Paper 169-28

Next Generation Warehousing with Version 9

Gary Mehler, SAS Institute Inc., Cary North Carolina

ABSTRACT

This paper covers the new data warehousing and management services available in Version 9 and shows how they help manage large, real-world ETL development environments. Topics covered include multi-user planning and administration, security management, the development–to-production lifecycle, deployment issues and scheduling, and the integration of data quality management.

INTRODUCTION

Data warehousing is the foundational layer of data upon which business intelligence processes can be built. Without a data mart or data warehouse that has a documented structure and the ability to accommodate updated data, analytic consistency over time cannot be achieved. After all, a data warehouse is typically thought of as the means to achieve one version of the truth for an organization. As advanced analytics are used more broadly to get a competitive edge, the data warehousing process becomes even more important. Furthermore, the structured and documented approach required to correctly do data warehousing is an important area to understand so its principles can be adopted in higher-level functions as well.

This paper takes the general view that data warehousing is more than just an ETL (**E**xtract, **T**ransform, and **L**oad) process for a data collection, and additionally includes the infrastructure upon which this layer is built. In other words, it includes the technical information (or metadata) about all the sources to extract, how to manipulate and transform them, and the structure of the data model upon which higher-level functions such as analytic analysis are performed. This will take us into the area of infrastructure configuration, to include information about the users of the infrastructure, how a development lifecycle should be approached, and how groups of people can work together on these types of projects. Finally, moving from a development to a deployment methodology, we will discuss how to safely and reliably operationalize the processes for regularly-scheduled use.

METADATA

Warehousing, like many other computer-based operations, has a lot to do with metadata, or data that describes basic tabular data as well as processes executed upon the basic data. In order to make infrastructure plans for a large-scale warehouse, metadata needs are important to consider as a first step in the planning process.

Version 9 of the SAS® ETL Server is highly dependent on metadata services. These services can be thought of as storage repositories for metadata, and are built into collections of repositories that service a particular function. As an example, the typical warehousing development methodology calls for a three-level model for Development, Testing, and Production use. These are typically called **DEV**, **TEST**, and **PROD**, respectively. This terminology will be used in the remainder of this paper.

Each level is represented as a collection of repositories that comprise the warehouse operation for that level. As will be covered next, these collections have their own needs for interoperation with minimal extra administrative overhead. Within a level, such as the **DEV** environment, multiple ETL developers may be working on various aspects of a larger project, and each of these projects can benefit from metadata independence during the early development phase. Called Change Management, this allows for separation of independent projects so that work can proceed in parallel. Without change management, an ill-considered change could effectively stop all work since all users would be subjected to all changes as soon as they're made.

Change management is key for real-world development of any type, and is especially important for ETL development environments where interdependence of data elements can be difficult to assess on the surface. As we'll see, change management allows developers to check out a particular object (such as a table definition), experiment with the impact of changes in a protected environment (a project workspace), and after its safety has been determined, check it back into a shared work area for other users to see.

Version 9 of the SAS ETL Server fully supports multi-level and change-managed development environments. Components of the suite include the following core components:

- **SAS Metadata Server**, the core metadata services layer for SAS applications
- **SAS Management Console**, the administrative workbench for SAS applications
- **SAS ETL Studio**, the ETL developer's workbench
- **The SAS System** for advanced application processing

Additional requirements can be met by other components that interoperate with the core suite, and are also part of the Version 9 product suite:

- **SAS Data Quality Server** for data cleansing, householding, and other matching
- **SAS OLAP Server** to build cubes
- **SAS Scalable Performance Data Server** for high-performance data access
- **SAS/Access products** for DBMS access
- **SAS Data Surveyors** for Enterprise Application access

Integrating all of these components to meet real-world warehousing needs is possible with Version 9 of the SAS ETL Server suite, and is shown in more detail in the remainder of this paper.

MULTI-LEVEL PLANNING

Some forethought is required when multiple developers, administrators, or end-users are brought into the picture. For starters, having multiple developers involved in the warehouse design and implementation process requires that a common approach be taken for the storage of documentation and implementation materials. This is important so that the work of these multiple warehouse developers can be integrated into a consistent and maintainable structure.

In addition to multi-tier metadata considerations, the full power of the SAS System to utilize client/server and n-tier configurations for application processing as well. This means that analytics and data management can be distributed across the enterprise to meet processing needs.

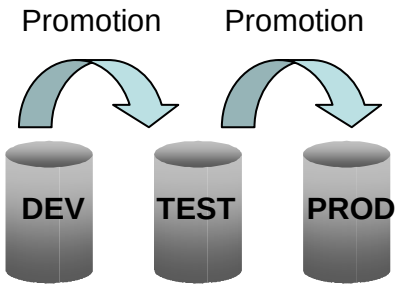

Figure 1: Multi-level environment to isolate the effects of change from the production warehousing environment. The promotion process is managed by administrative tools.

In addition to handling multi-level migration, the core metadata may need to be available for multiple distributed sites and have support for backup and restore to mitigate against hardware failure. These capabilities are supported with the general names of Replication, whereby a repository can be copied to another location. A topology of metadata servers can be created to support this operation in a straightforward fashion.

MULTI-USER ADMINISTRATION

The large number of people that might be involved in an enterprise-wide warehousing project means it is important that the tools used be aware of the roles of these individuals and help administrators make consistent choices throughout. This includes:

- creating project work areas for each ETL developer
- setting group permissions easily for all the ETL developers
- sharing common information as needed
- enforcing that updates are done with an audit trail
- using a check-out/check-in approach to keep overwrites from occurring

SAS Management Console is a single point of control for SAS administrative tasks. It is a Java-based application that provides a common user interface that can be extended through a Java plug-in interface to meet future needs as well as be customized by end-users to meet site-specific needs. As a basic application, it is provides access to metadata about various aspects of SAS System operation, and of applications such as SAS ETL Studio.

Its components include:

Figure 2. The general user interface of the SAS Management Console.

1. A menu bar for access to all functions
2. A toolbar for shortcuts to common functions
3. Context bar to indicate the active metadata repository
4. Navigation tree to show major functional areas
5. Main display area for work tasks
6. Status line to indicate connection status, userid

Looking more closely at the Navigation tree, the functional areas covered by the basic SAS Management Console are shown below. Additional plug-ins would register on this tree as well.

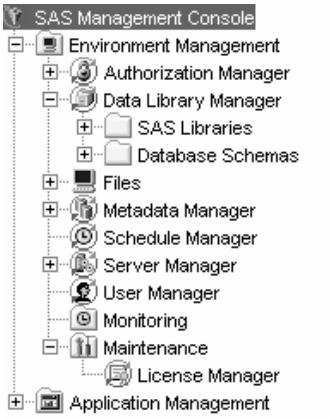

Figure 3. The main navigation tree of the SAS Management Console

The value of storing information in a common metadata repository is clear when the re-use of this information through the SAS Management Console is shown below. An administrator can define complex definitions for all data and application servers just once, and then share that information with the user community.

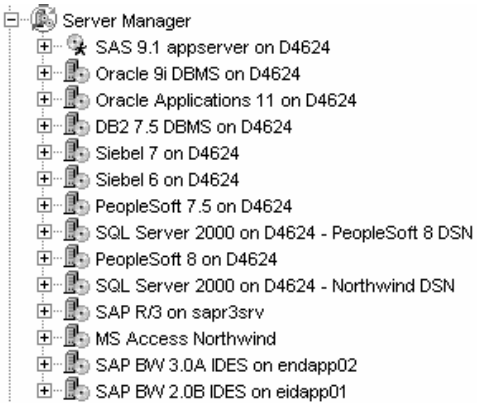

Figure 4. Common definitions are entered once and can be re-used by many users.

Putting all of the information in one place makes it easy for administrators to manage large-scale enterprises, but also requires authorization management to ensure that each user has permission to see just what they're supposed to see. In SAS Management Console, this is easy to manage through the Authorization Manager. In the figure below, specific users or groups of users can be allowed to use specific pieces of information in the repository.

Figure 5. By denying access to metadata, such as a server definition, an administrator can control access to who can use the server.

This is a very powerful feature, since in the metadata repository, each object can be administered in this way. That means that not just servers can have authorization rights set, but also things like tables, columns in tables or reports.

THE ETL DESIGN PROCESS

After administrative tasks have been completed, the actual ETL design process can get underway. ETL, after all, is about Extraction, Transformation, and Loading to support business intelligence needs, the real reason we're doing all this. SAS ETL Studio is the environment in which this is performed. This is the same basic user interface as is used in SAS Management Console.

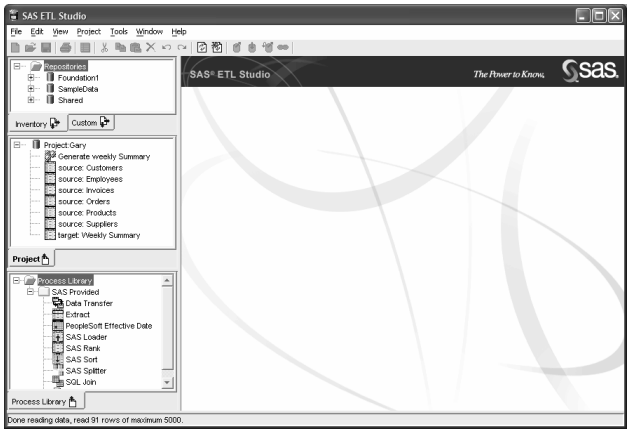

Figure 6. The basic layout of SAS ETL Studio. Multiple panels on the left provide work areas for repositories, projects, and processes.

Initial tasks in ETL Studio will revolve around defining sources to extract and targets to load. In the example here, the ETL developer has a simple task to perform: transform basic warehouse and transactional data into a weekly summary table. This would typically be done in a project area to contain the risk and scope of work, and then would be checked into a common area for others to use. Further, a grouping would be defined to keep the pieces together and help avoid confusion with components of other projects. Next, we will walk through the steps to create a project to accomplish this task.

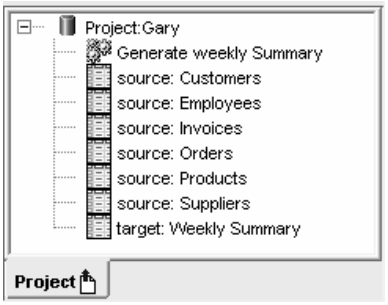

Figure 7. A close-up on the project tab. In this example, the user (Gary) is working in a project area with source tables to generate a weekly summary target table. The job that will do this is indicated as well.

To see how this project was constructed, we will go through a high level walk-through showing how the main portions are done.

Sources

An ETL process starts by extracting data from some external location, such as a database, SAS storage, or flat files. The ETL Studio component that focuses on this area is a source designer wizard. The wizard will walk us through each step of defining our source in metadata so it can be documented and re-used. The source designer presents the user with the listing of supported source types, including a large number for ODBC-based data. As with many operations in ETL Studio, a wizard-based interface helps the user make step-by-step decisions on how to complete their task.

Below we see an extract from the Source Designer wizard showing some sources to choose from.

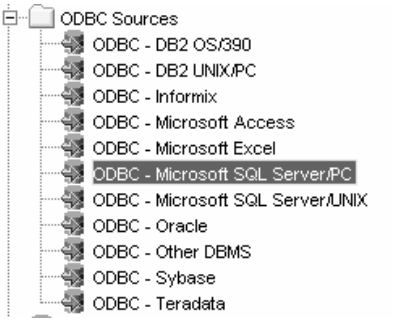

Figure 8. Selecting a source designer. Several ODBC types are specifically supported.

After the SQL Server ODBC type is selected, the user can specify the specific data source name (or the ODBC DSN) to use. Once specified, ETL Studio uses that information to help determine which tables to import. For this example, we can select a collection of source tables that we'll access via ODBC in our ETL job.

In the case of this example, the source data used is from a sample database in SQL Server. The wizard helps the user fill in the right answers to fully describe the source being defined. The figure below shows how we choose the tables of interest from the complete set of tables available from the database.

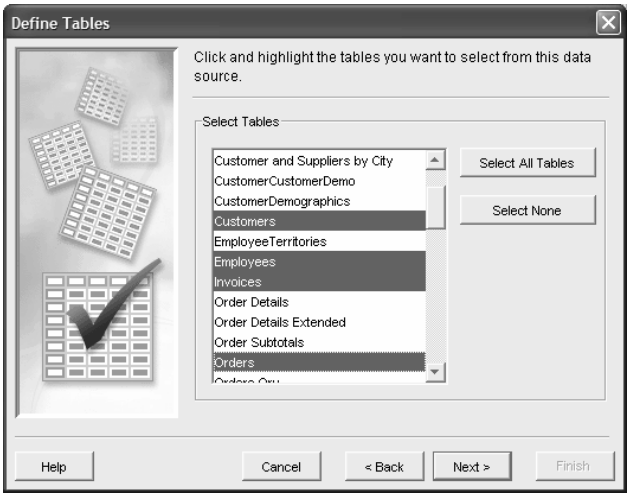

Figure 9. Selecting a group of tables to import metadata about.

Understanding the data is vital when determining how to design the job to meet the ETL objective. In ETL Studio, a wide array of assistance is available to do this. A common way to understand how the data model is constructed is to understand whether keys or indexes are present for source data. Below is how a user would understand how keys associate the grouping of tables together. This is a panel of the properties stored for any table registered in the repository.

The sources (or other elements such as targets or processing steps) can execute on any platform on which the SAS System is present and configured for remote processing. ETL Studio takes full advantage of multi-tier computing by making use of SAS Integration Technologies and SAS/CONNECT software. SAS code generation makes implicit use of metadata describing data locations to generate the appropriate code to execute at runtime for multi-tier deployment.

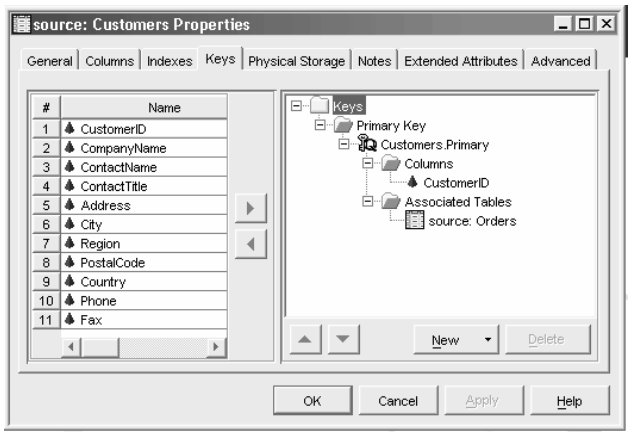

Figure 10. Properties of the Customers source table showing that the CustomerID column is a primary key. It also shows the table Orders is associated via this key.

In addition to keys and indexes, other properties of tables are available as well in metadata and can be viewed here as well. This includes the type of database, SAS library and schema. In addition, table-specific options can be set and retained. In the case in which the source table uses non-standard characters in column names, for example, options can be set to ensure that these are handled properly.

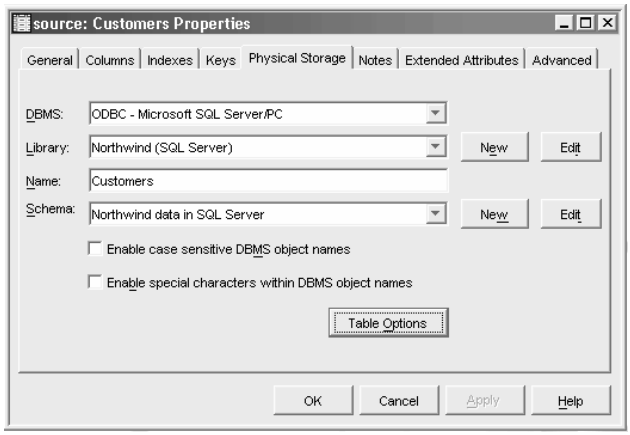

Figure 11. Details of how the Customers table is physically stored. In this example, we're seeing a table access via ODBC from SQL Server. Full information about its schema and other details are available to view or edit if needed.

By this point, we've seen how to register source tables from an external source like a database or ODBC source. Once registered, the metadata about those tables can be reused for many purposes. One use of this metadata is in constructing ETL jobs, which have a dependence on the type of information in these tables (such as column names and types). Another use of this metadata is to define new tables with similar characteristics. This is common when defining new target tables that are built from known source table elements.

Targets

The steps to add a new object to a project, in this case a Target table definition are aided through a wizard. ETL Studio uses wizards to assist with many common operations. This provides a powerful way to handle a range or ways to load targets, from default settings of UPDATE or APPEND, and whether DROP or TRUNCATE load steps are required. The New Object wizard is an example flow we'll see below.

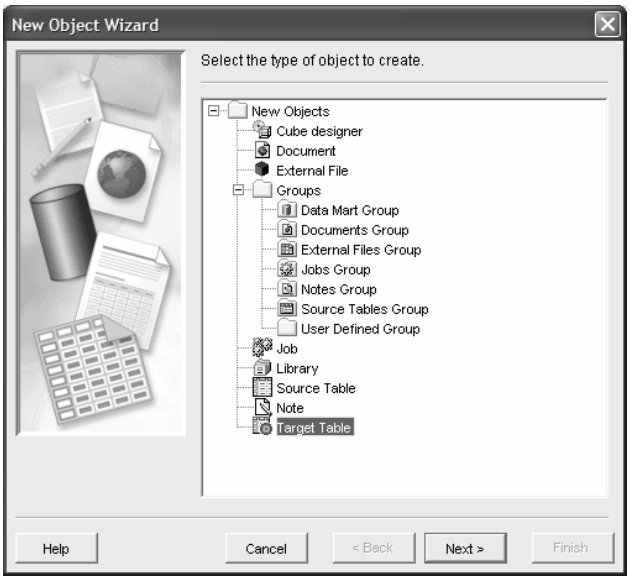

Figure 12. The New Object wizard is used to add a target table definition to a project. Other object types can be added in the same way.

Creating a target table definition based on existing columns is easy since the user can re-use column definitions from source tables. In this example, we're merging columns from the source tables we're going to use in our process. Below we see how a user was able to select columns from multiple input tables to construct the desired collection of columns for the new target table. This is a way that metadata can be re-used instead of requiring the user to re-enter data from memory.

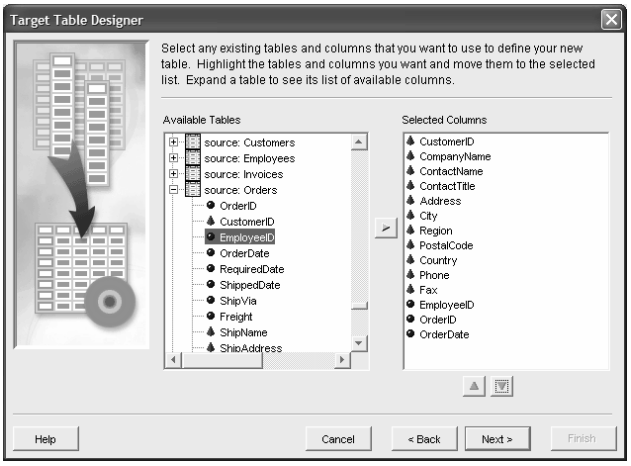

Figure 13. The Target Table Designer helps create new table definitions based on existing tables and columns. In this example, we chose the desired columns from tables used in the process to be designed next.

Transformation Jobs

After having studied our sources to know how to proceed, we can create the job to transform the sources into the desired target. To do this, the ETL Studio process Designer is used. In this simple example, the bulk of processing revolves around joining the source tables into the summary target table. Choosing the Join transformation, we simply drag and drop it onto the Process Designer canvas. After doing so, it helps us by showing the connections it needs to effectively do its work. In this case, a minimal join takes two source tables and joins them into a third target table. Templates show how this needs to be done.

In the example shown below, we will build a simple SQL join using the built-in Join transformation. This built-in transformation is a very powerful tool for specifying just about any part of the SQL syntax in a straightforward GUI as we'll see below.

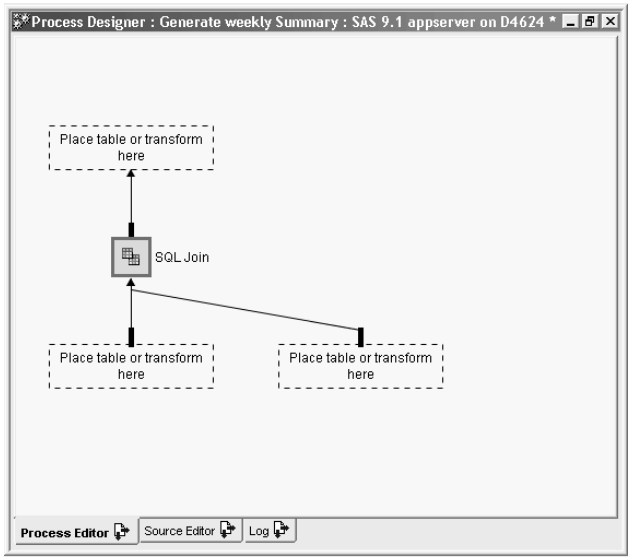

Figure 14. The basic Process Designer canvas showing a Join transformation that has been dropped onto it. Template placeholders indicate next steps to the user. If dropped into an existing job, this transformation could be dropped onto a template place-holder to link into the existing jobflow.

Filling in the blanks, we build the desired job as required to put the right data pieces together. Dragging and dropping tables into the Process Designer canvas is easy to do and gives visual feedback on how the ETL Developer is doing.

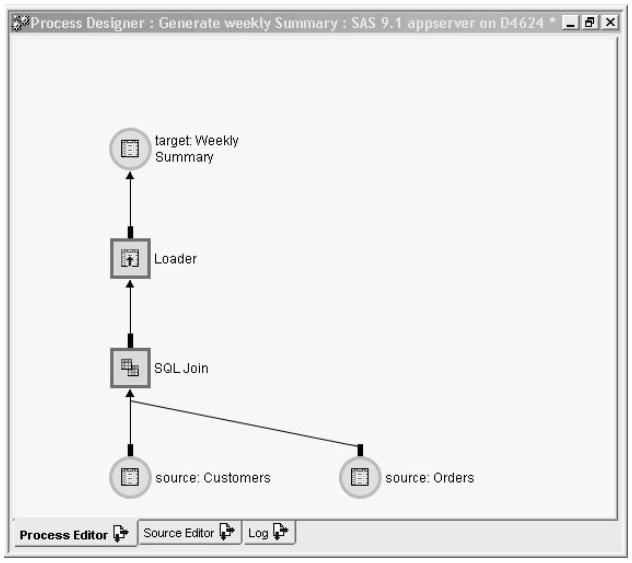

Figure 15. The ETL Developer has specified two input sources and the target output table on the Process Designer canvas by filling the template slots provided by the Join transformation.

Specifying details of how the central Join is going to work is possible through property settings for the Join. As with everything else, right clicking on the Join transformation gives access to its Properties panel. In it, we can review a number of SQL-oriented settings.

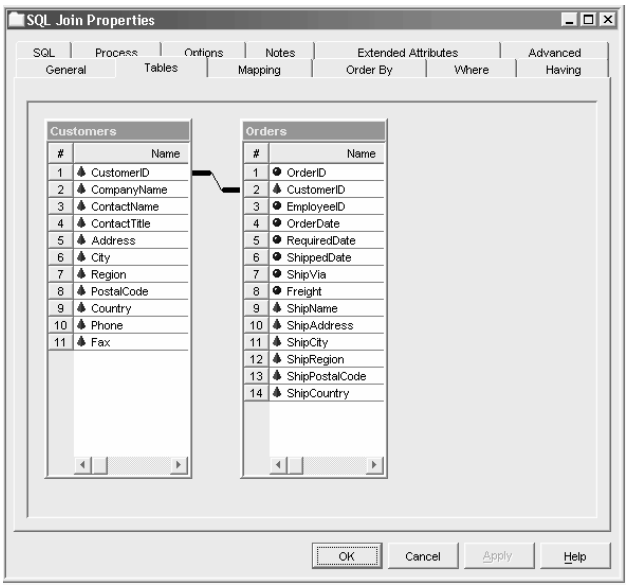

Figure 16. SQL Join properties for the Process. As expected, the auto-mapping of the primary keys makes joining multiple tables straightforward.

In our example, we need to filter on recent dates. That can be done with a data filter (a WHERE clause for SQL experts). Each part of the SQL language is handled through property tabs in a graphical form. Other parameters that can be specified include: ordering, having, and more complex mappings based on column-level expressions. Below we see how this can be done visually by building a filter.

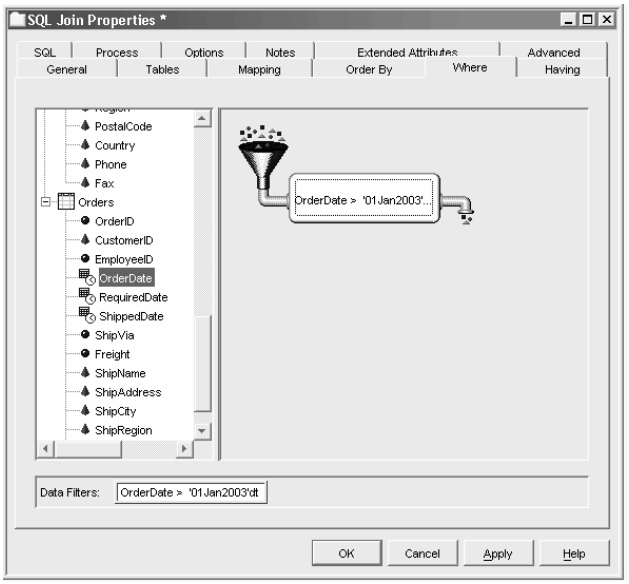

Figure 17. Building data filters visually.

Of course, the goal is to execute this efficiently, so ETL Studio is really writing SAS code to accomplish this, and we can review it in several ways. While adjusting the join properties, the user can switch to the SQL tab to review the syntax for accuracy. While viewing the syntax, the SQL code can be modified in-place, and the visual elements will reflect the updated SQL for the user. Below is the SAS SQL code generated to perform the join that was visually constructed for this example.

```
Proc SQL;
  create VIEW WORK.W4SMTIBY as
  SELECT Customers.CustomerID AS CustomerID,
    Customers.CompanyName AS CompanyName,
    Customers.ContactName AS ContactName,
    Customers.ContactTitle AS ContactTitle,
    Customers.Address AS Address,
    Customers.City AS City,
    Customers.Region AS Region,
    Customers.PostalCode AS PostalCode,
    Customers.Country AS Country,
    Customers.Phone AS Phone,
    Customers.Fax AS Fax,
    Orders.OrderID AS OrderID,
    Orders.EmployeeID AS EmployeeID,
    Orders.OrderDate AS OrderDate,
    Orders.RequiredDate AS RequiredDate,
    Orders.ShippedDate AS ShippedDate,
    Orders.ShipVia AS ShipVia,
    Orders.Freight AS Freight,
    Orders.ShipName AS ShipName,
    Orders.ShipAddress AS ShipAddress,
    Orders.ShipCity AS ShipCity,
    Orders.ShipRegion AS ShipRegion,
    Orders.ShipPostalCode AS ShipPostalCode,
    Orders.ShipCountry AS ShipCountry
  FROM nwindSQL.Customers
    INNER JOIN nwindSQL.Orders ON (Orders.CustomerID = Customers.CustomerID)
  WHERE Orders.OrderDate >  '01Jan2003'dt;
  quit;
```

Figure 18. The integrated Source Editor shows the generated SAS code to perform the process.

As our example evolves, we find that we need to amend our data to include additional information from another table. This is easy to do visually by just dropping another table onto the Process Designer canvas.

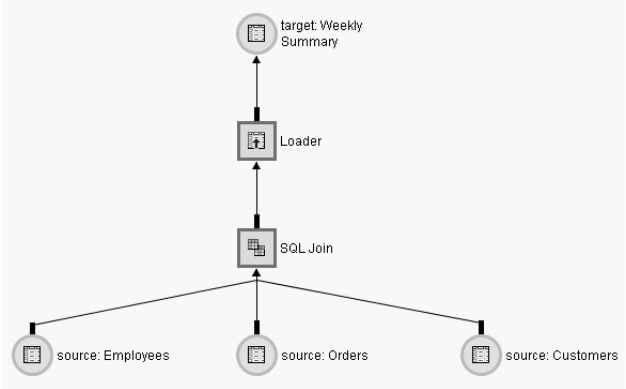

Figure 19. Multi-way Joins shown visually.

In addition to adding more tables to transformations like Join or Split by dropped them onto the process node, existing jobs can be modified in other ways as well. To insert a process step in an existing job, the user can drop it on one of the connector lines to insert it at that point. In addition, multiple flows can be connected by dragging an endpoint onto an open template slot or a transformation that can be extended (like Join or Split).

As the join is extended to accommodate the third table, properties of the join are extended as well. This means that when we return to the table join properties, we will see the third table reflected as well as its default column mappings indicated. Elements like the join transformation are limited only by the upper bounds present in the SAS System on which the code will eventually be executed, so the join transformation can accommodate up to 32 tables as input into a single Join.

Although this is a simple example, very large configurations can be created by combining basic transformations into larger, complex process flows. Transformations can feed right into other transformations without having to store data in an intermediate table. In most cases, a SAS View is used when propagating data across process steps.

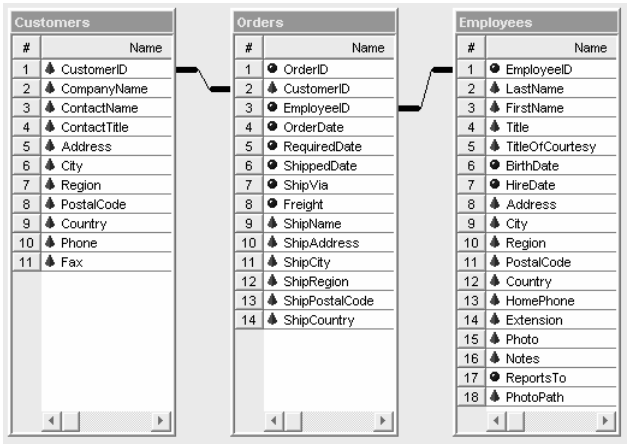

Figure 20. Table mapping specifics are visually shown, this time for three tables. They are joined on their primary keys by default.

If more complex parameters need to be set, that is possible by editing advanced properties for the join. Below we see that we could specify advanced join types (such as left, right, inner, or outer). As more tables are added to the join, the panel on the left shows the order in which the operations are performed. For ordered joins such as left and right, this is important to visualize, especially when larger numbers of tables are combined.

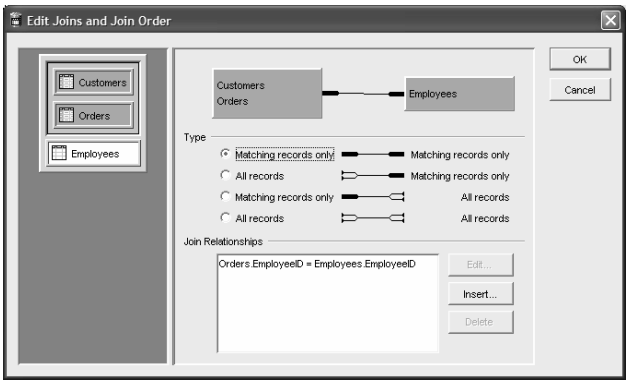

Figure 21. Specifying advanced join types to obtain the required results.

EXTENDING ETL STUDIO

What this paper has covered so far is the administrative configuration and basic ETL Studio operation to define sources, targets, and intervening transformations required to manipulate the data as required. Using the powerful set of basic transformations provided in ETL Studio, many things can be accomplished.

However, in the event that the pre-packaged transformations in ETL Studio don't meet all of the transformation requirements for your needs, they can be extended in a number of ways: user-written code, user-written transformations, and Java-based transformation plug-ins.

User-Written code

For cases in which a custom algorithm is needed for a single time of use, the full power of the SAS language can be called up to achieve the objective. If the algorithm is needed as part of a data flow (as in a transformation step), there is a basic transformation type called "User Written Code" that can be used. Just like any other transformation, it can be dropped onto the Process Designer canvas and linked into ETL processes. The use can then specify the SAS

language code to run as that process step. The code can reside in an external program file, or be stored in the SAS metadata repository, as shown in the following figure. For external files, host-specific path information is stored so that the code can be located and executed on any remote host.

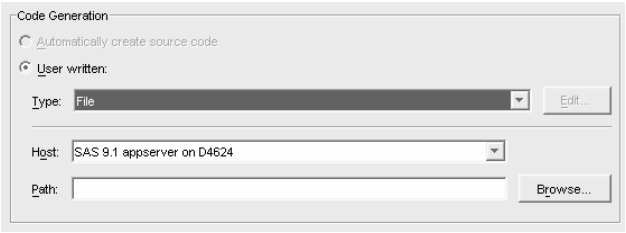

Figure 22. In a user-written code transformation, the user inputs information about how to locate the provided SAS language code. In this case, the user specifies a file location using a filesystem path that is relative to the execution host.

Another type of single-use user-provided extension can be done through pre-processing or post-processing steps for any entire process. In the figure below, we see that the user can store, edit, and maintain this code through the metadata server storage facility.

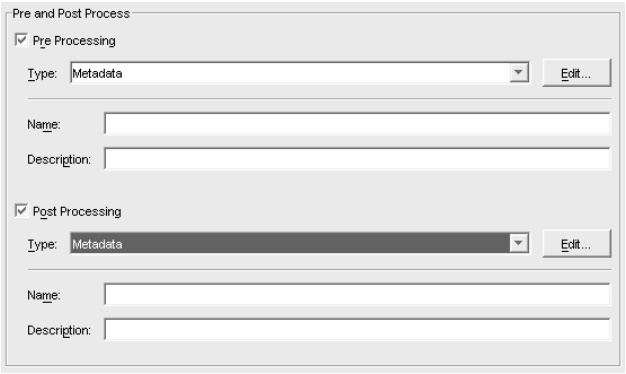

Figure 23. Pre- and post-processing steps for a complete process. In this figure, clicking on the Edit button allows SAS code to be entered through an embedded SAS language-aware editor and subsequently stored in the metadata repository.

User-Written transformations

In the event that user-written extensions need to be re-used as transformation steps generally, this can be done through the ETL Studio Transformation Generator. The generator helps the user specify SAS language or SAS macro code that can be stored as the internal steps to execute as part of a reusable transformation. This allows the great power of the SAS language to be reused by many users. In addition to reusability, the reader will recall that metadata information is real documentation that can help sites develop a library of documented, reusable SAS code modules.

As an example, we will walk through the process of adding a new transformation into our ETL Studio environment. This starts by naming and describing the new transformation step as well as determine a grouping location for the new transformation. For complex grouping structures, a format of Level1.Level2 can be used. For example, entering: "`Data Profiling.Summary Statistics`" would place the new transformation in the process folder "Summary Statistics" under "Data Profiling".

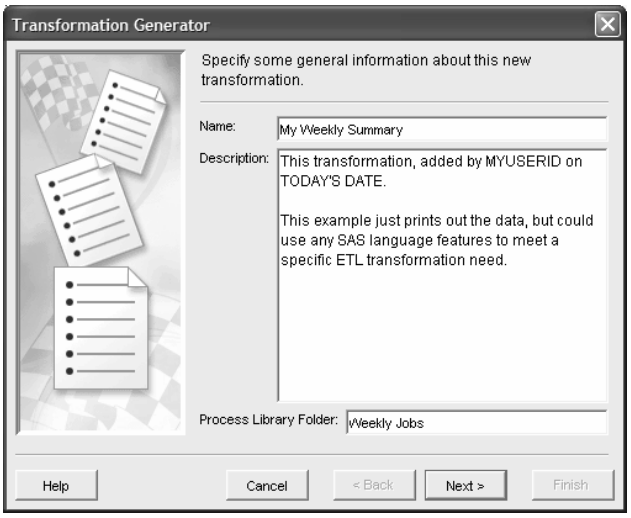

Figure 24. Beginning the transformation generation process. The user names the new transformation, assigns it to a grouping folder, and provides a place for documentation for the new item.

After naming the transformation, the user-written code can be added with the embedded SAS-syntax editor. This code can be SAS Data Step or SAS Macro code as required. Macro variables used in the transformation can be replaced later by specifying options to define each variable. For this simple example, we're just writing diagnostic code with PROC PRINT. %BYVAR will be an option to be specified when the transformation is dropped onto the Process Designer canvas later.

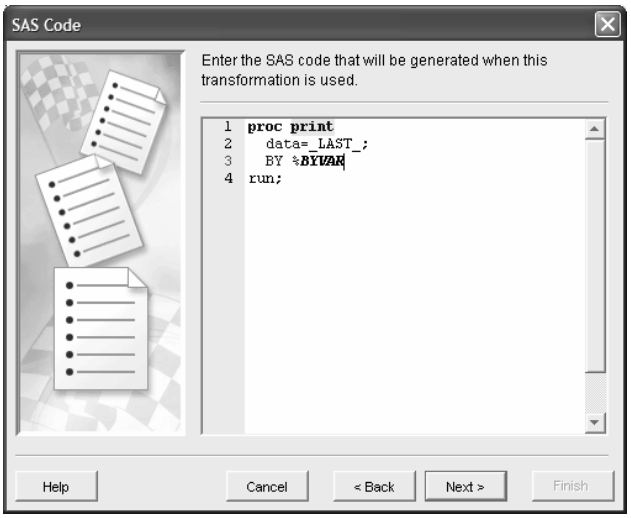

Figure 25. Entering the SAS language code for the new transformation.

After the SAS language code has been entered, the user has the opportunity to define the external names and descriptions that will be used by the ETL developer when the transformation is used. Below we see how any number of input or output variables can be defined.

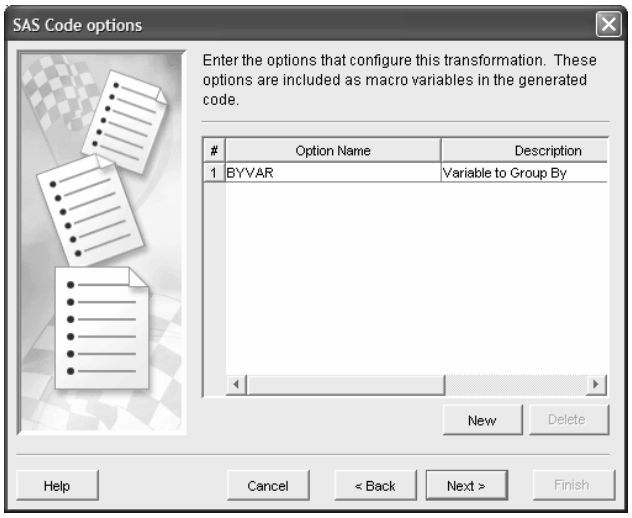

Figure 26. Specifying options for the transformation. Through SAS macro variable handling, external parameters can be passed to user-written transformation code.

After specifying the option parameters, we can specify whether this new transformation fits anywhere in a flow (has input and output tables), or belongs in some endpoint. As an endpoint, it either wouldn't have a template for its input or its output.

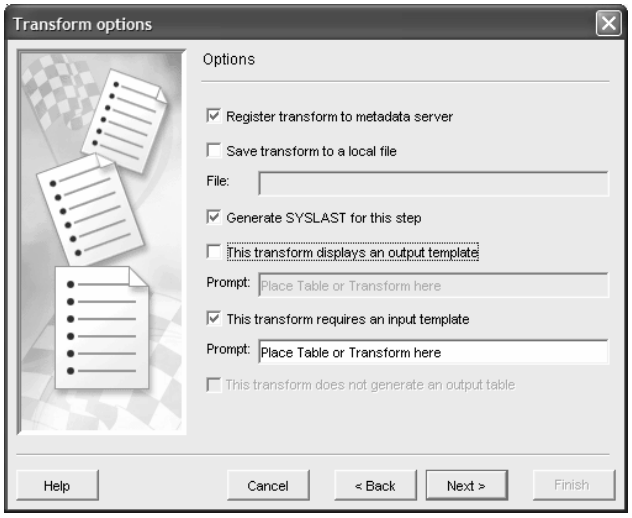

Figure 27. Additional options; the sample transformation won't have an output, so we can turn off the output template to guide the user.

When finished with the Transformation Generator, the new transformation is added to the library of transformations available to the ETL developer. So, in the scheme of what is being accomplished, the Transformation Generator lets an ETL developer actually design program components of ETL Studio that others can use and re-use. In this capacity the user of this Generator is more of an ETL process leader who can define new templated transforms for other ETL developers to re-use.

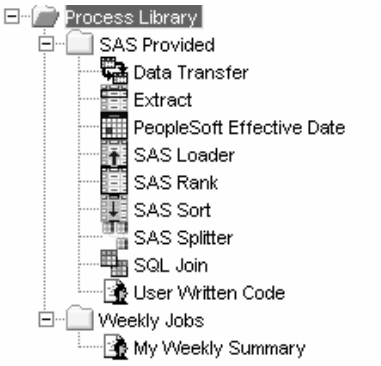

Figure 28. The user-provided transformation shows up as a reusable component.

When using the new transformation, it can be dragged onto the Process Designer canvas like any other transformation. In this example, our transformation requires an input, but doesn't generate output, so it shows the appropriate template areas on the canvas.

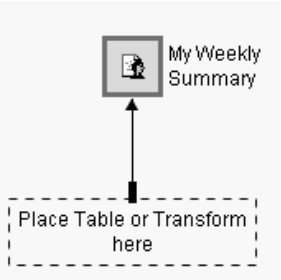

Figure 29. A reusable transformation in use. This transformation on asks the user to fill in an input table template.

If adding to an existing flow, the user can simply drag an existing node (such as a table) onto this template area to link it into the existing flow.

Java Plug-In Transformations

For more complex operations, ETL Studio can be extended in a number of ways by writing Plug-ins in the Java programming language. A user would choose this route if there were more complex operations required than could easily be represented in the SAS programming language, or if more complex visual elements were required to guide the user through the design and run-time environment of using the transformation.

Many SAS-provided pieces of the ETL Studio suite are provided using the plug-in interface, and there are several ways in addition to transformation plug-ins to extend ETL Studio. It supports a number of different points at which it can be extended through Java plug-ins. These include:

- Shortcut bad and the Tools menu
- General application menu and pop-up menus
- Adding items to ETL Studio display trees
- Transformations in the Process Library
- Adding tabs to property sheets for objects
- Source and Target designers
- Any other object that the user can create.
- Source code overrides for transformations and jobs

OLAP

OLAP cubes are a powerful way to represent dimensionally structured data for quick analysis. The Cube Designer plug-in to SAS ETL Studio provides a wizard-based way to define new OLAP definitions and actual cubes based on existing data. Below we see a sample screen from the Cube Designer.

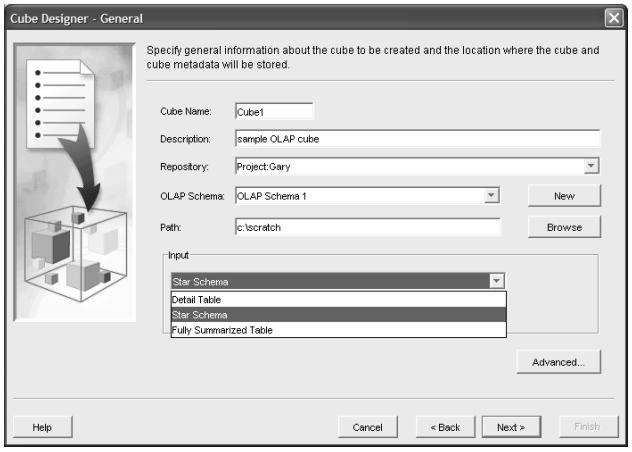

Figure 30. The Cube Designer helps the user build OLAP cubes in SAS ETL Studio.

DATA QUALITY

No discussion of advanced data warehousing would be complete without considering the need to understand and manage quality issues. Whenever data is assembled from a variety of sources, minimizing the effect of defective data is a major concern. In addition, reducing data redundancies and standardizing data elements increases the accuracy of the data, making it more usable.

SAS Data Quality Server can handle the task of removing erroneous data and match common data when merging data from previously disparate sources. Cleansed data leads to more usable results, so consideration of cleansing in your ETL processes delivers a good return on investment for the resultant warehouse.

SAS Data Quality Server can be used in many ways:

- to generate match codes for transformation, reporting, and analysis. This is done through the DQMATCH procedure, which is available as a transformation plug-in to ETL Studio
- to generate and apply schemes (DQSCHEME) that transform data sets, which can be done through another ETL Studio transformation.
- Data values can be case standardized (upper or lower case) using the SAS language function DQCASE, available in the ETL Studio's expression builder library.
- Other functions to determine gender or locale are also surfaced in the ETL Studio expression builder.

Used in conjunction with dfPower Studio software from DataFlux Corporation, the SAS ETL Server can help leverage the full power of data quality in ETL processes.

COMMON WAREHOUSE METAMODEL (CWM)

SAS Version 9 products are fully compliant with the open standard that supports common metadata interchange. This important standard allows for the sharing of metadata information across applications. In a typical large enterprise, a corporate or departmental data model is managed by a data architect. This data model describes how various pieces interrelate and helps prevent multiple versions of the truth from confusing a data warehousing environment. Using standard data modeling tools like ErWin or Rational Rose, large-scale data models are defined for use. These can be imported directly into ETL Studio.

In cooperation with Metaintegration Technologies Incorporated (see http://www.metaintegration.com), advanced capabilities for importing metadata model definitions from a wide range of products is supported. Below we see an example use of ETL Studio's Metadata Import wizard.

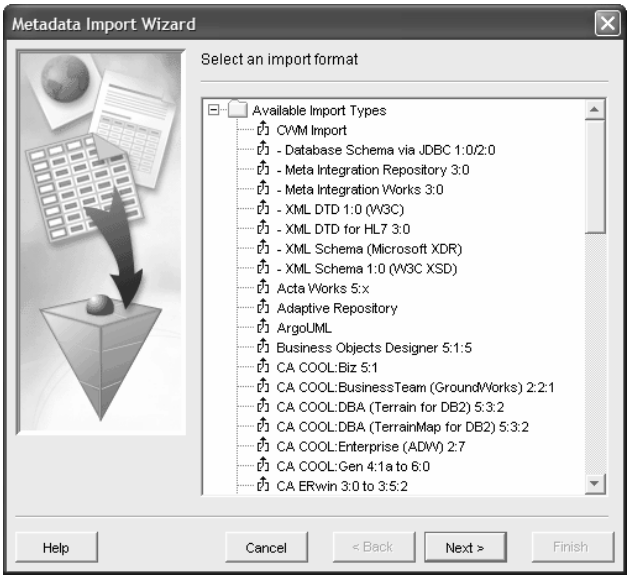

Figure 31. Among the wide range of metadata import types supported, the user can select the needed type.

The outcome of a CWM import are all the table definitions that comprise the model. In this way, a large number of target definitions can be transported from the data model design environment into the ETL design studio for implementation.

OTHER SOURCES

In addition to simple storage like DBMS or SAS tables for which it can be straightforward to understand the metadata such as column names and types, other types of data require a different approach. A few that we will discuss here include external flat files and enterprise applications (such as ERP or CRM operational systems). While a lot of information resides in these other systems and formats, a different approach is needed to understand the infomration. In the example below, the user is searching for a string present in table descriptions in their SAP R/3 system.

External Flat Files

External text files require special handling as well since their format includes useful information. Typical formats like comma separate files or tab separated files are a standard data interchange that includes column names in the first line of the file, but no other format or type infomration as metadata. The External File Interface in SAS ETL Studio takes this into account and reads some portion of the file to make educated guesses about the type and format of data in it.

Below we see an example comma separated file of company information. The External File wizard reads some portion of the file to determine the type of data in it to register as metadata.

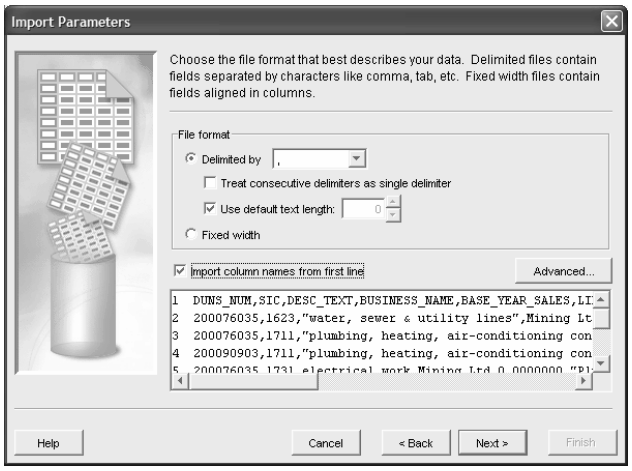

Figure 32. The Import External File Interface helps understand the format of external delimited or fixed width files.

Operational Systems

Operational systems like SAP R/3, SAP BW, Siebel, PeopleSoft, and Oracle Applications can be rich sources of information, but can be more challenging to understand since in many cases their metadata isn't stored in a standard form. In other words, while a DBMS makes heavy use of keys, indexes, and descriptive infomration in individual tables, operational systems can tend to place their metadata in yet more DBMS tables. Understanding the structure can be difficult and error-prone.

SAS ETL Studio provide plug-ins for the sources listed above that provide access to data from these operational systems and provides ways to search and navigate through them. These plug-ins are called SAS Data Surveyors for each type of system, and provide an easier way to find the needed content. In the figure below, the user can use a simple search string to help find the desired content.

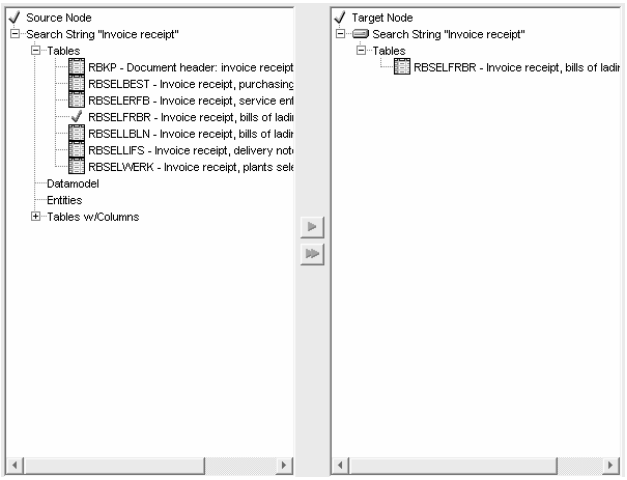

Figure 33. Using the SAS Data Surveyor for SAP R/3 plug-in to ETL Studio. The search string "Invoice receipt" is found in the descriptive text for tables. In this system, descriptive text bears very little resemblance to the actual tables names.

DEPLOYMENT

The preceding has described the administration and design of a warehouse processes. The real use of these processes occurs when the processes are deployed into the real world and scheduled for regular use. The standard scheduling package supported by ETL Studio is the LSF Scheduler from Platform Computing. The first part of the operation is to register a job for scheduling in ETL Studio.

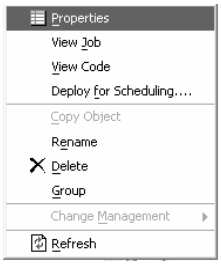

Figure 34. Actions that can be performed on a Job, showing deployment.

SAS Management Console handles the scheduling of actual jobs with the scheduler. When the job is registered with the Scheduling server, it is ready for deployment in a regularly scheduled production environment.

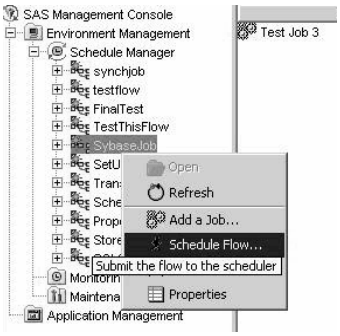

Figure 35. Job scheduling that takes place in SAS Management Console.

CONCLUSION

Version 9 of the SAS System provides infrastructure and warehousing products to meet the challenges of developing and deploying a next generation warehouse or data mart. Administrator- and developer-centric tools. An end-to-end solution encompassing data quality, deployment, and powerful analytics helps with any advanced ETL design and implementation requirements.

TRADEMARKS

CONTACT INFORMATION

The author may be contacted at:

Gary Mehler
SAS Institute, Inc.
100 SAS Campus Drive
Cary, North Carolina 27513
Gary.Mehler@sas.com