Paper 91

Migrating Databases from Oracle to Teradata

Phillip Julian, PNC Financial Services Group, Inc.

ABSTRACT

We carefully planned and provisioned our database migration from Oracle to Teradata. We had timelines, weekly progress and planning meetings, presentations comparing current state to the future state, detailed project plans for each task, Oracle DBAs, consultants from Teradata, and support from all departments. It was an ideal situation for moving the Enterprise to a new system of record based upon Teradata.

Our team had delays and data issues that no one could anticipate. I had researched every issue that may happen with SAS® upgrades, Teradata, and our particular environment. But the literature and support did not prepare us for anticipating or solving our migration problems. Instead of 6 months, we only had 6 weeks to finish the project.

We had no time to hire an army of experts, so we had to solve our own issues. I will describe those issues, our solutions, and our tricks that facilitated rapid development. I'm still surprised at our rapid progress, and I am thankful for the team's efforts and ingenuity. Our experiences should help others who are planning or performing database and SAS migrations.

Our industry is financial services, we are regulated by the federal government, and we must keep records for every change. Our software environment is SAS on UNIX, SAS on PC, Teradata, Oracle, and Data Integration Studio with the framework of SAS Credit Scoring for Banking. The UNIX hosts are multi-tiered with separate development and production platforms.

INTRODUCTION

As you plan and execute your conversion project, you may need extra time to respond to the 19 items on this list. Most of these items will be discussed later in more detail.

1. Getting the SAS contracts signed.
2. Getting the correct SAS Depot delivered.
3. Finding the UNIX software utilities for SAS/Access® software for Teradata.
4. Getting SAS/Access to Teradata working on the UNIX servers.
5. Getting SAS® software and Data Integration Studio installed on the laptop (64-bit, Windows 7).
6. Teradata views initially had many spool space errors.
7. Teradata engineers did not consult SAS experts on numeric precision in SAS.
8. Teradata numeric data had to be cast using SAS pass-through SQL.
9. BIGINT data was not visible to SAS/Access without casting the columns.
10. Very big or very small DECIMAL data in Teradata could cause runtime errors in SAS.
11. Data Integration Studio 4.9 did not include a Teradata custom transform for pass-through SQL.
12. Our development environment had recurring issues with Teradata libnames in Data Integration Studio.
13. Data Integration Studio objects were too large to export, and had to be broken into 10 pieces.
14. Java may need to be updated to prevent export errors in Data Integration Studio.
15. Validation of the unit test results took a long time.
16. A few omissions and errors occurred after conversion.
17. Training on Teradata.
18. Presentations to users on SAS/Access to Teradata and What's New in SAS.
19. Projects with deadlines in November, December, or January.

TERADATA DATA TYPES

Teradata tables were created with identical column names and table structures, compared to the existing Oracle tables. But column attributes were different for some numerical data, which can cause problems when SAS tries to read, store, or access those columns in Teradata.

- Some Oracle data types for numbers were converted to Teradata data types that cannot be read into SAS without casting to a compatible numeric or character type. When casting is required in Teradata for "big" DECIMAL data, then pass-through SQL must be used. Also, In-Database SAS Procs would be very inefficient processing this data.
    - Oracle numbers without a specified precision were represented as DECIMAL(38,15) in Teradata.
    - Numeric key fields (including our surrogate keys) were given the BIGINT data type.
    - Other numbers had Teradata data types with more than 15 DECIMAL digits.
    - Teradata designers did not know that SAS numerical data had a limit of 15 significant digits.
    - Teradata designers did not create data types based upon the range of values in a column. In most cases, data values were compatible with the SAS 15-digit limit.
    - Casting all BIGINT and big DECIMAL data to character would not be wise, since that uses more table space and is slower in queries when those columns are SQL join keys. Our surrogate keys are BIGINT data, and they are often used as join keys.
    - At this time, no Teradata views accommodate SAS procedures that access tables with BIGINT or big DECIMAL columns.
        - We only have access to Teradata views, but those views do not cast BIGINT or big DECIMAL columns to numbers that are easily consumed by SAS without using pass-through SQL.
        - The SAS/Access libname method does not recognize any BIGINT columns.
        - BIGINT columns will not appear in any Data Integration Studio 4.9 metadata for Teradata tables.
        - Big DECIMAL data can cause numeric overflow or underflow failures in SAS.
- Some string expressions in Teradata required testing both for NULL and the empty string. This definition of a string differs from SAS, which only has the empty string.
- Get your Teradata training completed early. Spend lots of time exploring databases and using Teradata Studio. You need to know your query tools and your databases. You will not be using only SAS to explore your data. Teradata Studio is an excellent tool for data exploration.

MIGRATING DATA INTEGRATION STUDIO

We had to upgrade to SAS 9.4 to get SAS/Access to Teradata, which housed our staging tables. Because of the SAS upgrade, we had to upgrade SAS Data Integration Studio from version 4.21 to version 4.9. Most of our conversion programs were in Data Integration Studio.

We began the conversion by exporting all of Data Integration Studio 4.21 into SPK files, and importing those SPK files into Data Integration Studio 4.9. This method was considered more reliable than an automated update using the Data Integration Studio migration tools. Several factors made the Data Integration Studio migration utilities a poor choice, compared to our export-and-import methods.

- The update may not convert our current Data Integration Studio programs to Data Integration Studio 4.9.
- User-written code is not converted, and is not tracked in Data Integration Studio metadata.
- I could not get precise details on what the automated update would do.
- We only used the SAS Credit Scoring for Banking structures, but we did not depend upon the Credit Scoring for Banking programs, so the automated update may not do much for us.
- The automated update did not decrease the complexity of the upgrade from Data Integration Studio 4.21 to Data Integration Studio 4.9.
- The SAS Community agreed that export-and-import looked like the best method, but any method would be very complicated.
- The SAS admin preferred the export-and-import method.

When using SAS Data Integration Studio, you should backup everything before you change it, and don't limit yourself just to active jobs and tables. Keep the backup SPKs for 6+ months, in case you discover conversion issues or omissions later. Also create backup SPKs for all major changes in Data Integration Studio, in case you need to revert those changes. When SPK exports are created, save the SPK export logs in case there are any issues with creating the SPKs. Keep all backup SPKs and SPK logs for 6+ months.

Keep all code, logs, and listings for all deployed jobs and for the SAS scheduler. If possible, keep historical data for the last 4 or 6 months to cover errors that are not detected until end of quarter or end of year processing.

Data Integration Studio backups may take some time to complete. You may need to export Data Integration Studio structures in smaller pieces, since the export may fail when too many objects are exported at one time. You may have Java memory issues, which require customization by setting Java options or upgrading Java. This may be difficult in a regulated environment where you have no administrative rights to your computer.

When data is imported into a new version of Data Integration Studio, you may have to import Data Integration Studio packages in a specific sequence, because some packages may be dependent on metadata that is in another package.

RESEARCHING PRODUCT REQUIRMENTS FOR THE UPGRADES

To reduce the risk of any SAS upgrade, you should research all SAS and database products that are going to be installed. You also need to look at any interactions between the products. In other words, what specific versions are required for the updated version of SAS that will be installed?

Ask your SAS admin for the SAS configuration report, as soon as the SAS Depot is created. Review the configuration to see if it has all the products that you expected to see. It may be a challenge to map the configuration names to products in SAS.

Look at all new SAS features, which should be verified as working before you announce their availability. Some new products may require other software, and you should create documents about how to get those new SAS features working. Some products will not work at all because you are not authorized to access them -- such as, exporting SAS Enterprise Guide® reports directly into Excel or PowerPoint. Most of the research can be done using the product documentation and consulting with technical support.

Even though it may not be in your job description, you may need to provide information that quickly solves issues in SAS installation, because your project timelines could suffer.

Answers about SAS product upgrades are found in the usual places -- emails and discussion with SAS Support, SAS product documentation, SAS Community, Google, lexjansen.com, and your own in-house experts. When you can't find answers, find experts who can provide the answers. In any case, allow time for research, especially when you don't have time to wait. Most of this research can be done before you have the new SAS Depot.

SAS and Teradata product requirements may be difficult to determine.

- If you don't ask the right questions, then you may not get the right answers. Most technical experts only provide information on specific products. Be prepared to draw your own conclusions about how the various products interact with one another.
- Keep your notes and emails, since you may talk to many people about many issues. Try to have one question per issue or information request. Be prepared to consult with many experts who only know part of your problem.
- You may need to contact your SAS Customer Loyalty representative, who can arrange meetings with several SAS product experts who may answer your questions. If you need that assistance, ask for it early since it may take some time to arrange the meetings.
- Unless you have the exact name of the feature or the product, you may get strange answers from technical support. Define your terms first, and be consistent in all communications. Even when the names come from SAS documents, the names may be confusing when they mean something different to technical support.
    - Some products and features have several names in SAS documents, which can add to the confusion. Confirm your taxonomy for SAS and Teradata products.
    - Example: Explaining what we planned to do with Data Integration Studio, and whether it would be using Oracle or Teradata to access the database tables.
- It was difficult to find out what features will be included in the new Data Integration Studio environment. I was assured that Data Integration Studio for Oracle had the same structures and custom transformations as Data Integration Studio 4.9 for Teradata. But the new Data Integration Studio 4.9 did not have a custom transform written for Teradata pass-through SQL.

FINDING THE SAS JOBS THAT NEEDED UPDATES

We needed a list of jobs to convert. So I created a UNIX file of search parameters to find deployed jobs for Data Integration Studio jobs that used Oracle. Then I executed the search.

```
#-- Create the fgrep search list.
cat > ~/sas/lst/grep_ext.txt
%let from
FROM etl_stage.
control-D

#-- Search the parameters and view the results.
fgrep -i -f ~/sas/lst/grep_ext.txt 1D_EXT*.sas | sort -u | more
```

The above list was used to create the tracking spreadsheet for converting Oracle jobs to Teradata.

TROUBLE GETTING SAS 9.3 AND SAS/ACCESS INSTALLED ON THE PC

A Windows 7 and 64-bit upgrade was required before PC SAS 9.3 or SAS/Access to Teradata or Data Integration Studio 4.9 would run. The Windows 7 install was a packaged release, and it was a piece of cake. The SAS software was not a packaged release, and we had to download the SAS Depot to the laptop – all 30 GB! It looked impossible after several days of waiting for the download to finish.

Network throughput was about 1% to 1.5% for one FTP session, which was too slow. So I aimed for 15% throughput by running 8 simultaneous FTP sessions to get different parts of the SAS Depot. Eventually I downloaded all 30 GB to my laptop in about one day. Then I was able to get all my SAS programs and utilities installed, so that I could start working on the Data Integration Studio tasks.

TROUBLE GETTING TERADATA TOOLS, AND GETTING SAS/ACCESS WORKING

We had the interactive client tools for Teradata and SAS. But SAS/Access for Teradata required a special set of server tools (Teradata Utilities) that could only be got from Teradata customer support. SAS/Access to Teradata would not run without it.

It was difficult to know what to ask for. SAS admin documentation had two different names for the Teradata tools, but the names did not match the Teradata documentation. To find the correct set of tools, you may need to talk to Teradata and SAS representatives, and get them to specify and deliver the required software by name and by version. This was a critical issue for us, and delayed our project, so we had to get it solved quickly.

SAS COULD GET TO TERADATA, BUT DATA INTEGRATION STUDIO COULD NOT

Development environments are always imperfect and challenging. Early on, we only had SAS with SAS/Access to Teradata in command-line mode and in batch mode on UNIX, with a temporary SAS license. But you work with whatever you have at the time, and make as much progress as possible towards the final goal.

Since we could not test with Data Integration Studio, we ran prototype Teradata queries in SAS/Access on UNIX to see if there were issues with our tables and views.

- Uncovered issues with spool space.
- Uncovered issues with poorly written views.
    - "Explain plan" revealed the issues in Teradata.

PLANS TO IMPROVE TERADATA DATA TYPES

SAS Note 39831 recommends creating views to solve the Teradata issue with BIGINT data. We extend that idea to cover issues with big DECIMAL data. We may not be able to create views in Teradata, but we can show what numerical columns need to be cast in views so that they are compatible with the 15-digit limit for SAS numbers.

- We did not create Teradata views to convert these numbers to data types that SAS can process directly. These numbers affect the libname access method in SAS/Access. SAS In-Database Procs will not work efficiently on these views, and they may fail to run.
    - Runtime errors may occur in SAS for large DECIMAL data.
    - The libname access method in SAS/Access will ignore BIGINT columns as if they did not exist.
    - SAS In-Database procedures will not process this numerical data inside the database.
    - Only Proc DS2 could read the data directly, but we did not have experience or expertise or time to rewrite our queries in DS2.
    - Too much production code would change if we converted our code to Proc DS2. Because of the great risk for errors, there would be no conversion to DS2 code.
- No Teradata query existed to find all BIGINT and big DECIMAL data.
    - We have a good Teradata dictionary query now, but it only works on tables and not views.
        - See the example Teradata query (below), and the program that creates the SQL query to find the min and max of all BIGINT and big DECIMAL data.
    - For views, we need to know the tables used and the type casting of columns that are big DECIMAL or BIGINT. We only had EXPLAIN PLAN in Teradata, which does not provide information that is easily consumable like a query of the Teradata data dictionary.
    - At this time, we still have no query that finds all tables that make up the Teradata views. We also had no time for that research.
    - We did not have a query of DECIMAL and BIGINT to find the minimum and maximum values, and determine which columns had numbers with more than 15 significant digits.
    - Our column naming was consistent across all tables and databases, so we could have cast columns based upon their name. But the Teradata tables were already in production. Last minute changes could slow down other conversion projects.
    - In an ideal environment, we would have permissions to create views in Teradata that allow us to fix the BIGINT and DECIMAL numbers so that SAS Procs and In-Database processing can occur without using pass-through SQL in Teradata.

The following code is the Teradata dictionary query mentioned above. You need to add your database names at the bottom of the query, and run it in Teradata Studio. Then run the SAS program that processes the CSV file to produce min and max queries of all big DECIMAL and BIGINT column.

Here are the program header instructions from the SAS program mentioned in the prior paragraph.

```
/*-------------------------------------------------------------------------*
 * Create queries that display the min and max values for columns that may
 * have more than 15 significant digits, which could cause a numeric
 * overflow in SAS.
 *
 * 1. Run the query to produce the spreadsheet.
 *     - TD_Dictionary_Qry.sql is run in Teradata Studio, and results are
 *       exported to CSV.
 * 2. Run this SAS job to create the query for Teradata Studio.
 *     - MinMax_Qry_Tables.sql queries the tables in Teradata, which I could
 *       not test because I did not have read access.
 *     - MinMax_Qry_Views.sql queries the stage and extraction level views
 *       in Teradata, based upon an email from Patrick O'Connor.
 *     - The view names should be correct for stage tables, but you may
 *       need to change other view names to get them working.
 *     - Some columns do not exist in the Teradata views.  They need to be
 *       removed to make the Teradata query run.
 *
 * The program and the query from step 1 (above) can be modified to include
 * more databases.  This program would have to be modified to support the
 * addition or change of databases.
 *
 * Phillip Julian, 4Feb2014
 *-------------------------------------------------------------------------*
```

Here is the generic Teradata SQL query, which is run in Teradata Studio to create the input for the SAS program. Insert your own database name expressions at the bottom for “Your_TD_Database_%” and “Your_TD_DB2_%”.

```
select * from (select DatabaseName, TableName, ColumnName, ColumnType,
CASE ColumnType
WHEN 'BF' THEN 'BYTE(' || TRIM(CAST(ColumnLength AS INTEGER)) || ')'
WHEN 'BV' THEN 'VARBYTE(' || TRIM(CAST(ColumnLength AS INTEGER)) || ')'
WHEN 'CF' THEN 'CHAR(' || TRIM(CAST(ColumnLength AS INTEGER)) || ')'
WHEN 'CV' THEN 'VARCHAR(' || TRIM(CAST(ColumnLength AS INTEGER)) || ')'
WHEN 'D ' THEN 'DECIMAL(' || TRIM(DecimalTotalDigits) || ',' || TRIM(DecimalFractionalDigits) || ')'
WHEN 'DA' THEN 'DATE'
WHEN 'F ' THEN 'FLOAT'
WHEN 'I1' THEN 'BYTEINT'
WHEN 'I2' THEN 'SMALLINT'
WHEN 'I8' THEN 'BIGINT'
WHEN 'I ' THEN 'INTEGER'
WHEN 'AT' THEN 'TIME(' || TRIM(DecimalFractionalDigits) || ')'
WHEN 'TS' THEN 'TIMESTAMP(' || TRIM(DecimalFractionalDigits) || ')'
WHEN 'TZ' THEN 'TIME(' || TRIM(DecimalFractionalDigits) || ')' || ' WITH TIME ZONE'
WHEN 'SZ' THEN 'TIMESTAMP(' || TRIM(DecimalFractionalDigits) || ')' || ' WITH TIME ZONE'
WHEN 'YR' THEN 'INTERVAL YEAR(' || TRIM(DecimalTotalDigits) || ')'
WHEN 'YM' THEN 'INTERVAL YEAR(' || TRIM(DecimalTotalDigits) || ')' || ' TO MONTH'
WHEN 'MO' THEN 'INTERVAL MONTH(' || TRIM(DecimalTotalDigits) || ')'
WHEN 'DY' THEN 'INTERVAL DAY(' || TRIM(DecimalTotalDigits) || ')'
WHEN 'DH' THEN 'INTERVAL DAY(' || TRIM(DecimalTotalDigits) || ')' || ' TO HOUR'
WHEN 'DM' THEN 'INTERVAL DAY(' || TRIM(DecimalTotalDigits) || ')' || ' TO MINUTE'
WHEN 'DS' THEN 'INTERVAL DAY(' || TRIM(DecimalTotalDigits) || ')' || ' TO SECOND(' ||
TRIM(DecimalFractionalDigits) || ')'
WHEN 'HR' THEN 'INTERVAL HOUR(' || TRIM(DecimalTotalDigits) || ')'
WHEN 'HM' THEN 'INTERVAL HOUR(' || TRIM(DecimalTotalDigits) || ')' || ' TO MINUTE'
WHEN 'HS' THEN 'INTERVAL HOUR(' || TRIM(DecimalTotalDigits) || ')' || ' TO SECOND(' ||
TRIM(DecimalFractionalDigits) || ')'
WHEN 'MI' THEN 'INTERVAL MINUTE(' || TRIM(DecimalTotalDigits) || ')'
WHEN 'MS' THEN 'INTERVAL MINUTE(' || TRIM(DecimalTotalDigits) || ')' || ' TO SECOND(' ||
TRIM(DecimalFractionalDigits) || ')'
WHEN 'SC' THEN 'INTERVAL SECOND(' || TRIM(DecimalTotalDigits) || ',' || TRIM(DecimalFractionalDigits)
|| ')'
WHEN 'BO' THEN 'BLOB(' || TRIM(CAST(ColumnLength AS INTEGER)) || ')'
WHEN 'CO' THEN 'CLOB(' || TRIM(CAST(ColumnLength AS INTEGER)) || ')'
WHEN 'PD' THEN 'PERIOD(DATE)'
WHEN 'PM' THEN 'PERIOD(TIMESTAMP('|| TRIM(DecimalFractionalDigits) || ')' || ' WITH TIME ZONE'
WHEN 'PS' THEN 'PERIOD(TIMESTAMP('|| TRIM(DecimalFractionalDigits) || '))'
WHEN 'PT' THEN 'PERIOD(TIME(' || TRIM(DecimalFractionalDigits) || '))'
WHEN 'PZ' THEN 'PERIOD(TIME(' || TRIM(DecimalFractionalDigits) || '))' || ' WITH TIME ZONE'
WHEN 'UT' THEN COALESCE(ColumnUDTName, '<Unknown> ' || ColumnType)
WHEN '++' THEN 'TD_ANYTYPE'
WHEN 'N' THEN 'NUMBER(' ||
         CASE WHEN DecimalTotalDigits = -128 THEN '*'
              ELSE TRIM(DecimalTotalDigits) END
         || CASE WHEN DecimalFractionalDigits IN (0, -128) THEN ''
                 ELSE ',' || TRIM(DecimalFractionalDigits) END || ')'
WHEN 'A1' THEN COALESCE('SYSUDTLIB.' || ColumnUDTName, '<Unknown> ' || ColumnType)
WHEN 'AN' THEN COALESCE('SYSUDTLIB.' || ColumnUDTName, '<Unknown> ' || ColumnType)
ELSE '<Unknown> ' || ColumnType
END
||
CASE WHEN ColumnType IN ('CV', 'CF', 'CO') THEN
    CASE CharType
        WHEN 1 THEN ' CHARACTER SET LATIN'
        WHEN 2 THEN ' CHARACTER SET UNICODE'
        WHEN 3 THEN ' CHARACTER SET KANJISJIS'
        WHEN 4 THEN ' CHARACTER SET GRAPHIC'
        WHEN 5 THEN ' CHARACTER SET KANJI1'
        ELSE ''
        END
    ELSE ''
    END as Column_Type_Length,
CASE WHEN (DecimalTotalDigits > 17 OR DecimalFractionalDigits > 17) THEN 'CAST'
     WHEN ColumnType = 'I8' THEN 'BIGINT'
     WHEN (DecimalTotalDigits between 16 and 17 OR
           DecimalFractionalDigits between 16 and 17) THEN 'MAYBE'
     END as Cast_Required
from DBC.ColumnsV
where (DatabaseName like ('Your_TD_Database_%') OR DatabaseName like any ('Your_TD_DB2_%')) and
      (ColumnType in ('I8') or DecimalTotalDigits > 15 or
       DecimalFractionalDigits > 15 or Cast_Required is not NULL)) Z
order by 1, 2, 3, 4;
```

Once you have created the CSV file, the SAS program inputs the CSV data using these data attributes:

```
            format DatabaseName $50. ;
            format TableName $40. ;
            format ColumnName $40. ;
            format ColumnType $8. ;
            format Column_Type_Length $20. ;
            format Cast_Required $8. ;
```

Now sort the query output, and process the file to produce the min/max query. Database names have been altered.

```
proc sort data=WORK.TD_Dictionary;
    by DatabaseName TableName ColumnName;
run;

data MinMax_Cmds;
    set WORK.TD_Dictionary (where=(DatabaseName in (Your_Name1', 'Your_Name_2'))) end=Done;
    by DatabaseName TableName ColumnName;
    retain From_Line From_Line2 DatabaseName2 TableName2;
    length From_Line From_Line2 DatabaseName2 TableName2 $ 100
           lineA lineB lineC lineD lineC2 lineD2 lineF $ 1000;

    if first.TableName then do;
        From_Line = "from " || compress(DatabaseName || '.' || TableName);
        if (DatabaseName = Your_Name_1') then do;
            DatabaseName2 = 'Your_Name_3';
            TableName2 = compress(TableName || '_V');
            From_Line2 = "from " || compress(DatabaseName2 || '.' || TableName2);
        end; else do; /*-- Only two databases selected --*/
            DatabaseName2 = 'Your_Name_4';
            TableName2 = compress(tranwrd(upcase(TableName), 'SUMRY', 'MNTH') || '_V');
            From_Line2 = "from " || compress(DatabaseName2 || '.' || TableName2);
        end;
    end;

    lineC =  compress("'"  || DatabaseName  || "'") || " as DB";
    lineC2 = compress("'"  || DatabaseName2 || "'") || " as DB";
    lineD =  compress(",'"  || TableName  || "'") || " as TableName";
    lineD2 = compress(",'"  || TableName2 || "'") || " as TableName";
    lineF =  compress(",'"  || ColumnName || "'") || " as ColumnName";
    lineA = ",min(" || strip(ColumnName) || ") as min_" ||
                substr(strip(ColumnName), 1, min(28, length(strip(ColumnName))));
    lineB = ",max(" || strip(ColumnName) || ") as max_" ||
                substr(strip(ColumnName), 1, min(28, length(strip(ColumnName))));

    if first.TableName then do;
        file "&DataLoc/MinMax_Qry_Tables.sql";
        put 'select' / @4 lineC / @4 lineD / @4 lineF / @4 lineA / @4 lineB / From_Line @;
        if last.TableName then put ';' //;  else put;
        file "&DataLoc/MinMax_Qry_Views.sql";
        put 'select' / @4 lineC2 / @4 lineD2 / @4 lineF / @4 lineA / @4 lineB / From_Line2 @;
        if last.TableName then put ';' //;  else put;
    end;
    else do;
        if ^last.TableName then do;
            file "&DataLoc/MinMax_Qry_Tables.sql";
            put 'UNION' / 'select' / @4 lineC / @4 lineD / @4 lineF /
                @4 lineA / @4 lineB / From_Line;
            file "&DataLoc/MinMax_Qry_Views.sql";
            put 'UNION' / 'select' / @4 lineC2 / @4 lineD2 / @4 lineF /
                @4 lineA / @4 lineB / From_Line2;
        end; else do;
            file "&DataLoc/MinMax_Qry_Tables.sql";
            put 'UNION' / 'select' / @4 lineC / @4 lineD / @4 lineF /
                @4 lineA / @4 lineB / From_Line @;
            if last.ColumnName then put ';' //;  else put;
            file "&DataLoc/MinMax_Qry_Views.sql";
            put 'UNION' / 'select' / @4 lineC2 / @4 lineD2 / @4 lineF /
                @4 lineA / @4 lineB / From_Line2 @;
            if last.ColumnName then put ';' //;  else put;
        end;
    end;
run;
```

The min/max query output looks like this, where it performs a union of all column ranges per table. You can run this program in Teradata Studio, and it should produce one table that you could export to Excel for analysis. One table is produced because the columns have the same names for all queries of all tables. This information could be used to create Teradata views that respect the 15-digit limit in SAS.

```
select
   'Your_Database' as DB
   ,'Table1' as TableName
   ,'Var1' as ColumnName
   ,min(Var1) as min_Var1
   ,max(Var1) as max_Var1
from Your_Database.Table1
UNION
select
   'Your_Database' as DB
   ,'Table1' as TableName
   ,'Var2' as ColumnName
   ,min(Var2) as min_Var2
   ,max(Var2) as max_Var2
from Your_Database.Table1 ;
```

WRITING THE TERADATA CUSTOM TRANSFORM

When we discovered that Data Integration Studio 4.9 had no Teradata custom transform for pass-through SQL, it was too late to hire consultants to write a custom transform. Without the transform, we were dead in the water.

- I spent a few days, and wrote my first custom transform for Data Integration Studio, since we needed it and we had no time to spare.
- I started by copying the Oracle custom transformation for pass-through SQL, and studying how it worked. Then I converted the custom transformation so that it produced SAS code that was compatible with Teradata pass-through SQL.
- Our new custom transform for Teradata used "information hiding" to simplify the where clauses by having drop-down menus to support three view methods and three date ranges. Otherwise, where clause queries would be hard to read and prone to coding errors.

PREPARATIONS TO SAVE TIME IN CONVERTING TO TERADATA

Many objects were added to the Data Integration Studio environment to speed up the conversion to Teradata. We had to test everything while not disturbing the existing environment, so we needed output datasets from unit testing to go to their own directory. A new libname was created for that purpose in Data Integration Studio. Then I copied all of the output table metadata, and edited the objects to use the new libname. Now a simple table replacement would redirect output to our testing library.

We also needed to get metadata from the Teradata input tables, especially when some tables had 400 columns. Again, I created Teradata libnames in Data Integration Studio for our databases. At that time, I wasn't sure that we could get Teradata metadata from Data Integration Studio because we had technical issues. So I made a guess that the Oracle metadata may be almost identical to the Teradata metadata, which was a good guess about 90% of the time. The Oracle metadata could be copied to the new Teradata metadata by using metadata editing features of Data Integration Studio. If Oracle metadata differed from Teradata metadata, then Data Integration Studio would raise an error, and then I would correct the metadata.

I created a detailed set of instructions for training, and a brief set of instructions as a check list of actions to perform the migration to Teradata. I often converted 4 jobs an hour, while running tests in 4 different sessions of Data Integration Studio. Note that steps 1 and 4 used the new objects that were created to simplify the conversion process. The brief checklist instructions are listed below.

1. Add the new Data Integration Studio objects:
   a. Teradata input table
   b. Custom transform -- "SQL Pass-Through Query TD"
2. Add an input port to "SQL Pass-Through Query TD"
3. Update the custom transform
   a. Copy and paste the Oracle columns into "SQL Pass-Through Query TD"
   b. Copy the Oracle table name without the database to "SQL Pass-Through Query TD"
   c. Use the same database in Teradata for "Options --> Connection"
   d. Set the Connection and Statement options for the Teradata transform
   e. Cast variables as needed -- all BIGINT, some DATE, and some TIMESTAMP
   f. Check the "Mappings" tab to make sure everything looks OK.
      i. Propagate columns, if needed in SQL Pass-Thru or Table Loader.
4. Unlink the old and relink the new objects.
5. Replace the output table with the table of the same name under TD folders.
6. Correct the control flow.
7. Save the job, and run it.
8. Save results, update tracking spreadsheet, and notify the testing resource.

When testing and Teradata conversion was all finished, no changes were required for the new Data Integration Studio code. Only the new libname location was changed to point to the production library, which is the way that they run today.

DATA MANAGEMENT, DATA GOVERNANCE, AND VALIDATION

Every test required evidence, so artifacts must be created and kept for every test. If you forgot to create a test log, then you had to rerun the program to get the SAS log.

When three people are developing and testing so many programs, the lone testing resource can get overloaded. So we made his job easier by doing our own testing and initial QA. If we found errors, then we corrected them before passing them on to testing.

Any differences had to be explained, and data research was needed for any unresolved difference. We had more than our share of differences to resolve because our surrogate keys would not be in synch until our project ended. Since we used Proc Compare, we had to find good keys to use instead of our surrogate keys.

You may start your key search by sorting with all columns, and then trimming the list until you find a sort order that gives good results. If you still have issues, then you had to look for a match in the common subset between the pair of files. If you still have no match, then you have a problem that should be corrected or explained. So we really missed the surrogate keys. And we needed a good SAS macro to make our testing easier.

When the macro finished, you had an RTF file on UNIX with the comparison results, which could be sent to the tester for analysis. I ran this through Proc Connect, but you can run it any way you like. This will run slow on large files because it does several sorts and several subsets.

Here is the macro that was used for testing, with a few names hidden to protect directory paths:

```
/*-- Define all libraries here!! These libnames are used in the macro --*/
libname ext_1D "xyz";
/*-- Library for Terdata test results--*/
libname ext_1TD  "xyz_teradata";
/*-- Library for the sorted datasets for Proc Compare analysis --*/
/*-- CHANGE THIS TO YOUR OWN LIBRARY FOR DATASETS AND RTF REPORTS --*/
libname R2_Home "wherever/you/live";
/*-- Set Compare options to print more columns with differences: 1.11E-11 --*/
%let Cmp_Options =  MAXPRINT=(10, 32627) Method=Absolute Criterion=-100000
ListCompvar;

%macro compare_datasets (
    base=ext_1D,    /*-- Oct 2014 saved datasets --*/
    compare=ext_1TD,/*-- Teradata test output datasets --*/
    out=R2_Home,    /*-- UNIX location for sorted dataset to compare --*/
    tbl=Tbl_Name,   /*-- Table to compare --*/
    Common=N,       /*-- Pick the common subset to compare --*/
                    /*-- Set to N when Base and Compare have same # of OBS --*/
    BD2=);          /*-- Make this equal to _BD2 for BD2 comparisons --*/

    /*-- Reset the length of the name so it's <= 32 characters --*/
    %if %length(&tbl) < 25 %then %let strlen = %length(&tbl);
    %else                        %let strlen = 25;
    %let tbl_Short = %substr(&tbl,1,&strlen);

    options linesize=110;
    ods rtf file="/Some_local_UNIX_Dir/Where_to_Save_files/dat/&tbl.&BD2..rtf";

    proc sort data=&base..&Tbl out=&out..&tbl_Short._1;
        by &columns;
    run;
    proc sort data=&compare..&Tbl out=&out..&tbl_Short._2;
        by &columns;
    run;

    %if &Common = Y %then %do;
        data &out..&tbl_Short._1_Only &out..&tbl_Short._2_Only Common;
            merge &out..&tbl_Short._1 (in=One keep=&columns)
                  &out..&tbl_Short._2 (in=two keep=&columns);
            by &columns;
            if      One and not Two then output &out..&tbl_Short._1_Only;
            else if Not One and Two then output &out..&tbl_Short._2_Only;
            else output Common;
        run;
        data &out..&tbl_Short._1_Only;
            merge &out..&tbl_Short._1 Common (in=OK);
            by &columns;
            if OK;
        run;
        data &out..&tbl_Short._2_Only;
            merge &out..&tbl_Short._2 Common (in=OK);
            by &columns;
            if OK;
        run;
        proc compare base=&out..&tbl_Short._1_Only
             compare=&out..&tbl_Short._2_Only &Cmp_Options;
            title "Comparing &tbl. only with Common data";
        run;
    %end;

    proc compare base=&out..&tbl_Short._1
         compare=&out..&tbl_Short._2 &Cmp_Options;
        title "Comparing &tbl. without checking for Common data";
    run;

    ods rtf close;
%mend compare_datasets;
```

See the definition of the macro for the documentation for the parameters. Define all libnames at the top of the code.

OTHER TASKS AFTER THE INITIAL CONVERSION IS COMPLETED

- Get the autoexecs and the SAS scheduler (for Data Integration Studio) working.
- Compare the autoexecs, SAS options, and SAS product lists for both versions on all hosts.
- Update UNIX scripts to use the new version on SAS.
- Keep the older Data Integration Studio around for a while until the new version is completely done and validated. If you can, keep the old Data Integration Studio for 3 to 6 months, in case you run into any conversion issues or omissions. Keep the SPK exports for 6 months or longer.
- Later when we wanted to run a test on the QA database, we had to use other tricks to create the test ... namely editing the programs after the fact using UNIX shell commands. Here is how that was done:
    - Batch edit all SAS programs to correct a value in the Teradata custom transform. In this case we wanted to run the QA version instead of the Production version of the Teradata databases. After some research and string searches, these were the UNIX editing commands that did the trick:

```
#-- Number of changes in all programs
grep -n 'nrquote(P);' *.sas | wc -l
94 of 263 files

#-- Shell statements to edit the programs
for x in *.sas
do
    echo $x ${x}_new
    sed s/'nrquote(P);'/'nrquote(Q);'/g $x > ${x}_new
    diff $x ${x}_new
    if [ $? -eq 0 ]
    then
        rm ${x}_new
    fi
done

#-- Now verify the record count of the number of changes
grep -n 'nrquote(Q);' *.sas_new | wc -l
#-- 92 of 261 files
#-- But only 60 files
grep -n 'nrquote(P);' *.sas_new | wc -l
0 files
```

    Next, backup the *.sas files, rename the *.sas_new to *.sas, and copy the edited files back to your production directory to run them. Restore the backup *.sas after your testing is completed.
- Create and deliver presentations on using Teradata with SAS, including examples of Teradata pass-through SQL, the Teradata libname access method, and Teradata In-Database technology.
- Create and deliver a presentation on What's New in SAS.
    - Cover What's New for all intervening versions, since SAS does not repeat itself on the What's New.
    - Present information, references, working examples, and an indexed list of relevant SAS papers to the user community. Post the papers on a shared storage device.
    - Make sure that new features work before you announce those features.

DID WE MISS ANYTHING?

Complicated and numerous changes are rarely 100% correct. You have to plan for the possibility that you missed something crucial. If you missed converting a program, would you have all the pieces and parts to fix it -- the old software, the old code, the old Data Integration Studio interface to visually show what needs to be fixed?

When everything passes your tests, you still need to worry about omissions and testing coverage. We had a few surprises, but we fixed them before they affected the data consumers. We did shut off the older Data Integration Studio 4.21, and it would have been nice to have it to fix one program that had not been converted. But conversion can also be done from the Data Integration Studio source code. You need a good comparison program (Beyond Compare) and a sharp eye to evaluate the differences and make the proper changes.

Searching for Oracle jobs in all UNIX development trees is a good final test to find any omissions.

What is it about projects with deadlines in November, December, or January? That's vacation time for many people, and you may have delays because there are not that many working days in November and December. The timing of the deadline adds to the risk of being late. In our case, we pushed the deadline forward to February, which was not too bad considering our challenges.

WHAT'S NEXT?

- More conversions and more data to learn, of course.
- Converting our modeling database to a new platform.
- Creating our Analytics Competency Center for Enterprise Data Management.

REFERENCES

SAS Institute Inc. 2014. *SAS/ACCESS® 9.4 for Relational Databases: Reference, Sixth Edition*. Cary, NC: SAS Institute Inc.
SAS Institute Inc. 2014. *SAS® 9.4 Intelligence Platform: System Administration Guide, Third Edition*. Cary, NC: SAS Institute Inc.
SAS Institute Inc. 2014. *SAS® 9.4 Companion for UNIX Environments, Fourth Edition*. Cary, NC: SAS Institute Inc.
SAS Institute Inc. 2014. *System Requirements for SAS® 9.4 Foundation for Solaris for x64*. Cary, NC: SAS Institute Inc.
SAS Institute Inc. 2014. *SAS® Credit Scoring for Banking 5.3: Administrator's Guide*. Cary, NC: SAS Institute Inc.
SAS Institute Inc. 2014. *SAS® Credit Scoring for Banking 5.3: Migration Guide*. Cary, NC: SAS Institute Inc.
SAS Institute Inc. 2014. *SAS® Credit Scoring for Banking 5.3: User's Guide*. Cary, NC: SAS Institute Inc.
SAS Institute Inc. 2014. *SAS® 9.4 Intelligence Platform: Desktop Application Administration Guide, Third Edition*. Cary, NC: SAS Institute Inc.
SAS Institute Inc. 2014. *SAS® Data Integration Studio 4.9: User's Guide*. Cary, NC: SAS Institute Inc.
SAS Institute Inc. 2014. *SAS® 9.4 In-Database Products: User's Guide, Fourth Edition*. Cary, NC: SAS Institute Inc.
Teradata Corporation. 2013. *SQL Data Types and Literals*. San Carlos, CA: Teradata Corporation.
Teradata Corporation. 2013. *SQL Functions, Operators, Expressions, and Predicates*. San Carlos, CA: Teradata Corporation.
"Teradata Developer Exchange". 2015. Available at http://forums.teradata.com/forum/teradata-studio
"Teradata Studio Features". 2015. Available at http://developer.teradata.com/tools/articles/teradata-studio
"Teradata Studio Download". 2015. Available at http://downloads.teradata.com/download/tools/teradata-studio
"Beyond Compare". 2015. Available at http://scootersoftware.com

CONTACT INFORMATION

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

Phillip Julian
PNC Financial Services Group, Inc.
131 N. Church St.
Rocky Mount, NC 27804
(252) 454-3604
Devo.spudboy@gmail.com