Parameter-related data in CommonGIS

1 General notes

A set of data can be viewed as consisting of units with a common structure, i.e. composed of *components* having the same meaning in each of the units. We shall call such units data *records*. For example, data about total population numbers in municipalities of a country in each census year have three components in each record: the municipality, the year, and the population number. This abstract view of data is independent of any representation model.

Any item (record) of data includes two conceptually different parts: one part defines the context of obtaining the data while the other part represents results of measurements, observations, calculations etc. obtained in the given context. The context may include the moment of time when the measurements were made, location in space, method of data acquisition, and the entity(-ies) the properties of which were measured (observed, calculated, ...). Thus, in our example, the municipality and the year define the context of measuring the population number.

We have found useful the general ideas concerning data structures and properties presented in the book "Architecture of systems problem solving" by G.J. Klir (Plenum Press, NY, 1985). Klir considers the situation of studying an object through observation of its properties. The properties can be represented as *attributes* taking on various *appearances* (manifestations). "For instance, if the attribute is the relative humidity at a certain place on the Earth, the set of appearances consists of all possible values of relative humidity (defined in some specific way) in the range from 0% to 100%". In the terminology we use in CommonGIS, Klir's "appearances" correspond to "values".

"In a single observation, the observed attribute takes on a particular appearance. To be able to determine possible changes in its appearance, multiple observations of the attribute must be made. This requires, however, that the individual observations of the same attribute, performed according to exactly the same observation procedure, must be distinguished from each other in some way. Let any underlying property *that is actually used* to distinguish different observations of the same attribute be called a *backdrop*. The choice of this term, which may seem peculiar, is motivated by the recognition that the distinguished property, whatever it is, is in fact some sort of background against which the attribute is observed." Klir's notion of "backdrop" corresponds to what we call "parameters".

According to Klir, there are three basic kinds of backdrops: *time*, *space*, and *population*. By "population" Klir means a set of any items, not only people. Examples of Klir's population are a set of manufactured products of the same kind, the set of words in a particular poem or story, a group of laboratory mice, etc.

Let us now give some more examples of attributes and parameters. Thus, the earlier mentioned census data (consisting of municipalities, years, and population numbers) characterise the population of a country. However, only the data component "population number" is directly related to the population and expresses some of its properties. The other two components do not provide any information about the phenomenon. Instead, they serve for reference of specific values of population

number to corresponding time moments (years) and territory fragments (municipalities). On this basis, we classify the year and municipality as the parameters of this dataset and the population number as the attribute. In general, parameters do not themselves contain any information about a phenomenon but relate items of this information (expressed by attributes) to different places, time moments, objects etc.

There is another difference between parameters and attributes: the values of parameters are chosen arbitrarily, and the corresponding values of attributes are fully determined by the choice made. Thus, in our example, the selection of the years when the population is counted is made arbitrarily by the authorities and could be, in principle, changed. The same refers to the municipalities for which the data are collected: one could decide to aggregate the data by smaller or by larger units of territory division, or to change the boundaries. At the same time, each value of the population number present in the database is inseparably linked to a specific year and a specific area. It is completely determined by the temporal and spatial references and cannot be set arbitrarily. Hence, parameters can be viewed as independent components of data and attributes as depending on them.

As we have mentioned, Klir distinguishes 3 possible types of parameters: space, time, and population (group of objects). An example of a population parameter is a set of political parties in data about distribution of votes by parties in an election. However, it should not be concluded that space, time, and population are *always* parameters. Klir noted: "Time, space, and population, which have special significance as backdrops, may also be used as attributes. For instance, when sunrise and sunset times are observed each day at various places on the Earth, the attribute is time and its backdrops are time and space ...". To give more examples, in data about moving objects, such as migratory animals, the observed locations are dependent on the objects and the selected moments of observation. Hence, here the space is an attribute. In results of an election the component reflecting which party won the election in each municipality is also an attribute.

Besides space, time, and population, other types of parameters may be encountered. Thus, the level of water in a river is an attribute in data about daily measurements of the water level. The same attribute will be a parameter in data about the flooded area depending on the level of water in the river. Hence, space, time, and population can be viewed as the most common types of parameters but not as the complete set of all possible types.

2 Parameters and attributes in CommonGIS

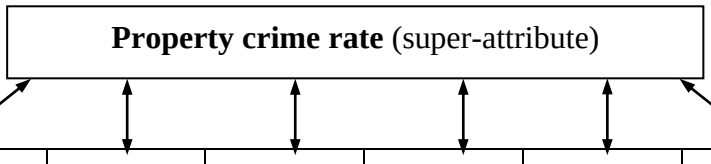
CommonGIS is primarily intended to deal with spatially referenced thematic data, i.e. attribute values referring to some spatial objects or locations. Hence, a CommonGIS dataset typically has one parameter of the type “space” (in short, spatial parameter), the values of which are these spatial objects and locations. However, there may be also other parameters.

Thematic data in CommonGIS are organised in tables with one record (row) per object (in principle, the objects are not necessarily spatial). Thematic data loaded in CommonGIS from external data sources (such as files or databases) must include object identifiers, which serve as references to the objects and play an important role in linking tables to geographic data. For correct loading of thematic data into CommonGIS, one must specify which field (column) of the data source contains the

identifiers of the objects. This is done either interactively in a dialog or by writing a special statement in a file (*.APP) describing a CommonGIS project (application).

The simplest case is when the set of the objects is the only parameter. In this case, each table column corresponds to one attribute and contains the values of this attribute referring to each of the objects.

In a case of multiple parameters, the internal data representation preserves the property of having exactly one table row per object. However, one attribute may be represented by multiple columns, where each column corresponds to a particular parameter value or combination of values of several parameters. Inside CommonGIS, there is still an individual attribute for each table column. However, there are also additional top-level attributes, or super-attributes, that unite together columns referring to different parameter values or combinations but containing the same object characteristic. The low-level attributes have references to the corresponding top-level attribute, which is called their "parent". The top-level attribute also has references to all its "children". The data structure is schematically shown below by an example.



State	**Property crime rate 1960**	**Property crime rate 1961**	**Property crime rate 1962**	**Property crime rate 1963**	**Property crime rate 1964**	**Property crime rate 1965**
Alaska	1544.9	1540.6	1564.6	1952.8	2016.0	2454.5
Minnesota	1424.0	1408.2	1498.9	1640.3	1925.5	1924.9
Washington	2175.5	2173.2	2270.1	2329.5	2636.6	2618.3
Montana	1985.4	1954.1	1928.8	2098.3	2023.1	2077.5
…	…	…	…	…	…	…

Here, "Property crime rate" is the super-attribute, or parent, for the attributes "Property crime rate 1960", "Property crime rate 1961", and so on. The same structure is used for any parameter-dependent attribute (not only time-dependent), for example, the attribute "Pesticide concentration in soil" specified for different types of soil. In this example, there will be one super-attribute "Pesticide concentration in soil" and as many columns as there are soil types, each column having its own "child attribute".

Here is an example with more than one parameters. The measurements of "Pesticide concentration in soil" might be done for different soil types (1st parameter), different pesticides (2nd parameter), and different years of observation (3rd parameter). The same data structure is used in this case as in a case of a single parameter: there is exactly one super-attribute for all columns with pesticide concentrations. However, each child attribute refers to a particular combination of values of the parameters "soil type", "pesticide", and "year".

3 Structures of external data sources

For correct handling of parameter-related data coming from some external source, CommonGIS needs to "know" what parameters these data involve and where these parameters and their values are. In an external source, the same data can be organised in different ways. Thus, the four example tables below have different structures but identical content:

Country	Gender	Age group	Population number
Albania	Male	Children	563953
Albania	Male	Adults	1104371
Albania	Male	Old people	86821
Albania	Female	Children	520186
...	...	...	...

Country	Gender	Population number (children)	Population number (adults)	Population number (old people)
Albania	Male	563953	1104371	86821
Albania	Female	520186	1026321	112252
Austria	Male	711127	2677100	453034
Austria	Female	681087	4672554	791762
...	...	...	...	...

Country	Age group	Population number (male)	Population number (female)
Albania	Children	563953	520186
Albania	Adults	1104371	1026321
Albania	Old people	86821	112252
Austria	Children	711127	681087
...	...	...	...

Country	Population number (male, children)	Population number (female, children)	Population number (male, adults)	Population number (female, adults)	Population number (male, old people)	Population number (female, old people)
Albania	563953	520186	1104371	1026321	86821	112252
Austria	711127	681087	2677100	2672554	453034	791762
...	...	...	...	...	...	...

The fourth table has the "canonical" form corresponding to the internal data representation in CommonGIS, i.e. there is exactly one table row for each spatial object. However, when CommonGIS loads such a table, it does not know the meaning of the columns. It cannot "guess" that the first column contains object identifiers and the remaining columns contain values of one and the same attribute corresponding to different combinations of values of two parameters, "Gender" and "Age group". Therefore, the system needs a formal description of this table in order to handle the data properly.

It should be noted that typical field names in a database are not so detailed and clear as in our example. They may look like "S0001" or "TNN" (the examples are taken from real CommonGIS projects). However, even if field names are quite clear for any human, CommonGIS cannot automatically "understand" them. Hence, some person who knows the semantics of a dataset must describe it in a formalised, machine-readable form.

CommonGIS does not require data to be transformed to the canonical structure before loading into the system. It can do the necessary transformation itself. However, it requires the original data structure to be properly described.

4 How to describe parameter-related data structures

4.1 Where data structures are described

Structures of parameter-related datasets are described in project (application) specification files. A CommonGIS application (project) file is an ASCII file with the extension *.APP. Application files are automatically created by the system when the user (after loading some data and setting visual parameters of map layers) selects the command "Save project" in the "File" menu.

A project description in an *.APP file specifies map layers and tables to be loaded in the system's memory for the visualisation and analysis. The layers are listed prior to the tables. Specification of a layer or a table may consist of several statements. A layer specification starts with the statement "LAYER…". A table specification starts with the statement "TABLEDATA…". The signal of the end of a layer or table specification is the next "LAYER…" or "TABLEDATA…" statement or the end of the file.

A description of a table structure must be included in the corresponding table specification, i.e. somewhere between the "TABLEDATA…" statement referring to this table and the next "TABLEDATA…" statement or the end of the file.

As it was demonstrated in the previous section, data can be organised in two different ways with respect to each individual parameter:

- There is a field (column) in the data source that contains values of this parameter. We shall call such a parameter "column parameter".
- The parameter is implicit: columns correspond to different parameter values, but the values themselves do not appear in the data. There is a person who knows the meaning of each column, including the correspondence to parameter values. If this person would want to include the data in a report and make the meaning clear to someone else, she/he would need to supply the table with a caption. In this

caption, the column titles would include the parameters and their values. Therefore, we call such implicit parameters "caption parameters".

Column and caption parameters are described differently. For the first case, CommonGIS provides a wizard that supports the creation of the parameter description. The wizard is activated through the menu item "Index a parameter-dependent table" in the "File" menu. If your table has only column parameters, you may use this wizard and then save the project. The description will be included in the project specification file. However, if your table has (also) caption parameters, the description has to be created manually. For this purpose, you will need to load a parameter-related dataset into the system, save the project, and then edit the resulting project file. For loading data into CommonGIS, use the menu command "File" > "Load data".

4.2 Description of a column parameter

A description of a column parameter consists of several lines. The first line must be

<PARAMETER>

and the last line in the description must be

</PARAMETER>

The upper and lower case letters are not distinguished; hence, it is also possible to write <parameter> or <Parameter>.

The lines between these two statements have the format

keyword=value

or

keyword=list_of_values

In the keywords, capital and small letters are not distinguished. Values may be enclosed in quotes. This is, in general, optional, but quotes become necessary when a value contains spaces or punctuation. Values in a list are separated by commas or semicolons.

Here is the explanation of the admissible keywords and their possible values. For better visibility in the text, the keywords are given in capital letters.

- COLUMN_NAME

Specifies the name of the column containing the values of the parameter. For example:

```
column_name="Scenario"
```

- DEPENDENT_COLUMNS

Specifies the columns that depend on this parameter. This must be a list of column names separated by commas or semicolons. For example:

```
dependent_columns=Lit(T/Ha),OrgL(T/Ha),Hum(T/Ha),CSoil(T/Ha)
```

The statement DEPENDENT_COLUMNS is optional. If it is absent, CommonGIS assumes that all columns of the table (except for the columns specifying object identifiers and names) depend on this parameter. In the above-given example, the

table contains the column "`Area(Ha)`" that does not depend on the parameter "Scenario". This column is not therefore included in the list of dependent columns.

When CommonGIS loads a table with a column parameter, it transforms it into the canonical form (see §2). For this purpose, it creates N columns from each parameter-dependent column, where N is the number of different parameter values found in the parameter column. Thus, in our example, the parameter "Scenario" has 4 different values: NAT, SCU, LRU, and ILL. Each of the dependent columns `Lit(T/Ha)`, `OrgL(T/Ha)`, `Hum(T/Ha)`, `CSoil(T/Ha)` will be transformed into 4 columns. From the column `Lit(T/Ha)`, CommonGIS will produce the columns `Lit(T/Ha) in scenario NAT`, `Lit(T/Ha) in scenario SCU`, `Lit(T/Ha) in scenario LRU`, and `Lit(T/Ha) in scenario ILL`. The other columns will be processed in the same way. However, the column `Area(Ha)`, which is not included in the list of dependent columns, will not be multiplied.

- ORDER

Specifies the desired order of parameter values. There are two possibilities:

- `order=sorted`

Indicates that the system must sort the values. The values are interpreted as strings and sorted alphabetically.

- `order=`*`list_of_values`*

Here, the values of the parameter are listed in the desired order. For example:

```
<Parameter>
column_name="Age class"
dependent_columns="BA(m2/Ha)"
order=1,2,3,4,5,6,7,8,9,10,11,12,13
</Parameter>
```

Here the values of the parameter "Age class" are integer numbers from 1 to 13. However, CommonGIS interprets them as strings. Therefore, sorting these values would result in the order 1, 10, 11, 12, 13, 2, 3, 4, and so on. This order is unusual and inconvenient. Therefore the person who described the table specified the desired order of the values to be `1,2,3,4,5,6,7,8,9,10,11,12,13`.

If the statement ORDER is absent, the parameter values will be presented to the user in the same order as they occur in the column.

This description method applies only to non-temporal column parameters. If a column parameter is temporal (i.e. its values represent some time moments), it must be described differently.

4.3 Temporal column parameters and temporal attributes

As we have explained in §1, time may play the role of either a parameter or an attribute. Temporal attributes and temporal column parameters are described in the same way.

Like space, time plays a particular role in CommonGIS. However, the system itself cannot distinguish time moments from other values in source data. It needs an indication where the time moments are and how they are encoded.

4.3.1 Specification of time moments in source data

Time moments may be specified as simple numbers, for example, years, or as complex structures, for example, dates consisting of a day in a month, a month, and a year. Compound time moments may be specified in two ways. First, all the components of a time moment may be packed in a single string. For example, the string "11.11.2003" represents the date November 11, 2003. In this case, there is a single table column containing the time moments encoded in this way. CommonGIS requires that all values in this column conform to one and the same scheme (template). For example, let the scheme be "dd.mm.yyyy". This means that each value in the column must start with 2 digits denoting the number of the day in the month, then a separator must come, then 2 digits denoting the number of the month, then again a separator, and then 4 digits denoting the year. The date April 1, 2003 must be encoded as "01.04.2003" but not "1.4.2003" and not "01.04.03".

The other possible way is to specify compound time moments as combinations of values in multiple columns. For example, calendar dates consisting of the day, month, and year, may be split into three table columns "Day", "Month", and "Year". CommonGIS is able to retrieve the date elements from these columns and construct dates from them provided that the data are properly described.

4.3.2 Time format schemes

Every time when a temporal parameter or attribute is described, it is necessary to specify the so-called time format scheme, which allows CommonGIS to interpret the values of the parameter.

For time moments represented by simple numbers, the time format scheme is just one of the following symbols (the meaning of each symbol is specified in the parentheses):

y (year)

m (month)

d (day)

h (hour)

t (minute)

s (second)

a (abstract number, for example, simulation step).

For compound time moments, the scheme is built from the above-listed symbols **y**, **m**, **d**, **h**, **t**, and **s** (but not **a**) and separators. Each symbol must be repeated so many times as the length of the corresponding component in the strings representing the time moments is, for example, "dd.mm.yyyy" or "dd/mm/yyyy". Actually, any symbol different from **y**, **m**, **d**, **h**, **t**, and **s** is treated as a separator. In transforming strings to dates according to a specified time format scheme, CommonGIS ignores the symbols that occur on the separator places. Therefore, the schemes "dd.mm.yyyy" and "dd/mm/yyyy" are, from the system's point of view, the same. CommonGIS does not check if dots or slash symbols are actually used to separate the date components. However, the schemes "dd.mm.yyyy" and "ddmmyyyy" are different. In the first case, the system will try to retrieve the month numbers from the positions 4 and 5 of each string and the years from the positions 7 to 10. In the second case, the system

will try to get the month numbers from the positions 3 and 4 and years from the positions 5 to 8.

4.3.3 Description of time references

As we have already mentioned, temporal attributes and temporal column parameters are described in the same way. There is only one small difference that will be indicated later.

A column or a group of columns that specify time moments must be described through a sequence of statements starting with

<TIMEREFERENCE>

and ending with

</TIMEREFERENCE>

As usual, capital and small letters are not distinguished.

If the time moments are specified in a single column, the description must include the statement

"column_name"="scheme"

Here *"column_name"* is the name of the column containing time moments. The use of quotes is optional. However, it is mandatory if the name of the column contains spaces or punctuation. The *"scheme"* is the time format scheme (see §4.3.2) telling CommonGIS how to interpret the values in the column.

If the time moments are specified in a group of columns, there must be a statement

"column_name"="scheme"

for each column of the group.

Here are a few example illustrating the various possibilities for specifying time moments:

```
"Step"="y"
```

This means that there is a column with the name "Step" in the source table, and this column contains time moments specified as simple numbers. The system should interpret these numbers as years.

```
"Date"="dd.mm.yyyy"
```

This means that there is a column with the name "Date" in the source table, and this column contains strings conforming to the scheme "dd.mm.yyyy". All the strings consist of 10 characters. The first 2 characters of each string specify the number of the day, the 4^{th} and 5^{th} characters specify the number of the month, and the characters from 7 to 10 specify the year.

```
"Day"="d"
"Month"="m"
"Year"="y"
```

This means that there are 3 columns in the source table, "Day", "Month", and "Year", the values from which must be integrated into compound time moments (dates). The column "Day" contains the day numbers, the column "Month" – month numbers, and the column "Year" contains years. The strings in these columns may have variable

lengths. Thus, it is not necessary that the 1st day of a month is represented by the string "01"; this may be simply "1".

Whatever the way of specifying time moments in source data is, CommonGIS always derives from them a single temporal attribute. The values of this attribute are time moments (special Java objects) rather than strings. The name of this derived attribute may be specified in the time reference description using the statement

ATTR_NAME=*"name"*

Here ATTR_NAME is a keyword and *"name"* is the name to be given to the new attribute. For example:

```
<TimeReference>
"Day"="d"
"Month"="m"
"Year"="y"
attr_name="Date"
</TimeReference>
```

According to this description, CommonGIS will retrieve date components from the columns "Day", "Month", and "Year" of the source table and construct from them special Java objects representing dates. It will create a new attribute the values of which are these Java objects and give the name "Date" to it (if the description would not contain the statement `attr_name="Date"`, the new attribute would receive the default name "_time_"). This attribute will be added to the internal representation of the table. The original columns "Day", "Month", and "Year" will be present in the internal table as well.

In a case when time moments are specified in a single column of the original table, it is possible to make the system replace the original column by the derived attribute in the internal representation of the table. For this purpose, the following statement is used:

```
attr_name="#replace#"
```

For example:

```
<TimeReference>
"Step"="y"
attr_name="#replace#"
</TimeReference>
```

According to this description, CommonGIS will transform the values from the column "Step" of the source table into special Java objects representing years, create a new attribute with these Java objects as values, and replace the original column "Step" by this new attribute. The new attribute will receive the same name "Step". If the statement `attr_name="#replace#"` is not included in a time reference description, the new attribute will be given the default name "_time_" and simply added to the internal table, and the original column "Step" will be preserved.

If the time references in the source data consist of several components (as, for example, dates), it may be desirable to specify how the system must display them to the user. This is done by specifying the time format scheme (see §4.3.2) for the derived temporal attribute:

ATTR_SCHEME=*"scheme"*

This statement is especially useful when the components of the time references are specified in several columns of the original table. If the format scheme is not specified, the system will try to apply one of its default schemes. The default scheme for dates is "dd.mm.yyyy", for times of the day - "hh:tt:ss", and for "complete" time moments including both date and time of the day the scheme is "dd.mm.yyyy; hh:tt:ss".

If compound time references are specified in a single column of the source table, the scheme describing the structure of the values in this column will be also used for displaying the values of the derived temporal attribute unless a different scheme is specified in the statement ATTR_SCHEME=*"scheme"*.

A source table may contain several time references with different meanings. CommonGIS recognises the following meanings:

OCCURRED_AT

VALID_FROM

VALID_UNTIL

The default meaning is OCCURRED_AT. If the meaning of the described time reference is different, this should be indicated through the statement

MEANING=*meaning*

Here *meaning* must be one of the keywords "OCCURRED_AT", "VALID_FROM", and "VALID_UNTIL".

Temporal parameter. If the described temporal reference is a parameter, this must be indicated through the statement

IS_PARAMETER=YES

Besides, it may also be necessary to specify which of the columns depend on this parameter. This is done in the same way as for non-temporal parameters:

DEPENDENT_COLUMNS=*list_of_columns*

(see §0).

Here are few example descriptions of temporal attributes and parameters taken from various CommonGIS projects:

```
<TimeReference>
"Year"="y"
"Month"="m"
"Day"="d"
attr_name="Date"
</TimeReference>
```

Describes a temporal attribute with values composed of days, months, and years, which are specified in three columns, "Year", "Month", and "Day", of the source table. The attribute has the name "Date".

```
<TimeReference>
"step"="y"
attr_name="#replace#"
is_parameter=yes
dependent_columns="BA(m2/Ha)"
```

```
</TimeReference>
```

Describes a temporal parameter (specifically, simulation step). The source table contains a single parameter-dependent column, `"BA(m2/Ha)"`.

```
<TimeReference>
meaning="OCCURRED_AT"
"JJJJMM"="yyyymm"
attr_name="Datum"
attr_scheme="mm.yyyy"
is_parameter=yes
dependent_columns="QN";"TNN";"TNM";"TMM";"TXM";"TXX";"SOS";"NMM";"RSS
";"RSX";"FMM";"FXX"
</TimeReference>
```

Describes a temporal parameter the values of which are composed from months and years. The original values are contained in the column with the name "`JJJJMM`" and conform to the time format scheme `"yyyymm"`, i.e. the values are strings of the length 6 characters where the first 4 characters specify the year and the next 2 characters the month. The resulting temporal parameter will have the name "Date"; its values will be displayed according to the scheme `"mm.yyyy"`. The table contains 12 columns depending on this parameter; the names of the columns are listed in the statement dependent_columns. It must be noted that, although this statement is broken here into two lines, there must be a single line with column names in the *.APP file.

```
<TimeReference>
meaning="OCCURRED_AT"
"year"="y"
attr_name="#replace#"
is_parameter=yes
dependent_columns="Population";"Index offenses total";"Violent crime
total";"Murder and nonnegligent Manslaughter";"Forcible
rape";"Robbery";"Aggravated assault";"Property crime
total";"Burglary";"Larceny-theft";"Motor vehicle theft";"Index
offense rate";"Violent Crime rate";"Murder and nonnegligent
manslaughter rate";"Forcible rape rate";"Robbery rate";"Aggravated
assault rate";"Property crime rate";"Burglary rate";"Larceny-theft
rate";"Motor vehicle theft rate"
</TimeReference>
```

This is also a description of a temporal parameter. It contains a quite long list of dependent columns. All the column names must be in the same line in the *.APP file.

4.4 Description of a caption parameter

Description of a caption parameter is more complex than that of a column parameter. First, the name and values of the parameter are not present in the source data and must be explicitly given in the description. Second, the correspondence between the parameter values and table columns must be specified.

A description of a caption parameter starts with the statement

<CAPTIONPARAMETER>

and ends with the statement

</CAPTIONPARAMETER>

Between these two statements, the following statements must come:

PARAM_NAME=*"name"*

Specifies the name of the parameter. The quotes are necessary only if the name contains spaces or punctuation.

IS_TEMPORAL=YES *or* IS_TEMPORAL=NO

Indicates whether the parameter is temporal. This statement is optional. By default, the parameter is assumed to be non-temporal.

TIME_SCHEME=*"scheme"*

For a temporal parameter, this statement specifies the time format scheme (see §4.3.2). For a non-temporal parameter, this statement is not used.

VALUES=*list_of_values*

Specifies the values of the parameters. The values in the list are separated by commas or semicolons. If a value contains spaces or punctuations, it should be enclosed in quotes.

ORDER=SORTED *or* ORDER= *list of values*

Specifies the desired order of the parameter values if it is different from the order of values in the statement VALUES. This statement is analogous to the ORDER statement for a column parameter (see §0).

Besides these statements starting with keywords, there must be one or more statements specifying the correspondence between table columns, attributes, and parameter values. Each correspondence statement has the format

attribute_name=list_of_column_numbers

This statement lists all the columns referring to one and the same attribute but to different parameter values. The order of the columns must correspond to the order of the parameter values in the statement VALUES. There are 2 possibilities for specifying the column list. First, this may be a list of column numbers separated by commas. Second, this may be a column range. In this case, the first and the last columns are specified, delimited by a semicolon. Instead the last column number, one may write the keyword LAST, for example: "Some attribute"=1;LAST.

Let us consider several examples of caption parameters.

```
<CaptionParameter>
param_name="Step"
is_temporal=yes
values=1;2;3;4;5;6;7;8;9;10
time_scheme="a"
"Conc. (ug/l)"=6,8,10,12,14,16,18,20,22,24
"Soil Water (mm)"=7,9,11,13,15,17,19,21,23,25
</CaptionParameter>
```

This sequence of statements describes a temporal caption parameter with the name “Step” (step of simulation) and values from 1 to 10. Two attributes depend on this parameter: `"Conc. (ug/l)"` and `"Soil Water (mm)"`. The columns corresponding to these two attributes alternate in the table: the columns with values of `"Conc. (ug/l)"` have even numbers starting from 6, and the columns with the values of the other attribute have odd numbers starting from 7. The columns 6 and 7 correspond to the value 1 of the parameter “Step”, the columns 8 and 9 – to value 2, and so on.

```
<CaptionParameter>
param_name="Age group"
```

```
is_temporal=no
values="Total";"0-5";"6-17";"18-29";"30-64";"65-over"
"Population"=2,LAST
</CaptionParameter>
```

This sequence of statements describes a non-temporal caption parameter “Age group” with values `"Total", "0-5", "6-17", "18-29", "30-64",` and `"65-over"`. The table contains only one attribute, “Population”, depending on this parameter. All table columns starting from the 2nd belong to this attribute. The column 2 corresponds to the parameter value `"Total"`, column 3 – to `"0-5"`, column 4 – to `"6-17"`, and so on. The table contains as many columns related to the attribute “Population” as there are parameter values. So, the last column corresponds to the parameter value `"65-over"`.

```
<CaptionParameter>
param_name="Species"
is_temporal=no
values=Pine;Spruce;Birch
"Price"=3,4,5,6,7,8
</CaptionParameter>

<CaptionParameter>
param_name="Timber type"
is_temporal=no
values=Logs;Pulpwood
"Price"=3,6,4,7,5,8
</CaptionParameter>
```

The table has 1 attribute, “Price”, and 2 caption parameters, “Species” and “Timber type”. Both parameters are non-temporal. The first parameter has 3 values: “Pine”, “Spruce”, and “Birch”. The second parameter has 2 values: “Logs” and “Pulpwood”. Respectively, there are 6 columns (numbered from 3 to 8) with values of the attribute “Price” referring to various combinations of parameter values. Column 3 corresponds to “Pine” and “Logs”, column 4 – to “Spruce” and “Logs”, column 5 – to “Birch” and “Logs”, column 6 – to “Pine” and “Pulpwood”, column 7 – to “Spruce” and “Pulpwood”, and column 8 – to “Birch” and “Pulpwood”. Pay attention to the way of specifying the correspondence between columns and parameter values. You may notice that the lists of columns in both descriptions are longer than the lists of parameter values.

In general, if there are two or more caption parameters, each attribute has two or more columns referring to one and the same parameter value. The rule for listing column numbers in the description of each caption parameter is the following. Let the parameter has N values and the attribute has M columns, where M>N. Then the column numbers are listed so that the items from 1 to N in the column list correspond to the parameter values from 1 to N in the parameter values list (the order must be heeded!), the items from N+1 to N*2 in the column list correspond again to the parameter values from 1 to N, and so on until all M columns appear in the list. If there is no column corresponding to some parameter value, 0 (zero) must stand on this place in the column list.

5 Table preprocessing in CommonGIS

When a table is loaded into CommonGIS from some external source, and this table has parameters or/and temporal attributes, the system performs a sequence of transformations (preprocessing) before allowing any use of this table. One of the

results of this preprocessing is the "canonical" table structure with exactly one row per object (see §3). The transformations take place after the table is completely loaded into the system's memory. They are done in the following sequence.

First, the system looks for descriptions of caption parameters (<CaptionParameter> … </CaptionParameter>). If any such description is found, the system constructs internal objects representing these parameters, creates "parent" attributes for the columns depending on the parameters, and sets references to the relevant parameter values (see §2).

Second, the system looks for descriptions of temporal attributes or temporal column parameters (<TimeReference> … </TimeReference>). On the basis of any such description, CommonGIS produces a temporal attribute, i.e. an attribute the values of which are special Java objects representing time moments. The column for this attribute is added to the table. If the time reference description contains the statement ATTR_NAME="#replace#", the original column, from which the new column was constructed, is removed. If the description specifies a temporal parameter rather than temporal attribute, CommonGIS automatically creates a column parameter description for the new column, which is then processed on the next stage.

Third, CommonGIS looks for column parameter descriptions (<Parameter> … </Parameter>), including the descriptions automatically created on the second step. For each column parameter, CommonGIS examines the content of the corresponding column and retrieves all possible parameter values from it. After that, CommonGIS creates a new table. For each parameter-dependent column of the source table, it constructs N columns in the resulting table, where N is the number of different parameter values found. Then CommonGIS copies the attribute values from the source table to the resulting one so as to preserve the correspondence between the attribute values and parameter values, as is schematically shown below.

Source table:

Object	**P** *(parameter)*	**A1** *(attribute)*	**A2** *(attribute)*
O1	pv1	a1v1	a2v1
O1	pv2	a1v2	a2v2
O1	pv3	a1v3	a2v3
O2	pv1	a1v4	a2v4
O2	pv2	a1v5	a2v5
O2	pv3	a1v6	a2v6

Resulting table:

Object	**A1; P=pv1**	**A1; P=pv2**	**A1; P=pv3**	**A2; P=pv1**	**A2; P=pv2**	**A2; P=pv3**
O1	a1v1	a1v2	a1v3	a2v1	a2v2	a2v3
O2	a1v4	a1v5	a1v6	a2v4	a2v5	a2v6

For each group of columns derived from the same source attribute, CommonGIS creates a parent attribute (see §2). Each column receives a reference to its parent and to the corresponding parameter value.

Parameter-independent columns of the source table are not multiplied. For every such column, CommonGIS creates a single column in the resulting table. Normally, each object has the same value of a parameter-independent attribute for all possible values of the parameter. Therefore, although the resulting table contains less rows than the source table, no data are lost but only repetitions of the same values reduced.

After the transformation on the basis of the first column parameter is done, CommonGIS replaces the source table by the resulting table and looks for the next description of a column parameter. The transformation will be performed as many times as there are different column parameters. If there are k column parameters P_1, P_2, …, P_k in the source table with N_1, N_2, …, N_k possible values, respectively, the system will rewrite the table k times. Finally, each column parameter will be turned to a caption parameter, and each parameter-dependent column of the source table will be multiplied $N_1 * N_2 * \ldots * N_k$ times. The ultimate goal is to get a canonical table structure with one table row per object. This canonical table will, probably, have hundreds or thousands columns. There is a CommonGIS project in which a table has about 30 000 columns after restructuring.

Only when the restructuring is completed, any operations with the data from this table (visualisation, filtering, computation, etc.) become possible.

As it may be guessed, the procedure of transforming a table with column parameters into a canonical form may be rather time-consuming in cases when there are several column parameters that have many different values. Therefore, if you need to work in CommonGIS with this table repeatedly, it makes sense to export the transformed table and include this transformed table in the project instead of the original table. In order to simplify the process of describing the structure of the exported table in the project file, CommonGIS automatically stores the description of the parameters of this table (note that they all are now caption parameters). Thus, if you export your table to the file “transformed_table.csv”, the system will automatically create a file with the name “transformed_table.csv.descr” in the same directory and write the description of the table structure in this file. You can include this description in the project (*.app) file in the section describing the table “transformed_table.csv”.