Paper AD01-2011

ETL Anatomy 101

Tom Miron, Systems Seminar Consultants, Madison, WI

Abstract

Extract, Transform, and Load isn't just for data warehouse. This paper explains ETL principles that can be applied to any application that handles data feeds for reporting, decision support, modeling, and, of course, data warehouse/mart. Here we will expand on my MWSUG 2010 paper "Introduction to Business Intelligence" by drilling down into the ETL component of BI systems. The goal is to demystify ETL terminology, explain ETL design, and show how your project can benefit from ETL or an ETL-like process.

What Is ETL?

ETL is Extract, Transform, Load. While ETL is often associated with formal data warehouse and data mart systems, ETL, or an ETL-like process is a key element of many reporting and analysis systems even if these are not considered part of a data warehouse.

The key feature of ETL is that it brings data from the outside world under the control of your application.

What We'll Cover

We'll discuss the key features of ETL systems, whether they feed a data warehouse or a business reporting system, including:

- Input and output
- Basic functions
- Design considerations
- Inside ETL (selected topics)
- ETL checklist

ETL In

Common ETL inputs are operational data and data "feeds."

Operational Data
Examples of operational data are sales transactions and web server logs. Operational data represents key business events and are often systems of record, meaning the event doesn't exist unless it's in the operational system.

Example: a sale isn't "real" until it's represented in the sales transaction system.

Typically, you will not be directly querying an operational data system. Two reasons for this:

1. Operational systems are usually optimized for insert, not query. They are optimized for insert to avoid delays or system vulnerabilities. When you can't add a sale to the sales system you've lost it!
2. Operational systems are too important to be subject to the query whims of data analysts. You do not want the sales system tied up because of a poorly written query.

Since you can't or shouldn't work directly with operational data you need to make a working copy of this data that you can manipulate without impacting other systems. ETL systems do just that. They "extract" from operational system and create a fixed-in-time snap shot of the data. End users can then use this snap shot for data analysis.
If you're extracting operational data you're working with ETL.

Data Feeds
Data feeds come from external sources such as financial data providers, off line extracts, Bob's month-end sales spreadsheet. Data feeds may represent operational data but don't come directly from operational systems.

Data feeds are problematic because data and file formats can differ widely. Data element names vary. The timing and reliability of the data may be in question. It is just these sorts of issues that make ETL necessary.

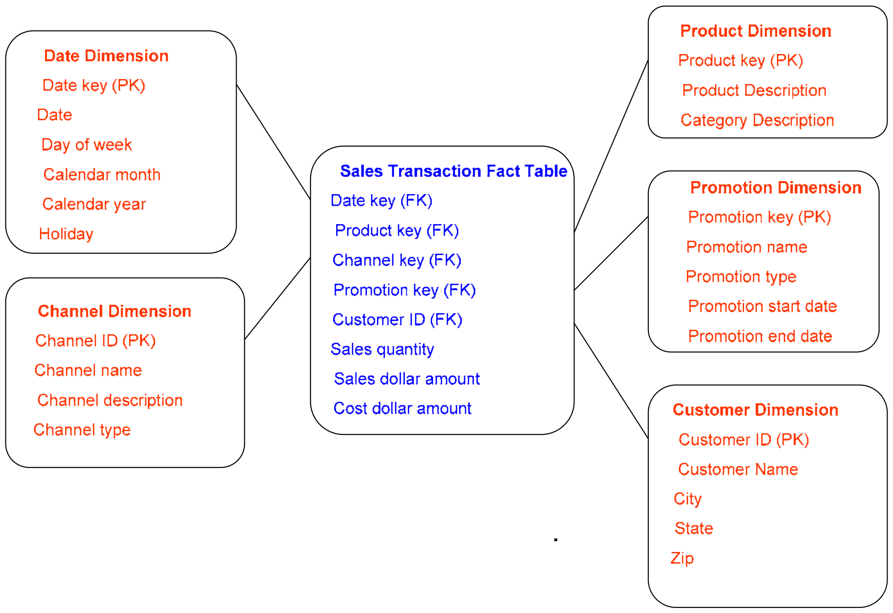
Live Feeds
Most ETL systems gather data on a periodic basis, e.g., update overnight starting at midnight. You may have a requirement for continuous or streaming data feeds. Here, we'll just acknowledge such needs but concentrate on the more common periodic update.

ETL Out

ETL output can vary as much as input. The key feature of ETL output is it's value-added nature. The inputs have been transformed in some way that makes the data more useful and more reliable. Common ETL outputs are dimensional databases, whether part of a data warehouse or not, and business reports.

Dimensional Data
When working with a formal data warehouse, ETL output is often a series of related tables arranged in a dimensional model. A full discussion of dimensional models is out of scope here, but It turns out that the dimensional model is useful outside the world of data warehousing so we'll give it a quick look.

A dimensional model looks like this:



The fact (or measure) table in the center is keyed to a number of dimension (or attribute) tables. The overall arrangement resembles a star, lopsided in this case, so it's also known as a star-schema.

Each unique attribute row, say a product in the Product Dimension table above, is identified by a key value that has no intrinsic meaning. It's just a number, unique in the table. This is called a surrogate key. When ETL output is a dimensional database it must be able to use an operational key, such as a product name, to look up the proper surrogate key. This will be discussed below.

Business Reports
Reporting, analysis, or graphics systems may have no specific component called "ETL", but most do include the basic ETL functions of getting outside data, validating or adding value in some way, and transforming that data to something that can be represented on the report. ETL concepts are useful for such non-data warehouse systems, many of which are large and complex. These systems have all the same issues as the formal data warehouse just without the warehouse sitting between the input data and the end user.

Basic ETL Functions

ETL functions can be broken down to any level of detail. Here, we'll look from a high level at some selected functions:

- Outside data interface
- Data formatting
- Handling time
- Data conformation
- Process management

Outside world interface
ETL is the interface between the wild, uncontrolled world outside of your application and your perfectly controlled system. Since you're designing the application, you control what's going on…most of the time.

The ETL system must have the technical means to access outside data: FTP, database access methods, spreadsheet translation, cross operating system transfers, etc. So a first task with ETL systems is to match up these technical means with the data you need to input.

Data formatting
The ETL system must be able to transform the input file and data formats into a common format, typically, relational tables. The ETL must have the means to accomplish data reformats and write the results to your database or otherwise publish the data.

Dealing with time
Most reporting systems deal with events at a discreet moment in time: sales last month, impressions per hour, average account balance first quarter. The ETL must be able to normalize the various input data to a common time frame and aggregate to the time grain required for output.

We'll look more at time issues below.

Data conformation
One way to think about data conformation is: an entity (attribute or fact) has the same meaning no matter where it's used. I would add: an entity is represented in only one way.

Example:
You have a sale for a product. How do you identify that product? An operational id (as opposed to the surrogate key) might be a product name. Problems arise when the sales staff calls the product Wonder Widget and production calls it Widget 8724. If your analysts need to relate manufactured products to sales you need to deal with these differing identifiers.

Process management
Process management is a catch all for handling the many possible process control, metadata, logging, auditing, and recovery functions. Every system will have its own requirements. Be aware that this aspect of the ETL is easy to under estimate. A comprehensive recovery system can take more time to write than the basic data handling features of the ETL.

ETL Design Considerations

Write it yourself?
Small ETL (or ETL-like) systems are probably less expensive to create in house using current tools for which you have in house skills. If skills or time are not available in house, or the system is complex, an ETL product may be considered.

Naming conventions
If I had a nickel for every hour programmers have wasted trying to remember if it's "transaction", "trans", or "tran"… Naming conventions is not a sexy topic but a very important design element to nail down. Common, published names for tables, files, columns, and business entities save time, frustration, and debugging and support effort.

Data volume and storage space
Disk space required for ETL intermediate processes and output storage must be considered. Use a spreadsheet calculation of anticipated rows times bytes per row. Be sure to account for intermediate and temporary data, indexes, backup, and growth.

Time
There are two time related issues to consider:

1. Update cycle time
2. Time precision

Cycle time relates to how often you update the data. Cycle time is dependent on the frequency of your data feeds. You can't do a complete update of the output database or reports any more frequently than your least frequent feed. And your update cannot be more current than the end date of the oldest feed in the update cycle. These limitations hold regardless of the time precision of the data. Example: a server log extract shows page hits per minute but the extract is produced daily. Your update can't be more frequent than daily.

Another cycle time issue relates to the synchronization of your data feeds.

Example:
A weekly sales summary is provided by your sales department. The weekly sales summary data is for the seven day period Sunday through Saturday. Your client, the marketing department, requires a Monday through Sunday summary. The ETL can take two weeks of input data and apply some proration algorithm to approximate Monday through Sunday sales. But: 1. Marketing's Monday-Sunday report is delayed by six days; 2. About one seventh of the sales are an approximation.

Precision (or grain) of the time data means: the snap shot frequency of your feeds, regardless of the update frequency. As with the update cycle you are limited by your feeds.

Example:
The server log shows impressions per hour but the orders transaction extract is by day. You cannot directly relate hourly impressions to orders. You can aggregate impressions by day and then relate them to orders. Or possibly, you could apply a disaggregation algorithm to the daily orders to distribute them by hour. But you cannot publish actual hour by hour impressions per order.

Data quality commitments
This aspect of the ETL is easy to overlook. It means: Nail down who's responsible for what level of data quality for each feed and push that responsibility as far upstream as possible, hopefully outside the realm of your ETL.

Example:
You contract with a vendor for financial information. Ensure that your contract specifies exactly when feeds must be available, what the file and data formats are, and, for example, that two weeks notice is required for any data format changes. Your ETL has to be coded to handle all data uncertainties. If you can reduce those uncertainties you can simplify the ETL and reduce support time. Of course, even a contractual commitment doesn't absolutely eliminate the possibility of bad data or delays.

Some data quality responsibilities cannot be handled by ETL in any case because not enough information is available to flag or correct the data.

Example:
An outside vendor provides store level sales data with stores identified by a proprietary code. There's an error in the vendor data and the wrong code is used for a series of stores. It is unlikely that you would be able to correct or even detect this in your ETL validation.

Metadata or not?
Do you want metadata generated by the system? Metadata examples: column name lists, row counts, update and insert counts, version stamps, any information about the information itself. Generating and maintaining comprehensive metadata can be a complex task so follow the rule: *Every solution has to have a problem.*

Audits?
Do you have compliance and auditing requirements? Identify audit points and data tracking requirements. Audits may require intermediate data storage. Factor this into disk space requirements.

Recovery and backup?
Can you use system level backup or do you need to develop a new back up system for your ETL?

What are the recovery requirements? Do you have a service agreements with downstream users?

It can be difficult or impossible to reproduce data feeds so at a minimum you should capture and store your raw data feeds through one successful ETL cycle. This allows you to reproduce the cycle without tapping your feed sources for resends.

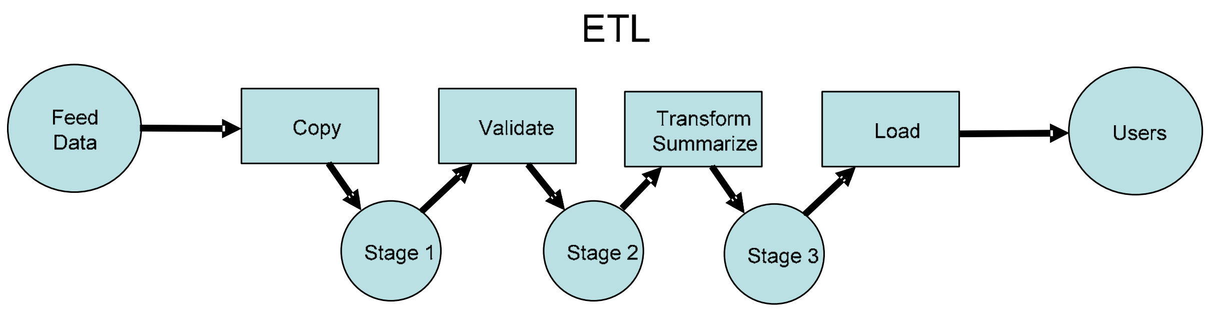
How will you handle attribute changes
This is a classic data warehouse problem but applies to reporting and business intelligence systems as well. What you do when you have 100,000 transactions with ABC, Inc. and they're bought out by DEF, Inc.? Do you go back and change the 100,000 transactions to DEF? Do use a "bought out" attribute and keep ABC? Murphy's Law of attribute changes: Any attribute that can change will change. More on this below.

Inside ETL

Let's looks at a few ETL issues in more detail:

1. Staging
2. Key look up
3. Attribute change

Staging
Staging refers to storing intermediate copies of data as it moves through the ETL. See below.



Staging facilitates recovery without having to revert to the original feeds. Stage data is also useful for audits.

Staging allows you to isolate changes introduced by ETL processes. For example, comparing the data in the Stage 1 and Stage 2 tables above allows you to examine what the Validate process does.

Staged data is useful during system development because it allows you break up the system and examine the output of each piece. It promotes modular systems.

With clever programming it may be possible to fly all data through the ETL without landing in an intermediate file. But unless this is necessary for performance, staging your data will almost certainly prove useful.

Key look up
If you are assigning surrogate (meaningless) keys to attributes from the feed files you will need some way of translating an operational key or text description to the surrogate key. Data feeds with information on Acme Bearings Incorporated may identify the company is various ways:

Acme Bearing, Co.
Acme Bearings
Acme Bearings, Inc.

The ETL needs to assign the official key (custID) and name from the customer table:

custID	**customerName**	**region**	**state**
4813	Pauls Pencils	MW	KS
4814	**Acme Bearings Incorporated**	NE	NY
4815	Home Again Cleaning	NE	ME

There are several approaches to this problem. As with the data quality commitment discussed above you may be able to enforce conformity onto your feed providers. But if names will be variable you'll have to develop a strategy to normalize them.

Even if company name never varies you still have to decide how to look up the custID (4814) for Acme Bearing Incorporated. Once you have the standardized name you could query the customer table directly on name to return the key. In a large and busy system this may impact end users of the table. Another strategy is to maintain a key table within the ETL "back room" that holds only the operational key, name in this example, and the surrogate key.

custID	customerName
4813	Pauls Pencils
4814	**Acme Bearings Incorporated**
4815	Home Again Cleaning

Querying this table will not impact end users and it is controlled directly by the ETL, not a database administrator who may have service commitments that do not align with the needs of the ETL process. Of course, a separate key look up table has to be maintained to match the customer table.

Attribute change
This issue is also known as "slowly changing dimensions."

Example:
Acme Bearings Incorporated changes their name to Bearings R Us. What do you do? This problem has been around as long as attribute tables themselves and there are three well defined strategies or "types" for handling it.

Type 1 – Simply change "Acme Bearings Incorporated" to "Bearings R US" in the customer table. Now all past transactions with customer key 4814 refer to "Bearings R Us" and the fact that transactions before the change referred to a different company name is lost.

Type 2 – Insert a new customer row, with a new custID key, for "Bearings R Us." Past transactions with the company original name are preserved and the new name is used going forward. The fact that "Acme Bearings Incorporated" and "Bearings R Us" represent the same entity is lost.

Type 3 – Incorporate history attributes (columns) in the original customer table.

custID	customerName	previousName01	previousName02
4813	Pauls Pencils	Pauls Smartphones	
4814	**Bearings R Us**	**Acme Bearings Incorporated**	**Wayne's Machine Shop**
4815	Home Again Cleaning	Maid For You	Septic Solutions

In addition to previous names you would probably record the date of change.

There is no best solution to this problem and there are ways to mitigate the limitations of each type. You need to give this careful thought.

ETL Checklist

Define data quality responsibilities.
Technical means to read all anticipated input file and data formats and write output formats.
Ability to reformat data
Time and date handling
Staging
Backup and recovery
Internal reporting, auditing, and metadata
Anticipate and provide for changes and corrections
System support

Finally…

ETL is not necessarily for data warehouse:

ETL system considerations apply any time you're handling data feeds from outside sources and must publish that data to the end user environment.

Contact

Your can contact the author at:

Tom Miron
Systems Seminar Consultants
2997 Yarmouth Greenway Dr.
Madison, WI 53711
608 625-4541
608 278-9964
tmiron@sys-seminar.com