What's Love Gotta Do WITH It
What’s Love Gotta Do WITH It

Jason O’Day, MBA
US Bank

**ABSTRACT**

It has become a need-it-now world, and many managers and decision-makers need their reports and information quicker than ever before to compete. As SAS® developers, we need to acknowledge this fact and write code that gets us the results we need in seconds or minutes, rather than in hours or days. SAS is a great tool for extracting, transferring, and loading data, but as with any tool, it is most efficient when used in the most effective way. Using the SQL pass-through techniques presented in this paper can reduce run time by up to 90% by passing the processing to the database instead of moving the data back to SAS to be consumed. You can reap these benefits with only a minor increase in coding difficulty.

**PROBLEM CONTEXT**

Over many years I have developed reports and production data for the business users to consume. The data was stored in DB2 in several data files that needed to be joined in order to create the reports. One of the other employees was proficient in SAS and had built code to do this, but it would run for over 8 hours and then error out due to space constraints. I was able to build a process that would run in the native DB2 environment therefore reducing the runtime and the resources to complete. The new process finished in minutes and completely tied out. I would like to discuss the what I did to make this work and show the ‘Real Time’ differences to convey the efficiencies that can be gain with this technique.

**Implicit Pass-Through Code**

**Implicit Pass Through requires a Libname Engine:**

LIBNAME mylib db2 user="&dbuser" password="&dbpass" db=&_dsn schema=&_schema PROC SQL;

PROC SQL;
CREATE TABLE WORK.contact AS
SELECT DISTINCT <SAS Code>
FROM mylib.bill1 AS a
INNER JOIN mylib.group_data AS b ON A.gr_key = B.key
INNER JOIN mylib.contact_data AS c ON C.groupkey = B.gr_key
LEFT JOIN mylib.bill2 AS d ON D.bill2_key = A.gr_key
QUIT;

PROC SQL;
CREATE TABLE billing AS
SELECT <SAS Code>
FROM mylib.billing
QUIT;

PROC SQL;
CREATE TABLE type AS
SELECT <SAS Code>
FROM mylib.billing AS bil
LEFT JOIN mylib.ref_values AS val ON VAL.type = BIL.type AND VAL.val_key=1
QUIT;

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.
What's Love Gotta Do WITH It

Jason O'Day, MBA
US Bank

Implicit Pass-Through Code (Continued)

PROC SQL;
CREATE TABLE WORK.billed AS
SELECT DISTINCT
  <SAS Code>
FROM WORK.contact AS cont
  LEFT JOIN WORK.billing AS bill ON CUR.billing_key=HEAD.bill1_key
  LEFT JOIN WORK.type AS type ON CUR.type=CUR.type
WHERE CUR.amt_billed ^=0
QUIT;

Implicit PROC SQL: TOTAL REAL TIME 76.77 seconds

Explicit Pass-Through Code

PROC SQL EXEC;
CONNECT TO db2 AS source (DSN=&_dsn USER="&dbuser" PASSWORD="&dbpass");
CREATE TABLE WORK.billed AS
SELECT * FROM CONNECTION TO SOURCE
  WITH contact AS
  (SELECT DISTINCT
    <Native Language Code>
    FROM &_schema.bill1 AS a
    INNER JOIN &_schema.group_data AS b ON A.bill1_key = B.key
    INNER JOIN &_schema.contact_data AS c ON C.groupkey = B.key
    LEFT JOIN &_schema.bill2 AS d ON D.bill2_key = A.bill1_key
  ),
  billing AS
  (SELECT
    <Native Language Code>
    FROM &_schema.gsbill AS gs
  ),
  applied AS
  (SELECT DISTINCT
    <Native Language Code>
    FROM &_schema..group AS group
    LEFT JOIN &_schema..cash AS paid ON GROUP.key = PAID.group_key
  ),
  type AS
  (SELECT
    <Native Language Code>
    FROM &_schema..billing AS bil
    LEFT JOIN &_schema..ref_values AS val ON VAL.type = BIL.type AND VAL.val_key = 1
  )
  SELECT DISTINCT
    <Native Language Code>
  FROM contact AS cont
  LEFT JOIN billing AS bill ON BILL.gs_key=CONT.gs_key
  LEFT JOIN type AS type ON BILL.type=CONT.type
  LEFT JOIN applied AS app ON APP.key = CONT.key
WHERE BILL.amt_billed ^=0
QUIT;
What’s Love Gotta Do WITH It

Jason O’Day, MBA
US Bank

Conclusion

Compared to the multiple hours the original process took before erring out, the DB2 explicit pass-through technique finished in less than 15 minutes conveying a massive reduction in runtime while still creating the accurate data we needed with no dropped records and no duplicates. When presenting the data and the code to the business user he was extremely happy with not only the results, but also the time savings. Per the example in the appendix below you will see that the time to run is approximately 95% faster in the explicit pass-through rather than the use of the implicit pass-through.