ETL Techniques
ETL stands for Extract, Transform, and Load, and refers to the process of curating data from multiple sources, conforming it to a unified data format or structure, and loading the transformed data into its new environment.

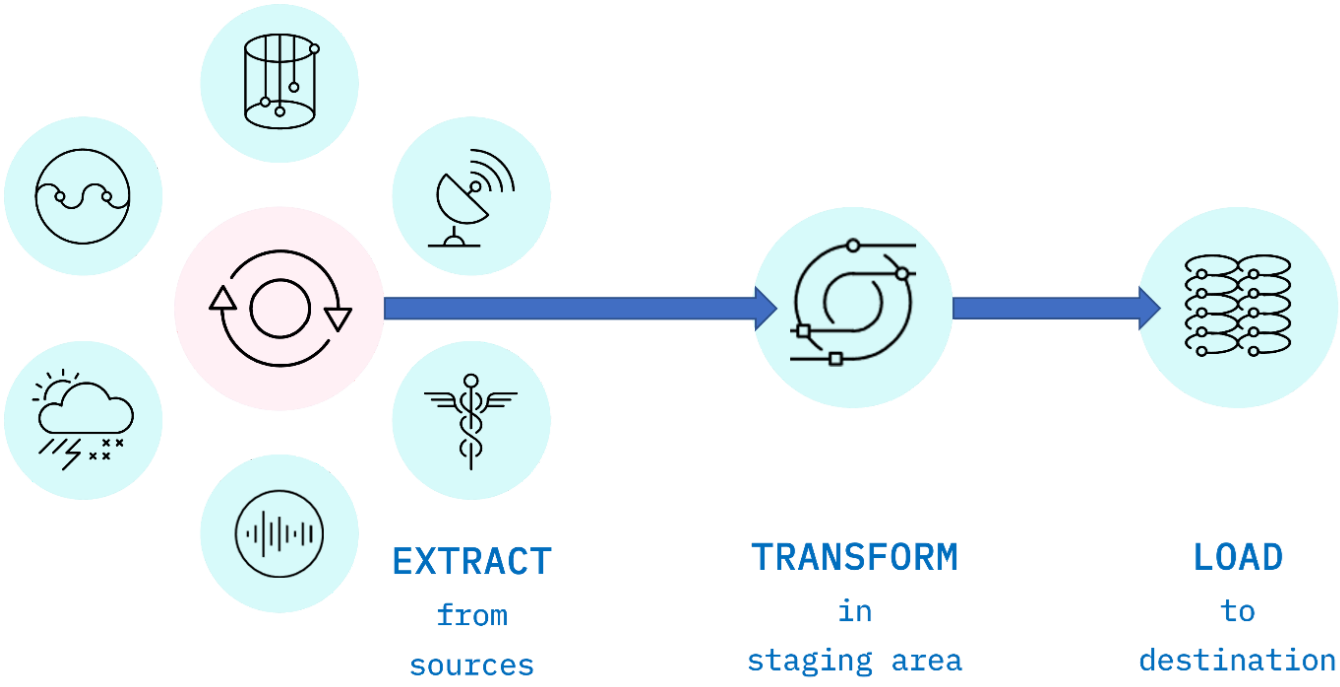

Fig. 1. ETL is an acronym used to describe the main processes behind a data pipeline design methodology that stands for Extract-Transform-Load. Data is extracted from disparate sources to an intermediate staging area where it is integrated and prepared for loading into a destination such as a data warehouse.

Extract

Data extraction is the first stage of the ETL process, where data is acquired from various source systems. The data may be completely raw, such as sensor data from IoT devices, or perhaps it is unstructured data from scanned medical documents or company emails. It may be streaming data coming from a social media network or near real-time stock market buy/sell transactions, or it may come from existing enterprise databases and data warehouses.

Transform

The transformation stage is where rules and processes are applied to the data to prepare it for loading into the target system. This is normally done in an intermediate working environment called a “staging area.” Here, the data are cleaned to ensure reliability and conformed to ensure compatibility with the target system.

Many other transformations may be applied, including:

Cleaning: fixing any errors or missing values

Filtering: selecting only what is needed

Joining: merging disparate data sources

Normalizing: converting data to common units

Data Structuring: converting one data format to another, such as JSON, XML, or CSV to database tables

Feature Engineering: creating KPIs for dashboards or machine learning

Anonymizing and Encrypting: ensuring privacy and security

Sorting: ordering the data to improve search performance

Aggregating: summarizing granular data

Load

The load phase is all about writing the transformed data to a target system. The system can be as simple as a comma-separated file, which is essentially just a table of data like an Excel spreadsheet. The target can also be a database, which may be part of a much more elaborate system, such as a data warehouse, a data mart, data lake, or some other unified, centralized data store forming the basis for analysis, modeling, and data-driven decision making by business analysts, managers, executives, data scientists, and users at all levels of the enterprise.

In most cases, as data is being loaded into a database, the constraints defined by its schema must be satisfied for the workflow to run successfully. The schema, a set of rules called integrity constraints, includes rules such as uniqueness, referential integrity, and mandatory fields. Thus such requirements imposed on the loading phase help ensure overall data quality.

ETL Workflows as Data Pipelines

Generally, an ETL workflow is a well thought out process that is carefully engineered to meet technical and end-user requirements.

Traditionally, the overall accuracy of the ETL workflow has been a more important requirement than speed, although efficiency is usually an important factor in minimizing resource costs. To boost efficiency, data is fed through a *data pipeline* in smaller packets (see Figure 2). While one packet is being extracted, an earlier packet is being transformed, and another is being loaded. In this way, data can keep moving through the workflow without interruption. Any remaining bottlenecks within the pipeline can often be handled by parallelizing slower tasks.

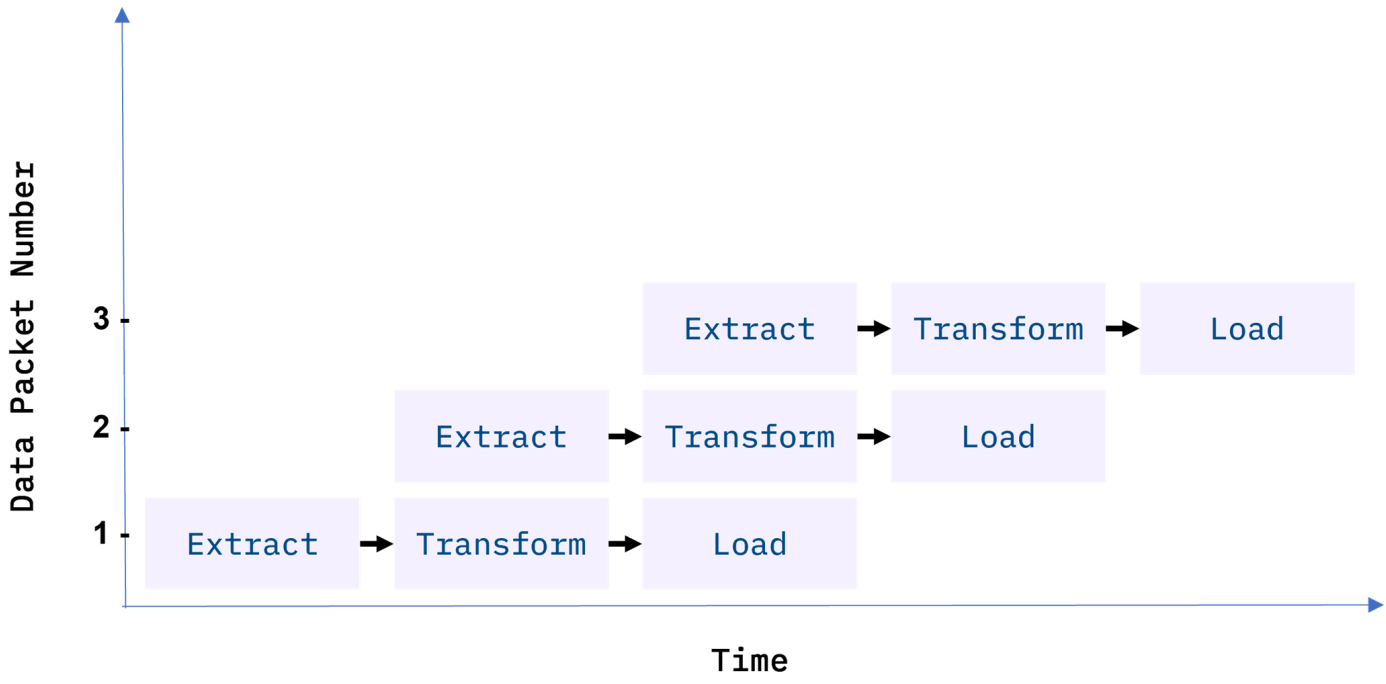

Fig 2. Data packets being fed in sequence, or "piped" through the ETL data pipeline. Ideally, by the time the third packet is ingested, all three ETL processes are running simultaneously on different packets.

With conventional ETL pipelines, data is processed in *batches*, usually on a repeating schedule that ranges from hours to days apart. For example, records accumulating in an Online Transaction Processing System (OLTP) can be moved as a daily batch process to one or more Online Analytics Processing (OLAP) systems where subsequent analysis of large volumes of historical data is carried out.

Batch processing intervals need not be periodic and can be triggered by events, such as

when the source data reaches a certain size, or

when an event of interest occurs and is detected by a system, such as an intruder alert, or

on-demand, with web apps such as music or video streaming services

Staging Areas

ETL pipelines are frequently used to integrate data from disparate and usually siloed systems within the enterprise. These systems can be from different vendors, locations, and divisions of the company, which can add significant operational complexity. As an example, (see Figure 3) a cost accounting OLAP system might retrieve data from distinct OLTP systems utilized by the separate payroll, sales, and purchasing departments.

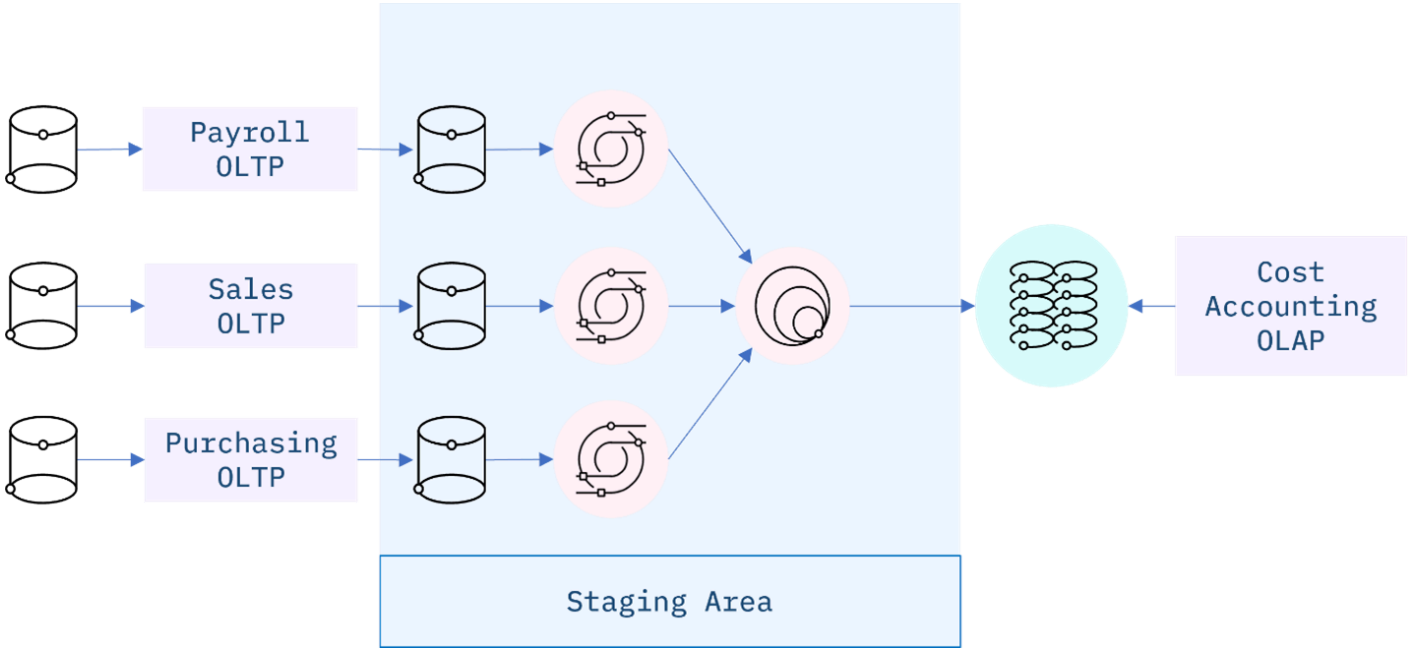

Fig 3. An ETL data integration pipeline concept for a Cost Accounting OLAP, fed by disparate OLTP systems within the enterprise. The staging area is used in this example to manage change detection of new or modified data from the source systems, data updates, and any transformations required to conform and integrate the data prior to loading to the OLAP.

ETL Workflows as DAGs

ETL workflows can involve considerable complexity. By breaking down the details of the workflow into individual tasks and dependencies between those tasks, one can gain better control over that complexity. Workflow orchestration tools such as Apache Airflow do just that.

Airflow represents your workflow as a directed acyclic graph (DAG). A simple example of an Airflow DAG is illustrated in Figure 4. Airflow tasks can be expressed using predefined templates, called operators. Popular operators include Bash operators, for running Bash code, and Python operators for running Python code, which makes them extremely versatile for deploying ETL pipelines and many other kinds of workflows into production.

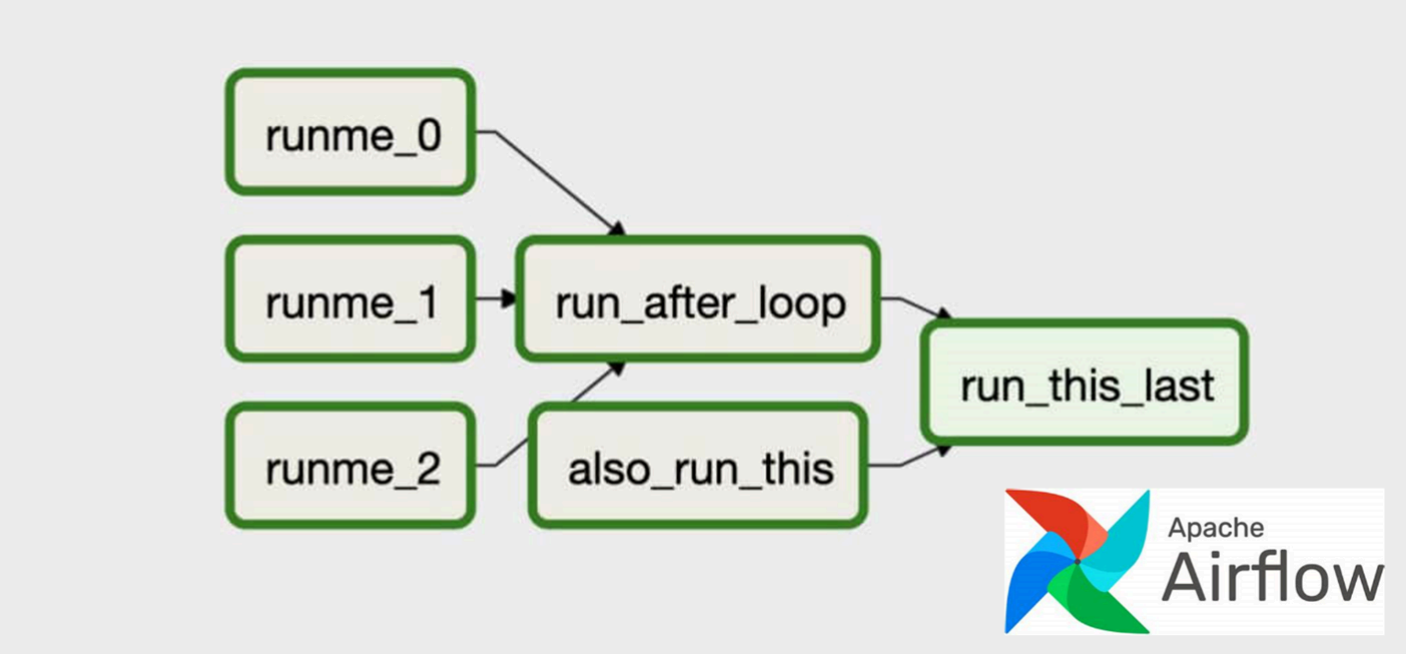

Fig 4. An Apache Airflow DAG representing a workflow. The green boxes represent individual tasks, while the arrows show dependencies between tasks. The three tasks on the left, 'runme_j' are jobs that run simultaneously along with the 'also_run_this' task. Once the 'runme_j' tasks complete, the 'run_after_loop' task starts. Finally, 'run_this_last' engages once all tasks have finished successfully.

Popular ETL tools

There are many ETL tools available today. Modern enterprise grade ETL tools will typically include the following features:

Automation: Fully automated pipelines

Ease of use: ETL rule recommendations

Drag-and-drop interface: "o-code" rules and data flows

Transformation support: Assistance with complex calculations

Security and Compliance: Data encryption and HIPAA, GDPR compliance

Some well-known ETL tools are listed below, along with some of their key features. Both commercial and open-source tools are included in the list.

Talend Open Studio

Supports big data, data warehousing, and profiling

Includes collaboration, monitoring, and scheduling

Drag-and-drop GUI for ETL pipeline creation

Automatically generates Java code

Integrates with many data warehouses

Open-source

AWS Glue

ETL service that simplifies data prep for analytics

Suggests schemas for storing your data

Create ETL jobs from the AWS Console

IBM InfoSphere DataStage

A data integration tool for designing, developing, and running ETL and ELT jobs

The data integration component of IBM InfoSphere Information Server

Drag-and-drop graphical interface

Uses parallel processing and enterprise connectivity in a highly scalable platform

Alteryx

Self-service data analytics platform

Drag-and-drop accessibility to ETL tools

No SQL or coding required to create pipelines

Apache Airflow and Python

Versatile “configuration” as code data pipeline platform

Open-sourced by Airbnb

Programmatically author, schedule, and monitor workflows

Scales to Big Data

Integrates with cloud platforms

The Pandas Python library

Versatile and popular open-source programming tool

Based on data frames – table-like structures

Great for ETL, data exploration, and prototyping

Doesn’t readily scale to Big Data