**WEEK-1: ETL and ETL Process:**

1. **ETL Fundamentals:**

* What is ETL process?

ETL: (Extract Transform Load)

Automated data pipeline engineering methodology where data is acquired and prepared for subsequent use and used in analytics environments such as data warehouse and data mart.

Process:

Step-1: Extracting data from multiple sources

Step-2: Transform data to a unified data format

Step-3: Loading transformed data into new environment.

Extraction:

Configuring access to data and reading it into an application:

* Web scraping
* Connecting programmatically via APIs

The data may be static or streaming online.

Transformation:

* Processing data
* Conforming to target systems and use cases.
* Cleaning
* Filtering
* Joining
* Feature Engineering
* Formatting and Data typing

Loading:

Moving data into a new environment (EG: DB, DWH, data mart)

Making the data readily available for analytics, dashboards, reports

* Use cases for ETL pipelines:

Digitizing analog media A grey rectangular sign with white text

Description automatically generated

Moving data from OLTP systems to OLAP systems

Dashboards

1. **ELT Basics:**

ELT Process: (Extract, Load, Transform)

Similar stages involved as ETL.

The order of performing tasks is different.

Step-1: Extracting data from sources asynchronously.

Step-2: Loading data as-is into destination system.

Step-3: Transforming data on demand.

Use Cases:

Demanding scalability requirements of Big data

Streaming analytics

Integration of highly distributed data sources

Multiple data products from the same sources

Why is ELT an emerging trend?

Big Data 🡪 Cloud computing

ELT separated the data pipeline from the processing i.e., it separates moving data from processing data.

Has more flexibility

No information loss as transformation is only performed only on required data.

1. **ETL vs ELT**

Differences between ETL and ELT:

|  |  |
| --- | --- |
| **ETL** | **ELT** |
| Transformations happen within the data pipeline. | Transformations happen in the destination environment. |
| ETL is rigid – pipelines are engineered to user specifications. | ELT is flexible – end users build their own transformations. |
| ETL can handle structured relational data, on-premises data. Hence, scalability is difficult. | ELT can handle both structured and unstructured big data in cloud. Hence, it solves scalability problems. |
| ETL workflows take time to specify and develop which means users must wait for the development team to implement the requested changes. | ELT supports self-serve, interactive analytics in real time which means in some applications, end users can easily connect to and experiment with the raw data, create their own dashboards and run their productive models themselves. |

The evolution of ETL to ELT:

Increasing demand for access to raw data is one of the primary causes for this.

ETL has intermediate storage facility called as **staging areas (**Holds raw extracted data or you can run processes prior to loading resultant transformed data into a DWH or a Data mart**)**.

The staging area fits the description of a data lake.

Saging areas 🡪 Private ETL landing zones and are not shared across company. It is a private area mainly used for developing, monitoring and tuning data pipelines and is built in data pipelines.

Self-serve data platforms are a new staging area.

ETL still has its place for many applications. Pain points for ETL to ELT:

Lengthy time-to-insight

Challenges imposed by Big Data

Demand for access to Siloed information.

1. **Data Extraction Techniques:**

Examples of raw data sources:

Paper documents Web pages Analog audio/video

Survey, Statistics, Economics data Transactional data

Social media IoT data Weather station N/w

Medical records Human Genomes

Techniques for extracting data:

There are many techniques available for extracting data depending on the data source and intended use of data.

1. OCR (Optical Character Recognition)

Used to interpret and digitize text scanned from paper documents so it can be stored as a computer-readable file.

1. ADC (Analog-to-Digital Converters)

They can digitize analog audio recordings and signals and CCDs (Charge-Coupled Devices) that can capture and digitize images.

1. Opinions, Questionnaires, and vital statistical data obtained through polling and censes methods.
2. Cookies, User logs and other methods used for tracking human or system behavior.
3. More techniques include:
   * + Web Scraping (used to crawl web pages in search of text, images, tables, and hyperlinks)
     + APIs (readily available for extracting data from all sorts of online data repositories and feeds, such as government bureaus of statistics, libraries, weather networks, online shopping, and social networks)
     + SQL languages for querying relational databases, and NoSQL for querying documents, key-value, graph, or other non-structured data repositories.
     + Edge computing devices such as video cameras that have built in processing that can extract features from raw data.
     + Biomedical devices (like microfluidic arrays that can extract DNA sequences).

Use Cases:

* Integrating disparate structured data sources via APIs.
* Capturing events via APIs and recording them in history.
* Monitoring or surveillance with edge computing devices.
* Data migration (direct to storage) for further processing.
* Diagnosing health problems with medical devices.

1. **Data Transformation Techniques:**

It is mainly about transforming the data into the data that suits the application.

Operations involved can be:

* Data typing
* Data structuring
* Anonymizing, encrypting
* Cleaning: duplicate records, missing values
* Normalizing: converting data to common units
* Filtering, sorting, aggregating, binning for accessing the right data at a suitable level of detail and in a sensible order.
* Joining data sources.

Schema-on-write Vs Schema-on-read:

|  |  |
| --- | --- |
| **Schema-on-write** | **Schema-on-read** |
| It is conventional ETL approach where data must be confirmed to define schema prior to loading to a destination. | It relates to modern ELT approach where schema is applied to the raw data after reading it from the raw data storage. |
| The idea is to have:  Consistency, Stability of data and to make querying much faster.  But, it comes with limited versatility of data. | This approach is versatile as it can obtain multiple views of data using adhoc schemas. Users potentially have access to more data (Enhanced storage flexibility) since it does not need to go through too much of pre processing step. |

Information loss in Transformation:

There can be many causes:

Raw data is usually much bigger than transformed data and contains noise and redundancy.

Hence information is a proper subset of data.

Shrinking data volume => shrinking information volume.

ETL -> information lost is not recoverable.

ELT -> information lost is recoverable.

Examples of information loss in transformation:

* Lossy data compression (converting float to int: the data after decimal point is lost)
* Reducing bit rates in audio or video.
* Filtering (It is usually temporary selection of data. But if it is a permanent selection, it can easily be discarded)
* Aggregation (avg yearly sale vs avg monthly sales)
* Edge computing devices (false negative signals designed only to signal alerts and not raw data)

1. **Data Loading Techniques:**

There are many data loading techniques:

* Full load:

You can load an initial history into a database after which you can use incremental load.

Involves loading data in one large batch.

This is used for examples:

When organizations want to start tracking transactions in a new data warehouse. They copy the existing transactions history from the old to the new system. Then incremental loading technique can be used.

* Incremental load:

To upload new data or already loaded data.

In incremental data loading, the target data store is appended to such that only the changes are loaded and they are basically not overwritten.

This is useful for accumulating historical data such as transactions, weather, and browsing history.

* Scheduled load and On-demand load:

Data loading processes can be scheduled or initiated on demand. Data is often loaded on a schedule.

You can schedule data loading on a periodic basis.

For example: Daily point-of-sale transactions can be loaded into a database at the end of each day, during off-peak hours. Loading tasks can be scheduled with tools such as Windows Task Scheduler, pr with cron on Unix-like systems.

You can load the data as required on demand.

On-demand loading is very common, and relies on triggering mechanisms such as: when the source data reaches a specified size. When an event is detected by the source system, such as motion, sounds or temperature changes, when a user requests data, such as online videos, music, or web pages.

* Batch and Stream data loading:

Batch and stream data loading are two ends of a spectrum of loading methods.

The volume velocity, and demand for the data determine whether the data is loaded in batches or streamed live.

Batch loading refers to loading data in chunks defined by some time windows of data accumulated by the data source, usually on the order of hours to days.

At the other end of the spectrum we have stream loading, which loads data in real time as it becomes available.

In between batch and stream loading, we have micro-batch loading. This is used when imminent processes need access to a small window of recent data.

* Push and Pull:

Data can be pushed to a server (or) pushed to clients by a server.

Push and pull data loading methods are based on client-server model.

A ‘pull’ refers to a client initiating a request for data from a server. The server then responds to the client’s request and delivers the data. (Eg: RSS feeds and email).

With ‘push’ technology, the client subscribes to a service provided by a server, so that the server can then push data to the client as it becomes available. (Eg: push notifications and instant messaging services).

* Parallel and Serial load:

Parallel loading can be employed on multiple data streams to boost loading efficiency, particularly when the data is big or has to travel long distances.

A white background with black lines

Description automatically generated

Similarly, by splitting a single file into smaller chunks, the chunks can be loaded simultaneously.

A white background with black lines

Description automatically generated