## Paper 105-2011

# Explicit SQL Pass-Through: Is It Still Useful?

Frank Capobianco, Teradata Corporation, Lexington, SC

#### ABSTRACT AND INTRODUCTION

SAS/ACCESS® offers two methods of passing SQL to Relational Databases for processing: implicit and explicit. Implicit SQL Pass-Through can be identified by the use of a SAS LIBNAME statement pointing at a relational database. As the name suggests, SAS® will attempt to convert such code to SQL that the target database can process. Over time, and especially with SAS 9.2, Implicit SQL Pass-Through has become increasingly powerful. On the other hand, Explicit SQL Pass-Through is a coding syntax that allows the user to write/submit database-specific SQL that SAS will pass untouched to that database. Because of the increased breadth and depth of Implicit SQL Pass-Through, Explicit SQL Pass-Through is no longer required to address many common scenarios. This paper describes those scenarios, as well as cases where (at least for the time being) Explicit SQL Pass-Through is still useful.

## RECOGNIZING SQL PASS-THROUGH CODING METHODS

As documented in the associated SAS Reference Guide, the SAS/ACCESS for Relational Databases software family has long offered the SAS programmer two methods of coding intended to cause processing in and by a target relational database. The first of these methods is known as Implicit SQL Pass-Through. As the name suggests, the programmer using this coding method implicitly expects SAS to convert that code to SQL and pass it to the target relational database for execution. Implicit SQL Pass-Through can be recognized by virtue of the fact it contains a SAS library reference pointing at the target relational database. For example, the following library reference points at a Teradata-resident database TD\_Workshop.

```
LIBNAME tdtrain TERADATA SERVER=DemoTDAT USER=instructor PWD=XXXXXX DATABASE=TD_Workshop;
```

Incorporating this library name, the following code excerpts thus represent two examples of Implicit SQL Pass-Through.

```
Example 1: PROC PRINT DATA=tdtrain.teradata_table (OBS=5);
Example 2: PROC SQL; CREATE TABLE mydataset AS SELECT * FROM tdtrain.teradata_table;
```

Novice programmers sometimes make the mistake of assuming that the second example shown above is actually an example of Explicit SQL Pass-Through because of the presence of PROC SQL. Although Explicit SQL Pass-Through is embedded in PROC SQL code, it is not synonymous with PROC SQL. As the name suggests, Explicit SQL Pass-Through is based on the premise that SAS will not alter or translate the code, but rather will submit what the programmer has coded verbatim to the relational database for execution.

There are two forms of syntax typically associated with Explicit SQL Pass-Through, namely

## **EXPLICIT SQL PASS-THROUGH SYNTAX FORM 1**

Again using Teradata as an example, the following syntax is typically used to pass Data Manipulation (DML) SQL to the database for execution:

## **EXPLICIT SQL PASS-THROUGH SYNTAX FORM 2**

This form may be used to pass any SQL - including Data Definition Language (DDL) commands - from SAS to the database for execution:

```
PROC SQL;
CONNECT TO TERADATA(USER= PASSWORD= SERVER=
DATABASE= <other connection options>);
EXECUTE(Teradata DML SQL Statement1) BY TERADATA;
EXECUTE(Teradata DDL SQL Statement1) BY TERADATA; EXECUTE(COMMIT) BY TERADATA;
.....
QUIT;
```

#### IMPLICIT SQL PASS-THROUGH ENHANCEMENTS WITH SAS 9.2 M3

With maintenance release three (M3) of SAS 9.2, a greater quantity and variety of SAS code is translated to SQL and passed to the target relational database than occurred with previous releases of SAS. As a result, the SAS programmer can now use Implicit SQL Pass-Through to perform the following kinds of analyses – without resorting to Explicit SQL Pass-Through coding:

## **DESCRIPTIVE STATISTICS**

When the target database is Teradata, DB2 under UNIX and PC hosts, or Oracle, SAS/ACCESS will convert the following elements of Base SAS code to SQL to be executed by that database: PROC FREQ, MEANS/SUMMARY, RANK, SORT, TABULATE, and REPORT.

## **SELECT DATA MINING ALGORITHMS**

Provided a special family of SAS-supplied user-defined functions are installed on the target Teradata system, the following SAS/STAT® 9.22 and SAS/ETS® procedures will be executed, in part, by the Teradata database: PROC REG, CORR, PRINCOMP, FACTOR, VARCLUS, CANCORR, SCORE and TIMESERIES.

The SAS® Scoring Accelerator for Teradata allows SAS Enterprise Miner we users to output Teradata User-Defined Functions to implement the associated SAS scoring functions in-database.

## **SELECT SAS FUNCTIONS**

The SAS 9.2 option SQL\_FUNCTIONS\_COPY can be used both to describe and extend the mapping of SAS functions to their Teradata equivalents. The Teradata database can also process SAS Formats and/or the SAS PUT statement in-database, provided a special family of SAS-supplied user-defined functions are installed on the target Teradata system.

#### SELECT TABLE MANAGEMENT FUNCTIONS

The SAS 9.2 option DBIDIRECTEXEC will further allow certain kinds of SAS PROC SQL statements, focused on table management, to be executed by the Teradata database. In particular, Implicit SQL Pass-Through code - based on PROC SQL - that creates and/or joins Teradata-resident tables will be executed by the Teradata database.

Implicit SQL Pass-Though also applies to SAS Data Steps to upload and download data from the Teradata database using the Teradata Parallel Transport (TPT) utility provided the associated SAS options are enabled. For example, the following SAS log from a Windows-based SAS server represents the execution of code that caused a SAS data set consisting of the integers from 1 to 100,000 to be uploaded to a Teradata database referenced by the SAS library reference "tdtrain" using TPT:

```
146 DATA tdtrain.test(TPT=YES FASTLOAD=YES
147 TPT_TRACE_LEVEL=2 TPT_TRACE_OUTPUT=tpttrace);
148 DO i=1 to 100000;
149 OUTPUT;
150 END;
151 RUN;

NOTE: The data set X.test has 100000 observations and 1 variables.
NOTE: Teradata connection: TPT Fastload has inserted 100000 row(s).
```

## WHERE EXPLICIT SQL PASS-THROUGH IS STILL USEFUL

The enhancements to Implicit SQL Pass-Through described above greatly expand the variety and quantity of SAS code that can be automatically translated to SQL and passed to a database - particularly a Teradata database - for execution. At the same time, Explicit SQL Pass-Through still has value in reducing the risk of an unwanted download of large amounts of data. As a rule of thumb, Explicit SQL Pass-Through offers the most value when the coding scenario at hand might result in implicit SQL query disqualification because the coding is so complicated it might violate one of the Implicit SQL Pass-Through rules embedded throughout the SAS/ACCESS Reference Guide. The following are examples of such coding scenarios:

- Multi-table joins subject to extensive or complicated WHERE clauses
- The need for aggregation (such as supplied by certain Teradata ordered analytic functions) not currently mapped via Implicit SQL Pass-Through
- The need for a Teradata function not currently mapped via Implicit SQL Pass-Through. SAS 9.2 has introduced the SQL\_FUNCTIONS\_COPY option for LIBNAME's, allowing the SAS programmer to determine (and potentially extend) the SAS functions that will be mapped from SAS to Teradata.
- SAS code embedded in a SAS Data step whose source is a database-resident table and whose destination is also a database-resident table.

#### HOW TO DETERMINE WHERE PROCESSING WILL TAKE PLACE

The SAS System Option SASTRACE can be used to see the commands that the SAS/ACCESS engine sends to the target database. The SASTRACE option has behavior that is specific to SAS/ACCESS software. A common setting for this option is

```
OPTIONS SASTRACE=',,,d' SASTRACELOC=SASLOG NOSTSUFFIX;
```

Suppose these options have been submitted using SAS interactively, and the SAS library reference "tdtrain" is subsequently defined using SAS/Explorer to point at a Teradata-resident database. Double-clicking on this library will result in the following SAS log entries:

```
TERADATA_1: Executed: on connection 1

SELECT TableName, CASE TableKind WHEN 'V' THEN 'VIEW' ELSE 'TABLE' END FROM DBC.TableSX WHERE ((TableKind = 'T' OR TableKind = 'V') AND UPPER(DatabaseName)='TD_WORKSHOP')

TERADATA: trget - rows to fetch: 17

TERADATA: trforc: COMMIT WORK
```

That is, the SAS log will contain each SQL command issued and executed - prefaced with a database-specific entry like "TERADATA\_n", where "n" is an integer indicating the number-in-sequence of the SQL statement as it occurred in that SAS session.

#### DATA STEP PROCESSING

Implicit SQL Pass-Through coding is particularly convenient to use when transferring data from a database to SAS, or vice-versa. Unfortunately, at this writing, data step coding can not be used to copy a database-resident table within the database without downloading to SAS (and then uploading) the data back to the database. For example, consider the following SAS code:

```
DATA tdtrain.fc_transactions_copy;
    SET tdtrain.fc_transactions;
RUN;
```

With the previously-described SASTRACE options enabled, this code will result in log entries similar to those shown immediately following this paragraph. Although the TPT option was not attached to this data step, the results would be nearly identical to those shown below – except for slightly clearer verbiage to the effect that the source data set was first downloaded and then uploaded back to the database.

```
TERADATA_12: Executed: on connection 3

CREATE MULTISET TABLE TD_Workshop."fc_transactions_copy" ("acct_nbr" DECIMAL(11),"tx_dte" DATE,"tx_time" FLOAT,"tx_amt"

DECIMAL(10,2));COMMIT WORK

TERADATA_14: Executed: on connection 2

SELECT "acct_nbr","tx_dte","tx_time","tx_amt" FROM TD_Workshop."fc_transactions"

TERADATA: trget - rows to fetch: 23

TERADATA_15: Prepared: on connection 3

USING ("acct_nbr" DECIMAL(11),"tx_dte" DATE,"tx_time" FLOAT,"tx_amt" DECIMAL(10,2))INSERT INTO

TD_Workshop."fc_transactions_copy" ("acct_nbr","tx_dte","tx_time","tx_amt") VALUES (:"acct_nbr",:"tx_dte",:"tx_time",:"tx_amt")
```

To cause such a copy to occur entirely within the database without an intermediate download and upload, the SAS 9.2 option DBIDIRECTEXEC and PROC SQL will produce the desired result, as shown by the following code:

```
OPTION DBIDIRECTEXEC=YES;
PROC SQL;
   CREATE TABLE tdtrain.fc_transactions_copy2 AS
   SELECT * FROM tdtrain.fc_transactions;
QUIT;
```

The associated SAS log excerpt shows the effect of DBIDIRECTEXEC is to cause SAS to submit CREATE TABLE AS syntax to the database for direct execution:

```
TERADATA_20: Executed: on connection 1

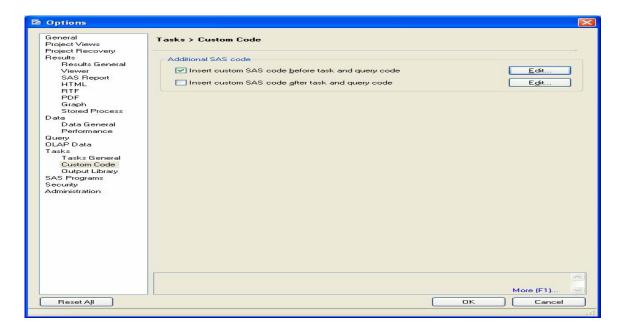
CREATE MULTISET TABLE "TD_Workshop"."fc_transactions_copy2" as ( select TXT_1."acct_nbr", TXT_1."tx_dte", TXT_1."tx_time",

TXT_1."tx_amt" from "TD_Workshop"."fc_transactions" TXT_1 ) WITH DATA
```

It should be clear to the reader from this example that - at this writing - PROC SQL-based coding is required to ensure in-database processing of table-to-table transformations.

## **APPLICATION TO ENTERPRISE GUIDE ®**

The behaviors and concepts described previously in this paper are also naturally applicable to SAS/Enterprise Guide and other SAS tools that ultimately generate SAS code. For example, the user can submit the SASTRACE and DBIDIRECTEXEC options to Enterprise Guide in a number of ways, such as through the Custom Code Task Option, as shown in the following figure:



**Figure 1. Enterprise Guide**One Way To Insert Debug Options

## CONCLUSION

The enhancements to Implicit SQL Pass-Through made available with SAS 9.2 M3 greatly expand the variety and quantity of SAS code that can be automatically translated to SQL and passed to a database - particularly a Teradata database - for execution. Among the analyses which can now be processed in-database are the common descriptive statistics represented by SAS procedures like MEANS and FREQ; certain data mining algorithms from SAS/STAT and SAS/Enterprise Miner; database-resident table management functions associated with copying & transformation; and an increased number of native SAS functions which can be exposed using the SAS 9.2 option SQL\_FUNCTIONS\_COPY.

At the same time, Explicit SQL Pass-Through is still useful in reducing the risk of an unwanted download of large amounts of data. The scenarios where this risk is greatest include: multi-table joins subject to extensive or complicated WHERE clauses; aggregations (such as supplied by certain Teradata ordered analytic functions) and other functions not currently mapped via Implicit SQL Pass-Through; and, SAS code embedded in a SAS Data step whose source is a database-resident table and whose destination is also a database-resident table.

#### REFERENCES

SAS/ACCESS 9.2 for Relational Databases: Reference, Third Edition, 2010.

## **CONTACT INFORMATION**

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

Name: Frank Capobianco
Enterprise: Teradata Corporation
City, State ZIP: Lexington, SC 29072

E-mail: frank.capobianco@teradata.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.