**ETL (Extraction, Transformtion and Loading)**

**Introduction**

Many of the Business Intelligence tools look way cool. They provide graphs, moving targets, drill-downs, and drill-through. But much of the work in an operational data warehouse involves getting the data from operational systems into the data warehouse so that business intelligence tools can display those pretty pictures. This paper addresses the extraction, transformation, and load components of data warehousing. We’ll look at issues in extraction, transformation, and loading and common approaches to loading data. We assume that source data structures are generally not similar to target data structures (e.g., flat files and normalized tables).

**The Plan**

At the risk of being a bit simplistic, extraction, transformation, and load requires three main steps:

* Read the source data
* Apply business, transformation, and technical rules
* Load the data

Figure 1 shows this data flow. After the process reads the data, it must transform the data by applying technology, transformation, and business rules to it. An example of a transformation rule is: “Convert values in field x to integer.” An example of a business rule is: “Customers must purchase products in the list of ‘Washer’, ‘Dryer’, “Refrigerator’.”  
Applying business, transformation, and technology rules to data means generating keys, transforming codes, converting datatypes, parsing, merging, and many other operations. Once the data is in an appropriate technical and business format, the ETL process can load it into target tables. Note that these steps can potentially be performed many places in the system chain. For example, extract programs could transform code values as they read from DB2 tables or VSAM datasets. A Perl program could parse a file and generate keys for use by bcp or DTS. Finally, stored procedures could split staging table rows in a set-oriented fashion.

**The Environment**

The extraction, transformation, and load environment consists of three architectures (data, application, and technology), and a group of people (management, developers, and support).

The data architecture includes the data itself and its quality as well as the various models that represent the data, data structures, business and transformation rules, and business meaning  
embodied *in* the data and data structures. Data architecture models include conceptual data models, logical data models, physical data models, and physical representations such as COBOL  
copybooks, C structures, and SQL DDL statements.

Technology architecture includes computers, operating systems, data management systems (e.g., Oracle, Sybase, Btrieve), networks, network elements, and the models that represent these. Technology models include network diagrams, computer specifications, and technology standards such as TCP/IP, SQL, and ODBC.

The application architecture includes operational systems, including any support programs such as the ETL programs. Application models include context diagrams, functional decomposition diagrams, state transition diagrams, and data flow diagrams.

The data, technology, and application model(s) that describe the source and target systems are *extremely* important to the success of the extraction, transformation, and load development  
process. If these models don’t exist, you’ll have to create many of them. *Good data, application, and technology models are essential to creating a data warehouse that meets the  
needs of the business.*

**Common Data Sources and Systems**

ETL processes populate SQL Server tables primarily from operational systems, such as Accounts Receivable, Customer Care, and Billing systems. Other data sources may include files from external data providers, the Internet, or data from user desktops (such as a list of products). File formats may include spreadsheets, text files, dBASE, or image files in JPG or GIF format. IBM host data files (in EBCDIC) usually get translated to ASCII either by FTP or a gateway. These days, almost all relational systems support ODBC or OLE-DB connections.

The data source also depends on what subject area(s) is implemented in the data warehouse. Common subject areas include customer, product, marketing, and sales. Financial data usually comes from the accounting package. Customer (people) data may come from customer care systems, marketing databases, or third-party databases. Products, features, and prices often come from the marketingdepartment.

**ETL Issues**

While an ETL solution belongs in the application architecture, the data, technology, and “person” architectures also influence the ETL approach. Data issues include:

* Quality of data
* How similar are the source and target data structures?
* What kinds of dependencies exist in the data?
* How is meta data used?
* How “complex” are the data relationships?

People issues include:

* What technology does management feel comfortable with?
* What in-house expertise does your shop have?
* Who will support the ETL processes?

Technology issues include:

* What is the volume and frequency of load?
* How much disk space will be needed?
* To what extent are source and target platforms interoperable?
* How will the ETL processes be scheduled?

These factors influence the approach to loading the data warehouse, which also affects the cost of the solution (in labor and/or products), and its ease of development, understandability, and  
maintenance. We’ll address each area in the following sections.

**Data Architecture Issues**

**Similarity of Source and Target Data Structures.** Target data structures should have been created from a physical data model, which in turn should have been created from a logical  
data model that was the result of modeling sessions with the business people. The more different the source and target data structures, the more complex the ETL algorithms, processing, and maintenance effort. In terms of physical implementation, flat files are often the most common data source, though these are usually derived from other file formats. If the source and target data structures are not similar, the load processes will typically have to parse the records, transform values, validate values, substitute code values, and generate keys, etc.

**Quality of Data.** Poor quality data should have been identified in a data assessment phase and, ideally, cleaned in the originating operational systems. Common data quality issues  
include missing values, code values not in the correct list of values, dates, and referential integrity issues. It makes no sense to load the data warehouse with poor quality data. As an example, if the data warehouse will be used for database marketing, the addresses should be validated to avoid returned mail.

**Complexity of the Source Data.** Depending on the sourcing team’s background, some data sources are more complex than others. Examples of complex source data may include  
multiple record types, bit fields, COBOL OCCURS clauses, and packed decimal fields. This kind of data will translate into requirements for the ETL tool or custom-written solution since they are unlikely to exist in the target data structures. Individuals on the sourcing team that are unfamiliar with these types may need to do some research in these areas.

**Dependencies in the Data.** Dependencies in the data will determine the order in which you load tables. Dependencies also tend to reduce parallel loading operations, especially if  
data is merged from different systems, which are on a different business cycle. Complex dependencies will also tend to make the load processes more complex, encourage bottlenecks, and make support more difficult.

**Meta data.** Technical meta data describes not only the structure and format of the source and target data sources, but also the mapping and transformation rules between them. Meta  
data should be visible (and usable) to both programs and people. Microsoft’s DTS uses meta data in the repository to transform and load data sources.

**Application Architecture**

**Logging.** ETL processes should log information about the data sources they read, transform, and write. Key information includes date processed, number of rows read/rows written,  
errors encountered, and rules applied. This information is critical for quality assurance, and serves as an audit trail. The logging process should be rigorous enough so that you can trace data in the data warehouse back to the source. In addition, this information should be available as the processes are running to assist in estimated completion times.

**Notification.** The ETL requirements should specify what makes an acceptable load. The ETL process should notify the appropriate support people when a load fails or has errors.  
Ideally, the notification process should plug into your existing error tracking system.

**Cold Start, Warm Start.** Unfortunately, systems crash. You need to be able to take the appropriate action if the system crashes with *your* ETL process running. Partial loads  
can be a pain. Depending on the size of your data warehouse and volume of data, you may want to start over (cold start) or start from the last known successfully-loaded records (warm start). The logging process should provide information about the state of the ETL process.

**People Issues**

**Management’s Comfort Level with Technology.** How conversant is management with data warehousing architecture? Will you have a data warehouse manager? Does management have  
development background? They may suggest doing all the ETL processes with Visual Basic. Comfort level is a valid concern, and these concerns will constrain your options. Seminars, magazine articles, and industry “expert opinion” will help you get your points across.

**In-House Expertise.** What is your shop’s tradition? VB/SQL Server? MVS/CICS/COBOL? VAX/VMS/DCL? UNIX/Sybase? Do your UNIX machines have the Perl distributions? ETL solutions  
will be drawn from current conceptions, skills, and toolsets. Acquiring, transforming, and loading the data warehouse is an ongoing process and will need to be maintained and extended as more subject areas are added to the data warehouse.

**Support.** Once the ETL processes have been created, support for them, ideally, should plug into existing support structures, including people with the appropriate skill-sets,  
notification mechanisms, and error-tracking systems. If you use a tool for ETL, the support staff may need to be trained. The ETL process should be documented, especially in the area of auditing information.

**Technology Architecture**

**Interoperability between Platforms.** There must be a way for systems on one platform to talk to systems on another. FTP (File Transfer Protocol) is a common way to transfer data  
from one system to another. FTP requires a physical network path from one system to another as well as the Internet Protocol on both systems. External data sources usually come on a floppy, tape, or an Internet server. More flexible (and more expensive) options for data access include database gateways such as OmniConnect.

**Volume and Frequency of Loads.** Since the data warehouse is loaded via batch programs, a high volume of data will tend to reduce the batch window. The volume of data also affects  
the back out and recovery work. Fast load programs (e.g., from Platinum) reduce the time it takes to load data into the data warehouse.

**Disk Space.** Not only does the data warehouse potentially have requirements for a lot of disk space, but there is also a lot of “hidden” disk space needed for staging areas and  
intermediate files. For example, you may want to extract data from source systems into flat files and then transform the data to other flat files for load by native DBMS utilities.

**Scheduling.** Loading the data warehouse could involve hundreds of source files, which originate on different systems, use different technology, and are produced at different times.  
A monthly load may be common for some portions of the warehouse and a quarterly load for others. Some loads may be on-demand, such as lists of products or external data. Some extract programs may  
run on a different type of system than your scheduler.

**Common Operations**

Listed below are some common operations you may need to perform on the data you load.

|  |  |
| --- | --- |
| **Operation** | **Possible Solution** |
| Generate a key | Identity column; primary key generator |
| Translate a code | if-then logic; lookup |
| Split data from one source into two targets | Multiple write statements |
| Merge two data sources into one |  |