Paper 2581-2014

How SAS® and Teradata® Dramatically Improve Data Analysis at CMS

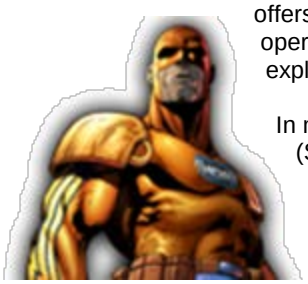
Rick Andrews, Centers for Medicare and Medicaid Services (CMS), Office of the Actuary

ABSTRACT

SAS offers advanced analytics while Teradata has developed one of the fastest databases known to mankind. The Office of the Actuary at CMS uses SAS and Teradata to perform many important tasks such as setting Medicare Advantage rates for providers, forecasting the cost of closing the Part-D Donut Hole, and estimating the future cost of healthcare in America. The major advantage to the Teradata platform is the speed at which data can be summarized as compared to legacy systems (billions of rows of data can be summarized in seconds where it once took days) and the SAS system provides many options for accessing and analyzing this data - whether you're on a mainframe, Windows®, or Unix through SAS® Enterprise Guide® or SAS® Enterprise Business Intelligence®.

INTRODUCTION

This paper will explain why using SAS in conjunction with Teradata makes it easier, faster, and more efficient to access Fee-for -Service (FFS) claims from the National Claims History (NCH) using the Integrated Data Repository (IDR). Legacy systems can be quite slow when compared to the speed of this new database, while the SAS System offers various mechanisms for accessing the data from any operating system. The paper will also compare implicit versus explicit pass-through queries using SAS Enterprise Guide.



In many ways, both companies are much like Stark Industries (SAS/Teradata 2013). Tony Stark utilizes his Avengers buddies Captain America, The Hulk and Thor to protect the world. Each superhero offers a unique skill to further the goal of the organization. SAS and Teradata co-created the League of Analytic Heroes, including Dr. Insight, Illumino, Megavox, Magnacomm, Predicta and Exploris to help the world via analytic innovation. Both companies are transforming the world and when working together, they achieve amazing results. Stark Industries develops advanced defense technologies, while SAS and Teradata offer advanced analytics integrated with data management capabilities to many organizations across all industries to improve the speed and versatility of analyzing data.

All three companies are led by innovative thinkers. CEO Tony Stark designed and implemented his powered suit of armor which empowers his body - and protects the world. Likewise, the CEO's of SAS and Teradata lead the creation of the Power to Know® which establishes advanced analytics to optimize the use of data throughout the organization. Interestingly, the SAS campus serves as headquarters for Stark Industries in the hit movie, Iron Man 3.

SPEED

By using a Massively Parallel Processing (MPP) Teradata platform running shared nothing architecture, the system can utilize the latest advances in data processing technologies. In computer architecture, the designation MPP refers to a computing system with many independent units or entire microprocessors that run in parallel. Within this class of computing, all units of processing elements are connected to become one very large computer. This is in contrast to the distributed computing system whereby a massive number of separate computers are used to solve a single problem. In shared nothing architecture, a symmetric multiprocessing node is independent and self-sufficient, and no single point of contention exists across the entire system.

VERSATILITY

The major strength of SAS has always been its statistical prowess, though year after year the SAS System has become more versatile, and it is now easy r to access and analyze data across the *spectrum of operating environments. The SAS* System originated on a mainframe, but tools like Enterprise Guide make accessing data a seamless proposition, whether information is stored on a mainframe, Windows®, or UNIX environment. Tools in the Enterprise Business Intelligence suite of products make it possible to access SAS procedures through web based applications and Microsoft Office® with little or no knowledge of SAS. As technologies have changed so has the SAS Institute's approach to data access and data manipulation techniques.

NATIONAL CLAIMS HISTORY (NCH)

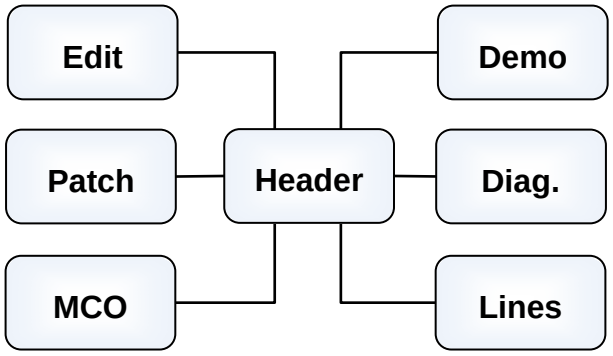
The "System of Record" for fee-for-service (FFS) claims at CMS is the National Claims History (NCH), which is located on a mainframe within the CMS Data Center. The benefits of using the NCH "flat" files is that: (1) there is no need to join a bunch of tables together as all of the associated "trailers" exist on one record for each claim; (2) the universe of claims typically used for analysis at CMS has been determined to be more than 99% complete.

However, due to historical technology limitations, the files are saved in a variable length format, typically on tape data sets and are separated by beneficiary state of residence, as defined by the Social Security Administration (SSA). They are also segmented by time, either inside the 18-month window or out, which are claims processed thru June of the following year. The major issues caused by using a variable length format include: (1) all of the values on each claim record must be read into memory, causing a huge amount of input/output (I/O) to occur when only a small amount of the information may be needed for the analysis and (2) the files must be retrieved from tape, which significantly slows down processing.

Example of NCH Physician/Carrier Record Layout													
Claim #	**Header**	**Trailers** (Variable Length)											
1	Fixed Portion	Edit		Patch		MCO	Demo		Diagnosis			Claim Lines	
2	Fixed Portion	Edit	Patch		MCO	Demo		Diagnosis			Claim Lines		

VARIABLE LENGTH STRUCTURE

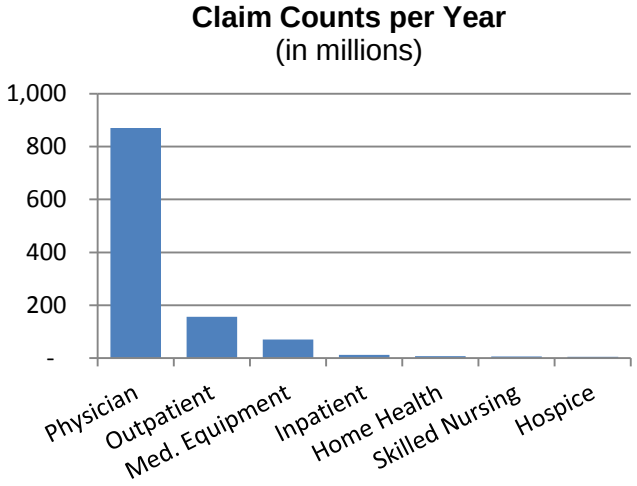
The variable length structure of the files is one reason why many legacy processes read all of the data from each record (claim}. It is a programming challenge to keep track of where the pointer should be when reading this data. Therefore, it is often easier to read all of the information so that incorrect results do not occur. The example of NCH Carrier/ Physician Record Layout above depicts how the records vary in length. If any of the data in the trailer portion of the claim are needed, it is safer to read the entire record. When this data is transposed into more of a database structure it potentially becomes more efficient to process because the system only reads the data requested by the query. One of the major issues with a database structure is the necessity to learn how the often highly normalized tables are related to one another for join operations.



FFS CLAIM COUNTS

FFS claims counts total approximately 1 billion records per calendar year. This can make processing the data very time consuming. To put the size of these files into perspective, imagine summarizing 1,000 spreadsheets that all contain 1 million rows of data, with every column filled with information.

Out of the seven claim types that exist, the Physician (a.k.a. Carrier) data make up over 75% of the total, over 800 million records. They are certainly not the largest files in the world, though can take quite a while to process if every byte of data must be read to obtain information found on the claim line trailer.



INTEGRATED DATA REPOSITORY (IDR)

The IDR is a database created by CMS to "integrate" the various data that exists across the Agency into a central location for ease of use. Information from FFS claims, Part-D Prescription Drug Events (PDE), the Enrollment Database (EDB), Provider of Service (POS), the Medicare Advantage and Prescription Drug System (MARx) and many other data files, were loaded into the database and linked together via a common structure. See https://www.cms.gov/apps/acronyms/ for additional information. This allows users to merge and utilize data across multiple systems within the CMS Data Center – an innovative effort that has been attempted, but never before successfully implemented at CMS.

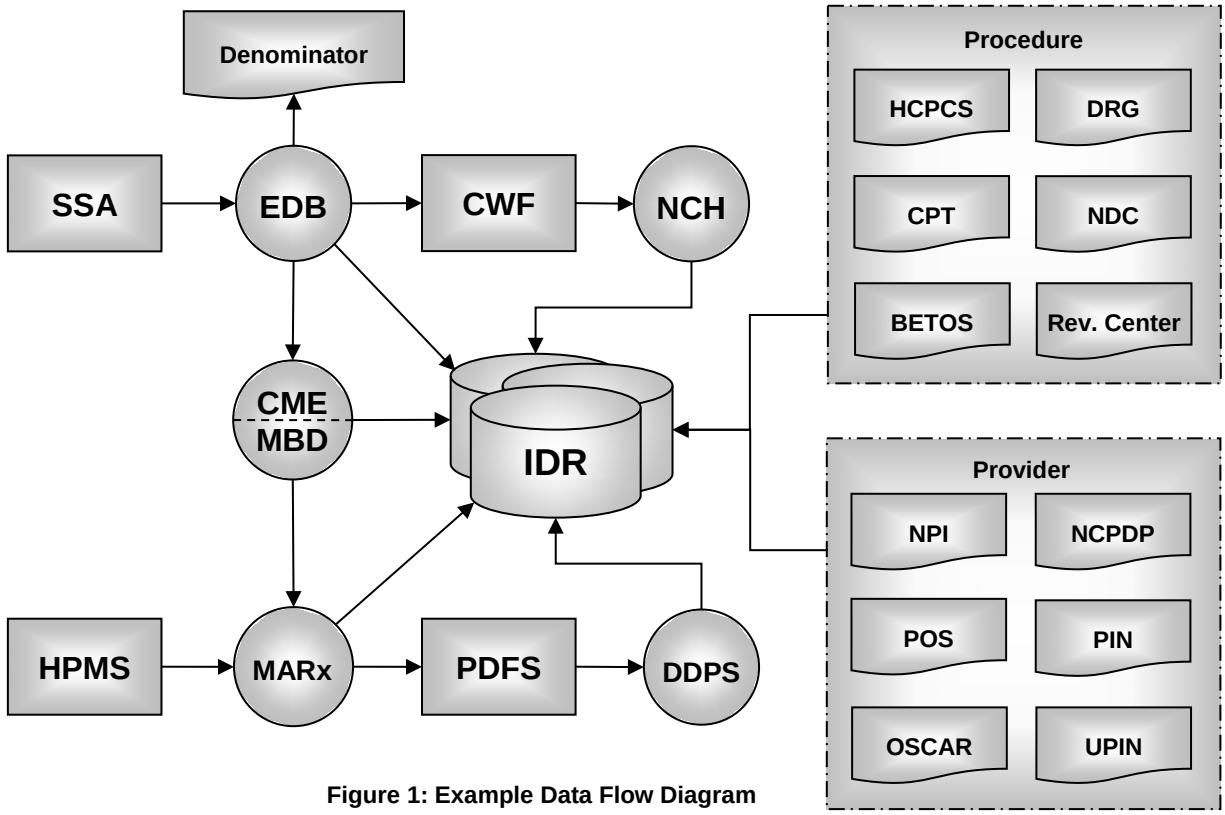
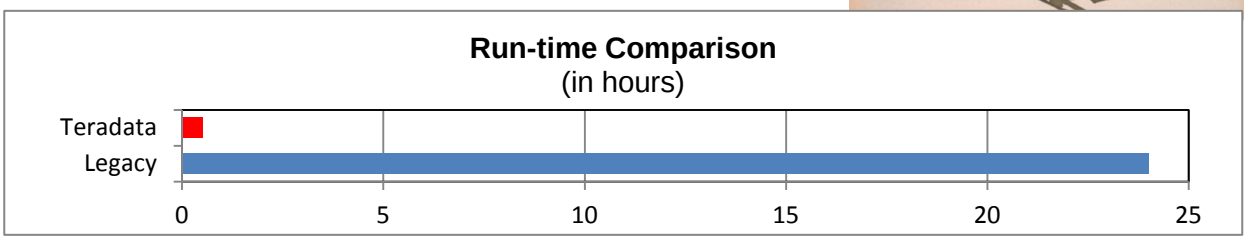


Figure 1: Example Data Flow Diagram

PROCESSING TIMES

Successfully implementing the IDR environment would likely result in much faster processing times. It is certainly hard to come up with a true apple to apples comparison when processing on the different computing systems. And a statement that the IDR is faster than the CMS mainframe is not necessarily an accurate one because it depends on the circumstances. But one thing that can be said is that it usually takes more than 24 hours to turn around one year of FFS claims data from the NCH, whereas the same results can be produced in less than 30 minutes on the IDR. Bottom line is, it's FAST! It's not even like night and day; it's more like a really cloudy night versus a bright sunny day.



ENTERPRISE ARCHITECTURE

Figure 6 depicts the flexibility of the SAS system to access the IDR. Whether connecting to the data using a personal computer, through a mainframe channel or an Enterprise Business Intelligence tool, SAS offers a mechanism to summarize data depending on the specific needs of the organization.

CMS uses all three tools in order to link legacy systems of data across platforms as well as end-user data downloaded from other agencies like the Census Bureau and the National Institutes of Health.

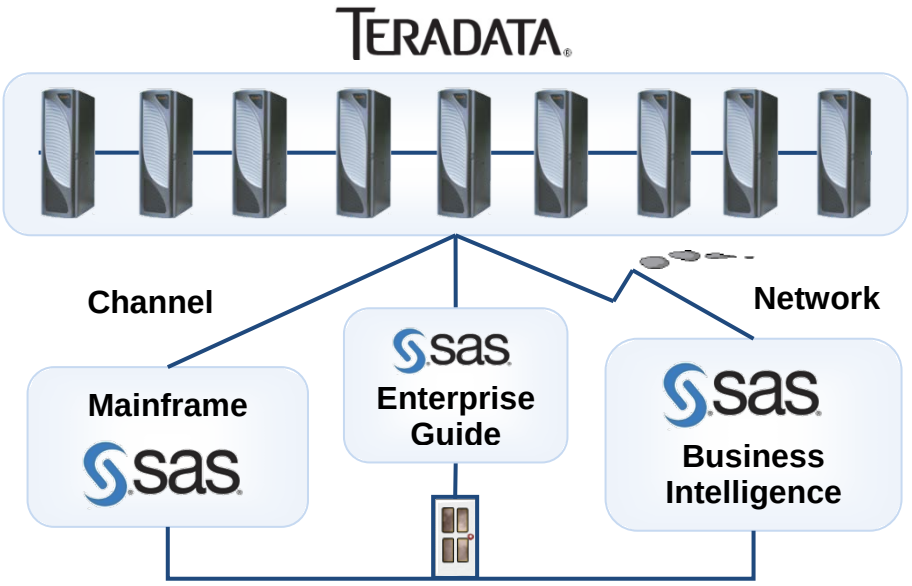


Figure 6: Enterprise Architecture Example

MAINFRAME (MF)

According to the Bible, "In the beginning, God made the Earth and the heavens", while according to the doctrine of large scale data processing, in the beginning, IBM® created the mainframe (Big Blue, as it is often referred to), which has been entrenched at CMS for many years. It is quite difficult to migrate away from systems that have been in place for more than twenty years. This is why the mainframe channel that exists between the IDR and Big Blue is so very important at CMS. Data can be seamlessly uploaded or downloaded to and from each platform programmatically, allowing for the end-user community to combine new world technology with legacy applications.

The SAS System offers a product called, SAS/Access to Teradata, which offers options to data users allows them to access the data by sending a request to the database implicitly or explicitly. An implicit query is one where SAS attempts to convert traditional SAS programming into SQL to be used by the database. This is very useful, although it can be problematic if used incorrectly as will be demonstrated later. An explicit query is one where the user sends a pass-thru query to Teradata for processing. This method removes the possibility of error with the interpretation of programming code, though knowledge of SQL specifically related to Teradata is required.

EXPLICIT EXAMPLE FOR (MAINFRAME)

1 The first thing that happens in a pass-thru query is the CONNECT statement. Notice the query is connecting to TERADATA and that various parameters are included, which are used only as examples. These parameters may or may not be required in every instance of a Teradata database, which will be demonstrated later.

2 The second step identifies the output data set to create in SAS. Notice in the FROM clause, CONNECTION TO TERADATA, which sends the pass-thru query via a nested table expression (a.k.a. a derived table).

3 The last step is the actual Teradata query that is "passed" thru to the database. The use of ANSI standard SQL is recommended so that queries are compatible across platforms and applications.

```
PROC SQL;
  CONNECT TO TERADATA (
    USER= &SYSUID
    MODE= TERADATA                    1
    TDPID= System_ID
    DATABASE= Database_Name );

  CREATE TABLE Work.SAS_Data_Set AS
  SELECT *
  FROM CONNECTION TO TERADATA          2
  (
   SELECT This, And, That
   FROM Teradata_Table
                                       3
  );

  DISCONNECT FROM TERADATA;
QUIT;
```

IMPLICIT EXAMPLE FOR (MAINFRAME)

```
LIBNAME Sandbox TERADATA
  USER= &SYSUID                    (1)
  TDPID= System_ID
  DATABASE= Database_Name;

DATA WORK.MY_HCPCS_TABLE;
  HCPCS = 'J1815'; OUTPUT;         (2)
  HCPCS = 'J1817'; OUTPUT;
RUN;

PROC DELETE
  DATA= Sandbox.MY_HCPCS_TABLE;
RUN;                               (3)

DATA Sandbox.MY_HCPCS_TABLE (
  DBCREATE_TABLE_OPTS=
    'PRIMARY INDEX(HCPCS)' );      (4)
  SET WORK.MY_HCPCS_TABLE;
RUN;
```

(1) The first thing that happens in an implicit pass-thru environment is the submission of a LIBNAME statement. Notice the query is connecting to TERADATA and that various parameters are included. These parameters may or may not be required for every instance of a Teradata database.

(2) In this example, a SAS data set is being created that contains HCPCS codes that are to be loaded into a Teradata table.

(3) If a Teradata table already exists, the SAS program will error out. The PROC DELETE ensures the table does not exist.

(4) The final step in this example is to load the SAS data set into a Teradata table. It is very important to create a primary index when loading data. This will cause the system to "hash" the information across all nodes in lieu of just one. If the primary index is not included all of the data will be loaded onto one and may very easily run out of available disk space. The primary index command is surrounded by quotes and sent to the database via data set option, DBCREATE_TABLE_OPTS.

SAS® ENTERPRISE GUIDE®

This point and click environment is quickly becoming the de facto standard for the research community. It provides a graphical user interface that is relatively easy to learn and use. It contains drag and drop features that are used in conjunction with "nodes" that are reasonably straight-forward and simple to utilize. Another nice feature is that traditional SAS programming can also be employed using the "program" node for traditional programmers.

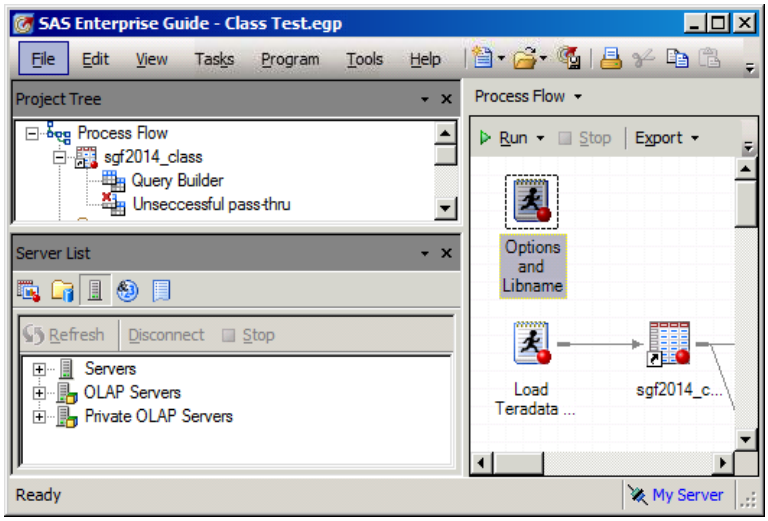


Figure 7: Enterprise Guide Example

DIFFERENCES IN CONNTECT STATEMENTS

```
PROC SQL;
  CONNECT TO TERADATA (
    AUTHDOMAIN= 'Domain_ID'
    CONNECTION= GLOBAL             (1)
    DATABASE= Database_Name );

  CREATE TABLE SASOUT.SAS_Data_Set AS
  SELECT *
  FROM CONNECTION TO TERADATA
  (
   SELECT This, And, That
   FROM Teradata_Table
  );

  DISCONNECT FROM TERADATA;
QUIT;
```

(1) The point here is that the pass-thru query used on the mainframe is exactly the same when used in SAS® Enterprise Guide® with the exception of the CONNECT statement. Since the query is being sent from a different environment on another operating system, the options might be slightly different, though the query is exactly the same.

The nice thing about using an explicit pass-thru is the code can be stored on any standard file server and then used on any operating system. If the mainframe is down for maintenance, this version can be used. This makes the query very portable across systems and can also be used via the web-based tools of SAS® Enterprise Business Intelligence®.

LOADING SAS DATA TO TERADATA

A nice feature of this method is that once a LIBNAME statement is submitted against a TERADATA table all of the point and click features are then available for use. This can make it very easy for a new user to take advantage of the reports that can easily be generated.

In the same breath, watch out for that evil villain, Frackture, who will make decisions on your behalf that might not be the most efficient. One challenging feature is that it can often be a little *too* easy to get an answer that is not necessarily accurate.


```
LIBNAME Sandbox TERADATA
  USER= &SYSUID
  TDPID= System_ID
  DATABASE= Database_Name;

PROC SQL;
  CREATE TABLE Work.SAS_Data_Set AS
  SELECT This, And, That
  FROM Sandbox.Teradata_Table;
QUIT;

DATA Work.SAS_Data_Set;
  SET Sandbox.Teradata_Table;
  KEEP This And That;
RUN;
```

As with any point-and-click environment, be cautiously optimistic when creating new processes. In the previous "IMPLICIT EXAMPLE FOR MAINFRAME" the ability to load SAS data directly to a Teradata database was demonstrated as a very useful tool when performing data analysis at CMS. Although many SAS functions are currently supported in Teradata, there are some unrecognized functions that, if encountered, may cause every record in the result set to be returned to SAS for analysis. This can cause a tremendous amount of I/O, which can be very time consuming.

These implicit options are suggested when developing queries for the first time or if additional documentation is desired. Information is sent to the SAS log that can be used to determine if a query will be sent to the database for processing of if SAS must pull data out of the database for processing.

```
OPTIONS
  MSGLEVEL=I              /* Writes informative messages to the log */
  SASTRACE=',,,ds'        /* Generate code trace for PROC SQL steps */
  SASTRACELOC=Saslog no$stsuffix  /* Writes a trace to the SAS log */
  DBIDIRECTEXEC           /* SQL is passed directly to the database */
  SQLGENERATION=DBMS      /* SAS PROCs generate SQL for in-database */
  SQL_IP_TRACE=SOURCE;    /* Writes major SQL commands to SAS log   */
```

This is an example of how to determine if a query is being performed in-database. The SAS data set called CLASS from the SASHELP library is loaded into a Teradata table then a PROC SQL is used to send a simple query to the database for processing. Notice there is no CONNECT statement, therefore an implicit pass-thru query will be used. Also, the NOEXEC option is used to prevent the query from executing. This is useful when testing a new process.

This query is okay because the log does not contain a statement like, "Unable to convert the query ".

```
SQL_IP_TRACE: pushdown attempt # 1
SQL_IP_TRACE: passed down query:
  select
    COUNT(*) as "Rec_Count",
    SUM(T1."Weight") as "Sum_Weight"
  from "dbname"."sgf2014_class" T1
```

```
PROC DELETE
  DATA= Sandbox.SGF2014_Class;
RUN;

DATA Sandbox.SGF2014_Class (
      DBCREATE_TABLE_OPTS=
       'PRIMARY INDEX(Name)' );
  SET SasHelp.Class;
RUN;

PROC SQL NOEXEC;
  CREATE TABLE Work.Test1 AS
  SELECT
     COUNT(*)    AS Rec_Count
    ,SUM(Weight) AS Sum_Weight
  FROM Sandbox.sgf2014_class;
QUIT;
```

This next example attempts to use a SAS function called USS, which returns the uncorrected sum of squares, against a Teradata table. This causes a message in the SAS log that the SQL statement was not passed to the DBMS, SAS will do the processing. This may not be much of an issue for a table with only 19 records in it; though imagine if the table contains 19 billion records. The query may never even finish.

```
PROC SQL NOEXEC;
  CREATE TABLE work. test2 AS
  SELECT
     COUNT(*)    AS Rec_Count
    ,SUM(Weight) AS SUM_Weight
    ,USS(Weight) as USS_Weight
  FROM Sandbox.sgf2014_class;
QUIT;
```

```
TERADATA_10: Prepared: on connection 0
SELECT * FROM dbname."sgf2014_class"

SAS_SQL:  Unable to convert query to a DBMS specific SQL statement due to an error.
ACCESS ENGINE:  SQL statement was not passed to the DBMS, SAS will do the processing.

TERADATA: trget - rows to fetch: 19
```

PROC SUMMARY

This next example uses a PROC SUMMARY to obtain the USS variable against a Teradata table and something very interesting happens. The SAS log indicates that the SQL statement was passed to the DBMS for fetching data. In this instance, the system was able to convert SAS code to SQL. Notice the statement below contains SQL code to create the USS value whereas the PROC SQL statement did not.

```
PROC SUMMARY
  DATA=Sandbox.SGF2014_CLASS
    NWAY MISSING;
  VAR Weight;
  OUTPUT
    OUT= Work.Test3
    SUM= SUM_Weight
    USS= USS_Weight;
RUN;
```

This is an example where certain procedures will perform as expected and others will not. Additional SAS software exists that are designed to process entirely within the database and will be discussed later.

```
ACCESS ENGINE:  SQL statement was passed to the DBMS for fetching data.

TERADATA_6: Executed: on connection 0
 select COUNT(*) as "ZSQL1",
   COUNT(*) as "ZSQL2", COUNT(T1."Weight") as "ZSQL3",
   MIN(T1."Weight") as "ZSQL4", MAX(T1."Weight") as "ZSQL5",
   SUM(CAST( T1."Weight" AS DOUBLE PRECISION)) as "ZSQL6",
   COALESCE(VAR_SAMP(CAST(T1."Weight" AS DOUBLE RECISION))*(COUNT(T1."Weight")-1),0)
     as "ZSQL7"
 from "dbname"."sgf2014_class" T1
 having COUNT(*) > 0
```

QUERY BUILDER

The query builder is a point and click application within Enterprise Guide that creates SQL statements against the data source. This can be a very useful tool when accessing Teradata because very little knowledge of SQL or SAS programming is needed to use the tool. The example in Figure 8 depicts a Teradata table that has been added to the Project Flow whereby right-clicking on the table provides access to the Query Builder.

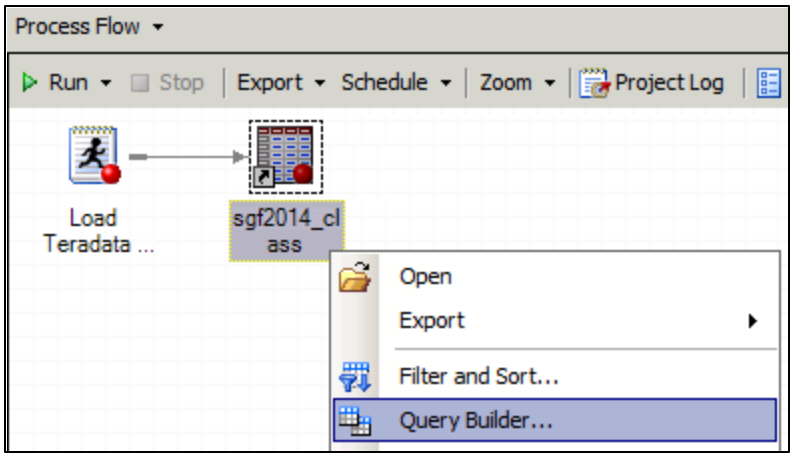


Figure 8: Query Builder in Enterprise Guide

SELECT DATA

The selection of data is easily done by dragging the Weight variable into the Select Data tab and choosing the SUM function under the Summary column as depicted in Figure 9. Note the reference to the Validate Changes Now selection within the Tools drop-down list. This will evaluate the implicit pass-thru query to see if it will process in-database. As previously mentioned, not all functions may be available without the appropriate software.

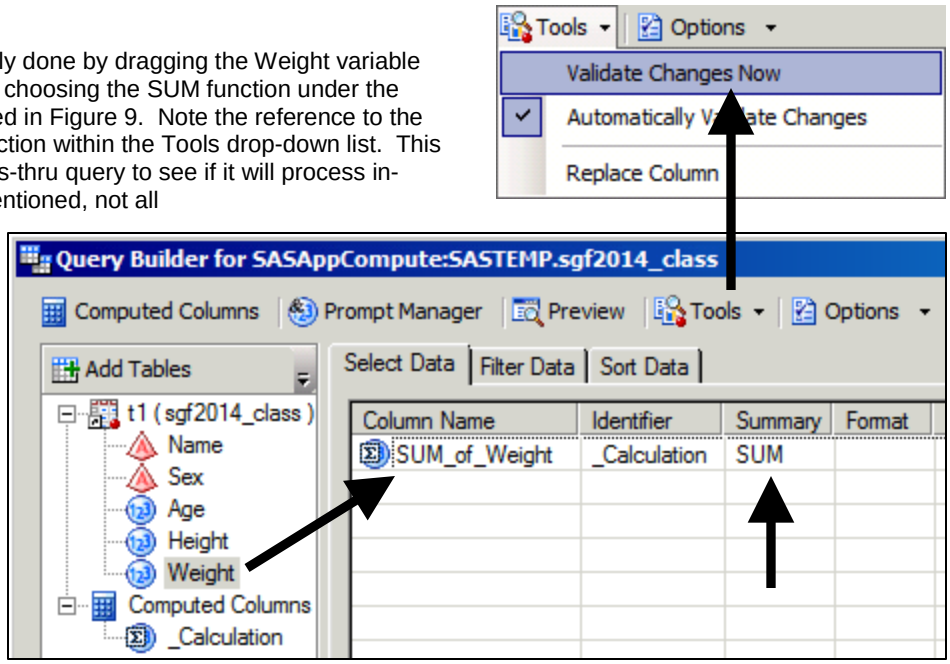


Figure 9: Selecting Data and Summary Function

VALIDATE

Figure 10 shows a pop-up window that contains the SQL generated by SAS and a note near the top depicting whether the expression is valid. In this simple example, the system converts the request successfully.

Figure 11 shows what can happen if a SAS function is currently not supported. The USS function is again used as an example. Notice the Validation pop-up indicates a valid query, which is correct because SAS will indeed create the statistic; though must first bring the data back to the computer it is running on.

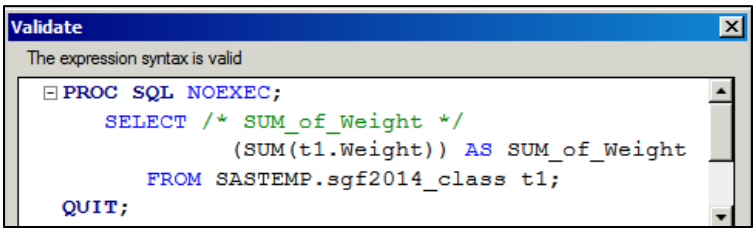


Figure 10: Validated SQL Pass-thru Query

These figures are not meant to discourage the use of the Query Builder. It is, in fact, a very useful tool when accessing Teradata. These comments are to make users aware that some options may not be accessible.

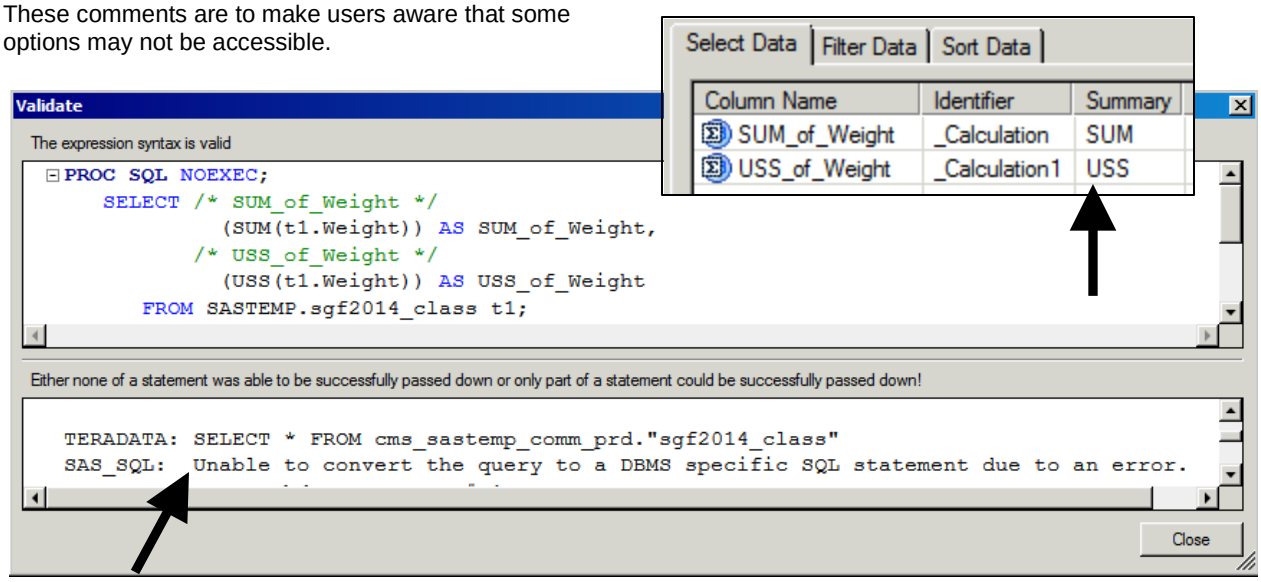


Figure 11: Validation of Unsuccessful SQL Pass-thru

Another useful choice, though not discussed in this paper, is the "**Generate source for explicit pass-through**" selection in the Options pull-down menu of the Query Builder. This will prevent the use of any unknown SAS function by the database and will cause the query to present an error in the log in lieu of just a warning message.

TIPS FOR EXPLICIT PASS-THRU

The previous implicit pass-thru exhibits were presented to demonstrate the ability to quickly create a query using Enterprise Guide against a Teradata table with little or no knowledge of SAS or SQL programming, although the use of implicit pass-thru queries should be used with caution. Inadvertently downloading millions, even billions of records can be quite time consuming. The ability to create true in-database processing is very important. As in the example of reading the entire NCH record in order to make sure the pointer is in the right place, explicit pass-thru queries are the safest way to ensure the most efficient processing. Below are examples of methods that can be used to safely produce in-database results.

1 The Teradata SQL **EXPLAIN** statement provides a good way to predict the performance before executing a piece of code. The statement appears before the SELECT statement as shown in this example. Take note of the `/*TOP 10*/` clause that is currently commented out. This will be explained later.

The output of the SAS data set will contain the results of the EXPLAIN statement. A quick way to determine if an error might exist in the query is to look at the last few records of the report. For instance, if the last record states the query is estimated to take seconds to complete, it is probably safe to say, there is no problem. If the last record indicates that it will take hours, there may be a missing join criteria or other issue with the query.

Once the EXPLAIN plan has been verified, it is a good idea to comment it out and then issue the **TOP 10** statement, which is applied after the SELECT clause. This will output the first ten records for testing purposes. Once the initial data have been verified, this too can be removed.

```
PROC SQL;
 CONNECT TO TERADATA (
  CONNECTION= GLOBAL
  AUTHDOMAIN= 'Domain_ID'
  DATABASE= Database_Name );

 CREATE TABLE SASOUT.SAS_Data_Set AS
 SELECT * FROM CONNECTION TO TERADATA
 (
  EXPLAIN
  SELECT                    1
/*TOP 10*/
     CLM.CLM_TYPE       AS CLM_TYPE
    ,COUNT(*)           AS CLM_CNT
    ,SUM(CLM.PMT_AMT)   AS PMT_AMT
    ,SUM(INL.CHRG_AMT)  AS CHRG_AMT

  FROM MDCR_CLM AS CLM

  INNER JOIN MDCR_CLM_INSTNL AS INL
    ON INL.CLM_NUM = CLM.CLM_NUM

  WHERE CLM.CLM_TYPE = 60
    AND CLM.FINL_ACTN_IND = 'Y'
    AND CLM.THRU_DT = '2012-01-01'

  GROUP BY 1
 );
QUIT;
```

Potential Performance Tips

- GROUP BY as opposed to DISTINCT
- TOP N as opposed to QUALIFY
- Aggregate to appropriate levels
- Use CAST when aggregating totals
- Use CASE to eliminate multiple passes
- Use literal dates to specify time periods opposed to joining to a calendar table
- Always use indexed variables in the WHERE clause

The following are SQL elements that may result in longer execution times; the EXPLAIN plan provides a good way to identify these potential issues up front.

- Complex Join Conditions (i.e. using functions in join conditions)
    - substr(a.cntrct_num,1,3) = substr(b.cntrct_num,1,3)
    - Extract(year from a.clm_thru_dt) = Extract(year from b.clndr_dt)
- Incorrect location of filter criteria (Outer Joins)
- Missing or incorrect columns in the join criteria

FUTURE STATE

The future state of accessing the IDR at CMS may include some of the additional SAS products described below, which could greatly improve data processing efficiency and analysis by eliminating the possibility that processing will need to occur outside of the database environment.

Selective Base SAS and statistical procedures were made available for in-database processing in the first maintenance release of SAS 9.2. In the second and third maintenance releases, the following Base SAS, SAS Enterprise Miner™, SAS/ETS®, and SAS/STAT® procedures are enhanced for in-database processing. Additional procedures are planned for incremental delivery in subsequent releases (Webb 2008). The procedures marked with an asterisk (*) require SAS Analytics Accelerator to run inside the database (SAS 2011a).

- CORR*
- CANCORR*
- DMDB*
- DMINE*
- DMREG*
- FACTOR*
- FREQ
- PRINCOMP*
- RANK
- REG*
- REPORT
- SCORE*
- SORT
- SUMMARY/MEANS
- TIMESERIES*
- TABULATE
- VARCLUS*

Predictive and descriptive model training typically requires many passes over the prepared, de-normalized data. SAS analytic and statistical functions, as well as SAS procedures, that execute repeatedly in model training can be moved inside the database. SAS Enterprise Miner procedures that are candidates for inclusion are DMDB (Data Mining Data Base), DMREG (Regression), NEURAL (Neural Network), DMVQ (Clustering), and others. Possible SAS/STAT software procedures include LOGISTIC, CATMOD, GENMOD, GLM, and many more (SAS 2007).

ANALYTICS ACCELERATOR

When performing in-database modeling, the SAS® Analytics Accelerator for Teradata® dynamically generates SQL, which is based on the procedure options and statements. It then submits the SQL code directly to the database. The code can be standard SQL that can be interpreted by any database, or it can be tuned specifically for Teradata. The choice for the type of SQL code is determined by the complexity of the required analysis (SAS 2011b).

FORMAT PUBLISHING AGENT

The process for exporting custom format definitions is called SAS® Format Publishing®. The SAS FORMAT procedure enables the creation of custom formats that replace raw data values with formatted character values. PROC FORMAT accomplishes this by creating a simple lookup table. SAS In-Database technology contains a tool for exporting these lookup tables from the SAS System to the Teradata system. This tool is delivered as part of the SAS/ACCESS Interface to Teradata and is known as the SAS Format Publishing Agent for Teradata (Webb 2008).

TABLE SERVER PROGRAMMING LANGUAGE

A further goal of SAS In-Database processing is to encapsulate the logic inside the DATA step implicit loop in a context that can be executed inside Teradata. Relative to the DATA step, Table Server Programming Language® (TSPL) is a lightweight implementation that can be executed outside a SAS process. The SAS 9.2 releases of SAS In-Database technology include the tools needed to translate a user's existing DATA step program into TSPL-based functions (Webb 2008).

SCORING ACCELERATOR

The SAS® Scoring Accelerator for Teradata® embeds the robustness of SAS® Enterprise Miner® scoring models directly in the highly scalable Teradata database. By using the SAS In-Database technology and the SAS Scoring Accelerator for Teradata, the scoring process is done inside the data warehouse, and therefore does not require the transfer of data (SAS 2011a).

CONCLUSION

The ability for the SAS System to seamlessly connect to the Teradata platform from any CMS operating system makes the use of these two applications very suitable to analytic operations. The speed at which data can be summarized by the database is tremendously advantageous when dealing with billions of claim records over multiple years and the versatility of SAS to perform data analysis as well as many various data management functions enables the end-user community to quickly adapt to ever changing requirements. In addition to the SAS Access to Teradata product, the technologies in SAS Enterprise Guide and Business Intelligence make it easier to access powerful database platforms like the IDR.

REFERENCES

Ballinger, C. (2007): "Born To Be Parallel", Why Parallel Origins Give Teradata an Enduring Performance Edge.

Clark, S. (2000): "Butler Group Research Paper", Research Paper Teradata, Butler 100101, October 2000.

Heath, S. (2013): "Beyond the lobby doors", 4 ways SAS is like Stark Industries.

Mann, R. (2009): "What's a DBA to do?", Teradata Magazine Online, Retrieved February 15, 2011, http://www.teradatamagazine.com/tech2techtemplate.aspx?id=11188

SAS (2007): SAS Institute, "SAS/ACCESS® 9.2 for Relational Databases", Reference, Fourth Edition

SAS (2011a): SAS Institute, "SAS® In-Database Processing", A Roadmap for Deeper Technical Integration with Database Management Systems.

SAS/Teradata (2014): http://www.analyticheroes.com/, Heroes of Analytics.

Webb, B. (2008): "SAS® In-Database Processing with Teradata", An Overview of Foundation Technology.

Weiming H., (2004): "Top Ten Reasons to Use PROC SQL", Proceedings of the Twenty-ninth Annual SAS Users Group International Conference, 29, 042-29.

Wikimedia Foundation Inc. 2005: Wikimedia Foundation, "Structured Query Language", Retrieved February 15, 2011, http://en.wikipedia.org/wiki/SQL.

CONTACT INFORMATION

Rick Andrews
Office of the Actuary
Centers for Medicare and Medicaid Services
7500 Security Boulevard
Baltimore, MD 21244
Phone: (410) 786-6395
E-mail: Richard.Andrews@cms.hhs.gov

The opinions of the presenter are his own and do not represent the Office of the Actuary, Centers for Medicare and Medicaid Services, or the Department of Health and Human Services.

Iron Man is a fictional character, a superhero that appears in comic books published by Marvel Comics.