Paper 137-2011

Best Practices in Data Integration: Advanced Data Management

Nancy Rausch and Tim Stearn, SAS Institute, Cary, NC USA

ABSTRACT

The success of every business activity—from supplier management to customer service—is dependent upon how well an organization manages its critical data. This paper discusses best practices and capabilities offered by the SAS® Data Integration products for planning, implementing, and deploying a production end-to-end data management environment to support intelligence initiatives across an enterprise. In addition, considerations for advanced data management and high-performance computing are covered.

INTRODUCTION

This paper will provide details and best practice recommendations on several advanced data integration topics . The paper will discuss tips for performance improvements, reporting, and managing your data integration environment . Information provided will include settings and configuration information that will help you integrate the best practices into your own integration scenarios .

USE GRID FOR INTERACTIVE SUBMITS FOR ENTERPRISE ENVIRONMENTS

In a larger enterprise environment, multiple users may be building, testing, and running complex SAS jobs all using the same set of common SAS servers and resources . This increased workload can slow down server performance and decrease response time . Leveraging SAS® Grid Computing as the interactive submit server allows users in this type of environment to achieve scalability by using advanced workload balancing . The latest version of SAS® Data Integration Studio supports additional enhancements to its interactive Grid submit feature that leverages workload balancing without sacrificing iterative job development and debugging capabilities. You can use this type of environment as a best practice to increase your job response time and overall system efficiency .

Figure 1: Submit Options

Data Integration Studio has two possible interactive run modes: submit to a workspace server and submit to a Grid server . When using submit to a workspace server, SAS will attempt to run the job on the Workspace Server machine, even if the server is already at capacity. All jobs will compete equally for the same CPU, memory and I/O resources. In contrast, using interactive submits to a Grid server give administrators the ability to configure and automate in the following ways:

1. Preemption of lower priority jobs so that higher priority jobs may run.
2. Implementation of resource utilization thresholds (for CPU, memory, etc) on each server. If resource thresholds are exceeded, Grid will automatically suspend jobs until utilization subsides and then automatically resume the suspended jobs where they left off.
3. Migrate jobs that have been suspended due to host utilization to another host that has more capacity.
4. Prevention of resource hogging by limiting the number of concurrent jobs (or concurrently used CPUs) by a user . When a user has reached the configured threshold, their jobs will wait in a queue, giving other users processing time when the server is busy.
5. Support for a “fair share” policy . If a user A has submitted a large number of jobs in a busy environment, the fair share policy will make some of user A’s jobs wait until other users have had a chance to run. For example, if User A submits 5 jobs while users B and C each submit , in a busy environment where only a single job slot is available, a round-robin, fair share policy would dispatch the jobs like so: A, B, C, A, A, A, A.

Data Integration Studio interactive submits to Grid allows administrators more fine-grained control over their environment while providing developers the benefit of finding the server on which their job can run most efficiently.

When interactively submitting job runs to a Grid, Data Integration Studio creates a session on the Grid and submits the job for processing . In previous versions, Data Integration Studio would create a new session for each job execution and terminate the session when the job finished . Starting with version 4.3, Data Integration Studio will keep the session open until the user closes the job . This supports incremental job development better, because :

1. Intermediate WORK tables created between steps are available when the job completes to support debugging . In previous versions, these tables were deleted when the Grid session was terminated.
2. Debugging features like Run From Selected Transformation, Run Selected Transformation ,Step , and Continue can be used in all situations . In previous versions, these features were not always available when submitting Data Integration Studio jobs interactively to Grid, since the intermediate WORK tables produced by a previous run of the job were not available after the Grid session(s) terminated.

Grid must be configured to optimally support the interactive use pattern described above . The default installation of Grid defines a fixed number of job slots by setting the MXJ parameter in the lsb.hosts configuration file to the number of cores available on the host . This default setting is not suitable for interactive use of the Grid, however, because:

1. When all job slots are consumed, no further jobs may start and subsequent requests to run a job will be queued.
2. Think Time
    a. After the first time a job is run, each Data Integration Studio session (whether it is currently running or idle) consumes a job slot.
    b. It is common for developers to run a Data Integration Studio job and follow this with “think time”, in which the job isn’t running . Think time activities might include examining job results, making changes to the job prior to the next submission and possibly leaving an idle job open for long periods while attending to other matters.
    c. An available job slot is consumed by an idle job during think time.
    d. The job slot will not be released until the Data Integration Studio job is closed.
    e. If this pattern repeats enough times, other users may be unable to submit their jobs even though the Grid has plenty of capacity.

To account for the “think time” associated with interactive use of the Grid, administrators should follow this strategy:

1. **Increase the number of available job slots**
    a. Set the MXJ parameter to a number that is at or close to the maximum expected, concurrent users.
2. **Utilize “resource thresholds” to account for high concurrency**
    a. Based on recommendation #1, the number of available job slots may be set well past the number of available cores.
    b. If all available job slots were to be used simultaneously, this might push the system past its capabilities, causing instability.
    c. Administrators should use resource thresholds, which allow one to cap the maximum utilization of several load indices (CPU utilization, memory utilization, etc).
    d. If the system exceeds one or more configured thresholds, Grid will throttle down processing by suspending jobs until the host is below the configured thresholds.
    e. Suspended jobs are restarted from the point at which they were suspended when capacity is available . Administrators can configure Grid to migrate suspended jobs to a less loaded host after a certain period to prevent an indefinite wait on busy servers.
    f. Resource threshold parameters are set in the lsb.hosts LSF parameter file.

In addition to the scalability and performance advantages it provides, SAS Grid Server allows administrators to exercise fine-grained control over large, complex environments . Through improvements to its integration with Grid, Data Integration Studio 4.3 allows administrators to implement these controls without sacrificing the rich interactive job development features Data Integration Studio users rely on . Administrators must take the interactive workload pattern of Data Integration Studio developers into account when configuring Grid.

VERSIONING FOR CONTENT MANAGEMENT

When working on large projects as part of a large development team, it becomes difficult to manage change in the content that you build . For example, someone on your team may add a component to a project that does not work well with the current version of something . Or you may need to rollback to a previous version because of a problem found in the current version . Keeping track of multiple versions of your work can become time consuming and error prone . Versioning can help with this challenge.

One of the major user workflow enhancements in the latest release of SAS Data Integration is integrated versioning support . This provides you with the ability to archive content to a 3rd party versioning system of your choice . SAS package files can be archived, including jobs along with optionally expanding out the source code . Version capabilities include the ability to difference between objects, the ability to rollback to previous versions of objects, and the ability to inspect which objects are contained within a specific version .

Versioning works by moving jobs and other content into a file, called a SAS package file which uses a ZIP format for compression, and then archiving that file in a versioning system . SAS Data Integration Studio creates the SAS package file and writes it into a source management system . To bring content back into your environment, SAS Data Integration Studio retrieves the file stored in the source management system, uncompresses the content, and places it back into the SAS metadata repository so that it is available to be used in your Data Integration Studio session . In this way you can create different versions of content, view and restore previous versions of content, view and track changes, and manage different versions for audit purposes .

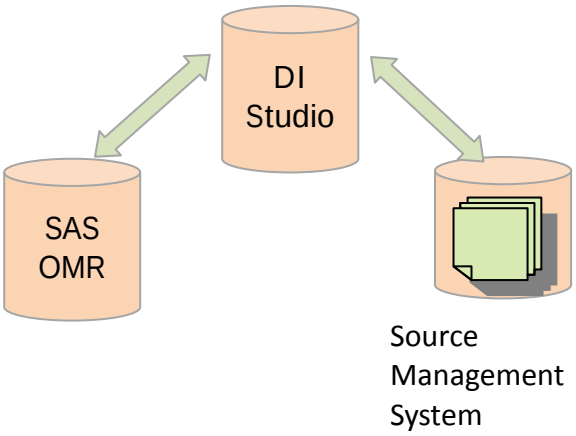

Figure 2: Versioning Architecture

Objects can be versioned independently or with other objects to make up a package of related content . This allows you to archive sets of objects that are logically related, such as all of the content in a project . Archiving sets of objects makes it easier when you roll back some changes to get to a state where objects all work together correctly, especially if the objects have had many changes applied to them over time . You can optionally also choose to generate the code for a job and store it along with the job as text content . This makes it easy to see the source code associated with a specific version of a job .

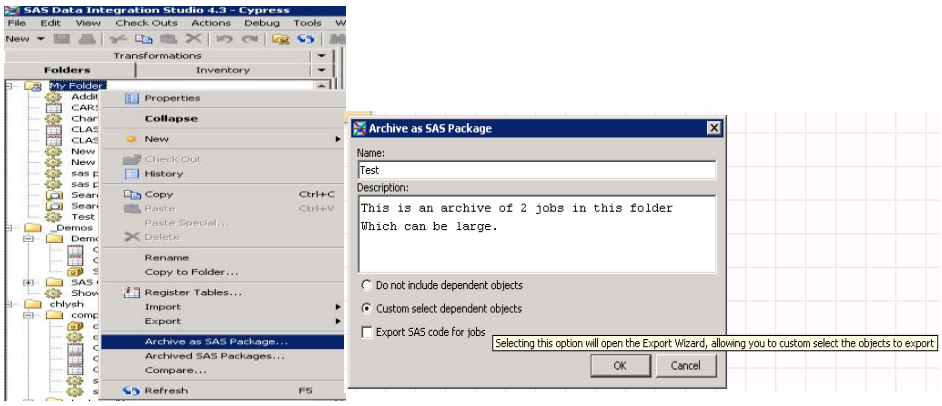

Figure 3: Versioning Option Examples

Once content is in the source management system, you can view archive results of any object to see when it was last versioned and by which user . This lets you identify previous version of objects that you may want to restore .

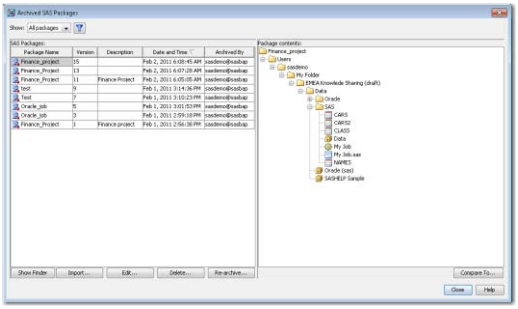

Figure 4: Version Content

There is also a differencing feature. You can select an object and view the differences between version of the selected object, or between an archived version and the current version of that object .

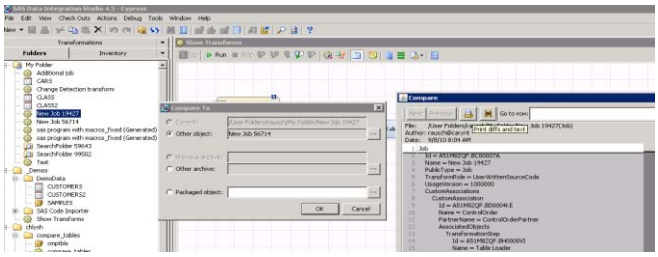

Figure 5: Differencing Options

Several out-of-the-box plugins are provided for integrating with two open source versioning systems, CVS, and SubVersion . Only one versioning scheme can be applied to a system . Users select which scheme they want to use by choosing the appropriate plugin provided with the install and moving it to the plugins directory . You will know which plugin has been selected by viewing the tools/options panel . If the plugin is appropriately installed, you will have a tab for configuring the interaction with the 3rd party plugin system in the available options panels.

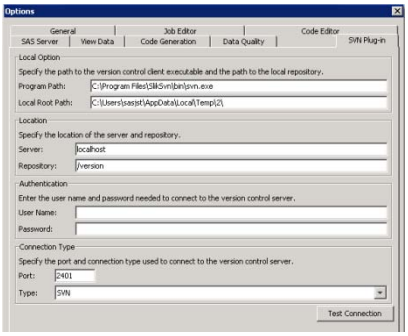

Figure 6: Configuration Options Screen

In addition, there is a documented application programming interface (API) to integrate with different source management systems .

STATUS AND PERFORMANCE MONITORING USING JOB REPORTING

As the complexity of your system grows and you place more and increasingly complex jobs into production, the need to monitor those jobs for status and performance becomes increasingly important . The enhanced job reporting capabilities in the latest release of SAS Data Integration have been developed to help meet this need . Status and performance information is captured into job logs and the performance reports and status dashboards have been developed to extract relevant information into tables that can be used for reporting . A set of prebuilt reports for several deployment options are shipped with the product . You can deploy these reports and get current and historical information on the status of your jobs .

You have several deployment options available to you for reporting . If you have the full SAS Business Intelligence suite, there are SAS Web Report Studio reports delivered with the install along with jobs to extract the log information and populate a reporting OLAP cube . This is the most robust job reporting environment; it supports drill down to understand specific areas of a job and its containing nodes, a dashboard, and other useful reporting features .

A simpler deployment option is available if the SAS Business Intelligence suite is not available . The second option uses the SAS® Stored Process Server . The install delivers some stored processes and jobs that extract the information from the logs, and provide an HTML web page with reports for job performance and status, both current and historical .

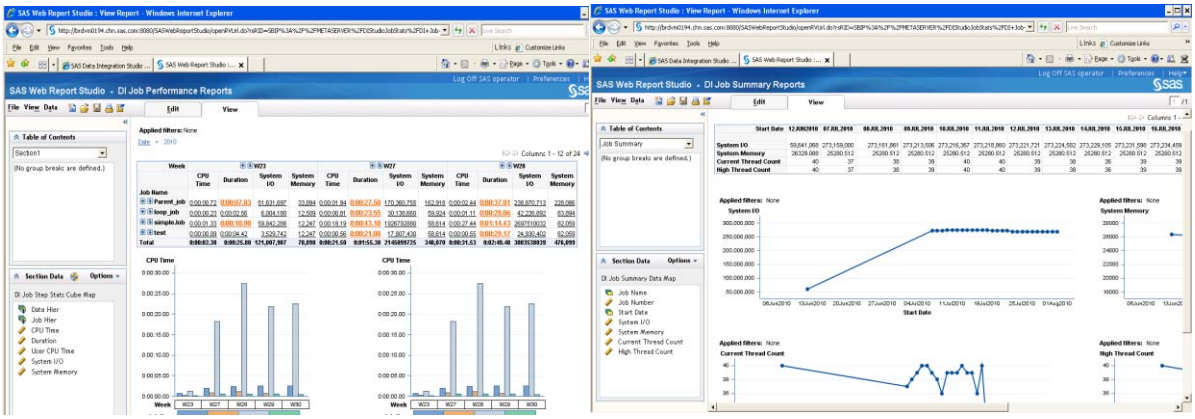

Figure 7: Example Job Reports

The above examples show the type of information that is available through these reports . You can view the overall performance of your jobs, compare one job's run against another run of the same or different jobs, and drill into a job to see which nodes contribute to the overall runtime . You can also see memory usage information for jobs, runtime and wall clock time performance numbers, and I/O use counts . You can see this information on demand, and also see historical reports of how a job or set of jobs has changed over time . You can also see job status, which jobs have run successfully and which failed, and details on any problems that may have occurred in your jobs .

All of the detailed information that is used to populate these reports comes from the job's run logs . That information is captured at runtime into the logs using SAS's Application Resource Monitoring (ARM) capabilities . These capabilities allow correlation to the hardware system that the job is being run on so that information such as memory and I/O system use can be captured and tagged to a specific job .

Using the job performance and status reporting can enable you to better monitor your system . You can use this feature to watch for problem areas, identify future computing resource needs, and better manage system complexity as it grows over time .

BATCH JOB DEPLOYMENT

When jobs are ready to put into production you may have to move your jobs from your development environment to your production environment . It may be that your site restricts access to production servers such that only some users have permission to place jobs on the production system . It is for these types of deployments that we have developed a command line interface for job deployment . By providing a command line interface, an administrator or other privileged user can place jobs into production on your behalf without having to have any specialized knowledge or experience with the Data Integration Studio application or your jobs . The command line process can also be scheduled to deploy to production if desired, and normal operating system type logging can serve as an audit trail for compliance purposes if needed .

USING THE EL-T PATTERN WITH SAS/DATAFLUX DATA INTEGRATION TOOLS

The availability and adoption of Massively Parallel Processing (MPP) databases is increasing every year . These systems promise high performance and linear scalability through share-nothing, distributed-processing architectures . While the original intent of these systems was to provide fast performance when analyzing large amounts of data, they also lend themselves to data integration tasks by supporting Parallel Processing.

Traditional Data Integration applications use the ET-L pattern: Extract, Transform and Load . Using this pattern, data is extracted from source systems into a staging area on the ETL server . Transformations are performed in the staging area with an ETL server and are loaded into the target system, often with bulk loaders for large data . This

pattern can scale to any data load by adding additional capacity to the ETL server . In addition, both SQL and procedural logical can be applied in a platform independent manner .

This pattern does not, however, take full advantage of the MPP capabilities and sometimes requires movement to and from the database server during transformation, to allow lookups to database tables.

In contrast to ET-L, EL-T operates as follows:

1. Query data from sources and bulk load it to the final target database in a staging schema.
2. Perform transformation of the data in the target database using SQL and database stored procedures.
3. Use database commands to publish data into the target schema for application usage.

This pattern takes advantage of the parallel processing power of MPP databases, which can accelerate processing while also maximizing your investment in the database's large MPP architecture . SAS and DataFlux have been steadily increasing their EL-T capabilities . Data Integration Studio supports the EL-T pattern with bulk Loaders to quickly move data from extract tables to the target database platform, SQL-based transformations that push processing to the database, and pass-Through SQL, which allows database stored procedures to be invoked as part of the process .

The screenshots below show a sample job that bulk loads a Teradata database table from an extract file, as well as support for optimized Teradata load methods (FastLoad, TPT, Restart) and Teradata loader options:

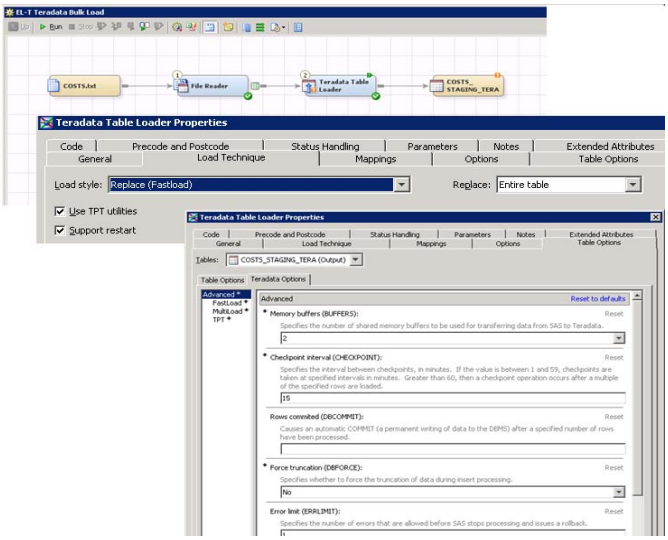

Figure 8: Example Bulk Load Options Screen

The same type of database specific load support is available for Oracle . Below is an example job that uses the Oracle Bulk Table Loader . Also depicted are options panels for the transformation, which show support for optimized load methods (like Direct Path) and pre/post load processing (index recreation, constraint disablement/enablement, statistics gathering.

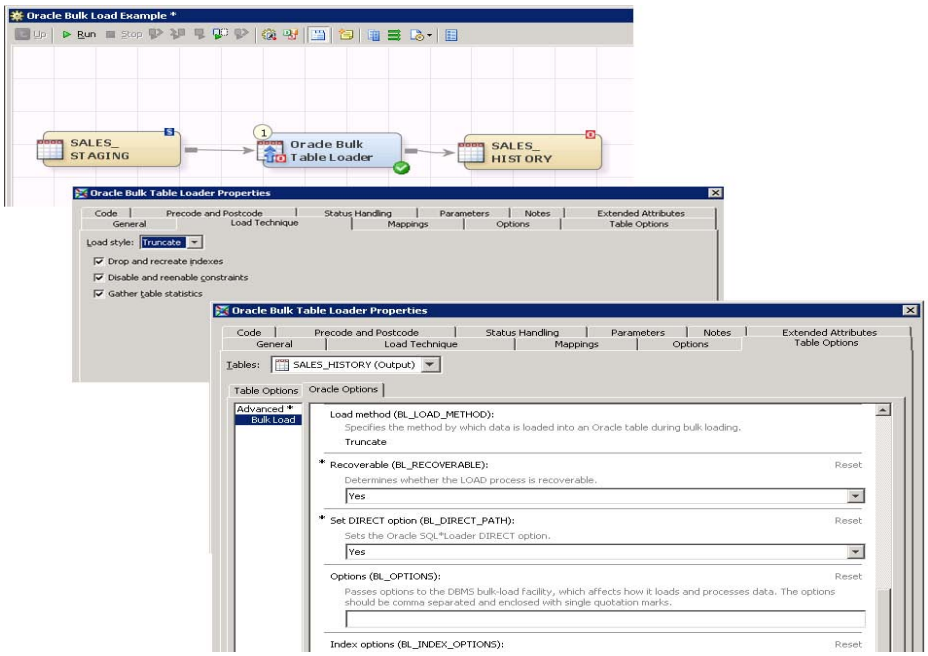

Figure 9: Additional Options Panels

While not all databases are associated with a specific loader, the generic Table Loader object can support bulk loading data into relational databases like SQL Server and DB2 through the BULKLOAD=YES and BL_OPTIONS Table Options, which allow one to specify any loader options .

Once data has been uploaded to the target database, EL-T jobs perform transformations on that data prior to data publication . This may be accomplished in Data Integration Studio through the SQL Join and SQL Set Operations transformations . SQL Join supports pushing processing down to the database . One can implement the Create Table as Select (CTAS) and Insert Into As Select (IIAS) patterns with SQL Join . The SQL Set Operation transformation supports the use of UNION/UNION ALL, INTERSECT/INTERSECT ALL and EXCEPT/EXCEPT ALL (also known as MINUS) methods . SQL set operations are almost always faster than equivalent join-based techniques, particularly for EXCEPT, which generally requires a “NOT IN()” clause if coded as a join . The screenshot below shows the Teradata bulk load job above extended with SQL Join and SQL Set Operation transformations:

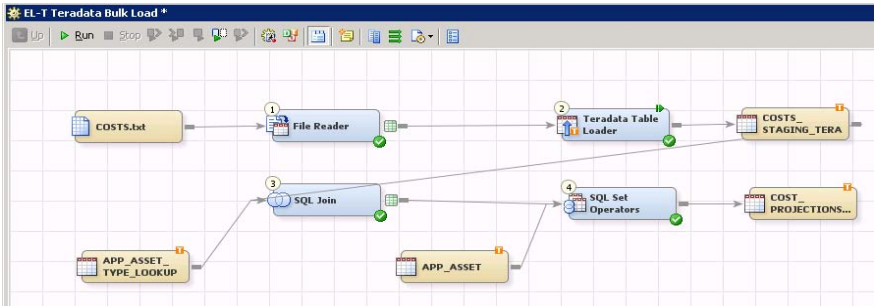

Figure 10: SQL Transformations Example

After loading the table to a relational database, it can be processed along with other RDBMS tables to implement lookups, transformations, derived columns and rule validations . Both the SQL Join and SQL Set Operations transformations have access to native database functions which are available in the Data Integration Studio Expression Builder.

Data Integration Studio EL-T jobs can also take advantage of database stored procedures. This can be accomplished through the User Written transformation, where the developer would insert SQL Pass-Through code to invoke the Stored Process .

DataFlux® Data Management Studio provides EL-T capabilities as well as part of the Process Job component . The screenshot below shows an EL-T Process Flow in Data Management Studio . The flow starts off with a Data Job, which invokes a system command to bulk load data into the target database platform (SQL Server in this case) .

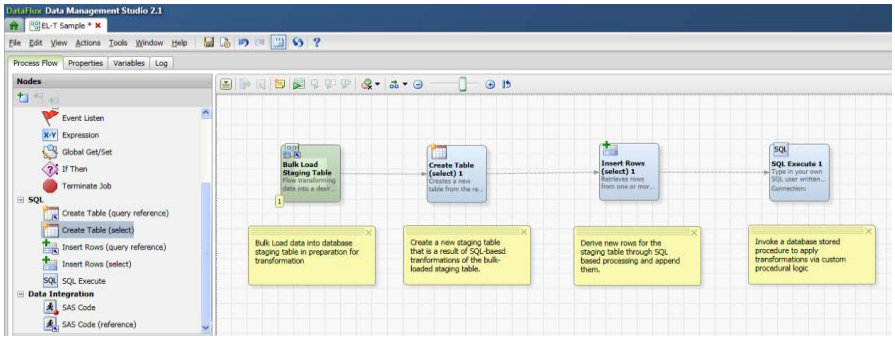

Figure 11: Example EL-T Process Flow In Data Management Studio

After uploading the data, the job applies a series of transformations via Create Table as Select and Insert Into as Select, which create a new table or insert rows into an existing table after transforming the data with SQL queries, which can make use of all database query logic and column level functions:

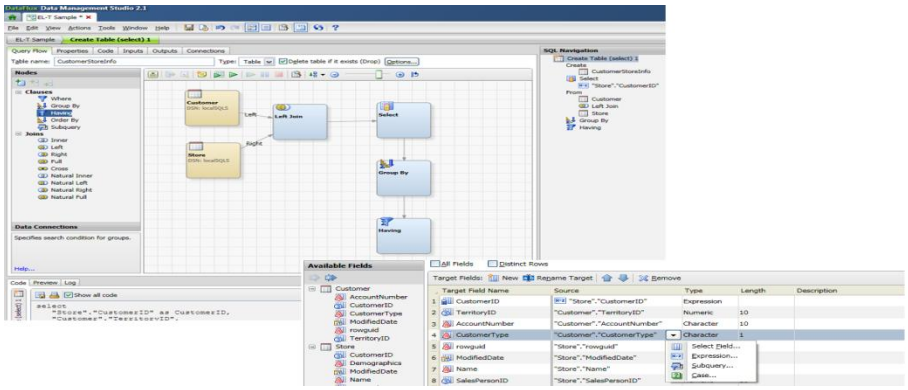

Figure 12: SQL Transformation Example

The SQL Execute transformation may be used to invoke any database statement that is legal in the target system . In this case the node is used to invoke a database Stored Procedure, which includes iterative data processing logic:

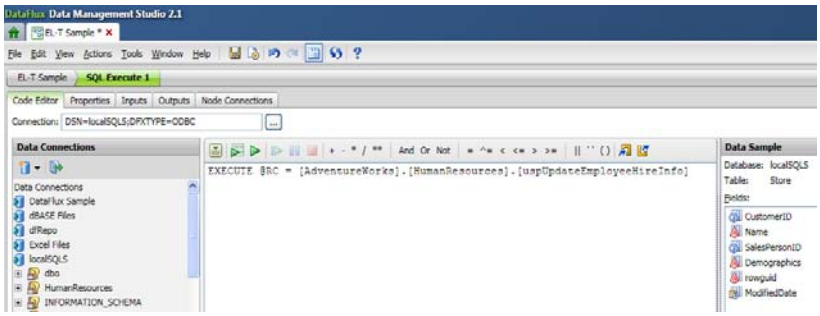

Figure 13: SQL Execute Transformation Example

While EL-T is a powerful strategy, in practice, a mix of ET-L strategies and EL-T strategies is often employed in real world projects . This hybrid approach is referred to as ETLT – Extract Transform Load Transform, where some transformations are applied to the data prior to loading it into the target database staging area . This can be useful to avoid loading unneeded data as well as to apply processing that cannot be done within the database because required 3rd party modules cannot run in-database . Additionally, the capability of a tool to support ETLT provides the flexibility to perform certain operations outside of the database in case the intensity and/or timing of the processing can't be accommodated in-database since it will interfere with BI or analytics processing, or the current size of the MPP appliance is not adequate to the scale of the processing and the MPP database cannot be scaled up in the timeframe required by the project.

In these and other cases, the flexibility to perform processing both inside and outside the database may be essential . Both SAS Data Integration Studio and DataFlux Data Management Studio support pure ET-L and EL-T approaches, as well as hybrid ETLT approaches.

DATABASE TABLE PARTITIONING: IMPROVING DATA INTEGRATION RUNTIME AND MANAGABILITY

Increasing data volumes and a trend towards “always on” data access challenge the ability of Data Integration developers to meet load windows . In fact, the concept of a “load window” is disappearing as geographically dispersed users demand more continuous access to data, which in some cases eliminates any downtime from the system . Database table partitioning is one approach to achieving acceptable performance and availability while minimizing the use of IT resources.

To an end user, a partitioned table looks identical to a regular table . At a physical level, however, the table is distributed across several different disk locations and/or storage devices . Support for partitioning methods vary by database vendor, but most provide at least the following types:

- Range Partitioning: the table designer defines several non-overlapping ranges that may be associated to a column (ex: 201102 < YEAR_MONTH < 201202) . Each range is associated with a partition in the table.
- List Partitioning: the table designer defines a list of values associated to column which define which records will be stored in each partition . As an example, if a table were partitioned by quarter, a list using a year/month field might look like (201101, 201102, 201103). Each list is associated to a partition in the table

Some vendors provide additional methods, including:

- Hash Partitioning: the table designer designates a “hash function” – often the modulus operator - to distribute data evenly across partitions . For example, if a table will have 10 partitions, the partition function might be: mod(<numeric column>,10) = 0 . Even distribution of data is important for consistent query performance . It is important to make sure the hash column is relatively evenly distributed.
- Composite Partitioning: This partitioning method combines two partitioning methods above to create two levels of partitioning. A common practice is to combine Range or List partitioning with Hash partitioning to create additional distribution of data within a Range or List definition . This allows parallel partition scans for a range/list during reporting or analysis queries.

The Data Integration Studio developer can make use of these techniques in a number of ways. Large fact tables may contain hundreds of millions or even billions of rows . If you tried to load this type of volume from a single large staging file into a unpartitioned database table, performance would be slow . The use of partitioned tables along with Data Integration Studio parallel-processing features allows one to load each partition in parallel, greatly decreasing the time required to load the table . This is particularly useful for the initial load of a large table.

The screenshot below shows a job that implements such a parallel loading scheme.

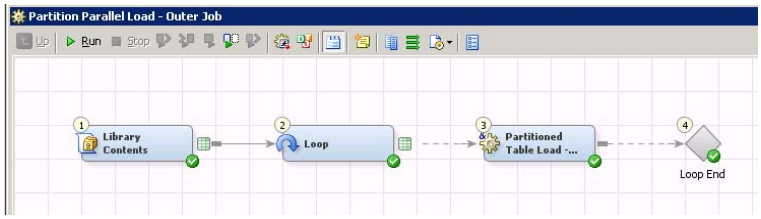

Figure 14: Example Parallel Loading Job

The job above utilizes the Loop transformation to iterate through a set of staging tables. The list is produced using the Library Contents transformation to query a staging library for all tables that follow a particular naming convention . The Loop transformation invokes an inner job for each staging table, passing parameters for the table to load as well as the partition name to load . The Loop transformation may be configured to run each iteration in parallel . Parallel runs may use multiple CPUs on a single server or distribute the jobs across a SAS Grid .

When working with the Loop transform, the inner job is parameterized and receives parameter values from the Loop transformation . These parameters are available to the inner job as macro variables, which may be used for any property within the job . In this case, the STG_TBL_NAME parameter is used to parameterize the physical table name of the staged load table, allowing each inner job to process a different staged load table . The PART_NBR parameter is used to form the complete partition name to be loaded as part of the Oracle Bulk Loader settings.

The screenshots below show the basic job setup and property customizations that use job parameters.

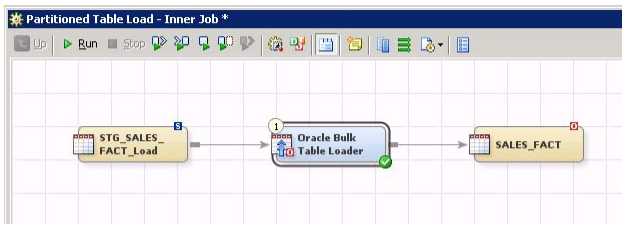

Figure 15: Inner job example

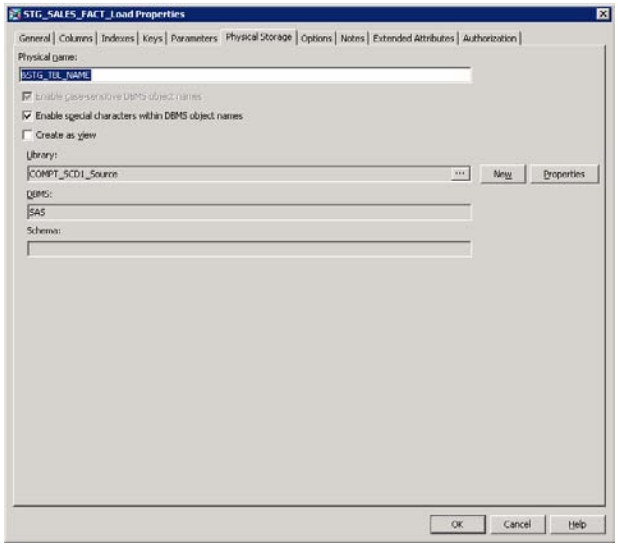

Figure 16: Input Table Physical Storage Metadata

Taken together, these capabilities allow you to increase the performance of your load jobs, limited only by the available resources (CPU, memory, I/O bandwidth) on your servers.

Local (Partition Aligned) Indexes

One of the most costly operations in loading a large table is maintaining indexes after a large load . For example, one recent customer had to maintain a 300 million record fact table, while inserting 8.3 million new records per month . Rather than rebuilding the indexes for the all 300 million records each month, the table was designed with month/year partitions and local/partitioned aligned indexes . Local indexes are constrained to each partition and the database allows them to be rebuilt without affecting the indexes for any other partition .

The ability to rebuild such indexes independently saves an enormous amount of time, which contributes to making data available sooner . This technology also allows unaffected partitions to remain available during new partition preparation/maintenance.

Partition Exchange

This technique is pivotal for incremental maintenance of partitioned tables . It allows new partitions to be built and indexed independently of the main table . When the new partition is ready, it is then exchanged or swapped with an existing partition in seconds or less, unobtrusively making the new data available for analysis . The data can be merged with the partitioned table so quickly because it is stored in the same physical location as the exchanged partition, making this a zero-data movement (or just metadata) operation .

Most databases transparently provide controls such that queries that were running on the partitioned table prior to the exchange request are allowed to complete against the existing data while new queries wait the miniscule amount of time it takes to complete the exchange operation . This allows new data to be added to the partitioned table while the table is in use and ensures that the queries return consistent results.

This technique is illustrated in the job screenshot below, along with explanations of each step:

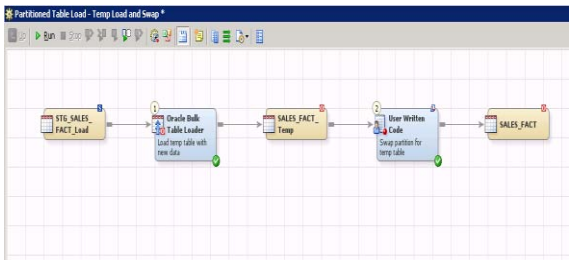

Figure 172: Example Job Using Parallel Processing

The first step is to bulk load the records for the new partition into a temporary table that has the same physical storage characteristics as the final partitioned target table . Note that the details for physical storage characteristics vary by database. For example, Oracle would require that the temporary table and the partition must reside in the same table space. For SQL Server, the temporary table and the final partitioned target table must share the same partition scheme. The second step is to build all indexes and enable constraints on the temp table. In the last step, the database specific statements required to “exchange” the temp table with the target partition would be run . These statements are illustrated in the screenshot below:

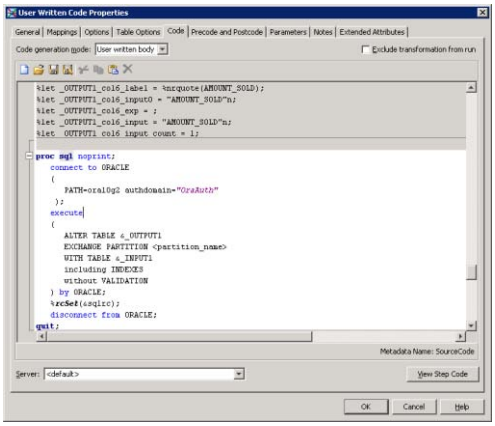

Figure 18: Example Database Syntax

As shown in the example above, this is performed through the use of Pass-Through SQL . The macro variables above are passed to the job by the User Written Code transformation and contain the required schema and table name information. The <partition_name> shown above would be replaced with the appropriate name from your naming scheme.

Round-Robin/Sliding Window Partitioning

This is a partitioned table maintenance technique that makes use of all of the other techniques described above . This is applicable when a specific data retention period has been defined by the business owners of the table . In the case of the pharmaceutical company mentioned above, they required that 36 months of sales data be retained online in a fact table . Defining such a retention period and marrying it with a round-robin partitioning scheme allowed the amount of data stored in the table (~300 million rows) to remain constant instead of growing by 8.3 million records per month . This was accomplished in the following steps:

1. Exchanging the oldest partition in the table with an empty archive staging table, creating an empty partition where the oldest data used to be.
2. Dynamically updating the table partition metadata to contain a new month/year value.
3. Exchanging the new data into the empty partition, as described under the “Partition Exchange” section above.

The approach is sometimes referred to as a “sliding window” . In this case, the “window” had a length of 36 months . It “slides” each month, moving forward in time, at which point the oldest data falls out of the table and the newer data is added to the front. This approach - which retained an approximately constant table size - produced several benefits: the amount of storage required to store the table online was fixed, holding total storage costs steady, and query performance against the table remained constant, since the amount of data stored in the table did not fluctuate appreciably from month to month.

MASTER DATA MANAGEMENT

Master Data Management is a technique that is being used increasingly by data integration specialists to manage and optimize source data . Master Data Management is essentially the ability to develop the best possible record for a specific resource, such as a Customer or a Product, from all of the source systems that might contain a reference to that resource . For example, a company with thousands of customers would consider customer an important resource and would like to have the actual person that is represented by the customer record be the same, whether the person is being contacted to notify them about an upcoming sale, or to understand if they recently called in with a problem . In other words, the Customer resource should be mastered or standardized so that the same record can be used throughout a company's different business practices . The value that mastering a resource such as Customer has to a company is that companies rely upon these resources to conduct their business . Once a data record has been mastered, it can be used as a source to all downstream consumers of the record . The mastered record represents accurate, up-to-date information that all consumers of that resource can rely upon .

SAS provides the ability to master data in this way using the SAS/DataFlux qMDM solution . This solution is a combination of software, templates, documentation, data model, and services that together provide the functionality and processes necessary to build and maintain a master entity database. An entity can be a customer, a product, a patient, a site or any other business data object you might define.

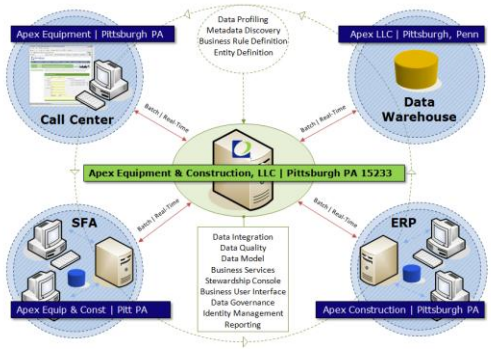

Figure 19: qMDM Architecture

The basic architecture for the qMDM Solution uses SAS' Data Management platform to build and manage the qMDM database. Data Management Studio can be used to build and maintain the qMDM, and Data Management Server and the associated Service Oriented Architecture can add support for real-time/transactional usage . There are several UI components available to interact with the mastered data in the qMDM database . There is a web administrative web based interface that supports updating records in the database, splitting apart or merging sets of records together, adding additional attributes, and adding information about data hierarchies . Additionally there are desktop user interfaces available to build, test, and deploy hubs .

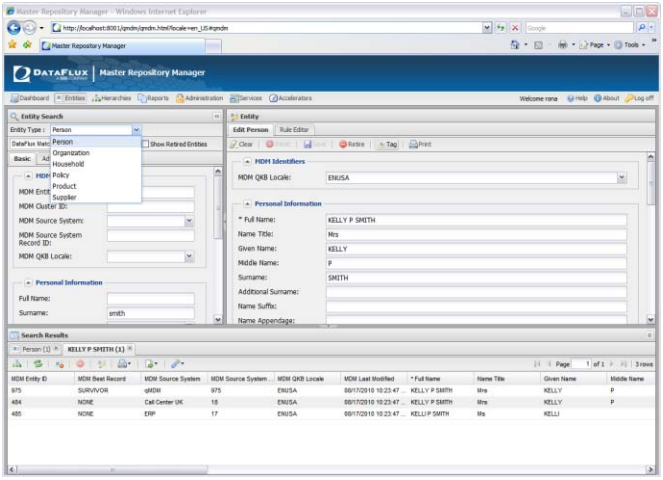

Figure 20: Master Data Manager Screen

When first deploying a master data solution, there can be a certain amount of iterative work involved in identifying the data silos that contain the data resource of interest, collecting that data from all of the identified source systems so that it can be cleansed and ready to load into the master data hub, and then building out the processes with the correct data rules to create the master records in the master data hub . During this process you will need to inspect the results and iterate through the load processes until the results appear as you expect. Part of the inspection process is to identify and fix problem areas . What you want to look for as potential problem areas include clusters that are too large (clusters containing hundreds or thousands of records) clusters that are potentially too small (containing 1 record), and data that is missing, out of place, or does not meet standards .

The qMDM solution provides out of the box reports and viewers for viewing clusters, identifying outliers, and fixing problem areas .

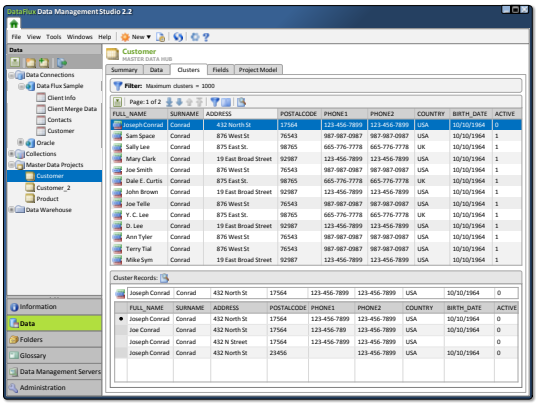

Figure 21: Cluster Visualization Example

The matching of records to identify records that are the same person from a set of similar records is known as “clustering” . This technique is something that is difficult to do with existing SQL or transformation logic . Often matching records together to generate clusters is better left to sophisticated technology such as the SAS/DataFlux match code technology . This technology includes techniques for probabilistic matching, that is if two records are similar to each other the DataFlux technology can create a score based on configurable rules as to how likely or how probable the records match, and are therefore correct to include in a cluster .

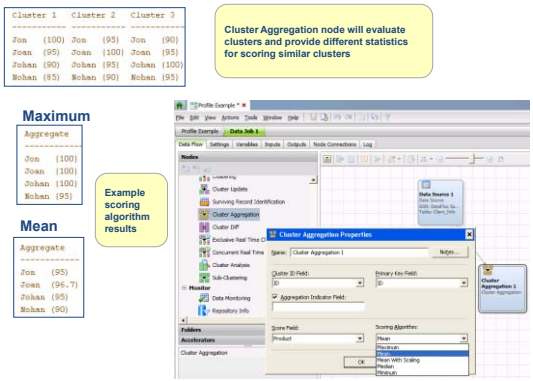

Figure 22: Example of Probabilistic Matching

Once the data has been clustered into master records, you can choose to store the data in the master hub or optionally update the source systems by propagating the master records back to the sources . SAS’s qMDM technology supports both models and there are pros and cons to either technique . Writing back data to the source systems where the records came from ensures that all systems have a single view if the truth at all times . However If you have many 3rd party systems contributing records to your master system, it may not be practical to persist updates back to those systems . Persisting data into a separate master hub provides a single source system that all downstream processes can pull from, which can be ideal in some scenarios . This approach can be a problem however if two contributing systems have updates for the same user at the same time that are different, such as two different addresses . In this case, one system will have to win, and a choice will have to be made .

In either of these scenarios, it is important to appoint a data steward, someone that can review and manage the data that is being mastered and who is a domain expert . Their role will help manage the data such that it stays continuously cleansed and mastered for all consumers independent of which technique you choose to manage the master records .

CONCLUSION

The success of every business activity—from supplier management to customer service—is dependent upon how well an organization manages its critical data. This paper discusses best practices and capabilities offered by the SAS® Data Integration products for planning, implementing, and deploying a production end-to-end data management environment to support intelligence initiatives across an enterprise. The tips and techniques provided can help speed up your development and job deployment performance .

ACKNOWLEDGMENTS

The authors wish to thank Jeff Stander and Mike Ames for providing screenshots and other content for this paper.

RECOMMENDED READING

- SAS Enterprise Data Management and Integration Discussion Forum, Available at http://support.sas.com/forums/forum.jspa?forumID=59
- Rausch, Nancy A., and Tim Stearn, 2011, “What’s New in SAS Data Integration”, Proceedings of the SAS Global Forum 2011 Conference, Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings11/134-2011.pdf .
- Ames, Michael and Steve Sparano, “ On the Horizon: Streaming Integration and Analytics”, Proceedings of the SAS Global Forum 2011 Conference, Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings11/404-2011.pdf .
- Hazejager, Wilbram and Pat Herbert, “Innovations in Data Management – Introduction to Data Management Platform”, Proceedings of the SAS Global Forum 2011 Conference, Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings11/141-2011.pdf .
- Hazejager, Wilbram and Pat Herbert, “Master Data Management, the Third Leg of the Data Management Stool: a.k.a. the DataFlux® qMDM Solution”, Proceedings of the SAS Global Forum 2011 Conference, Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings11/146-2011.pdf .
- Stander, Jeff. 2010. “ SAS® Data Integration Studio: Tips and Techniques for Implementing ELT.” Proceedings of the SAS Global Forum 2010 Conference. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings10/116-2010.pdf .
- Hunley, Eric, and Nancy Rausch. 2009. “What’s New in SAS Data Integration Studio 4.2.” Proceedings of the SAS Global Forum 2009 Conference. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings09/093-2009.pdf .
- Doninger, Cheryl, and Nancy Rausch. 2009. “Data Integration in a Grid-Enabled Environment.” Proceedings of the SAS Global Forum 2009 Conference. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings09/098-2009.pdf

CONTACT INFORMATION

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

Nancy Rausch
SAS Institute Inc.
Cary, NC 27513
Work Phone: (919) 677-8000
Fax: (919) 677-4444
E-mail: Nancy.Rausch@dataflux.com
Web: support.sas.com

Tim Stearn
SAS Institute Inc.
Cary, NC 27513

Work Phone: (919) 677-8000
Fax: (919) 677-4444
E-mail: Tim.Stearn@sas.com
Web: support.sas.com