# The Evolution of ETL

**-From Hand-coded ETL to Tool-based ETL**

**By**
**Madhu Zode**

Data Warehousing & Business Intelligence Practice

## ABSTRACT

*To build a data warehouse various tools are used like modeling tools to design a warehouse, database tools to physically build the database and loading the data and programming languages to extract the data from sources, apply business transformations and load it in consistent format. To achieve this goal in data warehousing, most organizations were using self-developed custom codes. The data were extracted and transformed using COBOL, PL/SQL codes, Perl scripts, Shell programs, etc and loaded into the target warehouse using SQL loader, BCP, BULK loading. But these hand-coded ETL resulted in a lot of hard-to-maintain "spaghetti" code and it is hard enough to get programmers to document it.*

*In the mid-1990s, vendors recognized an opportunity and began shipping ETL tools designed to reduce or eliminate the labor-intensive process of writing custom ETL programs. One major advantage of ETL tools is their ability to make the "meta data" available to the world outside of development. This whitepaper will be throwing light on the origin of ETL and its generation, comparison of Hand-coded ETL and Tool-based ETL, selection criteria for ETL tools and also the future trends coming in ETL tools.*

## INTRODUCTION:

**ETL** stands for **E**xtract, **T**ransform, and **L**oad. That is, ETL programs periodically extract data from source systems, transform the data into a consistent format, and then load the data into the target data store. During ETL process the data is extracted from an OLTP database or non-OLTP system and transformed to match the data warehouse schema and finally loaded into the data warehouse database. In its simplest form, ETL is copying a data from one database to another.

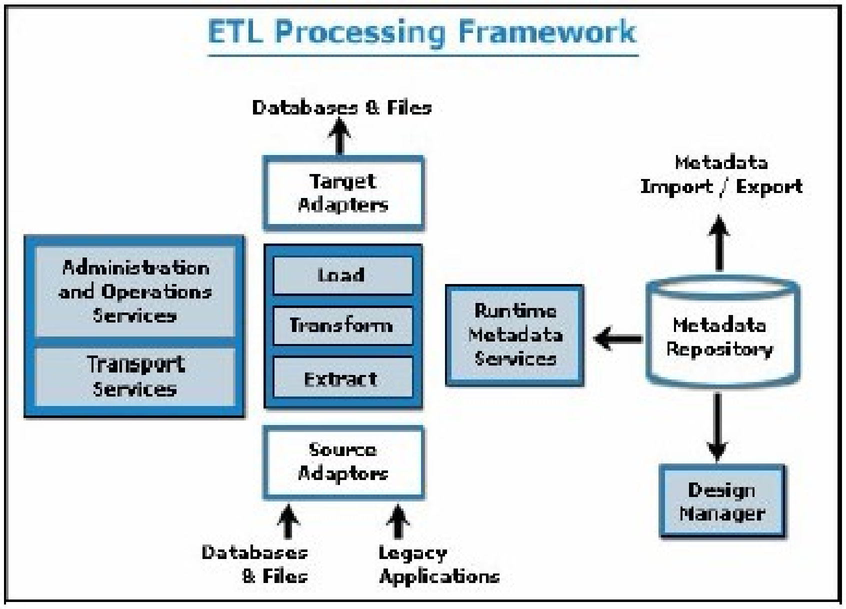
When defining an ETL for data warehouse, it is important to think of ETL as a process, not a physical implementation. The way of implementing ETL can vary from data warehouse to data warehouse and even between department data marts within a data warehouse. In the early 1990s, most organizations developed custom code to extract and transform data from operational systems and load it into data warehouses. In the mid-1990s, vendors recognized an opportunity and began shipping ETL tools designed to reduce or eliminate the labor-intensive process of writing custom ETL programs.

Regardless of how they are implemented, all ETL systems have a common purpose: they move data from one database to another. An ETL system consists of four distinct functional elements:

- Extraction
- Transformation
- Loading
- Meta data

## ETL FRAMEWORK

The diagram below depicts the major components involved in ETL processing. The bullets below describe each component in more detail:



- **Extraction:** This process extract the data from source systems (source system can be OLTP, legacy, mainframe, flat files, etc) data using adapters, such as ODBC, native SQL formats, or flat file extractors. These adapters consult metadata to determine which data to extract and how.
- **Transformation:** This process transforms the extracted data into a consistent data warehouse schema by applying the business rules on it. This process is also responsible for data validation, data accuracy, data-type conversion and business rules application. It is most complicated of the ETL elements.
- **Load:** The ETL Load element is responsible for loading the transformed data into data warehouse using target data adapters such as SQL loader, Bulk process, BCP, etc.
- **Metadata:** The ETL metadata element is responsible for maintaining the information about the data. The metadata repository makes metadata available to the ETL engine at run time.
- **Administration and Transport services:** The ETL process uses transport services such as network and file transfer protocol to move the data between source and target systems. ETL utilities let administrators schedule, run, and monitor ETL jobs as well as log all events, manage errors, recover from failures, and reconcile outputs with source systems.

The components described above used to be coded manually using native SQL codes, C, COBOL and other programming languages. But now-a-days these components are coming with most vendor-supplied ETL tool. These ETL tools combine all of these functions together in single, integrated package.

In the next sections we will be discussing about the hand-coded ETL and Tool-based ETL.

## HISTORY OF ETL

The main purpose of the OLAP systems is to extract the data from different sources, transform it into consistent format and load it into warehouse that is further used by analyst to make business decisions. Most of the organizations are faced with two major options for performing these so called ETL process:
1) Write a custom program in COBOL, C, or PL/SQL to extract data from multiple source files, transform the data, and load the target databases.
2) Purchase any ETL tool available in the market that can perform these tasks.

The first option led them to write hand-coded ETL programs which were hard to maintain and the second option has given way to ETL tools that are very much popular these days.

### HAND-CODED ETL PROCESS

When ETL tools were not in the picture, developers were using custom codes to perform the ETL operations. The programs written using this method were lengthy and hard to document. The ETL developer has to use different programming languages to perform the ETL task, such as; Perl scripts for extracting the data from source systems, performing transformations, and SQL Loader and PL/SQL Bulk procedures were used to load the data in target warehouse. Also some wrapper scripts like shell scripts were also being used to make these ETL as packaged programs. Writing these custom programs does not seem to be feasible option because there are some serious problems with this method of ETL process.

#### Strengths & Limitations

- **Strengths**
    - Though you have to create the metadata manually in hand-coded ETL systems but still you can more directly manage it by yourself.
    - Testing of the ETL code written is easy since many automated unit testing tools are available for hand-coded systems.
    - Hand-coded ETL are more flexible, you can do whatever you want without any limitations.

- **Limitations**
  - While designing a DW application, a provision has to be made to accommodate the changes that should not affect the existing design. To keep up with high volume of changes initiated by end users, hand-written ETL programs have to be continually modified and in many cases rewritten. The effort required to maintain these programs frequently becomes a major burden for the project.
  - Metadata is the backbone of any DW project. Hand-coding of ETL requires maintaining of metadata tables separately. Any new changes require changes to metadata tables that have to be done manually.
  - Hand-coded ETL programs are likely to have a slower speed of execution since hand-generated programs are typically single-threaded, while modern ETL tools generate multi-threaded, directly executable code that can run on parallel, high-speed engines.

## TOOL-BASED ETL PROCESS

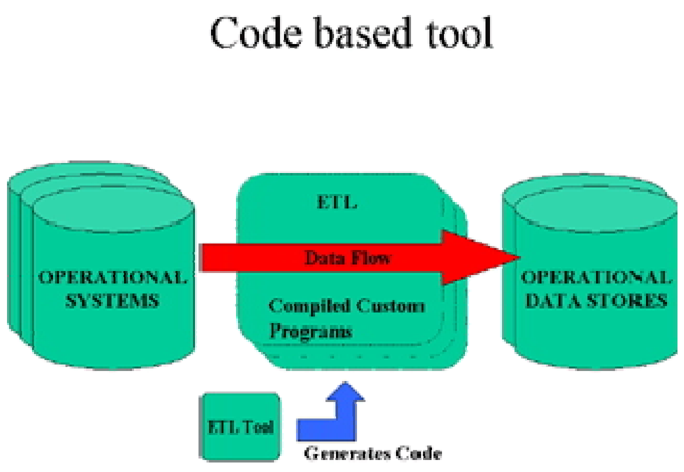
To avoid the overhead caused due to hand-coded ETL process, many vendors have developed ETL tools to perform extraction, transformation and loading process. An important function of these tools is to generate and maintain centralized metadata. As computer systems started to evolve from monolithic mainframes to distributed computing systems, and as business intelligence made its debut, the first ETL solutions were introduced. Initially these tools provided the ability to extract the data from mainframes and load into target database. But the ETL tools have matured enough now-a-days providing user-friendly GUI's, additional functionalities and performance benefits.

The ETL tools may be code-generators or engine-based tools. These graphical tools provide ease of development, by either generating the code or enabling the ETL process through its internal engine. They reduce development and maintenance time and cost.

There is still a debate about whether ETL engines or code generators, which have improved since their debut in the mid-1990s, offer the best functionality and performance. Since that time, several generations of ETL have been produced.

- **First Generation – Code Generators**

  To get rid of writing the complex hand-written codes, vendors started developing the ETL tools in mid-1990s and they started producing the legacy code generators.



Most of the code generating tools at that time was generating COBOL since data was largely stored on mainframes. Extract program was generated automatically as source code that was compiled, scheduled and run in batch mode. The data extraction from source files, transformation and loading of the data in database process used to run on server. These tools were single threaded and was not supporting the parallelism. These code generators were requiring intense programming in COBOL or C. Often, less metadata was generated automatically by first-generation ETL tools.

**Strengths & Limitations**

- **Strengths**
  - Tools are good at extracting the data from legacy systems.
  - Performance was good because of inherit performance of native compile code.

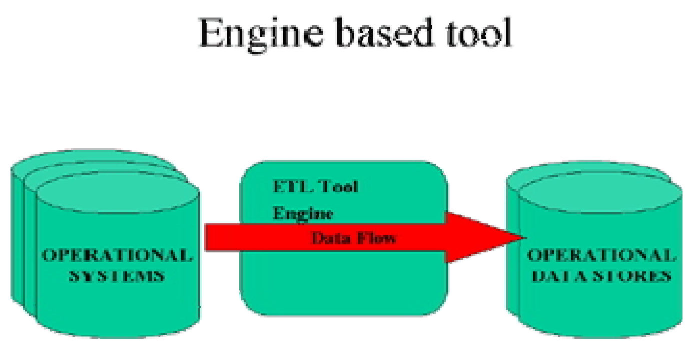
- **Limitations**
  - These tools required in-depth knowledge of programming in COBOL or C.
  - Proven less successful on relational database for handling large volume of data.
  - Unable to support parallel processing.
  - Many transformations require manual coding.

**Examples of First Generation ETL Tools**

**SAS/Warehouse Administrator** developed by **SAS Institute Inc**. in 1997.

- **Second Generation – ETL Engines**

  In the mid-to late 1990's vendors started delivering the "engine based" ETL tools to automate more of the ETL process.



An engine based ETL tool does not generate code but manages the ETL process using an internal engine. These ETL engines use language interpreters to process ETL workflows at runtime. The ETL workflows defined by developers in the graphical environment are stored in a Metadata repository, which the engine reads at runtime to determine how to process incoming data. This approach solved the problem of having to use different languages and required knowledge of only one programming language that is the language of ETL tool itself.

Another significant characteristic of an engine-based approach is that all processing takes place in the engine, not on source systems. Some engines support parallel processing which enables them to process ETL workflows in parallel. But to achieve parallelism in engine-based tool, partitions need to be done at server manager. Also in engine-based tools metadata repository is maintained at different server that needs to be connected from client application.

**Strengths & Limitations**

- **Strengths**
  - The strength of second-generation ETL tools was graphical interface and transformation features provided.
  - Engine-based approach is fast, efficient and multi-threaded.
  - ETL functions are highly integrated and automated.
  - Functions supported include scheduling, monitoring, aggregation, transformation.
- **Limitations**
  - All data, coming from various sources to go to target, had to pass through the engine that process data transformation row by row which make it very slow with significant volume of data.
  - The engines performing all the transformations became a bottleneck in the transformation process.

- Engine-based tools require dedicated Administrators that configure the engine platform to deliver the high performance.

**Examples of Second Generation ETL Tools**

- ✓ **Power mart 4.5** developed by **Informatica Corporation** developed in 1999.
- ✓ **Ardent DataStage** developed by **Ardent Software, Inc.** which is now owned by **IBM corp.**

## ➢ ETL Tools Today

The ETL tools available today are affluent in transformation features. Typically, they support multiple input or output database or flat files, multi-dimensional designs, surrogate key generation, various transformation functions and native database or O/S utility. They have internal metadata repositories that may be different from the data warehouse metadata repository. They eliminate the overhead of developing and maintaining the complex routines and transformations in ETL workflows. Also, these tools are providing user friendly GUI's which enables the developer to work without under going through training. These tools also have the features such as monitoring, scheduling, bulk loading, incremental aggregation, etc.

It has been seen lot many upgradations in both Code generating and Engine based ETL tools from 1990's till today. Previously only COBOL was used in which the code generating tools were generating the source code but today vendors are supporting many platforms in which the code-based tools can generate their code. Most of the code generating tools is using SQL to generate the code and also supporting all SQL features. The best part of code generator is the compile code that they produce which can run on any platform. Also the compiled code is not only fast but it also distributes the load across multiple platforms to increase the performance. The Engine based tools are also emerging very fast. They are coming with the ability to perform complex transformations at a faster speed of execution.

**Strengths & Limitations**

- **Strengths**
  - The ETL tools available today can handle most complex transformations at a faster speed of execution
  - The code generated by the code-based tools is can run on various platforms at very high speed and also enables organizations to distribute loads across multiple platforms to optimize the performance.

  - The most important characteristics of today's ETL tools are the type of parallelism they support. The tools like Ab-initio supports 3 types of parallelism viz. Data Parallelism, Component parallelism and Pipeline parallelism by which the developers can perform the complex processing of jobs at a considerable amount of time.
  - These tools are also having capability to read the XML data very efficiently. Informatica has the feature to read different types of XML file like XML Schema files, XML files, and DTD files.
  - These tools are also having in build scheduler using which we can schedule the workflows.
  - Also they are providing the features like incremental aggregation, slowly changing dimension, normalization of source data. Basically they are supporting all the data warehousing concepts like normalization, denormalization, slowly changing dimension, etc.
  - These tools are also providing the feature of version controlling. Ab-inito's EME (Enterprise Meta Environment) support helps developer to maintain the versions of source code without overwriting the original code, metadata maintenance and documentation.
  - The tools available today are providing the Debugger support with which developer can test the code for defect tracing.
- **Limitations**
  - The tools available today are not enough capable to support real-time applications.
  - Some tools are limited to the source database like OWB that supports only Oracle and flat files as source.
  - Many recent ETL tools do not support integration at the metadata level with end-user tools.

**Leaders in Today's ETL Tools**

✓ **Informatica Corporation's** Power centre is leading in the ETL technology market due to its high scalability and user-friendliness.
✓ **Oracle Corporation has** delivered highly scalable, user-friendly and feature-rich ETL solutions in its Oracle Warehouse Builder 10g ETL tool that is also supporting sources other than oracle and flat files.
✓ **Ab-initio's** co>operating system provides high performance using data parallelism, connectivity to various sources and strong metadata support.
✓ SQL Server 2005 Integration services (SSIS) developed by **Microsoft Technologies** meets most batch-oriented data integration requirement.

## HOW TO SELECT AN ETL TOOL

Selection of an appropriate ETL tool is the most important decision that has to be made in choosing components of a data warehousing application. The ETL tool operates at the heart of the data warehouse, extracting data from multiple data sources, transforming the data to make it accessible to business analysts, and loading multiple target databases. Unlike other components of a data warehousing architecture, it is very difficult to switch from one ETL tool to another. Due to a lack of standards, data definitions and transformation rules cannot be moved from one tool to another.

Following points need to be taken into consideration while selecting an ETL tool.

- **Platform Support:** The ETL tool should be platform independent and can run on any platform.
- **Source Type Independent:** The ETL tool should be able to directly read from the source database irrespective of the type of source whether it is a Mainframe source, RDBMS, flat file or XML file.
- **Metadata support:** The ETL tools Plays an important role in metadata because it maps the source data to target tables. So the metadata generation, which includes source data definition, transformation and target data models should be an automatic process.
- **Functional support:** The ETL tool should support data extraction from multiple sources, data cleansing, and transformation, aggregation, reorganization and load operations. It should have a strong transformation and data cleansing capabilities.
- **Ease of Use:** The ETL tool should be a general-purpose and user-friendly tool which developers can quickly get up to speed with the tool.
- **Version Controlling:** The ETL tool should support the version controlling mechanism though which the developer can maintain different versions of the source code without overwriting the original code. So if any error occurs in the new code then developer can quickly be able to rollback to the old code. Also the tool should prevent multiple developers from unknowingly working on the same extract so that at a time only one can make the changes in the ETL workflow.
- **Parallelism:** The ETL tool should support multithreaded operations and parallel code execution internally, so that a given job can take advantage of the inherent parallelism of the platform it is running on. Also it should support load balancing across servers and ability to handle large data volumes. When faced with very large workloads, the product should be able to distribute jobs across multiple servers.
- **Star Schema Support:** The ETL tool should have the built-in support for creation of slowly changing dimension, surrogate key generation, keying fact tables and building aggregated dimension.
- **Debugger support:** The ETL tool should support the run-time debugging of transformation logic. The developer should be able to see the data before transformation and after transformation.

- **Scheduling:** The ETL tool should support the scheduling of ETL jobs that will be time-driven and should not required human intervention to complete a particular ETL job. It should also support for command line scheduling using external scheduling programs.
- **Deployment:** An ETL tools should support an ability to group the ETL objects and deploy them into test or production environment without an intervention of an ETL Administrator.
- **Reusability:** An ETL tool should support the reusability of the transformation logics so that the developer need not require re-writing of the same transformation logic again and again.

  Besides these there are many more criteria for selection of an ETL tool like it should support an incremental aggregation, complex transformation using built-in transformation objects, able to join data from multiple sources, strong data warehouse administrative functions and more apparent is that the ETL tools should have robust SQL support.

## FUTURE TRENDS IN ETL TOOL

ETL won't go away but it will revolutionize into something else in the future. In this world of competition, all vendors are trying to build the ETL product that will support all the features described above. Today's vendors must focus on "connecting" to everything and providing more near real-time integration features and database-specific generation specialties to survive which is nothing but the Active data warehousing. Active data warehousing places such demand that goes beyond the ETL solutions.

### Where real-time is necessary

"I should receive and manipulate data in near real-time without affecting my operational system. I should be able to perform multiple tasks on multiple source systems at the same time"

This is what today's business users as well as ETL developers are expecting from ETL. Let's say that you are working for a Credit card company that uses an application to look for its customer's usage of the card at the end of their billing cycle. From this your business analysts makes the decisions towards the popularity of the credit card, platinum customers who are making most use of their credit card and thinking of new promotions to be given to such customers. For this you have to maintain Credit database where you will load the data from different sources such as credit applications, transactions, statements, billing cycle and customers account and personal information. You will need applications that will run on multiple source systems to gather the data at one go and separate application to integrate and manipulate this data and load it in a warehouse so that the analysts can make business effective decisions. So business could think that their data warehouse should provide the timely updates which means that if a person is doing a transaction using the credit card at a particular store then its details should be captured at that time itself and available in the warehouse so that they can take the timely decisions on that.

This type of requirement requires the parallel ETL that can assembles the data from such sources and validate it against the existing and next coming data. Your data warehouse has to be Active enough to accomplish such requirement. In active data warehouse, changes in the data value such as at transactional level or state (if the credit card user should be given some promotions) should trigger an event such as a mail should be sent to credit card user that he has been given some bonus points. The possibilities for use with an active warehouse are limited more by imagination and practical reality than by the tool.

One of the major challenges in data warehousing is the volume of data that is increasing day-by-day. Normally a data warehouse is loaded during off-hours so as to keep it ready for business users to query upon it during daytime. So even if the volume of data in a warehouse increases, the loading has to be completed in off-hours only; they cannot be extended. By keeping the increasing data volume in mind and time constraint of loading hours, ETL has to be efficient enough to adjust with this day-by-day increasing data volume. So the next generation ETL tools should be a rapid application development tool to substantially shorten the delivery time and improve application maintenance.

The new ETL tools of the future should contain Web-based business workflow solutions. Using Web services will mean larger presence of XML data. Companies are being compelled by their partners to move to Web-based data sources which has greater potential to integrate the data

## CONCLUSION

The contribution that these ETL tools have made to data warehousing development process is significant. Considering the emerging needs of data integration and data migration it is necessary that features provided by ETL tool should address these changing requirements. The key is to understand your current and future ETL processing needs and then identify the product features and functions that support those needs. With this knowledge, you can then identify one or more products that can adequately meet your ETL and BI needs.

## REFERENCES

1. Wayne Eckerson and Colin White. "Evaluating ETL and Data integration Platforms", TDWI.
2. Rob Karel. "The Forrester Wave: Enterprise ETL, Q2 2007", Forrester Research.
3. Sam Sterling. "Next On The Horizon: Parallel ETL", Teradata Magazine Online.
4. Mimno, Myers & Holum. "Selection of an ETL tool".
5. Yves de Montcheuil. "ETL Engine or Code Generation", Sunopsis corp.