Paper 144-2011

Using SAS® Software for Iterative Data Integration (DI)

Evangeline Collado, University of Central Florida, Orlando, FL, USA

ABSTRACT

This large, metropolitan university wanted to know if its graduate students are returning each semester and continuing to completion of their degree program (i.e., retention and graduation statistics). To accomplish this analysis a new data mart that contained all of the historical graduate cohorts, beginning with the 1997-1998 cohort year, and their enrollment information was required. A robust data integration (DI) process had to be developed that would loop through each cohort year, collect the data from multiple tables in the operational system for each master's, doctoral, and specialist student and load it into the data warehouse. After the cohorts were created, another DI process was needed to update the data mart with enrollment statistics for each subsequent year. We were able to accomplish this huge iterative task using a combination of SAS® Data Integration Studio, SAS® Stored Processes, SAS® macro variable, and SAS® Enterprise Guide®.

WHO WE ARE

The mission of Enterprise Decision Support (EDS), a division of Institutional Knowledge Management (IKM), is to provide data integration services and actionable information solutions through the delivery of business intelligence applications, and other knowledge management tools, to support executive and operational decision-making and planning at the University of Central Florida (UCF). The EDS team delivers information and reports in various formats tailored to user need and technical aptitude and we are responsible for system support, administration, and security of the university data warehouse and functional reporting data marts.

INTRODUCTION

Our university has been reporting undergraduate student retention and graduation statistics for many years via web applications developed using SAS® software. The College of Graduate Studies had developed their own method of reporting graduate student information using a combination of MS Access, MS Visual Basic for Applications (VBA), and MS Excel. A database was created in MS Access by running a series of VBA modules that extracted data from the university's Oracle-based operational system, then a series of MS Excel spreadsheets were generated and emailed to the appropriate audience.

Graduate Studies wanted a more robust, web-enabled delivery of retention and graduation analytics and they preferred that the atomic data be stored in the university data warehouse as the official reporting source. Members of the EDS team met with them to gather requirements for the creation of the graduate cohort data mart and web interface. The data integration process was separated into two phases: 1) the creation of each historical cohort and 2) the collection of annual enrollment statistics for each of the cohorts. The information delivery would be accomplished with a stored process using cascading parameters that would create multi-sheet MS Excel output. This paper provides only the details of the data integration process for the historical cohorts.

PHASE ONE: COHORT CREATION

Each cohort year consists of three admit terms: summer, fall, and spring. For example, the cohorts for the year 1997-1998 contain information about students admitted to a Master's, Specialist, or Doctoral graduate program in the Summer 1997, Fall 1997, and Spring 1998 semesters. The data for the graduate student retention cohorts would be extracted from our operational system, so the required data elements for the new data mart had to be mapped to the appropriate Oracle source tables. The decision to include certain fields in the table was based on current and anticipated reporting needs. When the data model was complete, and all source tables identified, the cohort creation data integration process began. The process flow for one semester's data was developed in SAS Data Integration Studio 3.4 and has since been migrated to version 4.21. When this process was successfully tested to ensure that the desired data was collected properly, an iterative job was created that would loop through each term in each cohort year, collect the historical data, and append it to the new table named GRAD_RETENTION_FACT. The process flow for a single term would then be used for each new subsequent cohort year in the future; thus, allowing for a re-usable process.

Figure 1 describes the process flow developed to create one term's data. The first step is to extract the group of students that meet certain criteria from graduate admissions information. The second step pulls in all the required information, such as codes and descriptions of the codes, about the program the student was admitted to. The third step collects bio-demographical data for each student. The resulting data set is then sorted by admit term and student identification number and appended to the data warehouse table.

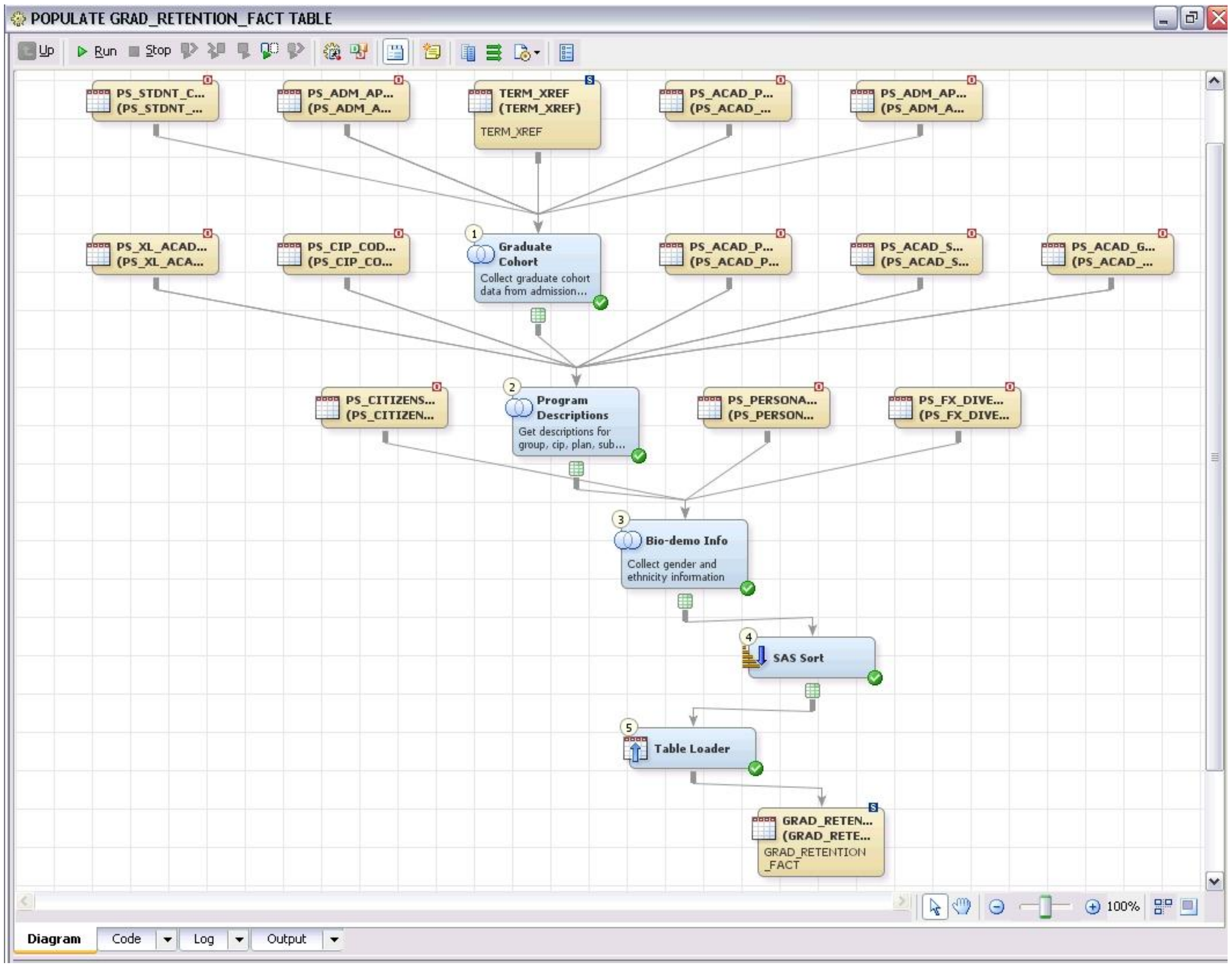
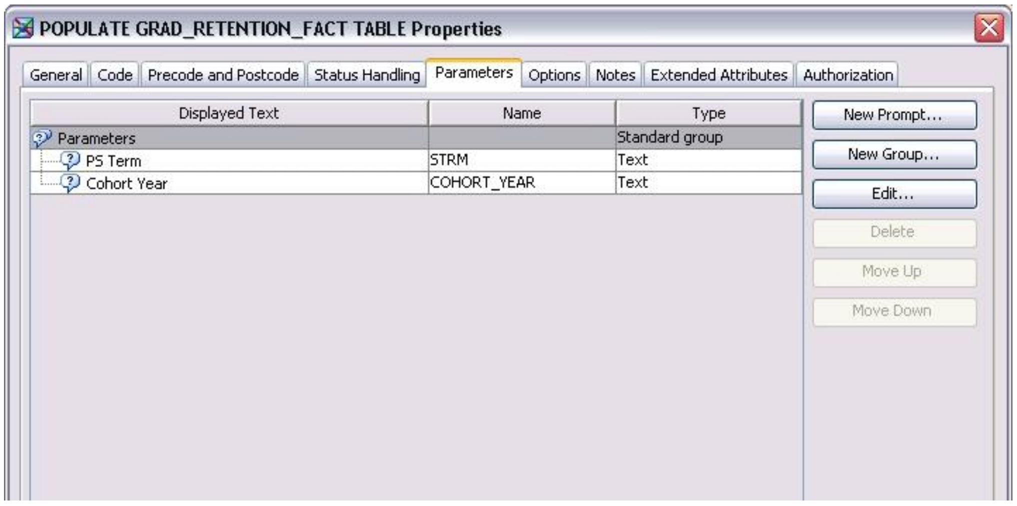

POPULATE GRAD_RETENTION_FACT TABLE
Up
Run
Stop
PS_STDNT_C... (PS_STDNT_...
PS_ADM_AP... (PS_ADM_A...
TERM_XREF (TERM_XREF) TERM_XREF
PS_ACAD_P... (PS_ACAD_...
PS_ADM_AP... (PS_ADM_A...
PS_XL_ACAD... (PS_XL_ACA...
PS_CIP_COD... (PS_CIP_CO...
1 Graduate Cohort Collect graduate cohort data from admission...
PS_ACAD_P... (PS_ACAD_P...
PS_ACAD_S... (PS_ACAD_S...
PS_ACAD_G... (PS_ACAD_...
PS_CITIZENS... (PS_CITIZEN...
2 Program Descriptions Get descriptions for group, cip, plan, sub...
PS_PERSONA... (PS_PERSON...
PS_FX_DIVE... (PS_FX_DIVE...
3 Bio-demo Info Collect gender and ethnicity information
4 SAS Sort
5 Table Loader
GRAD_RETEN... (GRAD_RETE... GRAD_RETENTION_FACT
100%
Diagram
Code
Log
Output

Figure 1. Data Integration Process to Build Graduate Cohorts

The term and cohort year had to be specified at run time so the appropriate data would be collected. Rather than hard-code this information each year, parameters would be used so the DI developer would be prompted to enter the term and cohort year at run time. Figure 2 shows the two parameters defined for this process flow: STRM and COHORT_YEAR.

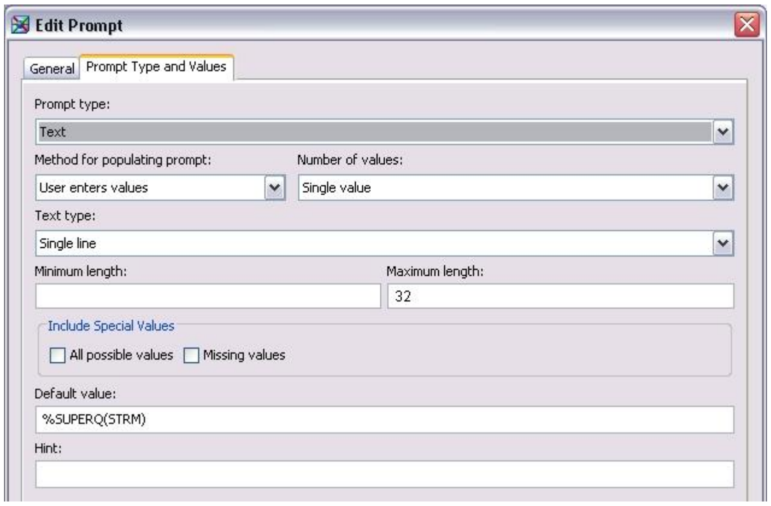


Displayed Text	Name	Type
Parameters		Standard group
PS Term	STRM	Text
Cohort Year	COHORT_YEAR	Text

Figure 2. Process Parameters

During the development and testing phase, the default values of the parameters were set to '1000', which is the STRM for Summer 1997, and '1997-1998', which is the COHORT_YEAR corresponding to that term. After successfully collecting the appropriate data, the DI process was deployed to a stored process that would be executed in SAS Enterprise Guide each year at the end of the spring term when the data for the new annual cohort was available in the operational system. Parameters would be created in this tool so the data integration developer would be prompted for the term and year each time the stored process was executed. Figure 3 displays the default value for STRM after the testing was complete. This value would then be written to the stored process code that was generated when the job was deployed to a stored process.

```
/* Parameter default value(s) for POPULATE GRAD_RETENTION_FACT TABLE  */
%let STRM = %SUPERQ(STRM);
%let COHORT_YEAR = %SUPERQ(COHORT_YEAR);
```



Prompt type: Text

Method for populating prompt: User enters values

Number of values: Single value

Text type: Single line

Minimum length:

Maximum length: 32

All possible values ☐ Missing values ☐

Default value: %SUPERQ(STRM)

Hint:

Figure 3. Default Parameter Value

Deploying a job developed in SAS Data Integration Studio to a stored process is an easy process as shown in Figure 4. Right-clicking on the job name displays a menu where **Stored Process → New . . .** can be selected. The Stored Process Wizard then guides the user step-by-step to complete the properties for the new stored process. Selecting the **Parameters** tab displays the parameters that have been created for this process flow (Figure 5). Clicking the **Test Prompts . . .** button shows how the prompts will be displayed when the stored process is executed (Figure 6).

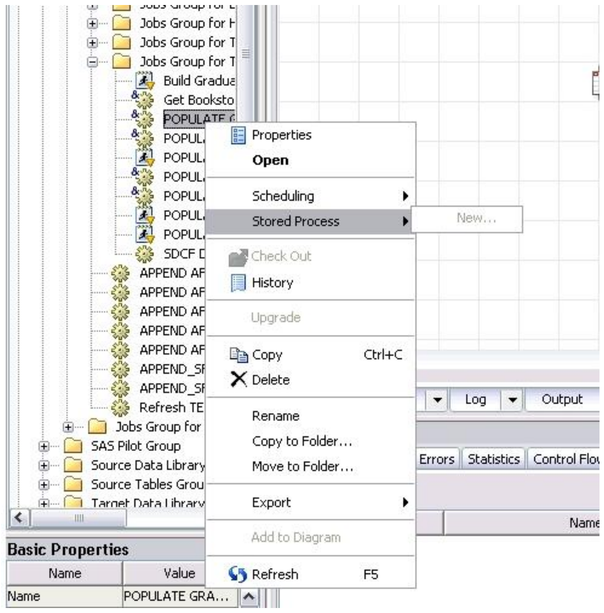
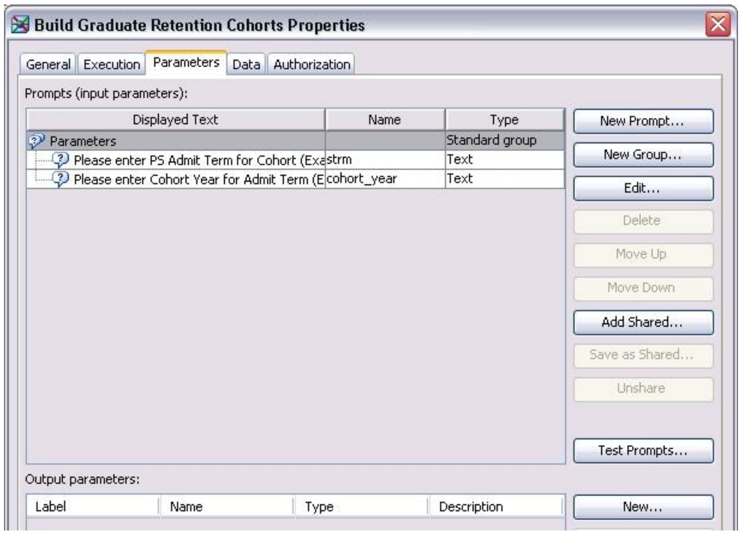


Figure 4. Stored Process Creation



Build Graduate Retention Cohorts Properties

General | Execution | Parameters | Data | Authorization

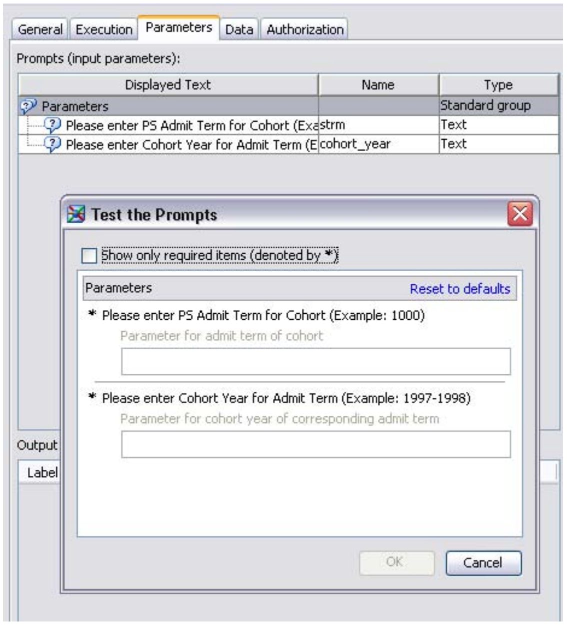
Prompts (input parameters):

Displayed Text	Name	Type
Parameters		Standard group
Please enter PS Admit Term for Cohort (Exa	strm	Text
Please enter Cohort Year for Admit Term (E	cohort_year	Text

Output parameters:

Label	Name	Type	Description

Figure 5. Stored Process Parameters



General | Execution | Parameters | Data | Authorization

Prompts (input parameters):

Displayed Text	Name	Type
Parameters		Standard group
Please enter PS Admit Term for Cohort (Exa	strm	Text
Please enter Cohort Year for Admit Term (E	cohort_year	Text

Test the Prompts

Show only required items (denoted by *)

Parameters Reset to defaults

* Please enter PS Admit Term for Cohort (Example: 1000)
Parameter for admit term of cohort

* Please enter Cohort Year for Admit Term (Example: 1997-1998)
Parameter for cohort year of corresponding admit term

Figure 6. Testing the Prompts

The amount of data that had to be collected spanned 39 terms so it would require someone to answer the prompts that many times. There had to be an easier way to collect the historical information! The solution to this problem was to develop an iterative job in SAS Data Integration Studio.

Figure 7 describes the process flow for the iterative job to create the 39 historical cohorts. The first step was to create a control table (Figure 8) that contained the STRM and COHORT_YEAR values that would get passed to the job described above. Thus, all terms between ‘1000’ (Summer 1997) and ‘1380’ (Spring 2010) were extracted from the TERM_XREF table (Figure 8). The COHORT_YEAR field was created based on another field in the source table by using an advanced expression. The format of the BOE_TERM_ID is YYYYTT, where YYYY is the four-digit year and TT is a term identifier: ‘05’ is summer, ‘08’ is fall, and ‘01’ is spring.

```
CASE WHEN SUBSTR(BOE_TERM_ID,5,2) IN ('05','08') THEN SUBSTR(BOE_TERM_ID,1,4)||
'-'||(PUT(INPUT(SUBSTR(BOE_TERM_ID,1,4),4.) + 1,4.))
WHEN SUBSTR(BOE_TERM_ID,5,2) = '01' THEN (PUT(INPUT(SUBSTR(BOE_TERM_ID,1,4),4.) -
1,4.))||'-'||SUBSTR(BOE_TERM_ID,1,4)
ELSE ''
END
```

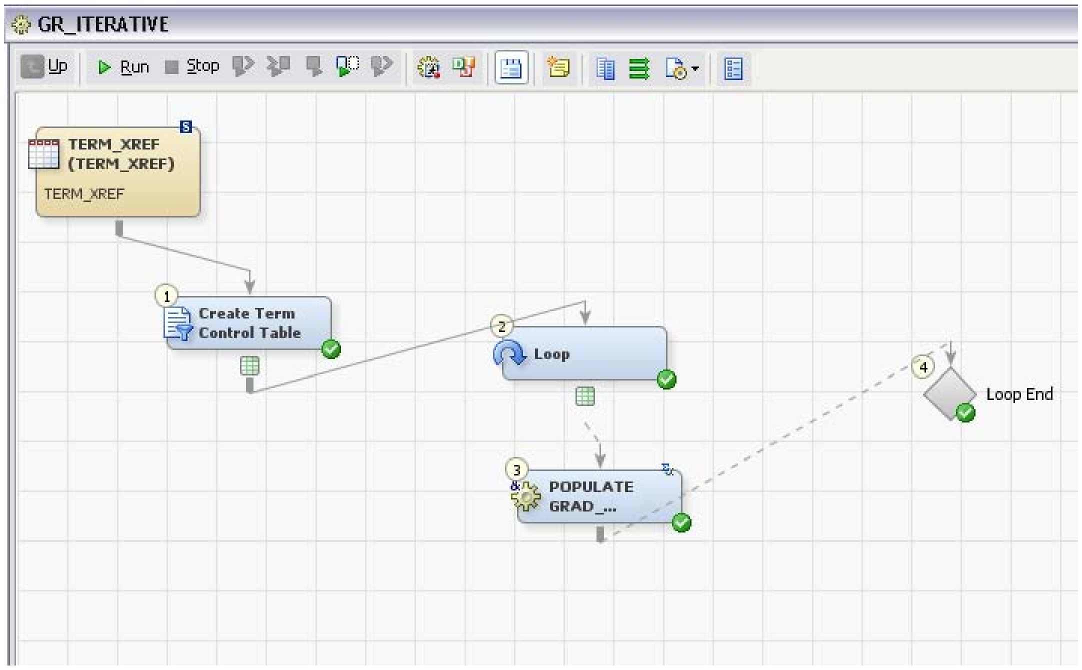
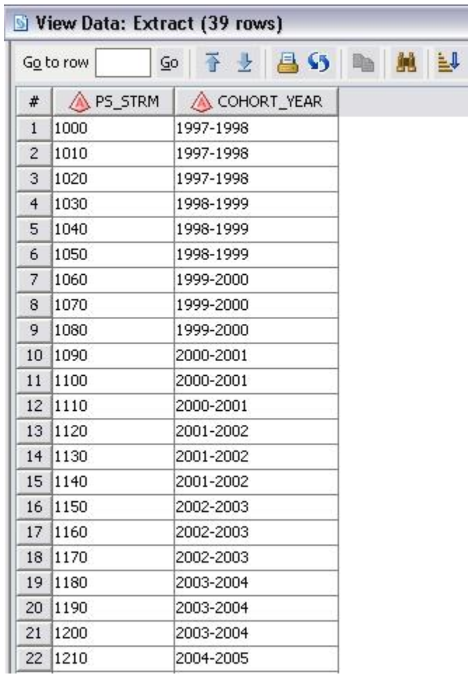


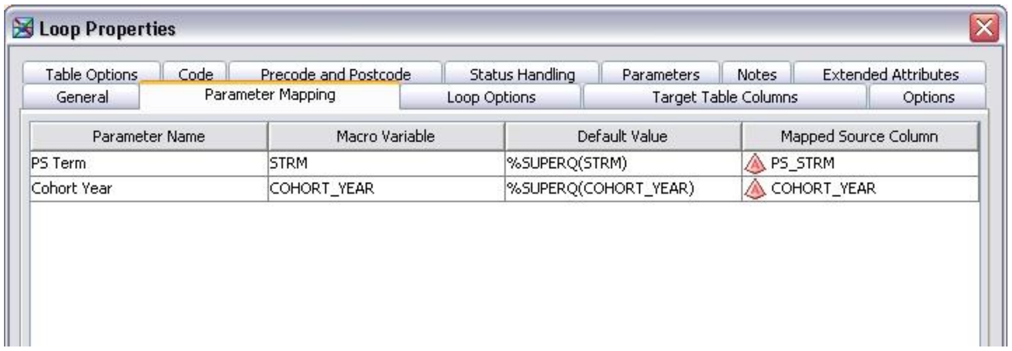
Figure 7. Iterative Process Flow


View Data: Extract (39 rows)

#	PS_STRM	COHORT_YEAR
1	1000	1997-1998
2	1010	1997-1998
3	1020	1997-1998
4	1030	1998-1999
5	1040	1998-1999
6	1050	1998-1999
7	1060	1999-2000
8	1070	1999-2000
9	1080	1999-2000
10	1090	2000-2001
11	1100	2000-2001
12	1110	2000-2001
13	1120	2001-2002
14	1130	2001-2002
15	1140	2001-2002
16	1150	2002-2003
17	1160	2002-2003
18	1170	2002-2003
19	1180	2003-2004
20	1190	2003-2004
21	1200	2003-2004
22	1210	2004-2005

Figure 8. Control Data

Next, a **Loop** transformation was added to the process flow, followed by the job developed for a single term as described previously in this document (drag and drop from the tree), and then a **Loop End** transformation. The **Parameter Mapping** tab on the **Loop** transformation allows selection of the source columns that map to the SAS macro variables (parameters) in the job (Figure 9). The **Loop Options** tab provides options for parallel processing but that didn't apply in our case so it was disabled. There were not any options to set in the **Loop End** transformation. When this iterative process flow was run, the GRAD_RETENTION_FACT table was created with all 39 cohorts. The first phase of the data collection process was now complete.


Loop Properties

Parameter Name	Macro Variable	Default Value	Mapped Source Column
PS Term	STRM	%SUPERQ(STRM)	PS_STRM
Cohort Year	COHORT_YEAR	%SUPERQ(COHORT_YEAR)	COHORT_YEAR

Figure 9. Parameter Mapping

PHASE TWO: CAPTURE ENROLLMENT STATISTICS

The next step in building the graduate retention data mart was to capture enrollment statistics over time for each student in each cohort. The College of Graduate Studies wanted to maintain the following information in the retention data mart:

- Candidacy date for doctoral students
- Degree information
- Annual credit hours taken
- Flags for each year, coded 1 or 0, to indicate if enrolled, graduated, withdrawn, dismissed, or dropped out

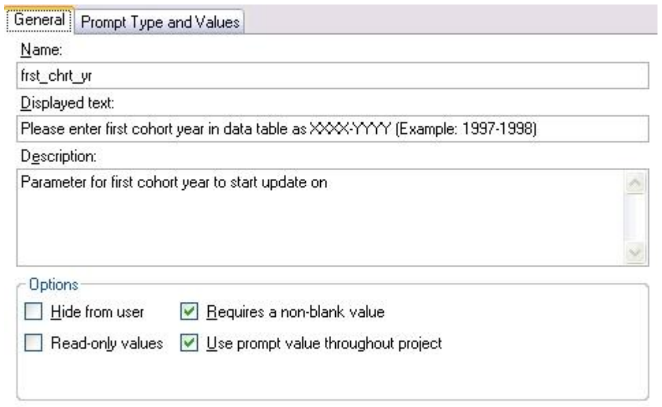
The fields that contain the above data were created on the GRAD_RETENTION_FACT table with null/missing values during the cohort creation process in phase one so this phase would collect the data and then update the table. A robust iterative process had to be developed that would start with the first cohort (Summer 1997), capture the enrollment information for each academic year starting with the cohort year and ending with the last available academic year (2009-2010 in this case), and then perform the updates of the fields for each cohort. Due to the complexity of this data collection and update process, the best solution was to create a SAS® program with macro variables that uses do-loop processing within macro code. This program would need to prompt for the first cohort year on the table, last cohort year on the table, and last available academic year at execution time so SAS Enterprise Guide was the tool selected for development of this phase.

The following process flow was outlined and the code was written via a code node in SAS Enterprise Guide, successfully tested, and implemented:

1. Collect candidacy dates for all doctoral students; update the CANDIDACY_DATE field.
2. Collect degree information – degree level, term awarded, and program information – then update all the degree fields.
3. If the student was in a doctoral program, check to see if a master's degree was received along the way; update the other degree fields.
4. Sum all enrolled credit hours for the student in the academic year; update the ENRL_CREDIT_YR*n* and ENROLLED_YR*n* fields, where *n* is the number of each academic year for the cohort. At the time the program was developed a maximum of 15 years of information was desired so *n* = 1 to 15. The cohort year is represented by *n* = 1.
5. If student was in a doctoral program, sum all dissertation and/or thesis credit hours; update the DRT_THE_CREDIT_YR*n* field.
6. If student was not enrolled, or did not receive a degree, check to see if he/she formally withdrew from the program; update WITHDRAWN_YR*n*.
7. If student was not enrolled, did not receive a degree, or did not withdraw, check to see if he/she was dismissed or discontinued; update DISMISSED_YR*n*.
8. If student did not satisfy any of the previous steps default to drop-out status; update DROP_OUT_YR*n*.

SAS® ENTERPRISE GUIDE® DEVELOPMENT

The first step of the program development was to define the project parameters. From the **View** menu, a selection is available to display the **Prompt Manager.** Selecting **Add** allows creation of a new prompt. Figure 10 shows the general definition of the first cohort year parameter (first_chrt_yr) and Figure 11 displays the type and default value for this prompt. A default value was not required but added to reinforce the format of the expected response.



General | Prompt Type and Values

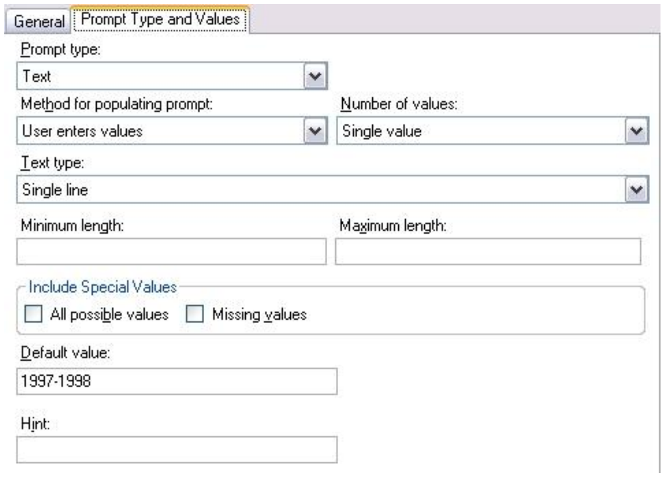
Name: frst_chrt_yr

Displayed text: Please enter first cohort year in data table as XXXX-YYYY (Example: 1997-1998)

Description: Parameter for first cohort year to start update on

Options
- ☐ Hide from user
- ✓ Requires a non-blank value
- ☐ Read-only values
- ✓ Use prompt value throughout project

Figure 10. General Tab of New Prompt Definition



General | Prompt Type and Values

Prompt type: Text

Method for populating prompt: User enters values

Number of values: Single value

Text type: Single line

Minimum length:

Maximum length:

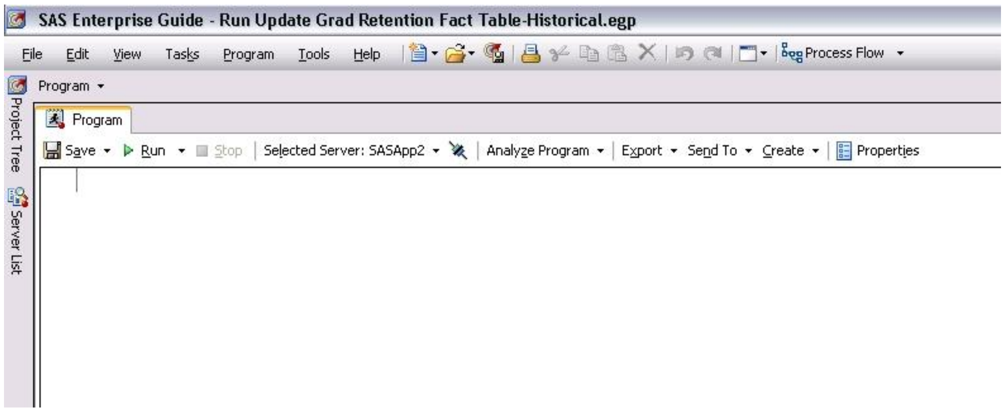
Include Special Values
- ☐ All possible values
- ☐ Missing values

Default value: 1997-1998

Hint:

Figure 11. Prompt Type and Values of New Prompt

Next, selecting **File** → **New** → **Program** opens a new window (Figure 12) which is very similar to the SAS Software **Enhanced Editor** (Figure 13). The program code is color-coded also and SAS Enterprise Guide 4.3 includes syntax tool-tips.



SAS Enterprise Guide - Run Update Grad Retention Fact Table-Historical.egp

Figure 12. New Program Window

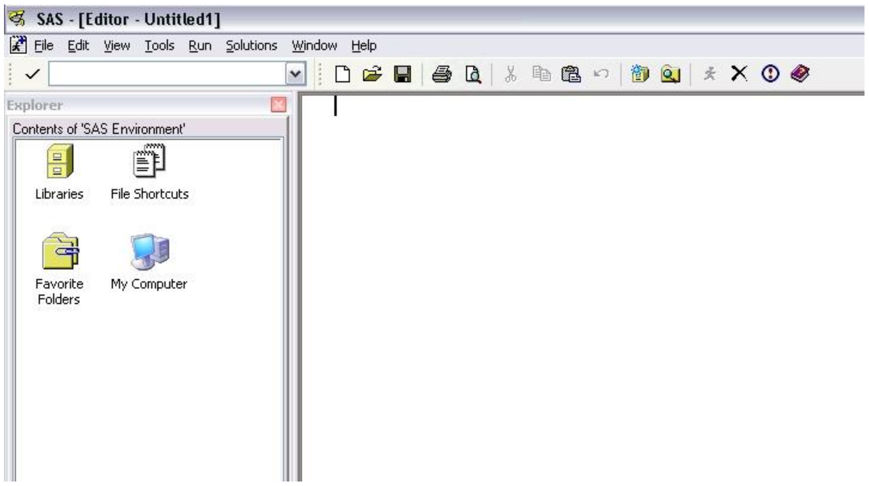

SAS - [Editor - Untitled1]
File Edit View Tools Run Solutions Window Help
Explorer
Contents of 'SAS Environment'
Libraries
File Shortcuts
Favorite Folders
My Computer

Figure 13. SAS® Software Enhanced Editor

At the beginning of the program, the following code is how the macro variables were created for the do-loop processing. If the first cohort year is more than 15 years back from the current academic year being processed then set to stop at 15.

```
%macro updt;
/* Create macro variables from collected parameters */
    /* Extract first four characters from first cohort year */
%let beg_yr = %substr(&frst_chrt_yr, 1, 4);
    /* Extract first four characters from last cohort year */
%let end_yr = %substr(&lst_chrt_yr, 1, 4);
    /* Extract 1st four characters from retention acad year */
%let file_yr = %substr(&lst_acad_yr, 1, 4);
    /* Calculate ending value for cohort year do loop */
%let chrt_num = %eval(&end_yr - &beg_yr);
    /* Calculate ending value for retained year do loop */
%let ret_num = %eval(&file_yr - &beg_yr);

/* Stop updates at year 15 - delete this section if we add more columns to the
table */
%if %eval(&ret_num) > 15 %then %let ret_num = 15;
```

The do-loop processing can now begin. We start by looping through each cohort year beginning with the one specified at run-time for first cohort year and ending with the one specified at run time for last cohort year. This is the outer cohort year loop.

```
%do i = 0 %to &chrt_num;
          /* Calculate cohort year to update */
    %let chrt_yr = %eval(&beg_yr + &i);
          /* Create macro var for criteria */
    %let chrt_yr_parm = &chrt_yr-%eval(&chrt_yr +1);
```

Next, for each academic year, we need to see if a degree was awarded for the admitted program. If a doctoral student and a degree was not awarded in the doctoral program then see if a master's degree was received. If no degree awarded in that academic year, we will check to see if enrolled and sum total credit hours and dissertation hours (if doctoral student).

If sum of hours is zero then we will look to see if student withdrew. If not withdrawn, then look to see if student was dismissed. If not graduated, enrolled, withdrawn or dismissed then student will be flagged as dropped out. All flags will get updated. This is the inner degree/retention update loop.

```
    %do j = 0 %to &ret_num;
          %let acad_yr = %eval(&chrt_yr + &j);       /* Calculate year of update */
          %let year = YR%eval(%unquote(&j) + 1); /* Calculate retained yr number */
          %let prev = YR%eval(%unquote(&j));     /* Calculate previous yr number */
```

The following section provides examples of where the macro variables are used in the program as it processes through each step outlined above in the process flow for the capture of enrollment statistics.

A single macro variable was needed that contained the values of the three terms in a given academic year when data was required from source table(s) for an entire year:

```
proc sql noprint;
    select distinct trim(ps_strm) into : strm separated by '", "'
    from source_tablename where substr(acad_year, 1, 4) = "&acad_yr";
quit;

Criteria example: where a.strm in ("&strm")
```

Three separate macro variables were needed when term processing was necessary:

```
proc sql noprint;
    select distinct trim(ps_strm) into : strm1 - :strm3
    from source_tablename where substr(acad_year, 1, 4) = "&acad_yr";
quit;

Criteria example:
    and (b.effdt between (select term_begin_dt from source_tablename
                 where acad_career = 'GRAD' and strm = "&strm1")
          and (select term_end_dt from source_tablename
                 where acad_career = 'GRAD' and strm = "&strm3"))
```

Criteria to create temporary lookup tables and/or update fields using cohort year:

```
    where cohort_year = "&chrt_yr_parm"
```

After data processing was complete and all fields were updated the loops were ended and the macro code block ended:

```
    %end;                              /* End inner degree/retention update loop */
    %let ret_num = %eval(&ret_num - 1);/* Calculate maximum number for next loop */
%end;                                  /* End outer cohort year loop             */
%mend;
```

By using do-loop processing with macro variables in a program, the entire collection and update process of phase two was executed with minimal manual intervention and the GRAD_RETENTION_FACT table contained all the required historical data.

CONCLUSION

The College of Graduate Studies at the University of Central Florida approached Enterprise Decision Support with a challenging request. They wanted a new robust method of reporting retention and graduation statistics to the UCF community which required the creation of a reporting data mart. They had been using MS Access, MS VBA modules, and MS Excel to generate multiple spreadsheets that were then emailed to the appropriate audience. They desired a web-enabled reporting environment so it was decided that the established cohorts would be stored permanently in the existing enterprise data warehouse and SAS software tools would be used to collect and update the data and create the web-based delivery application. A two-phase data integration process had to be developed to accomplish the data collection and update task.

Phase one – creation of historical cohorts – was accomplished using SAS Data Integration Studio by creating two jobs/process flows. The first job creates a single cohort for one term of a specified cohort year. The second process flow creates a control table containing each of the terms needed for the historical build and then iteratively creates each cohort using the first job.

Phase two – capture enrollment statistics – was accomplished using a program developed in SAS Enterprise Guide that used macro variables and do-loop processing to iterate through each academic year. The first and last cohort year and the academic year being processed were collected via prompts.

The GRAD_RETENTION_FACT table was built with all of the required historical cohorts and will need new cohorts and enrollment statistics added each academic year. The historical processes detailed in this paper are applied annually without much modification. The first process flow for phase one is used to create a new annual cohort by executing a SAS Stored Process within the SAS Enterprise Guide interface. The program used in phase two has been modified so that it processes only one academic year (the most recent) of enrollment statistics for each of the cohorts. Thus, the many tools available in the SAS® Intelligence Platform software suite enabled us to accomplish a huge iterative data integration task with processes that were extremely vigorous during the initial data capture and successfully re-usable for our ongoing needs.

RECOMMENDED READING

- SAS® Data Integration Studio 4.21 User's Guide
- SAS® 9.2 Stored Processes Developer's Guide
- Getting Started with SAS® Enterprise Guide® Tutorial – Available at http://support.sas.com/documentation/onlinedoc/guide/tut43/en/

CONTACT INFORMATION

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

Name: Evangeline (Angel) Collado
Enterprise: University of Central Florida, Enterprise Decision Support
Address: P.O. Box 160021
City, State ZIP: Orlando, FL 32816-0021
Work Phone: (407) 823-4968
Fax: (407) 823-4769
E-mail: Evangeline.Collado@ucf.edu
Web: www.iroffice.ucf.edu