

ETL provides the keys to keeping data relevant

Author: BALWANT RAI

Organization: Evaltech, Inc.

**Evaltech Research Group,
Data Warehousing Practice.**

Date: 05/01/04

Email: erg@evaltech.com



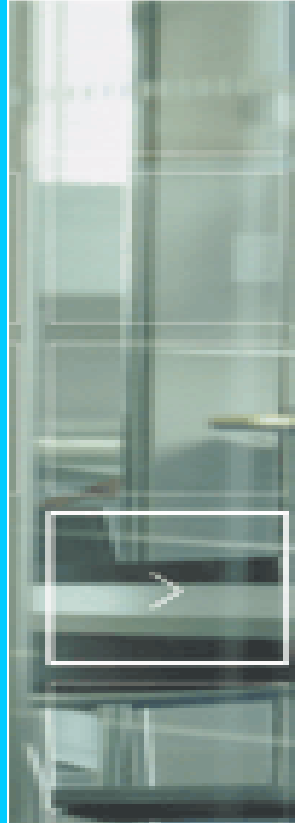
Abstract:

Three key processes commonly used to move data from one application or system to another are known as ETL, for extraction, transformation, and loading.

The goal of virtually every ETL application is to make data available to its audience in a timely fashion. Traditionally, businesses have relied on ETL routines to transfer data from old applications to new ones, or to move operational data into business intelligence systems, such as data warehouses and data marts.

Intellectual Property / Copyright Material

All text and graphics found in this article are the property of the Evaltech, Inc. and cannot be used or duplicated without the express written permission of the corporation through the Office of Evaltech, Inc.



ETL provides the keys to keeping data relevant

Managing Big Data

Can you, at this moment in time, imagine managing 500 terabytes of data? Or integrating billions of clicks from your web site with data from multiple channels and business units every day, may be more than once a day? It seems like an extreme scenario, but it's one that industry analysts uniformly predict organizations will be confronting within the next three to four years. The exact figures may vary slightly, but the consensus is solid: enterprises are going to be swamped with data, and they're going to have to figure out how to manage it or risk being left behind.

A rapid increase in data volume is not the only challenge enterprises will face. End users want more information at more granular levels of detail, and they want flexible, integrated, timely access. The number of users and the number of queries are also growing dramatically. Additionally, organizations are placing more emphasis on high-value, data-intensive applications such as CRM. All of these developments pose problems for enterprise data management. Fortunately, there is an effective answer to these problems—scalable data solutions, and more specifically, scalable ETL environments.

Scalability is defined as the retention or improvement of an application's performance, availability, and maintainability with increasing data volumes. This paper will explore the three dimensions of scalability as they relate to ETL environments, and will suggest some techniques that IT organizations can use to ensure scalability in their own systems.

In the case of performance, scalability implies the ability to increase performance practically without limit. Performance scalability as a concept is not new, but actually achieving it is becoming much more challenging because of the dramatic increases in data volume and complexity that enterprises are experiencing on every front. How long will users tolerate growing latency in the reporting provided to them? How long can enterprises keep adding hardware, installing new software, tweaking or building from scratch new applications? The fact is, yesterday's scalable solutions aren't working in the new environment.

The extract, transform and load (ETL) environment poses some especially difficult scalability challenges because it is constantly changing to meet new requirements. Enterprises must tackle the scalability problem in their ETL environments in order to successfully confront increasing refresh frequencies, shrinking batch windows, increasing processing complexity, and growing data volumes. Without scalability in the ETL environment, scalability in the hardware or database layer becomes less effective. In terms of implementing scalable data solutions, enterprises should adopt a "build it once and build it right" attitude. If an ETL environment is designed to be scalable from the start, the organization can avoid headaches later. Let's consider a situation in which this is not the case, and the ETL environment is architected without consideration for scalability. The first generation of this solution will be fine until data volumes exceed capacity. At that point, the organization will be able to make the fairly easy move to a second-generation environment by upgrading hardware and purchasing additional software licenses. Once this solution is no longer sufficient, however, the enterprise will find it more difficult and costly to evolve to a third-generation solution, which usually involves custom programming and buying point solutions. Finally, once the third-generation solution has reached its limits, the enterprise will need to rebuild its ETL environment entirely, this time using scalable and parallel technologies. Clearly, enterprises can save time and money by implementing a scalable ETL environment from the very beginning.

Effective data extract, transform and load (ETL) processes represent the number one success factor for your data warehouse project and can absorb up to 70 percent of the time spent on a typical warehousing project. ETL tools promise quick results, improved manageability and meta data integration with other common design and implementation tools. However, due to the potentially huge amounts of money involved in a tool decision, choosing the correct ETL tool for your project can present a daunting challenge. With a bit of internal questioning in advance

followed by a careful review of your key needs against the choices available on the market, you should be able to choose the most effective ETL tool for your project.

ETL tools perform, as you may guess, at least three specific functions all of which focus around the movement of data from one place (file type, server, location, etc.) or system to another. More encompassing than a simple file copy process, this class of software generally reads data from an input source (flat file, relational table, message queue, etc.); passes the stream of information through either an engine- or code-based process to modify, enhance, or eliminate data elements based on the instructions of the job; and then writes the resultant data set back out to a flat file, relational table, etc. As you may have guessed, these three steps are known as extraction, transformation and loading, respectively.

ETL Tools – These "complete" ETL tools provide a rich mix of functionality and connectivity, but may be significantly more expensive than tools found in the other categories. For extremely complex projects or those attempting to process massive amounts of data, these tools may present the only true option for ensuring success in the ETL phase of the project. In other cases, this class of tool may offer features that simply are not required in the existing environment.

Three key processes commonly used to move data from one application or system to another are known as ETL, for extraction, transformation, and loading.

The goal of virtually every ETL application is to make data available to its audience in a timely fashion. Traditionally, businesses have relied on ETL routines to transfer data from old applications to new ones, or to move operational data into business intelligence systems, such as data warehouses and data marts.

Due to the explosive growth of the Internet, however, ETL processes are now commonly used to support Web applications as well. For example, a manufacturer might use ETL routines to load a Web-based order-status system with production data from an internal legacy application. Similarly, a retailer might use ETL routines to relay order data from its online storefronts to its suppliers. ETL programs have become essential components of many e-commerce initiatives, including business-to-business and business-to-consumer applications.

In broad terms, ETL applications extract data from a source database, transform the data into a format suitable for a target database, and then load the data into the target database. In this analysis, the InfoWorld Test Center provides an overview of ETL processing.

Data extraction

To initiate an ETL process, programmers use extraction routines to read records in a source database and make the data in those records available for transformation processing. To extract data from a source database, programmers have three choices: They can write customized programs, rely on specialized ETL tools, or use a combination of both.

In practice, most programmers bolster third-party tools with customized programs. Referred to as user exits, these programs perform specialized functions that are unique to each environment.

Third-party ETL products are typically more effective and less costly than customized programs. Many ETL tools provide programmers with a single, intuitive interface that can be used to extract data from multiple database products. Businesses that rely on a hodgepodge of databases will benefit from unified access to those products.

ETL provides the keys to keeping data relevant

ETL tools are also preferable to custom programs because they can be used "out of the box"; there is no need to write code to open files, read records, and join tables -- these products perform those functions for you. Furthermore, leading ETL products include prebuilt extraction routines designed for popular ERP (enterprise resource planning) applications. Finally, as their acronym implies, ETL tools support not only extraction but also transformation and loading. In many cases, a single ETL tool can meet most of your data-movement requirements.

Data transformation

Once extraction routines have collected the data, transformation routines can prepare that data for its new home. There are several major transformation techniques, including aggregation, value translation, field derivation, and cleansing. Let's use a hypothetical data mart to examine these processes. Note, however, that the use of these routines is not limited to data marts.

Data marts are specialized applications that allow end-users to analyze broad trends in a highly intuitive manner. For instance, a marketing analyst might use a sales data mart to examine revenue per product for each of the past five years. Unlike order processing, manufacturing, and other operational applications, data marts do not require detailed information. In fact, summarized data is preferable because it reduces response times and enhances ease of use.

Before loading a data mart, programmers typically aggregate data. Aggregation routines replace numerous detail records with relatively few summary records. For example, suppose that a year's worth of sales data is stored in several thousand records in a normalized database.

Through aggregation, this data is transformed into fewer summary records that will be written to the sales data mart. Although programmers could write code to manually aggregate data, ETL tools are more efficient because they allow programmers to summarize data in one step, with no coding.

Value translation is another common data-transformation technique. Operational databases store encoded information to limit data redundancy and storage requirements. For example, SKUs (stock-keeping units) may be stored in invoice files because they are shorter than their associated product descriptions, and so on.

Because data marts contain summarized information and are designed for ease of use, programmers typically replace encoded data with clearer descriptions. Although programmers could accomplish this task with custom code, ETL tools are more efficient because they allow programmers to use value-translation lists to decode data.

Field derivation is a third technique used to transform data. Through field derivation, new information is created for end-users. For example, suppose our operational database contains one field for sales quantity and one for unit price. Rather than have end-users calculate revenue, programmers could create a revenue field during transformation. Leading ETL products enable programmers to use mathematical operations, statistical functions, string manipulation, date arithmetic, and conditional logic to derive new fields.

A fourth transformation routine, cleansing, has many uses. Programmers rely on cleansing algorithms to keep inaccurate data out of other systems. For example, cleansing routines typically verify that numeric fields contain numeric data, that dates and numbers are valid and reasonable, and so on. Cleansing routines can also be used in cases when one unique value is represented by a database in several ways. For example, IBM might be depicted as IBM Co., International Business Machines, IBM, etc. During cleansing, multiple versions of the same data element are replaced with a single value. Cleansing can also be used to adjust field attributes so they match those of the target database.

Data loading

After data has been transformed for the target database, programmers use load procedures to write that information to the new database. During this phase, you must determine whether to propagate data periodically or continuously. Periodic replenishment occurs regularly, such as daily, weekly, or monthly. Sometimes known as snapshot propagation, this approach captures the state of operational systems at a specific moment. If users require current information, continuous propagation can load data into the target database on a real-time basis. Continuous replication requires a dedicated, high-speed communications channel. Your affinity for periodic vs. continuous replenishment will depend largely on your users' needs for up-to-date information, as well as your system infrastructure and budget.

Most ETL tools support both periodic and continuous data loads. Advanced products also let you perform propagation on a net-change basis, which allows you to transmit only modified data, and consequently requires minimal communications overhead.

You can further categorize data loading by the method used to replicate data. In push replication, the source application "pushes" transformed data to the target application. In pull replication, the target application "pulls" data as it's needed, such as when an end-user runs a query. You can also design load procedures that use both push and pull processes. In this approach, the source application typically pushes the data to a staging database, where it is transformed and then pulled into the target application as needed. This "mixed-mode" approach requires more disk space, but can enhance performance.

Whether you use custom coding, third-party ETL tools, or a combination of the two to move data from one system to another, a thorough understanding of ETL practices, as well as of your source and target applications, is essential to success. Given the importance and price of many ETL solutions, you should also insist on a trial period so that you may test each solution in your environment.