ThH 12.01.2006

The NPM-system

Functionality offered and preconditions for using the system

1 Introduction

The NPM-system is a part of the PC-Axis system. The object of the NPM-system is to give the creators of statistical tables the opportunity to annotate every single data cell in the tables, using a set of predefined markers for status and quality of the cell values. The predefined markers are called “NPM-characters” in this document.

Use of the NPM-system depends upon certain additions having been made to the sql-database as well as to some of the system files in the PC-Axis system. Most of these additions were introduced in versions 2.0 and 2.1 of the datamodel for the sql-database. An overview of the additions will be given in this document.

From this it follows that the extended functionality in the NPM-system is accessible only if your sql-database is version 2.1 or higher.

2 Additions to the sql-database

The additions to the sql-database comprise the inclusion of one or more new database tables, the addition of columns to existing database tables, and the addition of a smaller number of records to existing database tables.

2.1 New database tables in the sql-database

The NPM-system is based upon the definition of a set of predefined "NPM-characters" for marking data cells in the statistical tables. The definition of the NPM-characters is stored in a database table called **SpecialCharacter**:

Table: SpecialCharacter					
Column name	**Datatype**	**NULL**	**Pr. key**	**For. key**	**Description**
CharacterType	varchar(2)	N	TRUE	FALSE	The code for the NPM-character
PresCharacter	varchar(20)	N	FALSE	FALSE	The character(s) to be presented in the table
AggregPossible	char(1)	N	FALSE	FALSE	Can a data cell marked with the special character(s) enter into an aggregation?
DataCellPres	char(1)	N	FALSE	FALSE	Should the data cell value be presented, or just the spec. char?
DataCellFilled	char(1)	N	FALSE	FALSE	What data can be stored in the data cell (N=nothing, V=value, F=facultative (filled or empty), 0=zero
PresText	varchar(200)	Y	FALSE	FALSE	Freetext explanation of the semantics of the NPM-character
UserId	varchar(20)	Y	FALSE	FALSE	
LogDate	Datetime	Y	FALSE	FALSE	

In this table the definition of the NPM-characters for an instance of the statistical database should be stored. A possible set of definitions may for example be:

Character-Type	**Pres-Character**	**Aggreg-Possible**	**DataCell-Pres**	**DataCell-Filled**	**PresText**	**User Id**	**Log-Date**
01	.	Y	N	0	Category not applicable		
02	..	N	N	N	Data not available		
03	…	N	N	N	Data not yet available		
04	:	N	N	N	Not for publication		
05	-	Y	N	0	Nil		
15	e	Y	Y	V	Estimated value		
20	*	Y	Y	V	Provisional or preliminary figure		

Guidelines and rules for filling in the data table:

- The **CharacterType** column contains the codes to be stored in the data tables in the statistical database.
- The **PresCharacter** column contains the character(s) that will be presented in the statistical tables.

- **AggregPossible**: If the value of the data cell may be used in arithmetical operations (in this case: aggregations), AggregPossible should be Y. If not, it should be N.
- **DataCellPres**: If the value of the data cell should be presented in the statistical table in addition to the special character(s), DataCellPres should be Y. If only the special character should be presented and not the data cell value, DataCellPres should be N.
- **DataCellFilled**: DataCellFilled shows what values the data cell may contain. There is a connection beween AggregPossible, DataCellFilled and the semantics of the NPM-character. If AggregPossible = Y, then it is possible to use the value of the data cell in calculations (e.g. aggregations), and, consequently, it should be ascertained during the loading of the table data that the table cell contains a valid value. If AggregPossible = N, on the other hand, the data cell should be empty, since the data cell is not supposed to take part in any calculation.

  Possible values for DataCellFilled are:
  - V - Value. The data cell should contain a value.
  - N - Nil. The data cell should be empty.
  - F - Facultative. The data cell may contain a value (e.g. 0) or be empty.
  - 0 - Zero. The data cell should contain a zero.

Due to **limitations** in the px-file format all NPM-characters with AggregPossible=Y and DataCellPres=N (nil-symbols) will be **stored** in the same way in the px-file (as a "-") and **presented** in the same way in the statistical tables (with the character defined by DataSymbolNil in the table MetaAdm, see chapter 2.3).

Due to **limitations** in the px-file format there should be a maximum of 6 special characters defined having both AggregPossible=N and DataCellPres=N. The 6 special characters are **stored** in the px-file as sequences of 1 to 6 dots, i.e ".", "..", "…", "….", "…..", and "……". The 6 PC-Axis dot sequences are assigned to the special characters of the SQL database in the following way:

- The records of the table SpecialCharacter are read in the order of the column CharacterType.
- The first special character with the properties AggregPossible=N and DataCellPres=N is assigned the PC-Axis dot sequence "."
- The next special character with the properties AggregPossible=N and DataCellPres=N is assigned the PC-Axis dot sequence "..", and so forth up to the 6th special character with the properties AggregPossible=N and DataCellPres=N, which is assigned the dot sequence "……".
- If more than 6 special characters with the properties AggregPossible=N and DataCellPres=N have been defined in the table SpecialCharacter, the excess ones will be treated as equal to the 6th one in the series.

If the statistical database is a multilingual one, the database table **SpecialCharacter** should be defined for the main language, and supplementary database tables for each of the additional languages should be added. For instance: A supplementary table for the special characters for English should have the following name and structure:

Table: SpecialCharacter_Eng					
Column name	**Datatype**	**NULL**	**Pr. Key**	**For. key**	**Description**
CharacterType	varchar(2)	N	TRUE	TRUE	The code for the NPM-character
PresCharacter	varchar(20)	N	FALSE	FALSE	The character(s) to be presented in the table
PresText	varchar(200)	Y	FALSE	FALSE	Freetext explanation of the semantics of the NPM-character(s)
UserId	varchar(20)	Y	FALSE	FALSE	
LogDate	datetime	Y	FALSE	FALSE	

The column CharacterType in the supplementary table serves both as the primary key for the table and as foreign key towards the database table **SpecialCharacter**.

2.2 Addition of columns to (existing) database tables

New columns have to be created in one metadata table and possibly in some data tables.

2.2.1 Additions to metadata tables

MainTable

The table MainTable will need one new column (it may already be implemented in some databases):

Table: MainTable					
Column name	**Datatype**	**NULL**	**Pr. key**	**For. key**	**Description**
SpecCharExists	char(1)	N	FALSE	FALSE	Y/E/N for using NPM-characters in this table

SpecCharExists – N means that the extraction program will not look for NPM-characters in the special NPM-columns in the data tables (see the next paragraph). E means that the special NPM-columns have been defined in the sql data tables, but the extraction program should not use them. Y, on the other hand, means that the extraction program a) expects the data table to contain the special NPM-columns, and b) will look for NPM-characters in these columns (and process them if there are any).

Content

The table Content will need one new column as well:

Table: Content					
Column name	**Datatype**	**NULL**	**Pr. key**	**For. key**	**Description**
PresMissingLine	varchar(2)	Y	FALSE	FALSE	Contains either NULL or a reference to a record in the table SpecialCharacter

PresMissingLine - This column is is intended and designed as a addition to the existing column PresCellsZero. The addition will offer more possibilities for the user.

PresMissingLine contains information for how data cells that have not been stored in the sql-database should be presented by PC-Axis when the resulting output table matrix is created. PresMissingLine relies on the value of the column PresCellsZero and on the property **DefaultCodeMissingLine**, which should be stored in the table MetaAdm (see ch 2.3). PresMissingLine should only be filled in when PresCellsZero = N.

The combination of PresCellsZero, PresMissingLine and DefaultCodeMissingLine allows for a number of possibilities. The main cases are (see table at next page):

- **PresCellsZero = Y**. The value of the data cells of missing records are interpreted (and presented) as zeroes (0). The value of PresMissingLine and DefaultCodeMissingLine is irrelevant in this case.
- **PresCellsZero = N**.
  - **PresMissingLine = <Code>**. PresMissingLine must contain one of the **CharacterType** codes from the table **SpecialCharacter**. This NPM character will be used for the data cells of the missing records.
  - **PresMissingLine = NULL**. In this case the NPM character defined in **DefaultCodeMissingLine** will be used for the data cells of the missing records.

The cases above are summarized in the table below:

		PresMissingLine	
	Values	**NULL (empty)**	**Contains a code for a special character**
PresCellsZero	**Y**	**PresCellsZero** indicates that the data cell value of missing records should be set = 0. **PresMissingLine** is not relevant.	**PresCellsZero** indicates that the data cell value of missing records should be set = 0. The code in **PresMissingLine** is irrelevant and will be ignored
	N	**PresCellsZero** indicates that the data cell value of missing records should be interpreted in a special way. Since **PresMissingLine** is empty, the special character stored in **DefaultCodeMissingLine** will be used.	**PresCellsZero** indicates that the data cell value of missing records should be interpreted in a special way. The special character stored in **PresMissingLine** will be used.

2.2.2 Additions to tables with statistical data

In the introduction to this documentation it was stated that the object of the NPM-system is to give the creators of statistical tables the opportunity to annotate every single data cell in the tables. In chapter 2.1 the system of NPM-characters used for this annotation was outlined. The annotation proper consists of storing the code for NPM-characters in dedicated columns in the data tables.

As an example consider the database table schema below.

A data table		
Column	**Datatype**	**Definition**
Variable1	varchar(20)	Variable 1, e.g. Region
Variable2	varchar(20)	Variable 2, e.g. Age
Variable3	varchar(20)	Variable 3, e.g. Sex
Time	varchar(20)	Time
Inhabitants	numeric	Content-column (measurement/estimate/enumeration)
Deaths	numeric	Content-column (measurement/estimate/enumeration)
Inhabitants _x	varchar(2)	The NPM-column for annotation of the Inhabitants' data cells
Deaths_x	varchar(2)	The NPM-column for annotation of the Deaths' data cells

The example shows a data table with three columns for variables (Variable1-Variable3), one column for Time and two columns for content, Inhabitants and Deaths. The two last columns in the table are the dedicated NPM-columns, where codes for the NPM-characters may be inserted. The naming convention for the NPM-columns is very simple: Add "_x" to the name of the corresponding content columns.

As the example shows, there should be one NPM-column for each content column. The position of the columns in the data table is not important. To avoid confusion, it is wise **not** to let the names of the regular content columns end in "_x".

The administrator of the statistical database might want to consider creating NPM-columns in all database tables, even if there are no plans at the moment for using NPM-characters in the statistical tables. In most database systems empty columns occupy little space or no space at all.

2.3 Addition of records to existing database tables

This type of additions concerns one data table in the database, **MetaAdm**. The following records should be added to the table (the text in the Value columns are examples):

Property	Value	UserId	LogDate
DATANOTAVAILABLE	..		
DATANOTESUM	++		
DATASYMBOLNIL	-		
DATASYMBOLSUM	##		
DEFAULTCODEMISSINGLINE	01		
PXDATAFORMAT	KEYS50		
KEYSUPPERLIMIT	1000000		

The four first records concern the internal workings of the NPM-system (see "Summation rules" later in this document for a more thorough explanation). The two last records contain instructions for how data extracted from the sql-database should be stored by PC-Axis.

2.3.1 DataNotAvailable

Occasionally empty (blank) data cells are stored in the sql-database. If the NPM-system has been implemented in the sql-database, these cells should have been marked with an NPM-character that explains why the cell value is missing.

If there is no NPM-character attached to an empty cell (either because NPM is not used for the maintable, or because the data cell for some other reason was not marked with an NPM-character), PC-Axis will have to "guess" the "meaning" of the empty data cell, i.e. how to store it in the px-file and present it in the resulting statistical tables.

During the data extraction empty data cells from the sql-database are always treated internally in the extraction program as instances of a category 3 NPM-character (i.e. an unknown or confidential value, more about categories of NPM-characters later in this document). The property **DataNotAvailable** gives you (the database administrator) the opportunity to determine how PC-Axis will **present** the empty data cells in statistical tables.

If the maintable under consideration uses NPM-characters, PC-Axis will compare the value of DataNotAvailable with the column PresCharacter in the database table **SpecialCharacter**. Valid values for DataNotAvailable is the PresCharacter-text of one of the NPM-characters defined in the database table SpecialCharacter, and valid NPM-characters for this comparison are the ones with AggregPossible=N and DataCellPres=N.

If the property **DataNotAvailable** is not included in the table MetaAdm, PC-Axis will assign the NPM-character ".." to the empty data cell. If MetaAdm contains a value for **DataNotAvailable** that is not equal to the PresCharacter-text of a valid NPM-character, PC-Axis will use the value given in MetaAdm for presentation of the empty cell and treat this value as an NPM-character of category 3.

Observe that DataNotAvailable concerns how PC-Axis will **present** the empty data cells in the statistical tables.

2.3.2 DataNoteSum

This property corresponds to the property with the same name in the px-file format. The property is copied to the resulting px-file as

DATANOTESUM="++";

The **value** of the property is the symbol used in the px-files in the following case:

- The program for extraction of data from the sql-database has tried to aggregate two (or more) data cells annotated with two (or more) different NPM-characters.
- The combination of these NPM-characters defines the data cells as a "summable" entity (see summation rules later in this document).

- The sum for the entity is stored in the px-file. In addition a DATANOTE is created that attaches the symbol DataNoteSum to the data cell. The datanote shows that the data cell was produced as a sum of entities with different (but summable) markers.
- When the sum is presented in the statistical table the DataNoteSum-symbol is presented as a marker for the figure in the data cell (e.g. 1234,5++).

If no presentation text is given for **DataNoteSum** in the table MetaAdm, the string "++" will be used.

2.3.3 DataSymbolNil

This property shows the symbol that PC-Axis should **present** for the value of a data cell with absolutely nothing in it (absolute nil), to distinguish the cell value from 0 (which may conceal a small value, e.g. 0.1).

In the px-file all data cells of this type is **stored** with the cell value "-" (no choice here). DataSymbolNil determines how this internal "-" will be **presented** in the statistical tables. In the example given here the same symbol has been chosen for both storing and presentation.

If no presentation text is given for **DataSymbolNil** in the table MetaAdm, the character "-" will be used.

2.3.4 DataSymbolSum

This property corresponds to the property with the same name in the px-format. The name of the property is poorly chosen, as the symbol is used in cases where no sum is possible! The reader will benefit from (mentally) substituting **DataSymbolNoSum** for the name of the property while reading this document (and perhaps otherwise as well…). The property is transferred to the resulting px-file as

DATASYMBOLSUM="##";

The **value** of the property is the symbol to be presented in the statistical table in the following case:

- The program for extraction of data from the sql-database has tried to aggregate two (or more) data cells annotated with two (or more) different NPM-characters.
- Unfortunately the combination of these NPM-characters constitutes an "unsummable" entity (see summation rules later in this document).
- In the px-file the result of this missing aggregation is stored as the symbol "……." (i.e. 7 dots).
- In the statistical tables produced from the px-file the symbol defined by DataSymbolSum (here the ##) is presented.

If no presentation text is given for **DataSymbolSum** in the table MetaAdm, the string "##" will be used.

2.3.5 PxDataFormat

The **PxDataFormat** property determines the strategy for storing data extracted from the sql-database in the resulting px-file. There are 2 alternatives for the Value column:

MATRIX – This keyword states that all extractions should be stored in the regular matrix format.

KEYS40 – The second alternative is the text **KEYS** followed by a number larger than 0. The number points to a ratio between the number of records read from the sql-database and the number of records in the specified output matrix.

$$\text{Ratio} = \frac{\text{No of recs read} * 100}{\text{No of recs in output matrix}}$$

The equation shows that the number should be interpreted as a percentage. If, for instance, the end user has specified an output matrix (statistical table) which will contain 1000 data cells and only 400 of these data cells are stored in (and read from) the sql-database, the ratio of stored data cells compared to the number of data cells in the output matrix is 40 per cent.

The ratio can vary from a very small number (close to zero) to 100 (no elimination and no aggregation) and even much higher (when eliminating or aggregating or both).

If the ratio percentage is equal to or lower than the number stated in **PxDataFormat**, the Keys-format will be used for storing the output matrix in the px-file (with a possible exception stored in the property **KeysUpperLimit**, see below). Otherwise, the matrix format will be used.

If no text is given for **PxDataFormat** in the table MetaAdm, the value "**KEYS20**" will be used.

2.3.6 KeysUpperLimit

In addition to the keyword PxDataFormat also the keyword KeysUpperLimit may influence the choice of format for the resulting px-file. The reason for this is the way PC-Axis handles the two formats. When reading a px-file in matrix format (a "classic" px-file), PC-Axis does not need to read more of the output matrix than the part that is to be displayed in the grid. When reading a px-file in Keys-format, however, all the keys must be read before PC-Axis can determine which of them to display. This means that the demand for internal memory is far greater for px-files in Keys-format than for px-files in matrix format.

The demand for internal memory can be quantified as a function of the number of data cells in the output matrix. If this demand is higher than the amount of available memory, PC-Axis will not be able to process the px-file. The number of data cells in the output matrix is calculated as the product of selected values for each variable (time included) times the number of selected contents. This is the number of data cells that PC-Axis will need to hold in the internal memory of the computer.

The keyword KeysUpperLimit contains the upper boundary for the output matrices for which the Keys-format can be used. If the output matrix is larger than this number, the px-file will be stored in the matrix format.

As the amount of accessible memory will differ from one computer to another, it is not possible to settle for an accurate limit. Through the keyword KeysUpperLimit each installation can determine its own limit. Our experience is that too large values in some cases may cause problems for the computer.

If no text is given for **KeysUpperLimit** in the table MetaAdm, the value "2 000 000" will be used.

2.3.7 Use of the Keys-format

Use of the Keys-format is only indirectly relevant to the NPM-system. Storing the data part of the px-files in the Keys-format is preferable for a number of data matrices. As the processing of the NPM-system creates an extra overhead on the extraction of data from the sql-database, using the Keys-format may in many cases reduce the processing time considerably.

Some tentative guidelines for using the Keys-format follow. The term "matrix" in the guidelines refers the output matrix, i.e. the matrix of the statistical table that the end user have specified.

- Storing data in the Keys-format has the advantage that missing database records ("category not applicable" or "nil") do not have to be brought into the processing. The Keys-format is, consequently, preferable for sparsely populated matrices. For densely populated matrices there are generally only small (or no) benefits to be gained from using the Keys-format.
- An exception to the last point is the case of larger data matrices. In these cases use of the Keys-format makes the processing of the extraction easier for the computer and should be the preferred alternative.
- On the other hand: If the data matrix is very large, the Keys-format is not an viable alternative, because the amount of memory in the computer sets a limit for PC-Axis' capacity for handling large data matrices stored in the Keys-format.
- Also, bear in mind that use of the Keys-format tends to increase the file size for the resulting px-files to some extent (depending on the density of the matrix).

To sum it up:

- For very large matrices, use the matrix format. You can enforce this policy for your total database installation by adjusting the keyword **KeysUpperLimit** to a suitable value.
- Otherwise, if processing time matters to you, choose the Keys-format . Set a very high value for PxDataFormat, e.g. **PxDataFormat=KEYS10000**.
- If file size matters to you, choose the regular (matrix) px-format (**PxDataFormat=MATRIX**).

- If neither of the two is important to you, let the extraction program decide, set e.g. **PxDataFormat=KEYS50**, which will constitute a practical compromise between processing time and file size.

3 Additions to PC-Axis files

For the NPM-system to work, you must have installed the 2005-version of the PC-Axis program, or a later version. In addition, you will have to make a few additions to (each of) your pcasqlxx.txt file(s) (one file for each language that you implement in your statistical database).

3.1 Additions to pcasqlxx.txt

3.1.1 A new section: `[SpecialCharacter]`

The following section should be included in the pcasqlxx.txt file(s) of your system right after the section [VSGroupLang2]:

```
[SpecialCharacter]
SpecialCharacter=SPECIALCHARACTER
CharacterType=CHARACTERTYPE
PresCharacter=PRESCHARACTER
AggregPossible=AGGREGPOSSIBLE
DataCellFilled=DATACELLFILLED
PresText=PRESTEXT
DataCellPres=DATACELLPRES
```

The [SpecialCharacter]-section is structured in the same way as most other database table sections in pcasqlxx.txt. The purpose of the section is to identify the name of the SpecialCharacter table and the names of important columns in this table.

If your statistical database is a multilingual one, the section shown above should pertain to the main language of your database. For the other languages used you should include short sections of this type (e.g. for English as the second language):

```
[SpecialCharacterLang2]
SpecialCharacterLang2=SPECIALCHARACTER_ENG
```

For database tables for secondary languages it is sufficient to identify the name of the tables.

4 How PC-Axis performs the data extraction

Below you will find an overview to give the end user an understanding of the basic processing steps of the extraction of table data from the sql-database.

1) After the metadata has been read by PC-Axis and the px-file has been initialized with the bulk of the metadata, PC-Axis calls OdbcCells.dll. OdbcCells is responsible for reading the table data from the sql-database.
2) OdbcCells reads the pxs-file for the extraction.
3) OdbcCells reads a number of keys from the pcasqlxx.txt that is relevant for the extraction.
4) OdbcCells reads some metadata from the sql-database.
5) OdbcCells establishes if the source maintable uses NPM-characters or not.
6) Database tables are created for the code lists, and the code lists are stored in these tables.
7) The structure for the output data matrix is determined and a corresponding database table is created.
8) OdbcCells reads the specified table data from the sql-database and stores the data read in the data table for the output data.
9) OdbcCells counts the number of records read and compares the number to the number of records in the total output matrix. These computations may in some circumstances decide if the output data is stored in Keys-format or in the regular matrix format.
10) OdbcCells examines the table data read to see if there are empty (blank) data cells in any of the records.
11) If there are empty data cells and the corresponding NPM-cells are empty, these NPM-cells are filled with the NPM-character defined in DataNotAvailable (an NPM-character belonging to category III, more about categories in the next chapter).
12) If the output data shall be stored in the matrix format, the output data matrix is expanded by inserting the cartesian product of data cells defined by (the data tables for) the code lists. The data cells of these new records are filled with the figure 0, while the corresponding NPM-cells are filled with either NULL (if PresCellsZero = Y) or with the NPM-character defined in PresMissingLine or DefaultCodeMissingLine.
13) An SQL-statement is created for aggregating the desired data groups from the records in the output data table. The code for the aggregation includes a number of calculations for each group (more about these calculations in the next chapter). The purpose of these calculations is to provide information for applying the NPM summation rules (next chapter).
14) The SQL-statement is executed and the aggregated data + a few metadata keys are written to the px-file.
15) Temporary data tables are removed.

5 Categories of NPM-characters and rules for aggregation

This chapter describes the principles that govern the work of the NPM-system.

5.1 Classification of NPM-characters into categories

Based on the contents of the columns **AggregPossible** and **DataCellPres** in the table **SpecialCharacter** the NPM-characters can be subdivided into 4 categories. The four categories are:

		AggregPossible	
		Yes	**No**
DataCellPres	**Yes**	**I Explanation to a value (marker)** The data cell contains a value. The presentation of the data cell should contain both the value and the marker Example: "Preliminary figure"	Selfcontradictory category! Presenting a value in the data cell is futile unless the value can participate in an aggregation.
	No	**II Null values** The data cell contains a nil value. The presented data cell should only contain the special character. Example: "Category not applicable"	**III Substitution for a value** The data cell has no value. The presented data cell should only contain the special character. Examples: "Data not available", "Not for publication"

As the table shows only three of the four theoretically possible categories have a place in the NPM-system. These three categories have been numbered from I to III (occasionally written 1 to 3).

The table below shows the subdivision into categories applied to the example of NPM-characters shown in chapter 2:

Character-Type	**Pres-Character**	**Aggreg-Possible**	**DataCell-Pres**	**Data Cell-Filled**	**PresText**	**Category**	**User Id**	**Log-Date**
01	.	Y	N	<0>	Category not applicable	II		
02	..	N	N	N	Data not available	III		
03	…	N	N	N	Data not yet available	III		
04	:	N	N	N	Not for publication	III		
05	-	Y	N	<0>	Nil	II		
15	e	Y	Y	V	Estimated value	I		
20	*	Y	Y	V	Provisional or preliminary figure	I		

5.2 Rules for aggregation

The category that an NPM-character belongs to determines how the NPM-character is treated in aggregations and what the result of an aggregation will be. The following table gives an overview of the possible situations that may arise when records with NPM-characters are part of a summation.

The table is called a *rule matrix*, and the cells in the matrix *rule cells*. In some of the rule cells there is a reference to another rule cell instead of a rule, e.g. **=B3**. A reference of this type means that the **situation** described in the rule cell is the same as the situation described for the other rule cell. Consequently, the same action applies for both cells.

The rule matrix is used as follows:

1) A set of data cells (and their respective NPM-cells) is input to an aggregation.
2) Find out what categories of NPM-characters are represented among the data cells in the set.
3) Find out which is the **highest** NPM-category (3 is higher than 2, and so on) that enters into the aggregation.
4) Use the column of the highest NPM-category as the starting point. Then find the text in the left-most column that most appropriately describes the inventory of data cells in your aggregation.
5) The rule for your aggregation is found in the rule cell that is the crossing point of the text line and the selected category column.

The other records entering into the aggregation are of the following types:		**Starting point: An aggregation contains one or more records that contains a special character belonging to one of the categories:**		
		Category I **"Marker"** Comment on an existing value: ”Preliminary figure”	**Category II** **"Nil-value"** Clarification to an existing value: ”Absolute nil”	**Category III** **"Non-printable value"** Replacement for a non-printable value: ”unknown value”
		A	**B**	**C**
0	The aggregation includes one record only	The special character **and** the value are presented	The special character is presented.	The special character is presented.
1	All records included in the aggregation have the same special char.	A regular aggregation. The special character is presented along with the sum value.	No aggregation necessary. The special character is presented.	Aggregation is not possible. Only the special character is presented.
2	No other record in the aggregation has a special character	A regular aggregation. The special character is presented along with the sum value.	A regular aggregation. Only the sum value is presented.	Aggregation is not possible. Only the special character is presented.
3	There are also records with other special characters from the same category	A regular aggregation. The special character "**DataNoteSum**" is presented along with the sum value.	A regular aggregation. Only the sum value is presented.	Aggregation is not possible. The special character "**DataSymbolSum**" is presented.
4	All records in the aggregation have special characters from the same category (but more special characters are represented)	A regular aggregation. The special character "**DataNoteSum**" is presented along with the sum value.	A regular aggregation. Only the sum value (=0) is presented.	Aggregation is not possible. The special character "**DataSymbolSum**" is presented.
5a	There are also records with a special char. from category I (only one type of special char. from cat. I)	**= A1**	A regular aggregation. The special character from category I is presented along with the sum value.	Aggregation is not possible. The special character "**DataSymbolSum**" is presented.
5b	There are also records with a spe-	**= A3**	A regular aggregation. The special character	Aggregation is not possible.

The other records entering into the aggregation are of the following types:		Starting point: An aggregation contains one or more records that contains a special character belonging to one of the categories:		
		Category I	**Category II**	**Category III**
		"Marker" Comment on an existing value: ”Preliminary figure”	**"Nil-value"** Clarification to an existing value: ”Absolute nil”	**"Non-printable value"** Replacement for a non-printable value: ”unknown value”
		A	**B**	**C**
	cial character from cat. I (more types of special characters from cat. I)		"**DataNoteSum**" is presented along with the sum value.	The special character "**DataSymbolSum**" is presented.
6	There are also records with a special character from cat. II	**= B5a + B5b**	**= B3**	Aggregation is not possible. Only the special character is presented.
7	There are also records with a special character from cat. III	**= C5a + C5b**	**= C6**	**= C3**

Notes to the contents of the table:

- Cell C6 in an earlier version had the same rule as cells C5a and C5b. For technical reasons (as explained in chapter 4, during reading from the sql-database the output matrix is expanded by adding empty data cells for the whole matrix, and these empty data cells may have NPM-characters of category II added with them) the rule had to be changed to its present form to ensure that processing with PresCellsZero=Y would give the same result as processing with PresCellsZero=N (i.e. so that C1 and C2 (PresCellsZero=Y) is represented in the same way as C6 (PresCellsZero=N)). (Actually one also has to take into account the type of special character defined in PresMissingLine og DefaultCodeMissingLine. Here the two are assumed to belong to category II.)
- It can be argued that this change is altogether not unreasonable. Aggregations will most often involve only one type of NPM-characters belonging to category III, which then is the category “responsible” for making the result of the aggregation non-presentable, and therefore deserves to be presented as the “culprit”, as the cell rule now does. Only marginally aggregations will include more types of NPM-characters belonging to category III. In these cases rule C6 is not wrong, but also not quite accurate, ascribing the reason for the non-presentability to only one of the NPM-characters (the “largest” one, when sorted alphabetically, will be chosen in this case).
- The “culprit” argument can be advanced with regard to rules C5a and C5b as well. Substituting the present form of rule C6 for these two cells as well will make the result of aggregations more transparent. This line of reasoning ultimately leads to reserving the special character **DataSymbolSum** to the cases where more than one type of special characters belonging to category is involved in the aggregation. The problem is that this is a situation that the current algorithm (see below) cannot record in an exhaustive way (except for C3 and C4, which are typical examples).

5.3 The algorithm for aggregation

Based on the rules of the rule matrix it is possible to formulate an algorithm for the aggregations. The algorithm must be able to

- decide which aggregation situation (rule cell) a given summation belongs to, and

- (to be efficient) at the same time perform the summation.

A ”pseudo-version” of the algorithm may look like this:

```
If only inserted records are involved in the aggregation group, then
   If a special sign is defined for inserted records, then
      Use sign for inserted record
   End if

ElseIf NPM characters are not used, then
   If records with empty cell values are part of the aggregation, then
      Use sign for data not available
   ElseIf the aggregation includes inserted records with no "mates", then
      If a special sign for inserted records is defined, then
         Use sign for inserted record
      End if
   End if

Else -- recs. from the database included and NPM characters used
   If there are special characters involved in the aggregation, then
      If the aggregation involves only one record, then
         If the special character belongs to category I, then
            A0
         Else
            B0-C0
         End if
      Else  -- more than one record included in the aggregation
         If the aggregation involves a special character of category III, then
            If the number of distinct NPM-characters involved equals 1,
            Or the aggregation involves a special char. of category II, then
               C1-C2, C6
            Else
               C3-C5
            End if
         Elseif the aggregation involves a special char. from category II, then
            If all NPM-characters in the aggregation belongs to category II, then
               If there are inserted records with no "mates"
               And the special sign for missing records belongs to cat. 3, then
                  C6
               ElseIf the number of distinct NPM-characters > 1, then
                  B3-B4
               Elseif (NoOf dist. NPM = 1)
               And ((NoOf records = NoOf NPM-characters)
               Or (sign for inserted rec = highest ranked NPM in aggr. group)) then
                  B1
               Else
                  B2
               End if
            Else  -- There are one or more cells with NPM-chars of cat. I as well
               If there are inserted records with no "mates"
               And the special sign for missing records belongs to cat. 3, then
                  C5a-C5b
               Else
                  If the number of distinct NPM-characters = 2, then
                     B5a
                  Else
                     B5b
                  End if
               End if
            End if
         Else – Only NPM-characters of category I is involved
            If there are inserted records with no "mates"
            And the special sign for missing records belongs to cat. 3, then
               C5a-C5b
            Else
               If the number of distinct NPM-characters > 1 then
                  A3-A4
```

```
                Else
                   A1-A2
                End if
             End if
          End if
       End if
   ElseIf there are inserted records with no "mates"
   And the special sign for missing records belongs to cat. 1 or higher, then
       Present the defined sign for missing record
   End if
End if
```

To be able execute this algorithm efficiently the data program must have access to the information that we may conceive of as stored in the following variables:

Variable name	Variable contents
NumberOfRecords	The number of records that enters into the aggregation group
NumberOfNPMSigns	The number of records in the aggregation group that has got NPM-characters
NumberOfDistinctNPMSigns	The number of distinct NPM-characters participating in the aggregation group
HighestCategoryNPMSign	The highest category of NPM-characters in the aggregation group
LowestCategoryNPMSign	The lowest category of NPM-characters in the aggregation group
NumberOfRecsFromDatabase	The number of records read from the sql-database in the aggregation group

This information can be provided by expanding the final SELECT statement with 6 group functions (COUNT, MAX, MIN, SUM):

```
SELECT COUNT(*)                        AS NumberOfRecords,
       SUM(Contents)                   AS Value,
       COUNT(Contents_x)               AS NumberOfNPMSigns,
       COUNT(DISTINCT Contents_x)      AS NumberOfDistinctNPMSigns,
       MAX(Contents_x)                 AS HighestCategoryNPMSign,
       MIN(Contents_x)                 AS LowestCategoryNPMSign,
       SUM(SqlMarkColumn)              AS NumberOfRecsFromDatabase
FROM   DataTable
GROUP BY ... ...
```

In addition to the result of the GROUP functions, other “metadata”, collected during the execution of the extraction program, is available at the time of saving the extracted data in a file:

- **NPMColumnsUsed** – For each maintable the property SpecCharExists (Y or N) shows if the NPM-system is used for the main table or not. NPMColumnsUsed is the internal representation of SpecCharExists in this program.
- **PresCellsZero** + **PresMissingLine + DefaultCodeMissingLine** – For each content variable in the sql-database a special character for presenting the data cell value of missing lines (records not stored in the database, but inserted during extraction) may be defined. If no special character is defined, a 0 (zero) should be presented. This information is derived for each content from the values of **PresCellsZero, PresMissingLine**, and **DefaultCodeMissingLine** and stored in a variable called PresMissingLine in the code listings below.
- **NumberOfBlankDataCells** – Each content variable has an indicator that shows how many blank (empty) data cells were read from the sql-database.

These additional metadata make it possible to speed up the final processing by treating the most frequent cases without entering the more “heavy” parts of the algorithm. The pseudo-algorithm above can now be expanded and rewritten as:

```
If NumberOfRecsFromDatabase = 0 Then –- No need to check for blank data cells!
   /* Inserted records only. Only possible when storing in Matrix format. */
   If Len(PresMissingLine) > 0 Then  -- Special char. for missing line defined
      Present NPM-character for missing line (PresMissingLine)
   Else
      Present 0  -- No special character for missing line defined. Present a zero
   End If
ElseIf NPMColumnsUsed = "N" Then  -- NPM-characters are not used for this matrix
   /* Database data exist, and so we will ignore possible PresMissingLines.
      Since NPM is not used, we only need to look for blank data cells!
   */
   EmptyCellFound = FALSE  -- Need a flag
   If NumberOfBlankDataCells > 0 Then  -- NB! Must check for possible NPM-chars!
      If NumberOfDistinctNPMSigns > 0 Then  -- Only possible is DataNotAvailable
         Present character for DataNotAvailable
         EmptyCellFound = TRUE
      End If
   End If

   If NOT EmptyCellFound Then
      If (NumberOfRecordsFromDatabase * 2) < NumberOfRecords Then  -- Inserted recs
      /* The sum group contains inserted records with no "mate" from the database
         If PresCellZero=N and PresMissingLine of cat. 3, then present spec. char.
      */
         If Left(PresMissingLine, 1) = "3" Then
            Present NPM-character for missing line (PresMissingLine)
         End If
      End If
   End If
Else  -- NumberOfRecsFromDatabase > 0 And NPMColumnsUsed = "Y"
   /* No short cut possible, we have to enter the main rule system */
   If NumberOfDistinctNPMSigns > 0 Then  -- One or more NPM-characters used
      If NumberOfRecords = 1 Then  -- Only 1 record in the aggregation group
         If HighestCategoryNPMSign = 1 Then  -- Must be a marker!
            A0
         Else  -- Zero value or a unavailable value
            B0-C0
         End If
      Else  -- (NumberOfDistinctNPMSigns > 0) And (NumberOfRecords > 1)
         If HighestCategoryNPMSign = 3 then  -- Unavailable value...
            If NumberOfDistinctNPMSigns = 1  -- ...but only one of them
            Or LowestCategoryNPMSign = 2 Then -- ...and there is also NPM of cat. 2
               C1-C2, C6
            Else -- (NumberOfDistinctNPMSigns > 1) And (LowestCategoryNPMSign <> 2)
               C3-C5
            End If
         ElseIf HighestCategoryNPMSign = 2 then  -- Zero value
            If LowestCategoryNPMSign = 2 then   -- and the same goes for No 2
               If there are inserted records with no "mate"
               And PresMissingLine belongs to cat. 3 Then -- unlikely, but poss.
                  C6
               ElseIf (NumberOfDistinctNPMSigns > 1) then -- more NPM-chars used
                  B3-B4
               ElseIf (NumberOfDistinctNPMSigns = 1)
               And ((NumberOfRecords = NumberOfNPMSigns)
               Or (PresMissingLine = HighestCategoryNPMSign)) Then
                  B1
               Else
                  B2
               End If
            Else – There are cells with NPM characters from category 1 as well
               If there are inserted records with no "mate"
```

```
               And PresMissingLine belongs to cat. 3 Then -- unlikely, but poss.
                  C5a-C5b
               Else
                  If NumberOfDistinctNPMSigns = 2 then –- only one from to cat. 2
                     B5a
                  Else –- more than 2 distinct NPM chars – only one from cat. 2
                     B5b
                  End If
               End If
            End If
         Else  -- Only special characters of category I are included
            If there are inserted records with no "mate"
            And PresMissingLine belongs to cat. 3 Then -- not unlikely here
               C5a-C5b
            Else
               If NumberOfDistinctNPMSigns > 1 Then
                  A3-A4
               Else
                  A1-A2
               End If
            End If
         End If
      End If
   ElseIf there are inserted records with no "mate"
   And PresMissingLine belongs to cat. 1 or higher Then
      Present character for missing record
   End If
End If
```