TAQ SAS Programming Issues

In this section, we introduce useful tips and programming advice to develop efficient, fast, and effective processing of TAQ data. Because the TAQ database is, by an order of magnitude, the largest database at WRDS paying attention to program efficiency issues and TAQ data organization is more important than usual (some of the examples below will save hours over the "naïve" implementations users might be tempted to use).

The operational significance of monthly vs. daily datasets

Before reviewing general techniques for improving SAS program performance, it is important to understand how the problem is exacerbated by the "legacy" organization of some TAQ data into MONTHLY datasets (all dataset prior to 2008, as of the writing of this overview).

Each monthly dataset is ordered by SYMBOL DATE TIME, as per this example:

Table 1: Record order in Monthly CT Dataset for Jan 1993
Dataset Name: CT9301

SYMBOL	DATE	TIME	PRICE	SIZE
A	19930104	9:35:25	10.375	5,300
A	19930104	9:36:49	10.375	25,000
. . . .		. . . .		. . . .
A	19930104	15:28:53	10.5	100
A	19930104	15:46:25	10.5	1,000
. . . .		. . . .		. . . .
. . . .		. . . .		. . . .
A	19930129	09:30:51	9.75	1,700
A	19930129	09:40:13	9.875	1,000
. . . .		. . . .		. . . .
. . . .		. . . .		. . . .
A	19930129	15:57:15	10	500
A	19930129	16:02:27	9.875	7,200
. . . .		. . . .		. . . .
. . . .		. . . .		. . . .
. . . .		. . . .		. . . .
F	19930104	9:32:47	43.25	58,200
F	19930104	9:32:50	43.25	100
. . . .		. . . .		. . . .
. . . .		. . . .		. . . .

F	19930129	16:01:03	46.375	30,600
F	19930129	16:01:15	46.125	100
. . . .	. . . .	. . . .		. . . .
. . . .	. . . .	. . . .		. . . .
. . . .	. . . .	. . . .		. . . .
MSFT	19930104	9:18:13	85.5	300
MSFT	19930104	9:19:06	85.375	1,000
. . . .	. . . .	. . . .		. . . .
. . . .	. . . .	. . . .		. . . .
MSFT	19930129	16:33:22	86.25	1,000
MSFT	19930129	16:37:26	86.125	1,500
. . . .	. . . .	. . . .		. . . .
. . . .	. . . .	. . . .		. . . .
. . . .	. . . .	. . . .		. . . .
ZZY	19930129	11:20:17	14.25	200

On the other hand, the daily files, since they each cover only one trading date, are each sorted merely by SYMBOL and TIME, as below:

Table 2: Record order in Daily CT Dataset for Nov 20, 2008
Data set name: CT_20081120

SYMBOL	DATE	TIME	PRICE	SIZE
A	20081120	9:28:25	17.000	100
A	20081120	9:28:25	17	2300
. . . .	. . . .	. . . .		. . . .
. . . .	. . . .	. . . .		. . . .
A	20081120	17:12:28	16.2129	8,247
A	20081120	17:13:00	16.5	100
. . . .	. . . .	. . . .		. . . .
. . . .	. . . .	. . . .		. . . .
F	20081120	4:15:44	1.31	3,500
F	20081120	5:59:00	2.20	500
. . . .	. . . .	. . . .		. . . .
. . . .	. . . .	. . . .		. . . .
F	20081120	19:59:50	1.33	949
F	20081120	19:58:58	1.33	1,300
. . . .	. . . .	. . . .		. . . .
. . . .	. . . .	. . . .		. . . .
MSFT	20081120	7:40:11	18.25	1,000
MSFT	20081120	7:48:58	18.5	100
. . . .	. . . .	. . . .		. . . .

. . . .	. . . .			. . . .
MSFT	20081120	19:58:46	17.75	100
MSFT	20081120	19:58:54	17.75	100
. . . .	. . . .			. . . .
. . . .	. . . .			. . . .
ZZ	20081120	16:21:07	1.8973	2,400

Are daily files 'better' than monthly files? It is more difficult to control the order of data produced from multiple monthly datasets, than the one produced by daily datasets. For example, you can easily read, in one step, any number of daily datasets and produce resulting datasets ordered by DATE/SYMBOL/TIME or by SYMBOL/DATE/TIME monthly datasets, as in the example below:

```
data mydata;
  set  taq.ct_20081119
       taq.ct_20081120
       taq.ct_20081121;
  by symbol date time;
 where SYMBOL in ('IBM','DELL','MSFT','F','USX');  ** Retrieve only these tickers **;
run;
```

The BY statement above tells SAS to read the three datasets in parallel, INTERLEAVING observations from the datasets to produce a SYMBOL/DATE/TIME order in MYDATA. This works only because the three datasets in the SET statement are already each sorted by SYMBOL/TIME (and since each has a constant date, they can each be considered to be ordered by SYMBOL/DATE/TIME -- or by DATE/SYMBOL/TIME).

If you remove the BY statement above (or if you used by DATE SYMBOL TIME), then the SET statement CONCATENATES the three datasets, reading them in sequence rather than in parallel. As a result, MYDATA is in DATE/SYMBOL/TIME order .

With monthly datasets, however, producing results in SYMBOL/DATE/TIME order is easy, but generating DATE/SYMBOL/TIME order is not (in terms of computing time). Consider this example:

```
data mydata;
  set   taq.ct9301
        taq.ct9302;
  by symbol date time;
  where date between ("10jan1993"d and "20feb1993"d) and  SYMBOL in ('IBM','DELL','MSFT','F','USX');
```

As above, this will INTERLEAVE observations from the two monthly datasets producing a SYMBOL/DATE/TIME record order. But if we remove the BY statement, we won't get the

DATE/SYMBOL/ORDER sequence produced in the daily dataset case. Instead the order will be MONTH/SYMBOL/DATE/TIME. The later is probably not a useful data order for some research tasks, and is one of the reasons all TAQ data will be converted to daily datasets.

Some efficieny tips when using taq data

Rule 1: Don't make subsets unnecessarily:

A common inefficiency is something like the following two-step program The first step extracts the subset of interest to the user, and the second (PROC MEANS) step generates a file with the number of trades, minimum trading price, and maximum trading price for each SYMBOL/DATE combination in the user's subset.

```
Data temp (keep=date symbol price);
  set taq.ct9301;
  where symbol in ('IBM',"MSFT","DELL") and time between ('09:30:00't and '16:00:00't);
run;
proc means data=temp n min max   noprint;
  by symbol  date;
  var price;
  output out=myresults   n=n_trades min=min_price max=max_price;
 run;
```

This is wasteful, since it reads the data twice (once for the DATA step and once for PROC MEANS) and writes it once. Instead, if you do the filtering within the PROC MEANS, the data is read only once, and is never written, saving considerable time and computer resources. Using the WHERE statement within the PROC MEANS (or any sas PROC) enables this approach, as below:

```
Proc means data=taq.ct9301 n min max   noprint;
  by symbol date;
  where symbol in ('IBM',"MSFT","DELL") and time between ('09:30:00't and '16:00:00't);
  var price;
  output out=myresults   n=n_trades min=min_price max=max_price;
run;
```

Rule 2: Effective use of data VIEWS.

The WHERE statement within the PROC MEANS has obvious advantages in the example above, but what if you want the proc means to be applied to several datasets at once, or you need to generate additional variables for analysis in the PROC MEANS? Unfortunately you can't directly tell PROC MEANS to read several datasets at once, or to create new variables for analysis. But you CAN tell PROC MEANS to read a data VIEW instead of a data SET, as here:

```
Data v_temp/view=v_temp;
   set    taq.ct_20090102
          taq.ct_20090105
          taq.ct_20090106
  by symbol date;
  where symbol in ('IBM',"MSFT","DELL") and time between ('09:30:00't and '16:00:00't);
  trade_value=price*size;
run;
proc means data=v_temp  n min max sum noprint;
  by symbol date;
  var price size trade_value;
run;
```

While the DATA V_TEMP step looks like it is reading data and creating a new dataset, it is really only creating a view, i.e. a script which ***won't be activated until it is used in the subsequent PROC or DATA step***. As a result, the data is read only once. It is as if the PROC MEANS is interleaving the trade data and creating the new variable on the fly. One caveat: if you use a given data view only once in program, then it is certainly an efficient tool, But the more times you access that view in your program, the less advantage is has over creating and using the analogous data set.

Rule 3: FIRST. and LAST. processing via a BY statement

What if you want to find the first and last price for each symbol in a set of trading days? You can do this by taking advantage of the BY statement, as follows:

```
Data open_close (keep=date symbol price_open price_close);
   set taq.ct2008_1201  taq.ct_20081202;
   by symbol date;
   retain price_open  /*Do not reset this variable to a missing value */;
   if first.date then price_open=price;
   if last.date then do;
     price_close=price;
     output;
   end;
run;
```

Use of the BY statement above tells SAS two things: (1) expect the incoming data to be sorted (and generate an error if it isn't), and (2) for each "BY-variable", generate two temporary dummy variables, indicating whether the record-in-hand is the first or last record for the current value of that BY-variable. As a result, every time a record is the first for a given date, we can assign the value of price to price_open, and when the last record for a given date is encountered, we assign the value for price_close and output the record. The result is one record for each SYMBOL/DATE combination. (The RETAIN statement tells SAS not to reset the value of price_open to a missing value every time a new observations is read. Note that if you only want open and close price within normal trading hours, you can add a

   where time between ’09:30:00’t and ’16:00:00’t:

statement after the by statement.

Rule 4 : DOW loop discussion

The SAS Dow loop procedure takes advantage of the fact that TAQ files are already sorted (for example, monthly files are sorted on symbol, date, and time). The fact that they are already sorted is used in SAS efficient programming. WRDS has a complete full detailed application on this issue (Support tab/Using TAQ Data Efficiently/SAS Dow Loop Approach).

TAQ Macros

Creating a List of Daily TAQ file names

Daily TAQ files only exist on trading days. The key idea behind this macro is to check whether the specific day is a trading day: i.e. being a weekday (weekday function in SAS between 2 and 6), and not a holiday (in other words the dataset should exist). Then, the code appends the needed datasets to open using a SAS set statement, while using the “open=defer” data set statement option, which ensures a sequential processing of your set statement through the where clause.

Sample SAS code that extract defined sample (use where on symbol) from daily data

```
%macro taq_daily_dataset_list(type=ct,begyyyymmdd=,endyyyymmdd=)
  / des="Autogenerated list of needed Daily TAQ datasets";

  %let type=%lowcase(&type);

  /* Get SAS date values for date range endpoints */
  %let begdate = %sysfunc(inputn(&begyyyymmdd,yymmdd8.));
  %let enddate = %sysfunc(inputn(&endyyyymmdd,yymmdd8.));

  %do d=&begdate %to &enddate     /** For each date in the DATE range */;
    %let yyyymmdd=%sysfunc(putn(&d,yymmddn8.));
    /*If the corresponding dataset exists, add it to the list */
     %if %sysfunc(exist(taq.&type._&yyyymmdd)) %then taq.&type._&yyyymmdd;
  %end;
```

```
%mend;

* using this macro;

data my_output;
set %taq_daily_dataset_list(type=ct,begyyyymmdd=20081010,endyyyymmdd=20081014) open=defer;
where symbol in ("MSFT","IBM","DELL")
      and time between "9:40:00"t and "10:15:00"t;
run;
```

Loop Daily Calculations and Append Results

If you intend to use both TAQ Trades and Quotes datasets, and merge to create inferences on buyer or seller initiated trades for example (Lee and Ready algorithm in our Support Page), we strongly advise you to follow an algorithm similar to the code below. Such algorithm ensures that the computation and matching are processed at the daily units, which generate a smaller processed output dataset, to be appended to existing output, while freeing the memory and disk space usage after each processing iteration (trading day iteration).

Here is the sample SAS code that merges Daily Trades and Quotes Datasets

```
%macro w_taq_analyze(fdate,ldate);
%let fdated = %sysfunc(InputN(&fdate,yymmdd8.));
%let ldated = %sysfunc(InputN(&ldate,yymmdd8.));
%let taq_files= ;
%do date = &fdated %to &ldated;
     %let ndate= %sysfunc(PutN(&date,yymmddn8));
     %let wkday= %sysfunc(weekday(&date));
     %let dct = taq.ct_&ndate. ; /* TRADES */
     %let dcq = taq.cq_&ndate. ; /* QUOTES */
     %if %index(2 3 4 5 6,&wkday)>0
     %then %if %sysfunc(exist(&dct))
     %then
     %do;
        /* User's Code for &dct &/or &dcq                       */
        /*     ->  merging, processing, and new var computation */
        /*     ->  output dataset, &dout                        */
     %end;
%if %sysfunc(exist(&dout))%then
  %do;
   proc append base=user_out_set data=&dout; run;
   proc sql; drop table &dout; quit;
  %end;
%end;
%mend;
```

WRDS Research
October 22, 2009