Paper 110 – 2010

Using PROC SQL to Build and Incrementally Update a Data Mart

Ben Cochran, The Bedford Group, Raleigh, NC

Abstract

Often SAS users need to access data from non-SAS sources. This is especially true when constructing a SAS data warehouse from other vendors' databases. While this task is not too difficult, sometimes unforeseen challenges can arise, especially when dealing with date values. This tutorial initially takes a look at several methods for accessing different kinds of data to do the initial load of the data warehouse. Then attention is given to various ways of doing incremental updates and how to overcome some potential problems.

This paper follows the tasks that were involved in a specific retail application and how a certain organization faced and overcame the challenges that accompany building and updating a data warehouse. To accomplish its objectives, this paper is divided into seven sections. The first section looks at the environment and issues surrounding the initial load of the warehouse. The second part looks at the inevitable task of data manipulation, specifically dealing with date values. The third part examines methods for finding out the maximum date value of transactions in the data warehouse. Next, the paper looks at finding the maximum date values in the operational data that feeds the warehouse. The fifth step looks at the method for comparing the maximum date of the warehouse transactions with the maximum date of the operational data. The sixth step looks at doing the actual updating itself. The seventh step looks at accomplishing the above by using the SAS/ACCESS® LIBNAME statement.

Part I: Introduction and Initial Load

Background: A national retail organization recently built a data warehouse that allows it to analyze the purchasing patterns of its customers. The organization wanted to find out who its best customers are so that ways might be devised to reward them, with programs like a frequent buyer's club. It also wanted to find out who its worst customers are, so that it can cease mailing them expensive catalogs on a monthly basis.

This organization has almost 100 stores all across the United States. Every night, each of these stores uploads its transactional data to a regional server. Then, every few days, the sales data is loaded into Sybase tables. The plans are to build a SAS data warehouse (or data mart) from the Sybase tables (figure. 1)

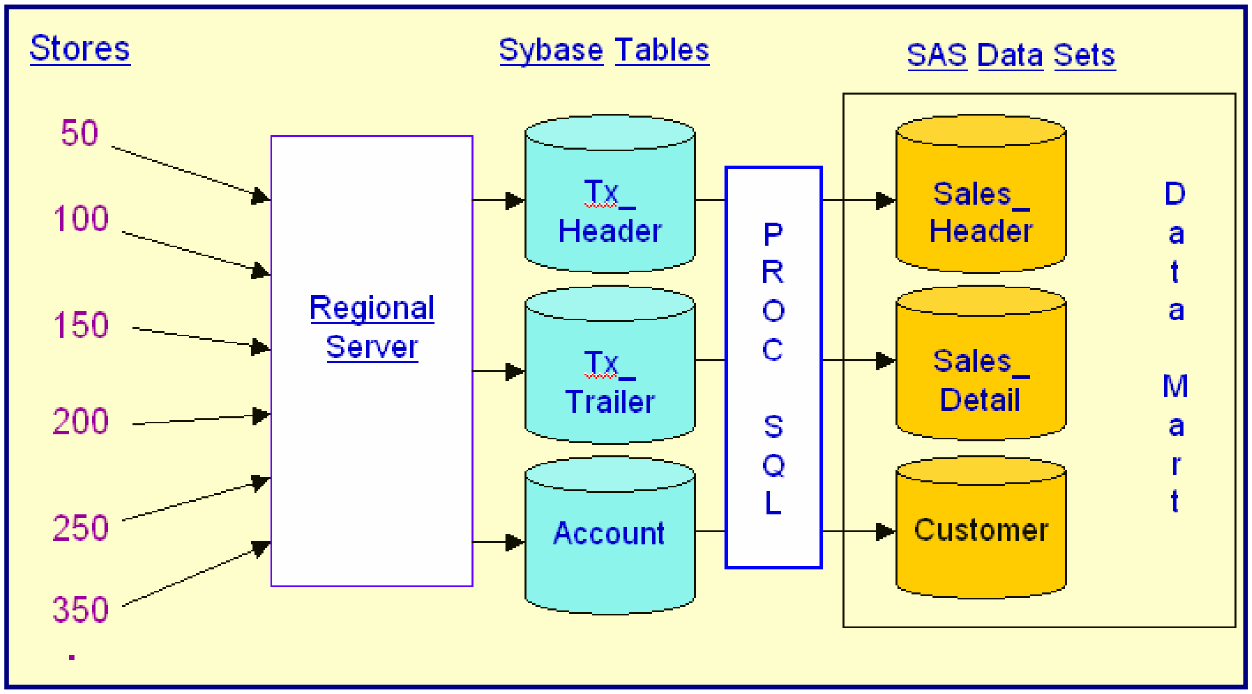
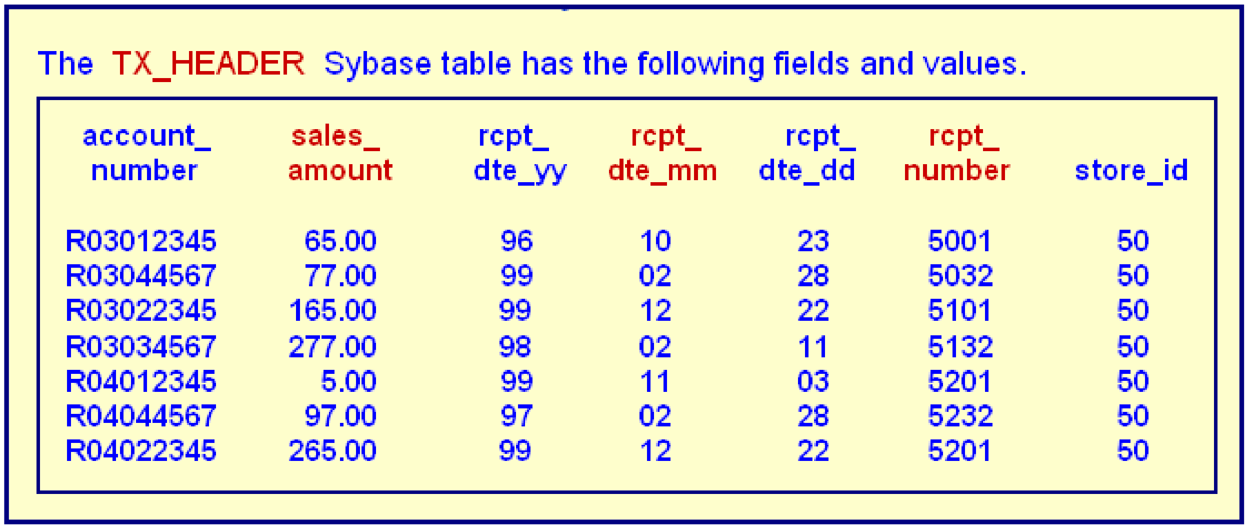


Figure 1.

For the sake of simplicity, this paper will focus only on the loading and updating of the Sales_Header SAS dataset from the Tx_Header Sybase table. Before the code can be written to perform the initial load, the column names of the Sybase tables need to be known. The column names for the Tx_Header Sybase table are shown below (Figure 2). Notice the column names. Especially notice that there are three separate fields for the date of the sale; rcpt_dte_yy, rcpt_dte_mm, and rcpt_dte_dd.



The TX_HEADER Sybase table has the following fields and values.

account_ number	sales_ amount	rcpt_ dte_yy	rcpt_ dte_mm	rcpt_ dte_dd	rcpt_ number	store_id
R03012345	65.00	96	10	23	5001	50
R03044567	77.00	99	02	28	5032	50
R03022345	165.00	99	12	22	5101	50
R03034567	277.00	98	02	11	5132	50
R04012345	5.00	99	11	03	5201	50
R04044567	97.00	97	02	28	5232	50
R04022345	265.00	99	12	22	5201	50

Figure 2. Sybase Data

Steps to Building the Data Mart

The following six steps were taken to build the data mart:

1. Do the initial load,
2. Perform any data manipulation necessary
3. Get the maximum date values from the data mart
4. Get the maximum date values from the DB2 table
5. Compare dates
6. If the maximum Sybase date is greater than the maximum data mart (SAS) date, then refresh the data.

Step 1: Do the Initial Load

Using the SAS/ACCESS Libname statement, the step that performs the initial load looks as follows:

```
libname  s_tables  SYBASE   user = BTC123  password = xxx  database = lwdw11A
            server = lwdw11A   preserve_tab_names=yes ;

proc sql;
      create table sas_3.sales_header as
      select  account_number, sales_amount, rcpt_dte_yy  as  sales_year,
             rcpt_dte_mm  as  sales_month,  rcpt_dte_dd   as   sales_day,
             rcpt_number,       store_id
      from  S_Tables.TX_HEADER  ;
quit;
```

Program 1.

All the rows that were in the TX_Header SYBASE table have now been read into SAS_3.SALES_HEADER. The next step is to perform any data manipulation.

Step 2: Data Manipulation
One example of the type of data that needs to be manipulated is to create a SAS date value from the three independent date columns read from of the SYBASE table. The following DATA step performs step 2.

```
data step_2;
      set sas_3.sales_header:
      sales_date = mdy(sales_mon, sales_day, sales_year);
 run;
```

.Program 2.

There may be many more types of data manipulation necessary at this point. The creation of a SAS date as seen here represents the kind of work that may be needed here.

Steps one and two could be done with a single PROC SQL step as shown next.

```
libname  s_tables  SYBASE   user = BTC123  password = xxx  database = lwdw11A
           server = lwdw11A   preserve_tab_names=yes ;

proc sql;
      create table sas_3.sales_header as
      select  account_number,  sales_amount,  rcpt_number,  store_id,
               mdy(rcpt_dte_mm, rcpt_dte_dd, rcpt_dte_yy) as sales_date
      from  S_Tables.TX_HEADER  ;
quit;
```

Program 3.

So now we have done the initial load and taken care of some data manipulation issues.

Step 3: Find the latest transaction date in the Data Mart

The next step is to find the maximum date and the record count from the data in the data mart. The next program does this by placing the maximum date in a macro variable named **&cutoff.** Also, in this step, the number of observations in the Sales_Header data set are found and placed in a macro variable named **&r_count.**

```
proc sql;
      select  max(sales_date),   count (*)
               into : cutoff,   : r_count
      from  sas_3.Sales_Header ;
quit;
```

Program 4.

Step 4: Find the latest transaction date in the Sybase table
The same approach is taken here by finding the maximum transaction date from the Sybase table and placing it in a
macro variable**, &LastSDate.** This is accomplished by first building a view (WORK.LATEST) that uses the MDY

function to create a SAS date. This value is put into a variable named DATE.   The second SELECT statement finds
the MAX value of DATE and places it into a macro variable named **&LASTSDTE.**

```
libname  s_tables  SYBASE   user = BTC123  password = xxx  database = lwdw11A
            server = lwdw11A   preserve_tab_names=yes ;

proc sql  noprint;
    create view latest as
    select  mdy(rcpt_dte_mm, rcpt_dte_dd, rcpt_dte_yy) as date
    from S_Tables.Tx_Header ;

    select max(date) into : lastsdte
    from latest;
quit;

%put %sysfunc(int(&lastsdte), date9.).;
```

Program 5.

So now we have the latest date from the Data Mart (&CUTOFF) and the latest date from the Sybase table (&LASTSDTE). The next step (Step 5) is to compare these two dates. If &LASTSDTE is **NOT** larger, then it is not time to do anything else.   If &LASTSDTE is larger than &CUTOFF, then it is time to run a program that updates the Data Mart (Step 6).   Both of these steps are run together as seen in the program step below.

Step 5 and 6: Check to see if the Sybase records have been updated

The next two steps are run at the same time and represents an occasion to conditionally execute a step in a program. This is done in a macro program called **refresh .**

```
%macro refresh;

     %if  &lastsdte   >   &cutoff          <----   Step  5

           %then  %do;                             Step  6

                   proc sql;
                          create table sales_header (drop = date) as
                          select *,  mdy(rcpt_dte_mm, rcpt_dte_dd, rcpt_dte_yy) as date
                          from  S_Tables.Tx_Header
                          where calculated date > &cutoff ;
                   quit ;

          %end;

%mend refresh;

%refresh;
```

Program 6.

Step 5 is found in the %IF statement . If the condition is true, then the PROC SQL step (Step 6) is run.   Step 6 creates a data set named WORK.SALES_HEADER. This data set contains only the new records that were placed in the SYBASE table since the last update or the initial load.   Steps 5 and 6 are to be run together every time the data mart is to be updated.

There is actually a step 7 that needs to be performed, which takes the new records and append them to the SAS_3.Sales_Header data set.   This is illustrated by the diagram below.

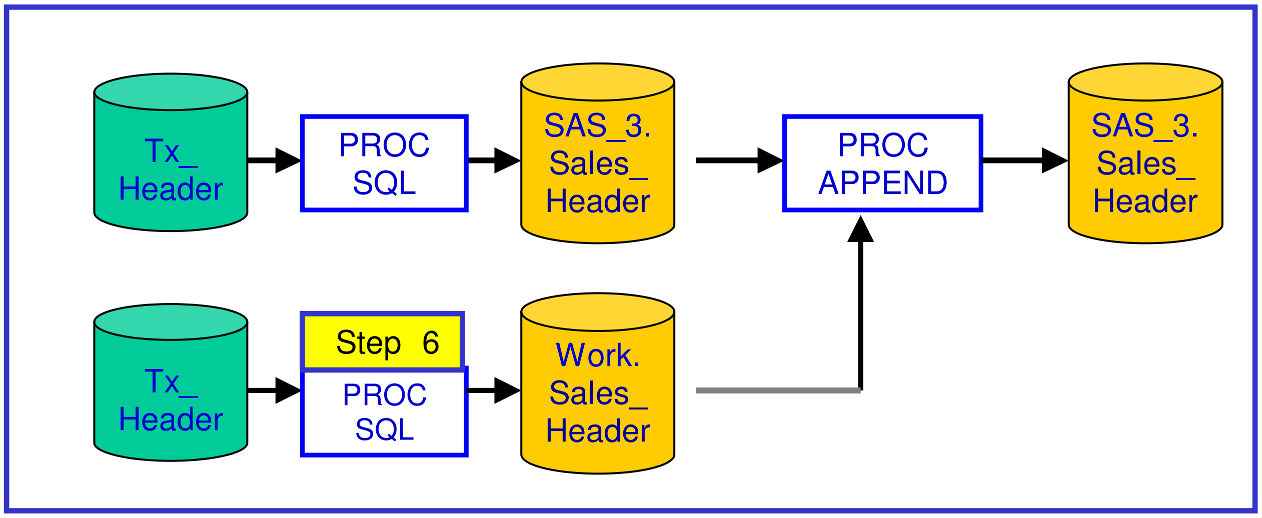


Figure 3.

The PROC APPEND code is shown below.

```
proc append base = sas_3.sales_header
                   data = work.sales_header ;
run;
```

Program 7.

Conclusion
These steps represent the basic work that needs to be done to build and incrementally update the Data Warehouse or Data Mart. There may be other minor steps that need to be done depending on the environment or installation.

Acknowledgments
I would like to acknowledge and greatly thank the Technical Support Department at SAS Institute for their helpful knowledge and expertise that they so freely gave.   I would also like to thank Bob Passmore for his SAS Macro knowledge.   He and I worked together on the project that was the inspiration for this paper. .

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

Ben Cochran
The Bedford Group
Raleigh, NC 27607
Phone: 919.741.0370
Email: bedfordgroup@nc.rr.com