SESUG 2022 Paper #106

Working with large decimals: how precise is my data with SAS?

Osmel Brito Bigott, Yenireth Gil, and María Victoria Daboín, DatAnalítica S.R.L.

ABSTRACT

SAS is a powerful tool for statistical analysis and calculation that makes easier when it comes to manipulating and reporting our data. However, what they have never told us about is what inconveniences can arise when it comes to having to be very precise in the quality of the numerical value of a calculation. In this sense, it is important to understand the ideal way to store numerical data when having an information repository in other storage systems (databases or files) other than SAS. In a project where we used Teradata to store results of historical statistical analysis in SAS we had faced with the question: How precise were numerical values in SAS to store decimals? And how should we upload them to a Tersdata Data Warehouse? It turns out that this answer depends a lot on the operating system where SAS is installed (Windows, Linux). In this paper we want to show the findings we made about the precision of numerical values and how we were able to solve the problem faced.

INTRODUCTION

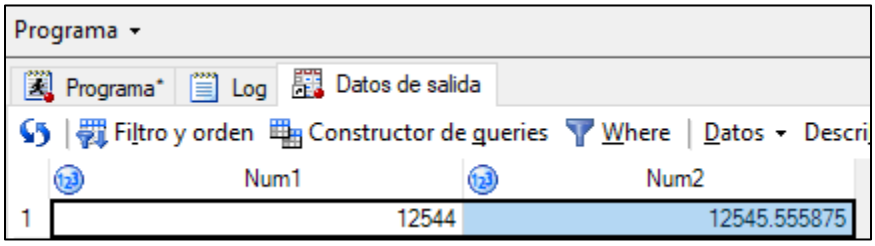
The treatment of numbers is often ignored, since we tend to visualize data in the simplest way. When we talk about numerical figures associated with monetary expressions, it is sufficient for us to visualize in thousands or millions and always trying to use a maximum of two decimals, but not in all cases can we simplify the numbers so that we ignore the entire mantissa of the number. Our aim in this paper is to demonstrate to our readers how SAS stores and treats long decimals.

DECIMAL PRECISION IN SAS

SAS stores numerical values in a minimum of 3 bytes and a maximum of 8 bytes, depending on the precision of the data. The following code, for example, illustrates a fractionally large number in decimal representation.

```
data Aux;
      length Num1 3. Num2 8.;
      Num1=12545.555875145893;
      Num2=12545.555875145893;
run;
```

It is important to note that each variable uses the same number, but uses different lengths. The following image shows the result of creating a temporary table using both types of lengths in a SAS installation on Linux.



Output 1. Uso de dos longitudes para una variable numérica en UNIX

In the previous image it is clear that when we use 3 bytes for a numerical variable we have two things to take into account 1) The value only reserves the entire part of the number but not as it originally appears in the code but is rounded by the nearest lower number 2) The decimal part is completely removed from the display of the number.

It is important to note that, to some extent, the number assigned in the code is preserved, but the totality of decimals is not shown. Surely, using a format, we can observe the totality of the number, but replicate the code example below where the maximum number of positions is used..

```
data Aux;
    length Num1 3. Num2 8.;
    format Num1 Num2 32.31;
    Num1=12545.555875145893;
    Num2=12545.555875145893;
run;
```

SAS has a maximum limit on how many values it will show us, even though we explicitly indicate that it will show as many decimals as possible..

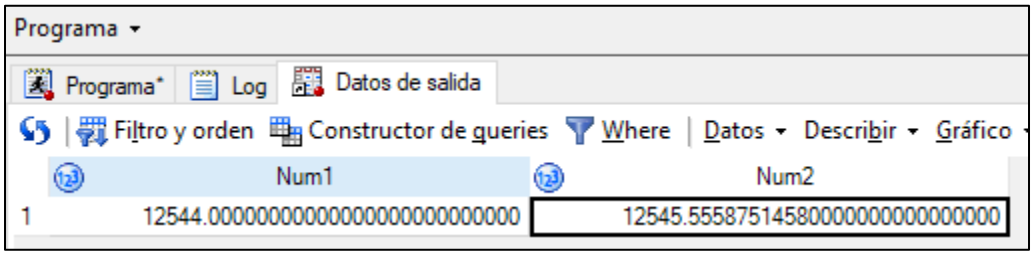


Figure 1 - Using formats for a numeric variable in Linux

The following table we will make a representation of positions to see how much SAS shows us

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	2	3	4	5	.	5	5	5	8	7	5	1	4	5	8	9	3		

Table 1 - Decimal precision displayed in tables with formats

The variable was truncated in 3 bytes, but 8 bytes only showed us up to 16 positions including the decimal, even indicating the format 32.2. This is because visually they are limited to 16 significant digits, but what happens if we use 8 bytes and separate the whole part and decimal.

```
data Aux;
      length Num2 8.;
      format Num2 Decimal Integer 32.31;
      Num2=12545.555875145893;
      Decimal=Num2-int(Num2);
      Integer=int(Num2);
run;
```

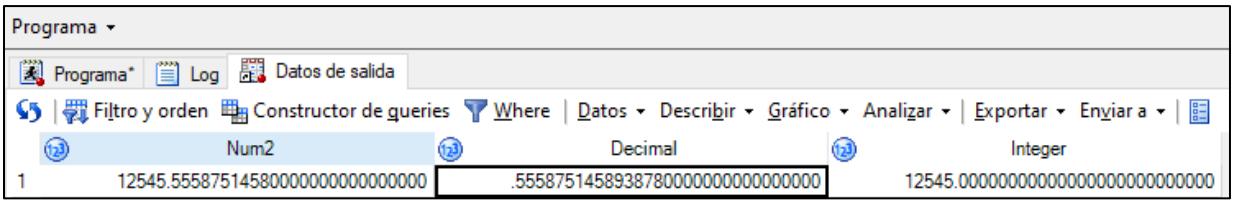


Figure 2 - Numeric variable's integer and decimal parts on UNIX

In the decimal column, we notice that there are more decimals than the original value in the code 12545.555875145893 has added us 3 decimal places that aren't included in the explicit value. Let's graph them in a table to see what happened.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
.	5	5	5	8	7	5	1	4	8	9	3	8	7	8					

Table 2 - Decimal precision displayed in table

It happens that each real number does not have an exact binary representation, which means that the number itself brings a rounding error in its storage and also the SAS formats include some rounding that is not exactly explicit, however, in this case there are no rounding terms, but we see more decimals than those originally expressed in the number. If we wanted to demonstrate what would be the error that we could find in a number of that precision it would be enough to find the binary representation closest to the value we are working on that in this case would be between 2^{13} = 8192 and 2^{14} = 16384, starting from this concept we are located in the minimum detectable representation for a number of 8 bytes 2-52 and the exponent 52 we subtract the smallest exponent of the two detectable binary representations $2^{-(52-13)} = 2^{-39}$ which leaves us a rounding error $1.818989E^{-12}$ which means that we will have detectable errors in decimal E^{-12} when we are working on a precision figure `12545.555875145893`, which in the example of table 1 corresponds to position 13.

WHAT IS THE BEST WAY TO STORE NUMERICAL DATA FROM SAS TO A DATABASE?

Because we found that, despite using formats and trying to achieve the greatest possible accuracy, we will always encounter some failure in rounding or truncating of data, Several tests were conducted to discover how to display the largest number of numerical positions in a variable in a visual manner.

1. You run the risk of the data being truncated or rounded to decimal places if you format a number in a way that contains as many decimal places as possible.
2. As long as the number is formatted with the maximum possible decimal places and sent as text, we have the same problem.
3. When we take the integer part of the number and the decimal part separately to the highest precision of a format and then transform and join the two together as text, at least the greatest number of numerical positions will be preserved with the error caused by binary representation, but the error will only affect negligible decimal positions, which would not affect economic indicators or rate calculations.

CONCLUSION

We wanted to show in this paper the problems we faced manipulating numeric fields in a SAS deployment used for consolidation and reporting of financial data having a Teradata database as a final repository.

RECOMMENDED READING

- *Base SAS® Procedures Guide*
- *SAS® For Dummies®*

CONTACT INFORMATION

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

DatAnalitica
7801 NW 37TH ST
DO80Q75172N
DORAL,
FLORIDA 33195-6503