Paper 086-2012

Have Your Web Reports Remember the Filters of Your Users

Frank Poppe and Andres van Antwerpen, PW Consulting, Culemborg, the Netherlands

ABSTRACT

SAS® Web Report Studio offers several ways of filtering your data. However, often users would like to define a filter only once, and have this filter applied automatically when opening different web reports.

This paper describes a way of accomplishing just that: the user makes a selection in SAS Information Delivery Portal, which then is stored and applied automatically to an information map using the SAS Stored Process Server. In this way, each web report that uses this information map will automatically show source data filtered for the user's selection.

INTRODUCTION

The SAS Information Delivery Portal is an excellent means of delivering management information to a wide range of executives – from the CEO at the central office to the department store manager.

However, not every member of your staff needs access to all information; usually, a brand manager will be most interested in the data concerning his own brand, while the CEO may need a report comparing statistics of several divisions.

This paper describes a method of keeping all information in one source table, while delivering specific segments exactly to those people who need them. More importantly, a user will only need to define her preference once, and this preference will automatically be remembered in each SAS Web Report she opens.

HOW IS IT BEING USED?

We exploit a stored process to allow the user to specify his preferences. This stored process presents an html form to the user. If there are preferences know for this user from a previous occasion these will be shown. Using the selection options within the form the user can select other preferences, and on clicking the okay button the stored process calls itself. This will then show the same form with the adjusted values. The technique to have a stored process call itself is based on the method described by (a.o.) Mason [2011].

The stored process is surfaced to the user on a page in the portal, in an URL navigator portlet.

Also in the portal pages are links to various web reports that are based on relational information maps (the technique described in this paper applies to relational information maps only, no to information maps based on OLAP data). These information maps are set up in such a way that the preferences specified will be picked up.

The next paragraphs first describe the stored process to specify the preferences, and subsequently the steps will be described to have them surface in the query generated by the information map.

SPECIFYING AND SAVING PREFERENCES THROUGH THE STORED PROCESS

The URL within the navigator portlet uses the `_stprun` directive. This makes the user credentials, known and authenticated within the portal session, available in the stored process as SAS macro variables (this directive is described in Usage Note 37516, but not documented in the "*Stored Processes Developer's Guide*"). The user account, available in the form `username@domain`, uniquely identifies the user. The username is used as the name for a SAS data set and the data set is stored in a persistent SAS library.

We have chosen to use a separate table for each user, instead of using one table with a column that identifies the records for a specific user. This way we avoid the possibility of locking problems when several users try to write to the table at the same time.

Each time the stored process runs, it first checks whether a table with the name of the user already exists. If so, the preferences are read from the table and presented in the html form. If this appears to be the first time this specific users uses the system the HTML form is presented with only possible choices.

When the choices are being committed the stored process runs a second time and the preferences are written to the data set and again presented as an html form.

USING A STORED PROCESS IN AN INFORMATION MAP

Users are allowed to access data only through information maps; that is, they are not allowed to access tables directly. We will adapt the information maps, in order to have them apply the stored preferences to the data.

This is done by including the table with preferences in the relational structure, and setting up the relations such that only records that satisfy the preferences will be returned (the SQL query may get a bit complicated, more on this later on).

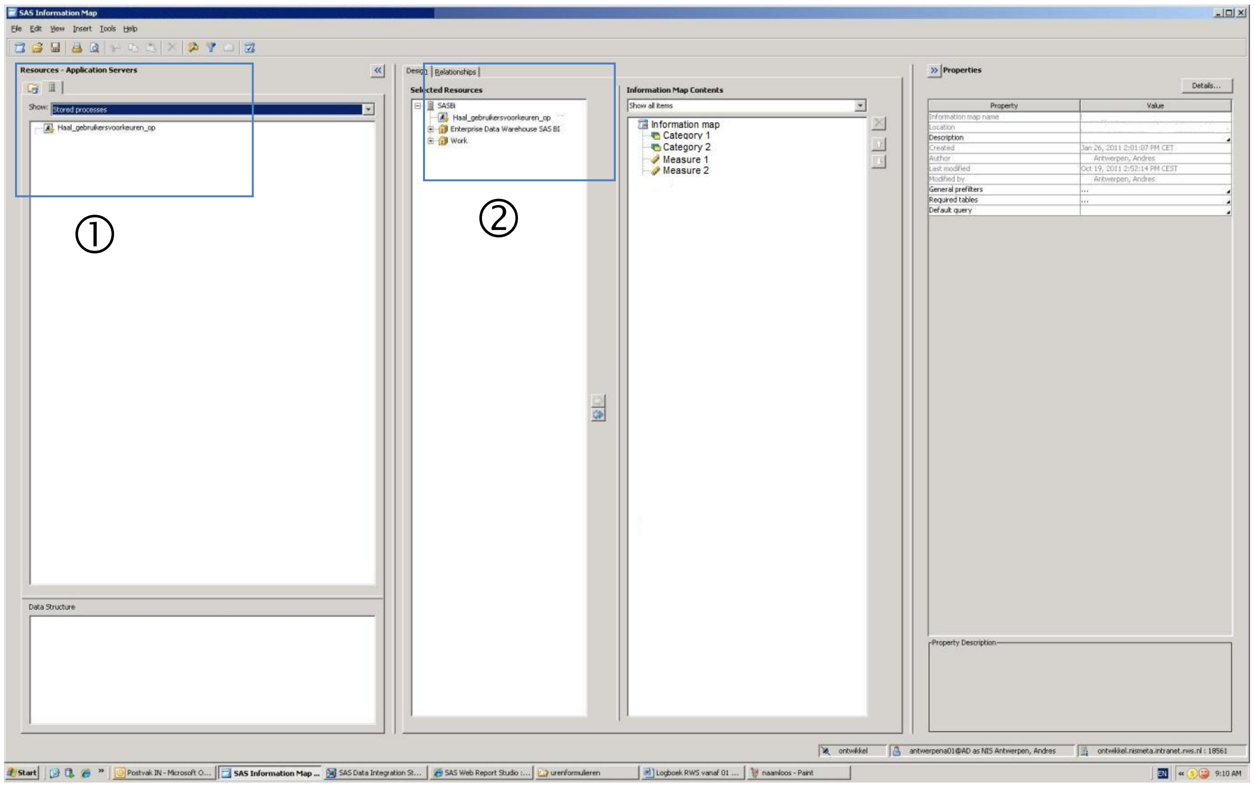
Some steps are needed to accomplish this.

As we have seen in the previous paragraph there is a separate tables for each user, but in the information map we can obviously use only a single table.

To accomplish this we define a table in the metadata with an arbitrary name (e.g. *preferences*) with as libref *work*, and this table is being joined to the 'real' data. As each user will have his own work space server session, and thus a separate work library, this means that this generic metadata object is a different physical table for each user.

But this physical table in the work library still has to be created: the content of the preferences table for this user have to be copied to the temporary table. Here again a stored process is being used. A lesser known feature of information maps is that you can include a stored process in an information map, and this stored process is run before the query is being executed.

Figure 1 shows the design pane of an information map, with on the left in the resources pane the stored process list selected. The parts denoted with ① and ② have been enlarged below for readability.



Property	Value
Information map name	
Location	
Description	
Created	Jan 26, 2011 2:01:07 PM CET
Author	Antwerpen, Andres
Last modified	Oct 19, 2011 2:52:14 PM CEST
Modified by	Antwerpen, Andres
General prefilters	...
Required tables	...
Default query	

Figure 1. Information map design

①

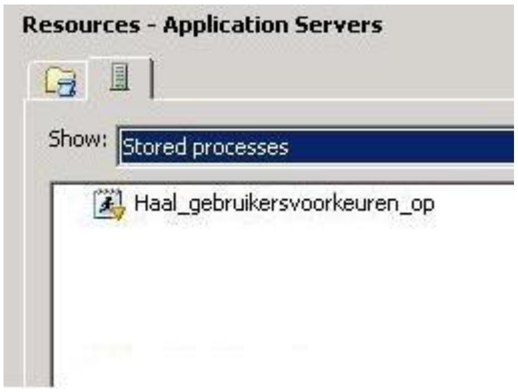


Figure 2. Enlargement 1 from figure 1

②

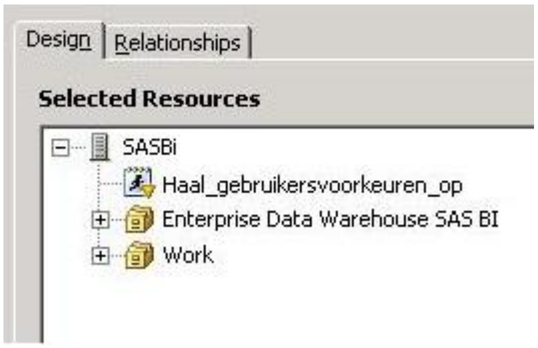
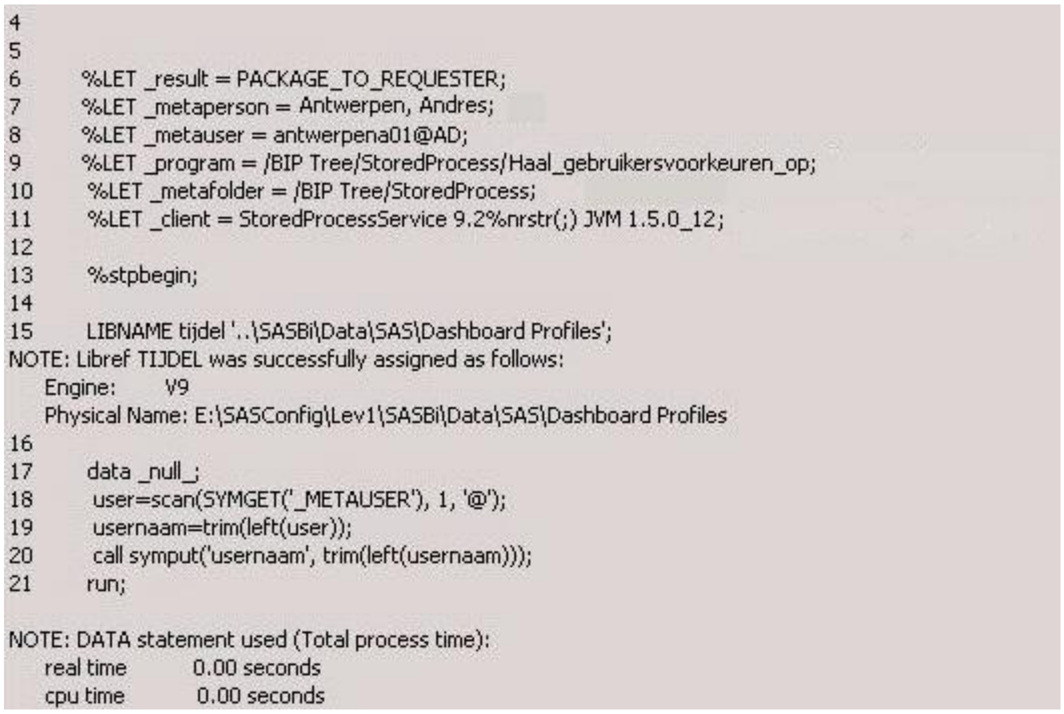

Design
Relationships
Selected Resources
SASBi
Haal_gebruikersvoorkeuren_op
Enterprise Data Warehouse SAS BI
Work

Figure 3. Enlargement 2 from figure 1

To have this stored process know for which user the preferences have to be copied, we use the `%processbody` directive. Similarly to a stored process called from the Information Delivery Portal this will make the user credentials available as macro variables.

The stored process is added to the information map and the first thing that will happen when the information map is being accessed is to run the stored process. Using the macro variable containing the account name (in the form `user@domain`) the table for this user is copied from the persistent SAS library to the temporary data set in the work library.

In this way each time the Information map is accessed, the temporary table is filled with the preferences of the current user. Figure 3 shows part of the log from running a test query in Information Map Studio.



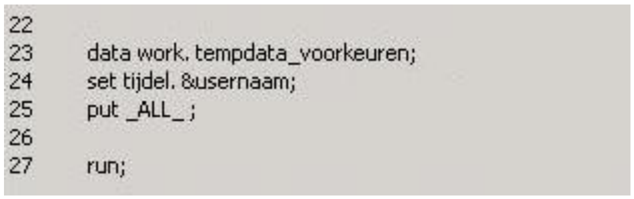
```
4
5
6      %LET _result = PACKAGE_TO_REQUESTER;
7      %LET _metaperson = Antwerpen, Andres;
8      %LET _metauser = antwerpena01@AD;
9      %LET _program = /BIP Tree/StoredProcess/Haal_gebruikersvoorkeuren_op;
10      %LET _metafolder = /BIP Tree/StoredProcess;
11      %LET _client = StoredProcessService 9.2%nrstr(;) JVM 1.5.0_12;
12
13      %stpbegin;
14
15      LIBNAME tijdel '..\SASBi\Data\SAS\Dashboard Profiles';
NOTE: Libref TIJDEL was successfully assigned as follows:
    Engine:       V9
    Physical Name: E:\SASConfig\Lev1\SASBi\Data\SAS\Dashboard Profiles
16
17      data _null_;
18       user=scan(SYMGET('_METAUSER'), 1, '@');
19       usernaam=trim(left(user));
20       call symput('usernaam', trim(left(usernaam)));
21      run;

NOTE: DATA statement used (Total process time):
    real time          0.00 seconds
    cpu time           0.00 seconds
```

Figure 4. Log from testing the query (part 1)

The `%let` statements at the start of the log have been generated automatically as a result of the `%processbody` directive. There the information to identify the authenticated user and the stored process that is actually running is stored into SAS macro variables.

The data step extracts the username from `&_metauser` (using only the part before the '@').



```

data work. tempdata_voorkeuren;
set tijdel. &usernaam;
put _ALL_ ;

run;
```

Figure 5. Log from testing the query (part 2)

The next part of the log shows the final part of the preparation: the DATA step copies the data set for this particular user to the data set that has been registered in the metadata (which here has been given the name `tempdata_voorkeuren`; 'voorkeuren' being Dutch for 'preferences'). So now the data set that is being used in the query actually has been created and filled with the relevant data.

GENERAL SYNTAX OF THE SQL QUERY

Figure 6 shows part of the same window as depicted in figure 1, but now the *Relationships* tab has been selected.

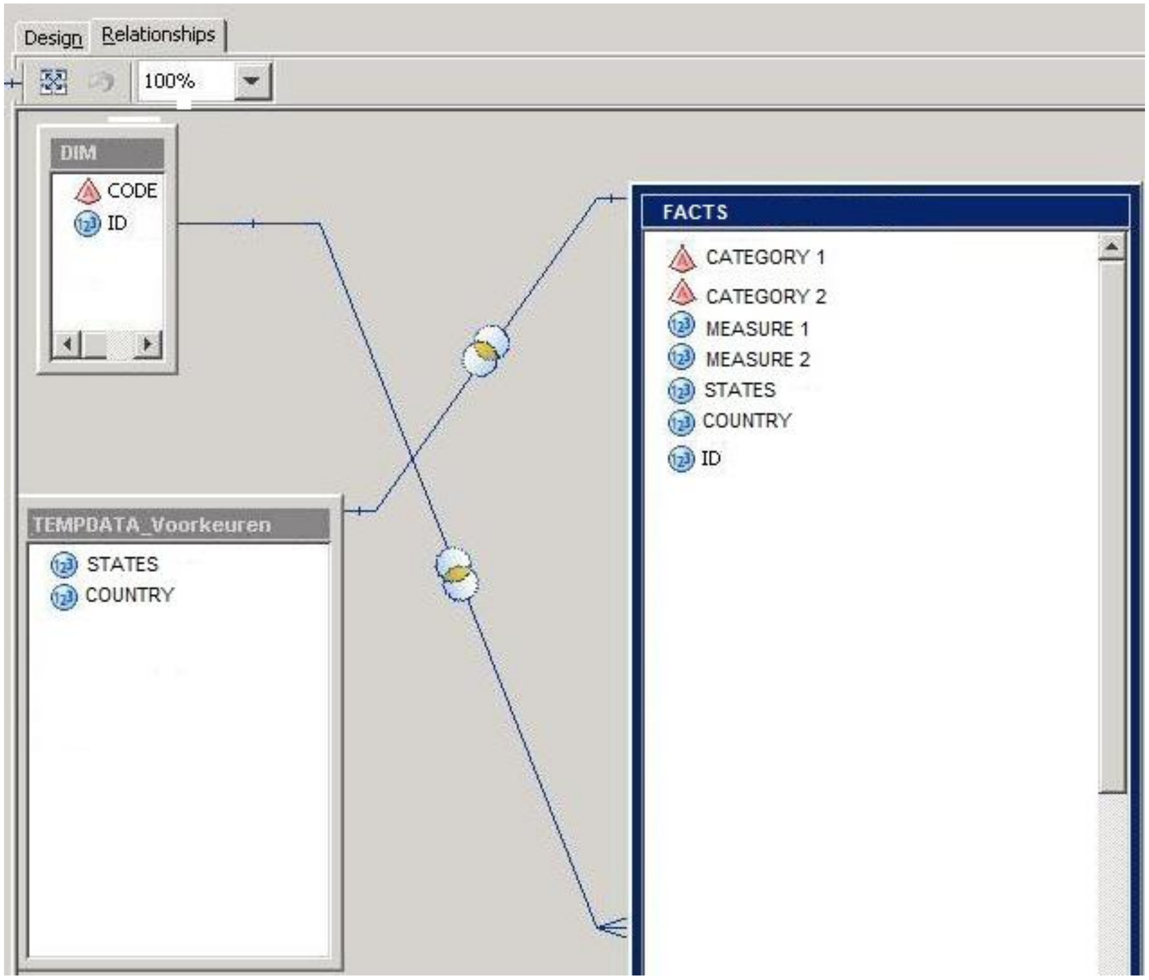


Figure 6. Information map relationships

The variables in the table with preferences, `TEMPDATA_Voorkeuren`, contain the choices of this user on those areas.

These variables are compared with the corresponding variables in the large table on the right. So suppose the variable CATEGORY1 contains the preference for country and CATEGORY2 that for state, then if the preference for country is USA without a preference for CATEGORY2 the record of records for "USA" will be returned. If the preference calls for data concerning the state "Wyoming", records for that level will be returned.

The conditions in these queries are a bit more complicated than just simple comparisons. This has to do with the fact that users can be interested in data from different geographical levels, but also in differing detail. The next paragraph explains this in more detail, and subsequently the consequences for the query are outlined.

HIERARCHICAL ORGANIZED DATA

The data can be organized in a hierarchical way (countries – state – etc., company – division – unit – etc.). The requirements for the application stated that users should be able to choose all of the following options:

- the aggregate total for a single specified large unit (e.g. the total for USA);
- the totals for all the smaller units within a specified large unit (e.g. totals for each state within the USA);
- the totals for a single specified smaller unit (e.g. the total for Wyoming).

To realize this we added the option "ALL" to the available choices. If we take the example of countries and states the first option can be specified by just specifying “USA” as the country, leaving the selection for state blank. The second option is specified by selecting the combination “USA” for country and “ALL” for state, and the third by specifying “USA” and “Wyoming”.

This then has to be translated into a working SQL query. The design tab of SAS Information map Studio allows for a custom definition of a relation, and we used this function to support this choice for an aggregate:

```
 (CASE
 (WHEN <<PREFERENCES.STATES>> = "ALL" THEN
     <<SOURCEDATA.COUNTRY>>="USA"
ELSE (<<SOURCEDATA.COUNTRY>>="USA" &
     <<SOURCEDATA.STATE>>=<<PREFERENCES.STATE>>
END
```

This can be read as follows.
In the case of an “ALL” specification for state, only filter on country, otherwise filter both on country and on state. (It might be tempting to filter only on state, but there are countries with states or provinces of the same name.)

Caution is advised here anyway, since one can easily make things too complicated to understand properly. Moreover, customers may wish to have support for a very complex set of choices.

It is tempting to try and support all these wishes with a single information map that can be used in more than one web report but it is better to stick to the rule ‘one information map for one web report’.

THE OPTIONS AVAILABLE TO EACH USER

In the previous sections we have mentioned that users define their own preferences from the options that are available for them, without going into the question whether they all have the same options or if those differ.
In the organization for which the system we describe here was developed we had to deal with a great variety of organizational divisions and responsibilities. Below the major departmental division some departments only have a single next level, while others might have five or six levels. The system also had to take into account that some managers should be able to ‘see’ several levels deep into the organization, while others should be allowed to see only one level deep.
For each user the allowed choices are stored in a table. This table is the source for the options presented in the html form from which the preferences can be selected. For new users the table can be filled with a default set of options based on the type of user. This part of the application is not described further in this paper. It has conceptually no impact on the parts described here.

BONUS: AUTOMATIC REFRESH OF OTHER PORTLETS AFTER A CHANGE OF SELECTION

The stored process that allows the user to specify a set of preferences can be extended so that the content of a portlet on the same portal page is automatically refreshed when the user changes the selection.

Chapter 3 outlined how this stored process will run two times. The second time, as it presents the new selection, a JavaScript function is added which will execute onLoad().

When another portlet contains HTML that has an inline frame, it is possible to have the JavaScript function refresh the contents of this inline frame.

CONCLUSION

Using a stored process to insert user specific preferences into an information map is a useful method to provide users with the possibility to store their preferences in such a way that they surface in different web reports. This also opens the possibility to let them specify these preferences through the Information Delivery Portal, and make them persistent across reports and also across sessions.

Usually, a manager wants to see things from a variety of perspectives. So first she may need an overall report containing aggregate financial information, then she may want to see detail records for her customers. This data may be in different source tables or not; in any case, one will want to have a multitude of web reports to present this data. Then, this method is very useful for defining a preference for a certain product or country only once, and then access a range of differing reports without ever to specify this preference again.
Another user may view exactly the same reports, but will be presented with different data, for another range of products, or at a higher level of detail.

REFERENCES

- Philip Mason [2011]. Using SAS® Stored Processes with JavaScript and Flash to Build Web-Based Applications. Proceedings SAS Global Forum 2011. http://support.sas.com/resources/papers/proceedings11/257-2011.pdf.
- SAS 9.2 Stored Processes: Developer's Guide. http://support.sas.com/documentation/cdl/en/stpug/61271/HTML/default/viewer.htm#titlepage.htm

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Frank Poppe
PW Consulting
PO Box 373
4100 AJ Culemborg
The Netherlands
+31 6 2264 0854
Frank.Poppe@PWconsulting.nl
www.pwconsulting.nl

Andres van Antwerpen
PW Consulting
PO Box 373
4100 AJ Culemborg
The Netherlands
+31 6 57322742
Andres.van.Antwerpen@PWconsulting.nl
www.pwconsulting.nl