Paper 116-2010

SAS® Data Integration Studio: Tips and Techniques for Implementing ELT

Jeff Stander, SAS Institute Inc., Lake Mary, Florida

ABSTRACT

Are you looking to improve performance of your data integration processes? One best practice is to stage data inside the database and direct SAS® to do its data integration work inside the database. Extract, load, and transform (ELT) is an increasingly popular way to minimize data movement and gain efficiencies using your existing database management system. This paper explains how to enable ELT processing using the newest features of SAS® Data Integration Studio 4.21 and SAS/ACCESS®.

INTRODUCTION

Several trends are influencing applications of data integration, including increasing data volumes, new database technologies, and innovations in SAS Data Integration Studio and SAS/ACCESS.

First, there is an ever increasing demand for more data integration processing inside the database management system. This is due to growing data volumes. Some studies indicate that data volumes are doubling every 11 months. This increase in data volumes means we have to be more efficient on how we process data. Database consolidation is one strategy organizations are undertaking in order to be more agile.

Another recent trend is the innovations in database technologies and data warehouse appliances. More computing power is becoming available to do data integration inside the database. These technologies have extremely efficient ways to handle large data volumes without having to move data back and forth.

SAS Data Integration Studio now has enhanced pass-through Structured Query Language (SQL) generation for most database management systems, and provides better support for in-database processing.

For many years, users of Base SAS have used implicit and explicit SQL Pass-Through capabilities found in PROC SQL and SAS/ACCESS software to push processing into the database management system. SAS Data Integration Studio provides many of the same capabilities, minus the hand coding, by using metadata to control the processing.

SAS DATA INTEGRATION STUDIO – AN EXAMPLE ELT JOB

When using ELT, you make an architectural decision to use the database management system to do the transformation. Using this technique, you move your data into the database management system (DBMS) and then use SQL to perform DBMS-specific processing on the data. The basic difference between extract, transform, and load (ETL) and ELT is that you do not extract the data to perform operations on it. Architecturally you extract the data from the sources, load by using an ODS or staging area on the target, and then use the database to complete the transformation. See Figure 1. Some benefits of using an ELT approach include: reduced data movement, ability to leverage DBMS scalability and parallel processing capabilities, and in some cases reduced storage requirements due to using a centralized data store.

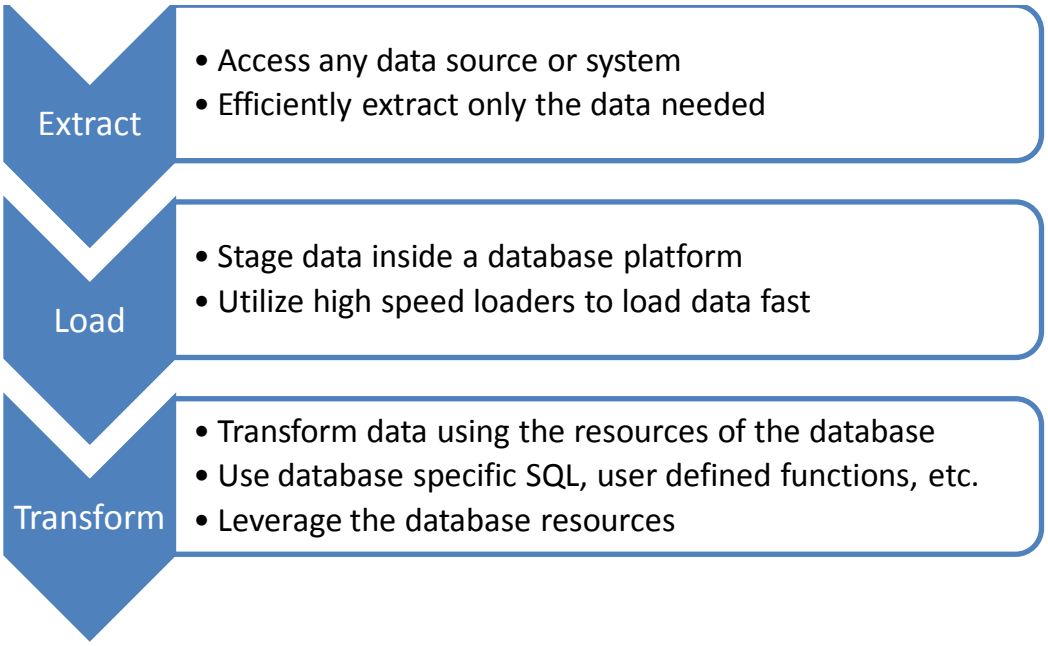


Figure 1: Extract, Load, and Transform Process

SAS Data Integration Studio 4.21 enables users to quickly develop optimal data integration workflows, minimize data movement, reduce overall storage requirements, and significantly improve overall data integration performance. SAS Data Integration Studio makes it easy to do ELT processing.

The following example shows a SAS job implementing an ELT process inside Teradata. You can use this technique with other databases as well. In Figure 2, we are extracting data, loading it into a staging area, and then completing data transformations and creating some target tables inside the database.

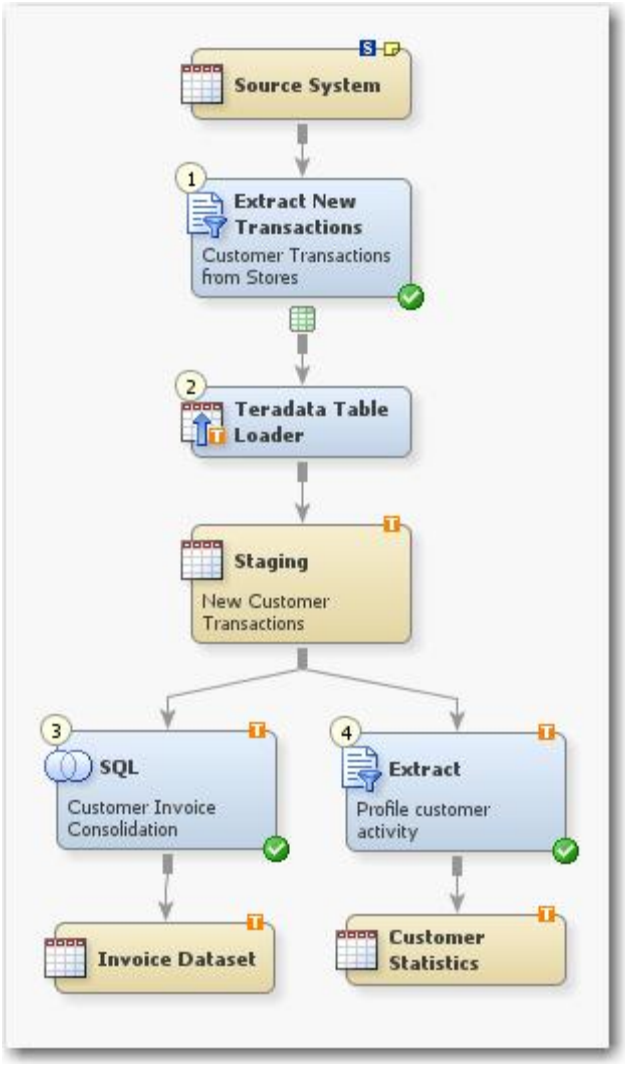


Figure 2: Completed ELT Process in SAS Data Integration Studio

As Figure 2 shows, you can enable ELT processing using standard features found in SAS Data Integration Studio 4.21. Let's look at 11 key capabilities of SAS Data Integration that leverage the power of the DBMS for data integration.

1: EXTRACT DATA FROM SOURCES

SAS Data Integration Studio provides a library of standard transformations, including one for extracting. The Extract transformation enables you to select tables, columns, and rows to extract from various sources. Among others, the sources can be any database management system, file format, or SAS data set. The result is a view that simplifies future load and transform steps and supports future in-database calculations of root data elements.

Extracting data from the source systems can be done using a number of technologies including SAS Data Integration Studio and its integration with SAS/ACCESS Engines, SAS Data Surveyors, and Base SAS. Using SAS Data Integration Studio, you select the data that you want to extract from the source systems using the connections to the various databases, files, queues, applications, and so on. In addition, SAS Data Integration Studio provides several transformations for changed data capture (CDC) that enables you to read only the changed data and propagate those data forwards.

By default, the Extract transformation creates a SAS view. (See Figure 3.) This means that data is not read into a SAS data set; rather the view contains simply logic to extract the data when needed. The SAS view can be used by table loaders in SAS Data Integration Studio when you load data into the DBMS. You have the option to create a

SAS data set. For example, you might want to create a SAS data set when you have data that needs to be extracted at a certain point in time, and the target database is not available at that same point in time.

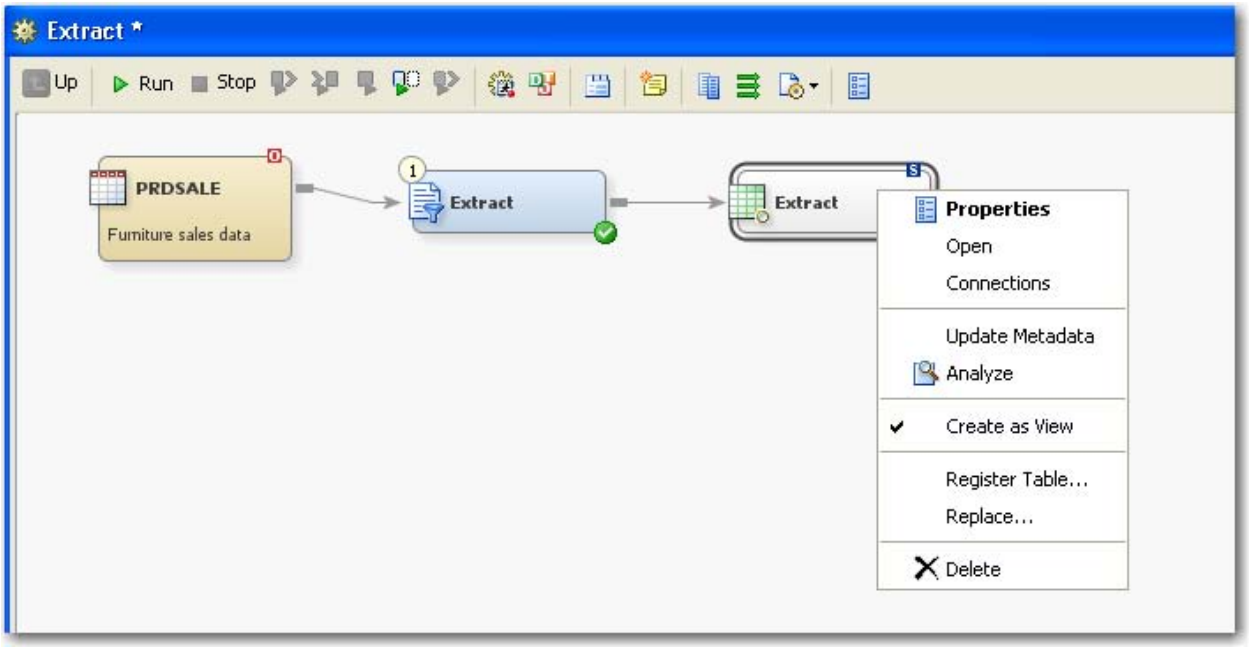


Figure 3: The Extract Transformation by default creates a SAS View

2: LOAD AND STAGE DATA INTO THE DBMS

After extracting the data, we will load the data into a DBMS. When loading the data, consider the choice of load technique and load utility to provide optimal performance with large data volumes.

SAS Data Integration Studio Table Loaders support loading data from any data source that SAS can access into a database management system. In addition, you have several choices on how data gets loaded, from using PROC SQL inserts, to using high speed bulk database loaders, or trickle feed data into the DBMS.

Most databases have load utilities that SAS/ACCESS can leverage. For example, in Oracle environments, SAS can use the parallel loading capabilities of SQL*Loader. The **Table Options** tab in SAS Data Integration Studio 4.21 is a useful place to specify options per job about what type of load technique to use. For example, use Bulkload=YES to use the DBMS load utilities. (See Figure 4.) On the **Options** tab of the LIBNAME object, you can also specify which bulk loader to use.

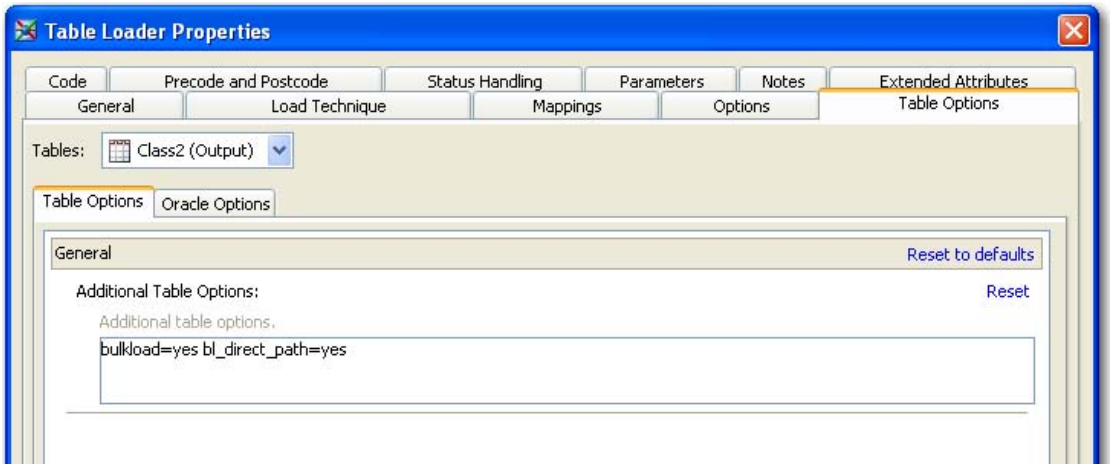


Figure 4: Table Loader's Table Options

Introduced in SAS Data Integration Studio 4.21 is a Teradata Table Loader. It uses the new Teradata Parallel Transporter (TPT) technology to achieve high performance loads. This technology offers many benefits. For example, if a load is interrupted, the Teradata Parallel Transporter allows the user to restart it at the record level.

The Teradata Table Loader also supports other Teradata utilities, including MultiLoad, FastLoad, and TPump. To effectively apply Teradata Table Loader, the user should consider the appropriate technique for the data being loaded (for example, FastLoad for a new, empty table; MultiLoad for adding data to already populated tables; and TPump for continuously moving data into the tables). The user can also direct SAS Data Integration Studio to choose the appropriate method at load time. By setting properties of the Teradata Table Loader, you can affect how the load takes place. (See Figure 5).

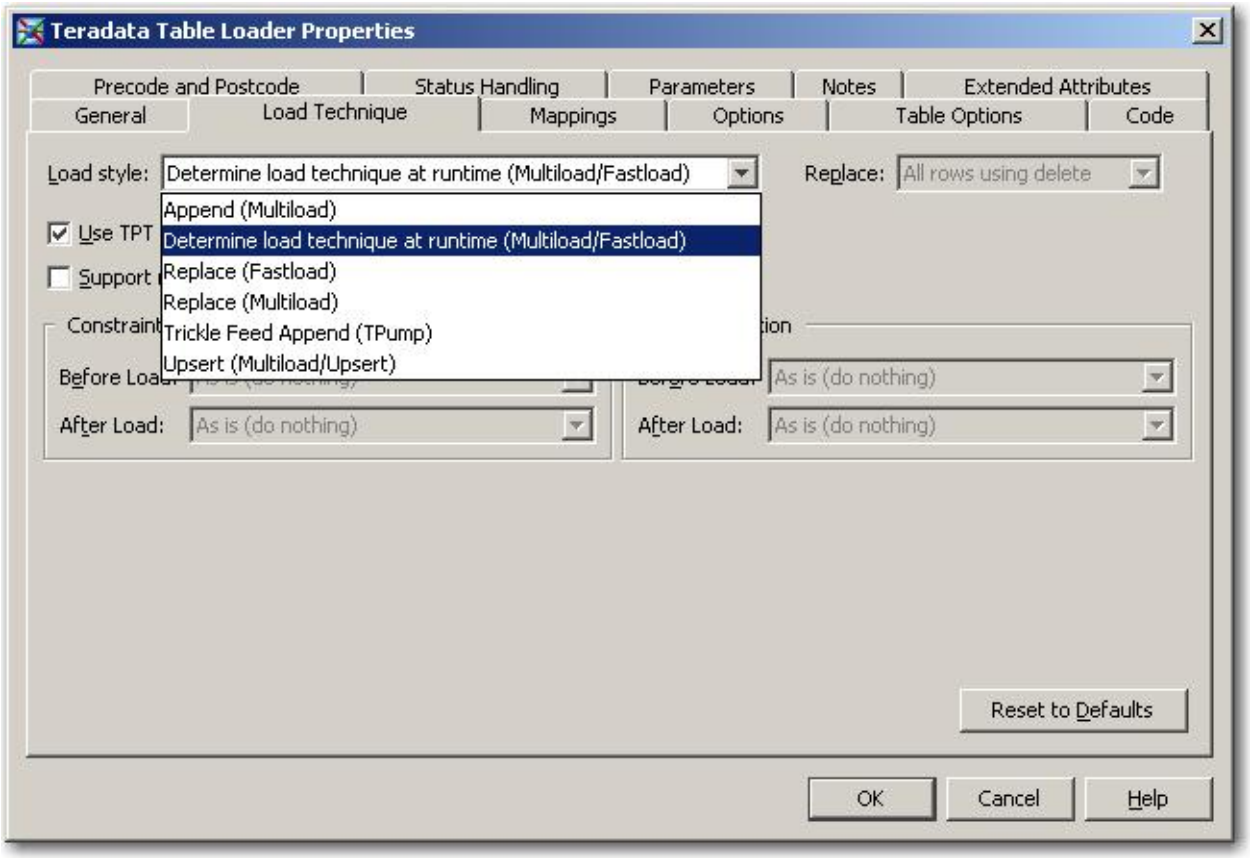


Figure 5: Teradata Table Loader Properties

3: TRANSFORM DATA INSIDE THE DBMS USING THE EXTRACT TRANSFORMATION

Once the data is loaded, SAS Data Integration Studio can leverage SQL transformation logic that runs inside the Database. SAS Data Integration Studio provides two multi-purpose SQL based transformations: Extract, and SQL Join. Both the Extract and SQL Join transformations leverage SQL Pass-Through to produce the SQL that is executed in the DBMS engine.

First let's look at the Extract transformation. With this transformation, you can select data, apply filters, perform aggregations, and transform the data, all by using SQL.

4: USE DBMS SPECIFIC FUNCTIONS TO TRANSFORM DATA

Leverage DBMS-specific functions in your ELT processes, as they will run in the database. The SAS expression builder lists the Teradata functions along with links to usage and help documentation. See Figure 6.

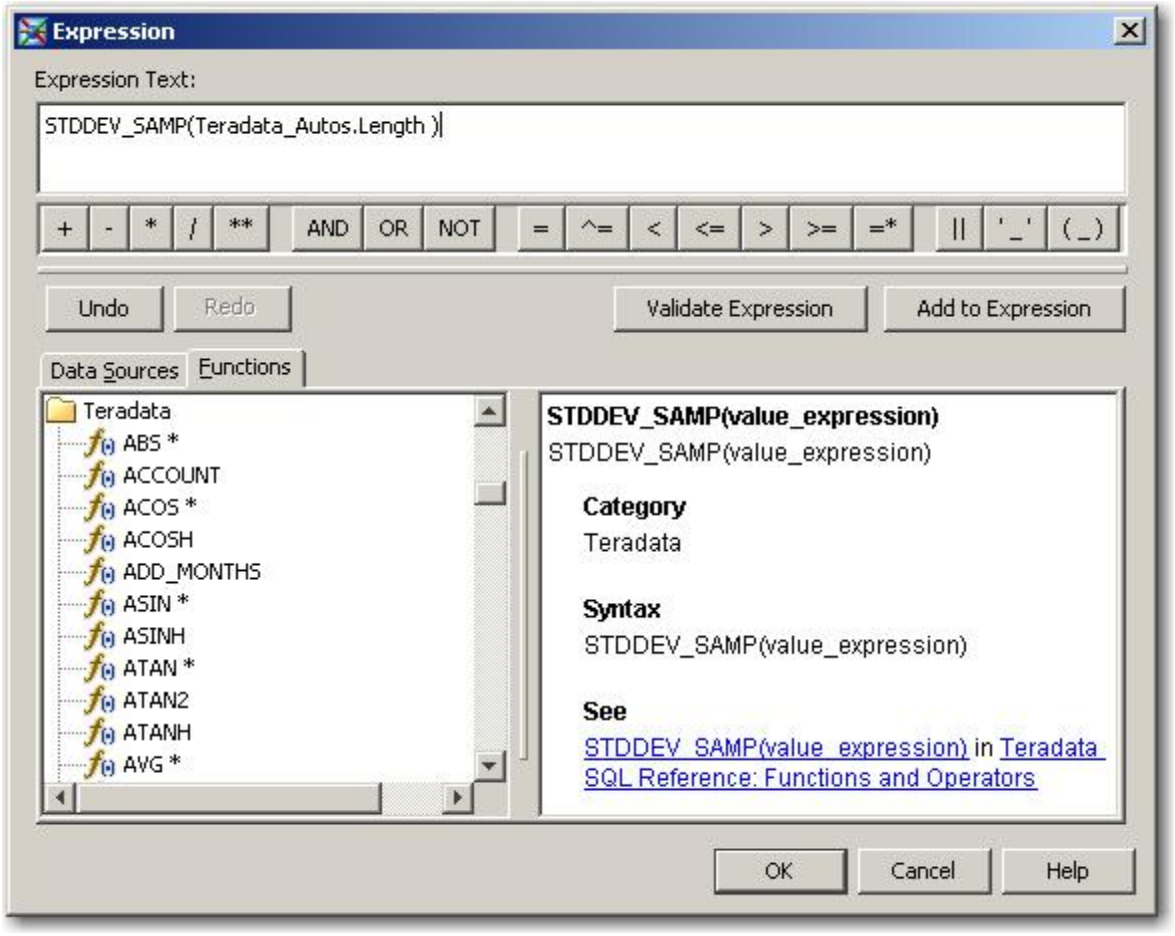


Figure 6: Examples of Database Functions Inside the SAS Expression Builder

5: REDIRECT SAS TEMPORARY TABLES INTO THE DATABASE

By default, the Extract transformation creates a SAS view in the SAS WORK library. When designing ELT processes, you should redirect SAS tables and views to be stored inside the database. That supports the goal of no data movement. You can set this as a property on a table, on a transformation, on an individual job, or as a global option under the **Tools > Options > Code Generation** menu. See Figures 7 and 8.

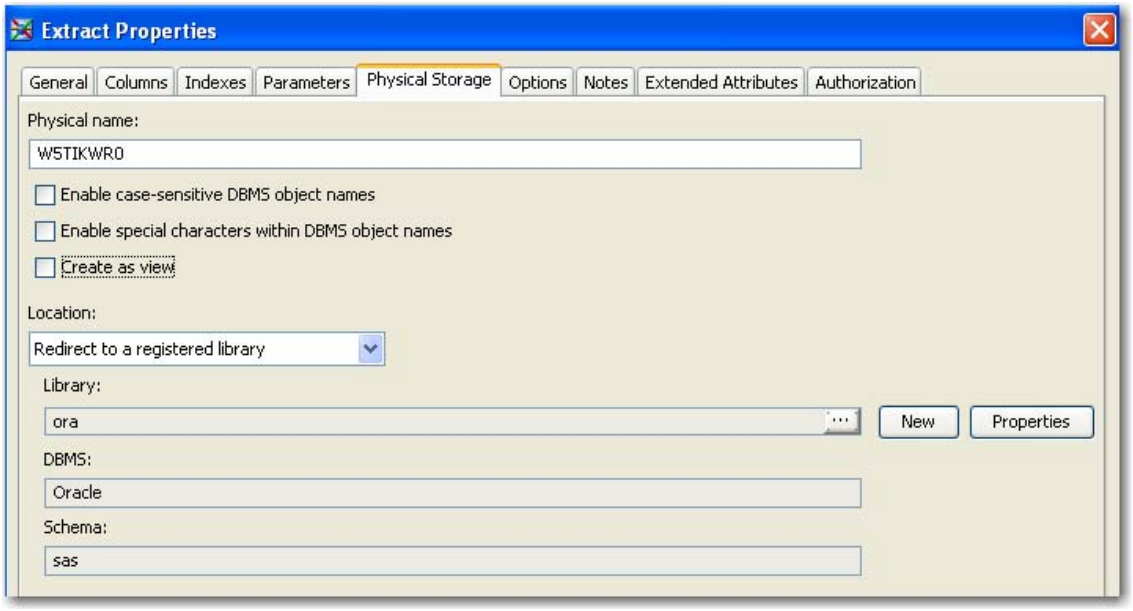


Figure 7: Storing Temporary Tables Inside the Database Instead of SASWORK

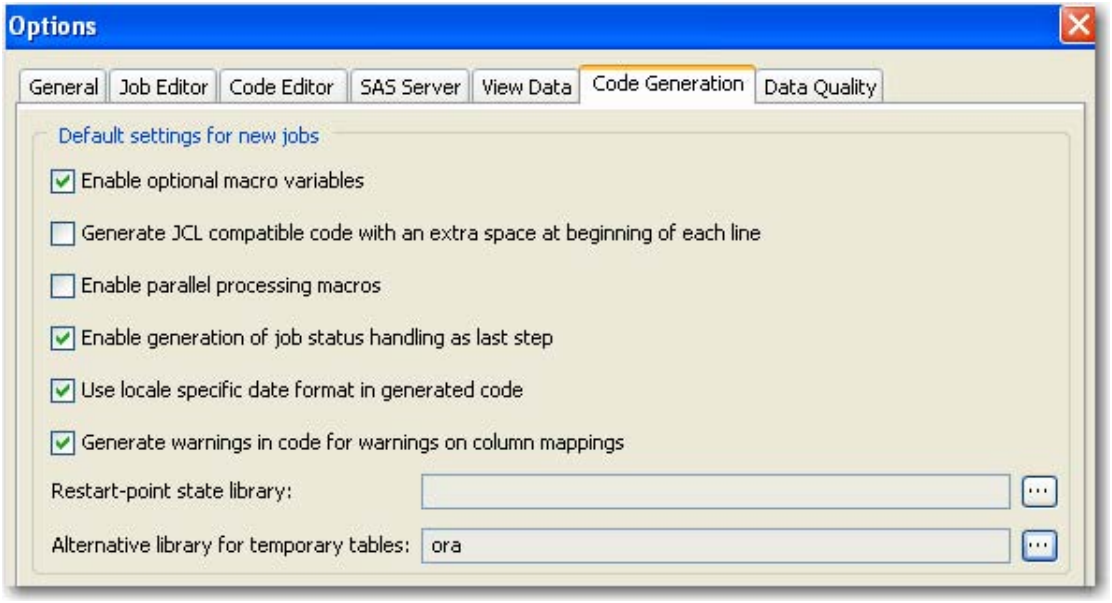


Figure 8: Using the Code Generation Tab to Set an Alternate Library for All Temporary Tables

6: CHECK TO VISUALIZE WHAT WILL RUN INSIDE THE DATABASE

SAS Data Integration Studio provides the flexibility to execute SQL within both the SAS engine and the DBMS engine. When designing ELT processes, you can click on the "Check Database Processing" on the Actions menu, to visualize what will run in the database. SAS analyzes whether any functions or formats unique to SAS are being used, and will report back if the job is ready to run completely inside the DBMS. This helps users design ELT processes that will run completely in the database. See Figures 9 and 10.

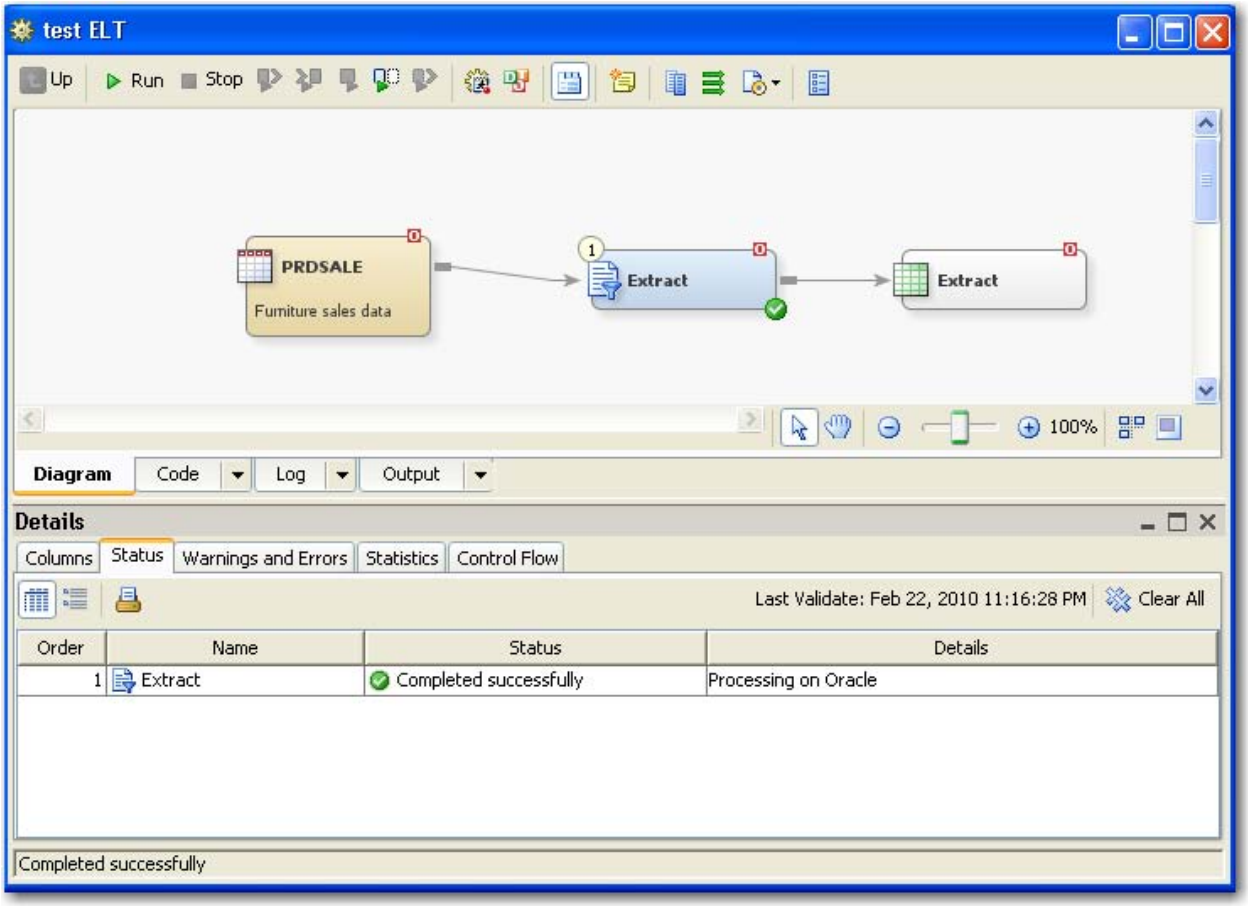


Figure 9: Check database Processing results in an indication of what will run on the database and what will not.

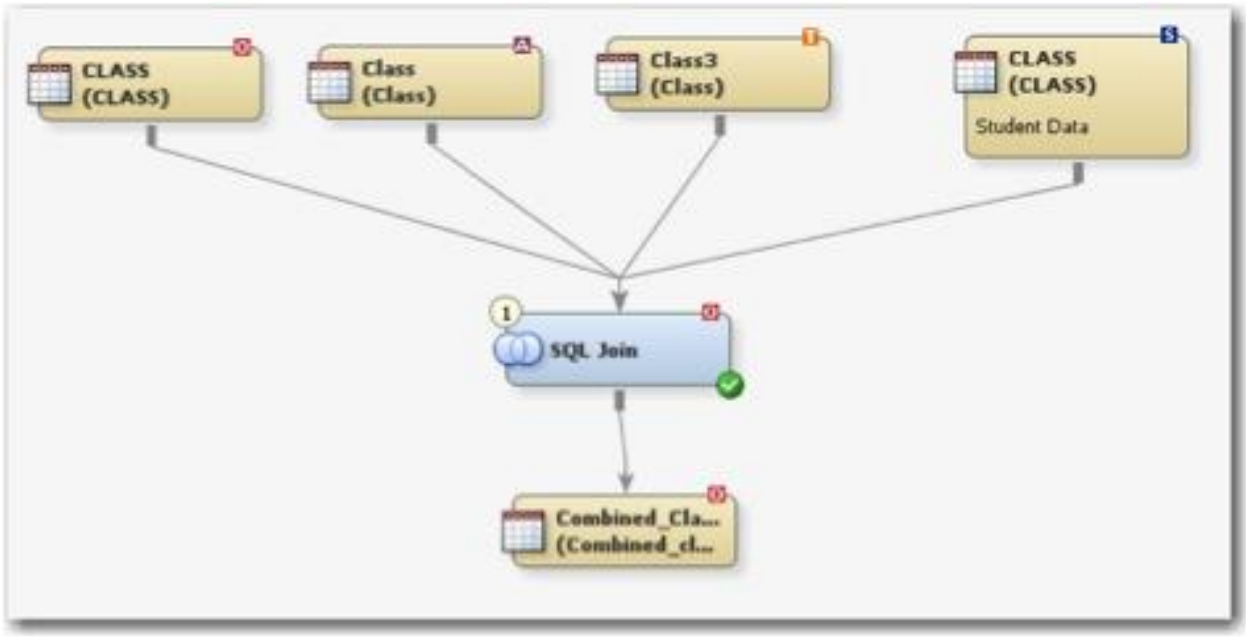


Figure 10: An ELT job: Note the visual indicators that show where data resides, and where processing will take place.

7: TRANSFORM DATA USING SQL JOIN

The SQL Join transformation is a multi-purpose transformation that supports queries, filters, extracts, aggregations, joins, and more. Similar to the Extract transformation, the SQL Join transformation enables you to use expressions to control SQL generated for the database.

When joining two or more tables that are in different databases, you can tell SAS to move the data into a DBMS before doing the join. To set this property, click on the table that you want to stage inside the database. In this case, the Microsoft Access table called Product was selected. Then you set the value of the Upload Library Before SQL field to 'ora', and set the value of the Use Bulkload for Uploading field to 'YES'. See Figure 11.

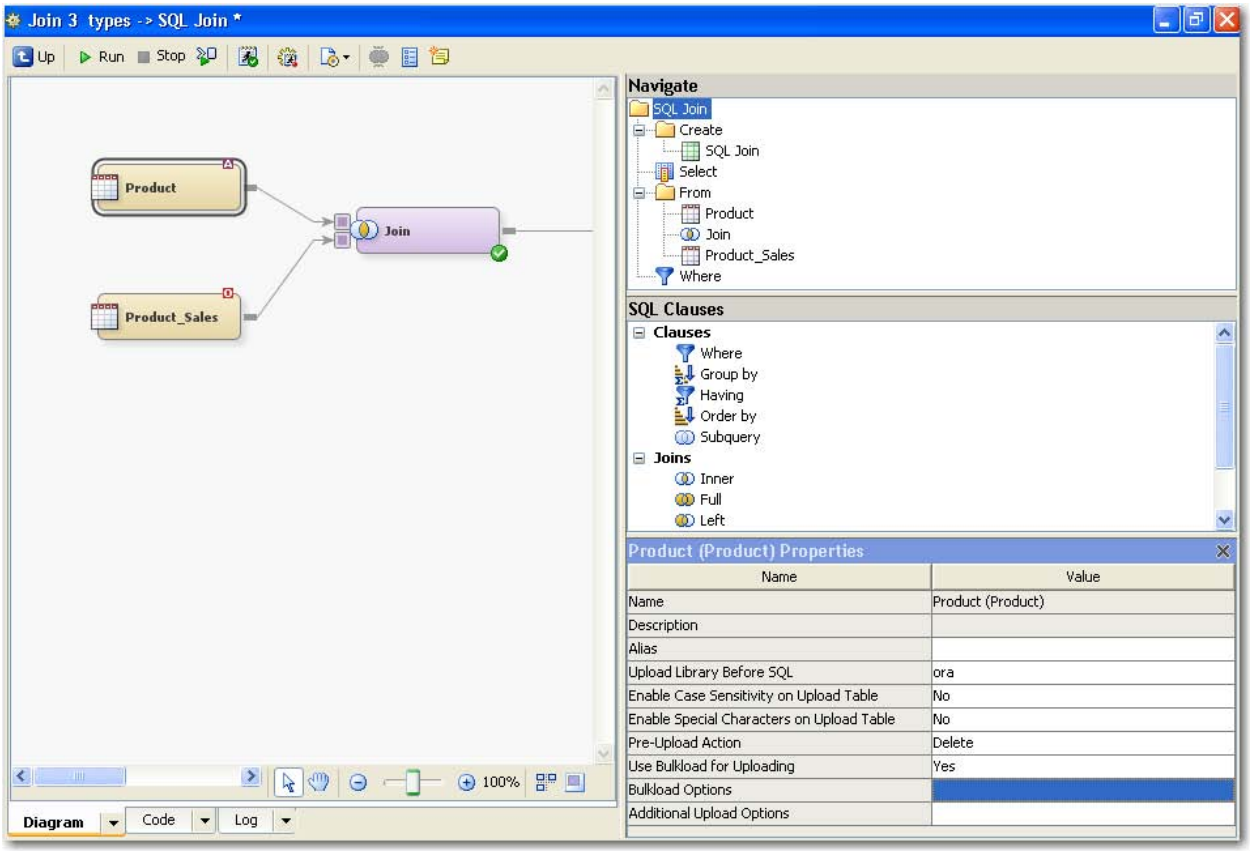


Figure 11: The SQL Join transformation can automatically upload data to the DBMS before doing the join.

On the properties for the SQL Join transformation, you can set the value of the **`Pass Through`** field to **`Yes`**. This will ensure that the SQL is pushed down into the DBMS. See Figure 12.

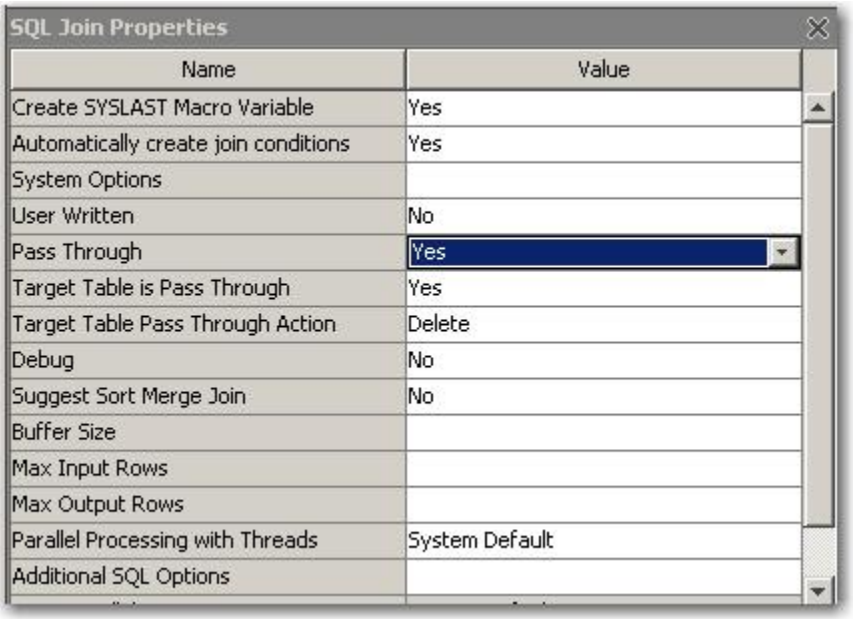


Figure 12: SQL Join Properties

8: TRANSFORM DATA USING A DATABASE USER-DEFINED FUNCTION

SAS Data Integration Studio can leverage user-defined functions (UDFs) or Vendor-defined functions that reside in your DBMS. These functions can do many different things. For example, some SAS customers use functions that are created using the SAS Scoring Accelerator for Teradata to do scoring inside the Teradata database.

To show how to use database UDFs from SAS Data Integration Studio, I've created a simple UDF inside Oracle called RiskRating.

```
Oracle UDF named RiskRating

BEGIN
IF MOD(variable1, 2) = 0 THEN
  RETURN 'High Risk';
END IF;
IF MOD(variable1, 2) <> 0 THEN
  RETURN 'Low Risk';
END IF;
END;
```

9: EXTEND THE SAS/ACCESS SQL DICTIONARY

Before we can use the UDFs in the DBMS, we have to tell SAS about them. SAS/ACCESS software has a SQL dictionary that lists the SAS functions and how they map to DBMS functions. You can extend the list of DBMS functions to include DBMS user-defined functions.

The following code, lists the functions currently mapped to DBMS functions:

```
/* list the existing functions in the dictionary */
libname ora oracle user=sas pw=sas sql_functions_copy=saslog;
```

The output of this program is the following:

```
1790  libname ora oracle user=sas pw=XXX sql_functions_copy=saslog;

 SAS Function Mappings provided by SAS ACCESS engine:
    SAS              DBMS
FUNCTION NAME  FUNCTION NAME
-------------  -------------
   ABS            ABS
   ARCOS          ACOS
   ARSIN          ASIN
   ATAN           ATAN
   CEIL           CEIL
   COS            COS
   COSH           COSH
   DATETIME       SYSDATE
   EXP            EXP
   FLOOR          FLOOR
   LOG            LN
   LOG10          LOG
   LOG2           LOG
```

Adding new UDFs to the SAS/ACCESS SQL Dictionary:

```
/* Add a function called Riskrating to SQL dictionary*/
data work.newfunc;
  sasfuncname='RISKRATING';
  sasfuncnamelen=10;
  dbmsfuncname='RISKRATING';
  dbmsfuncnamelen=10;
  FUNCTION_CATEGORY = "";
  FUNC_USAGE_CONTEXT = "";
  FUNCTION_RETURNTYP = "";
  FUNCTION_NUM_ARGS = 1;
  CONVERT_ARGS = 0;
  ENGINEINDEX = 0;
  output;
 run;

libname ora oracle user=sas pw=sas
sql_functions="EXTERNAL_APPEND=work.newfunc"
  sql_functions_copy=saslog;
```

This SAS code can be inserted as pre-code before the extract runs. Alternatively, you can make it available to your entire environment by adding as an option to the LIBNAME object in metadata. See Figure 13.

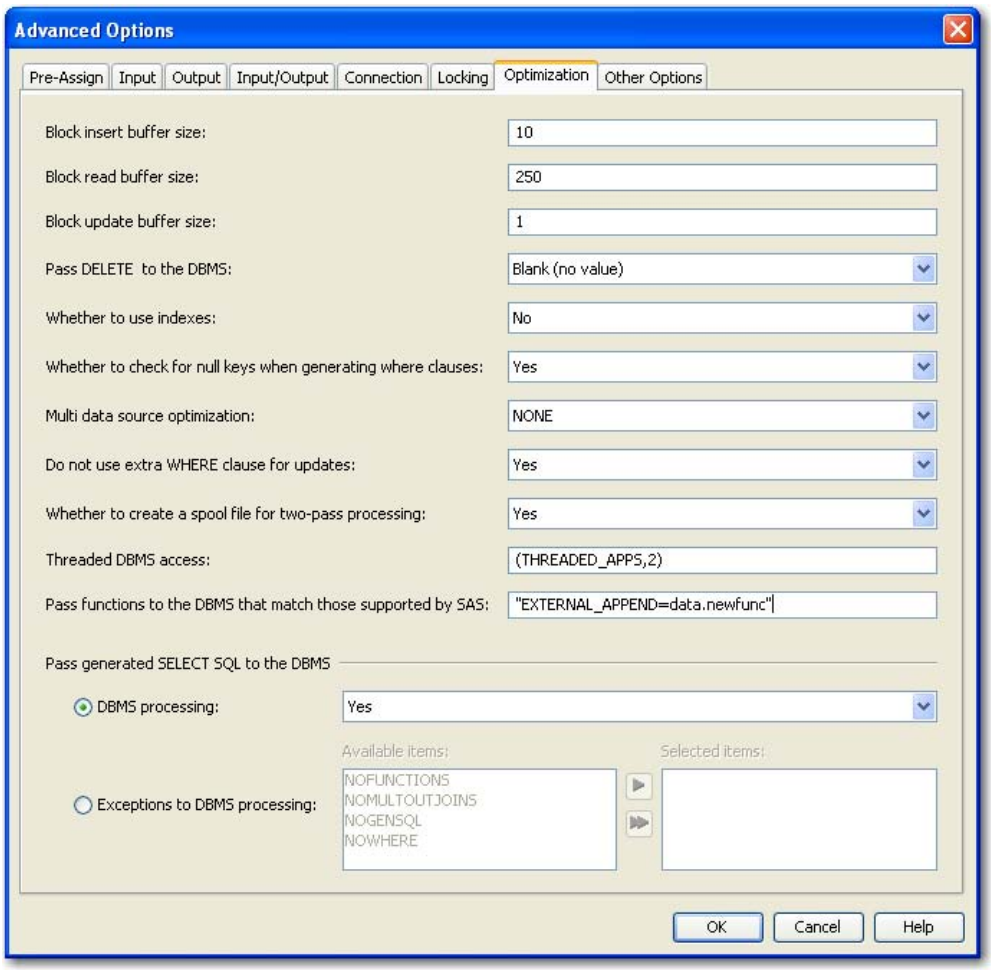


Figure 13: Extending the List of Available Functions for a LIBNAME Object

10: ADD NEW DATABASE FUNCTIONS TO THE EXPRESSION BUILDER FUNCTION LIST

You can add additional database functions to the Expression builder function list in SAS Data Integration Studio 4.21. This involves two steps.

First, edit the distudio.ini file typically found in the installation directory. On my PC, it can be found at: C:\Program Files\SAS\SASDataIntegrationStudio\4.2\distudio.ini

Add the following line to the distudio.ini file:

```
JavaArgs_13=-DExpressionPanelAdditionalFunctionsPath=
"Additional_Functions.xml"
```

Next, we will create a file called Additional_Functions.xml using an editor such as notepad. Add the following text to the file Additional_Functions.xml to add user defined function called RISKRATING to a folder called Oracle:

```
<Categories>
    <Category Name="Oracle" DisplayName="Oracle">
        <Function Name="RiskRating" DisplayName="RiskRating"
Description="Oracle RiskRating User Defined function"
Insert="RiskRating(&lt;n&gt;)" ToolTip="This string is not used"
HelpFileName=""/>
    </Category>
```

```
</Categories>
```

After saving this file, restart SAS Data Integration Studio. The next time you bring up the expression builder function list, you will see the new functions listed. See Figure 14 and Figure 15.

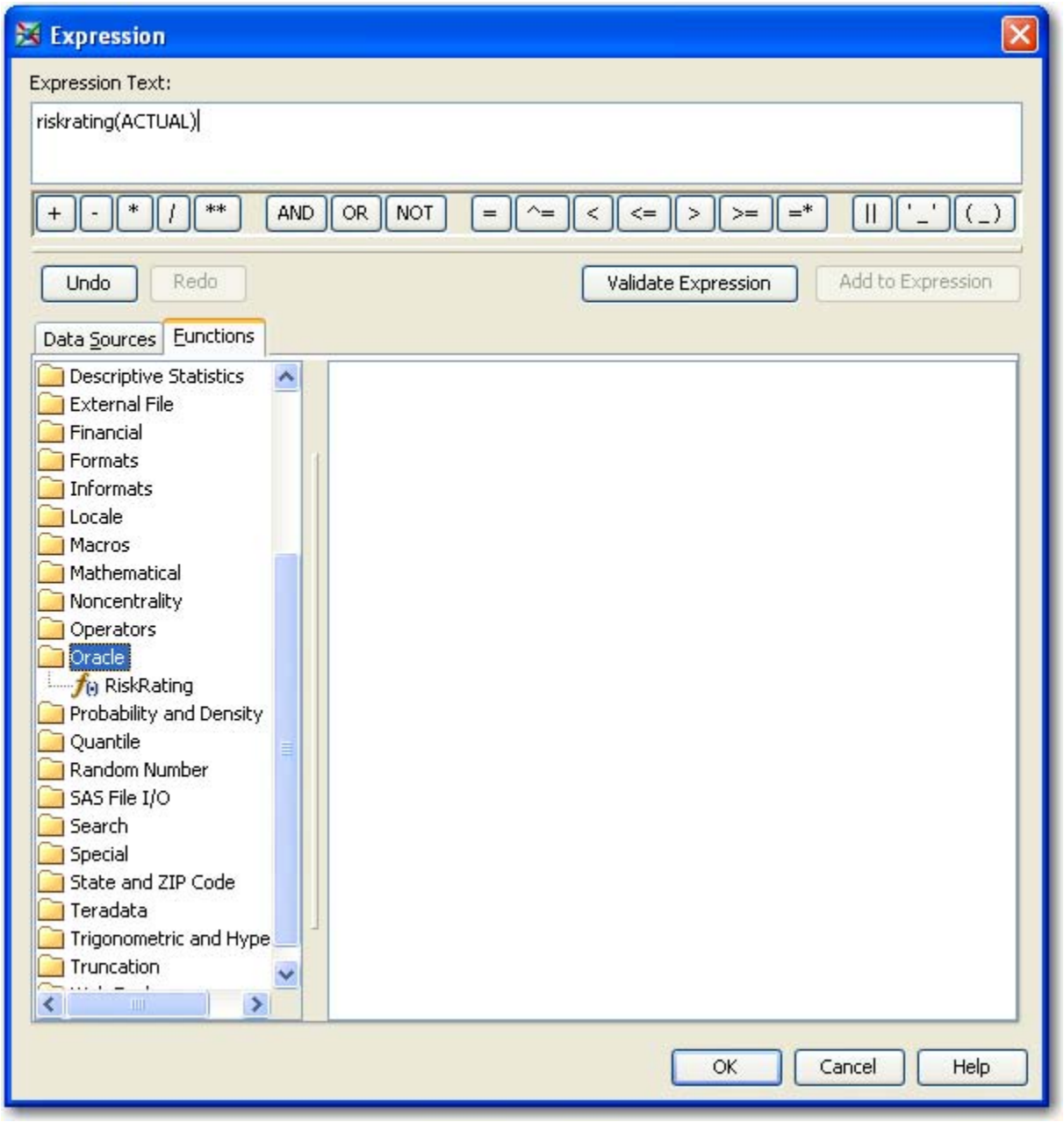


Figure 14: Expression Builder Function List Shows Newly Added User-Defined Function in Oracle Called RiskRating.

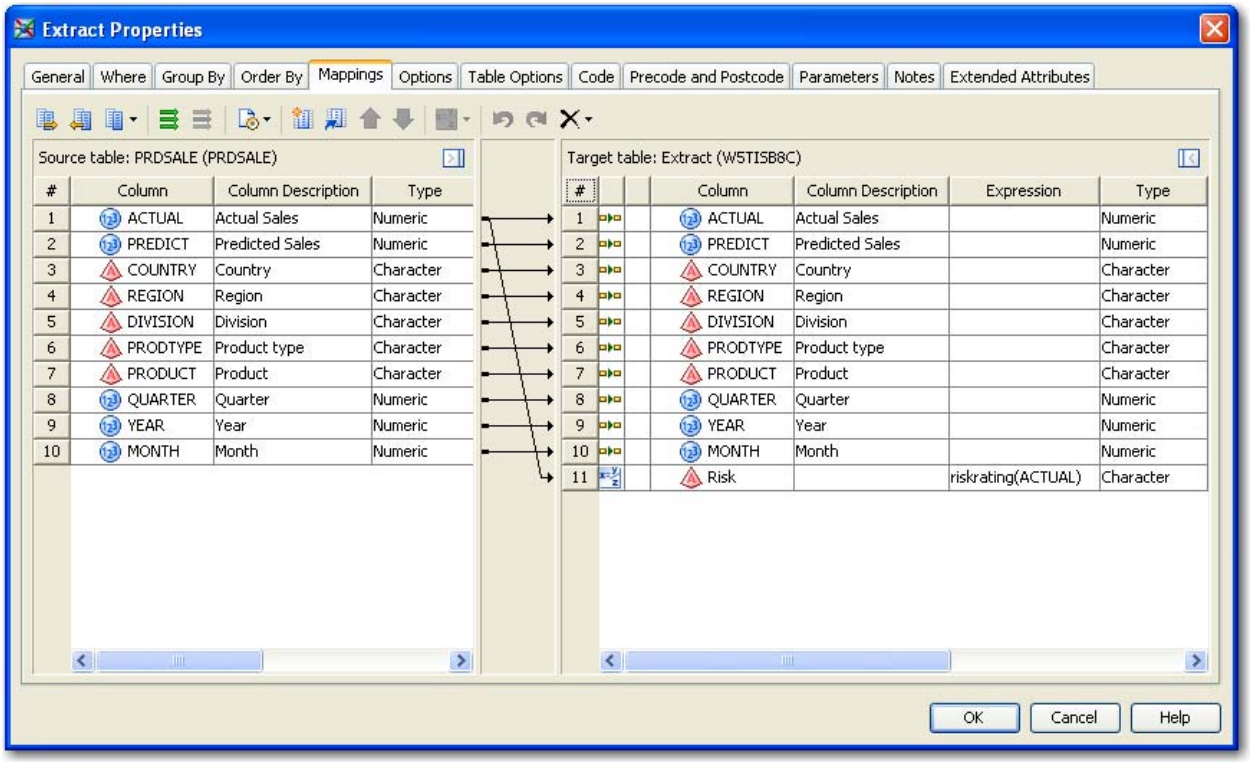


Figure 15: Using the Newly Added Function: RiskRating

11: COLLECTING INFORMATION ABOUT WHICH SQL IS PUSHED DOWN TO THE DATABASE

SAS provides options and information that you can use and review to enable better performance. The following code examples show how to collect information that can help identify DBMS performance issues. This information can also be used to validate the strategy you are using to improve the interaction between SAS and your DBMS.

You can set SAS options to collect pushdown information. The following SASTRACE statement shows the SQL that is being passed to the DBMS. The following option turns on the SQL tracing feature:

```
options sastrace=',,,d' sastraceloc=saslog nostsuffix;
```

Identifying SQL flowing through the ACCESS engine can help identify SQL issues, especially if the SQL were part of a code generation process or PROC SQL Implicit Pass-Through. You can use the SQL captured in the log as input into DBMS query explain tools to help you gather more information as part of your analysis.

You can set options to collect timing information. The following SASTRACE statement gives you processing times inside the DBMS. Using these measurements will help you identify SQL that needs further refinement. The following option turns on the timing feature:

```
options sastrace=',,,ts' sastraceloc=saslog nostsuffix;
```

Timing information, along with assurances that SQL intended to be processed in the database is making it to the Database, are a good foundation for process improvement. The SAS/ACCESS documentation identifies and explains SASTRACE options and functionality.

The following statement turns off the SASTRACE option:

```
options sastrace=off;
```

CONCLUSION

We've demonstrated that we can implement ELT processes using SAS Data Integration Studio. So what's the impact of running an ELT process versus an ETL process? It really depends on the scenario. ELT is not the answer to everything. Performance differences depend on many variables including size and speed of your SAS server, size and speed of your DBMS platform, data volumes being moved or not moved, and other variables.

Using ELT processes can significantly improve performance of some long running data integration jobs. The following example job shows 3 different ways of using SQL Join. (See Figure 16.) The first one runs completely in SAS with data being extracted into SAS and loaded back into the database. The second one uploads data from one of the source systems and does the join in the database. The third example shows data pre-staged in the DBMS, and the joins are all running in database, with zero data movement to SAS.

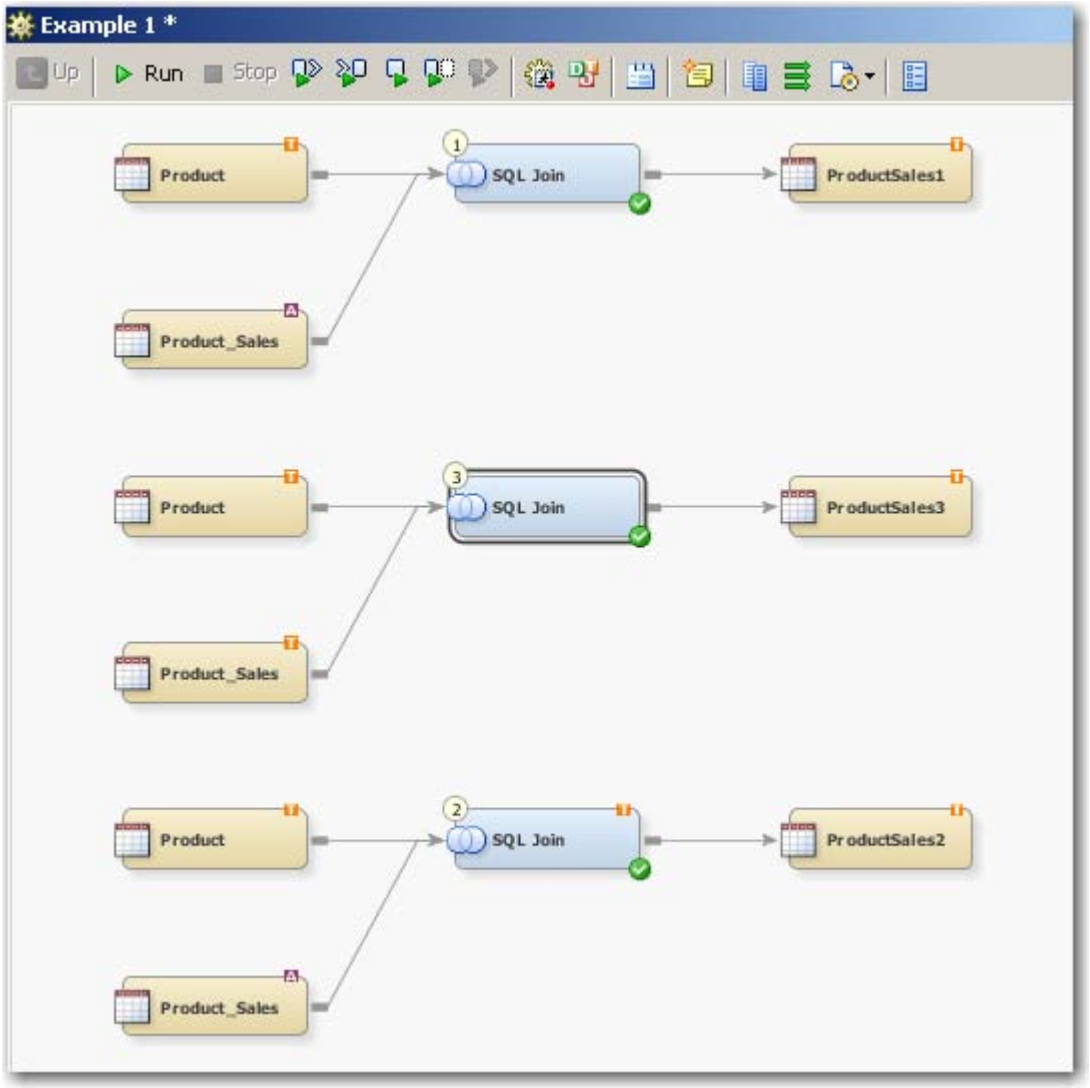
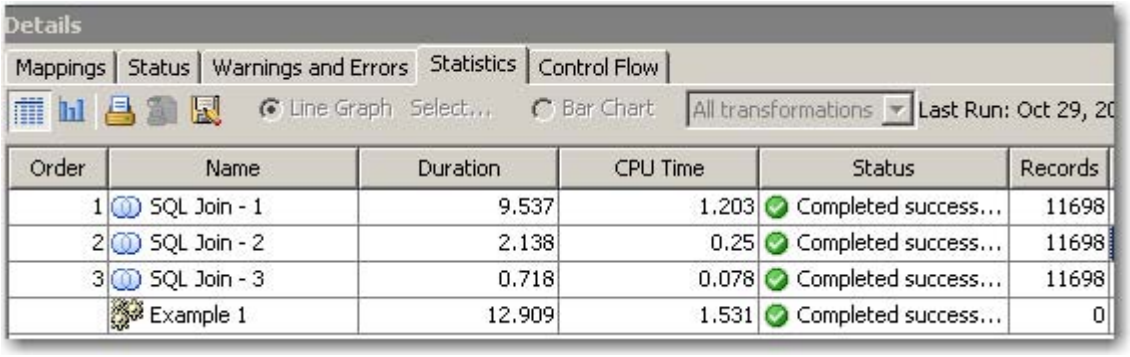


Figure 16: Job Showing Three Types of Joins and Comparing ELT Versus ETL

In this particular job, the job statistics suggest we can see significant improvements in run times, as well as CPU usage and I/O using the ELT technique. See Figure 17.



Order	Name	Duration	CPU Time	Status	Records
1	SQL Join - 1	9.537	1.203	Completed success...	11698
2	SQL Join - 2	2.138	0.25	Completed success...	11698
3	SQL Join - 3	0.718	0.078	Completed success...	11698
	Example 1	12.909	1.531	Completed success...	0

Figure 17: Job Statistics Showing Example Run Times of Three Different Joins

What does the future hold? SAS continues to enhance and develop SAS Data Integration Studio to support more in database processing. In the next version, we are planning to include additional specialized Table loaders for specific databases, and we are looking to include more native database functions in the expression builder function library.

REFERENCES

Plemmons, Howard. 2009. "Top Ten SAS® DBMS Performance Boosters for 2009." *Proceedings of the SAS Global Forum 2009 Conference.* Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/sgf09/309-2009.pdf.

SAS Institute Inc. 2009. *SAS Data Integration Studio 4.21 Users Guide* . Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/onlinedoc/etls/index.html.

SAS Institute Inc. 2009. "SQL_FUNCTIONS= LIBNAME Option." In *SAS/ACCESS 9.2 for Relational Databases Reference, Second Edition.* Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/cdl/en/acreldb/61890/HTML/default/a002238044.htm.

CONTACT INFORMATION

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

Jeff Stander
SAS Institute Inc.
801 International Parkway, Suite 500
Lake Mary, FL 32746
E-mail: jeff.stander@sas.com
Web: www.sas.com