Sunopsis

WHITE PAPER

Is ETL Becoming Obsolete?

Why a Business-Rules-Driven "E-LT" Architecture is Better

Introduction

In today's fast-paced, information-based economy, companies must be able to integrate vast amounts of heterogeneous data from disparate sources in order to support strategic IT initiatives such as Business Intelligence or Corporate Performance Management, including Master Data Management, Data Warehousing and Data Marts. At the same time, IT organizations are under constant pressure to get more done with fewer resources. The only way to satisfy these conflicting goals is to adopt a cost-effective integration solution that enhances the productivity of the IT organization and helps to streamline a broad range of integration initiatives.

Over the past several years many companies have turned to commercial ETL (Extract, Transform, Load) tools as a means to reduce the effort associated with the most common integration approach: manual coding. Using a centralized ETL "engine" as an integration hub, and powered by a single, platform-independent language, most ETL tools do reduce the effort associated with point-to-point manual integration. However, many of the users of these tools have also encountered a number of issues that are a consequence of the traditional ETL architecture. This begs the question: *is the conventional approach to ETL obsolete?*

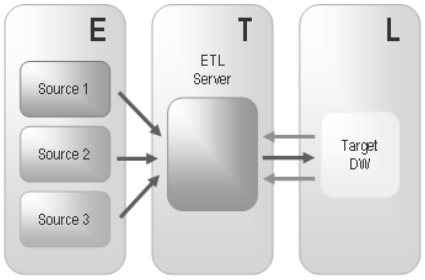

Figure 1: The traditional ETL approach

The Problems with the Traditional ETL Approach

Traditional ETL tools operate by first Extracting the data from various sources, Transforming the data on a proprietary, middle-tier ETL engine, and then Loading the transformed data onto the target data warehouse or integration server. Hence the term "ETL" represents both the names and the order of the operations performed, as shown in Figure 1 above. Most ETL tools use a graphical programming paradigm to shield the user from the complexity of coding the transformations and make the tools easier to learn and use. While in theory ETL tools should improve developer productivity, this is not always the

case. The primary issues with the traditional ETL approach fall into the following three categories: productivity/maintainability, performance and cost.

Productivity/Maintainability

Virtually all ETL tools use some kind of graphical programming paradigm as an alternative to manual coding. While at first glance many of these data-flow oriented GUIs look similar, there are significant differences that impact the number of intermediate steps that must be defined, and the maintainability and reuse of the job and its component pieces over time.

The conventional ETL approach first requires the user to describe what he/she wants to do in plain English. That means describing the business rules as text and then detailing precisely how to implement these rules, step by step. This phase of development is often referred to as designing the data flow. It requires a strong understanding of the structure of the data and the architecture of the IT system. As illustrated in Figure 2 below, the data flow needs to be defined manually, and repeated for each individual process. Users must understand not only what the overall transformation is supposed to do, but also define what each incremental step is needed to perform it. This usually ends up as a long chain of multiple mapping operations tied to a number of temporary outputs. Defining all of these intermediate stages requires additional analysis and development work, and hence, has a negative impact on productivity.

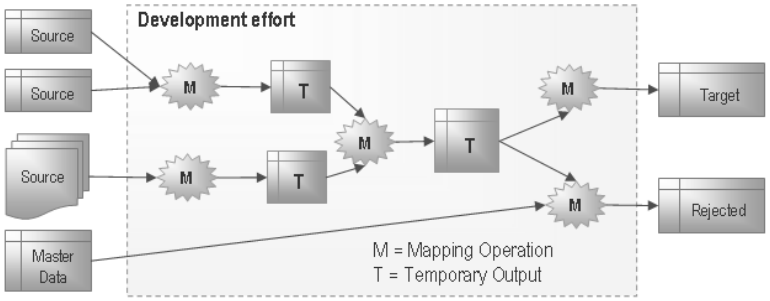

Figure 2: Fragmented data flow using traditional ETL tools requires more effort

What's worse, when there is a need to change a business rule or accommodate additional data sources or targets, significant rework may be required to the existing data flow since the transformations are highly fragmented, requiring edits to many sub-blocks. This can make maintenance even more challenging – especially when the person maintaining the processes is not the same as the person who wrote them.

Performance Issues with Traditional ETL

The data transformation step of the ETL process is by far the most compute-intensive, and is performed entirely by the proprietary ETL engine on a dedicated

server. The ETL engine performs data transformations (and sometimes data quality checks) on a row-by-row basis, and hence, can easily become the bottleneck in the overall process. In addition, the data must be moved over the network twice – once between the sources and the ETL server, and again between the ETL server and the target data warehouse. Moreover, if one wants to ensure referential integrity by comparing data flow references against values from the target data warehouse, the referenced data must be downloaded from the target to the engine, thus further increasing network traffic, download time, and leading to additional performance issues.

Let's consider, for example, how a traditional ETL job would look up values from the target database to enrich data coming from source systems. To perform such a job, a traditional ETL tool could be used in one of the following three ways:

- Load look-up tables into memory: The entire look-up table is retrieved from the target server and loaded into the engine's memory. Matching (or joining) this look-up data with source records is done in memory before the resulting transformed data is written back to the target server. If the look-up table is large, the operation will require a large amount of memory and a long time to download its data and re-index it in the engine.

- Perform row-by-row look-ups "on the fly": For every row, the ETL engine sends a query to the look-up table located on the target server. The query returns a single row that is matched (or joined) to the current row of the flow. If the look-up table contains, for example, 500,000 rows, the ETL engine will send 500,000 queries. This will dramatically slow down the data integration process and add significant overhead to your target system.

- Use manual coding within the ETL job: Use the ETL engine only for loading source data to the target RDBMS and manually write SQL code to join this data to the target look-up table. This raises the question: why would you buy an ETL tool that requires manual coding on the target server, knowing that you lose all the benefits of metadata management and development productivity by doing so? Unfortunately, this is what many users end up doing once they notice a 10x degradation in the overall performance of the integration process (when compared to the same operations executed by manual code).

Cost Issues

Most ETL tool purchases are justified based on potential labor savings. Unfortunately, there are other up-front and recurring costs that must be considered in the ROI analysis. The most obvious initial cost is that of the dedicated server and proprietary ETL engine software. Because these middle-tier components carry out all the compute-intensive transformation operations, a powerful server is required, and in some cases multiple servers and run-time engines are necessary to meet the throughput requirements. There are also

ongoing hardware and software maintenance costs associated with these assets. This can result in hundreds of thousands of dollars in additional hardware, software and maintenance expenses. In addition, as the Data Warehouse grows to accommodate higher throughput demands, the ETL hub server will need to scale up with it, necessitating additional hardware and software purchases in the future.

Conventional ETL tools also have a number of hidden costs, including the consulting expenses required for setup and tuning, and the rip-up and re-write of code as integration requirements evolve over time.

A Better Approach: "E-LT" Architecture + Business Rules

Introduction to the E-LT Architecture

In response to the issues described above, a new architecture has emerged, which in many ways incorporates the best aspects of both manual coding and ETL approaches in the same solution. Known as "E-LT", this new approach changes where and how data transformation takes place, and leverages the existing developer skills, RDBMS engines and server hardware to the greatest extent possible. In essence, E-LT moves the data transformation step to the target RDBMS, changing the order of operations to: Extract the data from the source tables, Load the tables into the destination server, and then Transform the data on the target RDBMS using native SQL operators. Note, with E-LT there is no need for a middle-tier engine or server as shown in figure 3 below.

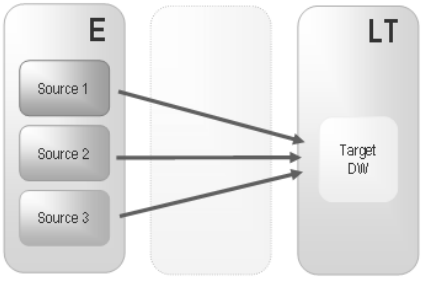

Figure 3: The "E-LT" approach requires no middle-tier engine or dedicated ETL server

Why the ETL Market is Changing

When commercial ETL tools first appeared in the 1990's the most widely used RDBMSs such as Oracle, DB2, Teradata and Sybase did not support a rich enough set of SQL operators to handle the complex data transformation tasks required for data warehouse applications. Hence the dedicated ETL engine and

proprietary transformation language emerged as the best alternative to laborious manual coding at the time.

However, over the past decade the RDBMS vendors have increased the functionality of the SQL provided to programmers by an order of magnitude, while improving the performance and reliability of their engines at the same time. For example, the CASE…WHEN statement (equivalent to an IF…THEN…ELSE…) can be used for complex transformation rules. Outer joins (LEFT OUTER, RIGHT OUTER, or FULL JOIN) can be used to easily join data sets in a variety of different manners. Ranking and windowing functions (MIN OVER PARTITION, MAX OVER PARTITION, LEAD, LAG, and RANK) allow for more effective handling of complex aggregations of large volumes of data.

Complementing the richer language support, RDBMS vendors now provide a long list of out-of-the-box features and utilities that enable impressive performance when executing ETL-type operations. Some features are dedicated to efficient loading data from sources to targets, while others directly process various data formats such as XML files. These are just a few examples of what can be done with the native SQL solutions provided with RDBMS packages today.

E-LT Architecture Offers Better Performance with Bulk Processing

By generating efficient, native SQL code, the E-LT approach leverages the powerful bulk data transformation capabilities of the RDBMS, and the power of the server that hosts it. In addition, since the data is loaded directly from source systems to the target server, only one set of network transfers are required, instead of two or more as with the traditional ETL approach. Only relational DBMS engines can perform set operations and bulk data transformations, enabling processes to achieve the highest performance. Inserts and updates are handled as bulk operations and no longer performed row-by-row. Thanks to "set-processing" logic, the E-LT approach can achieve exceptional performance with data transformations up to ten to twenty times more efficient than traditional ETL tools.

Business-Rules-Driven Approach Brings Better Productivity and Maintainability

With a business-rules-driven paradigm, the developer only defines what he/she wants to do, and the data integration tool automatically generates the data flow, including whatever intermediate steps are required, based on a library of "knowledge modules". The "what to do", i.e. the business rules, are specified using expressions that would make sense to business analysts, and are stored in a central metadata repository where they can be easily reused. The implementation details, specifying "how to do it", are stored in a separate knowledge module library, and can be shared between multiple business rules within multiple ETL processes. The key advantage of this approach is that it is very easy to make

incremental changes either to the rules or to the implementation details, as they are, in essence, independent. When a change needs to be applied to operations logic (e.g. creating a backup copy of every target table before loading the new records) it is simply implemented in the appropriate knowledge module. That change is then automatically reflected in the hundreds of processes that reference it, without having to touch the business rules. With a traditional ETL approach, such a change would require opening every process to manually add new steps, increasing the risks of errors and inconsistencies. This makes a huge difference in developer productivity, especially in long-term program maintenance.

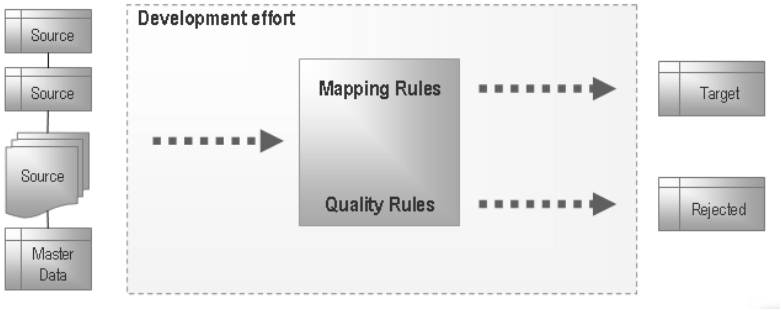

Figure 4: End-to-end business rules eliminate the need to define intermediate steps and improve overall productivity and maintainability

Combine E-LT with Business Rules to Lower the Total Cost of Ownership

Last but not least, E-LT has proven to be the most cost effective approach to data integration. With no middle-tier ETL engine and no dedicated ETL server hardware required, the initial hardware and software capital costs are significantly lower, as are the ongoing software and hardware maintenance expenses. E-LT software also tends to be less expensive because it does not require the development of a proprietary engine for transformations. It uses any standard RDBMS to execute the ETL jobs. This means significant savings for both the software vendor and the customer. These savings are on top of the developer productivity improvement, which enables IT organizations to dramatically reduce the cost of developing and maintaining comprehensive Data Warehouses.

Conclusion: Traditional ETL is Indeed Becoming Obsolete

Now that we've compared the traditional ETL approach to the newer Business-rules-driven E-LT paradigm the answer to our original question is clear: conventional ETL tools are indeed becoming obsolete, and tools based on business rules and E-LT are beginning to take their place. Widely used for data integration projects of all kinds, Sunopsis Data Conductor exemplifies the E-LT

architecture described in this paper. Data Conductor not only offers the highest level of performance possible for data transformation and validation, but also utilizes a unique business-rules-driven approach to defining integration processes, resulting in a dramatic reduction in the development and maintenance workload. And finally, by eliminating the need for dedicated ETL servers and proprietary engines Data Conductor offers the lowest total cost of ownership and an ROI realized during the very first project. For more information on Data Conductor and the other Sunopsis integration products please visit www.sunopsis.com.

Boston, USA
+1 781 238 1770
1 888 740 0300
info-us@sunopsis.com

Bonn, Germany
+49 (0) 170 296 4124
info-de@sunopsis.com

London, UK
+44 (0) 870 351 4408
info-uk@sunopsis.com

Lyon, France
+33 (0)4 78 33 92 20
info-fr@sunopsis.com

Milan, Italy
+39 02 89307064
info-it@sunopsis.com

Paris, France
+33 (0)1 40 21 42 21
info-fr@sunopsis.com

Rome, Italy
+39 06 5163139
info-it@sunopsis.com

Singapore
+65 6559 6139
info-apac@sunopsis.com

WP-IsETLObsolete -0206-2.2