Intro to ETL

# ETL Basics

## What is ETL?

ETL stands for **extract, transform, and load**. ETL processes are processes that extract data from one or more data sources, transform that data to suit our data mart, then load that data into our data mart.

ETL processes are often very large and complex, but at their core have a lot of the same basic elements. There are a lot of excellent ETL tools available that help streamline the creation of these components, such as SQL Server Integration Services (SSIS). In industry it is most common to use these kinds of tools for the generation of ETL.

ETL processes very rarely execute all at once. An ETL process will be made up of many smaller interdependent tasks. These tasks can each take a considerable amount of time and resources. To help coordinate these tasks and provide a save point between them, staging tables or files are frequently used to pass data between tasks.

In this course we will be building a very simple ETL process to load our example data mart. We will be using T-SQL to create this ETL and explore the concepts, but in practice this is rarely done for anything except very simple data marts.

ETL stands for **extract, transform, and load**. It is a set of processes that load a data mart from one or more sets of source data.

## Basic ETL Steps

Despite the name, real-life ETL processes involve more steps than just extracting, transforming, and loading. We will be working with a few in this course.

### Initialize

Each ETL cycle will typically have control, audit, administration tasks that need to be completed prior to starting the first extract. This may include tasks such as:

* Verifying the successful completion of the previous cycle
* Logging the start of the process in a control table
* Verifying that required sources are present
* Creating backups

The initialization step typically only needs to be run once per ETL cycle.

An initialization step prepares an ETL cycle for execution. It executes once per cycle.

### Extract

An ETL process will extract from one or more source data sets. This frequently involves extracting data from a database or a file into a staging table that matches the source format. In order to successfully extract data, the source data set must be in an expected format, so the extract step does also serve as a preliminary validation of source data. Any data rejected at this phase should be reported to the source system so it can be fixed.

Extract processes are purpose-built for the source they work with, so one extract step needs to be created for each source you are working with. When extracting into a staging area, a table will be created for each data source and will closely match the format of the source data set.

*NOTE: A single repository (e.g. a database) may be the origin of multiple sources of data. In the context of an extract, the source is the data set, not the repository.*

An extract process reads data from a data source. One extract process will exist for each source data set. Extract processes usually load into stage tables that match the source format.

### Transform

Data transformation is usually the largest and most complex portion of an ETL process. The purpose of the transformation step is to clean and homogenize the data, and convert it to the format required by the destination tables. This is especially true when dealing with multiple different repositories or systems, which may all have different structures, standards, and conventions.

There are too many different possible transformation tasks to list here, but some common ones include:

* Converting data types
* Splitting or merging columns
* Calculating new column values based on existing ones
* Integrating multiple sources
* Mapping source values to destination values (e.g. converting “ON” to “Ontario”)
* Creating surrogate keys
* Identifying updated SCD records(Slowly changing Dimensions)

As with the extract step, it is common for the transform step to write to a stage table. These stage tables would closely match the destination format.

The relationship between source data sets and destination tables is many to many – one source may be used by many destination tables, and one destination may receive data from many sources. Usually one process will exist for each source-target combination, but processes that are very dependent on each other might be combined.

A transform process cleans and modifies source data to prepare it to be loaded into destination tables. One transform process will usually exist for each required source-target combination, but highly interdependent processes might be combined. Transform processes usually load into stage tables that match the target format.

### Validate

Validation is implicitly part of the extract, transform, and load steps (e.g. if your source file is malformed, your extract process will fail) and for some processes, that validation is sufficient. However, for more complex, sensitive or error-prone processes, distinct validation steps may be built.

Validation can look for any unusual patterns in the data, but some common validations include:

* Abnormally small or large source data sets
* Abnormal distributions of data
* Invalid values
* Failure to adhere to business process rules
* Duplicate or missing values

When anomalies are found, the validation step may be configured to take several different steps, depending on the severity of the issue. These might include:

* Halting the ETL
* Create a report
* Notify an operator
* Notify users
* Notify source systems
* Quarantine offending data

Validation steps are most commonly found between the extract and transform, or between the transform and load. Multiple validation steps may exist per ETL process.

Validation steps look for anomalies in the data and take appropriate actions when anomalies are found. Validation can occur at any point and multiple validation steps may exist per ETL process.

### Load

Once the data is prepared it is loaded into the target tables. At its simplest this is just a straight update or insert into the target tables. Usually load processes are more complex than that, and involve tasks such as recreating indexes, updating views, updating metadata, and notifying users.

Load processes are purpose built for their target tables, so one process needs to be created per target.

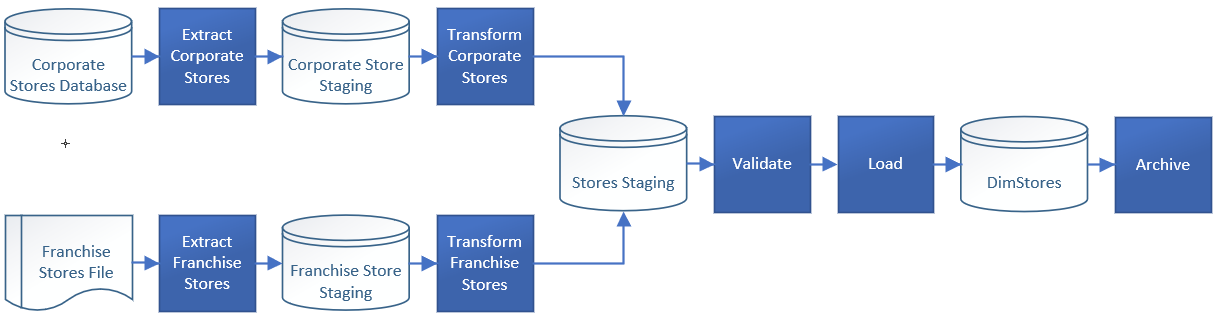
A load process writes data into the final target tables. One load process needs to exist for each target.

### Purge and Archive

Analysts will only require a certain amount of history in a data mart. Data that is no longer required can be purged. Data that is removed from the data mart might be archived for a period of time. Archived data is no longer present in the tables, but could be accessed or reloaded if required. Most commonly, archives are used to satisfy audit requirements.

## Process Summary

A completed ETL process will often look something like the following.



In the above example, we have the following:

* Store data is being retrieved from multiple sources that need to be integrated. In this case, a database and a file.
* Extract processes read source data into stage tables. One extract process and one stage table are created for each source.
* Transform processes read from the source staging tables and write to a target stage table. These processes would include logic that defines how the two sets of data should be integrated. One transform process is created for each source-target relationship. One stage table is created for each target.
* A validation step examines the target staging data. It will take appropriate action if anomalies are found. It might halt the process or send a notification, for example.
* A load step reads staged data into our target table, DimStores.
* An archive process purges old data from our table and creates an archive as necessary.

## ETL Tasks

There are a lot of different steps and processes and tables here, it seems very complex. Why do we bother creating all of this? Why would we not just create a single process that loads our data?

There are several advantages to segmenting the ETL processes this way.

**Coordiation of Tasks**

Tasks in an ETL process are dependent on earlier steps in the process, but they’re often not dependent on all preceding steps. In our example above, the transformation for corporate stores is dependent on the extract for corporate stores, but not on the extract for franchise stores.

Splitting tasks allows them to be more easily scheduled. This prevents tasks from having to wait for tasks that they’re not truly dependent on.

This also helps in complex dependencies, such as where one source is used to load multiple target tables. It makes it easier for each task to wait for all the tasks it is dependent on without having to incorporate them all into the same process.

**Parallel Processing**

When ETL is split into separate processes, independent processes can be run in parallel, reducing the total time required to complete an ETL process. For example, extract steps are not usually dependent on each other. In a process that requires multiple sources, running the extracts in parallel can help improve speed.

**Restart and Rerun**When an ETL process fails, staging tables provide us with a save point that allows us to restart the process partway through. Breaking ETL processes into separate tasks allow us to restart from the failed step instead of restarting from the beginning.

**Testing**Atomic steps are easier to unit test. Each step can be tested independently and do not need to be retested when other steps in the process change.

**Templates**ETL processes are purpose built for their sources and targets, so its not possible to avoid creating new processes as new sources or targets are added. However, tasks of the same type are usually broadly similar. Breaking an ETL process into tasks allows us to create templates that can be used to reduce development time and prevent bugs.

**Support**Breaking tasks up makes it much easier to debug and find the source of any breakdowns in the process.

# Glossary

Since we have learned a lot of new terms, let’s review some of the key terms.

**Archive**

An offline store of data that has been removed from tables.

**Business Key**

A natural key that can be used to uniquely identify a dimension record.

**Control Table**

A table that can be used to log the start and finish of ETL tasks. Can be used to help coordinate and audit ETL tasks.

**Destination Table**The tables in a data mart that an ETL process loads.

**ETL**

“Extract, Transform, Load”. A process that is used to load data into a data mart.

**ETL Cycle**

An execution of a set of ETL processes.

**ETL Tool**

A tool used to code and execute ETL processes. Common examples are SSIS, Oracle Data Integrator, Ab Initio, or Informatica. These processes make it easier to connect to a variety of sources, perform data transformations, and load data to a database.

**Extract Step**

The step of an ETL process that reads data from a source data set. Usually an extract process is built for each data set.

**Initialize Step**

The step of an ETL process that prepares the process and environment for execution.

**Integration**

The merging of multiple source data sets into a single repository. Transform steps will contain logic governing how to merge different sets of data, including which data sets to give preference to when conflicting data exists.

**Load Step**

The step of an ETL process that writes data into the target data mart, usually from a staging table. Usually a load step is built for each target table.

**Mapping**

A data transformation that converts values used in a source system to values used by the target data mart. This can help keep data homogenous, particularly when working with a variety of source systems.

**Non-key Attribute**

Any attribute of a dimension table that is not part of the business key or surrogate key.

**Purge**

A process that removes data from a data mart table when it is no longer required.

**Source Data Set**

A collection of data that is being provided to be loaded into a data mart. Source data sets may come in a variety of different forms, such as the results of a SQL query, or via an XML, JSON, or CSV file.

**Source Repository**

A system that stores data and may be the origin of one or more source data sets.

**Source System**

See “Source Repository”

**Stage**

A repository, most commonly a database, that is used to store transitional data being processed by various steps in an ETL process.

**Target Table**

See “Destination table”

**Transform Step**

The step of an ETL process that coverts source data sets into the format expected by the target tables. Transform steps are frequently the largest and most complex part of an ETL process, though that is not always the case. A transform step is usually created for each source-target combination, but multiple transform steps serving the same target may be combined, particularly in the case of complex integration logic.

**Validate Step**

A step of an ETL process that evaluates data for anomalies prior to loading. Multiple validation steps may exist at various points in an ETL process, and simple ETL processes may contain no validation at all.