Paper 1567-2014

Iterative Programming In-Database Using SAS® Enterprise Guide® Query Builder

Frank Capobianco, Teradata Corporation

ABSTRACT

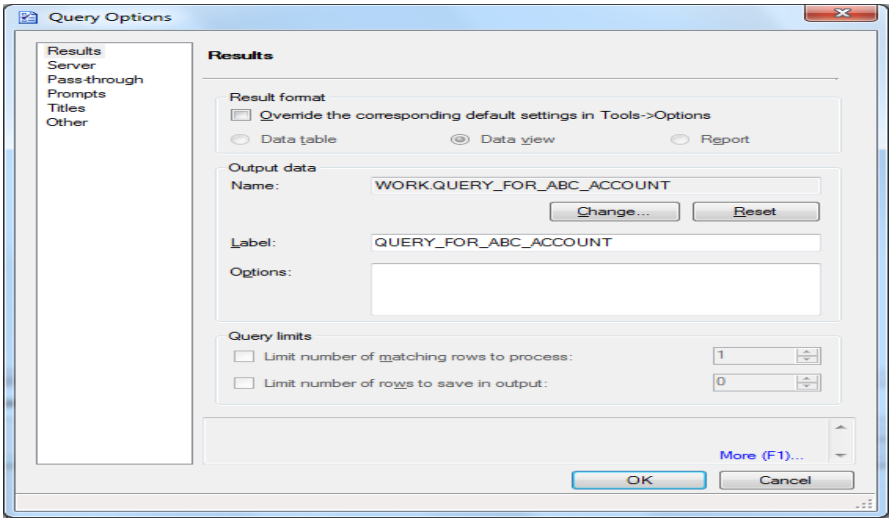
Traditional SAS® programs typically consist of a series of SAS DATA steps, which refine input data sets until the final data set or report is reached. SAS DATA steps do not run in-database. However, SAS® Enterprise Guide® users can replicate this kind of iterative programming—and have the resulting process flow run in-database—by linking a series of SAS Enterprise Guide Query Builder tasks that output SAS views pointing at data that resides in a Teradata database, right up to the last Query Builder task, which generates the final data set or report. This session both explains and demonstrates this functionality.

ENABLING TECHNOLOGY

The behavior described in this paper is part of SAS' in-database processing improvements begun in 2007. The enabling technology required is SAS Enterprise Guide Release 4.3 or later, running on top of SAS/ACCESS® software for relational databases, such as Teradata, beginning with SAS Release 9.2M3 or later.

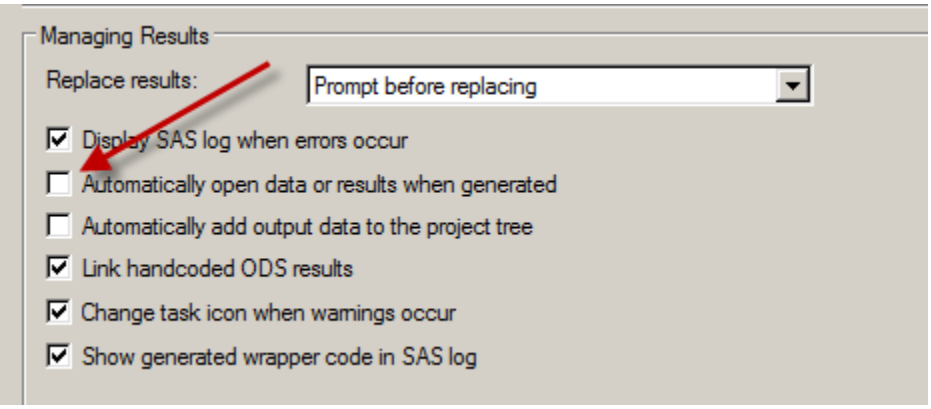
QUERY BUILDER SAS VIEW OUTPUT OPTION

The key to creating a series of linked Query Builder tasks that when executed, will run entirely in-database, is to configure all of the intermediate tasks to output a SAS data view – rather than a SAS Data Table or Report. Display 1 below shows the recommended default setting.



Display 1. Query Builder Results Option

To ensure such views are not automatically opened when created, the user should also uncheck the default option to "Automatically open data or results when generated" as shown in Display 2 below.

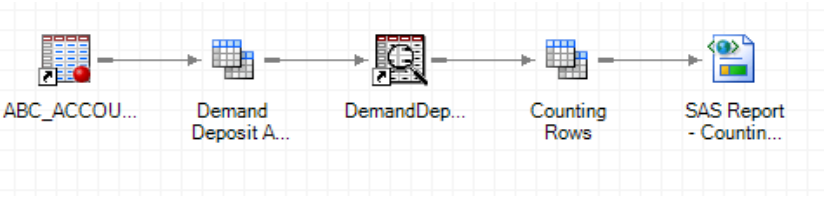


Display 2. Query Builder Results Option

When SAS Enterprise Guide processes a flow of such linked Query Builder tasks – with only the last task in the string creating either a SAS data set or report, it will combine the resultant SQL into a single query. In this way, the user can solve business problems in an iterative process of stepwise refinement, and yet have the process flow execute entirely in-database.

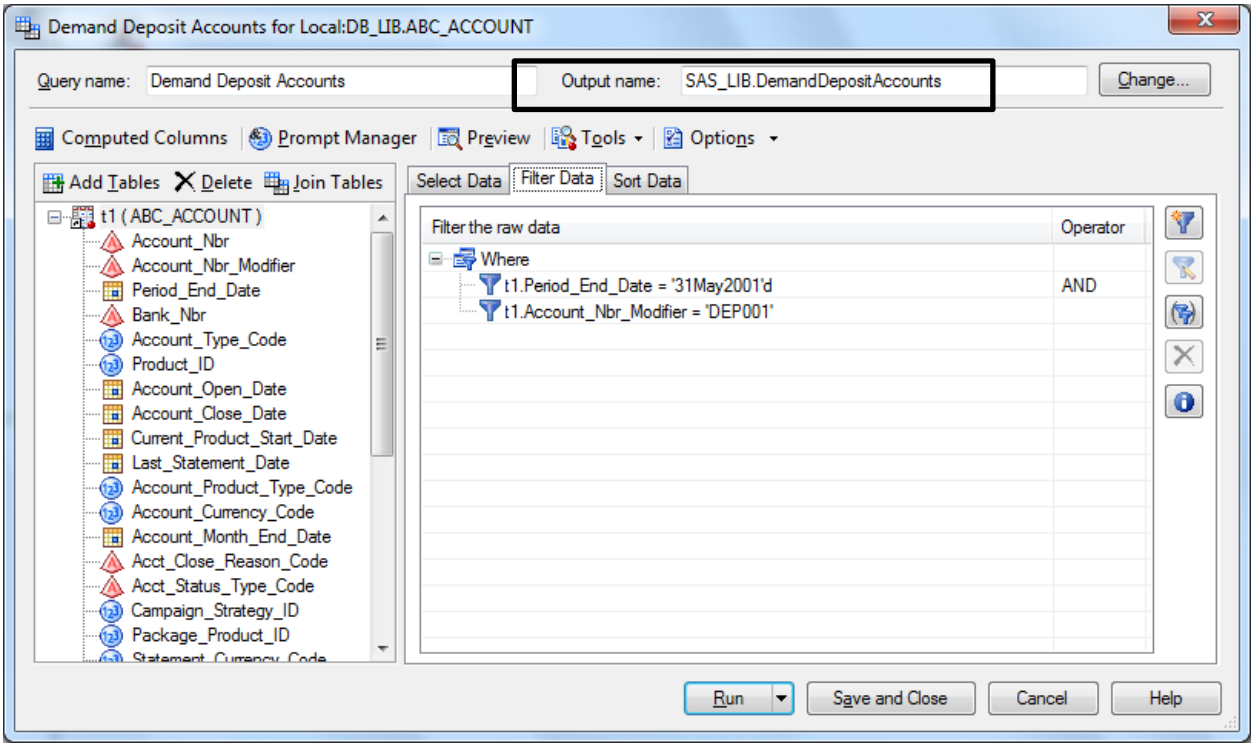
DEMONSTRATION

To demonstrate this concept, suppose the user wishes to calculate the size of a very large population from a database-resident table using two linked Query Builder tasks. In the first Query Builder task, the user will define the population as a SAS data view. In the second Query Builder task, the user will count the number of rows in this data view, outputting the result as a report. Display 3 below shows a process flow consisting of two Query Builder tasks – one called Demand Deposit Accounts and the other Counting Rows. The first task outputs a SAS view, while the second simply returns the number of rows in that view.



Display 3. Query Builder Process Flow

Display 4 below shows the creation of the SAS data view, called DemandDepositAccounts, stored in a local SAS library SAS_LIB, identifying demand deposit accounts at a mythical bank at the close of a specific period.



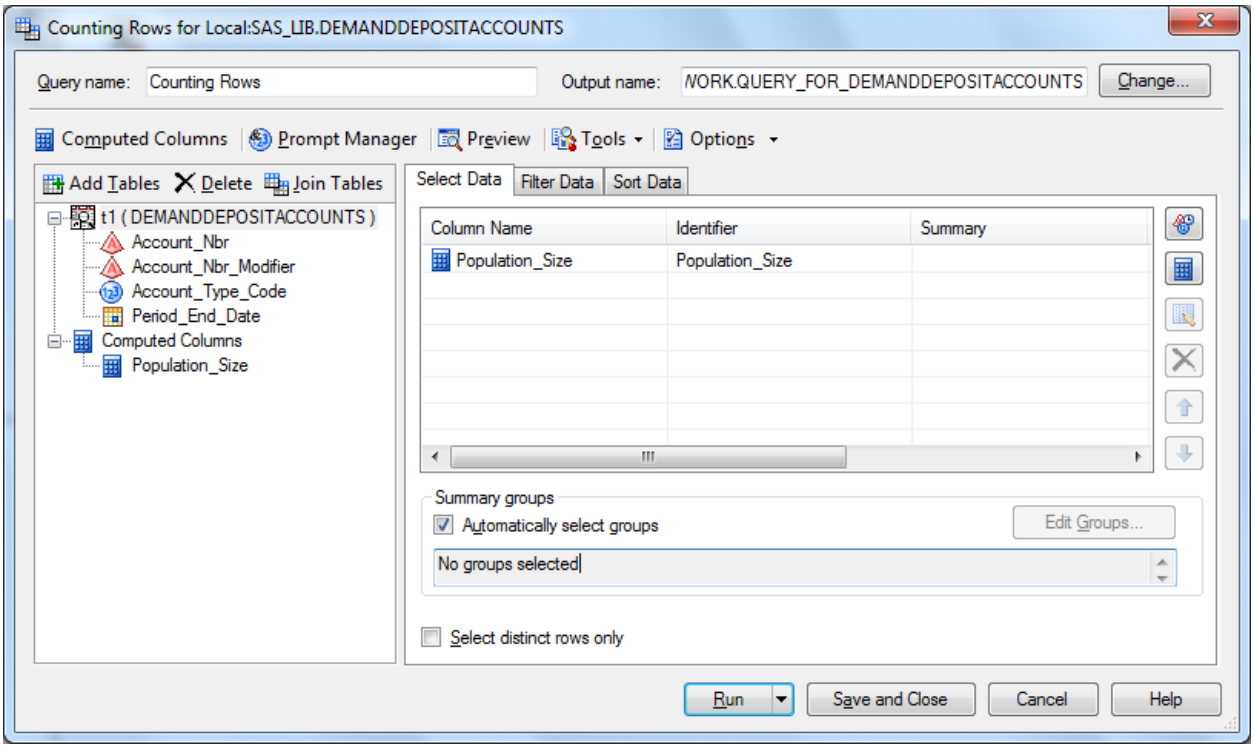
Display 4. SAS View Output By First Query Builder Task

This task executes in a fraction of a second, because it simply defines a view – rather than returning actual data, as shown in the following excerpt from the SAS log:

```
20          PROC SQL;
21             CREATE VIEW SAS_LIB.DemandDepositAccounts AS
22             SELECT t1.Account_Nbr,
23                    t1.Account_Nbr_Modifier,
24                    t1.Account_Type_Code,
25                    t1.Period_End_Date
26                FROM DB_LIB.ABC_ACCOUNT t1
27                WHERE t1.Period_End_Date = '31May2001'd AND t1.Account_Nbr_Modifier =
'DEP001';
NOTE: SQL view SAS_LIB.DEMANDDEPOSITACCOUNTS has been defined.
28          QUIT;

NOTE: PROCEDURE SQL used (Total process time):
      real time           0.03 seconds
      cpu time            0.00 seconds
```

Display 5 below shows the configuration of the follow-on Query Builder task, called Counting Rows, where the previously-set default option has been overwritten to output a SAS report. This task returns the single value Population_Size, which is the count of rows in the population defined by the input data view.



Display 5. SAS View Output

The following excerpt from the associated SAS log for the second Query Builder task shows how SAS has combined the two Query Builder tasks into a single query, executed in-database.

```
TERADATA_18: Executed: on connection 0
 select COUNT(*) as "Population_Size" from "ADSEMDEMO"."ABC_ACCOUNT" t1 where
(t1."Period_End_Date" = DATE'2001-05-31') and
(t1."Account_Nbr_Modifier" = 'DEP001')

TERADATA: trget - rows to fetch: 1

Summary Statistics for TERADATA are:
Total row fetch seconds were:                         0.000006
Total SQL execution seconds were:                     0.401909
Total SQL prepare seconds were:                       0.047843
Total seconds used by the TERADATA ACCESS engine were     0.509056

24          QUIT;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.47 seconds
      cpu time            0.00 seconds
```

CONCLUSION

Business problems often are best solved through an iterative process of stepwise refinement. The ability of SAS Enterprise Guide to combine a series of Query Builder tasks outputting SAS data views into a single in-database query allows the user to apply that process without incurring the download of anything but final data to SAS.

CONTACT INFORMATION

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

Name:	Frank Capobianco
Organization:	Teradata Corporation
City, State ZIP:	Lexington, SC 29072
Work Phone:	803-414-4497
Email:	frank.capobianco@teradata.com