Paper 078-2013

SAS® Data Integration Studio: The 30-Day Plan

John Heaton, Heritage Bank

ABSTRACT

Deciding whether to use an Extraction, Transformation and Loading tool on your Business Intelligence project or to use code is normally a much debated topic. Once this decision has been made and you have completed some form of training you should be ready to begin defining you jobs and extracting data into your solution, Correct? Well maybe not, you may want to spend a little time planning your next steps before you start to develop a large number of jobs.

The paper outlines the typical steps needed to design your ETL architecture in order to be successful when implementing an ETL tool like SAS® DI Studio.

INTRODUCTION

Business Intelligence solutions are generally made up of three core components:

1. Data Integration
2. Reporting
3. Analytics

Tools like SAS® DI Studio significantly aid in the design and development of the data integration layer. Often when companies purchase an ETL tool they immediately want to begin development without any planning. This is the fastest way to diminish the investment in an ETL tool and create an unmanageable environment. Over the years a standard blueprint has emerged for business intelligence solutions with regards to the data layer. The data layer is typically broken into three main sections.

1. Staging – area for data to be placed before it is transformed and absorbed into the warehouse.
2. Foundation Layer – a data model which supports the information collected through the different business processes within the organization.
3. Access and Performance – a data model which is optimized for information delivery and interrogation.

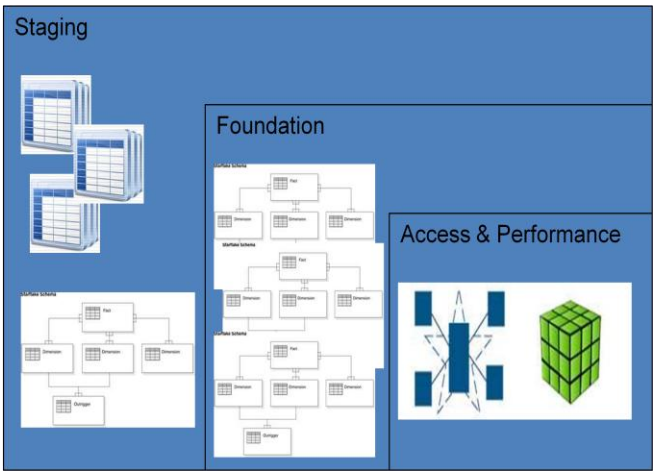


Figure 1. Data Integration Layer

As can be seen from the figure above, the different layers perform different functions. The staging area is a before and after image of the data, the foundation layer captures the information in its atomic format modeling the relationship between the information found in the different business process and the access and performance layer models the data in an appropriate manner based on the questions posed by the business and the different tools accessing the information. The access and performance layer may change multiple times over the lifecycle of the solution as business requires different information and analysis. The foundation layer should be fairly static as it is modeled on the business processes and the information captured during the business process.

ETL ARCHITECTURE

When we think of architecture we normally think of technical servers and how they are implemented within the environment. The ETL architecture does take some of this into account but is more focused on how, where and why information is stored and moved from one location to another. It is important to understand the flow of information through your solution and configure the necessary components to support your ETL architecture before starting to develop jobs. The first in the series of these main decisions is to map out what libraries you will need and how they will be used.

LIBRARIES AND LOCATIONS

SAS® uses libraries to abstract the location of the storage from the underlying technology. When developing the ETL architecture it is important to understand how to make use of the different types of libraries within your environment to map efficiently to the different data layers.

To outline the principles lets use a fictitious case study on Corp X whose primary information is stored within a n ERP (Enterprise Resource and Planning, Oracle, SAP, Peoplesoft etc) and a CRM (Customer Relationship Management – Salesforce.com, Oracle, SAP) within a relational database. They have made the decision to build a business intelligence solution and store the information within a central corporate repository within a relational database.

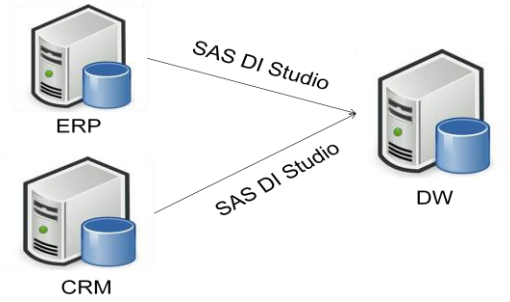


Figure 2. Proposed ETL Architecture

The figure above outlines the simplified proposed solution. In this solution the entire data layer would exist on the Relational DW database. From an initial glance this may be an acceptable solution. But let's investigate this a little further.

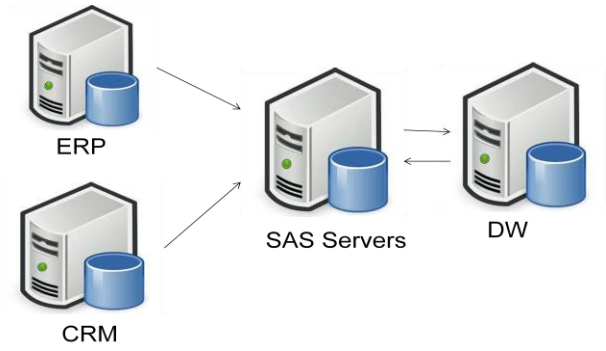


Figure 3. Actual ETL Architecture

So in reality the SAS Servers sit between the source systems and the target data warehouse. When information is extracted from the source environments it is read into the SAS Sever and written from the SAS Server to the DW server. This doubles the number of network trips between the proposed architecture in figure 2 and the actual architecture in figure 3.

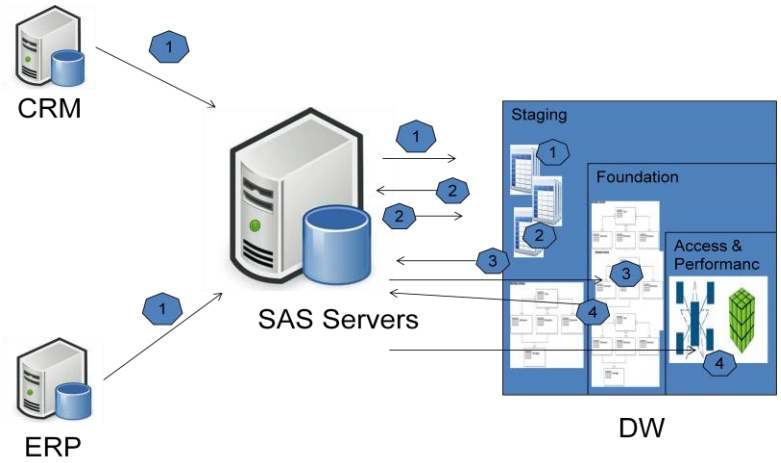


Figure 4. Actual ETL data movement

So in the first trip the information is extracted from the source and then written to the DW Server (1). Secondly information in the DW Server will be extracted from the DW Server (3) to the SAS Server, transformed and manipulated and then written back to the DW server to the foundation layer. Information from the foundation layer (4) will be extracted from the DW Server to the SAS Server, aggregated and manipulated and then written back to the DW Server to reside in the Access and Performance Layer (4). The above ETL architecture would greatly increase the network traffic and the number of times information is moved outside of a server.

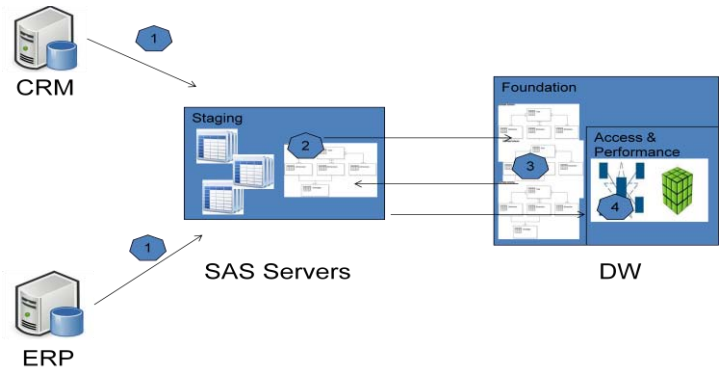


Figure 5. Staging Area on SAS Server

By moving the staging area onto the SAS Server and using SAS® for the data storage in place of a relational data you greatly reduce the number of network trips and the number of times information is moved. An added benefit to this approach is that the SAS Server can do all the processing internal to the server without having to read or write data across the network greatly benefitting performance. Depending on the Access and Performance layer you may also decided to use SAS Base Library located in the SAS Server. The reasons for this may include the inclusion of OLAP cubes within the architecture. It is important to prototype these and understand the implications.

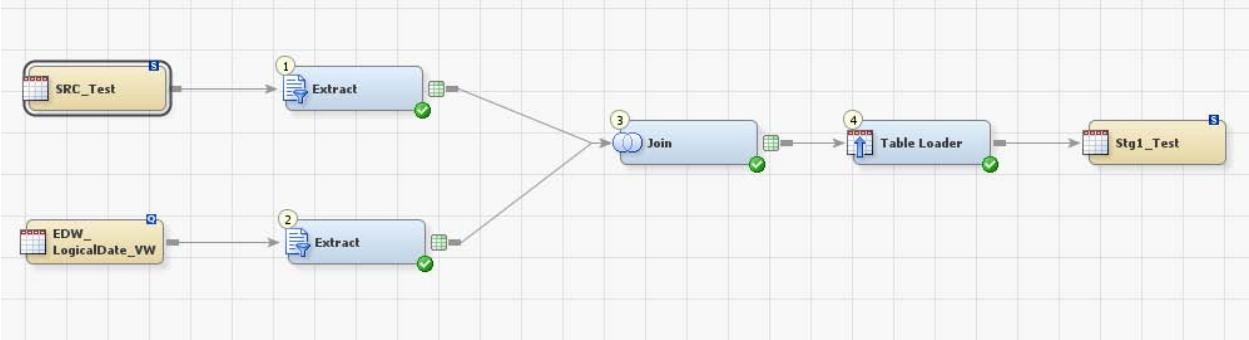
Therefore in the above ETL architecture you would define libraries from the following:

1. Source Libraries – Database Library
    a. ERP
    b. CRM
2. Staging Library – SAS Base Library
3. Foundation Library – Database Library
4. Access and Performance Library – SAS Base Library and or Database Library

ETL architecture has a single goal which is to reduce the numbers or records read and written. By reducing the dataset and frequency you read your data set you will reduce the time to transform and load information. Setting up your libraries efficiently to match an optimal architecture will reduce the amount of rework in the future and standardize the way information flows through the solution.

LOGICAL DATES

Information is loaded from different source systems at potentially different times. In your solution you will need to develop a concept of a logical date. The logical date represents the period of time for which the information is relevant. For example, extracting information from an ERP system from 10pm to 2am in the morning the logical date from the system may be set to the start of the extract to signify the close of business day. The logical date is added to each record in the staging area so that there is a record when the information is extracted and to which logical date it belongs. By following this technique datasets can be extracted multiple times, stored within the staging area and then processed chronologically based on the logical date. Depending on the frequency of the extract the logical date could remain at the date level or may be a date timestamp.



Display 1. SAS® DI Studio Job with Logical Date

As can be seen from the job above a table exists with my current and last logical dates. These are cross joined and loaded into the target table. The added column being LogicalDate is included in the dataset and therefore allows to keep multiple versions of the same data in the table.

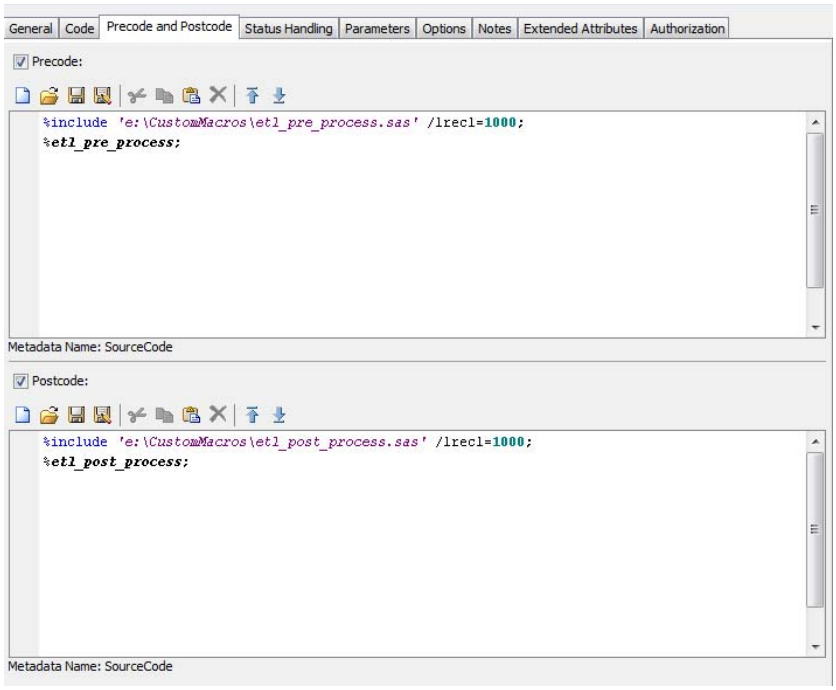
PRE AND POST PROCESS CODE

ETL Jobs typically will perform some form of pre and or post processing. Pre and Post processing is easier to build into the ETL initially rather than have to add it at a later date. Initially this may only be stub code (code which does not perform a function), however in future there may be a requirement to perform some pre and post actions. In order to do this the SAS Autos needs to be configured. In the reference section please refer to paper outlined to configure the initial setup. Once completed remember to get the drive shared so that you can map to it within Windows Explorer. This will allow you to write custom macros to use as pre and post processes.

```
%macro etl_pre_process();
LIBNAME STAGE meta library="Stage";

%global
runJobTime
runJobDate
runJobStatus
runRecCount
LogicalRunDate
;

%let runJobTime = %sysfunc(time(), time10.2);
%let runJobDate=%sysfunc(date(),yymmdd10.);
%let runJobStatus= In Progress;
%let runRecCount =0;
%let runDateTime = %sysfunc(DATETIME(),datetime22.3);
proc sql;
```

```
  insert into STAGE.Jobs
   (Job_ID ,
    Job_Date ,
    Job_RunDate,
    Job_Name  ,
    Job_Start_Time ,
    Job_ETLUser ,
    Job_MetaUser ,
    Job_Server  ,
    Job_RC ,
    Job_Status )
    Values
    ("&JOBID",
       "&runJobDate",
       "&runDateTime"DT,
   "&ETLS_JOBNAME",
       "&runJobTime",
       "&ETLS_USERID",
       "&ETLS_USERID",
       "&IOMSERVER",
       &JOB_RC,
       "&runJobStatus");
*/
quit;
%mend etl_pre_process;


%macro etl_post_process();
LIBNAME STAGE meta library="Stage";

%local
l_endJobTime
;

%let l_endJobTime = %sysfunc(time(), time10.2);
%let runJobStatus= Completed;


proc sql;

  update STAGE.Jobs
       set Job_End_Time = "&l_endJobTime",
        Job_Records = &runRecCount,
         Job_Status = "&runJobStatus"
   where
       Job_ID = "&JOBID" AND
       Job_Date = "&runJobDate" AND
       Job_Name =  "&ETLS_JOBNAME" and
    Job_Start_Time =       "&runJobTime";

quit;
%mend etl_post_process;
```

The pre and post process macros above track the time a job was started and the time it completed. These macros are included in each job by configuring the pre and post process.



Display 2. SAS® DI Studio Job Properties

INCREMENTAL OR FULL DATA LOADS

In most cases data warehouse solutions load only information which has changed since the last load. There are exceptions to this, which is where you need a snapshot of information on a period by period basis. Different data sources may or may not have the capability to provide only changed data. There are an array of different technologies available to implement as well to detect and provide change data. Loading only change data is preferable to reloading entire data sets on a period basis. SAS® DI Studio does have some transforms (Change Data Capture) to take advantage of certain database technologies if implements. For others a more manual approach needs to be developed and included into the jobs.

If the dataset has a date which can be used to detect updated or new data since the last load this can be included into the jobs to extract only changed or new information. If no columns are available to detect changes set operations such as minus are effective in determining which records have changed since the last extract. Extracting data based on a date should be done and filtered at the source. Set operations can be costly depending on the dataset and requires a copy of the previous load. For this reason it is more efficient to do this within the staging area.

You will need to review each dataset individually and determine how to detect, extract and load changes. Either way you will need to include the logical date so that you can identify the dataset to the correct period.

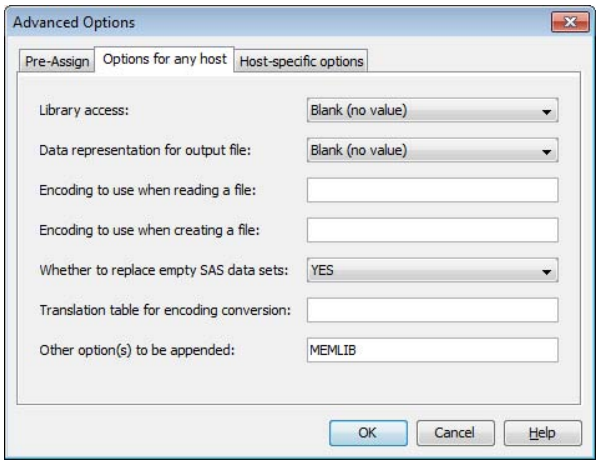
SAS DISK LIBRARY OR MEMORY LIBRARY

SAS has the capability to create a library based in memory. A memory based library is huge performance benefit to place transformations into memory (joins etc). These cannot be used if you wish to use the restart features as memory based objects are only available whilst the job is executing. To a memory based library you need to add an entry into the sasv9_usermods.cfg on the Compute Server. The default is 2GB which is normally insufficient. Below is an example to increase the size from 2GB to 16GB.

```
/*
* sasv9_usermods.cfg
*
* This config file extends options set in sasv9.cfg. Place your site-specific
* options in this file. Any options included in this file are common across
* all server components in this application server.
*
```

```
* Do NOT modify the sasv9.cfg file.
*
*/
-MEMMAXSZ 16G
```

To configure the library you would create a normal SAS based library and then enter MEMLIB under Options -> Advanced Options -> Options for any host -> Other option(s) to be appended.



Display 3. SAS® Management Console New Library Wizard Advanced Options

Once this is done the LIBNAME Statement should be as follows:

```
LIBNAME SASSTGM  BASE "E:\Stage" MEMLIB ;
```

SCDI OR SCDII LOADER

For dimension loading SAS® DI Studio has and SCDI and a SCDII loader. The SCDII loader can perform all the operations of the SCDI loader and more but not vice versa. For this reason the SCDII loader should be used as default for all dimensions whether type SCDI or SCDII. To enable this you need to have a few pre requisites.

1. Business key – used to uniquely identify a record, this is the natural primary key and may be one or more columns. When a new value is detected then a new record is created automatically.
2. Detect Changes – these are the columns which are in addition to the business key on which if changes have occurred from the last load then a new record will also be created.
3. Allow multiple updates per day – this is a setting under options and should be set to yes. If not only changes from the first time it is run in a day will be registered and applied.

To use the SCDII as an SCDI loader your business key and detect changes column will be the same physical column. However the transform will not allow this. So in order to accomplish this I would recommend adding a column to all your dimension tables called Business_Key and making this column equal to your natural key column which will be used to detect changes. For example if I have a product table with a product_code. When new product codes are added I would like a new record. In my staging table I would create a new column called Business_Key. I would map my product_code column to my product_code and to my business_key. This way I have two columns to use for the SCDII loader and can accomplish the task.

This may seem like additional work, however in future if you need to update your dimension to track changes it is as simple as reconfiguring the SCDII transformation and not restating you dimension, data and rebuilding your mapping.

DATA MODELING

The style and technique used to model your data warehouse is an essential decision. There are two industry accepted modeling styles namely:

1. 3rd Normal Form
2. Star Schema

As the ETL architecture has different layers, different modeling techniques and styles can be used to store information as it flows through the ETL architecture. The following are suggested styles for each data layer:

1. Staging
    a. Source Stage – initial portion of the stage area which represents the information as presented in the source system. This includes the logical date so that multiple copies of the source data can be stored within a single table.
    b. Stage 1 – initial portion of the staging area used to represent the data which will be processed. This includes only a single copy of information for a single logical date and is direct mirror of the table structure within Source Stage.
    c. Stage 2 – final staging area used to represent the data after it has been transformed. This is a direct mirror of the data model within the foundation area.
2. Foundation – this area represents the information collected through the different business processes. This is normally modeled as a 3rd normal form or star schema data model. This is purely based on preferences and standards within the organization. Using a star schema does make it slightly easier to aggregate data into the Access and Performance area.
3. Access and Performance – this area is optimized for information delivery and should be modeled in the most optimal way for the tools which you will use to access the information. For SAS® the best option for this layer is a star schema's model and large renormalized aggregated tables. OLAP and ROLAP cubes may be included into this area as well. Whenever possible Information Maps should be used to abstract the physical tables from the users so that any changes can be managed and downstream objects do not become invalid.

By modeling your source stage and initial staging area on your source and the final staging area based on the foundation it creates a single area where data can be audited and validated. This makes it easier to troubleshoot and provide easier tracking and tracing of issues.

DATA MAPPING

Before you start creating jobs in SAS® DI Studio it is important to have a blueprint. This makes it easier to understand the number of jobs which need to get created and gives you a method of delegating which jobs get developed by different team members.

Source			Job		Target - Source Stage		
Folder	Library Name	Table Name	Folder	Job	Folder	Library	Table Name
01. Oracle ERP -> 01. Source	Oracle - ERP	Items	00. Source Stage	SRC_ERP_Items_I	01. Oracle ERP -> 02. Stage	Oracle - ERP Stage	ERP_Items
01. Oracle ERP -> 01. Source	Oracle - ERP	Customer	00. Source Stage	SRC_ERP_Customer_I	01. Oracle ERP -> 02. Stage	Oracle - ERP Stage	ERP_Customer
01. Oracle ERP -> 01. Source	Oracle - ERP	Orders	00. Source Stage	SRC_ERP_Orders_I	01. Oracle ERP -> 02. Stage	Oracle - ERP Stage	ERP_Orders
02. Microsoft CRM -> 01. Source	MS SQL - CRM	Customer	00. Source Stage	SRC_CRM_Customer_I	02. Microsoft CRM -> 02. Stage	MS SQL - CRM Stage	CRM_Customer
02. Microsoft CRM -> 01. Source	MS SQL - CRM	Products	00. Source Stage	SRC_CRM_Products_I	02. Microsoft CRM -> 02. Stage	MS SQL - CRM Stage	CRM_Products
02. Microsoft CRM -> 01. Source	MS SQL - CRM	Campaigns	00. Source Stage	SRC_CRM_Campaigns_I	02. Microsoft CRM -> 02. Stage	MS SQL - CRM Stage	CRM_Campaigns

Figure 6. Source to Source Stage Blueprint

The blueprint about outlines the movement of data from the source into the Source Stage. The Source Stage area is important as it is an image of the source information at a point in time. This area allows you to reconcile your information to a static source extract and should your process fail, you can reprocess without having to extract from the source system again. The jobs for this area are very simple and normally do not include a lot of logic and may be limited to a subset of the data based on change since last load time. The job names are important as you can see from the naming convention the information is coming from the source (SRC) to the Source Stage (ERP or CRM), the table it is loading and the way information is being loaded (I – insert only / append mode). Locations, libraries, source and target tables are all included in the blueprint to ensure a comprehensive picture.

This concept would be replicated to should the flow of information from Source Stage to Initial Staging, from Initial Staging to Final Staging, Final Staging to Foundation and from Foundation to Access and Performance.

This blueprint in effect becomes a lineage diagram outline the flow of information. It is an important artifact to track the progress of development and determine this impact of any changes down the line. The time invested in building the blueprint definitely pays handsome dividends during the development lifecycle.

EXTRACTING, LOADING AND TRANSFORMING THE DATA

Best practices and standards are developed over time and do not necessarily apply to all situations and organizations. Over the years there are a few best practices which have stood the test of time and are outlined below:

1. Naming conventions.

a. Table prefixes – tables are normally ordered alphabetically within tools there prefixes work a lot better to group similar tables.
    i. Staging Tables
        1. STG1 for initial staging tables
        2. STG2 for final staging tables
    ii. Foundation Tables
        1. D_ for dimension tables
        2. F_ for fact tables
b. Job Names
    i. SRC_TRG_Table_Name_Action – the job name should be broken into four parts.
        1. SRC this represents the source where the information is coming from
        2. TRG this represents the target where the information is going to
        3. Table Name is the target table name the information will be loaded into
        4. Action is the function performed by the job
            a. I – insert / append
            b. D – delete
            c. T – truncate
            d. U - update
            e. SCDII – SCDII job

    For example ERP_STG1_Customers_DI – this job would move information from the ERP Source Stage area to the Initial Staging placing data into the customers table using a delete insert statement.
c. Folder Names – prefix folders with a numeric so that they are sorted correctly in the SAS® tools.

2. ETL
    a. Remote and local table joins – do not create instances of remote and local table joins, this occurs when you are joining two tables from different libraries trying to filter data from one based on another. This will lead to very poor performance as SAS will first extract the data then join and filter. If you have large tables this is costly. As an alternative pass parameters from one job into another to extract the information or write some custom PROC SQL code embedded within SAS® DI Studio using parameters. The later yields the best performance.
3. Data Model
    a. For dimensions add the following columns to allow auditing
        i. Create_By – identifies the job which initially created the record
        ii. Create_Date – identifies when the record was created
        iii. Update_By - identifies the job which updated the record last
        iv. Update_Date – identifies when the record was last updated
        v. Rec_Src_Sys – which source system the record came from
        vi. Business_Key – the business key for the record
        vii. Rec_ValidFrom_Date – the valid from date
        viii. Rec_ValidTo_Date – valid to date of the record
        ix. Rec_Version – version number of the record
        x. Rec_Curr_Ind – the current record indicator
    b. For fact tables add the following columns
        i. Create_By – identifies the job which initially created the record
        ii. Create_Date – identifies when the record was created
        iii. Rec_Src_Sys – which source system the record came from

    If you are using surrogate keys remember to add the columns which were used to derive the surrogate keys onto your fact table in addition to the surrogate keys. If data needs to be restated or your surrogate keys are incorrect, these columns allow you to recomputed the surrogate keys without reloading.

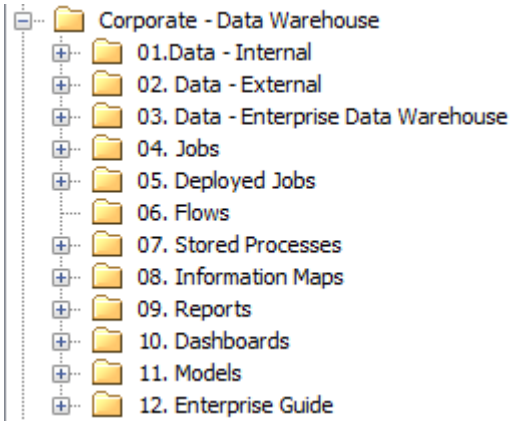
4. Views
    a. Relational views are great tools to abstract tables and logic from the ETL tool. This may sound counter intuitive but consider this. You are extracting information from a packaged application. You use SAS® DI Studio to connect to the application and extract the information. The application undergoes and upgrade and the table structure changes. This will cause a ripple effect for the ETL causing failure and a lot of changes. If you added a layer of views over the source tables and mapped the tables into the views, when the upgrade was complete you could remap the views and everything downstream would work normally. SAS® DI Studio would not even know there was a change or upgrade.
    b. Data type changes may be easier within views than multiple statements within many mapping to convert data types.

5. Constraints or no constraints – this is a hotly debated topic whether to include constraints or not. Normally in a warehouse constraints are not added so as to allow for the use of the truncate command within a relational database system. This greatly reduces time instead of deleting records. Some tools use constraints to build default relationships and therefore sometimes they are placed on tables. Due to the use of surrogate keys in a warehouse, primary and foreign key constraints are not very effective at enforcing integrity but do decrease performance of loading data.
6. Switch off the automation settings within your jobs
    a. To begin with until you understand and have worked with the automation settings within SAS® DI Studio switch these off as they can be a big source of frustration and unexplained behavior.

To find additional best practices within your environment perform a small prototype and find out what works and what does not. Firm up a set of basic standards and best practices to start with and as time continues enhance and add to the list. As new product versions are available these will also affect the best practices so it is important to review these periodically.

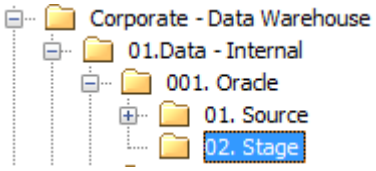
SETTING UP THE METADATA

The metadata for a enterprise warehouse can become very disorganized very quickly if the proper structure is not created to start with. At the start of the project you need to create folders for data. These should be for internal and external sources of information as well as a specific area for the data warehouse. Additional folders should be created to segregate the different objects which can be created during the development lifecycle for example reports, dashboards, models, stored processes etc.



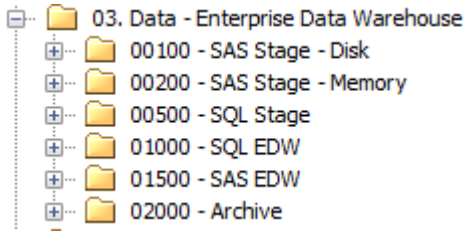
Display 4. SAS® Management Console Metadata Structure

Areas such as the data should be further sub divided to cater for the source and the Source Stage. The library definitions for these different data layers would reside within the same folders.



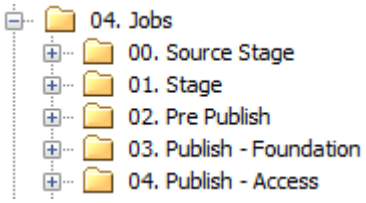
Display 5. SAS® Management Console Metadata Source Structure

Likewise the data warehouse can be sub divided into the data layers in the ETL architecture also with the respective libraries under the different folders.



Display 6. SAS® Management Console Metadata Data Structure

Separating the jobs folder into the different data layers makes managing and maintaining the jobs a lot easier.

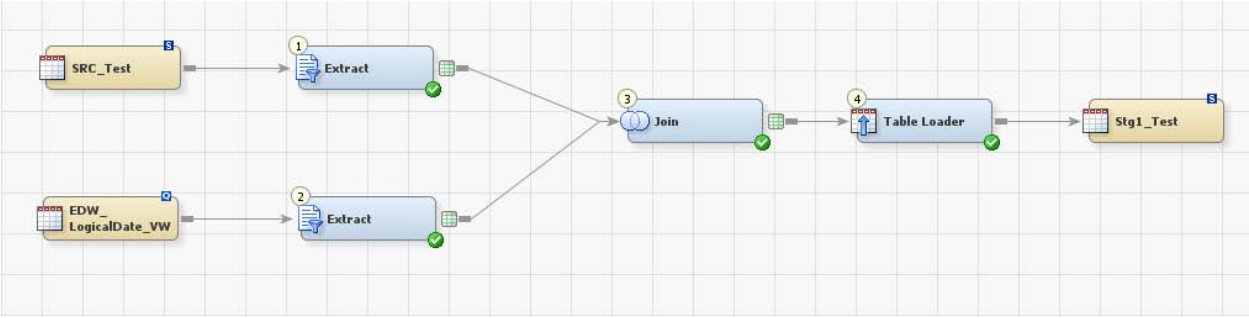


Display 7. SAS® Management Console Metadata Jobs Structure

When the blueprint is development the metadata structure should be created at the same time to ensure the two remain in synch. In conjunction with this activity libraries can be developed and defined as part of the initial ETL architecture and then placed into the respective folders.

CREATING THE TEMPLATE

Template jobs are an easy way to understand what needs to happen at each of the different layers within the ETL architecture.



Display 8. SAS® DI Studio Source Stage to STG1 Template

As can be seen from the mapping above information is extracted from the Source Stage and placed into the Initial Staging area. This construct would be the same for nearly every Source Stage to Initial Stage job. This job can be copied and the source table and target table updated to reflect the new tables. Each transformation can be reviewed and updated to reflect the new tables. This saves significant amounts of time. Pre and Post code is already added to the properties and therefore is automatically included into the new job. Keeping your jobs as simple as possible make them easier to maintain and there is a greater potential to create templates.

CONCLUSION

The first 30 days should be spent planning and developing the blueprint for SAS® DI Studio. At the end of this period you should have a very well defined plan to build and develop your data warehouse. By taking the time to understand SAS® DI Studio and develop your ETL Architecture the implementation of SAS® DI Studio should be a lot smoother.

REFERENCES

Fehd, Ronald. Paper 267-30 *A SASautos Companion: Reusing Macors*. Atlanta, GA, USA: SAS

CONTACT INFORMATION

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

Name: John Heaton
Enterprise: Heritage Bank
Address: 400 Ruthven Drive
City, State ZIP: Toowoomba, Queensland, Australia, 4350
Work Phone: +61 4694 9129
Fax: +61 4694 9785
E-mail: Heaton.j@heritage.com.au
Web: http://www.heritage.com.au