**Real-Time ETL (Extract, Transform, Load) *[Written 2012, for work done in 2007, and an idea conceived in 1999]***

Sometimes I have my head buried and focused in my work and can lose track of what is happening, or in this case, not happening in the mainstream. Recently I have done some research on the net and come up with very little regarding real-time ETL or transactional ETL.

As someone that understands the necessity for "buzzword compliance," I searched high and low on the terms "real-time ETL" and "transactional ETL" and was very surprised to see very little on the subject. What I did read left me scratching my head. It was mostly explanations as to why is was not possible or feasible.

**Real-Time ETL is possible**

I know Real-Time ETL is possible because I and the team I am a solutions architect for have been doing it for the past five years. Sometimes when you do something you just assume that everyone else is doing it too.

**Defining ETL**

I am assuming my reader knows what ETL is, but I think we need to come to a common definition of the term for my discussion to make sense to you. ETL stands for **Extract**, **Transform**, and **Load**. The "extract" we should all be familiar with. It involves getting the data out of the source system. We also know that "load" involves getting the data into the source system. I want to note there is nothing in the acronym that suggests the means that we must extract or load. It is usually some batch process, but it can also be a "connected feed" that is real-time (or some degree of latency). It can also be an SQL trigger on the source system that feeds some intermediate staging location. Now the "transform" term is a little dicier and subject to some interpretation.

When it comes to technical terms I have find we techies like to take common words and severely restrict their definition to a much narrower scope. I tend to look at the term "transform" in a broader context. First, when transforming data we might need to deal with data type issue, for example, moving an int from a dbms that is defined as 2-bytes to a dbms that defines it as 4-bytes. We might need to upcast or downcast the precision of a timestamp. We may want to transform a char to a varchar or the other way around depending on what data types we have available in our target dbms.

We sometimes even have to make data type transformations for more exotic things like converting a date stored as a number to an SQL date. In an old healthcare medical record database I have worked with we needed to transform a 5-byte timestamp that was stored in a character array to an integer field in the target system.

But data transformation can involve much more than data types. In the case of certain types of warehousing or data mart implementations we might need to take discrete row data and roll it up or pivot it. And of course, row data in the source system is sometimes in a very denormalized state and we need to normalize it as part of the data transformation process and store it multiple tables in the target dbms.

I think why database professionals do not see real-time ETL as viable is first, they might have a narrow definition of the term "transform," second, folks that publicly discuss ETL tend to work in the data warehouse domain... which brings me to my next subject.

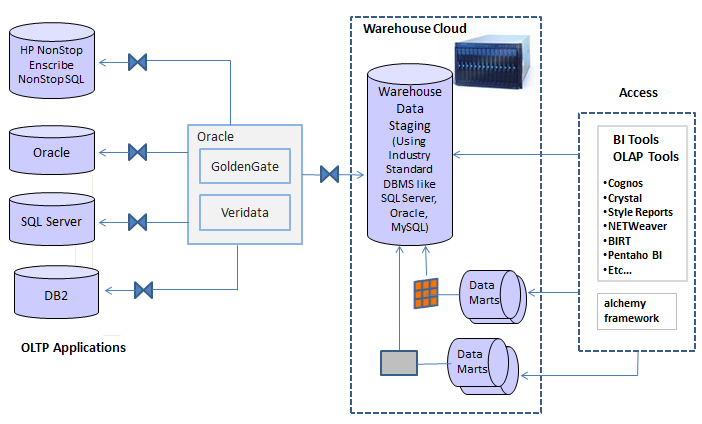
**The ETL Specturm**

**ETL is not just confined to data warehousing activities!**

The concept of extract, transform, and load is practiced in many other venues of software. ETL activities are done ...

* In Real-Time of near real-time interfaces utilizing interface engines such as Clover Leaf and protocols such as EDS or HL7.
* In upgrade work that involves migrating from one version of product software to another.
* In **Master Data Management** activities where source and target systems need to be kept in sync.
* Between development and production systems.
* Between internal development and product release systems.

Now that we have my operating definition of ETL and have discussed the ETL spectrum, I want to get back to the subject of Real-Time ETL starting with a picture.



**Real-Time ETL Requires a Dynamic Pipe or Feed**

This should be obvious. What may not be so obvious is how we achieve the feed. Database occupations tend to be very segregated in terms of work scope. Of course the industry only has three or four generic occupations we all use: Database Administration, Data Base Architect, Database Analyst. But because such deep knowledge is required many of us specialize in particular facets of data management.

Physical DBA's that work in production OLTP (Online Transaction Processing) environments are acutely aware of a tiny little matter :-) called disaster recovery or DR. While production systems use mirrored disks and even multiple controller paths to those disks they still need a means of recovering from a catastrophic system failure. OLTP environments cannot afford to drop transactions or lose data. Think NYSE, Visa, Amazon, Boeing, The Mayo Clinic, DOD, etc... . Enterprise data is an organizations life blood. DR solutions need to capture transactional activity real-time. Batch is just not going to cut it.

**My Accidental Journey Into Real-Time ETL**

About five years ago I was approached by my manager. He told me a customer that was leaving contacted him to see if we could build them a data archival system. The whole story for this is rather long and much of it not related to the subject at hand so I will not digress.

On the surface it seemed like a pretty simple problem to solve. We needed to move data from their source system into a more affordable dbms. Of course, they also wanted a front end application. And their data was stored in a proprietary dbms (H-P NonStop Enscribe). But all of that was really not the challenge.

The real challenges were ...

* They were running an enterprise OLTP application with over 3000 concurrent users and would be for the duration of their move to the new vendor.
* Their move to the new vendor was going to take 14 months from implementation to go-live.
* They did not want to wait until they went live to start the archival project. They wanted to make the transition and retire their system in 60 days or less in order to save millions of dollars.

Hmmmmmm.... batch was out of the question. As an architect I have to dance in many functional and technical camps. Fortunately I had a decent undertstanding (though not a lick of practical experience) with DR practices. I knew many of our customers were using a product called Golden Gate for their disaster recovery solution. In a nutshell, Golden Gate is a product that reads the transaction logs of the dbms and reconstructs the CRUD (Create, Read, Update, Delete) transaction to a DR system.

One thing I could immediately take to the bank is that any software that is used for disaster recovery that has a history in production has to be rock solid. Big check mark there. A production DR solution cannot be resource invasive to the production system. The beauty of a dbms is its ability to trace every transaction for recovery. The transaction logs can be segregated from the OLTP resources in a way that does not impede the performance (to a large degree) of the OLTP environment. And, the transaction is going to contain every single CRUD event for whatever periodicity they set before rolling the logs to tape. Another big check mark was I needed to begin extracting data from a system that would still be in use for at least 14 months. Finally, the DR solution essentially builds a mirrored database on another system. Winner, winner, chicken dinner! We have the trifecta that I needed to take on this gig.

**Real-Time ETL - What is Real? What is Practical?**

If you go back up to the diagram you will see that a product like Golden Gate can be used to feed data from OLTP source systems into a centralized data warehouse. In dealing with data warehousing activities there are essentially four types of OLAP (On-line Analytical Processing) operations. They are:

1. Slice and Dice (partial extraction)
2. Pivot
3. Drill-down
4. Rollup

Of our fours OLAP operations, we are presented with a few challenges. If we extract the data from the source system and do "rollup" as part of the transformation process how do we know the data that we rolled up is clean? **WE DON'T!** For any of the four OLAP operations, if we are batch loading how do we know that the data has not changed on the source system? **WE DON'T!** unless we go and reload it.

**Understanding the Nature and Persistance of the Data You Are Working With**

There are instances when a row of data on the source system has come to a stable persistant state, meaning a state where it is no longer being updated. For example, closed accounts or closed episodes of care in a medical record. There are typically going to be markers and flags in the source database that can tell us the state of a row of data. But, most data is fluent and can be changed. For instance, we cannot assume that month end receivables that have been closed are not going to change. Someone could discover an accounting error months down the road that requires modification to the electronic books. If the error is widespread it could tell a totally different story than than the one that is currently being told in the warehouse that we have rolled up and reported on!

Remember, with batch feeds you are disconnected from the source. Gross changes to the source system might not be reflected in the target which can impact decision-making when viewing and summarizing data across a wider spectrum.

We could write an entire book on this subject but for now we need to move on.

**Regarding Data Governance**

Of the four OLAP operations if you are rolling up data into your warehouse that is not persistant on the source system how do you validate it? If you are pivoting data or rolling up data as an algorthmic function as part of the extract process, the job of data governance between what you have on the source system and what you have on the target system becomes quite a challenge.

As a quick aside, we sometimes use a product called veridata in the systems we build. This product ensures and audits that the target system data matches what we received from the source system. Those audit reports can be stored and referenced and come in handy when the data on the source system is purged.

**Data Storage Is Cheap**

If you noticed in the diagram above our warehouse is essentially positioned as the target system from our various source system feeds. As part of the feed we can implement the OLAP slice and dice function as part of the extraction process itself. In other words, in our Real-Time feed we are only bringing into the warehouse the columns and the rows from each table that we need.

Disks are cheap (see the I in RAID) and SAN solutions are readily available and easy to implement and administer. And, by bringing the data from the source system directly into the warehouse before transforming it (with the exception of slice and dice) we end up with traceable data governance back to the source systems (assuming they are not purging the data on the source system).

**Advantages of Real-Time ETL**

* Greater access to the freshest data for decision makers.
* The source system and the target system stay in sync with one another as they are connected though a zero or near-zero latency pipe.
* The target system gets updated as activity occurs on it. However, it may not be desireable to delete rows on the target system that have been deleted on the source system. The golden gate software allows us to configure it so certain data is not deleted on the target system. (I do not know if other vendors' products that do ETL by reading the dbms transaction logshave this functionality)
* It saves the resources consumed when performing batch ETL directly againt tables within an OLTP space.
* Makes data governance between the source and target systems easier to deal with.

**Disadvantages / What Real-Time ETL Does Not Do For Us (There is no silver bullet)**

* It does not make rolling up and summarizing data in a mart or warehouse any easier than batch ETL.
* Requires monitoring of the transactional feed to catch any missing transactions (though this has benefits when it comes to data governance)

**Parting Words and Gifts**

Disaster recovery (DR) products have been around for years. They are solid and proven in the marketplace. For whatever reason these products have gotten little use outside of the OLTP environment. For the past five years I and the team I work with have built data warehouses, data marts, and data archival systems using these tools. We have used them for master data management syncronization and for upgrading product level enterprise dbms driven OLTP software that have long implementation times and must run 24/7 with little to zero downtime between switchover.

Real-Time ETL also has great benefit by "off-boxing" data from OLTP system for regular 'ol reporting (a subset of what we now call Business Intelligence or BI.) Miminizing ad

-hoc and scheduled reporting through "off-boxing" frees resources and allows for greater scalability of the OLTP environment.

If properly implemented real-time ETL allows the decision makers access to the most current data. In the digital world we live in access to timely information is worth its weight in gold.

If you made it this far my hope is that this has given you some food for thought. I also thank you for taking the time to hear what I have to say on the subject.

I like to get feedback but do not have time to combat all the spammers out there. The best means to contact me is by sending me a message through LinkedIn (I think you need to be a member to do so). Once you signup, just request to add me to your network and I will accept. I will do my best to respond to your message.

Special Thanks to Kate Matthews for editing this piece. She said she did not understand a word of it :-) but her markup was invaluable.

**How do businesses use ETL?**

Data plays a critical role in nearly every business operation. For it to be valuable, it must be moved and prepared for use, which means you need an ETL process. The use cases for ETL include:

* Migrating data from one application to another
* [Data replication](https://www.talend.com/resources/l-data-replication/) for backup or [redundancy analysis](https://www.talend.com/resources/what-is-data-redundancy/)
* Operational processes like migrating data out of a [CRM](https://en.wikipedia.org/wiki/Customer-relationship_management) into an ODS (Operational Data Store) to enhance or enrich the data and then pushing it back into the CRM
* Putting data into a [data warehouse](https://www.talend.com/resources/what-is-data-warehouse/) in order to ingest, sort, and transform it for business intelligence
* Migrating applications from on-premises to cloud, hybrid cloud, or multi-cloud infrastructures
* Synchronization of key systems

**How ETL and ELT processes work**

ETL is typically a three-step process. It is becoming increasingly common for data to be extracted from its source locations, then loaded into a target data warehouse and transformed *after* loading. This process is called ELT instead of ETL. Find out more about [ETL vs ELT](https://www.talend.com/resources/elt-vs-etl/).

Here are the three steps of the ETL process:

**Step 1: Extraction**

The goal of ETL is producing cleansed, accessible data that can be used for analytics or business operations. Raw data must be extracted from an array of sources, including:

* Existing databases
* Activity logs like network traffic, error reports, and more
* Application performance and anomalies
* Security events
* Other transaction activity that must be reported for compliance

The extracted data is then sometimes placed in a destination like a [data lake](https://www.talend.com/resources/data-lake-vs-data-warehouse/) or data warehouse.

**Step 2: Transformation**

The transformation stage of ETL is where the most critical work takes place. This means applying any business rules to the data to meet reporting requirements. Transformation changes the raw data to the correct reporting formats. If the data is not cleansed, it then becomes more challenging to apply the business rules for reporting.

Transformation is achieved through a series of rules and regulations that your team outlines. The standards that ensure [data quality](https://www.talend.com/resources/definitive-guide-data-quality/) and accessibility during this stage should include:

* **Standardization** — defining what data will be in play, how it will be formatted and stored, and other baseline considerations that will define every step that follows.
* **Deduplication** — reporting duplication to data stewards; excluding and/or discarding [redundant data](https://www.talend.com/resources/what-is-data-redundancy/).
* **Verification** — running automated checks to compare similar information like transaction times and access records. Verification tasks further prune unusable data and can red-flag anomalies in your systems, applications, or data.
* **Sorting** — maximizing efficiency in data warehouses by grouping and storing items like raw data, audio, multimedia, and other objects in categories. Transformation rules will determine how each data piece is classified and where it will go next. Often, the ETL process is used to build aggregation tables for summary reporting. This will require sorting and then aggregating.
* **Other tasks** — you define and set to automatically execute.

These transformation steps pare down what was once a mass of unusable material into a data product you can present in the final step of ETL: the load phase.

**Step 3: Loading**

The last step in a typical ETL process is to load extracted, transformed data into its new home. There are two typical ways that data can be loaded into a data warehouse — full load and incremental load.

Executing a lone ETL cycle or a scheduled series of them can be done by executing a task from a command line or GUI interface. But there are certain things to watch out for. Managing exceptions, for example, can become a cumbersome process. Often data extracts can fail if one or more systems are down. Bad data in one system can impact data being extracted from another, so monitoring and error handling are key activities.

***Learn more about ETL in this three-minute video***

**Modern data practices: ETL vs. ELT**

ELT is a tried-and-true process of managing data’s journey from the source to a state where it’s ready to be used for [business intelligence](https://www.talend.com/resources/what-is-business-intelligence/). But with [modern, cloud-based infrastructure technologies](https://www.talend.com/blog/2016/03/14/the-five-phases-of-hybrid-integration-part-ii/), systems now are able to support large data storage and scalable compute power at lower costs. Therefore, it becomes advantageous to keep data in a large and ever-expanding data pool, with endless fast processing power to maintain the extracted raw data.

In some ways, ELT is a preferable approach to [data management](https://www.talend.com/resources/l-data-management/), because all the raw data can be kept until it is ready to be used. It therefore becomes easy to focus exclusively on data extraction and loading. However, it’s important not to forget about data transformation, which can have numerous benefits, including:

* Preparing data for [machine learning](https://www.talend.com/resources/what-is-machine-learning/), which will dramatically improve the accuracy of the algorithms
* [Cleansing data](https://www.talend.com/resources/what-is-data-cleansing/) to improve data quality
* Deduplication and matching of data
* Standardizing data values

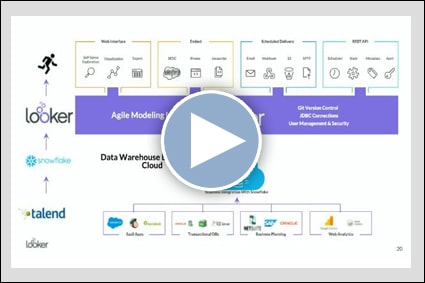
Even in a data lake-centric world, data transformation is critical to extract the full value out of enterprise data.

**ETL tools vs. hand-coded ETL**

Many IT professionals wonder if hand coding isn’t a better option than investing in yet another tool. Many ETL functions can be accomplished with hand coding, but in most cases it's more scalable and less expensive to use a tool for ETL processes.

There are numerous challenges with hand coding. Managing, supporting, and reusing hand coding is complex. It can be very difficult for one developer to learn another developer's code, let alone reuse it. ETL tools create a visual representation of a data flow that is much easier to understand. With hand coding, we often see developers rewriting other developers' code, because it's easier to rewrite the code than it is to learn what someone else has done. This is why maintenance costs are often 2x higher with hand coding. This same barrier makes it much less likely that one developer will reuse another developer's code.

Data integration tools automatically include features such as parallelization, monitoring, and failover. If you wanted those features when hand coding, you would need very skilled coders to learn all those techniques. Finally, a custom-coded approach to data integration inhibits scaling and innovation because developers skilled at working with custom-coded integrations are scarce. The upfront savings you might enjoy when forgoing an ETL tool are swallowed up by the vast increase in maintenance costs.



Watch Making Today’s Data Rapidly Consumable now.   
[**Watch Now**](https://www.talend.com/resources/making-todays-data-rapidly-consumable/)

**ETL and moving to the cloud**

Cloud compatibility has proven critical for many organizations. As companies move to the cloud, they often have to re-evaluate their existing data integration and ETL tools because many traditional tools don't work well in the cloud. Not only that, but most enterprises have multiple clouds or find that they wish to migrate from one cloud provider (e.g. [AWS](https://www.talend.com/resources/data-integration-aws/), [Azure](https://www.talend.com/solutions/information-technology/azure-cloud-integration/), or [Google Cloud Platform](https://www.talend.com/resources/google-bigquery/)) to another.

An ETL tool should be able to work well on any cloud provider and should be able to migrate easily as companies change providers. You won't want to rewrite your data pipelines when you change cloud platforms; instead, you want your data pipelines to be portable so that you can plug in and plug out different clouds, storage technologies, data processing technologies, and cloud databases.

**What to look for in an ETL tool**

[ETL tools](https://www.talend.com/resources/etl-tools/) have the potential to simplify this process a great deal. The features you should look for in an ETL tool are:

* Variety of connectors — there are many systems and applications in the world; the more pre-built connectors your ETL tool has, the more time your team will save.
* Open-source — open-source architectures typically provide more flexibility while helping to avoid vendor lock-in.
* Portability — it's important, as companies increasingly move to hybrid cloud models, to be able to build your data integrations once and run them anywhere.
* Ease of use — ETL tools should be easy to learn and easy to use with a GUI interface to make visualizing your data pipelines simpler.
* A transparent price model — your ETL tool provider should not ding you for increasing the number of connectors or data volumes.
* Cloud compatibility — your ETL tool should work natively in a single cloud, multi-cloud, or hybrid cloud environment.

**ETL with Talend — Talend Data Integration vs Talend Open Source Data Integration**

Talend provides robust data integration tools for ETL processes. Using Talend for data integration allows users to complete integration jobs ten times faster than hand coding, at a fifth of our competitors' cost.

There are two versions of the Talend data integration software — [Talend Open Source Data Integration](https://www.talend.com/products/data-integration/data-integration-open-studio/) and [Talend Data Management Platform](https://www.talend.com/products/data-integration/data-management-platform/). Take a look at a comparison between the two. The Talend Open Source Data Integration product is powerful and valuable for many use cases; it's a great tool for an individual developer to build and run data pipelines.

As customers build larger teams, want more collaboration, want enterprise-class support, and desire even better productivity, then they will want the commercial edition. Talend Data Management platform offers additional features like management and monitoring capabilities, data quality built right into the platform, and additional support on web, email, and phone. Both offer native multi-cloud functionality, scalability for any kind of project, and 900 built-in connectors.

ETL plays a critical role in helping you make use of data in the enterprise. Talend makes ETL processes easy. Download Talend Open Studio for Data Integration today and start delivering trusted data at the speed of business.