanduse`. Not that the check box Apply domain to column is selected. Click OK. The domain of the column `Distrlanduse` is now a class domain, called `Dlanduse`. Click OK in the Column properties dialog box.
- Open the Columns menu and select the Aggregation command. The Aggregate Column dialog box is opened.
- Select for the Column: `Area`, the function: Sum.
- Select the check box Group by, and select the column `Distrlanduse`.
- Type for the Output Column: `Areadistrlu`. Click OK.
- Type for the description: `Area occupied by each land use type in each district`. Click OK.

Now you know the area of each district, and the area of each land use type in each district. The combination of the two will give us the percentage of each district occupied by several land use types.

☞

- Type the following formula on the command line:
  `Residential=IFF(Landuse="Residential",100*`
  `Areadistrlu/Areadistrict,0)↵`
- Select for the domain Perc, and click OK.

The column `Residential` shows the percentage of each district occupied by houses.

Now do the same thing for the land use types `Commercial` and `Institutional`.

☞

- Locate the mouse pointer on the command line, and press the up arrow. The previous formula is displayed again.
- Edit this formula so that it looks like:
  `Commercial=IFF(Landuse="Commercial",100*`
  `Areadistrlu/Areadistrict,0)`
- Press Enter. Select the domain Perc, and click OK.
- Then, edit the previous formula so that it looks like:
  `Institutional=IFF(Landuse="Institutional",100*a`
  `Readistrlu/Areadistrict,0)`
- Press Enter. Select the domain Perc, and click OK.
- Open the Columns menu and select Sort. Sort the table on the column `Distrlanduse` and compare the names in that column with the values in the columns `Residential`, `Commercial` and `Institutional`.

Whenever there is a combination of a district with the land use type `Residential`, the column `Residential` displays the value of the area within the district occupied by residential buildings.

It is quite obvious that the way the columns `Residential`, `Commercial` and `Institutional` are ordered, is not efficient. The data in these columns all relate to the district, and should be written only in the 13 records of the table `District`, in stead of the 717 building blocks. We will use these columns in a later exercise to join them in the table `District`. Most of the other columns created in this exercise can be deleted.

☞

- Select the column `Totalarea` by clicking the name of the column, and press the Delete key on the keyboard. Confirm the deletion.
- Sort again by Domain.
- Break the Dependency Link of the columns `Institutional`, `Commercial`, `Residential`, `Areadistrlu`, `Distrlanduse`, `Areadistrict` and `Popdensc`, by pressing the Break Dependency Link button in their Column properties dialog box.
- Delete the columns `Popdensc`, `Areadistrict`, `Distrlanduse` and `Areadistrlu`.
- Close the table `Citybl`.

Summary: Aggregate functions

Aggregations are either performed on:

– all records of a column,

– all records that belong to the same class as determined by a grouping column.

Aggregate functions can be performed on value columns as well as on class and ID columns.

– In the case of value columns, the following aggregations can be performed: *average, weighted average, count, minimum, median, weighted median, maximum, predominant, weighted predominant, standard deviation, weighted standard deviation* and *sum.*

– In case of class or ID columns, only the *median*, the *count* and the *predominant* aggregate functions are meaningful.

The result of an aggregation can be written into a new column of the same table, in a new column of a new table, or in a new column of another existing table.

5.6 Table joining

In the previous exercise you saw that you can store results of an aggregation in another table, which has the same domain as the group by column used in the aggregation, in order to reduce the data redundancy. In this exercise you will look more in detail how you can use columns from one table in another table. The process of linking tables in called *joining*.

To join tables you need two tables, one that receives data and one that provides data. Futhermore, you need a *common domain* in the two tables.

When you want to combine information from two tables you may have one of the following four situations:

– **1. The domain of the current table is the same as the domain of the other table from which you want to join a column** (see figure 5.3-A). You can directly obtain data from the second table: the link between the two tables is through the common domain of both tables.

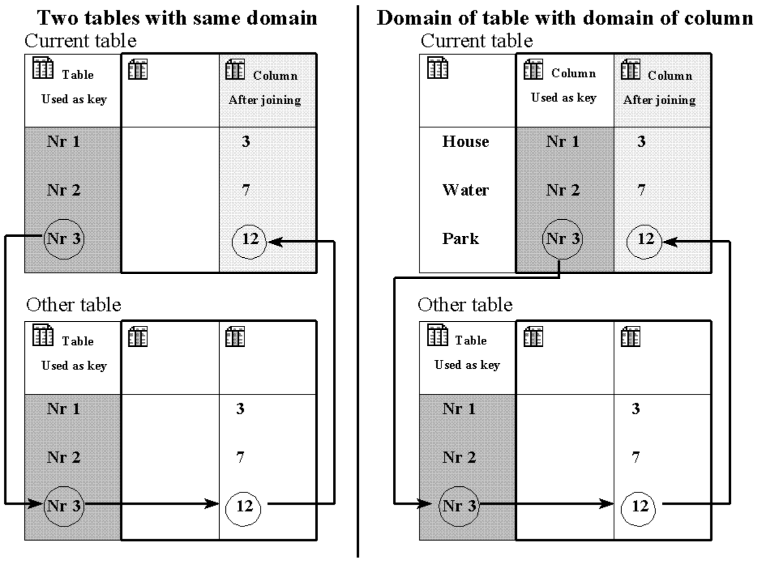


Figure 5.3: Two different ways of table joining. Dark shaded columns are used as key. The arrows indicate how the link is made. Fig 5.3-a (left): The domain of the current table is the same as the domain of the other table from which you want to join a column. Fig 5.3-b (right): The domain of a column in the current table is the same as the domain of the other table from which you want to join a column

– **2. The domain of a column in the current table is the same as the domain of the other table from which you want to join a column** (see figure 5.3-B). Specify a *key column*, in the current table; you can then directly obtain data from the second table. The link between the two tables is through the selected key column in the current table and the domain of the second table.

– **3. The domain of the current table is the same as the domain of a key-column in the other table from which you want to join a column**. In this case you have two possibilities:

The key column in the other table contains unique class names, ID's or values. In that case you only have to specify the *key column* in that table (figure 5.4-A)

The key column in the other table does not contain unique class names, ID's or values. In that case direct joining is not possible, since there may be more than one possibility to join. To solve that, you need to *aggregate* the values via the *key column* (figure 5.4-B)

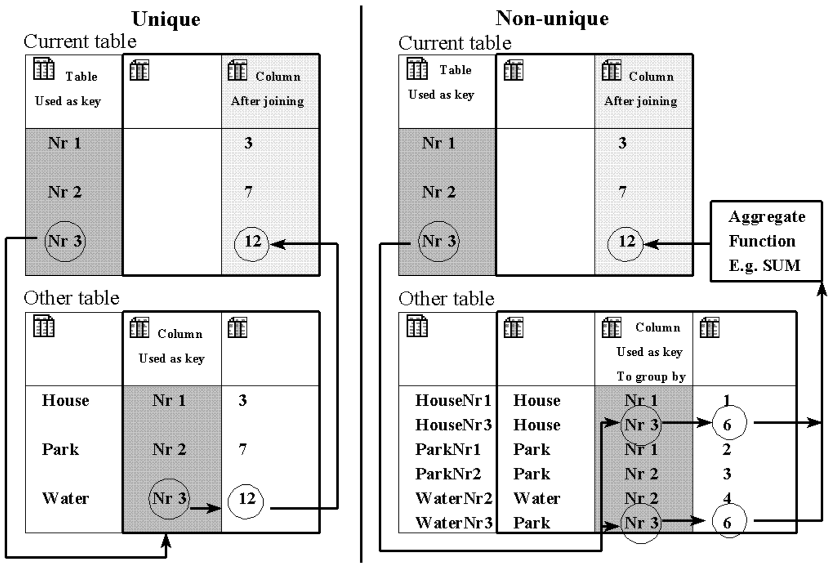


Figure 5.4: The domain of the current table is the same as the domain of a column in the other table from which you want to join a column. Dark shaded columns are used as key. The arrows indicate how the link is made. Fig 5.4-a (left): The column in the other table contains unique class names, ID's or values. Fig 5.4-b (right): The column in the other table does not contain unique class names, ID's or values

– **4. The domain of a column in the current table is the same as a domain of a key column in the other table from which you want to obtain data**. In this case we also have the same two possibilities as in the third situation shown above (see figure 5.5).

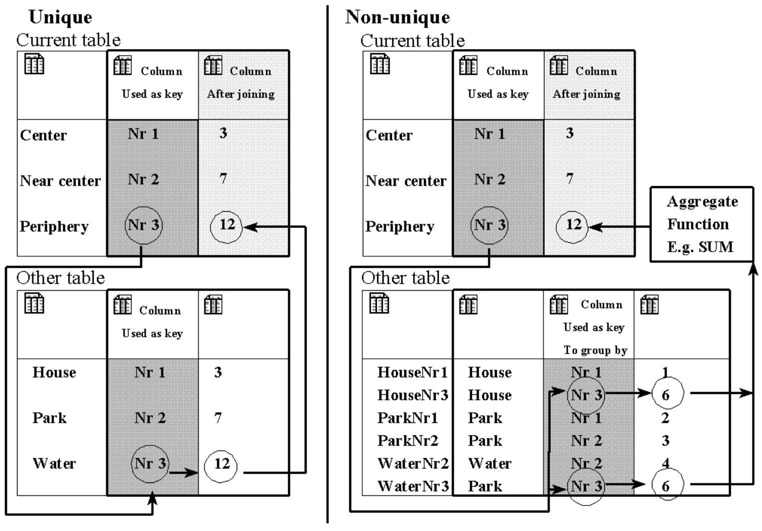


Figure 5.5: The domain of a column in the current table is the same as a domain of a key column in the other table from which you want to obtain data. Dark shaded columns are used as key. The arrows indicate how the link is made. Fig 5.3-a (left): The key column in the other table contains unique class names, ID's or values. Fig 5.2-b (right): The key column in the other table does not contain unique class names, ID's or values

Joining via table domains

Polygon histogram

First you will join two tables which both have the same domain (figure 5.3-a). The first table is the attribute table `Citybl`, with which you have been working in the previous exercises. To this current table, a column from *Polygon Histogram* `Citybl` will be joined. This polygon histogram contains statistical information on each city block: the number of polygons, the perimeter and the area.

! You have not yet worked with histograms in the previous exercises. Histograms will be treated extensively in chapters 6 (histograms for satellite images) and in chapter 7 (histograms of maps).

☞

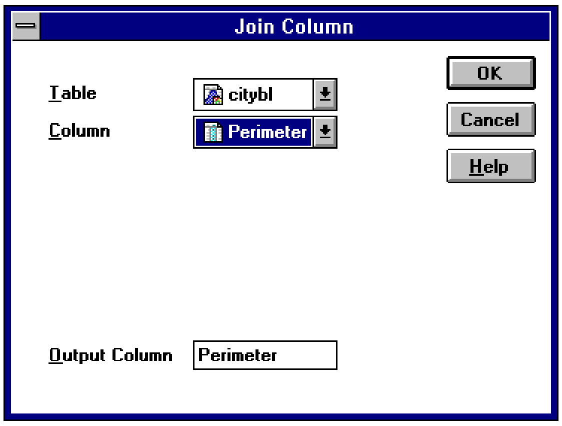
- Open the polygon histogram `Citybl` and evaluate its contents.
- Open the Edit menu and select the Properties command to find out the domain of the table. Click Cancel.

The polygon histogram `Citybl` has the same domain as the attribute table `Citybl`. Note that the column `NrPol` (number of polygons) only contains the value 1. This is logical, since each city block has a unique identifier, and therefore occurs only once. The column `Perimeter` contains the length of the border of each city block (in meters). We want to read this column into the attribute table `Citybl`.

☞

- Close the polygon histogram `Citybl` and open the attribute table `Citybl`.
- From the Columns menu, select the Join command. The Join Column dialog box is opened.
- Select for the Table: the polygon histogram `Citybl` and for the Column: `Perimeter`.

Note that the output column name also changes to Perimeter. The Join Column dialog box now looks like figure 5.6.



Join Column

Table: citybl

Column: Perimeter

Output Column: Perimeter

Figure 5.6: The Join Column dialog box used for joining a column from two tables with the same domain

☞

- Click OK in the Join Column dialog box.
- Click OK in the Column properties dialog box.
- Check the result and close the table `Citybl`.

Column `Perimeter` is now added to the attribute table `Citybl`. We could use this column, together with the column `Area` (that was in fact also obtained from the polygon histogram through the same procedure) to evaluate the shape of the city blocks. A shape factor can be obtained by the ratio of the perimeter and the area.

! If you want to use a column stored in a table, other than the one you are working with, you can explicitly refer to it without importing it into your active table. You can use the following general syntax:
`Table.Column`
where `Table` is the name of the external table and `Column` is the name of the column in this table that you wish to use.
In this case you might thus have obtained the same result if you type on the command line:
`Perimeter=Citybl.hsa.Perimeter`
`Citybl.hsa` is the polygon histogram.

Joining a key column with a table

In this exercise, you will perform a join as shown in figure 5.3-B: the current table has a key column with a domain that is the same as the domain of another table from which you wish to join a column. We would like to know for each city block which percentage of the population consists of schoolchildren. This information is available in the column `Pchildren` in the `District` table, which you have created in exercise 5.1. The information of column `Pchildren` in the `District` table can be joined to the table `Citybl`, because table `Citybl` has a column (`District`) which has the same domain as the `District` table, in which the column of interest `Pchildren` is located.

☞

- Open table `District` and open the table `Citybl`. Check if the domain of the column `District` in the table `Citybl` is the same as the domain of the table `District`.
- Make the table window that displays table `Citybl` the active window.
- From the Columns menu, select the Join command.
- Select for the Table: `District`, for the Column: `Pchildren`. Note that check box Key column is selected. Select for the Key Column: `District`. The Output column: `Pchildren` is already given.

The Join Column dialog box now looks like figure 5.7.

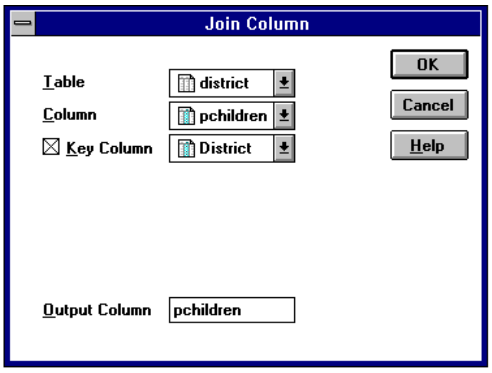


Table	district
Column	pchildren
☒ Key Column	District
Output Column	pchildren

Figure 5.7: The Join Column dialog box, using a key column in the current table to join a column from another table. The other table has the same domain as the key column in the current table

☞

- Click OK in the Join Column dialog box.
- Change the value range in the Column properties dialog box so that minimum = 0, maximum = 100 and precision = 1.0. Click OK.

Column `Pchildren` is now added to attribute table `Citybl`. We could use this column, together with the column `Population` to calculate the number of schoolchildren in each city block.

! Design the formula yourself for calculating the number of schoolchildren per city block.
Also join the table `Citybl` with the table `Houses` (that was imported from dBase in exercise 5.2). Use the column `Houses` and the column `Population` to calculate the average number of persons per house for each cityblock, and for each district.

Table joining combined with aggregation

In this exercise you will deal with a join as shown in figure 5.4-B: the domain of the current table is the same as the domain of a column in another table from which you wish to join. We would like to know the total area and the total population for each district. Besides that, we would like to know the percentage cover of residential, commercial and institutional buildings in each district.

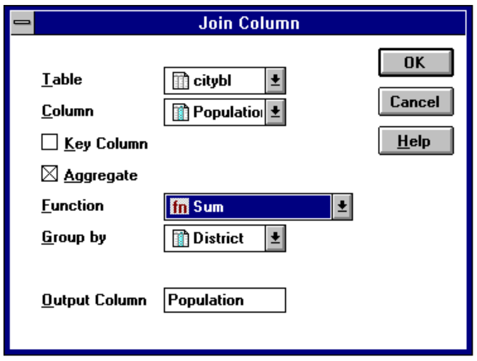
The information on areas and population is available in table `Citybl` for each city block. We also know for each city block in which district it is located. Thus we can use the information from the table `Citybl` and bring it into the table `District`. But as table `Citybl` contains 717 records, and table `District` only 13, you will have to do an aggregation, while doing the join operation.

To join information from table `Citybl` into table `District`, you have to link the table `District` through a *group by column* `District` from the table `Landuse`, which contains the columns of interest.

☞

- Make the table window that displays the `District` table the active window.
- From the Columns menu, select the Join command.
- Select for the Table: `Citybl`, for the Column: `Population`. Clear the check-box Key Column. Select the check-box: Aggregate, and select the function: Sum. Select Group by: `District`. The name of the Output column: `Population` is already given.

The Join Column dialog box now looks like figure 5.8.



Join Column

Table	citybl
Column	Populatio
☐ Key Column	
☒ Aggregate	
Function	Sum
Group by	District
Output Column	Population

Figure 5.8: The Join Column dialog box using an aggregation function (SUM) to join a column (`Population`) from another table (`Citybl`) via a group by column (`District`) into the current table (`District`) which also has the domain (`District`). The aggregate function is needed since the table `Citybl` contains many records for the same item of the domain `District`, while table `District` only contains one record for each domain item

☞

- Click OK in the Join Column dialog box.
- Change the value range in the Column properties dialog box so that minimum = 0, maximum = 100000 and precision = 1.0. Click OK.

When you define the value range you can also use scientific notation: e.g. `1e+6` is the same as `1000000`.

Column `Population` is now added to the attribute table `District`. We could use this column, together with the column `Pchildren` to calculate the number of schoolchildren in each district.

! Design the formula yourself for calculating the number of schoolchildren per district.

Finally, you can also join the columns `Residential`, `Commercial` and `Institutional` to the table `District`. In table `Citybl`, information on the areas of residential, commercial and industrial land use types is available per city block. These columns were created in exercise 5.5.

☞

- Repeat the same procedure as described above for these three columns. Select the aggregation function AVG (average). For the output columns, select domain Perc (percentage) with a Precision of 0.01.
- Close the table `District` and the table `Citybl`.

Summary: Table joining

In order to be able to join tables, the domain of the table or one of its columns should match the domain of the other table or one of its columns. There are four methods in which tables can be joined:

– The domain of the two tables is the same.

– The domain of a column in the current table is the same as the domain of the table from which you want to obtain data.

– The domain of the current table is the same as the domain of a column in the table from which you want to join a column.

– The domain of a column in the current table is the same as a domain of a column in the table from which you want to obtain data.

5.7 Displaying results as graphs

Data in a table can be presented as graphs in a *graph window*. In this exercise you will have a look at the graph window and at the way in which you can display the results of the previous exercise. In the last part of exercise 5.6, you have joined the columns of table `Citybl` into the `District` table, and obtained the columns `Residential`, `Commercial` and `Institutional` in the last table. These columns contain the percentage cover of each of these land use types within each district of the city of Cochabamba.

! If you did not succeed in the last exercise, or if you start with this exercise without having done exercise 5.6, you can use table `Distric1`, which already contains the joined columns, and use table `Distric1` whenever we speak of the table `District`.

☞

- Open table `District`.
- In the table window, open the Options menu, and select the Show Graph command.

The dialog box Graph is opened. In this dialog box, the columns of the table are presented in the X-Axis and Y-Axis list boxes which can be used to select the X and Y axes of the graph.

☞

- Select column `District value` from the X-Axis list box.
- Select column `Residential` from the Y-Axis list box.
- Click OK.

The Edit Graph dialog box is opened and has options to select a graph type and a display color.

☞

- Select Bar for graph type. Type for the Y-Axis name: `Percentage`.
- Type for the Y-Axis Bounds: `0` and `100`.
- Click OK.

The Graph window is opened. The graph shows the percentage cover of residential buildings in each district.

You can add the other two graphs showing the percentage of commercial and institutional buildings.

☞

- From the Graph menu choose Graph Management. The Graph Management dialog box is opened.
- Click the button Add Graph. The Add Graph dialog box is opened.
- Select column `District value` from the X-Axis list box.
- Select column `Commercial` from the Y-Axis list box.
- Click OK. The Edit Graph dialog box is opened.
- Select Bar for the graph type.
- Type for the Y-Axis name: `Percentage`.
- Type for the Y-Axis Bounds: `0` and `100`. Click OK.
- Repeat the same procedure for the graph of column `Institutional`.
- You can change the display options of each of the graphs, by clicking a graph name in the Graph Management dialog box, and then clicking the Edit Graph button. When you are satisfied with the result, you can close the dialog box.
- From the Options menu, choose Legend. A legend is displayed.
- Close the graph window.
- Close the table window.

Summary: Graph display

– Data in columns can be displayed as graphs of different types: lines, steps, bars, needles and points.

– Several graphs can be combined in one graph window.