Paper SAS2802-2016

Dealing with Nanoseconds in SAS® Datetime Values in Transaction Processing

Richard D. Langston, SAS Institute Inc.

ABSTRACT

This paper describes a technique for dealing with the precision problems inherent in datetime values containing nanosecond data. Floating-point values cannot store sufficient precision for this, and this limitation can be a problem for transactional data where nanoseconds are pertinent. Methods discussed include separation of variables and using the special GROUPFORMAT feature of the BY statement with MERGE in the DATA step.

INTRODUCTION

It's hard to believe that nanoseconds matter. Not too many years ago the term didn't even have meaning. A billionth of a second. But now there is significance. There are so many transactions happening around the world that billionths of seconds might not even separate them.

I was contacted by a new SAS® user who was facing the problem of handling datetime values where nanoseconds were a factor. The problem is that a SAS datetime value is the number of seconds since January 1, 1960. So a datetime value at midnight, January 1, 2016, is 1,767,225,600 seconds. That value already needs 9 digits of precision without any fractions. For billionths of seconds, eight additional digits are needed, for a total of seventeen digits. But a floating-point value cannot consistently store complete data values for as many as seventeen digits, so it is not possible to use a SAS datetime value for billionths of seconds.

A different approach was needed in order to ensure that billionths of seconds could be represented. And the user would further need to group transactions into different "buckets," which might be based on milliseconds (thousandths of seconds) or microseconds (millionths of seconds) or some other interval value. Further, the maximum value for a bucket needed to be matched against all observations in the bucket to determine all that met the maximum.

USING A SEPARATE VARIABLE

My first suggestion was to use a separate variable for billionths of seconds, and have it represent the number of nanoseconds instead of a fraction.

Consider this generated comma-separated test data as input:

```
2015-03-03 23:00:00,123456101,t1 ,38
2015-03-03 23:00:00,123456201,t2 ,12
2015-03-03 23:00:00,123456301,t3 ,5
2015-03-03 23:00:00,123456401,t4 ,9
2015-03-03 23:00:00,123456501,t5 ,29
2015-03-03 23:00:00,123457101,t6 ,30
2015-03-03 23:00:00,123457201,t7 ,43
2015-03-03 23:00:00,123457301,t8 ,33
2015-03-03 23:00:00,123457401,t9 ,43
2015-03-03 23:00:00,123457501,t10 ,5
2015-03-03 23:00:00,123458101,t11 ,24
2015-03-03 23:00:00,123458201,t12 ,48
2015-03-03 23:00:00,123458301,t13 ,47
2015-03-03 23:00:00,123458401,t14 ,11
2015-03-03 23:00:00,123458501,t15 ,35
```

The following code reads in the datetime value, followed by the number of billionths of seconds, a label, and a value:

```
data ts1; infile mydata dlm=',';
     input datetime:ymddttm30. ns label $ value;
     date = datepart(datetime);
     time = timepart(datetime);
     format date e8601da. time e8601tm. ;
     drop datetime;
     run;
proc sort; by date time ns; run;
```

USING THE GROUPFORMAT OPTION WITH PICTURE FORMATS

In order to allow for groupings based on different kinds of intervals, we introduce PICTURE formats with different multipliers for that purpose:

```
proc format;
     picture bound1d other='9'        (mult=1e-8);
     picture bound2d other='99'       (mult=1e-7);
     picture bound3d other='999'      (mult=1e-6); /* milliseconds */
     picture bound4d other='9999'     (mult=1e-5);
     picture bound5d other='99999'    (mult=1e-4);
     picture bound6d other='999999'   (mult=1e-3); /* microseconds */
     picture bound7d other='9999999'  (mult=1e-2);
     picture bound8d other='99999999' (mult=1e-1);
     run;
```

So, for example, if we want to show the number of microseconds, we can apply the BOUND6D format. This will multiply the billionths of seconds by 1E-3 (that is, divide by 1000) so that the formatted result will be the integer value that is the number of microseconds. Our first data value of 123456101 nanoseconds becomes 123456 microseconds. Any nanosecond from 123456000 to 123456999 will fall into this "bucket" of 123456 microseconds.

We are interested in determining the maximum value for our “value” variable for a given interval, and displaying information for all transactions that contain that value. The value can be computed by the MEANS procedure, and we will want to merge the value back in to show results. However, our BY values need to be our formatted interval. To do this, we can use the GROUPFORMAT option on the BY statement for the DATA step. Note that although procedures such as MEANS separate their BY groups by formatted values, the DATA step does not do this by default, so the GROUPFORMAT option is necessary to ensure the proper grouping.

Consider this macro that will demonstrate this process:

```
%macro doit(intervalfmt);
 /* Get our maximum value for each grouping */
proc means data=ts1 noprint;
     var value;
     output out=new max=maxvalue;
     by date time ns;
     format ns &intervalfmt;
     run;

 /* See what they are */
data _null_; set new;
```

```
        put ns=15. ns=&intervalfmt maxvalue=;
        run;

     /* run through our data, using the max value as we see fit */
   data ts3; merge ts1 new; by date time ns groupformat;
        format ns &intervalfmt;
        if value=maxvalue then put date= time= ns=15. label= value=;
        run;
   %mend;

   /* Try it with microsecond boundaries */
   %doit(bound6d.);
```

We see that PROC MEANS has computed the following values, showing nanoseconds both without and with our bucketing for microseconds:

```
ns=123456101 ns=123456 maxvalue=38
ns=123457101 ns=123457 maxvalue=43
ns=123458101 ns=123458 maxvalue=48
```

An important point to make here: the ns value written out by PROC MEANS is not necessarily the actual ns value corresponding to the observation containing the maximum value. For example, the third observation shown has ns=123458101, but in the original data, we see that value=48 for ns=123458201. PROC MEANS is choosing the first value it encountered for the BY group, since GROUPFORMAT is the default grouping mechanism for PROC MEANS. This will not cause us problems with the MERGE because we are also using GROUPFORMAT.

Indeed, the results of our MERGE are as follows:

```
date=2015-03-03 time=23:00:00 ns=123456101 label=t1 value=38
date=2015-03-03 time=23:00:00 ns=123457201 label=t7 value=43
date=2015-03-03 time=23:00:00 ns=123457401 label=t9 value=43
date=2015-03-03 time=23:00:00 ns=123458201 label=t12 value=48
```

Note that the MERGE will show us all observations that have a match with the maximum value.

USING THE GROUPFORMAT OPTION WITH FUNCTION-AS-LABEL FORMATS

Here we can change the type of format we use to a function-as-label format, using the FCMP procedure to define the function. This is advantageous if the bucketing process is more complicated than a simple multiplication. In this example, we still use a basic ratio of our value divided by a constant.

```
%macro create_format(interval);

proc fcmp outlib=work.myfncs.intvl;
function compute_bucket(x);
  val =  floor( x / &interval);
  return (val);
endsub;
run;

options cmplib=work.myfncs;
```

```
proc format;
     value compute_bucket(default=14) other=[compute_bucket()];
     run;

%mend create_format;
```

We still have a format as before, just not a PICTURE format. We can still apply our macro and get the same results with a factor of 1000 (to obtain microseconds from nanoseconds).

```
%doit(compute_bucket.);
```

So now if we need to make our bucketing algorithm more complex, it can be easily introduced into the COMPUTE_BUCKET function via PROC FCMP.

CONCLUSION

With this approach, the user can allow for "nanosecond bucketing" to group observations as needed, without the limitations of floating-point precision. Using separate variables with the GROUPFORMAT option, and paying special attention to the formats, you can achieve successful bucketing.

CONTACT INFORMATION

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

Rick Langston
SAS Institute Inc.
rick.langston@sas.com