Paper 099-31

Base SAS® vs. SAS® Data Integration Studio: Understanding ETL and the SAS Tools Used to Support It

Danny Grasse and Greg Nelson
ThotWave Technologies, Cary, North Carolina

Abstract

Every data warehouse, data mart and data hut needs it. Every good business intelligence interface depends on it. It has been the hallmark of what SAS programmers have done for over 30 years –beat data into submission (a.k.a. ETL - extract, transform and load data). Now all grown up and ready to take on the world, SAS® software's ability to get at just about any data source, massage, cajole, beat, cleanse and transform information and then make it sing and dance in your tool of preference makes SAS the ideal choice for modern decision-support applications.

So which tools, technologies and/or approaches can we use for ETL and which make sense for any given application? We will explore the world of ETL from the perspective of the tasks that we have to accomplish and compare using Base SAS tools versus Data Integration Studio (or as it is now known – SAS Data Integration Studio). To that end, we will highlight what a good ETL system should be able to do by taking a lesson from Ralph Kimball and his book and articles that outline the 38 subsystems for ETL. We will touch on several key tasks found in ETL and show you how to accomplish these using both Base SAS and SAS Data Integration Studio. In addition, we will summarize the major capabilities of each approach as a quick reference for management.

Introduction

ETL – or Extract, Transform and Load – has long been an integral part of the vocabulary of data warehousing practitioners. One of the fundamental concepts in data warehousing is that we have to maintain a single version of the truth. One version of the truth means that we cannot necessarily rely on the fact that the original source data, often contained in operational systems, won't change over time. Moreover, the truth may be derived from combining lots of different data sources. This data integration is powerful – it's what creates the intelligence out of data. To protect the integrity of the data – and to maintain operational equilibrium between operational systems and decision-support systems – it was decided that we needed to "suck" data from these operational or tactical/data-to-day applications. The data then has to be staged in a location that gives us some ability to control changes over time and, of course, integrate that data with other useful bits of data from other parts of the organization. The overall process of extracting data, massaging it into something useful and then loading it into a

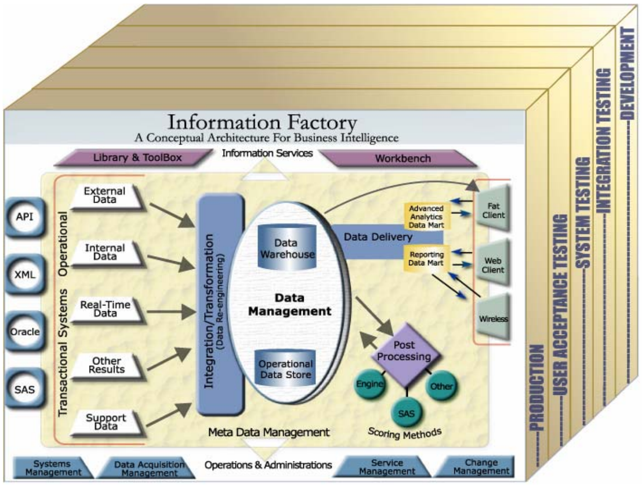


Figure 1. Corporate Information Factory.

non-volatile environment or data warehouse is referred to as ETL.

Note that data in the diagram above flows from left to right. As it leaves the operational systems, it is integrated, conformed and made useful by the ETL process. (Diagram above adapted from Imhoff & Sousa, 2002).

Since the mid 1980s, we have learned a lot about what is practical and useful as it relates to data warehouse design as well as what things should happen in these processes. One of the thought leaders in this area has been Ralph Kimball. From his early days, he described a number of design patterns that was eventually known as "dimensional" data warehouse design (Kimball, 1996; Kimball & Margy, 2002). As SAS programmers, we know intuitively that Dr. Kimball was right. For example, the idea of writing reports that require understanding, navigating and eventually joining data contained in a 3rd normal form relational database was depressing at best. In his first book (Kimball, 1996), he described the necessary design that would make the data not only useful, but that gave us the ability to capture changes in dimensions.

Subsequent to this early work, Dr. Kimball has blessed us with a rich history of articles and commentary on the state of data warehousing. One area that recently was publicized was his work on best practices for ETL. In his seminal article, "The 38 Subsystems of ETL: To create a successful data warehouse, rely on best practices, not intuition," Dr. Kimball described the necessary components that every ETL strategy should contain.

Note that we used the term "strategy" and not tool or system. It is clear from reading his article and subsequent book that he is not giving us a cook-book approach to ETL, but rather an integrative look for how we should approach ETL.

Later in this paper, we will attempt to categorize those 38 subsystems and apply what we know about SAS to how these might be handled in SAS. Specifically, we wanted to take the user through some of the more common tasks that we would ask of any ETL tool (i.e., software) and compare what we can reasonable do with Base SAS and SAS Data Integration Studio, a modern ETL tool that supports GUI-based development.

Evolution of SAS

Before we dive into Kimball's review of what subsystems are required for a good ETL strategy, let's first talk about SAS as a toolset. SAS has been around for over 30 years – starting out in academic research as a set of foundational tools for statistical analysis. One of the early inventions at SAS was the DATA Step. The DATA Step, SAS software's answer to how we process records of data, is still a core component of the SAS System. Over time, SAS recognized that in order to satiate the need for statistically derived results, you had to not only massage or transform the data but also present that data in meaningful and relevant ways. As SAS evolved from a set of PROCs (procedures) and the DATA step to an entire platform for decision support, so. too, the tools evolved from a 4GL (4th Generation language) to something more akin to what users need to build complex data warehouse, business intelligence and advance analytics applications. So as we think about just one piece (ETL) within that entire information architecture, let's not forget that a clear benefit of SAS is that it does answer the call for an integrated, emprise-class platform.

ETL Best Practices

As Kimball cites in his article, we typically spend about 70 percent of our time in data warehousing "doing" ETL. What are these tasks/activities or as Kimball called them "subsystems" of ETL all about? First, let's categorize them into a subset of concepts that we could reasonably cover in paper of this type. As we started to dissect Kimball's 38 subsystems, we decided the following six areas were represented well enough to show value in this paper. These include:

1. Design and data profiling.
2. Source data extraction.
3. Transformation and loading.
4. Change data capture.
5. Quality review, auditing and exception handling/management.
6. Integration with the production environment and business process components.

While we don't pretend to cover all 38 subsystems in detail, categorizing them into these six major areas proves helpful in evaluating the capabilities within SAS. Below we briefly describe which of the 38 subsystems aggregate up into these six areas of ETL "competency" and then describe how you would accomplish these using a combination of SAS tools and our own best practices. Not everyone will agree 100 percent on the best way to do any one thing in SAS. We do not expect our competitors in the SAS consulting space to share our opinions on what is best practice, but after 20 years in the business and over 50 enterprise-class, data-warehousing projects behind us, we think there are some lessons to be learned and shared.

Design and Data Profiling

While Kimball makes no overt reference to data-warehouse design activities in his "38 sub-systems" article, there is clearly a strong undercurrent of "plan – design – replan" in his work. This is especially clear in his newer book on ETL (Kimball and Conserta, 2004). We strive for a good balance between up-front planning and design, but we also tend to be practical in our efforts. Like good decision makers, we try to make those design decisions in the context of a good understanding of what is actually in the data. To that end, we believe a logical and physical design for any data warehouse should include a healthy dose of reality. Make sure you build time into your process time to actually go out and explore the data. Look at the missing values. Is the cardinality in the data what you expected? I know, the data administrators told us that the data was clean, is it?

In Kimball's article, we find the following subsystem most relevant here.

> **3. Data profiling system.** Column property analysis including discovery of inferred domains, and structure analysis including candidate foreign key — primary relationships, data rule analysis, and value rule analysis.

In SAS the critical first step is to look at the data -- generate listings, frequency counts and summary statistics on the fields in the source data systems. The classic SAS programmer will no doubt recognize this as one of those steps that they take without thinking. Over the years, we have all seen specialty macros published in papers and on Web sites that do exactly that kind of thing. Even in our own *thinking data*® Toolkit for Data Warehousing we have a number of macros that automate the discovery of data anomalies.

In the new world of GUI-based tools, it is really quite easy to get a quick understanding of your data before you make fundamental design decisions. DataFlux, a wholly owned subsidiary of SAS, has created a flagship product, DataFlux® dfPower® Software, that allows you to visually inspect your data quickly and without programming. It is designed for the business user or IT analyst to support data profiling via an intuitive point-and-click interface. This solution allows users to connect directly to their data sources and automatically discover and correct data quality issues. As a design time tool, dfPower helps users understand important information about the quality of their data and includes column-level statistics (e.g., count, null & blank counts, min, max, etc.), frequency distributions (including pattern counts and uniqueness useful in determining de-duplication/cleansing strategies) as well as relationship analysis and outlier analysis. Relationship analysis helps uncover redundant values across multiple tables, i.e., helps you highlight orphaned records.

Unfortunately in Data Integration Studio, we don't find much in the way of data profiling other than what you can do in Base SAS – namely what you could get out of a PROC Contents. Integration with dfPower might be recommended.

Source Data Extraction

SAS software has the ability to get at just about any data source in the world. From relational database systems to legacy mainframe systems to flat file and proprietary applications, SAS can either access them natively (native access engines) or through any number of methods including ODBC/JDBC or open standards supported with SAS® Integration Technologies. As a foundation, these are supported in both Base SAS as well as tools like Data Integration Studio. In the Fall of 2005/ Winter of 2006, DataFlux software will be able to take advantage of SAS' extensive access engines.

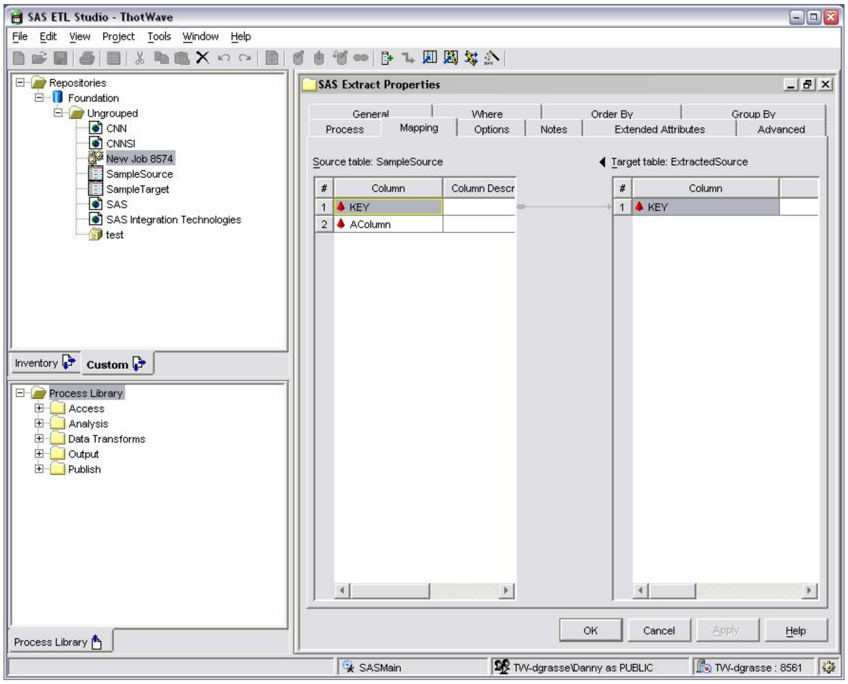
Kimball gave direct reference to data extraction in one of his subsystems:

> **1. Extract system.** Source data adapters, push/pull/dribble job schedulers, filtering and sorting at the source, proprietary data format conversions, and data staging after transfer to ETL environment.

From our read of this subsystem, we wanted to comment on the following attributes.

- Source data adapters (including data conversions and filters).
    - As mentioned above, SAS can access just about any known data source through their native access engines and open standards approaches. This functionality is surfaced in both Base SAS and Data Integration Studio through metadata.
    - The SAS System has over 200 numeric, character and date conversion routines or functions that allow for complex manipulations on the data. These are accessible through both the DATA Step and through PROC SQL. By definition these are also available in Data Integration Studio.
    - Data Integration Studio has built-in access to all SAS formats and conversion features. Whenever a datum is accessed, in-formats, formats, mathematical manipulations, substring-ing, etc., can all be applied through its GUI interface. And, as described below, you can always make 'user exits' – calls to custom SAS code - to accomplish the step.
- Push/pull/dribble.
    - We will touch on this later when we talk about how you might productionize the ETL process, but there are a number of methods to feed a SAS process – whether we pull data from the source system, push it out, in real time or staged, or create a mechanism for slowly feeding the data throughout the day.
- Filtering & Sorting.

- Base SAS has conditional logic support ('if-then-else'), so you can act on whatever observations or columns you wish. Data Integration Studio has a built-in extract transform that provides this need.



Source table: SampleSource

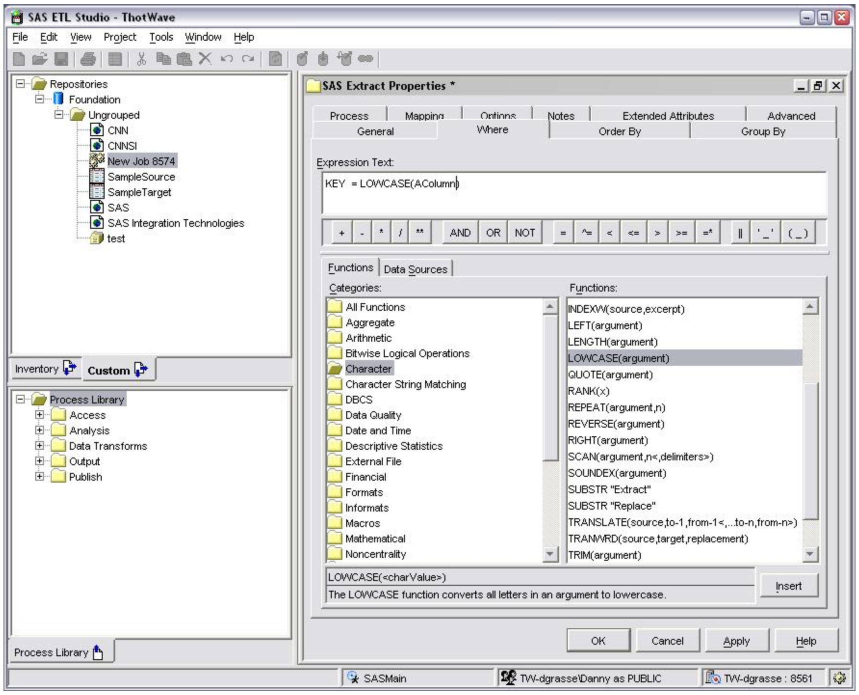
#	Column	Column Descr
1	KEY	
2	AColumn	

Target table: ExtractedSource

#	Column
1	KEY

Figure 2. Data Integration Studio: Dropping a column while extracting from a source table.

- In the provided images you see that a subset of columns is pulled across, and only those rows matching the condition are retained. We've found it useful to pull across exactly the rows /columns needed in the 'E' of ETL for the subsequent 'T'. If additional columns are needed later, pulling them through is simple in Data Integration Studio.



Expression Text:

```
KEY = LOWCASE(AColumn)
```

LOWCASE(<charValue>)
The LOWCASE function converts all letters in an argument to lowercase.

Figure 3. Data Integration Studio: filter out rows on some criteria using the 'where clause' builder.

- Data staging (versus accessing).
  - SAS doesn't require the data to be staged separately from the source system, although it is certainly possible. That is, data can be accessed in place in its native form without having to land the data a second time.

Transformation and Loading

Those of us who have been involved in ETL for a while will quickly realize that transformation and loading is truly the meat and potatoes of the data warehouse. This is where we do the magic that connects source systems and target systems to create usable data huts for consumption. Our methodology calls for the creation of design documents around how this mapping should occur. In fact, we call for an SMT document that is integral to the design of a data warehouse. SMT stands for source-movement-target, and is a document that describes what is to be accessed; including formats and expected values, how it should be moved or transformed, and what the target values should look like (again, expected values, conformed dimensions, etc.). This design document is an essential tool for ensuring that everyone involved is speaking the same language. Its detail is the 'one version of the truth' for mapping the source data to the final target structure. This is especially useful with Data Integration Studio since, when creating a job in the Process Designer, you work backward. Knowing what the target table should look like is pretty much a requirement.

Throughout Kimball's work, he spends a considerable amount of time and effort making sure we understand the role of the transformation and loading parts of ETL. In his article, he gave reference to 16 of his subsystems that pertained to our ability to beat data into submission and get it into the right format for the target tables. Those included the following:

> **5. Data conformer.** Identification and enforcement of special conformed dimension attributes and conformed fact table measures as the basis for data integration across multiple data sources.
>
> **9. Surrogate key creation system.** Robust mechanism for producing stream of surrogate keys, independently for every dimension. Independent of database instance, able to serve distributed clients.
>
> **12. Fixed hierarchy dimension builder.** Data validity checking and maintenance system for all forms of many-to-one hierarchies in a dimension.
>
> **13. Variable hierarchy dimension builder.** Data validity checking and maintenance system for all forms of ragged hierarchies of indeterminate depth, such as organization charts, and parts explosions.
>
> **14. Multivalued dimension bridge table builder.** Creation and maintenance of associative (bridge) table used to describe a many-to-many relationship between dimensions. May include weighting factors used for allocations and situational role descriptions.
>
> **15. Junk dimension builder.** Creation and maintenance of dimensions consisting of miscellaneous low cardinality flags and indicators found in most production data sources.
>
> **16. Transaction grain fact table loader.** System for updating transaction grain fact tables including manipulation of indexes and partitions. Normally append mode for most recent data. Uses surrogate key pipeline (see subsystem 19).
>
> **17. Periodic snapshot grain fact table loader.** System for updating periodic snapshot grain fact tables including manipulation of indexes and partitions. Includes frequent overwrite strategy for incremental update of current period facts. Uses surrogate key pipeline (see subsystem 19).

18. Accumulating snapshot grain fact table loader. System for updating accumulating snapshot grain fact tables including manipulation of indexes and partitions, and updates to both dimension foreign keys and accumulating measures. Uses surrogate key pipeline (see subsystem 19).

19. Surrogate key pipeline. Pipelined, multithreaded process for replacing natural keys of incoming data with data warehouse surrogate keys.

20. Late arriving fact handler. Insertion and update logic for fact records that have been delayed in arriving at the data warehouse.

21. Aggregate builder. Creation and maintenance of physical database structures, known as aggregates that are used in conjunction with a query-rewrite facility, to improve query performance. Includes stand-alone aggregate tables and materialized views.

22. Multidimensional cube builder. Creation and maintenance of star schema foundation for loading multidimensional (OLAP) cubes, including special preparation of dimension hierarchies as dictated by the specific cube technology.

23. Real-time partition builder. Special logic for each of the three fact table types (see subsystems 16, 17, and 18) that maintains a "hot partition" in memory containing only the data that has arrived since the last update of the static data warehouse tables.

24. Dimension manager system. Administration system for the "dimension manager" who replicates conformed dimensions from a centralized location to fact table providers. Paired with subsystem 25.

25. Fact table provider system. Administration system for the "fact table provider" who receives conformed dimensions sent by the dimension manager. Includes local key substitution, dimension version checking, and aggregate table change management.

While we don't have the time nor energy to comment on each of these in detail, we “thot” it would be important to talk about a few of them.

- Conforming dimensions and facts.
  - Integrating data from multiple sources and formats is a key component to successful data warehousing. Data integration means creating one version of the truth in how we reference information – particularly dimensions. If one system used male and female and another M and F, we normalize these values into a single set of values – most likely the ones that provide business value to those reporting on the data.
  - To normalize the data, we not only make the data structurally identical, but also filter out invalid values, standardize the content, de-duplicate the values and create that single version of the truth. Common tasks for software in this process includes standardizing, matching and surviving – or populating the target tables with those that are deemed as “good”.
  - In Base SAS we have a tremendous amount of functionality at our disposal for doing the matching and retention of good values. In Data Integration Studio, we have standard transformations as well as the ability to create customer transformations that can be dropped onto the process editor. And in DataFlux software, we have a robust library to transformation routines that not only helps us profile the data for quality, but also productionizes the standardization process.
- Creation of surrogate keys.
  - A surrogate key is an identifier that is created for our facts and dimension tables so that we don't have to worry (nor be confused by) the original key found in the operational system.

    This key is used to match fact records with dimension records and is usually a sequential number starting at 1. We simply increment the values each time a new record is added. Unfortunately in SAS, we don't have an auto-incrementer, but this is very easily accomplished in both the DATA step and in PROC SQL.

    - In Data Integration Studio, there is a surrogate key generator that will handle the creation of a new key for you when you load the table. Of course, this can also be done in Base SAS.

- Building summary tables and cubes.
    - Once the star schema is built, often you will want to create a summary or aggregate table that is more appropriate for reporting in cases when the details records are not required. As SAS programmers, we are intimately familiar with the concept of creating summary level tables and understand the nuances that drive these decisions whether or not we can report off of summary data or detail is required. Of course for most statistical analyses, we require the detail-level data. Our perspective for analytic data warehouses is to never throw away the details as you can always create summary views of the data.

Change Data Capture

One of the fundamental design decisions to be made is how we deal with change over time. Just like death and taxes, it is inevitable. A number of subsystems speak to the concept of change-data capture. Kimball gave reference to three of his subsystems that pertained to how we should manage change in our ETL strategy:

> **2. Change data capture system.** Source log file readers, source date and sequence number filters, and CRC-based record comparison in ETL system.
>
> **10. Slowly Changing Dimension (SCD) processor.** Transformation logic for handling three types of time variance possible for a dimension attribute: Type 1 (overwrite), Type 2 (create new record), and Type 3 (create new field).
>
> **11. Late arriving dimension handler.** Insertion and update logic for dimension changes that have been delayed in arriving at the data warehouse.

From our read of these subsystems, we wanted to comment on the following attributes.

- Slowly changing dimension processor.
    - As is typical of a data mart or warehouse that uses a star schema, we often want to keep track of changes in data that might affect how we report on that data over time. Specifically, Kimball talks about Type I, II and III changes that will affect how we handle the data coming over from the source system and update/replace those records in our target system.
    - When we see new data coming in from the operation system, we have to make a decision about how to handle that change. We have three options:
        - We can overwrite or update the old value (Type I)
        - We can create a new record and create some mechanism for recreating historical references to that data – depending on the date for the report that is being requested (Type II).

    - We can retain both as alternative "realities". For the latter, we usually create a new column and put the old value in the new column to allow for alternatives to reporting.

- CRC-based record comparisons
  - A CRC-based comparison is a way of very quickly checking whether or not a change has occurred in a field. CRC stands for Cyclic Redundancy Check. The CRC is used to verify that no change has occurred in the data. Typically, a CRC is created as a mathematical calculation and stored as a large binary number. Because computers are good at this type of thing, the number can be divided by fixed number and that number is used to verify that nothing has changed (a checksum). If the checksum is different from the last time it was calculated, then we know the value has changed. CRC values change even if only one bit in the file changed, which makes it extremely reliable for checking integrity of files transmitted between computers.
  - SAS currently does not support a CRC algorithm for checking to see if two values are the same. But like with anything in SAS, we have our ways! A typical approach would be to just perform a lexical comparison (If "2345 Kildaire Farm Road" = "2345 Kildaire Farm Rd."). Of course in this example, it would flag a change since "Road" and "Rd." are different and we would have had to make design decisions about whether to use Type I, II or III logic in our loading process.
  - Regardless of what type of change is warranted, we have built a library of macros that handle Type I, II and III SCDs (slowly changing dimensions) and it's very easy to handle these things as changes occur in the data.

Quality Review, Auditing and Exception Handling

In software development terms, this heading could be called simply 'defensive programming'. The subsystems in Kimball's article that touched on these concepts are outlined below. The way we grouped these together essentially makes the entire ETL process self-regulating. If you're familiar with defensive programming concepts, these subsystems should be both clear and self-evident.

> **4. Data cleansing system.** Typically a dictionary driven system for complete parsing of names and addresses of individuals and organizations, possibly also products or locations. "De-duplication" including identification and removal usually of individuals and organizations, possibly products or locations. Often uses fuzzy logic. "Surviving" using specialized data merge logic that preserves specified fields from certain sources to be the final saved versions. Maintains back references (such as natural keys) to all participating original sources.
>
> **6. Audit dimension assembler.** Assembly of metadata context surrounding each fact table load in such a way that the metadata context can be attached to the fact table as a normal dimension.
>
> **7. Quality screen handler.** In line ETL tests applied systematically to all data flows checking for data quality issues. One of the feeds to the error event handler (see subsystem 8).
>
> **8. Error event handler.** Comprehensive system for reporting and responding to all ETL error events. Includes branching logic to handle various classes of errors, and includes real-time monitoring of ETL data quality

With Kimball's descriptions above, we think about these challenges in two ways: improving data quality and what do if it doesn't happen.

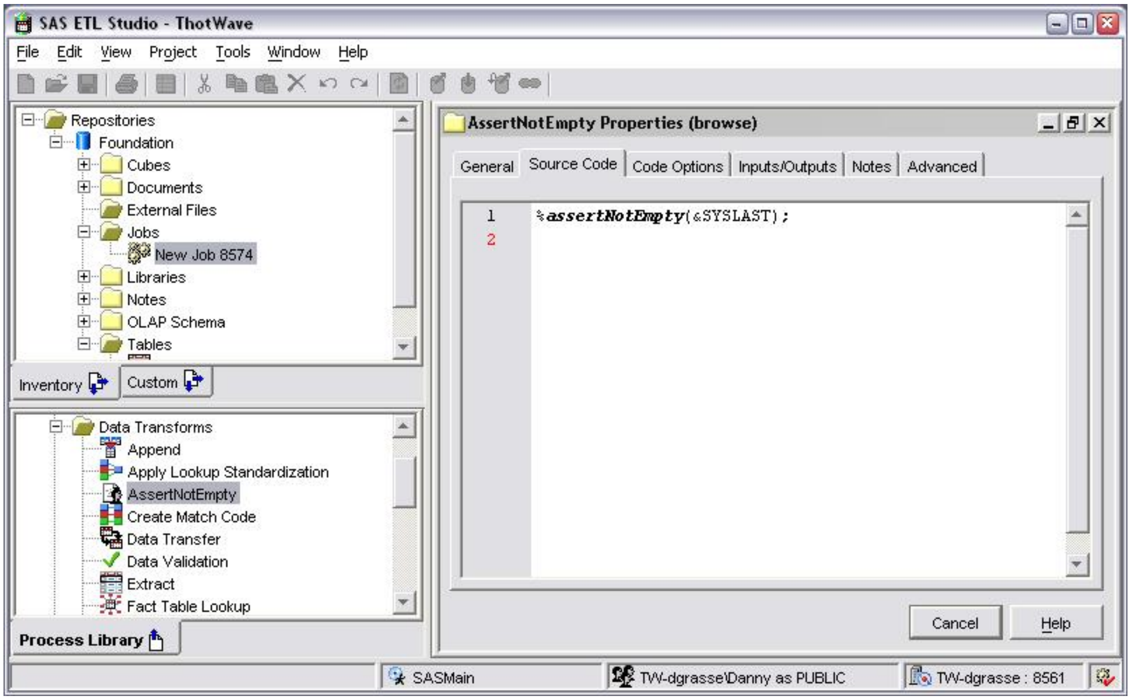
For the improvement part, we believe this is all about understanding risk. If we assume that our data is clean and don't do anything about it, what are the consequences if something goes wrong. Improving data quality is all about data profiling, data reviews and having a good process in place that ensures some level of daily inspection of the data that is being acquired. After all, our organizations have spent a lot of time and energy on having a good process in place.

The more SAS jobs, one or more programs, needed to complete the ETL process, the more opportunities for problems to arise. One good analogy given to complex, multi-stage systems is that of balancing a chain upside down. It seems at times a miracle if every link in the process actually works. Hence the absolute need to verify the state of the data as often as feasible. One example: If a required data set was not created, all subsequent DATA steps relying on it will fail. If the SAS program includes several DATA step that rely on that early, missing DATA step, the log will include multiple ERROR statements all cascading from its non-existence.

Without following the advice in these subsystems, the only way to discover the cause of the halted system (whenever someone discovers the process has indeed halted) is by scouring over SAS logs to find the root cause. The defensive way to work is to write a simple '%if %exists' bit of logic to assert the desired condition. If it fails, the subsystems above say to halt the ETL process and feed this information into the error event handler. This will report it appropriately and immediately. (Recovery from an aborted system is discussed in the next section.)

Assertion logic probably is handled most easily via utility macros that the ETL programs call. Data Integration Studio has a preset directory added as its macro autocall library within the SAS9 configuration directory structure (~/SASMain/SASEnvironment/SASMacro). Unless you modify the .cfg file used by the workspace server to suit your particular needs, dropping your macros in this directory is considered a best practice. A nice alternative is to turn your assertion macros into Data Integration Studio software transformations. By use of the transformation wizard, it's simple to create this transform as a simple pass-thru step addable to the job.

Here is a screen-shot of a simple assert as a custom transform. Notice the transform code is only the macro call. The macro itself would check for any observations in the table, and, if empty, would communicate this to the event system and, if appropriate, even abort the process. The macro variable &SYSLAST is provided by Data Integration Studio as the previous data set in the current job.



```
%assertNotEmpty(&SYSLAST);
```

Figure 4. Data Integration Studio: add SAS code to a custom transform.

And here is a sample flow that may use this assertion. Notice how the results of the 'SAS Extract' step from SampleSource is what is being checked, and if it succeeds, the data is loaded into the SampleTarget table.

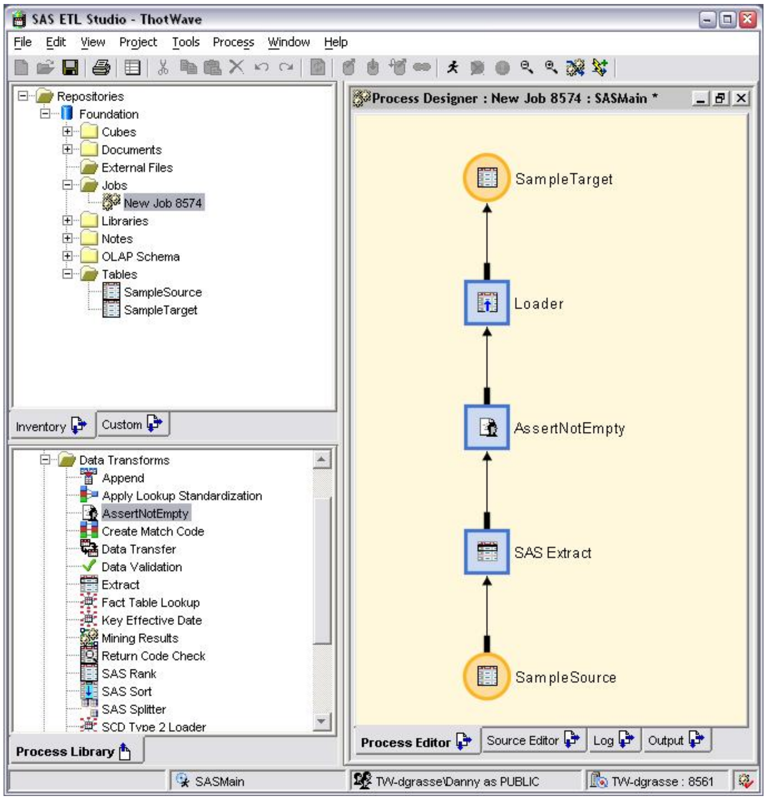
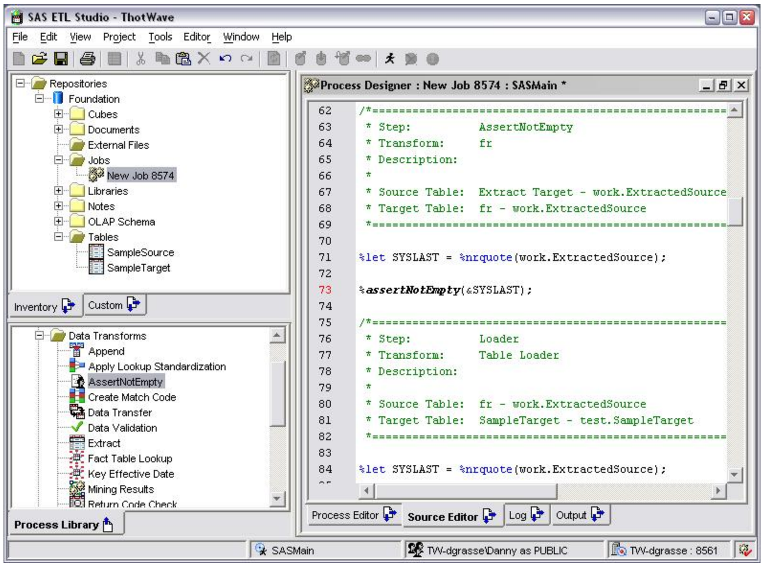


Figure 5. Data Integration Studio: add custom transform to flow.

So what does this code look like? Without showing all that's generated, here is the assert-related code. Notice how the input data set is simply passed through to the loader using the Data Integration Studio-provided macro variables &SYSLAST and &_OUTPUT.



```
/*=====================================================
 * Step:          AssertNotEmpty
 * Transform:     fr
 * Description:
 *
 * Source Table:  Extract Target - work.ExtractedSource
 * Target Table:  fr - work.ExtractedSource
 *=====================================================

%let SYSLAST = %nrquote(work.ExtractedSource);

%assertNotEmpty(&SYSLAST);

/*=====================================================
 * Step:          Loader
 * Transform:     Table Loader
 * Description:
 *
 * Source Table:  fr - work.ExtractedSource
 * Target Table:  SampleTarget - test.SampleTarget
 *=====================================================

%let SYSLAST = %nrquote(work.ExtractedSource);
```

Figure 6. Data Integration Studio: custom transform code snippet.

Of course all of the above could easily be accomplished with good old Base SAS using the same concepts (precondition checking, out of range value checking, statistical variances, etc.).

Integration with the Production Environment and Process Related Components

As we all learned long ago, it's not all about the technology. We cannot expect all of our ills to be fixed by a click of the mouse. That's where process is so important. Integrating products like Base SAS and Data Integration Studio into how you think about your ETL strategy is critical.

Kimball identified at least 13 subsystems that had to do with process and how we needed to update our processes to be consistent with the technology offerings. These included:

26. **Job scheduler.** System for scheduling and launching all ETL jobs. Able to wait for a wide variety of system conditions including dependencies of prior jobs completing successfully. Able to post alerts.

27. **Workflow monitor.** Dashboard and reporting system for all job runs initiated by the Job Scheduler. Includes number of records processed, summaries of errors, and actions taken.

28. **Recovery and restart system.** Common system for resuming a job that has halted, or for backing out a whole job and restarting. Significant dependency on backup system (see subsystem 36).

29. **Parallelizing/pipelining system.** Common system for taking advantage of multiple processors, or grid computing resources, and common system for implementing streaming data flows. Highly desirable (eventually necessary) that parallelizing and pipelining be invoked automatically for any ETL process that meets certain conditions, such as not writing to the disk or waiting on a condition in the middle of the process.

30. **Problem escalation system.** Automatic plus manual system for raising an error condition to the appropriate level for resolution and tracking. Includes simple error log entries, operator notification, supervisor notification, and system developer notification.

31. Version control system. Consistent "snapshotting" capability for archiving and recovering all the metadata in the ETL pipeline. Check-out and check-in of all ETL modules and jobs. Source comparison capability to reveal differences between different versions.

32. Version migration system. Development to test to production. Move a complete ETL pipeline implementation out of development, into test, and then into production. Interface to version control system to back out a migration. Single interface for setting connection information for entire version. Independence from database location for surrogate key generation.

33. Lineage and dependency analyzer. Display the ultimate physical sources and all subsequent transformations of any selected data element, chosen either from the middle of the ETL pipeline, or chosen on a final delivered report (lineage). Display all affected downstream data elements and final report fields affected by a potential change in any selected data element, chosen either in the middle of the ETL pipeline, or in an original source (dependency).

34. Compliance reporter. Comply with regulatory statutes to prove the lineage of key reported operating results. Prove that the data and the transformations haven't been changed. Show who has accessed or changed any such data.

35. Security system. Administer role-based security on all data and metadata in the ETL pipeline. Prove that a version of a module hasn't been changed. Show who has made changes.

36. Backup system. Backup data and metadata for recovery, restart, security, and compliance requirements.

37. Metadata repository manager. Comprehensive system for capturing and maintaining all ETL metadata, including all transformation logic. Includes process metadata, technical metadata, and business metadata.

38. Project management system. Comprehensive system for keeping track of all ETL development.

As we think through the implications of each of these, there are a few tidbits of relevance for our implementations in SAS.

- Version control, change control, promotion, backup and recovery.
- Scheduling, dependency management and restartability, including parallelization.
- Metadata management and impact analysis.
- Project management and problem escalation.

While all of these features are integral to how tools like Base SAS and Data Integration Studio live and operate in the IT environments of organizations it is instructive to note which of these are purely process and which can be aided by technology.

Version control, Change control, Promotion, Backup and Recovery

Just like the challenges that we as SAS programmers faced when building SAS/AF® application there is a slight disconnect when considering the full software development lifecycle. That is, we design, develop, code, test and promote. Rarely are all of these integrated seamlessly for the programmer. In the case of Base SAS all we really have is a development environment, DMS, Enterprise Guide, etc., with no hope of accomplishing any of the other tasks purely in SAS alone. For Data Integration Studio, we have seen a huge leap from do it all yourself, to a modest attempt to create a simplified promotion process for going from test to integration testing. Metadata migration from platform to platform is still a work in progress.

Data Integration Studio attempts to be a development tool in many ways. One process-related feature is its support of change management. This feature allows multiple developers to work on the same ETL process within the confines of the metadata context. It provides basic support for typical change-management behavior: checking in and out modules as well as tracking when, by whom, and attaching comments whenever a job or table is committed. These data are stored as metadata. And there is no 'diff-ing' mechanism. Specifically, you cannot view or rollback to a previous version of a job or table. Furthermore, any user-written code, since it's not metadata, is not managed within Data Integration Studio. External tools should be used to manage code, such as the open-source 'cvs' and the commercial 'ClearCase'. These tools are version control management systems, from which you can retrieve historic versions of files.

Scheduling, dependency management and restartability (including parallelization)

Unlike our experience with the maturity of some aspects of Data Integration Studio for integrating well with the entire software development life cycle, one area in which SAS excels is in scheduling. Instead of writing their own scheduler, SAS partners include a third-party scheduler called LSF Scheduler. This is a system for scheduling and launching all ETL jobs and can handle some fairly complex dependencies between jobs.

In the area of monitoring progress and reporting, through a dashboard or otherwise, Data Integration Studio is not quite as full featured. The product is able to post simple alerts when things, such as LOGs or simple e-mails, go awry. We would like to see a stack-trace included in that alerting feature so the administrator would have half a chance to know where to start looking – especially for >1000 lines of log that they have to parse. It is this premise along with the idea of having subscribable events, with attachments, that gave birth to ThotWave's *thinking data®* Event System – which is an add-on to SAS – that helps us accomplish these goals.

Related to scheduling is the idea of restartability. Being able to resume a job that has halted, or to back out of a whole job and restart it, is key to real-world data warehousing. The Event System also allows for this feature which is not available in Data Integration Studio by itself.

Another area in which SAS has excelled is in taking advantage of multiple processors and even grid-based environments. Their strategy rests on the SAS®9 platform which introduced grid computing and improved SMP processing capabilities. With MP CONNECT, an ETL process can be run so that the developer doesn't have to request the intermediate data sets to be written to disk. He also doesn't have to wait until another process is finished before moving on to the next step. With grid technology, these processes can occur by taking advantage of lots of computing resources throughout the organization – not just the big servers dedicated to these tasks.

Metadata management, Security and impact analysis

With regard to being able to manage information in the metadata, SAS has done a tremendous job in the SAS®9 platform to help us centrally locate information about data, the business uses of that data and process-related elements. The metadata server in SAS®9 is truly a comprehensive system for capturing and maintaining all ETL metadata, including all transformation logic and includes process metadata and technical metadata. However, not a lot of business and or application level metadata is included because for these, we still need reference tables.

One clear advantage of Data Integration Studio and the metadata server is we have a tremendous amount of metadata at our disposal about where the data came from, how it was transformed, and the processes that it underwent and when all of this happened. Kimball spoke of this when he identified some of capabilities to capture lineage information and what would happen if we were to change something (impact analysis). Base SAS does not have any of these features and they have to be built through custom coding.

Finally, as it relates to security, Base SAS relies on the host operating environment to enforce security. Data Integration Studio relies on metadata through the SAS® Management Console to enforce authorization rules through users, groups and roles. This affords the designer control of who has access to what throughout the entire information chain. The one weak area in all of this is the auditing feature of the SAS® Metadata Server. While it is possible to create audit records, it is neither helpful for human consumption nor is it without scalability issues.

PROJECT MANAGEMENT AND PROBLEM ESCALATION

In software development, we rely on good technology and good processes. In light of those two being present, we rely heavily on project management. A good project manager follows the principals set forth by the Project Management Institute in the nine knowledge areas of project management. These include managing all aspects of how a project should be run and how we deal with change. The nine knowledge areas are:

- Project Integration Management
- Project Scope Management
- Project Time Management
- Project Cost Management
- Project Quality Management
- Project Human Resource Management
- Project Communications Management
- Project Risk Management
- Project Procurement Management

Each knowledge area contains some or all of the project management processes that help us keep things in scope and on track. In his article, Kimball calls for a comprehensive system for keeping track of all ETL development plus a way for us to raise an error condition to the appropriate level for resolution and tracking. Both of these subsystems are purely process goals for ETL as implemented in SAS.

Putting it all Together

Throughout this paper we tried to identify with what Kimball was referring to as critical subsystems for a successful ETL strategy. At the end of the day, we need to think about the entire solution for our data integration approach and how this fits into the overall perspective of getting useful, clean data into the hands of people who can make decisions.

As we saw, in order to build a world-class data warehouse, we need to accomplish at least 38 “things.” Kimball identified these and we tried to map them into what SAS is really good at. The bottom line for us is that Data Integration Studio should be part of your ETL strategy. It gives us the power of the whole of SAS – including Base SAS. It extends those capabilities beginning with metadata and security and lets us focus on more of what is important, such as defining the relationship between source and target tables and less on the cumbersome aspects of promotion, scheduling and maintaining our metadata.

When Data Integration Studio is not enough, the developer can call out to the rest of SAS for introducing custom code.

Approaches for Solving the Rest of the Challenges

Of course, one of the things that makes SAS a world-class environment is the fact that it affords us the ability to extend and enhance the system through user exits, custom transformation, plug-ins to the SAS® Management Console and lower-level integration through the published APIs. We've given a few examples on extensions that we made when we referenced the *thinking data*® Event System. We've create a SAS software-based API that allows users to create their own transformations. It also allows for complex modeling of how a SAS job should be run (parallel and sequential as well as dependency management) as well as a facility for subscription-based events that can be triggered from both good events, such as job completion and information and bad events, such as errors, warnings and preconditions not being met. SAS doesn't do everything, but we have found that it can do just about anything we want through core features and the ability to extend the tools through integration.

References:

Inmon, W. H., Claudia Imhoff and Ryan Sousa. The Corporate Information Factory. Wiley. New York. 2nd edition. 2002.

Inmon, W.H., Building the Data Warehouse. QED/Wiley, 1991

Imhoff, Claudia. 2001. "Intelligent Solutions: Oper Marts – An Evolution in the Operational Data Store." DM Review. Brookfield. 2001.

Kimball, Ralph. The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. John Wiley & Sons, 1996

Kimball, Ralph. "The 38 Subsystems of ETL: To create a successful data warehouse, rely on best practices, not intuition." Intelligent Enterprise." December 4, 2004.

Kimball, Ralph and Conserta, Joe. The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. John Wiley & Sons, 2004

Kimball, Ralph, Laura Reeves, Margy Ross, and Warren Thornthwaite. The Data Warehouse Lifecycle Toolkit: Tools and Techniques for Designing, Developing, and Deploying Data Warehouses John Wiley & Sons, 1998

Kimball, Ralph and Ross, Margy. The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (Second Edition). John Wiley & Sons, 2002

Biography:

Danny Grasse Sr. Consultant

Danny is a senior consultant who specializes in business application development. His origins are in C/C++ and Java development, and he has extensive experience with database modeling and Web-interface construction. He has worked with Data Integration Studio for almost two years in both development and instructional capacities. Danny has been with ThotWave for 5 years.

Greg Nelson, President and CEO

Greg has just celebrated his 20th year in the SAS eco-system. He started out as a social psychology student doing statistical analysis then quickly moved into applications development. Greg is the president and CEO of ThotWave Technologies where he supports an entire organization focused on helping customers leverage their investment in SAS. Prior to ThotWave, Greg spent several years in consulting, media and marketing research, database marketing and large systems support. He holds a bachelor's degree in psychology and Ph.D. level work in social psychology and quantitative methods.

About ThotWave

ThotWave Technologies, LLC, is a Cary, N.C.-based consultancy and market leader in real-time decision support, specializing in regulated industries such as life sciences, energy and financial services. ThotWave recognizes the difference between simply accessing data and making data work for business. ThotWave commands the juncture of business and technology to help companies improve their operational and strategic performance. Through products, partnerships and services, ThotWave enables businesses to leverage data for faster, more intelligent decision making.

Contact information:

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

Danny Grasse	dgrasse@thotwave.com
Greg Nelson	greg@thotwave.com

ThotWave Technologies, LLC

2504 Kildaire Farm Road

Cary, NC 27511

(800) 584 2819

http://www.thotwave.com

Appendix A. Kimball's 38 Sub-systems of ETL

The 38 Subsystems

1. Extract system. Source data adapters, push/pull/dribble job schedulers, filtering and sorting at the source, proprietary data format conversions, and data staging after transfer to ETL environment.

2. Change data capture system. Source log file readers, source date and sequence number filters, and CRC-based record comparison in ETL system.

3. Data profiling system. Column property analysis including discovery of inferred domains, and structure analysis including candidate foreign key — primary relationships, data rule analysis, and value rule analysis.

4. Data cleansing system. Typically a dictionary driven system for complete parsing of names and addresses of individuals and organizations, possibly also products or locations. "De-duplication" including identification and removal usually of individuals and organizations, possibly products or locations. Often uses fuzzy logic. "Surviving" using specialized data merge logic that preserves specified fields from certain sources to be the final saved versions. Maintains back references (such as natural keys) to all participating original sources.

5. Data conformer. Identification and enforcement of special conformed dimension attributes and conformed fact table measures as the basis for data integration across multiple data sources.

6. Audit dimension assembler. Assembly of metadata context surrounding each fact table load in such a way that the metadata context can be attached to the fact table as a normal dimension.

7. Quality screen handler. In line ETL tests applied systematically to all data flows checking for data quality issues. One of the feeds to the error event handler (see subsystem 8).

8. Error event handler. Comprehensive system for reporting and responding to all ETL error events. Includes branching logic to handle various classes of errors, and includes real-time monitoring of ETL data quality

9. Surrogate key creation system. Robust mechanism for producing stream of surrogate keys, independently for every dimension. Independent of database instance, able to serve distributed clients.

10. Slowly Changing Dimension (SCD) processor. Transformation logic for handling three types of time variance possible for a dimension attribute: Type 1 (overwrite), Type 2 (create new record), and Type 3 (create new field).

11. Late arriving dimension handler. Insertion and update logic for dimension changes that have been delayed in arriving at the data warehouse.

12. Fixed hierarchy dimension builder. Data validity checking and maintenance system for all forms of many-to-one hierarchies in a dimension.

13. Variable hierarchy dimension builder. Data validity checking and maintenance system for all forms of ragged hierarchies of indeterminate depth, such as organization charts, and parts explosions.

14. Multivalued dimension bridge table builder. Creation and maintenance of associative (bridge) table used to describe a many-to-many relationship between dimensions. May include weighting factors used for allocations and situational role descriptions.

15. Junk dimension builder. Creation and maintenance of dimensions consisting of miscellaneous low cardinality flags and indicators found in most production data sources.

16. Transaction grain fact table loader. System for updating transaction grain fact tables including manipulation of indexes and partitions. Normally append mode for most recent data. Uses surrogate key pipeline (see subsystem 19).

17. Periodic snapshot grain fact table loader. System for updating periodic snapshot grain fact tables including manipulation of indexes and partitions. Includes frequent overwrite strategy for incremental update of current period facts. Uses surrogate key pipeline (see subsystem 19).

18. Accumulating snapshot grain fact table loader. System for updating accumulating snapshot grain fact tables including manipulation of indexes and partitions, and updates to both dimension foreign keys and accumulating measures. Uses surrogate key pipeline (see subsystem 19).

19. Surrogate key pipeline. Pipelined, multithreaded process for replacing natural keys of incoming data with data warehouse surrogate keys.

20. Late arriving fact handler. Insertion and update logic for fact records that have been delayed in arriving at the data warehouse.

21. Aggregate builder. Creation and maintenance of physical database structures, known as aggregates, that are used in conjunction with a query-rewrite facility, to improve query performance. Includes stand-alone aggregate tables and materialized views.

22. Multidimensional cube builder. Creation and maintenance of star schema foundation for loading multidimensional (OLAP) cubes, including special preparation of dimension hierarchies as dictated by the specific cube technology.

23. Real-time partition builder. Special logic for each of the three fact table types (see subsystems 16, 17, and 18) that maintains a "hot partition" in memory containing only the data that has arrived since the last update of the static data warehouse tables.

24. Dimension manager system. Administration system for the "dimension manager" who replicates conformed dimensions from a centralized location to fact table providers. Paired with subsystem 25.

25. Fact table provider system. Administration system for the "fact table provider" who receives conformed dimensions sent by the dimension manager. Includes local key substitution, dimension version checking, and aggregate table change management.

26. Job scheduler. System for scheduling and launching all ETL jobs. Able to wait for a wide variety of system conditions including dependencies of prior jobs completing successfully. Able to post alerts.

27. Workflow monitor. Dashboard and reporting system for all job runs initiated by the Job Scheduler. Includes number of records processed, summaries of errors, and actions taken.

28. Recovery and restart system. Common system for resuming a job that has halted, or for backing out a whole job and restarting. Significant dependency on backup system (see subsystem 36).

29. Parallelizing/pipelining system. Common system for taking advantage of multiple processors, or grid computing resources, and common system for implementing streaming data flows. Highly desirable (eventually necessary) that parallelizing and pipelining be invoked automatically for any ETL process that meets certain conditions, such as not writing to the disk or waiting on a condition in the middle of the process.

30. Problem escalation system. Automatic plus manual system for raising an error condition to the appropriate level for resolution and tracking. Includes simple error log entries, operator notification, supervisor notification, and system developer notification.

31. Version control system. Consistent "snapshotting" capability for archiving and recovering all the metadata in the ETL pipeline. Check-out and check-in of all ETL modules and jobs. Source comparison capability to reveal differences between different versions.

32. Version migration system. Development to test to production. Move a complete ETL pipeline implementation out of development, into test, and then into production. Interface to version control system to back out a migration. Single interface for setting connection information for entire version. Independence from database location for surrogate key generation.

33. Lineage and dependency analyzer. Display the ultimate physical sources and all subsequent transformations of any selected data element, chosen either from the middle of the ETL pipeline, or chosen on a final delivered report (lineage). Display all affected downstream data elements and final report fields affected by a potential change in any selected data element, chosen either in the middle of the ETL pipeline, or in an original source (dependency).

34. Compliance reporter. Comply with regulatory statutes to prove the lineage of key reported operating results. Prove that the data and the transformations haven't been changed. Show who has accessed or changed any such data.

35. Security system. Administer role-based security on all data and metadata in the ETL pipeline. Prove that a version of a module hasn't been changed. Show who has made changes.

36. Backup system. Backup data and metadata for recovery, restart, security, and compliance requirements.

37. Metadata repository manager. Comprehensive system for capturing and maintaining all ETL metadata, including all transformation logic. Includes process metadata, technical metadata, and business metadata.

38. Project management system. Comprehensive system for keeping track of all ETL development.