Paper 157-31

"Proc Photo" : A Generic Method to Capture Time-Stamped States of Operational Tables

Richard Massé, Université Laval, Quebec City, Canada
Luc Simon, Université Laval, Quebec City, Canada

ABSTRACT

The procedure described hereafter is useful to collect and store in an efficient manner successive states of selected operational tables.

Using a series of SAS® macros, operational tables are read and copied in a different " historical database ". At selected intervals, operational tables are read again, and another series of macros identifies the differences between the current state of the tables and the last occurrence in the historical database. Finally, the historical tables are updated to reflect the current state of the operational tables.

Sample SAS data statements that easily exploit the historical database or data warehouse are also presented.

To minimize the programming and maintenance effort, a generic process was developed. This system uses SAS to capture ORACLE operational tables and write into an ORACLE historical database, but it could easily be adapted to other databases. Theses procedures are mostly platform independent.

The target audience of this paper comprises many people involved in the information analysis process such as system architects, ETL specialists and information analysts. A non-technical conceptual overview of “Proc Photo” is presented. Detailed code will be available upon request.

INTRODUCTION

In many organizations, exploitation by analysts of the information contained in the operational systems represents a significant challenge, partly because these information systems are rarely designed to keep the historical dimension of the records. In most instances, only the latest state is kept in the operational database.

Often, modification of the operational system is impracticable, when not impossible. Data warehousing can solve this problem. Unfortunately, many data warehouse solutions are themselves a challenge to deploy, with uncertain return on investments (ROI).

The Institutional Research Office at Université Laval was faced with such a situation, and we developed a SAS based solution that allowed us to create and manage an historical copy of the operational tables that contain the institutional information our office needs to access. This solution makes use of generic macros, and could be adapted to different environment that have similar data warehousing needs.

OVERVIEW OF " PROC PHOTO "

Our general approach is to create a copy of the selected operational tables in a different database, while adding the following variables in order to obtain a versioned copy of each table :

Name of new variables added	Possible values	Note
VERSION_NUMBER	1..∞	Equals 1 for initial loading. Incremented by 1 for each subsequent writing of a given record of the table.
ADDITION_DATE	Valid date	Date of the initial creation of the record. Never modified.
DEACTIVATION_DATE	Valid date	Set initially to 31/12/3999. Set to the current date whenever a new version of the record is written.
ACTIVE_FLAG	Yes, No	Set initially to Yes. Changed to No whenever a new version of the record is created or if the record has been deleted in the operational table.

The process is illustrated using the following " CLIENT " table in an operational database. This table will be copied in a new table in the historical database (Data warehouse).

CLIENT (operational)			
PKey	A	B	C
1	1	2	3
2	6	5	6
3	7	8	9
4	10	11	12
5	13	14	15

+

CLIENT (Historical)							
PKey	VERSION_ NUMBER	A	B	C	ADDITION_ DATE	DESACTIVATION_ DATE	ACTIVE_ FLAG
1	1	1	2	3	2005-06-15	3999-12-31	YES
2	1	4	5	6	2005-06-15	3999-12-31	YES
3	1	7	8	9	2005-06-15	3999-12-31	YES
4	1	10	11	12	2005-06-15	3999-12-31	YES

Figure 1 : Data contained in operational and historical tables (initial loading)

After the initial loading, both tables are almost identical, the only difference being the four new variables. These new variables are sufficient to archive successive states of the operational table.

Naming convention

In order to rely on fewer parameters when calling generic macros, we adopted the following naming convention : Tables in the data warehouse are given the same name as in the operational database, except for a different prefix

- OP_ → Indicates an operational table
- HIMG_ → Indicates a historical table

In the first example, the CLIENT table in the operational database would be named OP_CLIENT and its counterpart in the data warehouse would be called HIMG_CLIENT. There is no need to modify the operational structure in order to conform to this naming convention, as aliases or table views can simply be defined for this purpose.

Macros for the initial setup of the data warehouse structure

The two following macros are useful to create the proper table structure in the data warehouse, and for the initial loading of the data :

%CREATE_DW_STRUCTURE

This macro is executed to create a replicate structure of a specified operational table into the data warehouse

```
%CREATE_DW_STRUCTURE(operational_library,dw_library,table_name,alias);
```

Parameter	Description
operational_library	Library standing for the Operational ORACLE Database containing the desired table or view.
dw_library	Library standing for the Historical ORACLE Database (DW) containing the desired table or view.
table_name	Name of the desired table or view. Do not specify the prefix (OP_ or HIMG_).
alias	Alias (4 to 7 letters) to identify ORACLE primary key constraint as well as index names.

Example : %CREATE_DW_STRUCTURE(lib_oper,lib_hist,client,clie);

%INITIAL_DW_LOAD

This macro is executed once to copy data from the operational table into the data warehouse.

```
%INITIAL_DW_LOAD(operational_library,dw_library,table_name);
```

Parameter	Description
operational_library	Library standing for the Operational ORACLE Database containing the desired table or view.
dw_library	Library standing for the Historical ORACLE Database (DW) containing the desired table or view.
table_name	Name of the desired table or view. Do not specify the prefix (OP_ or HIMG_).

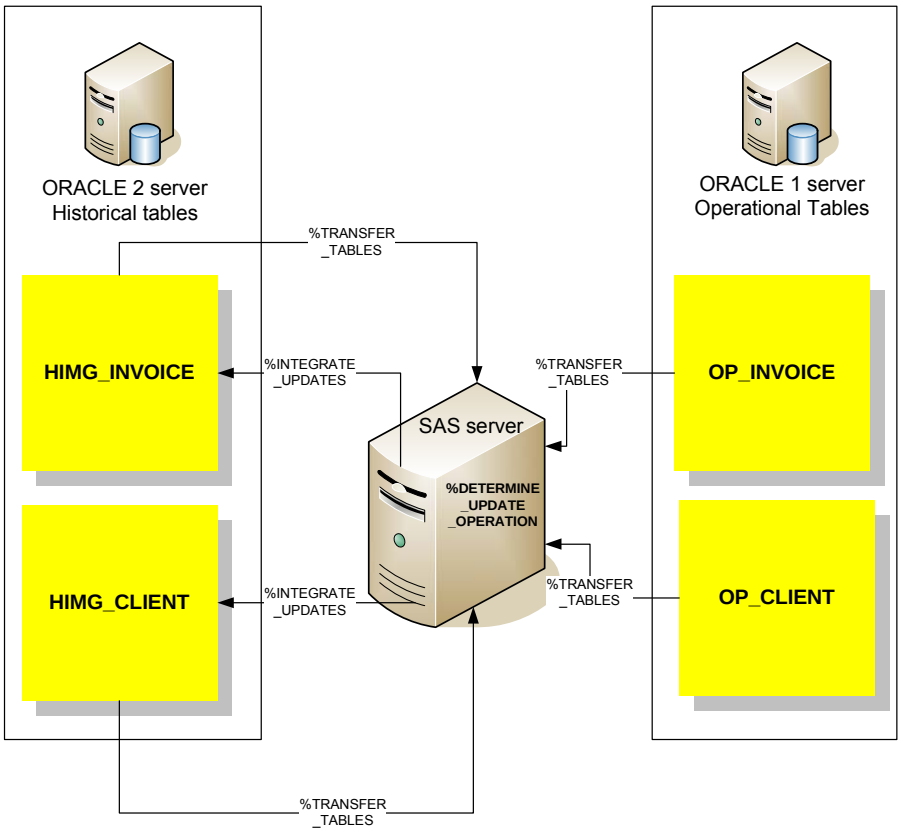
Example : %INITIAL_DW_LOAD(lib_oper,lib_hist,client);

The core of " PROC PHOTO " is comprised of three main macros designed to periodically synchronize the operational tables with the data warehouse :

- **%TRANSFER_TABLES** → Copy of required data in a " staging " area.
- **%DETERMINE_UPDATE_OPERATIONS** → Determination of all the required modifications that need to be applied to the data warehouse records to accurately reflect the current operational state.
- %**INTEGRATE_UPDATES** → Integration in the data warehouse of the output of %DETERMINE_UPDATE_OPERATIONS

By first running a procedure to configure the workspace and set the libraries used as staging area, these don't need to be specified each time the following macros are executed.

Flow Chart of used macro's



%TRANSFER_TABLE

Execution of this macro results in the transfer in a staging area of the content of both the operational table and its data warehouse counterpart.

```
%TRANSFER_TABLE(operational_library,dw_library,table_name);
```

Parameter	Description
operational_library	Library standing for the Operational ORACLE Database containing the desired table.
dw_library	Library standing for the Historical ORACLE Database (DW) containing the desired table.
table_name	Name of the desired table. Do not specify the prefix (OP_ or HIMG_).

Example : %TRANSFER_TABLE(lib_oper,lib_hist,client);

% DETERMINE_UPDATE_OPERATIONS

Execution of this macro results in the compilation of all the updates that will need to be applied to the DW table to reflect the current state of the operational table, following a record by record comparison of the data in both tables.

```
%DETERMINE_UPDATE_OPERATIONS(table_name,alias);
```

Parameter	Description
table_name	Name of the desired table. Do not specify the prefix (OP_ or HIMG_).
alias	Alias to identify temporary files generated by the procedure. It is important that different tables treated in a given batch are given ***different aliases***.

Example: %DETERMINE_DW_OPERATIONS(client,clie);

%INTEGRATE_UPDATES

Execution of this macro updates the information contained in the data warehouse table.

```
%INTEGRATE_UPDATES(table_name,alias,key);
```

Parameter	Description
table_name	Name of the desired table. Do not specify the prefix (OP_ or HIMG_).
alias	Alias to identify temporary files generated by the procedure. It is important that different tables treated in a given batch are given ***different aliases***.
key	Specify the key from the operation table. This key must never contain the variable VERSION_NUMBER

Example: %INTEGRATE_UPDATES(client,clie,seq_client);

STEP BY STEP WALKTHROUGH

Consider tables CLIENT and INVOICE from an operational database that needs to be replicated in the data warehouse. The SAS library *lib_oper* references the operational production database, while *lib_hist* references the data warehouse. For this example, the current date is June 15th 2005.

Step 1 : Creation of the corresponding table in the data warehouse

```
…
libname lib_oper oracle … ;
libname lib_hist oracle … ;
%CREATE_DW_STRUCTURE(lib_oper,lib_hist,client,clie);
%CREATE_DW_STRUCTURE(lib_oper,lib_hist,invoice,invo);
…
```

Step 2 : Copy data from the operational table into its DW counterpart

…

```
libname lib_oper oracle … ;
libname lib_hist oracle … ;
%INITIAL_DW_LOAD(lib_oper,lib_hist,client);
%INITIAL_DW_LOAD(lib_oper,lib_hist,invoice);
```

…

Notes : Steps 1 and 2 are only executed once, and never re-executed on the same table.
Steps 1 and 2 can be executed either in *development* or *production* environments, simply by specifying the proper library.

The next figure depicts the content of both tables after Step 2.

Consider that a month has passed, and it has been decided to update this table in the data warehouse every month. Current date is now July 15th 2005. If we examine the current state of the operational table, we see that some records have been modified, created or suppressed :

- Creation of record # 5 where A=13, B=14 et C=15
- Modification of record # 2 : A=6
- Suppression of record # 1

OP_CLIENT				
PKey	A	B	C	
~~1~~	~~1~~	~~2~~	~~3~~	(Deleted)
2	6	5	6	(Updated)
3	7	8	9	
4	10	11	12	
5	13	14	15	(Added)

The core of PROC PHOTO is executed in steps 3 thru 5

Step 3 : Transfer both versions of every table in the staging area.

…

```
libname lib_oper oracle … ;
libname lib_hist oracle … ;
%TRANSFER_TABLE (lib_oper,lib_hist,client);
%TRANSFER_TABLE (lib_oper,lib_hist,invoice);
```

… and so on for every other table that needs updating.

Step 4 : Compile all the updates that will need to be applied to the DW table to reflect the current state of the operational table.

…

```
%DETERMINE_UPDATE_OPERATIONS (client,clie);
%DETERMINE_UPDATE_OPERATIONS (invoice,invoice);
```

… and so on for every other table that needs updating.

Step 5 : Update the information contained in the DataWarehouse tables.

…

```
libname lib_hist oracle … ;
%INTEGRATE_UPDATES (client,clie,seq_client); /* ALIAS MUST BE THE SAME AS IN STEP 4 */
%INTEGRATE_UPDATES (invoice,invo,seq_invoice);
```

… and so on for every other table that needs updating.

After this last step, the data warehouse exactly reflects the status of the operational tables as of July 15th, 2005.

HIMG_CLIENT								
PKey	VERSION_NUMBER	A	B	C	ADDITION_DATE	DESACTIVATION_DATE	ACTIVE_FLAG	
1	1	1	2	3	2005-06-15	2005-07-15	NO	(Deleted)
2	1	4	5	6	2005-06-15	2005-07-15	NO	
2	2	6	5	6	2005-07-15	3999-12-31	YES	(Updated)
3	1	7	8	9	2005-06-15	3999-12-31	YES	
4	1	10	11	12	2005-06-15	3999-12-31	YES	
5	1	13	14	15	2005-07-15	3999-12-31	YES	(Added)

Lastly, lets suppose that in the operational table record #2 is modified on July 20th 2005, to read A=2 and B=2. The following figure depicts the state of the data warehouse table after Proc Photo is executed on July 21st 2005. Note that there is now a third version of record #2 in the data warehouse.

HIMG_CLIENT							
PKey	VERSION_NUMBER	A	B	C	ADDITION_DATE	DESACTIVATION_DATE	ACTIVE_FLAG
1	1	1	2	3	2005-06-15	2005-07-15	NO
2	1	4	5	6	2005-06-15	2005-07-15	NO
2	2	6	5	6	2005-07-15	2005-07-15	NO
2	3	2	2	6	2005-07-21	3999-12-31	YES
3	1	7	8	9	2005-06-15	3999-12-31	YES
4	1	10	11	12	2005-06-15	3999-12-31	YES
5	1	13	14	15	2005-07-15	3999-12-31	YES

(Updated)

Once the system is properly configured, execution of Proc Photo is straightforward and produces an updated version of the data warehouse.

QUERY OF THE DATA WAREHOUSE

Execution of the PROC PHOTO macros results in the creation of a data warehouse containing selected tables with additional fields that can be used to select specific versions of the data. Here are some examples of selection criteria that can be inserted in the queries of the data warehouse :

Selection of the latest version (last update)

With this selection criteria, an exact DW copy of an operational table as it was last updated can be queried :

```
ACTIVE_FLAG = "Yes"
```

Selection of the state of the data warehouse on a given date

To set the content of the DW as it appeared on July 15th:

```
ADDITION_DATE <= 2005-07-15 AND
DEACTIVATION_DATE > 2005-07-15
```

Comparison between two dates

In some circumstances, it maybe useful to be able to compare data as it appeared on two different dates in the DW. Two different SAS libraries created for this purpose :

```
libname abc "s:\data_2005_08_01"; /* Image of the DW on 2005-08-01 */
libname def "s:\data_2005_09_01"; /* Image of the DW on 2005-09-01 */

proc means data=abc.table_a;
var …
run;

proc means data=def.table_a;
var …
run;
```

Results from these two procedures can be compared and analysed to uncover any discrepancies in the data as it appeared in the DW on the two dates specified. Retroactive corrections performed on the operational tables can be detected and their impact on previously reported numbers can be quantified. For example, let's suppose the mean salaries are computed and reported yearly for different groups of employees. At the end of the 2001 fiscal year, a number was reported. A few months later, a collective agreement was settled which contained a provision for retroactive payments for some employees. The consequence is that the 2001 mean salary will differ if it is computed from data available at the end of FY 2001 or from data available six months later… and **both** numbers can be computed from the data warehouse.

Audit on modifications to the operational database structure

It is possible to use PROC PHOTO to monitor changes occurring in the operational database. A simple report can periodically provide data on the number of records updated in the data warehouse for any or all of the tables monitored. Such Quality Control measures can insure prompt reaction when the content of critical tables is modified in the operational system. Another type of QC report can be obtained to monitor changes that can occur in the structure of the operational database, like adding new tables or modifying columns in a table. For this purpose, the dictionary of the operational database can be captured in the data warehouse by PROC PHOTO : Either include the DBA_TABLES and DBA_TAB_COLUMN if you are in an ORACLE environment or the more generic SASHELP.VTABLE and SASHELP.VCOLUMN that can return the dictionary of any database accessible by SAS.

TECHNICAL CONSIDERATIONS

Note concerning the required environment

It is possible to automate the PROC PHOTO process. The system scheduler can be programmed to execute the different macros at a specified times. The frequency can be adjusted to suit particular situations, with the understanding that more frequent updates will have a much larger impact on the cumulative processing time than on the amount of disk space required.

The poster will present some performance stats compiled in our own environment.

How to freeze the operational tables to ensure simultaneous updates of many tables

Since the Proc Photo process is executed in a linear fashion, one table at a time, with a significant processing time especially for large tables, it may be useful to "freeze" all the tables that will be processed in a given batch to ensure that the updates will be exactly synchronized. The following option can be specified in the libname statement and will also not interfere with concurrent operation of ORACLE databases:

```
dbconinit="set transaction isolation level serializable"
```

Query optimization to execute PROC PHOTO

Given the actual performance of PROC PHOTO when executed for our Institutional Research DataWarehouse (about 5 million records in the largest table, 3 Go of total disk space, processing time 120 minutes), we did not need to install triggers or similar mechanisms to flag, directly in the operational tables, the individual records that need to be updated in the data warehouse. Organizations that need to process much larger tables and/or need to more frequent updates should consider implementing such mechanisms to optimize the transfer to the staging environment.

Working with tables and/or table views

The operational structure specified in PROC PHOTO to define the source data can be either a table or a table view. Table views can simplify access to some data by performing some pre-processing, like merging composite keys in a single column. It can also be useful to use table views in order to rename some columns to provide more significant names to data warehouse users.

When defining a table view, an important consideration is the appropriate selection of columns to be captured in the data warehouse, especially for large tables. If too many columns are selected, more changes than necessary may be detected by Proc Photo (in our case, modifications to a record in the operational database that are inconsequential for the reporting needs of the IR Office), and the data warehouse size will increase needlessly. On the other hand, if an important column was not included and one later determine that it needs to be captured, all existing records in the data warehouse will have to be updated, in effect instantly doubling its size. Obviously, finding the proper balance is particularly important for the largest tables.

CONCLUSION

PROC PHOTO was developed as a relatively low cost solution for the data warehousing need of the Institutional Research Office at Université Laval. By making use of SAS macros and a generic approach to create a versioned copy of a number of operational tables, the core functionality of a data warehouse is achieved. Most reports produced by our IR office are now being redesigned as queries to the data warehouse.

Since most reports are produced yearly or quarterly by our IR Office and, more importantly, since it was determined that all pertinent information would indeed be captured in the data warehouse if it was updated at least monthly, we have now decided to update most of the tables in the data warehouse every two weeks. The update frequency can be adapted very easily to suit different needs.

An important consideration should be that critical reports already published from operational databases can be faithfully replicated by the IR Office from the data warehouse. This may mean that the frequency of updates is modified, for no other reason than to coincide with production dates of some important operational reports.

ABOUT THE AUTHORS

Since 1989, Richard Massé works at Université Laval, first in the IT department where he successively manned the help desk for SAS users, became a functional analyst and later acted as institutional data manager. He is currently implementing a data warehouse tailored to the needs of the Institutional Research Office.

Luc Simon was trained as a chemist and molecular biologist, did bench lab research for a number of years before moving to research administration and then to the Institutional Research Office. He participates in the redevelopment of the Office¹s Information Infrastructure, the core of which is the data warehouse.

Request to the authors:
(418) 656-2131 ext. 4258
Richard.Masse@vrex.ulaval.ca