The major stages of data engineering lifecycle into five stages -

1. Generation

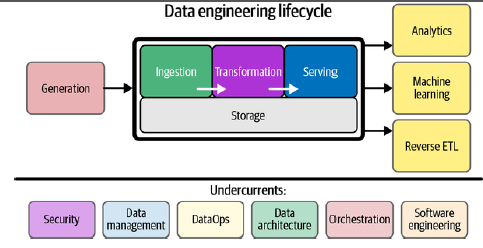
2. Storage

3. Ingestion

4. Transformation

5. Serving data

Figure. Components and undercurrents of the data engineering lifecycle



The data engineer’s job is taking raw data input in the source system schema and transforming this into valuable output for analytics.

1. Generation: Source Systems -

Source system could be an IOT device, an application message queue, or a transactional database.

A data engineer consumes data from a source system but doesn’t typically own or control the source system itself also needs to have a working understanding of the way source systems work, the way they generate data, the frequency and velocity of the data, and the variety of data they generate.

Sources produce data consumed by downstream systems, including human-generated spreadsheets, IOT sensors, and web and mobile applications. Each source has its unique volume and cadence of data generation. A data engineer should know how the source generates data, including relevant quirks or nuances (a peculiarity of action, behaviour or difference or distinction in expression, meaning, response, etc.).

One of the most challenging nuances of source data is the schema.

Two popular options are schemaless and fixed schema.

Schemaless doesn’t mean the absence of schema. Rather, it means that the application defines the schema as data is written, whether to a message queue, a flat file, a blob, or a document database such as MongoDB.

A more traditional model built on relational database storage uses a fixed schema enforced in the database, to which application writes must confirm.

2. Storage

Storage runs across the entire data engineering lifecycle, often occurringin multiple places in a data pipeline, with storage systems crossing overwith source systems, ingestion, transformation, and serving.

For example, cloud data warehouses canstore data, process data in pipelines, and serve it to analysts. Streaming frameworks such as Apache Kafka and Pulsar can function simultaneous-ly as ingestion, storage, and query systems for messages, with object stor-age being a standard layer for data transmission.

Not all data is accessed in the same way.

Data Retrieval patterns will greatly vary based on the data being stored and queried.

This brings up the notion of the “temperatures” of data. Data access frequency will determine the temperature of your data. Data that is most frequently accessed is called hot data.

For example, in systems that serve user requests. This data should be stored for fast retrieval, where “fast” is relative to the use case. Lukewarm data might be accessed every so often—say, every week or month.

Cold data is seldom queried and is appropriate for storing in an archival system. Cold data is often retained for compliance purposes or in case of a catastrophic failure in another system.

Type of storage solution is depends on your use cases, data volumes, frequency of ingestion, format, and size of the data being ingested.

3. Ingestion

Source systems and ingestion represent the most significant bottlenecks of the data engineering lifecycle.

Two major data ingestion concepts: batch versus streaming and push versus pull.

Batch versus streaming -

Batch ingestion is simply a specialized and convenient way of processing this stream inlarge chunks—for example, handling a full day’s worth of data in a single batch.

Streaming ingestion allows us to provide data to downstream systems—whether other applications, databases, or analytics systems—in a continuous, real-time fashion.

Here, real-time (or near real-time) means that the data is available to a downstream system a short time after it is produced (e.g., less than one second later).

Batch ingestion is a one-way door: once data is broken into batches, the latency for downstream consumers is in-herently constrained.

Batch processing remains an extremely popular way to ingest data for downstream consumption, particularly in analytics and ML.

Push versus pull -

Push model of data ingestion, a source system writes data out to a target, whether a database, object store, or filesystem.

In the pull model, data is retrieved from the source system.

The line between the push and pull paradigms can be quite blurry; data is often pushed and pulled as it works its way through the various stages of a data pipeline.

4. Transformation

Without proper transformations, data will sit inert, and not be in a useful form for reports, analysis, or ML.

Typically, the transformation stage is where data begins to create value for down-stream user consumption.

Immediately after ingestion, basic transformations map data into correcttypes (changing ingested string data into numeric and date types, for ex-ample), putting records into standard formats, and removing bad ones.

Later stages of transformation may transform the data schema and apply normalization.

Downstream, we can apply large-scale aggregation for re-porting or featurize data for ML processes.

Data preparation, data wrangling, and cleaning—these transformative tasks add value forend consumers of data.

5. Serving Data

Data serving is perhaps the most exciting part of the data engineering lifecycle.

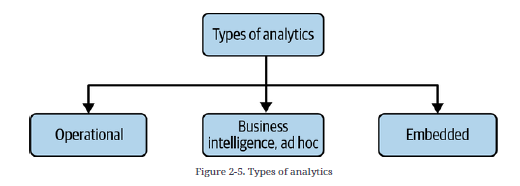
This is where the magic happens. This is where ML engineers can apply the most advanced techniques.

Let’s look at some of the popular uses of data: analytics, ML, and reverse ETL.

Analytics –

The bulk of analytics used to encompass BI

Figure. Types of analytics



Business intelligence -

BI marshals collected data to describe a business’s past and current state.

BI requires using business logic to process raw data. Note that data serving for analytics is yet another area where the stages of the data engineering lifecycle can get tangled.

Operational analytics -

Operational analytics could be a live view of inventory or real-time dash- boarding of website or application health.

Embedded analytics -

With embedded analytics, the request rate for reports, and the corresponding burden on analytics systems, goes up dramatically; access control is significantly more complicated and critical.

Machine Learning-

The responsibilities of data engineers overlap significantly in analytic sand ML, and the boundaries between data engineering, ML engineering, and analytics engineering can be fuzzy. For example, a data engineer may need to support Spark clusters that facilitate analytics pipelines and ML model training. They may also need to provide a system that orchestrates tasks across teams and support metadata and cataloging systems that track data history and lineage.

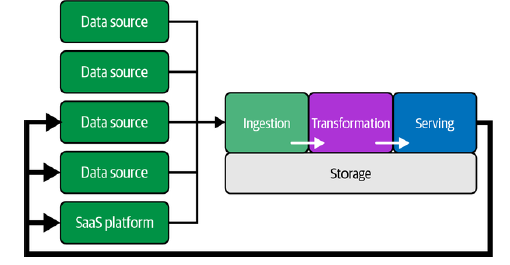
The feature store is a recently developed tool that combines data engineering and ML engineering. Feature stores are designed to reduce the operational burden for ML engineers by maintaining feature history and versions, supporting feature sharing among teams, and providing basic operational and orchestration capabilities, such as backfilling.

Reverse ETL -

Reverse ETL takes processed data from the output side of the data engineering lifecycle and feeds it back into source systems.

Reverse ETL allows us to take analytics, scored models, etc., and feed these back into production systems or SaaS platforms.

Figure. Reverse ETL



Reverse ETL has become especially important as businesses rely increasingly on SaaS and external platforms. For example, companies may want to push specific metrics from their data warehouse to a customer data platform or CRM system. Advertising platforms are another everyday use case, as in the Google Ads example. Expect to see more activity in reverse ETL, with an overlap in both data engineering and ML engineering.