Variant Tables

Table of Contents

- 1. Overview
- 2. Example Model
- 3. Variant Table Structure (CU61, CU62, CU63)
- 4. Maintaining Variant Table Entries (CU60, CU60E)
  - 4.1 Table Maintenance CU60
  - 4.2 Excel-Import CU60E
- 5. Linking Variant Tables to Database Tables
- 6. Variant Tables in Dependencies

1. Overview

Variant tables are used to store combinations of characteristic values. They can be referenced in dependencies to infer, check and restrict values. More specifically, they are used in:

- selection conditions, preconditions, constraints and procedures to check the consistency of the values entered,
- procedures and constraints to infer values,
- constraints to restrict values for a characteristic.

One advantage of using them is that you only need to change the variant table, if the interdependencies between characteristics change, and not the dependencies. This greatly simplifies the maintenance tasks when a model changes.

The relevant transactions for variant tables are the following:

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CU61</td>
<td>Create variant table</td>
</tr>
<tr>
<td>CU62</td>
<td>Change variant table</td>
</tr>
<tr>
<td>CU63</td>
<td>Display variant table</td>
</tr>
<tr>
<td>CU64</td>
<td>Variant table structure list</td>
</tr>
<tr>
<td>CU60</td>
<td>Variant table maintenance</td>
</tr>
<tr>
<td>CU59</td>
<td>Transfer variant table contents to database table (Report RCCUVTDB)</td>
</tr>
<tr>
<td>CU60E</td>
<td>Excel upload for variant table contents</td>
</tr>
</tbody>
</table>

Remark: In order to delete a variant table, you need to first remove all the lines in the CU60 and then the table itself is deleted in the CU62. However, once a variant table was maintained with a change number, the table can no longer be deleted.

2. Example Model

In order to discuss the various features of variant tables in a concrete manner, we will use a very simple example model, which describes the properties of a Personal Computer (PC). It consists of 4 single-valued characteristics with the following values:

- CH_PCTYPE (OFFICE, MEDIA, GAMING)
- CH_PROCESSOR (3GHz, 4GHz, 5GHz)
- CH_MEMORY (8GB, 16GB, 32GB)
- CH_HARDDISK (1TB, 2TB, 3TB)

Using a variant table we will build the following combinations of characteristic values:
As you can see, each column in a variant table represents a characteristic and every row corresponds to an allowed combination of characteristic values. In this particular example, an Office-PC has a 3 Gigahertz Processor, 8 Gigabytes of Memory and a 1 Terabyte Harddisk. Similarly, the properties of a Media- and Gaming-PC are displayed in the corresponding rows of this table.

3. Variant Table Structure (CU61, CU62, CU63)

In order to create a variant table you start with the transaction CU61, where you first need to enter a name for the table. We will call our table VT_PC.

<table>
<thead>
<tr>
<th>CH_PCTYPE</th>
<th>CH_PROCESSOR</th>
<th>CH_MEMORY</th>
<th>CH_HARDDISK</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFICE</td>
<td>3 GHz</td>
<td>8 GB</td>
<td>1 TB</td>
</tr>
<tr>
<td>MEDIA</td>
<td>4 GHz</td>
<td>16 GB</td>
<td>2 TB</td>
</tr>
<tr>
<td>GAMING</td>
<td>5 GHz</td>
<td>32 GB</td>
<td>3 TB</td>
</tr>
</tbody>
</table>

In the field Date the current date is entered by default, so you don’t need to worry about that. In the next screen you need to enter a description for the variant table. Note that the Status is currently 0(In Preparation) - this needs to be changed later on.
As you can see, in this screen there exists also the possibility to link the variant table to a database table. This functionality will be discussed in section 5.

After pressing on Characteristics you enter the following screen:
Here you can enter the characteristics that you want to use in the variant table. We set the characteristic CH_PCTYPE as a Key Field. This means that each entry in the column CH_PCTYPE must be unique. By using dependencies we will use this characteristics later on to infer values for CH_PROCESSOR, CH_MEMORY and CH_HARDDISK in section 6.

Remark: Marking a characteristic as a Key field is only relevant if you want to infer values using dependencies.

After pressing Back(F3) you return to the previous screen.
There you need to change the Status to 1(Released). Finally, confirm everything by pressing Save(Ctrl+S).

In the transactions CU62 the variant table structure can be changed and in the CU63 it can be displayed.

4. Maintaining Variant Table Entries

4.1 Table Maintenance CU60

So far we have only specified which characteristics are to be used in the variant table. In our case these are the characteristics CH_PCTYPE, CH_PROCESSOR, CH_MEMORY and CH_HARDDISK. However, we have not specified any values so far. This is done in the transaction CU60.

Upon entering this transaction you see the following screen, where you need to enter the name of the variant table.
After pressing Maintain(F6) you see the following screen:

Table Maintenance

Table  VT_PC

Effectivity

Change number

Valid from  25.03.2013
Here we have already entered the characteristic values for our model according to the table from section 2.

By using the Check (Shift+F4) functionality, you can make sure that your variant table complies with the standard (Key Field entries are unique, no empty entries etc.). For the table maintenance the properties of the used characteristics (single/multi-valued, restrictable, entry required) are irrelevant.

Remark: In order to enter multiple values in a multi-valued characteristic you need to double-click on a field in the table maintenance CU60. In the pop-up that appears, you can select the characteristic values using radio-buttons. Note that this representation is not IPC-compatible. However, there exists the possibility to normalize a variant table with multiple values by navigating to Table Untag. As a result, a row of the form

```
01 AA, BB
```

is converted to

```
01 AA
01 BB
```

In this representation the variant table is IPC-compatible again. Also note that this normalization should be used cautiously in conjunction with Key Fields, as this might result in errors during the variant table Check (Shift + F4).

Alternatively to the Standard View of the CU60, there exists the possibility to display the allowed value combinations in a matrix form. By selecting Table Decision table Presentation: Matrix, the following screens appears:
Each column in this matrix corresponds to an a priori possible value combination. Since our variant table contains 4 characteristics, with 3 values each, there 3·3·3·3=81 columns. The variant table restricts this list to 3 combinations - these are the green columns in the table. In this matrix representation these 3 value combinations are marked with a red X. This way one can view and maintain allowed value combinations very quickly.

In addition this matrix form, there is the possibility to select Presentation: Number of combinations under Table as a presentation style. This provides a list view of all possible combinations - the allowed combinations are marked with a + sign.

### 4.2 Excel-Import (CU60E)

Apart from the standard transaction CU60 there also exists possibility to fill a variant table with values using an Excel-spreadsheet. This is done with the transaction CU60E.

In order to do that, it is necessary that there is already a variant table structure in your system, which contains all the characteristics you want to use. Moreover, the Excel-file that you use to import the variant table content must of a certain form:

- The columns must have the same order as in the variant table.
- Use no header lines. The Excel table must only contain the characteristic values.
- All characteristic values must have text format. (This also holds for numerical values.)
- The characteristic values only the language-independent format must be used.
- The Excel-file must have .csv format.

If we were to fill the variant table for our example with the CU60E instead of the CU60 from section 4.1, the corresponding Excel-file would look as follows:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OFFICE</td>
<td>3 GHz</td>
<td>8 GB</td>
<td>1 TB</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>MEDIA</td>
<td>4 GHz</td>
<td>16 GB</td>
<td>2 TB</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>GAMING</td>
<td>5 GHz</td>
<td>32 GB</td>
<td>3 TB</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After you have prepared your Excel-file, you need to enter the name of the variant table and the location of the file in the transaction CU60E:
After pressing *Execute* (F8) the content of the Excel-file is transferred to the variant table.

**Remark:** Note that the CU60E does not support a change-import - when an Excel-file is uploaded, all entries in the variant table are deleted and replaced by the values from the Excel-file. In particular, a change-import using change numbers is not supported in the CU60E.

A more detailed documentation for the CU60E can be found in the attachment of note 516885.

### 5. Linking Variant Tables to Database Tables

There are certain situations where it is useful to link a variant table to a database (DB) table. For example, using very large variant tables in dependencies can result in long response times. In this case, a link to a DB table can significantly improve the performance, since the DB table is called in dependency knowledge, instead of the variant table.

**Remark:** More information on performance best practices for variant tables can be found in note 917987.

When linking a variant table to a DB table, there are two scenarios which must be distinguished:

**I. An existing DB table is linked to a variant table**

In this scenario you start with a DB table (e.g. the table MARA from the material master). By linking it to a variant table, the content of the DB table can be used in dependencies via the variant table without making use of variant functions.

In order to do that, you need to first create the characteristics which are supposed to be the columns of the variant table later on. Note that you don't need to rebuild all of the columns of the DB table using characteristics. Creating the characteristics that you eventually use is sufficient. Next you create the variant table structure in the CU61 with the characteristics that you have created. After entering the DB table name in the field `Database Table` of the CU61, the link between the DB table and the variant table is active.

**II. An existing variant table is linked to a DB table**

In this scenario you start with an existing variant table, whose content will be transferred to a DB table. For performance and maintenance reasons, this can be useful for very large variant tables (>100,000 cells).

In order to do that, you need to create a DB table, whose column fields have a format which is "suitable" for the characteristics of the variant table - for example, if the characteristic is a five-digit numerical value, then the DB field must be NUM(5) - but NUM(6) or larger would also be fine. You just have to make sure that the DB field is large enough, so that parts of the characteristic values are not cut off. Once the DB table was created in the SE11, you switch to the CU62 were you enter the DB table name in the Basis Data of the variant table. After saving, these two objects are linked. In order to actually transfer the data from the variant table to the DB table, you need to execute to the transaction CU59, where you just enter the name of the variant table.

Now, whenever the variant table is called in dependency knowledge, the DB table is called instead - you don’t need to make any adjustments in the dependencies for this to work. Note that all the maintenance tasks must now be done in the DB table - changes to the variant table content in the CU60 do not have any effect.

### 6. Variant Tables in Dependencies

Variant tables can be used in all types of dependencies (selection conditions, preconditions, procedures and constraints). The structure of a table call is always the same:
It is initialized with the keyword `table`, followed by the name of the variant table. In parenthesis you start a column-name and set it equal to some characteristic in your dependency - the various column-characteristic pairs are separated by commas. Note that you do not have to use all the columns from the variant table. As an example, let's consider our example model for a PC. The following call:

```
table VT_PC
    (  CH_PCTYPE = $self.CH_PCTYPE,
        CH_PROCESSOR = $self.CH_PROCESSOR,
        CH_MEMORY = $self.CH_MEMORY,
        CH_HARDDISK = $self.CH_HARDDISK
    )
```

can be used in a procedure. Once the characteristic `CH_PCTYPE` is entered in the configurator, the other characteristics `CH_PROCESSOR`, `CH_MEMORY` and `CH_HARDDISK` are inferred, according to the table from section 2. This works because `CH_PCTYPE` is a Key-Field in the variant table.

Note that in procedures, the characteristics in the dependency must have the `$self`, `$root` or `$parent` prefix. Characteristics which are inferred must have the `$self` prefix.