

Data movement issues: Explicit SQL Pass-Through can do the trick

Kiran Venna, Dataspace Inc.

ABSTRACT

Data movement between Teradata and SAS will have huge impact on run time of SAS ETL Job. Usage of explicit SQL Pass-Through will reduce data movement and will also improve the query performance. Proper ETL planning by using explicit SQL Pass-Through whenever it is possible will be very helpful in attaining required results by using optimal resources. explicit SQL Pass-Through can also enhance execution of SAS macros, when a Teradata Table is used to create a SAS data set. This paper presents few case studies related to design of explicit SQL Pass-Through in ETL and SAS Macros. Issues with explicit SQL Pass-Through with SAS macro variables and also with Remerging summary statistics with data are presented along with solutions.

INTRODUCTION

SAS and Teradata play an important role in decision support systems (DSS). Teradata is used for the purpose of data warehousing, where in large amounts of data can be stored and retrieved quickly. SAS is powerful in doing reporting and analytics owing to lot of custom procedures.

ETL with SAS often involves moving data in both directions i.e. from SAS to Teradata and Vice-versa. Data movement between SAS and Teradata will increase I/O time and will also increase run time of an ETL Job significantly. Overall ETL Job run time also depends on where the queries are executed. Owing to parallel architecture, Teradata can execute query in parallel, which can complete ETL Job in significantly less time.

Explicit SQL pass-through is essential for decreasing run time of SAS Job and is main theme of this paper. In this paper, topics covered include different ways of accessing Teradata tables, different case studies of ETL Design involving SQL pass-through and general problems involved in explicit pass through along with their solutions.

Topics covered in this paper are given below.

1. Different categories to access Teradata tables and advantages/disadvantages with each approach.
2. Case study 1: ETL design involving lot of Teradata tables with write access to Teradata permanent tables and final output as SAS table.
3. Case study 2: ETL design involving lot of Teradata tables with no write access to Teradata permanent tables and final output as SAS table.
4. Case Study 3: ETL design involving many Teradata tables with few reference tables in SAS.
5. Case study 4: Advantage of using explicit SQL pass-through in SAS macro.
6. General Issue with macro variables in explicit SQL pass-through and solution to it.
7. Issue of remerging summary statistics with data in explicit SQL pass-through and solution to it

Different categories to access Teradata tables and advantages/disadvantages with each approach

Accessing Teradata Tables in SAS belong to 2 different categories. In first category, data is accessed with help of a libname statement. Second category uses the connect statement in PROC SQL and where in queries are directly passed to DBMS.

In libname category, first libname is defined for Teradata as shown below.

```
libname teralib teradata server=myserver user=myuserid pwd=mypass;
```

After defining libname, Teradata table can be referenced in DATA Step as shown below.

```
data teralib.tab1;  
  set teralib.tab(keep =name);  
run;
```

Teradata table can also be referenced in PROC SQL by referencing libname and is known as Implicit SQL Pass-Through. An example of Implicit SQL Pass-Through is presented below.

```
proc sql;  
  create table teralib.tab3 as  
  select * from teralib.tab  
  where name like '%ABC%';  
quit;
```

Second category of accessing Teradata table is known as explicit SQL Pass-Through, which uses connect statement as shown below.

```
proc sql;  
  connect to teradata (server=myserver user=myuserid pw=mypass);  
  execute(create table edwwrkuser.tab4 as  
  select * from edwwrkuser.tab)  
  with data primary index(cust_id)) by teradata;  
  execute(commit work) by Teradata;  
  disconnect from teradata;  
quit;
```

Main advantage of accessing through Libname statement is that, SAS programmer is often comfortable with the syntax and functions of SAS. Implicit SQL Pass-Through is also very efficient as many SAS queries and functions are passed directly to Teradata for data processing. DBDIRECTEXEC= System option lets the Implicit SQL Pass-Through optimize handling of SQL statements by passing them directly to the Teradata for execution. Disadvantage of Implicit SQL Pass-Through is that, some queries and SAS functions are not passed to Teradata. This will lead to huge I/O processing as this will lead to moving datasets from Teradata to SAS and also results in inefficient query processing.

Main advantage of using explicit SQL Pass-Through is it decreases data movement and also due to parallel processing capabilities of Teradata it enhances query performance. Disadvantage with explicit SQL Pass-Through is to understand various nuances of Teradata SQL.

Case study 1: ETL design involving lot of Teradata tables with write access to Teradata permanent tables and final output as SAS table.

In this case study, SAS Programmer has write access to create Teradata permanent tables. A representative example of this case study is shown below consists of 2 steps. In first step, explicit SQL Pass-Through is used to create 2 intermediate tables and finally a target Teradata table is created. Intermediate tables are dropped as part of cleanup process. Second step

involves usage of Data step to create SAS table from final Target table with Teradata Bulk export utility known as fast export.

First step i.e. explicit SQL Pass-Through is shown below.

```
proc sql;
connect to teradata (server=myserver user=myuserid pw=mypass);
/* 1st table created*/
execute(create edwwrkuser.staging_customer as
select * from edwwrkuser.Cusomter table
where create_dt between '2017-01-01' and '2017-01-31' ) with data primary index(cust_id)) by teradata;
execute(commit work) by Teradata;

/* second Teradata table is created*/
execute(create table edwwrkuser.staging_txn_tbl as
select cust_id, txn_id from PD_EDW_DB.Cusomter table
where order_nb = 1) with data primary index(cust_id)) by teradata;
execute(commit work) by Teradata;

/* final Teradata to be created*/
execute(create table edwwrkuser.Final_cust_txn
select a.* , b.txn_id from edwwrkuser.staging_customer a
inner join
edwwrkuser.staging_txn_tbl
On a.cust_id =b.cust_id) with data no primary index)by teradata;
execute(commit work) by Teradata;

/* cleaning up permanent table*/
execute( drop table edwwrkuser.staging_customer)by teradata;
execute(commit work) by Teradata;
execute( drop table edwwrkuser. Final_cust_txn)by teradata;
execute(commit work) by Teradata;

disconnect from teradata;
quit;
```

One thing to remember while creating Teradata table is to use the key words “with data primary index (yourindexcolumn)” or “with data no primary index” as shown in the above code. Unique Column/Unique combination of columns has to be used for primary index. Bad choice of Primary index will cause data skewing leading to wastage of space and poor query performance.

Once final Teradata table is created in explicit SQL Pass-Through, this table can be used to create SAS table by using DATA Step or by any SQL Pass-Through method. An example of Data step is shown below.

```
libname teradb Teradata user=**** pw=**** FASTEXPORT=YES;
libname saslib '/u/mystuff/sastuff/hello';

data saslib.staging_customer_new;
set teradb.staging_customer_new;
run;
```

One main advantage of using Teradata permanent table is that fastexport utility of Teradata can be used with it. Fastexport utility uses multiple sessions to quickly transfer large amounts of data from Teradata table to a SAS table. To use Teradata fastexport utility fastexport=yes has to be mentioned in SAS Libname statement as shown above.

Case study 2: ETL design involving lot of Teradata tables with no write access to Teradata permanent tables and final output as SAS table.

Case Study 2 is the most commonly encountered scenario, as many of SAS users usually do not have rights to create permanent table in Teradata. explicit SQL Pass-Through can be still used, as every user who has access to Teradata database will have right to create volatile table in Teradata. Teradata volatile tables are similar to a SAS work table, as these tables are persistent for only a particular session. ETL is similar to first case study discussed earlier except few changes. These changes are related to create table statement and moving final volatile table to SAS dataset.

The changes pertaining to create table are mentioned below.

1. connection =global option has to be added in connect statement.
2. Create table statements has to be modified as create volatile table
3. At the end of create table statement “on commit preserve rows” needed to be added. Strangely default for volatile table is “on commit delete rows”. Sometimes when a user/developer forgets mentioning on commit preserve rows will be surprised as this will lead to empty table.

These changes related to create table statement are highlighted in the below code.

```
proc sql;
connect to teradata (server=myserver user=myuserid pw=mypass connection = global);

/* 1st volatile table created*/
execute(create volatile table staging_customer as
select * from edwwrkuser.Cusomter table
where create_dt between '2017-01-01' and '2017-01-31' )
with data primary index(cust_id)on commit preserve rows) by teradata;
execute(commit work) by teradata;

/* second Teradata table creation is not shown*/

/* final Teradata table to be created*/
execute(create volatile table Final_cust_txn
select a.* , b.txn_id from staging_customer a
inner join
staging_txn_tbl
on a.cust_id =b.cust_id)
with data no primary index on commit preserve rows)by teradata;
execute(commit work) by Teradata;
disconnect from teradata;
quit;
```

There are two changes with respect to moving final volatile Teradata table to SAS Data set from first case study and are described below.

1. Adding connection=global dbmstemp=yes to Libname statement to access Teradata volatile table
2. Fast export cannot be used with Teradata volatile tables.

Below code creates SAS dataset from Teradata table and changes from first case study are highlighted.

```

libname tdttemp teradata server=server user=userid pwd=password connection=global dbmstemp=yes;

libname saslib '/u/mystuff/sastuff/hello';

data saslib.staging_customer_new;
  set tdttemp.staging_customer_new;
run;

```

Case Study 3: ETL design involving many Teradata tables with few reference tables in SAS.

In the two case studies discussed earlier, all tables involved in ETL were in Teradata, so explicit SQL Pass-Through was pretty straightforward scenario. In this case study, one or two tables are in SAS and the rest of tables are in Teradata and final output is needed in SAS. Best approach in this scenario to move one of SAS table to Teradata followed by explicit SQL Pass-Through. This approach is useful as we are moving one or few SAS table to Teradata, rather than moving many tables to Teradata. This approach will be efficient both in terms of I/O and query processing.

Moving of SAS table to Teradata depends on the write access of the SAS Programmer. If SAS Programmer has write access to create Teradata permanent tables moving of one SAS table to Teradata can be done in DATA Step as shown below.

```

libname teralib teradata server=server user=userid pwd=password ;

libname saslib '/u/mystuff/sastuff/hello';

data teralib.staging_customer_ref(fastload=yes dbcreate_table_opts='primary index(cust_id)');
  set saslib.cust_ref;
run;

```

There are 2 important noteworthy points in above query that is usage of 2 dataset options. First dataset option is fastload and this dataset option uses multiple sessions to quickly transfer large amounts of data from SAS to a Teradata. Second Data set option is dbcreate_table_opts, which creates primary index on a Teradata table. The dbcreate_table_opts = Data Set option needs a key word primary index followed by column name in parenthesis as shown above.

Once permanent Teradata table is created as shown above then this table is used in explicit SQL Pass-Through along with other tables in Teradata and followed by creating final output in SAS (approach similar to case study 1).

If SAS Programmer does not have write access to create Teradata permanent table, moving of SAS table to Teradata can be done in DATA Step as shown below. Below code has two noteworthy points. First point is usage of connection=global, dbmstemp=yes options in libname statement and lack of fastload dataset option.

```

libname tdttemp Teradata user=**** pw=**** connection=global dbmstemp=yes;
libname saslib '/u/mystuff/sastuff/hello';

data tdttemp.staging_customer_ref;
  set saslib.cust_ref;
run;

```

Once volatile Teradata table is created as shown above, then this table is used in explicit SQL Pass-Through along with other tables in Teradata and is followed by final output in SAS (approach similar to case study 2).

Case study 4: Advantage of using explicit SQL pass-through in SAS macro.

Inspiration of this topic came from developing data scrubbing global macros in one of the projects. Each of data scrubbing macro works on Teradata table and has to traverse through various business rules and a final dataset is created in SAS. Macro used for implicit pass through and DATA Step's used to take more than 2 hours. This often can delay in production Jobs. Once this macro was run through explicit SQL Pass-Through, this macro was completed in less than 10 minutes. Basic example of one of this data scrubbing macro is presented below.

```
%macro test(tddbnm =, tdtblnm=, saslibnm =, sastblnm=);
proc sql;
connect to teradata (server=myserver user=myuserid pw=mypass);

/* one of the data scrubbing step*/
execute(UPDATE &tddbnm.&tblnm
        SET name_indicator = 'BAD'
        WHERE customer_name is null) by Teradata;
/* this could be many steps and many scenarios not shown for simplicity*/
disconnect from teradata;
quit;
/* moving data to SAS*/
proc sql;
connect to teradata (server=myserver user=myuserid pw=mypass);
create table &saslibnm.&sastblnm as
select * from connection to teradata
(Select * from &tddbnm.&tdtblnm);
quit;
%mend;
```

General Issue with macro variables in explicit SQL pass-through and solution to it

In explicit SQL Pass-Through general problems arise while accessing a SAS macrovariable present in double quotes. Generally, Macro facility does not look into macro variables placed in single quotes, double quotes have to be used. Once we keep macro variables in double quotes and reference it, we get an error saying that a particular column does not exist. Double quotes are used for column names in Teradata, which explains the reason for the error.

This issue can be solved in two ways; first one is by using single quotes while creating macro variable followed by referencing the variable as shown below.

```
%let start_dt = '2017-07-01';
%let end_dt = '2017-07-31';
proc sql;
connect to teradata (server=myserver user=myuserid pw=mypass);
execute(create table edwwrkuser.staging_customer as
select * from edwwrkuser.Cusomter table
where create_dt between &start_dt and &enddate) with data primary index(cust_id)) by teradata;
execute(commit work) by Teradata;
disconnect from teradata;
quit;
```

Second one is by using %bquote function on macrovariable in single quotes as shown below.

```
%let start_dt = 2017-07-01;
%let end_dt = 2017-07-31;
create_dt between %bquote('&start_dt') and %bquote('&end_dt')
```

Issue of remerging summary statistics with data in explicit SQL pass-through and solution to it

Implicit SQL Pass-Through allows having fewer non-aggregated columns in a group by clause. Below code gives an example fewer non-aggregated columns in a group by clause.

```
proc sql;
create table teradb.reserve_custagg as
select hhold_id,
       cust_id,
       count(res_id) as cnt_reserve from teradb.customer_table
group by hhold_id;
quit;
```

Above query works in SAS and gives an important note in Log as shown below.

"NOTE: The query requires remerging summary statistics back with the original data."

This remerge is special feature of PROC SQL where in this procedure makes two passes to data and is not allowed in Teradata SQL. If an attempt is made to put the above query in explicit pass through, we get an error as shown below.

SELECT Failed. [3504] Selected non-aggregate values must be part of the associated group.

To run the above query in explicit SQL Pass-Through through we need the to create two derived tables in a query, in first derived table there should be all the columns we want and second derived table we need to have aggregation at the column level we want. Finally joining derived tables on the column/columns on which aggregation was done as shown in below query

```
select a.*, b.booking from
(select hhold_id, cust_id, from edwwrkuser.booking_table)a
inner join
(sel hhold_id, count(reserv_id) as cnt_reserve from edwwrkuser.booking_table
group by 1)b
on a.hhold_id = b. hhold_id
```

Please note papers written by Jeff Bailey and Harry Droogendyk explain in detail about bulk load utilities and how to write certain SAS equivalent code in Teradata respectively are referenced below.

CONCLUSIONS

Running queries in explicit SQL Pass-Through often saves tremendous response time for SAS ETL Job's and SAS Macro's. explicit SQL Pass-Through can be employed irrespective of write access to Teradata Permanent tables. Moving data between SAS and Teradata and Vice versa is faster with Teradata Permanent table as fastload and fastexport can be used. Issues with macro variables in explicit SQL Pass-Through can be fixed with %bquote. Remerge feature of PROC SQL can be emulated in explicit SQL Pass-Through by carefully rewriting the query.

ACKNOWLEDGEMENTS

I would like to specially thank, Kirk Paul Lafler for encouraging me to write papers.

I would like to thank, Lakshmi Nadiya Chintalapudi, Charannag Devarapalli, Anvesh Reddy Perati, Srirama Reddy for helping me with proof reading and giving their valuable suggestions.

REFERENCES

1. SAS Institute Inc. 2014. SAS/ACCESS® 9.4 for Relational Databases: Reference, Sixth Edition. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/documentation/cdl/en/acreldb/67589/PDF/default/acreldb.pdf>
2. http://www.info.teradata.com/HTMLPubs/DB_TTU_15_00/index.html
3. <https://support.sas.com/resources/papers/proceedings11/142-2011.pdf>
4. <https://support.sas.com/resources/papers/EffectivelyMovingSASDataintoTeradata.pdf>
5. <https://support.sas.com/resources/papers/proceedings15/2004-2015.pdf>
6. http://analytics.ncsu.edu/sesug/2016/DM-224_Final_PDF.pdf

CONTACT INFORMATION

Kiran Venna
kiranvenna@gmail.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.