What's ETL?

The more acronyms you know...

Justin
Sep 14, 2020

21 2

The TL;DR

ETL stands for **Extract - Transform - Load**, and it's the process of moving data around your internal systems to get it ready for analysis.

- Data at most companies is **siloed**, **dirty**, and **hard to work with**: not ideal for analyzing it quickly and effectively
- ETL is the process of **moving**, **cleaning**, and **arranging** that data in a way that makes your Data Science team happy (well, as happy as they'll ever be)
- **Scheduling** is one of the most frustrating parts of ETL, and an ecosystem of tools has popped up to make it easier

You've probably heard that 70% of Data Science is just moving and cleaning data, and nobody gets too excited about that. But ETL powers analysis at pretty much every company out there, so it's worth understanding.

Background: the SDH problem

They say (well, I say) that all software is bad and complicated, but some is useful: the same thing is true about data. Most data is subject to what I call **SDH**: it's **s**iloed, **d**irty, and **h**ard to work with.

→ *Siloed*

Because of how applications are architected, **data tends to be stuck in the system that generates it**. If you're using Stripe for payments, your payment data is in the Stripe system. If you're using Salesforce for your CRM, your sales data is in the Salesforce system. And if you're using a PostgreSQL database to back your application, your user data is yet another system (yours).

The problem is that for analysis – both for operational dashboards and longer term Data Science™ – the value comes from **the integration of data**, or being able to work with your payments, sales, and user data all in one place. You need to get it out of these siloed systems and into a central location.

→ *Dirty*

Data is *dirty*. Things are spelled incorrectly, some payment amounts are USD and others CAD, some engineer changed your storage format a year ago without telling anyone...the list goes on. **You need to clean this data** before analyzing it, and ideally do that on a regular basis.

→ *Hard to work with*

Data has a *format*: it can be events data, user data, aggregated click counts, or any other form of captured reality. And the format that your data gets *generated* in is rarely the format you're going to *analyze* it in. You need to **transform it into something useful** by aggregating, pivoting, filtering, and all of that fancy stuff you learn how to do in basic Excel classes.

By now, you get it; we need to do a bunch of stuff to data before we can analyze it. That's exactly what ETL is – it's taking care of the SDH problem.

What's ETL?

ETL stands for Extract - Transform - Load:

- **Extract** data from where it lives
- **Transform** it into a useful format through cleaning and aggregation
- **Load** it into a warehouse for storage and analysis

Let's walk through each one of these so we can get a better idea of what they mean. Even though this *looks* like a 3 step process, it's often carried out via one big SQL or Python file, but we'll cover that in the next section.

→ *Extract*

You need to get your data out of Stripe and Salesforce (and whatever else you're using) so you can transform it and move it somewhere more useful.

→ *Transform*

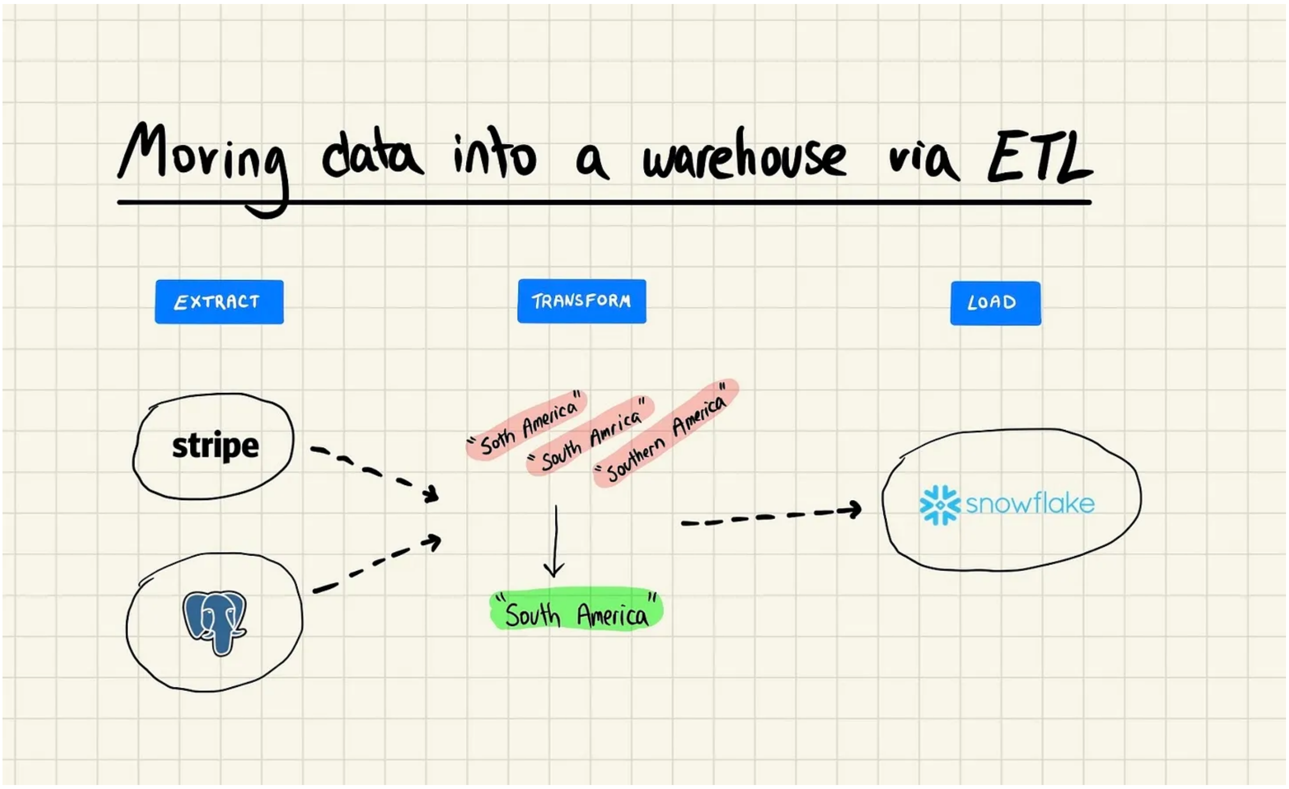
This is where you clean and transform your data into a useful format. Here's what a typical transform workflow might look like:

- Remove dollar signs from the "order_value" column (clean) →
- Fill in missing values in the "username" column with "anonymous" (clean) →
- Aggregate orders on a monthly basis (transform) →
- Join order data to user data to get shipping location (transform)

Typical ETL workflows can have tens or even hundreds of these kinds of transformations per dataset.

→ *Load*

After getting extracted and transformed, your data is pretty much ready to go; it just needs somewhere to hang out. Companies typically set up specialized storage called a **data warehouse** to store these newly ETL-ed datasets. Data warehouses like AWS's Redshift are optimized for big analysis instead of quick transactions. **Loading** is just the process of taking your transformed data and getting into the warehouse.



The **atomic unit of ETL** is usually called an **ETL job**: one set of extract, transform, and load operations aimed at creating a specific dataset. Sometimes, you'll also see people call them **data pipelines**.

> 🚨 **Confusion Alert** 🚨
>
> It took me years to really understand what ETL meant; the big aha! moment was when I finally got to build one myself at work. This post should hopefully help; but if you're stuck, try seeing if you can ask someone on the engineering or data team at work for a real life example.
>
> 🚨 **Confusion Alert** 🚨

ETL in practice: tools and scheduling

ETL isn't always some giant, scary application: it can just be a single little Python file scheduled to run every day. Generally, the programming languages and frameworks that you use to build ETL jobs depend on what systems your data sits in. For working with the Stripe API, you might use Python, and for working with Salesforce data you might use SQL.

One of the hardest parts of ETL is **scheduling**. Since your systems are generating new data all of the time, you need to constantly extract, transform, and load it. In theory you'd want to do this every single time a new data point is generated (e.g. a new order), but that would be kind of ridiculous. Instead, ETL operations are typically **batched into a daily or hourly job**, depending on your use case.

It turns out that getting your code to run consistently at the same time every day is actually *not* that simple. Most popular operating systems have a utility called **Cron** pre-installed that lets you run programs at scheduled times, but it doesn't always work. Scheduling is hard:

- Things *always* go wrong in ETL jobs, and **handling those errors** is a pain
- ETL jobs typically have **dependencies**: you need to wait for one operation to complete in order to start a second one

As companies have been working with more and more data over the past few years, new and exciting tools are being developed (as we speak) to make this process easier. The most popular one outside of Cron is called Airflow: it was originally built by AirBnB, and then open sourced and incubated inside of Apache. Airflow is a purpose built tool for ETL, and ships with granular scheduling, detailed logs for error handling, and special tools for setting and managing dependencies.

> 🔗 **Workplace Example** 🔗
>
> At DigitalOcean, we used Airflow to run and manage some of our ETL pipelines. One example was event data for our managed databases product. I built an ETL job that extracted that event data from its source (a Kafka stream), transformed it into a more useful format of one row per cluster, and loaded it into our warehouse.
>
> 🔗 **Workplace Example** 🔗

"ETL" in conversation

> *"Just write the SQL you want and we can get it scheduled"*

Write the SQL that you need to get the data and transform it, and we'll make sure it runs when you need it to.

> *"The dashboard is still broken because of this messed up ETL job"*

The dashboard is showing incorrect or stale data because the ETL job powering it (creating the dataset to visualize) isn't working properly.

Terms and concepts covered

`SDH`

`ETL`

`ETL job`

`Scheduling`

`Dependencies`

`Cron`

`Airflow`

Further reading

- As cloud storage continues to get cheaper and cheaper, some companies are reversing the ETL order and doing ELT instead: loading data into the warehouse and *then* transforming it
- Airflow is popular, but not everyone loves using it. New scheduling tools like Prefect, Astronomer (built on Airflow), and Dataform are making things

simpler

2 Comments

Write a comment...

Sidhant Goyal Writes Sidhant's Newsletter Nov 17, 2020

Hey Justin, would love an independent post on dbt in your inimitable style - understand the warehouse and EL in/out (Fivetran/Census) parts of the BI stack but don't fully get the 'T' in ELT (dbt/Airflow) super well.

1 Reply Gift a subscription Collapse •••

1 reply by Justin

1 more comment...